Cisco CCNA (and CCDA) Study Guide

By Dan DiNicolo, January 25th, 2007 Posted in **General.** http://www.2000trainers.com/

Dan's Free CCNA Study Guide is a collection of tutorials designed to help students preparing for Cisco's CCNA exam. These tutorials are not a brain dump, a series of cram sheets, or anything of the sort. Quite to the contrary, they are a series of articles designed to help students learn many of the core concepts that they will need to be familiar with both for the exam and in their real-life interactions with Cisco switches and routers.

Will reading through every tutorial in this guide guarantee you a passing score on the CCNA exam(s)? Of course not! The tutorials in this guide are meant to help you learn and understand. If they help you to pass the exam, that's great. You should note, however, that this guide is not updated to reflect every change that Cisco makes to its exams in terms of topics and depth of knowledge required. Things like that can, do, and almost certainly will change.

The links below will help you to navigate through the chapters and individual tutorials that make up this study guide.

Reader Hint: Because tutorials appear in reverse chronological order in the main body of the chapter index pages, use the "All Chapter XX Tutorials" list in the right-hand sidebar as your navigation guide. Additionally, you'll find a link at the bottom of every individual article that will bring you to the next (or previous) article in that particular chapter. When you're all done reading, test your knowledge with our **free CCNA Practice Exams (SEE FINAL PAGE FOR LINKS).**

Chapter 1: Understanding Network Models

Dan's Free Cisco CCNA/CCDA Study Guide

http://www.2000trainers.com/

Understanding Network Models: Introduction and Modular Design

Posted: August 25, 2005

By: Dan DiNicolo

The following is part of a complete CCNA/CCDA study guide by Dan DiNicolo, posted on the 2000Trainers.com web site. Each chapter of the book is presented as a series of articles, and you will find a complete TOC for a given chapter at the bottom of each article once the entire chapter is posted. In the meantime, a complete list of the articles posted in the series can always be found here.

A new chapter will be posted as a series of articles each week (approximately), until the entire study guide has been posted. The materials in the text are applicable to both Cisco's CCNA and CCDA certification exams. In cases where materials apply to one exam only, a note to that effect will be posted within each article.

When preparing for your CCNA or CCDA, concepts that relate to network models will appear again and again. While these might not seem like the most fascinating topics, I can guarantee a solid understanding of the models and how they relate to network communication and design is critical – both on the exams and in real life. Models exist to help illustrate concepts. A fundamental understanding of their goals and responsibilities will ultimately lead to a better appreciation of why networks work the way they do.

In this chapter we'll take a look at three different network models, relating them to network communication and design. These include:

- **The Open Systems Interconnect (OSI) Model**. Developed by the International Organization for Standardization (ISO), this model is the foundation upon which most network communication protocols are designed.
- **The TCP/IP model**. The TCP/IP protocol suite is a culmination of the work of many different companies, individuals, and organizations that took place over many years. While the protocols that make up the suite can be loosely mapped to the OSI model, they are more commonly referenced using the 4-layer TCP/IP model.
- **The Cisco hierarchical network design model**. This model is concerned with the design of networks to meet performance, security, capacity and scalability requirements.

The key to appreciating the various models is in relating them to real-life networking concepts and scenarios. Be sure to take the time to understand the functions of each model's layers, as well as the protocols and equipment found at each.

Modular Design

One thing you will quickly notice about network models is that they tend to follow a layered design. I personally prefer looking at them as being modular, because it better conveys the idea of separate and changeable elements. The reason for the distinct sections within various models is simple – think of it as "a place for everything, and everything in its place". Consider how roles and responsibilities often get separated in real life. At a restaurant, one person may cook your dinner, another serves it, and yet another washes the dishes when you're done. In this way, any one element can be replaced or altered without having a huge impact on the others. Network models work in much the same way, except that each layer has different responsibilities with respect to network communication.

The main reasons for following a layered or modular design include:

- **The separation of functions**. For example, one layer might be worried about communication reliability, while another is concerned with the technical details of data transmission.
- **Ability to make changes easily**. If changes need to be made to a given layer, these can usually be isolated, not requiring a redesign of other layers.
- **Simplification**. By dividing roles and responsibilities into different layers, the complexity of networking can be broken down into more manageable sections. This also makes network communication an easier subject to teach and learn.
- **Standardization**. If a layered model is an industry standard, vendors can use the model as a blueprint to design systems capable of interoperating.

The Open Systems Interconnect (OSI) Model

The most popular network communication model by far is the 7-layer Open Systems Interconnect (OSI) model, designed by the ISO in the 1970's. While today we might take for granted the ability to communicate between different systems, this was not always easy or possible. In the golden days of computing, equipment from IBM couldn't talk to that of Digital, making interoperability difficult to impossible. The goal of the ISO was to create a reference model that would clearly define network functions and responsibilities, ultimately allowing different systems to interconnect and communicate. It is worth noting that the OSI is primarily a reference model, although an actual implementation of an OSI protocol suite does exist. In reality, most network protocol stacks tend to map to the model roughly, but seldom exactly.

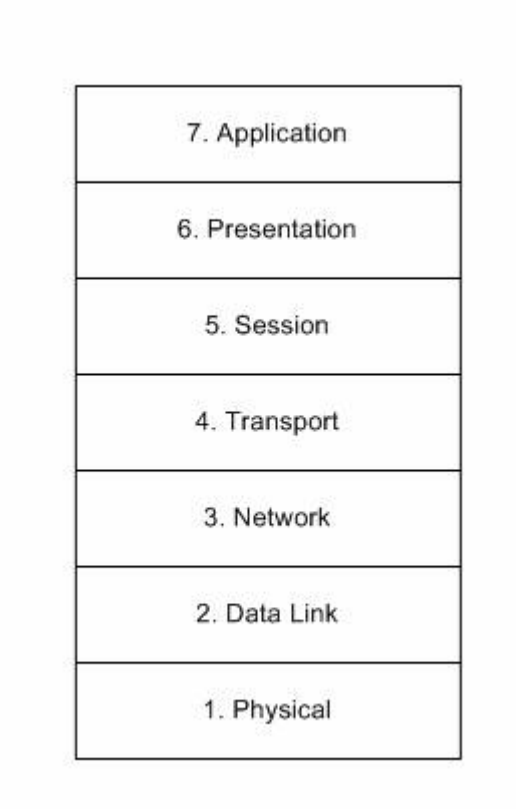


Figure 1-1: The OSI Model and associated layer numbers.

Each of the seven layers is known not only by a name, but also by a number. To that end, you'll often hear the Network layer referred to as Layer 3, or the Transport layer as Layer 4. Remembering the layer names and numbers is critical. Many people get these layers confused in the beginning, so consider using the first letters of each layer to create a mental reminder. Of these, the most popular is probably "All People Seem To Need Data Processing". If that doesn't work for you, make up one that will help you remember the order of the layers easily.

EXAM TIP: To remember the names and order of the layers in the OSI model, use the first letter of each layer to create a mental reminder (mnemonic), such as **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing.

Network Peers

A critical concept to understand when looking at network communication models is the idea of peer layer communication. Peer layer communication is a way of defining how the different layers in the OSI model interact with one another when systems communicate. On a single system, each layer has one or two neighboring layers - the layer above it, and the layer below it. For example, the Network layer will interact with both the layer above (Transport) and the layer below (Data Link). When preparing to send data over the network, the Network layer receives the data from the Transport layer, makes some additions, and then passes it down to the Data Link layer where it is formatted further.

When we extend the model over the network to include another system, you'll need to consider what I call horizontal peering. In network communication models, any given

layer communicates only with that same layer on another system. So, when the Data Link layer adds some information to the data prior to passing it over the network, this added information will be of use only to the Data Link layer (its peer) on the receiving system. Similarly, the information added by the sender at the Transport layer will only be of use to the Transport layer on the receiving system.

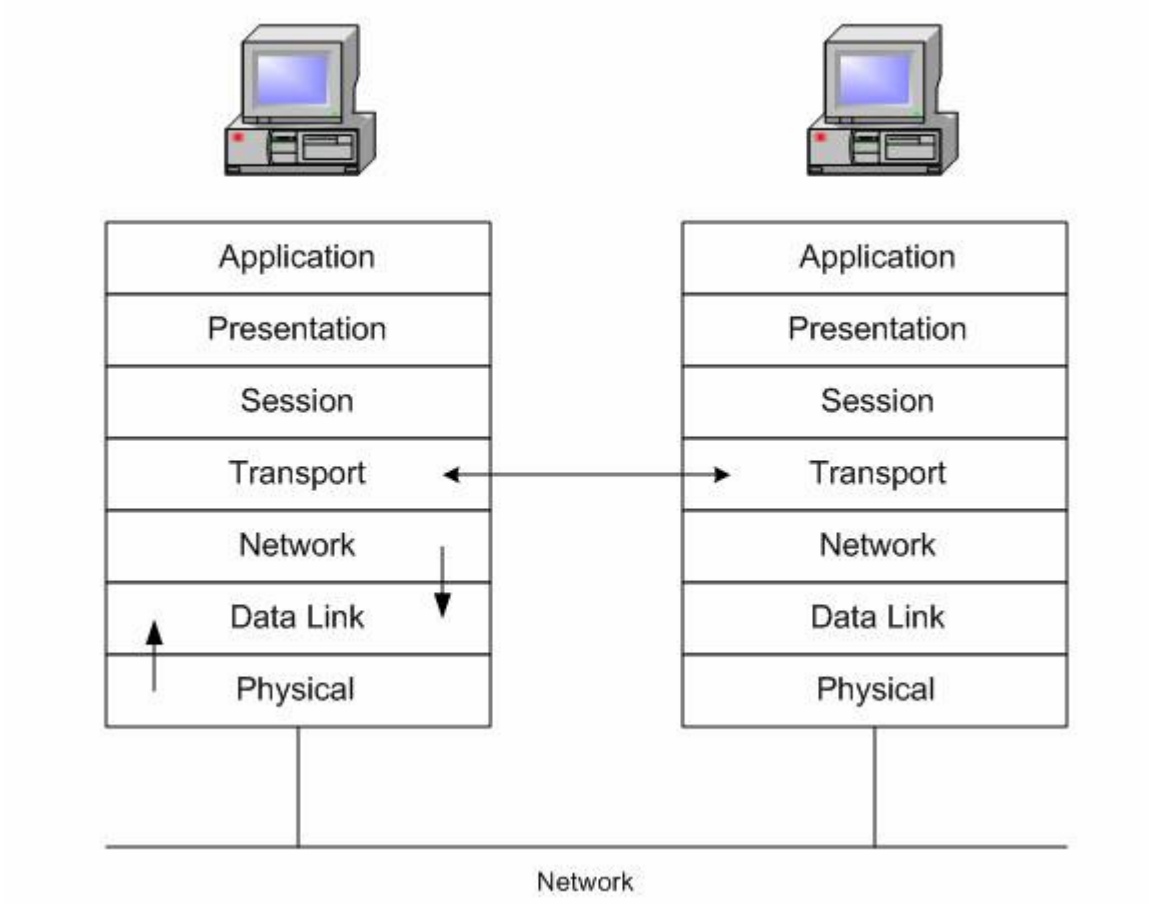


Figure 1-2: Peer layer communication

This concept may seem a little confusing at first, but think of it like this. Ultimately, a packet is going to be created that will be sent over the network and received by another system. It starts with data created at the Application layer, and layers add information (the whole process being referred to as encapsulation) as the data travels down the OSI model. The packet is then transmitted across the network. Once it arrives at the receiving system, parts that were added are now stripped away in reverse order at each layer. Remember that each layer on the sending system provides information that is used by the same layer on receiving system. If it all seems a little theoretical at the moment, do not worry – we'll ultimately apply this to how a real TCP/IP packet is created.

Protocol Data Units

You are probably already familiar with the term 'packet'. It is often used to generically describe data that will be passed between systems over a network. As data flows down the layers of the OSI model, the various layers encapsulate it, usually by adding header (and possibly trailer)

information. At each layer where this happens, the term used to describe the data is different, and it should be properly referred to by its Protocol Data Unit (PDU) name. Table 1-1 outlines the names used to describe Protocol Data Units at different OSI layers.

Table 1-1: OSI Layers and Protocol Data Units (PDUs).

OSI Layer	Data Unit
Application	Data
Presentation	
Session	
Transport	Segment
Network	Packet
Data Link	Frame
Physical	Bits

Notice that once passed to the Transport layer, data is properly referred to as a segment.

Application Layer

The Application layer is the top layer of the OSI model, and is considered to be the place where the user interacts with the network. This interaction usually occurs by running a program, such as a web browser or a spreadsheet application. This layer doesn't really worry about the network. Instead, it simply knows how to make a request for something (this is the data), and then what to do with the reply (the data that was requested). In the case of a user browsing a website, the client application (the web browser) makes an HTTP request that will be understood by the receiving application at the other end (the web server). This is sometimes referred to as program-to-program communication. The lower layers of the model concern themselves with how this data is actually encapsulated and transmitted.

Table 1-2: Examples of common Application layer programs and services.

Application	Purpose
Word Processor	Creating documents, possibly to be saved to a network server.
Web Browser (HTTP)	Access to internet web services
Email Client (SMTP, POP3, IMAP)	Sending and receiving email
Telnet	Remote terminal session
File Transfer (FTP)	File transfer

TIP: Take the time to familiarize yourself with the various protocols and applications that exist at the different OSI layers.

Presentation Layer

The Presentation layer is primarily responsible for data representation and formatting, ensuring that data can be viewed correctly. These formats are sometimes referred to as the "data syntax" of the applications in use. For example, different systems may use different schemes to represent data. While one system might use ASCII or EBCIDC, another might use UNICODE. Since these schemes contain different character possibilities, it is the responsibility of the Presentation layer to make sure they are displayed in the correct or common format between the client and the server. Further to this, the Presentation layer is also where data compression and encryption are generally considered to take place.

Table 1-3: Examples of common Presentation layer formats.

Data Formats	Purpose
ASCII, EBCIDC, UNICODE, RTF	Text encoding formats
MPEG, AVI, QuickTime	Video encoding formats
JPEG, PNG, TIFF	Graphics formats
MIDI	Sound format

Session Layer

The Session layer is responsible for the creation, management, and termination of sessions between systems. A session is best described as a type of managed connection between systems for the purpose of a specific type of communication. For example, a session might be created for the purpose of user authentication, or to initiate a file transfer.

The Session layer is also responsible for coordinating how the communication between systems takes place, which is known as dialog control. In some sessions, only a single system is allowed to communicate at any point in time, referred to as half-duplex. The Session layer would be responsible for determining whose turn it is in these situations, and for how long each system is allowed to communicate. In other cases, both systems can communicate at once, which is also known as full duplex. If the communication stream were somehow interrupted, the Session layer would be responsible for recognizing this and re-establishing the session.

Table 1-4: Examples of Session layer protocols.

Protocol	Purpose
Network File System (NFS)	Unix file system access
Structured Query Language (SQL)	Local or remote database queries
Remote Procedure Call (RPC)	Client-server communication mechanism
AppleTalk Session Protocol (ASP)	AppleTalk client-server communication mechanism
X Windows	Remote desktop sessions

EXAM TIP: Remember that the protocol data unit (PDU) of the Application, Presentation, and Session layers is “data”.

Transport Layer

The Transport layer has three main responsibilities in terms of the exchange of data between systems. These include:

- Data segmentation.
- Establishment of end-to-end connections between hosts.
- Using flow-control mechanisms to ensure that data is sent at rates that the receiver can handle.

Data Segmentation

At any given point in time there may be many applications passing data down to the Transport layer. Data segmentation is the process by which the Transport layer uniquely handles all data passed to and from different upper-level applications. This is usually implemented in the form of source and destination port numbers that are defined within a particular application. For example, if a user is browsing the web and checking email at the same time, each program would be passing data and waiting for a reply on a unique port number. The Transport layer ensures that data is passed to the correct application.

Connection-Oriented Sessions

When a connection-oriented session is established between systems, acknowledgements are returned to the sender as proof that segments reached their destination. If an acknowledgement is not received, the associated data will be resent. This system is known as positive acknowledgement with retransmission. As such, connection-oriented communications are often referred to as being reliable. Connection-oriented sessions also make use of sequence numbers, a system by which packets are numbered to be sure that they are reassembled in the correct order at the receiver. The upside of these techniques is that they provide confirmation that the segments actually arrived, and are properly sequenced. The tradeoff is the overhead involved, in terms of the relatively slower communication that takes place due to these reliability mechanisms.

There are three main phases to a connection-oriented session. These include:

- **Call Setup**. When a connection is being established, a path known as a virtual circuit is created between the sender and receiver.
- **Data Transfer**. Once the path is created, data is transmitted sequentially to the receiver.
- **Call Termination**. When an established connection is no longer required, the virtual circuit is terminated.

For connection-oriented transport protocols, sessions between two systems are usually initiated using what is often referred to as a three-way handshake. In this process, a client first contacts a server requesting a session. The server then replies with an acknowledgement and its session parameters (for example, how much data it can receive at once). Finally, the client completes the session initiation process with its own acknowledgement. After data has been transferred and a connection-oriented session is complete, a similar process closes the session.

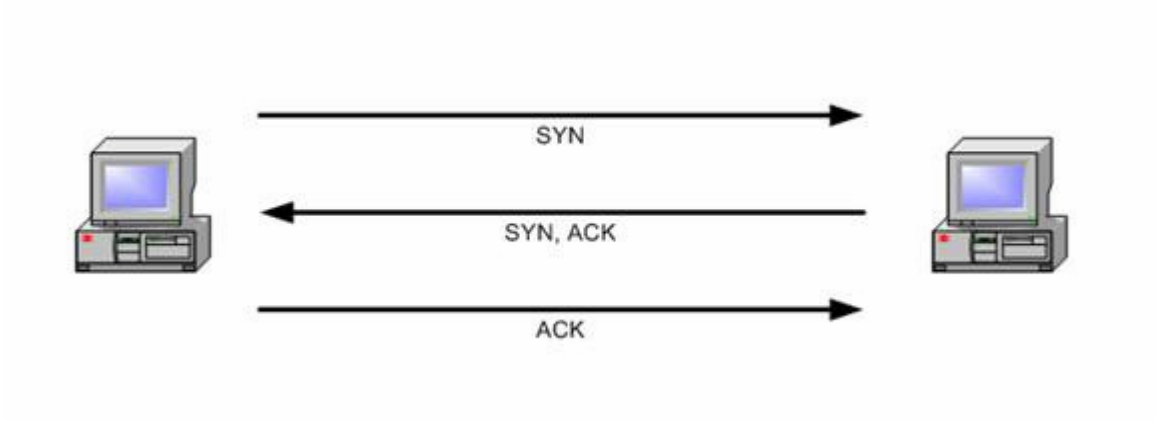


Figure 1-3: Connection establishment.

Connectionless Sessions

In contrast, connectionless sessions communicate without receipt acknowledgements or sequence numbers. Connectionless protocols aren't without merit; they just don't have any reliability mechanisms built in, since they're mainly built for speed. Their lower overhead means faster communication, but also means that reliability is left to a higher layer in the model. For example, if a file was being transferred with a connectionless protocol, any missing parts would need to be re-requested (if necessary) by the receiving application – the protocol would not account for these missing parts automatically. Connectionless protocols are most often used by applications with a high degree of time sensitivity. Consider a streaming audio application – it would have little to gain from acknowledgements, but benefits greatly (in terms of speed) from the lower overhead.

Table 1-5: Examples of Transport layer protocols.

Protocol	Purpose
Transmission Control Protocol (TCP)	Connection-oriented TCP/IP sessions
User Datagram Protocol (UDP)	Connectionless TCP/IP sessions
Sequenced Packet Exchange (SPX)	Connection-oriented sessions in IPX/SPX protocol suite.

EXAM TIP: Remember that the protocol data unit (PDU) of the Transport layer is a segment.

Flow Control

Another function of the Transport layer is to employ flow control mechanisms to ensure that systems don't send data at rates beyond what the receiving system can handle. In network environments, systems use a portion of memory referred to as buffer space to hold data that has been received more quickly than they can process it. However, once this buffer space fills, systems run the risk of dropping data that they can't find room for. To account for this, the Transport layer on the receiving machine will pass status information to the sender, asking it to stop sending segments if its buffers become full. Once buffer space becomes available, another message is passed to the sender allowing it to resume transmission.

In order to try and regulate the flow of data between the sending and receiving systems right from the start, systems using connection-oriented communication negotiate what is referred to as a window size. The window regulates how much data can be sent before acknowledgements must be received. Once acknowledgements for data sent have been received, the sender can resume the sending of data. Some protocols (such as TCP) implement this using what are known as sliding windows.

This is best illustrated with an example. Let's say that as part of establishing a connection, two systems agree on an 8K window size. If the sender is preparing to transfer 20K of data, it can send off the first 8K (say segments 1 through 8) before having to wait for acknowledgements. Once acknowledgements for segments 1 through 3 are received, segments 9 through 11 can be sent, and so on. The window slides as acknowledgements are received, allowing more data to be sent. When data is sent, a retransmit timer is also set at the source machine. If an acknowledgement is not received before the timer expires (perhaps because of a network problem or congestion), the sender resends the data, and resets the timer. When a receiving system's buffers are full, it will send back a message with a zero window size.

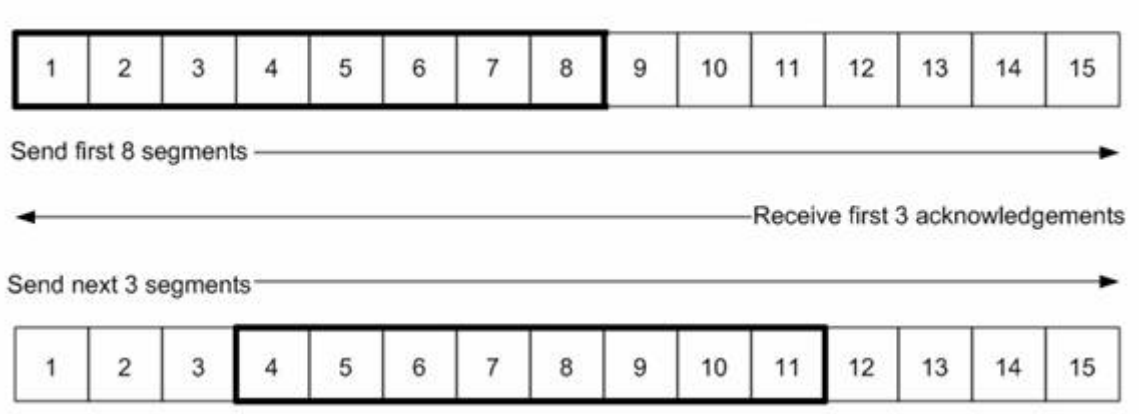


Figure 1-4: Sliding windows.

EXAM TIP: When a protocol (like TCP) uses sliding windows for flow control, the relationship between segments sent and acknowledgements received is usually not one-to-one. For example, a destination host can generally confirm receipt of many segments with a single acknowledgement.

Other methods of flow control found at the Transport layer include multiplexing and parallelization. Multiplexing involves combining multiple segments along a single channel to better utilize bandwidth. Parallelization is a technique used on slower networks, increasing throughput by passing data to the network in parallel streams.

A great example of multiplexing is found in the HTTP 1.1 protocol, which uses TCP. Previously, browsing a website required a separate connection for each and every object to be downloaded – this included all graphics, html files, and so forth. By multiplexing different transport segments over a single connection, HTTP 1.1 eliminates the overhead associated with setting up and tearing down multiple connections.

Network Layer

The Network layer of the OSI model is commonly referred to as Layer 3, and has the following responsibilities:

- **Routing**. When a host on one network wishes to exchange data with a host on another, packets will be sent to a router interface. After determining where the packet should be forwarded next using information found in its routing table, a router will switch the packets out of the optimal interface. This process will take place at each router encountered on a packet's journey to the destination host. Routing protocols are used to allow routers to exchange information with one another.
- **Network Addressing**. Each host on a routed internetwork will have at least one network address. A network address is made up of two parts – the first part identifies the network, while the second identifies a unique host on that network. These addresses have different formats depending on the routed protocol in use – we'll look at examples shortly.

Table 1-6: Examples of Network-layer protocols.

Protocol	Purpose
Internet Protocol (IP)	TCP/IP addressing and routing

Internetwork Packet Exchange (IPX)	IPX/SPX addressing and routing
Internet Control Message Protocol (ICMP)	Diagnostics and error notification
Internet Group Management Protocol (IGMP)	Multicast group management

EXAM TIP: Remember that the protocol data unit (PDU) of the Network layer is referred to as a packet. In some cases, you may also see this PDU referred to as a datagram.

Routed Protocols

Routed protocols are those whose addressing schemes differentiate between a network and host portion, allowing a host and its network to be uniquely identified. The main responsibility of a routed protocol is moving data between hosts, across networks. Examples of routed protocols include IP, IPX, and AppleTalk. Consider the IP, IPX, and AppleTalk addresses below. Notice the split between the network and host portion of the address in each case.

Table 1-7: IP, IPX, and AppleTalk addresses.

Address	Protocol	Comments
149.234.16.3	IP	IP addresses are represented in dotted-decimal notation. In this case, 149.234 represents the network, while 16.3 represents the host.
75E3F210.001022EA27BE	IPX	IPX addresses are represented in hexadecimal. The first 32 bits (75E3F210) represent the network; the final 48 bits (001022EA27BE) represent the host.
1578.6	AppleTalk	AppleTalk addresses are represented in dotted-decimal notation. The network number is a 16-bit number, in this case '1578'. The host is represented by an 8-bit node number, in this case '6'

Routing Protocols

Routing protocols are those used by routers to exchange information stored in their routing tables. For example, if you wanted three routers on your network to exchange information with one another dynamically, you would configure each with a common routing protocol. This would allow them to 'talk' to each other, exchanging information about the networks they know about. Common routing protocols that you'll find on networks include the Routing Information Protocol (RIP), Open Shortest Path First (OSPF), and Interior Gateway Routing Protocol (IGRP). These routing protocols and others will be discussed in detail in Chapter 8.

EXAM TIP: Routers exist at the Network Layer of the OSI model

Data Link Layer

The Data Link Layer of the OSI model acts as an interface between the Network and Physical layers. The main responsibilities of the Data Link layer include:

- **Data framing and physical addressing**. When data is passed to the Data Link layer, it is framed for transmission using various LAN and WAN protocols. This allows network protocols to be transmitted over different network technologies including Ethernet, Token Ring, and Frame Relay as examples. Hardware or Media Access Control (MAC) addressing is used to uniquely identify hosts at the Data Link layer. Since they make forwarding decisions based on MAC addresses, bridges and switches are examples of equipment found at this layer.
- **Flow control, error checking, and frame sequencing**. Data Link layer devices are capable of transmitting flow control codes that identify whether upper layer protocols are capable of receiving data at the current rate. Error checking is provided in the form of a Cyclic Redundancy Check (CRC), a simple mathematical calculation performed on each frame to ensure it hasn't been corrupted in transit. Frame sequencing reorders frames that were received in a different order than they were sent.
- **Interacting with Network layer protocols**. When a host receives a frame, the frame header contains information on which Network layer protocol the data will be passed to. The Data Link layer helps to make network technologies independent of the upper layer protocols in use.

Table 1-8: Examples of Data Link layer protocols.

Technologies	Purpose
Ethernet (802.3)	Contention-based LAN technology
Token Ring (802.5)	Token-passing LAN technology
Wireless LAN (802.11)	Wireless LANs
Frame Relay	Packet-switched WAN technology
ISDN	Digital dial-up connections

EXAM TIP: Remember that the protocol data unit (PDU) of the Data Link layer is referred to as a frame.

The Data Link layer is actually comprised of two sub-layers (defined by the Institute of Electronics and Electrical Engineers – the IEEE), called Media Access Control (MAC) and Logical Link Control (LLC).

Figure 1-5: Data link sub-layers.

Logical Link Control

The Logical Link Control sub-layer is the upper layer of the two, and has responsibility for separating Network layer protocols from the underlying network technology. Because of this, network protocols and technologies can be changed fairly easily. The LLC is primarily concerned with providing Service Access Points (SAPs) between the MAC sub-layer and Network layer protocols. For example, the frame header might contain SAP addresses stating that the contents are destined for layer 3 protocols like IP (SAP address 06) or IPX (SAP address E0). The flow control and frame sequencing mechanisms described earlier are also the responsibility of the LLC. The LLC is defined in the IEEE 802.2 standard.

You will often see Data Link layer SAPs referred to as DSAPs and SSAPs. DSAPs (Destination Service Access Points) are the interfaces used by the LLC to interact with the Network layer.

Conversely, SSAPs (Source Service Access Points) are the interfaces used by the Network layer to interact with the LLC.

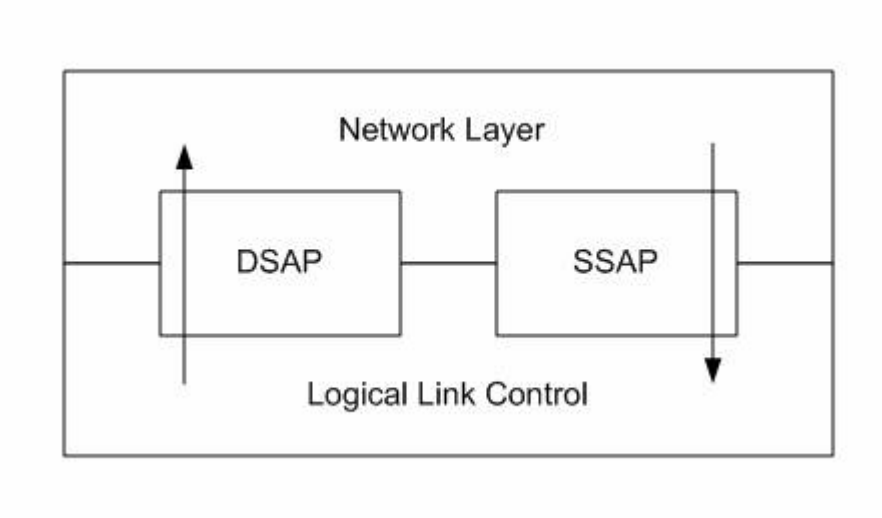


Figure 1-6: LLC and Network layer interaction.

Media Access Control

The Media Access Control sub-layer is primarily responsible for MAC addressing and the hardware functions of the Data Link layer. This sub-layer handles the transmitting and receiving of data over the network, converting data to (and from) bits according the rules of the technology in use. On an Ethernet network, this would involve adding information such as the source and destination hardware addresses, and then following the rules of Carrier Sense Multiple Access with Collision Detection (CSMA/CD) transmission. Ethernet (including CSMA/CD) and other Data Link layer technologies will be discussed in detail in Chapter 2. The MAC sub-layer is defined by IEEE standards such as 802.3 (Ethernet), 802.5 (Token Passing), and others.

EXAM TIP: Bridges and switches exist at the Data Link layer of the OSI model.

Physical Layer

The Physical layer of the OSI model is concerned with the electrical, optical, and mechanical properties of the network, including elements such as voltage, media, connector types, signal regeneration, and so forth. The physical layer doesn't actually alter packets, but rather acts as the transmission facility over which the actual bits (1's and 0's) are sent. This isn't limited to plain old copper wire – it can include optical signals, radio waves, and infrared transmissions to name but a few. Examples of equipment found at the Physical layer include network cabling, hubs, and repeaters. A number of popular Physical layer standards are listed in Table 1-9.

Table 1-9: Physical layer standards.

Technology	Purpose
High Speed Serial Interface (HSSI)	High speed serial communications
EIA/TIA-232	Low speed serial transmissions
V.35	ITU-T serial transmission standard

The 4-Layer TCP/IP Model

The Department of Defense TCP/IP model is a 4-layer model that defines areas of responsibility much like the OSI, while providing insight into the functions of the different protocols that make up the TCP/IP suite. The model provides an excellent point of reference when compared to the OSI. We won't look at all the details of the TCP/IP model just yet – the majority will be covered in Chapter 4. My feeling is that the data encapsulation process is much better explained using a popular protocol suite.

To begin, let's take a look at how the TCP/IP model maps to the OSI model. While the names of the TCP/IP layers are different, they generally encompass the same responsibilities as one or more OSI layers. Consider the diagram below.

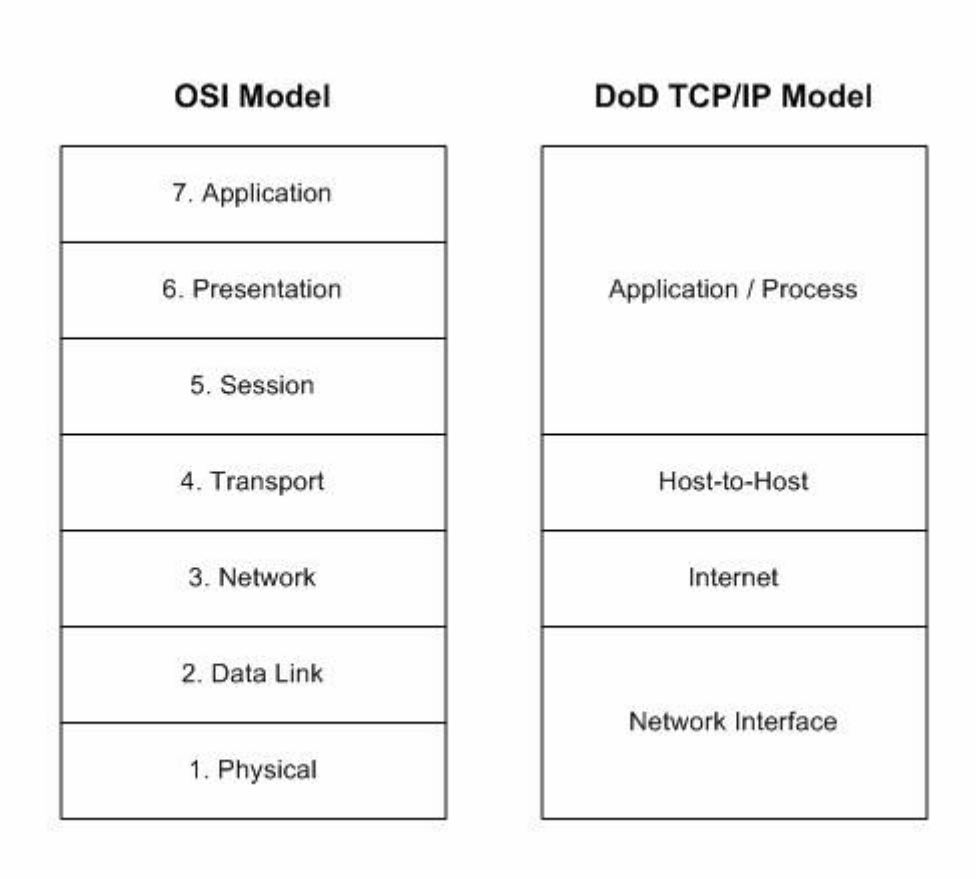


Figure 1-7: OSI and TCP/IP models.

EXAM TIP: Although the layers of the TCP/IP model technically use different names, Cisco will still refer to protocols by their associated OSI layer name. For example, Cisco will describe TCP as being a Transport layer protocol.

For the sake of illustration, I've included some of the key protocols that make up the TCP/IP suite in Figure 1-8. Be aware that the terms data, segment, packet, and frame still apply as data is encapsulated in the TCP/IP model.

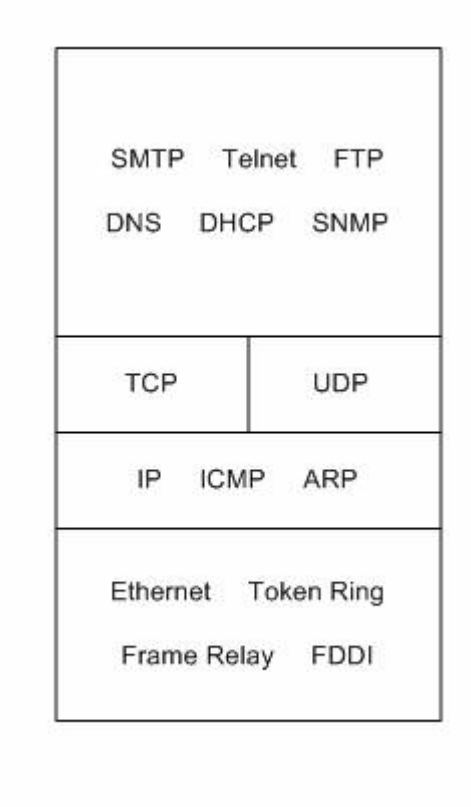


Figure 1-8: TCP/IP protocol stack including common protocols and network technologies.

Data Encapsulation Process

The primary reason for looking at any network model is to better understand how systems communicate. In real-life, network communication requires that data be encapsulated by the sender, transmitted over the network, and then de-encapsulated by the receiver. This is best illustrated by looking at what happens when one system running TCP/IP sends data to another. The list below outlines 5 simplified steps in a typical TCP/IP data transfer over an Ethernet network. Note that each layer considers whatever has been passed down to it from an upper layer as “data”. It doesn’t concern itself with what was added by the upper layers.

Step 1. Data is created by an application such an FTP client program. Let’s assume that a file transfer is being initiated with a local FTP server.

Step 2. The data is passed to the Host-to-host (Transport) layer, where it is encapsulated to include source and destination port numbers. These uniquely identify the applications that the data should be passed between. For example, if this data were being sent to an FTP server, the destination port would be TCP 21. The data is now considered to be a segment.

Step 3. The data is passed to the Internet (Network) layer, where it is again encapsulated to include information such as the source and destination IP addresses. The data is now considered to be a packet.

Step 4. The data is passed down to the Network Interface (Data Link) layer, where it is encapsulated for Ethernet to include source and destination MAC addresses, as well as the an

error-checking mechanism known as a cyclic redundancy check (CRC). The data is now considered to be a frame.

Step 5. The data is converted to a series of bits, and transmitted across the network.

EXAM TIP: A CRC is also often referred to as a Frame Check Sequence (FCS).

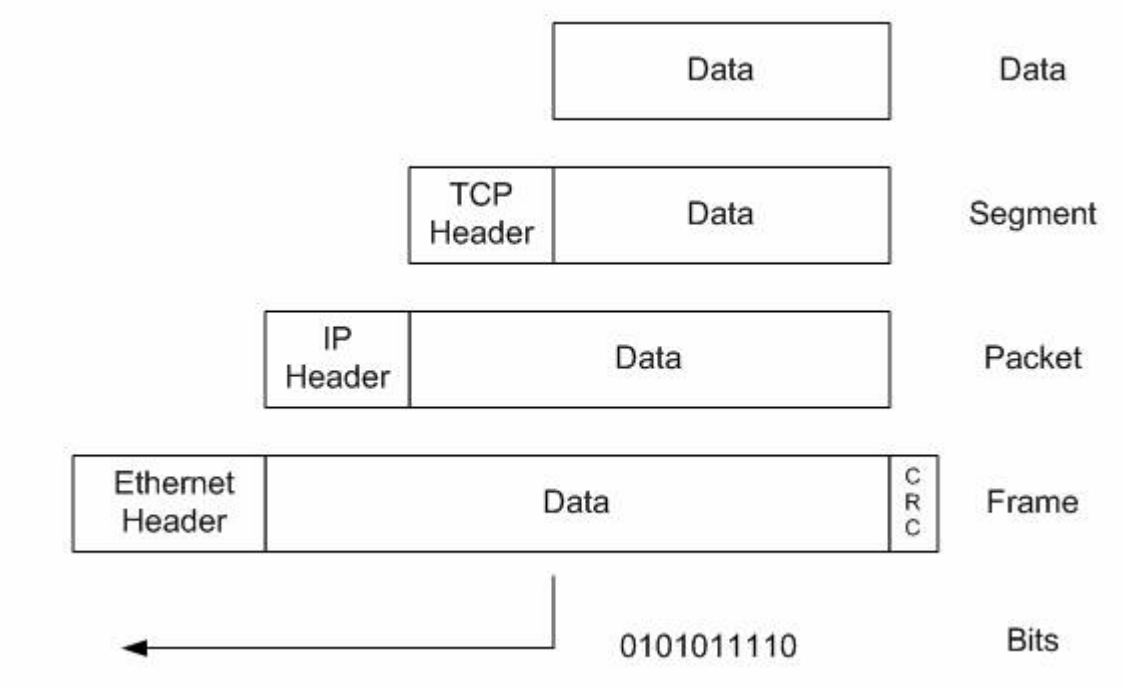


Figure 1-9: TCP/IP data encapsulation process.

Note that upon reaching the destination host, the entire process happens in reverse, with each layer de-encapsulating the data by striping away the information that was added at each layer. Eventually, the required data is passed to the FTP server as intended by the FTP client application. Consider the frame captured in Figure 1-10 using Ethereal, a network protocol analyzer. Notice that each heading area directly corresponds to the encapsulation process just defined (with the exception that the program shows the layers in reverse order).

Ethernet II

Internet Protocol, SrcAddr: 192.168.0.1 (192.168.0.1), DstAddr: 192.168.0.135 (192.168.0.135)

Transmission Control Protocol, Src Port: 4653 (4653), Dst Port: ftp (21), Seq: 2739356837, Ack: 204742999

File Transfer Protocol (FTP)

Figure 1-10: FTP session packet capture.

STUDY TIP: Ethereal can also be downloaded from http://www.ethereal.com/download.html

Cisco Hierarchical Network Design Model and Chapter Summary

When it comes to network design, you're pretty much left with two options – a flat design, or one that involves some type of hierarchy. A flat design can be very limiting in terms of performance and scalability, and in all but the smallest networks would not be recommended. For example, on a flat network issues like broadcast traffic can quickly overwhelm network systems and negatively impact performance. In contrast, a hierarchical design will allow for unique divisions of responsibility to be created on the network. Thus a higher degree of performance, reliability, scalability and security can be achieved. The Cisco network design model is a reference model for creating hierarchical networks that attempts to account for these factors, while also providing an insight as to where different network elements should be deployed and why.

The Cisco network design model consists of three layers. These include:

§ The Core Layer.

§ The Distribution Layer

§ The Access Layer

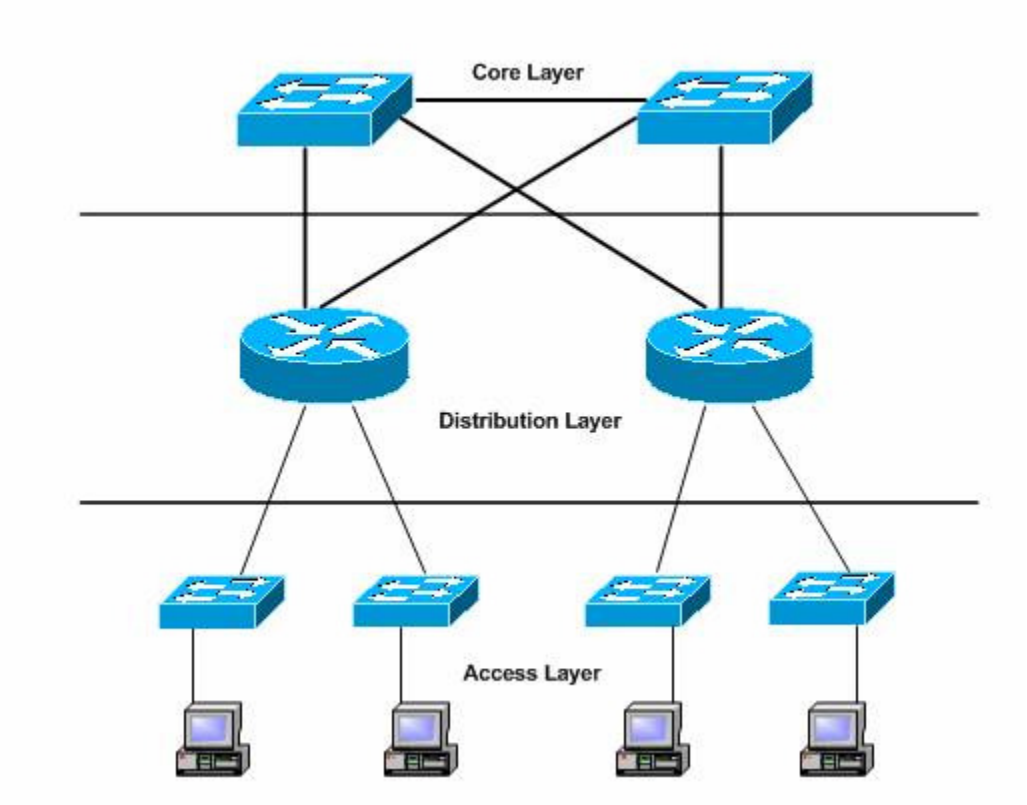


Figure 1-11: Cisco hierarchical network design model.

The Core Layer

The core layer describes what is often referred to as the network backbone. Its main responsibility is ensuring that data is passed at high speeds between different sites. Because of this high-speed requirement, the backbone should usually make use of switching technologies instead of routing. While we'll look at the differences between switching and routing in later chapters, for now it is sufficient to say that switching is significantly faster than routing.

The core layer should also provide a high degree of reliability and fault tolerance. This is usually implemented using higher-end equipment and redundant links. For the most part, the core layer should not be scaled to include additional equipment if performance is deteriorating. In such cases, backbone switches should be replaced with better performing models. By replacing equipment, the core layer maintains a constant diameter, helping to avoid the introduction of additional latency.

As a general rule, anything that slows down performance should be kept away from the core layer. Beyond routing, this also means avoiding features such as access lists, firewall and intrusion detection system (IDS) sensors - these inspect traffic based on network addresses and applications, and can negatively impact performance.

Distribution Layer

The distribution layer of the Cisco model acts as the intermediary or demarcation point between the core and access layers. It is at this layer that routing is usually handled, between the access layer, the core layer, and between the different access layer broadcast domains. Other functions of the distribution layer include route aggregation and redistribution, media translation, implementation of security policies, and broadcast domain definition.

Address aggregation involves setting up a network addressing scheme such that multiple networks may exist behind a single routing table entry. This helps make routing tables smaller and more efficient. On IP networks, this is usually a function of the routing protocol used, and the use of addressing schemes that incorporate Variable Length Subnet Masks (VLSM) and/or Classless Inter-Domain Routing (CIDR). VLSM is a special way of designing IP addressing with custom subnet masks, while CIDR provides the ability to aggregate ranges of addresses – both will be discussed in detail in Chapter 5. Route redistribution is a process by which the networks learned by one routing protocol are shared with another routing protocol, a concept that will be explored in more detail in Chapter 8.

Media translation involves just what the name suggests – for example, the distribution layer would be the point at which your network transitions between Cat5 Ethernet LANs and on to your fiber-optic backbone.

Security policies (in the form of ACLs or Access Control Lists) are usually implemented at the distribution layer as well. Configured on routers, these lists allow you to filter the types of traffic that users or systems are allowed to pass between connected networks. For example these lists might allow only certain systems or applications the ability to transfer data between subnets.

Finally, the distribution layer is also where broadcast domains interconnect. Routers act as the demarcation points between broadcast domains, since a router is configured not to allow broadcast traffic to pass by default.

Access Layer

The access layer of the Cisco model represents the point where network users gain access to local resources. High bandwidth is provided at the distribution layer through the use of switches, which help to separate hosts into smaller collision domains. Often referred to as the workgroup

layer, the access layer also provides network access to smaller home or branch offices by way of Frame Relay, ISDN and demand-dial connections.

Summary

We began this chapter with a look at the importance of network models, including the reasons for their modular nature. A look at the OSI model stressed the importance of understanding the concept of layered communication, protocol data units, and the functions of each layer. Do not underestimate the importance of remembering not only the various functions of each, but also the protocols, data units, services, and types of equipment found at each layer.

A look at the TCP/IP model provided a comparison with the OSI model, including the mappings between the layers of each. Examining the data encapsulation process helped to provide perspective on how a real network protocol goes about preparing data for network transmission.

Finally, an overview of the Cisco network design model provided insight into Cisco's perspective on the proper design of hierarchical networks. Be sure to understand not only the layers but also the equipment and functional details of each.

Understanding Network Models: Review Exercises

Chapter 1 Exercises

1. For each description provided on the left, enter the appropriate layer, term, or protocol on the right.

a) UDP is an example on a protocol at this OSI layer.	
b) The term is used to describe a protocol data unit at the Data Link Layer.	
c) The Data Link Layer of the OSI model is made up of these two sub-layers	
d) EIA/TIA-232 is an example of a standard found at this layer of the OSI model.	
e) Security policies are generally implemented at this layer of the Cisco network design model.	
f) RPC is an example of a protocol at this layer of the OSI model.	
g) The term is used to describe a protocol data unit at the Transport Layer.	
h) Frame Relay exists at this layer of the OSI model.	
i) ICMP is an example of a protocol at this layer of the OSI model.	
j) This layer of the OSI model is concerned with the creation of end-to-end connections between hosts	
k) This protocol is concerned with network	

addressing and routing in the TCP/IP protocol stack.	
l) Users gain access to the network at this layer of the Cisco network design model.	
m) A packet is the protocol data unit of this OSI layer.	
n) High-speed switching and a constant diameter characterize this layer of the Cisco network design model.	
o) This type of addressing is used at the Data Link layer of the OSI model.	
p) These are the 7 layers of the OSI model from top to bottom.	
q) ARP exists at this layer of the OSI model.	
r) POP3 exists at this layer of the OSI model.	
s) RTF is an example of a file format found at this OSI layer.	
t) This is a connection-oriented transport protocol in the IPX/SPX suite.	
u) A hub is an example of equipment that exists at this layer of the OSI model.	
v) Broadcast domains interconnect at this layer of the Cisco network design model.	
w) Encryption and compression happen at this layer of the OSI model.	

Answers to Chapter 1 Exercises

Answers to Question 1:

a) Transport Layer
b) Frame
c) Logical Link Control (LLC) and Media Access Control (MAC)
d) Physical Layer
e) Distribution Layer
f) Session Layer
g) Segment
h) Data Link Layer
i) Network Layer
j) Transport Layer
k) Internet Protocol (IP)
l) Access Layer
m) Network Layer
n) Core Layer
o) Media Access Control (MAC) addressing
p) Application, Presentation, Session, Transport, Network, Data Link, Physical
q) Network Layer
r) Application Layer
s) Presentation Layer

t) Sequenced Packet Exchange (SPX)
u) Physical Layer
v) Distribution Layer
w) Presentation Layer

Review Questions (Multiple Choice)

Review Questions

1. At which OSI layer do routers operate?

A. Network
B. Data Link
C. Transport
D. Session

2. Which of the following is not an example of a Session Layer protocol?

A. SQL
B. RPC
C. XWindows
D. ASCII

3. By what term is a Transport layer Protocol Data Unit (PDU) known?

A. Packet
B. Frame
C. Segment
D. Datagram

4. Which of the following represent Transport Layer protocols? (Choose all that apply)

A. TCP
B. IPX
C. ICMP
D. UDP

5. The Logical Link Control communicates with the Network Layer using which of the following? (Choose all that apply)

A. MAC address
B. Source Service Access Points
C. Port Numbers
D. Flow Control

6. At which layer of the Cisco network design model are Access Lists usually defined?

A. Core
B. Access
C. Distribution
D. Network

7. Which of the following are characteristics of connection-oriented communications? (Choose all that apply)

A. Positive Acknowledgement
B. Route Definition
C. Addressing
D. Sequence Numbers

8. At which TCP/IP layer does Internet Protocol reside?

A. Network
B. Internet
C. Network Interface
D. Data Link

9. Which of the following is used to determine whether a frame has been corrupted in transit?

A. Flow Control
B. ICMP
C. CRC
D. MAC Addressing

10. At which OSI Layer does data framing occur?

A. Network
B. Session
C. Transport
D. Data Link

11. Which Cisco network design layer do high-speed switching and a constant diameter characterize?

A. Access
B. Core
C. Distribution
D. Host-to-host

12. At which OSI Layer are data representation, compression, and encryption defined?

A. Data Link
B. Session
C. Presentation
D. Application

13. Which of the following is not a benefit of layered network communication models?

A. Separation of network functions
B. Changes to one layer affect the others
C. Makes networking easier to explain
D. Helps ensure interoperability between vendors

14. Which of the following defines the correct PDU order in the data encapsulation process?

A. Data, Packet, Segment, Frame, Bits
B. Data, Segment, Frame, Packet, Bits
C. Data, Segment, Packet, Frame, Bits
D. Data, Frame, Packet, Segment, Bits

15. What are the phases of a connection-oriented communication session? (Choose all that apply)

A. Call Setup
B. Data Framing
C. Data Transfer
D. Call Termination

16. At which TCP/IP Layer does program-to-program communication take place?

A. Process
B. Host-to-host
C. Internet
D. Network Interface

17. True or False. MAC addresses change as data moves between routers across an internetwork.

A. True
B. False

18. The TCP/IP Network Interface Layer maps to which layer(s) of the OSI model? (Choose all that apply)

A. Data Link
B. Network
C. Session
D. Physical

19. Bridges are said to exist at which layer of the OSI model?

A. Network
B. Transport
C. Data Link
D. Physical

20. Which of the following are devices commonly found at the Network Layer? (Choose all that apply)

A. Router
B. Bridge
C. Switch
D. Hub

21. Which Cisco network design layer is responsible for providing connectivity to home or branch offices using demand-dial connections?

A. Core
B. Distribution
C. Access
D. Data Link

22. Which Layer of the OSI model is primarily concerned with network addressing and routing?

A. Layer 4
B. Layer 2
C. Layer 5
D. Layer 3

23. True or False. A connectionless session makes use of sequence numbers.

A. True
B. False

24. Which of the following represent Transport layer flow control mechanisms? (Choose all that apply)

A. Buffering
B. Windowing
C. Multiplexing
D. Parallelization

25. High Speed Serial Interface (HSSI) exists at which OSI Layer?

A. Data Link
B. Network
C. Physical
D. Application

Answers to Review Questions

1. A. Routers operate at the OSI Network layer.

2. D. ASCII is a Presentation layer format.

3. C. Transport layer PDUs are referred to as segments.

4. A and D. TCP and UDP are both Transport layer protocols.

5. B. The LLC communicates with the Network Layer using Source (and Destination) Service Access Points.

6. C. Access Lists are usually defined at the Distribution layer.

7. A and D. Positive Acknowledgement and Sequence Numbers are characteristics of the OSI Transport layer.

8. B. IP resides at the Internet layer of the TCP/IP model. Note that the Network layer refers to the OSI model.

9. C. The Cyclic Redundancy Check (CRC) is calculated to be sure that frames have not been corrupted in transit.

10. D. Data framing occurs at the Data Link layer of the OSI model.

11. B. High-speed switching and a constant diameter characterize the Core layer.

12. C. The presentation layer is responsible for data representation, compression, and encryption.

13. B. Read the question carefully. Layered models are concerned with changes in one layer not

affecting the others.

14. C. Data is encapsulated in the order Data, Segment, Packet, Frame, Bits.

15. A, C, and D. Call setup, data transfer, and call termination are the three phases of connection-oriented communication sessions.

16. A. Program-to-program communication takes place at the TCP/IP Process layer.

17. A. MAC addressing changes at each router than a frame encounters as it is routed across an internetwork.

18. A and D. The TCP/IP Network Interface layer maps to the Data Link and Physical layers of the OSI model.

19. C. Bridges exist at the Data Link layer of the OSI model.

20. A. Routers are network equipment found at the Network layer of the OSI model.

21. C. Demand-dial connectivity for branch offices is found at the Access layer of the Cisco network design model.

22. D. Layer 3 (the Network layer) is the OSI layer concerned with network addressing and routing.

23. B. Connectionless sessions do not make use of sequence numbers.

24. A, B, C and D. Buffering, windowing, multiplexing and parallelization are all examples of Transport layer flow control mechanisms.

25. C. HSSI is a Physical layer standard.

Exam Quick Review Guide

Exam Quick Review

Benefits of layered models:

- Less complexity, easier to learn.
- Ensures interoperability between vendors and equipment
- Provide documented standards
- Modular, allowing changes to be made to one layer without affecting others.

Layers of the OSI Model:

- Application
- Presentation
- Session
- Transport
- Network
- Data Link
- Physical

Remember – **A**ll **P**eople **S**eem **T**o **N**eed **D**ata **P**rocessing.

Protocol Data Units and OSI Model:

- Application, Presentation and Session = DATA
- Transport = SEGMENT
- Network = PACKET
- Data Link = FRAME
- Physical = BITS

Functions, Protocols, and Equipment at OSI Layers:

- **Application**. Concerned with program-to-program communication, where the user interacts with the network. Protocols and applications found here include Telnet, FTP, Email, Word Processors, and Web browsers.
- **Presentation**. Concerned with data representation, compression, and encryption. Protocols and standards found here include ASCII, EBCDIC, Unicode, TIFF, JPEG, MPEG, AVI, and MIDI.
- **Session**. Concerned with creation, management and termination of sessions between hosts. Protocols include NFS, SQL, RPC, ASP, and XWindows.
- **Transport**. Responsibilities include data segmentation, flow control, establishment of end-to-end connections, and the reliable delivery of data. Connection oriented protocols include TCP and SPX. UDP is an example of a connectionless protocol.
- **Network**. Responsible for logical addressing and routing. Protocols found here include IP, IPX, ICMP, ARP, and IGMP. Routers exist at this layer.
- **Data Link**. Responsible for data framing, flow control, and interacting with Network layer protocols and the physical media. Comprised of 2 sub-layers, the Logical Link Control and Media Access Control. LLC is responsible for flow control, frame sequencing, and Layer 3 interaction. MAC is concerned with media access and physical addressing. Bridges and switches exist at this layer.
- **Physical**. Concerned with the electrical, optical, and mechanical properties of the network including cabling, connectors and physical connectivity. Standards found here include HSSI, V.35, and EIA/TIA-232. Hubs and repeaters exist at the layer.

TCP/IP Model Layers:

- **Process** / **Application**. Maps to the Application, Presentation, and Session layers of the OSI Model.
- **Host-to-host**. Maps to the OSI Transport layer.
- **Internet**. Maps to the OSI Network Layer.
- **Network Interface**. Maps to the OSI Data Link and Physical layers.

Cisco Layered Model:

- **Core Layer**. Network backbone that connects locations. Characterized by high-speed switching, constant diameter, and redundancy.
- **Distribution Layer**. Intermediary between the Core and Access layers, it is responsible for route aggregation, media translation, access lists, and broadcast domain definition. Routing usually occurs at this layer.
- **Access Layer**. Point of connectivity for end users via switches or hubs. Responsible for collision domain definition, providing access to small offices using ISDN, Frame Relay, and demand-dial routing.

Chapter 2: Networking Fundamentals

Network Fundamentals: LANs and WANs

Posted: September 01, 2005

By: Dan DiNicolo

Stay updated with our RSS feed!

The following is part of a complete CCNA/CCDA study guide by Dan DiNicolo, posted on the 2000Trainers.com web site. Each chapter of the book is presented as a series of articles, and you will find a complete TOC for a given chapter at the bottom of each article once the entire chapter is posted. In the meantime, a complete list of the articles posted in the series can always be found here.

A new chapter will be posted as a series of articles each week (approximately), until the entire study guide has been posted. The materials in the text are applicable to both Cisco's CCNA and CCDA certification exams. In cases where materials apply to one exam only, a note to that effect will be posted within each article.

Whether you're preparing for the CCNA or CCDA, the most critical requirement is a solid understanding of networking theory. While you personally may be more worried about router configuration or the related commands, do not underestimate the emphasis placed on general networking concepts. For all intents and purposes, your knowledge of these concepts will likely be the difference between passing and failing the exams, so be prepared.

There are a variety of elements involved in how communication happens between systems across an internetwork. These not only include equipment and cabling, but also topologies, transmission methods, and technologies. Take the time to understand these different elements and their responsibilities, and you'll find both exams much less challenging, hopefully even simple.

The material to be covered in this chapter includes:

- *An overview of LANs versus WANs*
- *An overview of basic network equipment including repeaters, hubs, bridges, switches, and routers*
- *Network transmission methods*

- *Network cable types and wiring standards*
- *Media access methods*
- *Network topologies*
- *Ethernet*
- *Token ring*
- *Fiber Distributed Data Interface (FDDI)*
- *LANs and WANs*

The most basic concept that you'll need to understand before going any further is the difference between LANs and WANs. A Local Area Network represents a group of connected computers, usually within a given office or building. For the most part, LANs are distinguished by their limited geographic distance and relatively high transmission speeds - anywhere from less than ten and up to hundreds of megabits (millions of bits) per second. Common LAN technologies include Ethernet, Token Ring, and FDDI, each of which we'll look at in this chapter.

Wide Area Networks interconnect LANs over great geographic distances, sometimes spanning the globe. WAN technologies differ from those found on a LAN, and are mainly characterized based on their ability to span large distances and lower relative speeds. For the most part, WANs rely on the infrastructure of telecommunications service providers to deliver connectivity over these long distances. Common speeds found on WANs vary widely, ranging anywhere from a few kilobits (thousands of bits) per second, up to multiplemegabits (millions of bits) per second and higher. Examples of WAN technologies include simple dial-up modem connections, Frame Relay, ISDN, ATM, and others. WAN technologies and protocols will be looked at in detail in Chapter 11.

Basic Network Equipment

On any LAN or WAN you're likely to come across a variety of different hardware devices. Examples include hubs, switches, and routers to name but a few. When considering any given piece of network equipment, it's important to understand both the role it plays in the communication process, as well as how it relates to the OSI model. Furthermore, you should be able to explain why you might choose one piece of hardware over another. For example, you should be able to outline the reasons why a switch provides performance advantages over a hub. The equipment that we're going to look at in this section includes repeaters, hubs, bridges, switches and routers.

Repeater

Although you're not likely to run into too many traditional LAN repeaters any more, they once served an important purpose. A repeater is a powered network device that exists for the purpose of regenerating a signal as it travels along the network, allowing longer distances to be spanned. As data travels over a network, signal strength tends to degrade as it moves farther along its path, a concept referred to as attenuation. When signals enter a repeater, the device boosts their strength, but doesn't actually modify the data. Most commonly found on Ethernet bus networks using coaxial cable, repeaters are considered to exist at the Physical Layer of the OSI model.

Hub

A hub is a network connectivity device that you're almost certainly familiar with. In the most basic sense, a hub is really nothing more than a multiport repeater. As signals are sent between systems connected to a hub, they are not only regenerated, but also forwarded out each port. In this way, all devices plugged into a hub are exposed to all traffic passing between systems,

regardless of whether that traffic is actually relevant to them. Like repeaters, hubs are also considered to exist at the Physical Layer of the OSI model – they neither modify the data stream nor make any decisions. Instead, they act as a conduit by which systems can interconnect and communicate.

The limitations of a hub become evident as a network grows. While providing basic connectivity, a hub does nothing to prevent network collisions or broadcasts from reaching all connected systems. For this reason, all devices connected to a hub are considered to be part of the same broadcast domain and the same collision domain – concepts we'll look at shortly. In order to be able to control traffic to a greater degree, you would need to implement devices capable of making forwarding decisions based on source or destination addresses. Examples of such devices include switches, bridges, and routers.

While you're probably familiar with Ethernet hubs, there are a variety of hubs that exist for different network technologies, including Token Ring. In a Token Ring environment, systems are connected to something that looks similar to a hub, but is properly referred to as a Multi-Station Access Unit (MSAU). We'll discuss the details of how an MSAU works in the Token Ring section. For now it is enough to understand that it is also a Physical Layer device.

EXAM TIP: Hubs and repeaters exist at the Physical Layer of the OSI model.

Bridge

As the size of LANs grow and the amount of traffic on a network begins to have a noticeable effect on performance, a new way is needed to help separate or segment traffic. A bridge is a Layer 2 device that acts as an intermediary to which different network segments (or sections) are attached. Used to separate traffic, the role of a bridge is to inspect all frames as they reach one of its interfaces, and make a decision on whether to forward these frames to other connected segments. It does this by examining the destination MAC or BIA (Burned In Address) address of frames that it encounters. MAC Addresses are also known as Physical addresses. If the destination MAC address is connected to the same segment as the sender, the bridge will not forward the frame, thus helping to reduce unnecessary traffic on other segments. Note that the bridge doesn't change the frames; it simply makes forwarding decisions based on the destination MAC address. However, a bridge does do error checking on frames – it will calculate a value in the frame called the Cyclic Redundancy Check (CRC). The CRC is a calculation added to the frame by the sender and recomputed by bridges as well as the destination system. If the value is always the same, the frame is valid. If not, the frame is corrupt and will be discarded.

A bridge keeps track of all systems on its connected segments in something referred to as its MAC address table. This table specifies the segment on which a system with a given MAC address is located. On older bridges, these tables needed to be created manually by inputting MAC addresses. On more recent bridges, this table is created dynamically as the bridge inspects frames that it receives. It does this by also looking at the source MAC address of frames, and adding them to its table according to the interface on which they were received. Table 2-1 provides a basic example of a bridging table.

Table 2-1: MAC address (bridging) table.

Interface	**MAC Address**
E0	010123E4A201
E1	010123E5AB07
E1	010123F4121A

E2	010123C14298

Note in the example above that there are 3 active interfaces on the bridge, and that interface E1 (Ethernet 1) has two connected systems that the bridge currently knows about.

There are limitations to the magic of a bridge. Some traffic will be destined for all computers, while other traffic will be meant for a select group of computers (referred to as broadcasts and multicasts respectively); a bridge will always forward these types of traffic to all connected segments. To that end, it can be said that a bridge segments the network into different collision domains, while all systems remain part of the same broadcast domain.

What is a collision domain? The answer is incredibly simple. A collision domain is the group of network systems between which data collisions can occur. On an Ethernet network, for example, a collision can occur if two systems attempt to communicate at the same time. Obviously the higher the number of collisions, the worse a network will perform.

Consider Figure 2-1, which shows a bridge separating two collision domains. Computers A and B are susceptible to having their data collide with each other. However, this will not affect Computer C, since frames between A and B will not be forwarded to other segments by the bridge. Computers A and B exist in one collision domain, while C exists in another. When a network is segmented into a number of smaller collision domains, performance can increase dramatically.

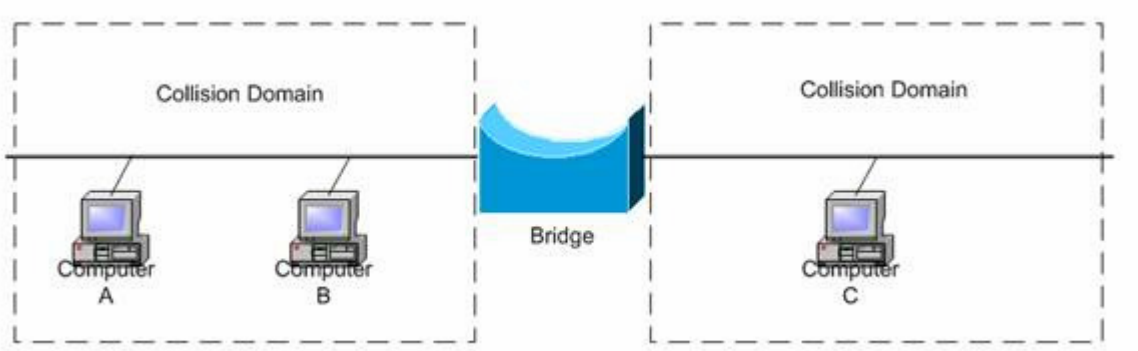


Figure 2-1: Collision domain diagram.

A number of different types of bridges exist, in order to meet different network requirements:

- **Transparent Bridge**. This is by far the most common type of bridge. It is referred to as transparent because it simply inspects frames for the purpose of making forwarding decisions, but doesn't change them. When started, a transparent bridge learns about the computers on its connected segments (and builds its MAC table) by inspecting the source MAC addresses of systems as frames are sent.
- **Translational Bridge**. This type of bridge is used to connect segments that use different network technologies, such as Ethernet and Token Ring. A translational bridge will not only forward frames as necessary, but will also reframe packets for the underlying network when moving between segments using different technologies. On most networks today, however, segments with dissimilar technologies are usually connected using routers rather than bridges.
- **Remote Bridge**. In cases where large geographic distances separate LANs, a dial-up or wireless link might be used to connect segments. This can be accomplished with remote

bridges, which use the connection as a type of extension between LAN segments. Remember that because the device is only a bridge, broadcasts and multicasts will still be forwarded across this (relatively) slow link, and may adversely affect performance. Again, routers are more commonly used to connect remote locations.

Switch

When you think of a switch, simply consider it to be a bridge with more ports. Their higher port density helps to make switches a practical and powerful performance replacement for hubs, though more expensive. Much like a bridge, ports on a switch define different collision domains. In this way, a network can be microsegmented into many very small collision domains, especially if each device is connected to its own dedicated port. If multiple systems are connected to a switch port via a hub, the hub-connected systems exist in the same collision domain. Note that while the switch helps to create a number of smaller collision domains, broadcasts and multicasts are still forwarded to all ports. The process by which a switch or bridge forwards broadcast or multicast traffic to all ports is sometimes referred to as “flooding”.

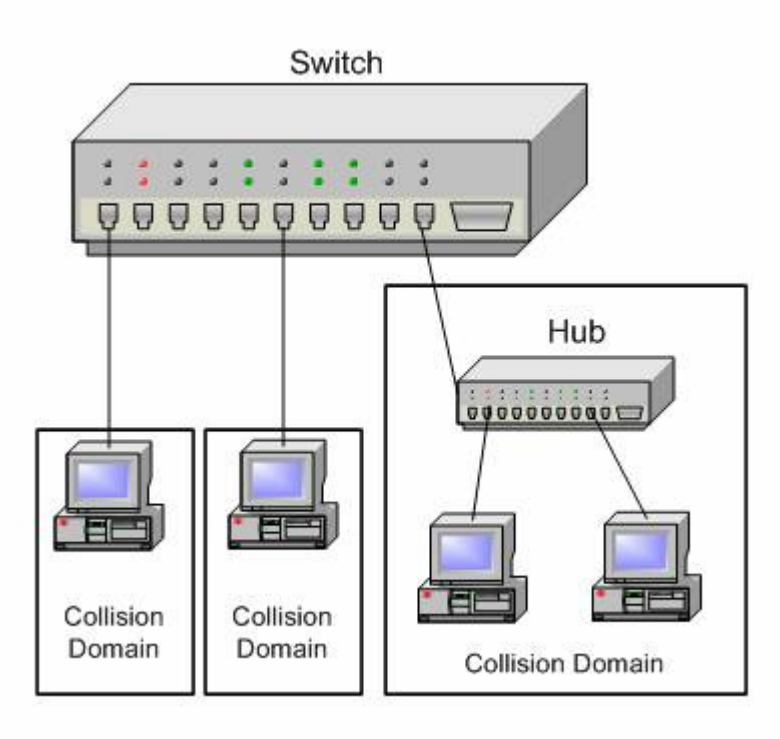


Figure 2-2: Switch collision domains.

When a frame enters a switch, the switch looks up the destination hardware address in its MAC table, and will only forward the frame to the port where the destination MAC address exists. If the switch doesn’t yet know about the destination address (perhaps because a system was just recently turned on), the switch will forward the frame to all ports, a concept referred to as flooding. Switching is usually handled by hardware referred to as Application Specific Integrated Circuits (ASICs). These special chips allow switching to take place at what is sometimes referred to as wire-speed. This offers significantly faster performance than a bridge, which usually stores its forwarding logic in software. Much like a bridge, a switch will also calculate the CRC on a frame to be sure it isn’t corrupt, though different configurations are possible. We’ll look at different switching methods in Chapter 3.

EXAM TIP: Remember that a bridge or switch segments a network into a greater number of smaller collision domains.

Switching significantly increases performance on a LAN, and replacing hubs with switches should be a primary consideration when attempting to improve network performance. In fact, if every device is connected to its own switch port, collisions will not occur, since every device will be in its own collision domain. The absence of collisions gives you the ability to make use of 100% of the available bandwidth. Consider Figure 2-3. In it, users are connected directly to their own 10 Mbps switch port. The server is connected to a 100 Mbps port. In this scenario, each user has access to a full 10 Mbps of bandwidth to the server, collision free.

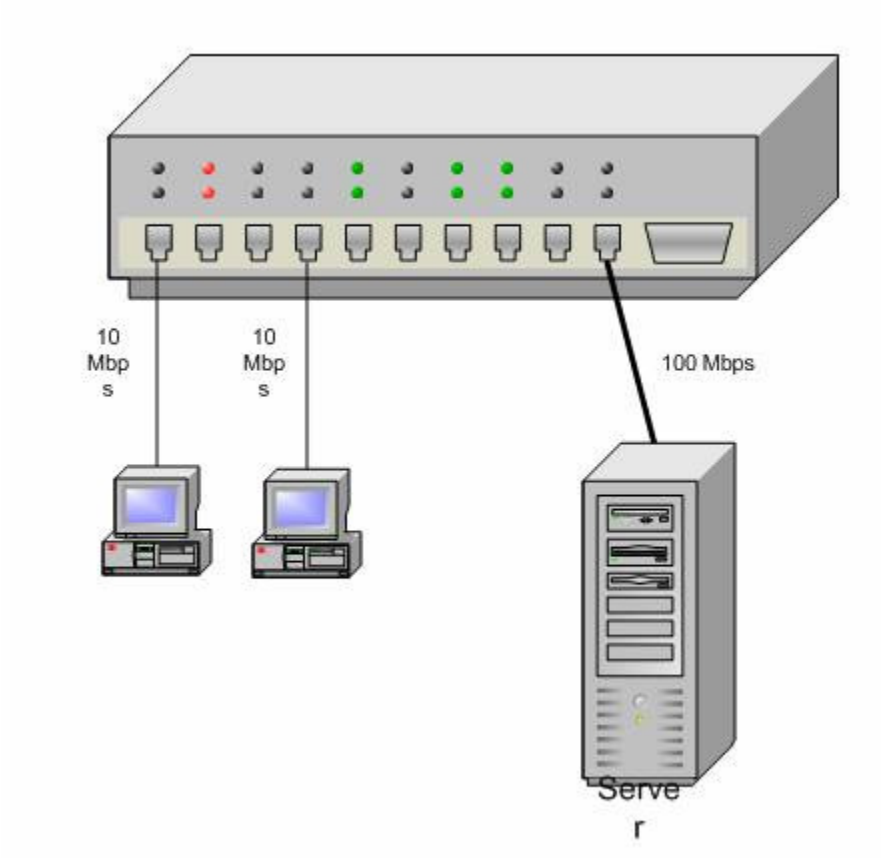


Figure 2-3: Switch with 10 and 100 Mbps ports.

You may have noticed that switches are often described according to an OSI layer - for example Layer 2 or Layer 3. A Layer 2 switch performs switching based on MAC addresses, as previously described. A Layer 3 switch does this as well, but also includes integrated routing functionality. Layer 3 switching concepts are looked at in detail in Chapter 8.

EXAM TIP: An Ethernet switch functions in a manner similar to a transparent bridge.

Router

The main function of a router is to determine the path that packets should take in attempting to get from one network to another. A router will always have at least two physical interfaces, but

depending on the model may have many more. For example, a router may have one Ethernet, one Token Ring, and one serial interface - each of which connects to a different network. By default, a router will know about the networks on which it has a configured interface, and will be able to forward packets between them. When a router doesn't connect directly to a network, it needs to know where to forward a packet next – this would be another router, referred to as the next hop. Consider the very basic routing table shown in Table 2-2.

Table 2-2: Basic routing table.

Network	Interface / Next Hop
10.0.0.0	E0
172.16.0.0	T0
192.168.0.0	S0
11.0.0.0	10.0.0.2

In Table 2-2, we can see the following:

- Network 10.0.0.0 is connected to the Ethernet interface
- Network 172.16.0.0 is connected to the Token Ring interface
- Network 192.168.0.0 is connected to the Serial interface
- Network 11.0.0.0 can be reached by sending packets to IP address 10.0.0.2, the next router in the path to that network.

This doesn't mean that network 11.0.0.0 is directly connected to the next router – in fact, a packet may be forwarded to many other routers on its journey. The only thing this particular router knows is that the next place to send the packet is the address 10.0.0.2 – once there, it's up to that router to figure out where the packet gets forwarded to next.

Note that since it exists at Layer 3, a router is concerned with network (or logical) addressing. As such, a router doesn't forward packets based on MAC address, but instead on the addressing of the routed protocol in use – for example IP, IPX, or AppleTalk. When a router receives a packet, it inspects the destination network address, and forwards the packet according to information found in its routing table. Ultimately the packet should reach the router that is connected to the destination network, where it will be forwarded to the destination host.

The actual communication process that takes place when two systems communicate over a network with routers is a bit more complex. Consider Figure 2-4, where two hosts communicate through a single router.

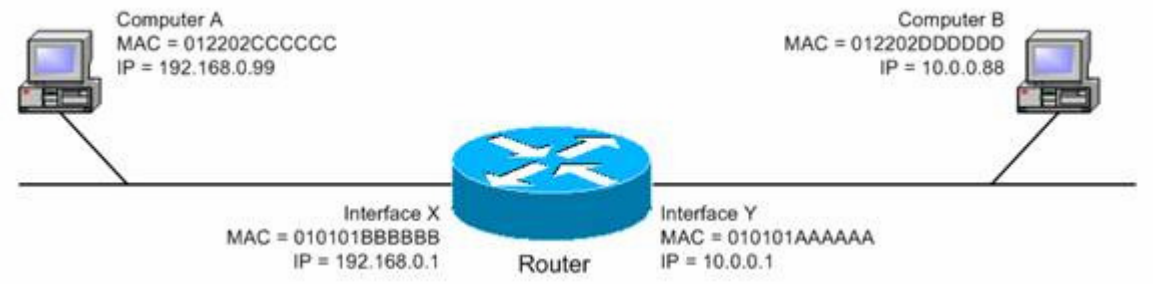


Figure 2-4: Hosts communicating via router.

In the example, Computer A needs to send a packet to Computer B using TCP/IP. Notice that Router 1 is connected to the two networks on which these hosts reside. The following steps outline the basic process that will take place in getting the packet from A to B.

1. Computer A will first determine that Computer B is on a different network, based on the destination address. At the Network layer, it will add source and destination IP addresses to the packet. In this case, the source is the IP address of A, while the destination is the IP address of B.
2. Given that Computer B is on a remote network, Computer A will need to forward the packet to the router, its configured default gateway. Before it can send the data to the router, it still needs to frame the packet. As such, source and destination MAC addresses need to be added. In this case, the source MAC address will be that of Computer A, while the destination address will be that of interface X on the router.
3. Once the frame has reached the router, it will strip away the MAC addressing and pass the packet up to the network layer. At this point, the router will determine the route to the destination network using its routing table.
4. Before sending the packet out interface Y, it will need to be reframed. The new source and destination MAC addresses must still be added. In this case, the source MAC address will now be that of router interface Y, while the destination MAC address will be that of Computer B.

Note that the source and destination IP addresses never changed in the example above. Striping away and recreating the MAC framing would have happened at each router had there been many in the path between Computers A and B. The process of route determination and reframing at each router is what makes routing approximately 30-40% slower than switching.

On a small network, you might manually define the next hop to each network in a router's routing table. However, as an internetwork grows, this gets much more complex and also does nothing to account for network failures. On a large network, you'll want routers to communicate with each other using routing protocols. Routing protocols allow routers to communicate with each other by dynamically exchanging information about the networks that they know about. In this way, routers 'learn' from one another. Examples of routing protocols include Routing Information Protocol (RIP) and Open Shortest Path First (OSPF). Routing protocols are particularly helpful when a network error occurs. If a network path becomes unavailable (say due to a router failure), a router will find out about it via routing protocol updates (or lack of) and will switch to using a redundant path if one exists. The details of how routing protocols function will be looked at in Chapter 8.

Routers are best known for providing multiple paths to different networks and connecting networks that use different technologies. However, they are also capable of controlling network traffic using features such as access lists. Access lists allow you to control which hosts can communicate with different networks, as well as the types of traffic that can move between networks. Access lists will be looked at in detail in Chapter 9.

Another important responsibility of a router is to act as a demarcation point between broadcast domains. By default routers do not pass broadcasts, thus acting as a barrier that stops broadcasts from overwhelming an internetwork. What is a broadcast domain? A broadcast domain defines the group of computers to which a broadcast will travel. In Figure 2-5, a broadcast sent by Computer A will go to all computers in its broadcast domain. Computer B, who is part of a completely separate broadcast domain, will never see it. Notice that the router in the diagram is what separates the broadcast domains.

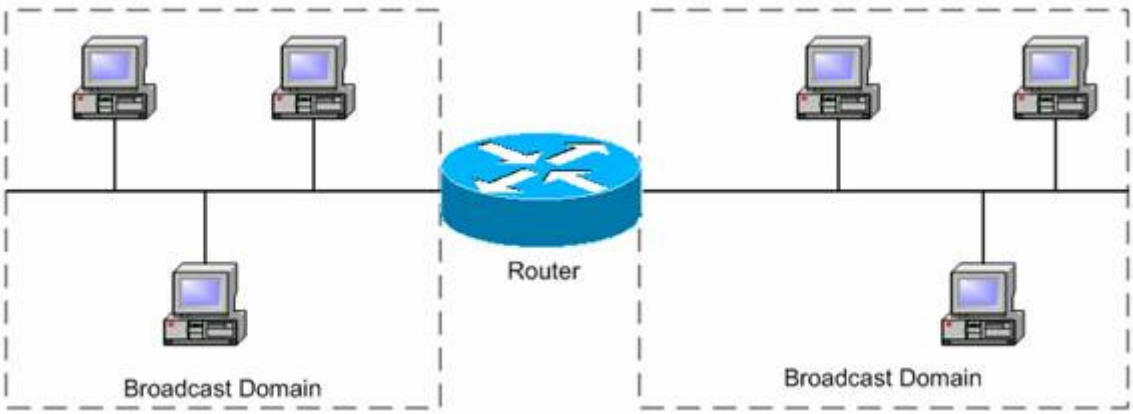


Figure 2-5: Broadcast domains.

EXAM TIP: Remember that a router acts as a boundary between broadcast domains.

Network Transmissions

When traffic is passed between hosts on a network, three different transmission mechanisms are possible. These include unicasts, multicasts, and broadcasts.

Unicasts

A unicast is the most simple network transmission. As the name suggests, it is a direct transmission from one system to one other system only. As such, the destination address will always uniquely identify a single host for whom the data is meant. In a shared Ethernet environment (where a system might be exposed to all frames), systems would check to see whether the destination MAC address matched their own. If it did, it would process the frame. If not, it would discard the frame. On an IP-based network, the address 192.168.1.24 represents a unicast address.

Multicasts

Unlike unicasts, which are meant for a single host, a multicast is meant for a group of systems. Think of multicasts as a one-to-many transmission method. Multicasts are generally used when traffic such as video needs to be passed to many hosts at the same time. In this way, a sender would transmit a single stream of data, which would in turn be picked up by many different hosts. On IP networks, a special group of addresses is reserved for multicasting, those in the Class D range. When multiple hosts need to receive a multicast, they are all configured with an identical multicast IP address. When they receive traffic destined for this shared address, they process it. Do not confuse a multicast address with a regular IP address. In this example, all systems still have a unique IP address, but also “listen in” on a configured multicast address.

Broadcasts

The final type of network transmission is a broadcast. Quite simply, a broadcast is a transmission destined for all hosts. A special destination address designates a broadcast – in Ethernet, the

broadcast address is FF-FF-FF-FF-FF-FF. When a host sees frames with this destination MAC address, it knows it has to process the frames. While excessive broadcasts on a network are generally undesirable, many network services depend on this type of transmission.

Network Cabling

While network cabling may not be the most exciting topic in the world, you'll definitely need to understand the basics of different LAN media. The three main types of media found on LANs include coaxial cable, twisted pair, and fiber optics. In this section we'll look not only at the properties of each but also maximum distances, signaling, and how cables are wired.

Coaxial Cable

It really wasn't all that long ago that coaxial cabling was the defacto standard for wiring LANs. On Ethernet LANs that used coaxial cable, individual computers were connected to the cable using devices such as BNC-T connectors or external transceivers. These connected to the system's network card, and then to a run of cable that went to computers in either direction. At either end of this long segment, you were required to use a terminator that would absorb signals and prevent them from bouncing back down the wire and creating collisions.

Coaxial networks had a major downside. Since every system was connected to this same run of cable, a break or disruption at any point could bring down the network. Even as other wiring cabling standards became popular, coaxial cable was often used as backbone cabling between LANs. Its ability to span longer distances made it useful, even though the speeds at which data could be passed were fairly limiting by today's standards.

While it's not commonly used anymore, it is still important to know something about the two most popular types of coaxial cabling:

ThickNet. This type of coaxial cabling is used with Ethernet 10Base5 networks and is able to span distances of up to 500 meters. Originally used to directly connect computers, it eventually became popular in backbone implementations between LANs. Systems connected to the cable using an external transceiver unit that actually tapped directly into the wire. The transceiver was then connected to a network card using an Attachment Unit Interface (AUI) cable. You'll learn more about external transceivers and AUI connections when we look at Cisco router ports.

ThinNet. A much thinner and more flexible type of coaxial cable, ThinNet is used on Ethernet 10Base2 networks and can span distances of up to 185 meters. This was usually the media of choice for connecting computers on a LAN. In ThinNet networks, computers connect to the network via a BNC-T connector attached to the network card.

Unshielded Twisted Pair (UTP)

By far the most common type of cabling that you'll come across today, unshielded twisted pair (UTP) cabling originates it the world of voice, being the same type of wiring used for telephone connections. UTP provides a number of advantages over coaxial cable, the main being the flexibility it provides in wiring a network. Instead of having to connect devices along a single length of cable, individual systems can be connected to switches or hubs using a patch cable with RJ45 connectors. UTP is most commonly used to wire Ethernet networks.

UTP cables are made up of pairs of copper wires twisted together. The twisting serves an important purpose – it helps to eliminate electromagnetic interference (EMI). EMI is a common problem on networks using copper wire. Signals from one wire pair might interfere with another (referred to as crosstalk), while powerful external electrical devices may also impact transmission capabilities. When using UTP cables, a common mistake is to unravel the twisting too far – this will certainly degrade signal strength and make the wires more prone to interference.

The category of the cabling defines how many wire pairs you'll find in a given cable. Voice grade cable, also known as Category (or simply 'Cat') 3, uses only two pairs and is used for telephone service and 10Mb Ethernet. Cat 5 wiring, on the other hand, uses 4 wire pairs and is the minimum required for 100Mb Fast Ethernet. For the most part, buildings today are usually pre-wired for Cat 5, although Cat 3 may still be found in older environments. You may also come across what is known as Cat 5E – this version of Cat 5 simply has more twists per inch of wiring, providing better resistance to EMI and higher transmission capabilities.

Creating Ethernet Cables

Although you may think it unlikely, I can guarantee that you'll find plenty of network support people in the world who don't know how to wire a simple patch cable properly. Understanding the roles of the wires in twisted pair cabling goes a long way towards explaining the communication process between connected systems. What we're going to look at here is the way in which different network cables are created using Cat 5.

Even though there are 8 wires (4 pairs) in Cat 5 wiring, only 2 of those pairs are used to transmit and receive data. You may have heard the term 'tip and ring', especially in the world of telephony. The four wires that transmit and receive data are considered to be the tip, while the other wires provide grounding and are referred as the ring. When using Cat 5 cabling, there are two possible transmitting and receiving channels, allowing systems to communicate in full duplex if plugged into a switch. Full-duplex means that systems can both send and receive data at the same time – note that many older network cards will only support half-duplex, while newer cards almost certainly will support both. Full- and half-duplex will be looked at in detail when we discuss Ethernet.

Like just about anything else in the networking world, there are standards for wiring UTP cables. These standards are defined by the Electronic Industries Alliance/Telecommunications Industry Association (EIA/TIA) and fall into two flavors – 568A and 568B. For the most part, the B standard is more popular. However, many government contracts require that the A standard be used. To be honest, it's well worth knowing both, as will become clear in a moment. When creating network cables, you'll notice that they are made up of individual wires of different colors. Some appear as a solid color, while others appear white with a colored stripe. The proper terminology is to call the orange wire 'orange' while calling the white wire with the orange stripe 'white-orange' – the background color should always be specified first.

A straight-through cable has two ends that are identical. When looking at cables, compare the RJ-45 ends side-by-side with the snap-in clip facing down. If you're looking at a straight-through cable the ends will be the same, while on a crossover cable, the ends will be different. To be even more precise, a crossover cable simply has one end wired according to the A standard, and the other wired to the B standard – that's why knowing both standards is so useful.

Table 2-3 outlines the wiring order for 568A and 568B. I've included both the pin numbers and the colors to help you along. Note that the pin numbers represent the individual wires from left to right, when looking at an RJ-45 connector with the clip facing down. The leftmost will always be pin 1, and the rightmost pin 8.

NOTE: Some vendors engage in the bad habit of not wiring their pre-packaged cables to the 568A and 568B standard. It's important to recognize that the wire numbering is most important during the communication process, not the colors. However, for the Cisco exams you should be familiar with the wire color and numbering standards listed here.

Table 2-3: 568A and 568B Wiring table.

568A	568B
1. White-Green	1. White-Orange
2. Green	2. Orange
3. White-Orange	3. White-Green
4. Blue	4. Blue
5. White-Blue	5. White-Blue
6. Orange	6. Green
7. White Brown	7. White-Brown
8. Brown	8. Brown

If you look closely, you'll notice that only the orange and green pairs switch places between the standards.

To understand what the pins do, let's take a look at a straight-through cable. In the example below, note that the cable is plugged into a PC at one end and a hub at the other. Note that the pin-outs on the PC and hub have different roles, and that only 4 wires are actually used in the communication process – pins 1, 2, 3 and 6:

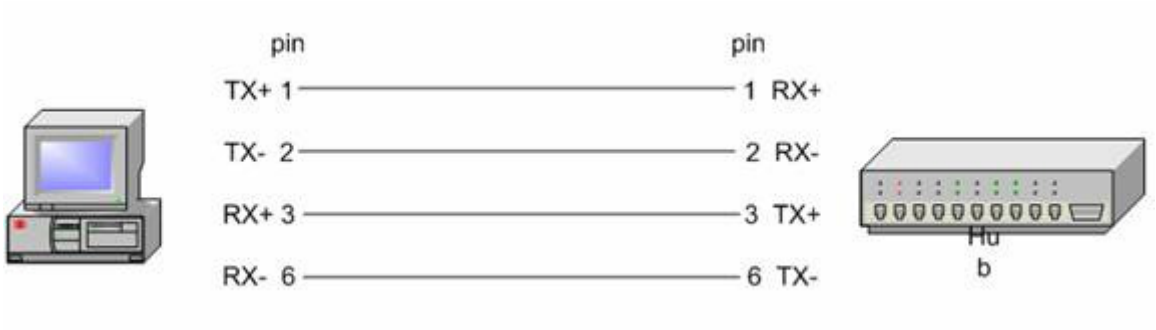


Figure 2-6: Straight-through connection - PC to hub.

So how do you know when to use a straight-through cable and when to use a crossover? That's simple. You always use a straight cable when you're connecting ports of different types. When you're connecting devices of the same type, you always need a crossover cable. The table below outlines different scenarios and the cables required. Hub and switch ports are wired in the same way, as are PC and router ports.

Table 2-4: Cable types used to connect devices.

Connection	Cable Required
PC to Hub	Straight
PC to Switch	Straight
Router to Switch	Straight
Router to PC	Crossover

Hub to Hub	Crossover
Hub to Switch	Crossover
Switch to Switch	Crossover

Note from Figure 2-7 that the crossover cable actually ensures that the transmit pins from one system connect to the receive pins of the other device when two devices have identical ports.

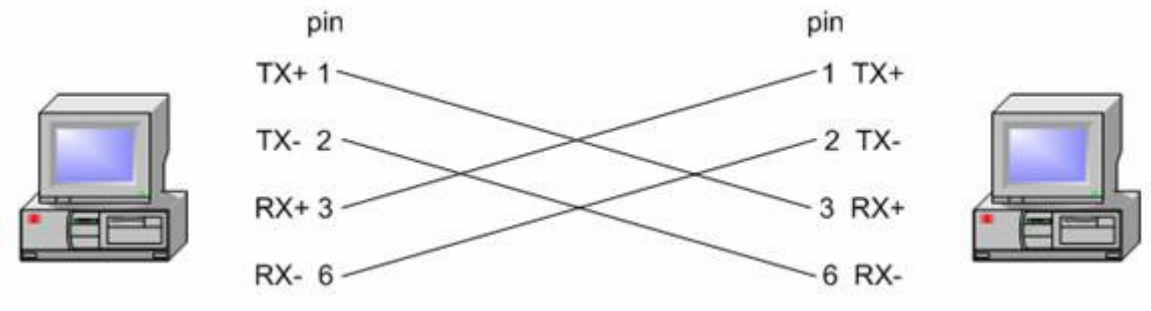


Figure 2-7: Crossover connection - PC to PC.

There is one exception that you should be aware of. Many hubs include what is known as an uplink port, which is usually marked with an “X”. The uplink port provides a crossover function internally. Because of this, you can connect two hubs together with a straight cable, as long as one end is connected to an uplink port.

Shielded Twisted Pair (STP)

Shielded Twisted Pair (STP) cabling is very similar to UTP, with one key difference. STP incorporates an additional conductive foil shielding around each pair of wires. While this helps further cut down on EMI, STP is more expensive and can also be thicker and harder to install than UTP cables. STP cabling is most often used on Token Ring networks.

Fiber Optic Cabling

Fiber optic cabling is another option for network media, though considerably more expensive. This is not only in terms of the cost of fiber optic cables, but also related components such as network cards and switching equipment. Unlike UTP and STP, which run over copper wires, fiber optic cabling instead sends pulses of light through a pure glass core encased within Kevlar sheathing. The fact that light isn’t susceptible to EMI provides great advantages, especially in places with a high degree of possible interference such as factories or warehouses. Fiber optic cabling also allows data to be transmitted along much greater distances than traditional copper wire, in some cases up to hundreds of kilometers. Using fiber does have some disadvantages outside just cost - it’s significantly more difficult to install, and subject to various bending limitations. It is worth noting that fiber cables will have two separately encased fiber stands, one used for transmitting and the other used for receiving. In other words, the strand connected to the transmit port on one device will be connected to the receive port on the other device.

Different types of connectors can be used to connect to fiber optic ports. The most common are round plug-style ST connectors and square block-style SC connectors. It’s important that you make sure that you’ve purchased the correct cables to match the ports on your particular equipment.

The two main types of fiber cabling used on networks are single mode and multimode. Single mode fiber can span much greater distances and carries a single ray of light up to a number of kilometers. There are a number of factors involved in the distances that fiber optic cabling can span, though it mainly relates to the micron diameter of the glass core. To be clear, a micron is one millionth of a meter. The most common diameter for fiber optic cabling is 62.5 microns. As the diameter on the cable decreases, the distance that can be spanned increases. Single mode fiber is most commonly used on networks that employ long-wavelength optics (LX).

Multimode fiber can also be used in long-wavelength optics, but is more commonly used with short-wavelength (SX) technologies. Multimode fiber carries many different light signals at once, each at different angles of refraction. It can only travel shorter distances, up to approximately 550 meters.

Fiber optic cabling is becoming popular on LANs, especially for use with Gigabit Ethernet. While fiber optic connections to the desktop aren't common, you'll often see them used for backbone or trunk connections between switches, and frequently for connections to servers as well.

Media Access Methods

There are a variety of methods by which data is merged onto a network, a concept referred to as the media access method. The media access method used depends on the way in which a particular technology such as Ethernet or Token Ring communicates. This section will look at the three most popular methods – contention-based, token passing, and polling.

Contention

Contention-based media access describes a way of getting data on to the network whereby systems 'contend for' or share the media. On a contention-based network, systems can only transmit when the media is free and clear of signals. In this way, devices listen to the media, and if no other system is transmitting, they can go ahead and send data. In cases where more than one system finds the network free and attempts to transmit, a data collision will occur, and systems will need to retransmit. On busy networks, the number of collisions can quickly get very high, adversely affecting performance. Remember that in this scenario, only a single system truly has access to the media at any given time, even though multiple systems may have data to send.

The best example of a contention-based network technology is Ethernet, which uses a scheme called Carrier Sense Multiple Access with Collision Detection (CSMA/CD). The fact that Ethernet is contention-based is a reason why many people thought that the technology would never be a good solution for large networks. As time passed, different techniques were developed to provide a way for contention-based networks to scale to larger sizes. A great example is the use of switches to segment a network, thus significantly reducing (or even eliminating) collisions.

Token Passing

A more orderly scheme for moving data between network systems is found when token passing is used. In token-passing media access environments, a special frame referred to as a token repeatedly circles the network, passed from system to system. If a system has control of the token, it can transmit data. If it doesn't, it must wait for the token to become available again.

While this might sound like a very slow way to go about passing data, it's important to understand that the token moves around the network at incredibly high speeds. Understand also that

because this method isn't contention based, there won't be any collisions, further increasing performance

Examples of technologies that use token-passing media access include Token Ring and Fiber Distributed Data Interface (FDDI), both of which will be described in detail later in this chapter.

Polling

While contention and token-passing methods are by far the most popular ways in which PCs access LAN media, some technologies rely on a technique called polling. Polling is a deterministic way of allowing systems access to the network while also avoiding collisions. When used, a device referred to as the master polls systems to see if they have data to transmit. In this way, polling is similar to token passing, except that the central device controls the order in which systems are contacted. The downside of polling is that when the master device fails, the network fails. Most popular in mainframe and minicomputer environments, polling is a technique used in protocols such as Synchronous Data Link Control (SDLC).

Network Topologies

Network topologies describe both the physical and logical layouts of a network. A particular technology such as Token Ring might lead you to believe that the network is physically connected in one big circle of cable. In reality, the ring is formed in circuitry and the physical network appears as a star. Common topologies that you should be familiar with include Bus, Ring, Star, Hybrid, and Mesh.

Bus Topology

A bus topology is by far the most simple. A bus is comprised of a single run of cable to which individual systems are attached. When a failure occurs on a bus network, it affects the entire network since the path for data transfer is disrupted. While you may still find Ethernet bus networks in smaller environments, they are becoming less and less common. In general, bus topologies are relatively inexpensive but tend to be more prone to failure.

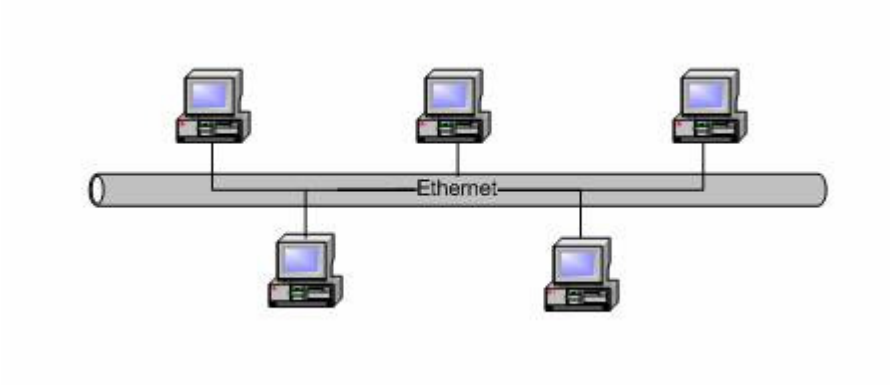


Figure 2-8: Bus Network.

Ring Topology

As the name suggests, a ring topology is comprised of a number of systems connected in a type of loop. In most cases, systems connect to a hub-type device within which the actual ring is formed. For this reason, you'll often see ring topologies described as being a star-ring, where data is passed through the network along the ring circuitry, but the physical layout actually appears to be a star. Logical ring / physical star layouts are commonly found in both Token Ring and FDDI environments.

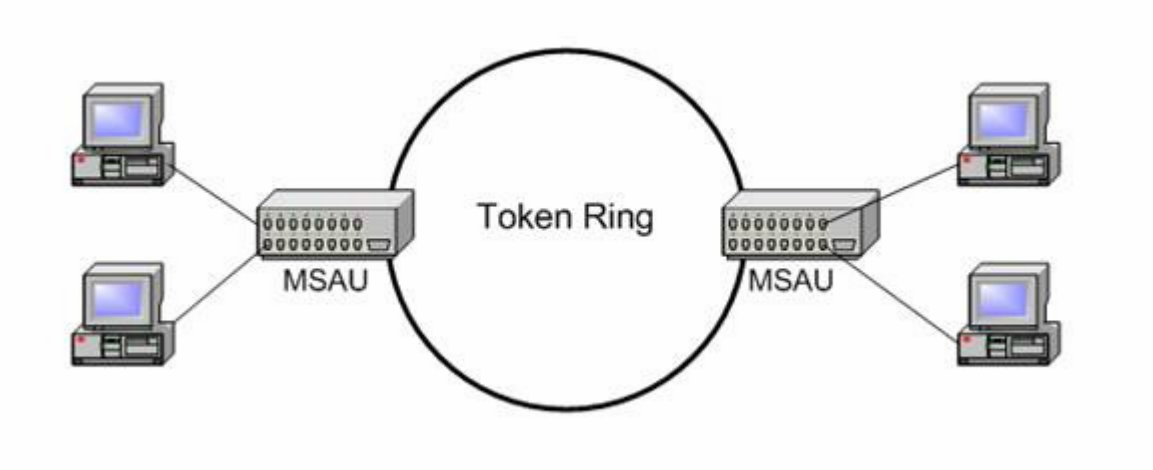


Figure 2-9: Ring topology with MSAUs.

Star Topology

By far the most common network topology found today, a physical star is created when systems connect to a central device such as a hub or switch. Systems branch out from this central device, creating the star-like appearance. The main benefit of this topology is the fact that a break in a cable only affects the particular connected system and not all others. Recognize, however, that a single point of failure still exists – if the hub or switch fails, all connected systems will not be able to communicate.

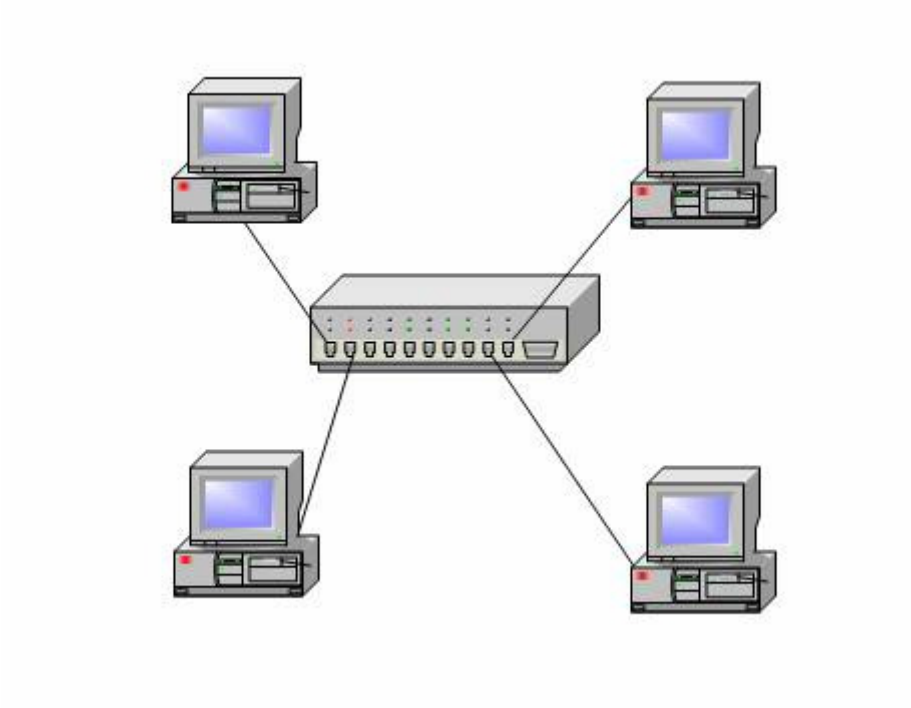


Figure 2-10: Star topology.

Hybrid Topology

Most large networks designed today tend to be variations of star topologies. However, many networks will be comprised of a number of different topologies rather than just one. For example, a company's network might be a star-bus hybrid, where systems connect to hubs (forming the star) and then hubs interconnect using a bus. Various hybrids are possible, including stars, rings, and buses. Figure 2-11 shows a simple star-bus hybrid topology.

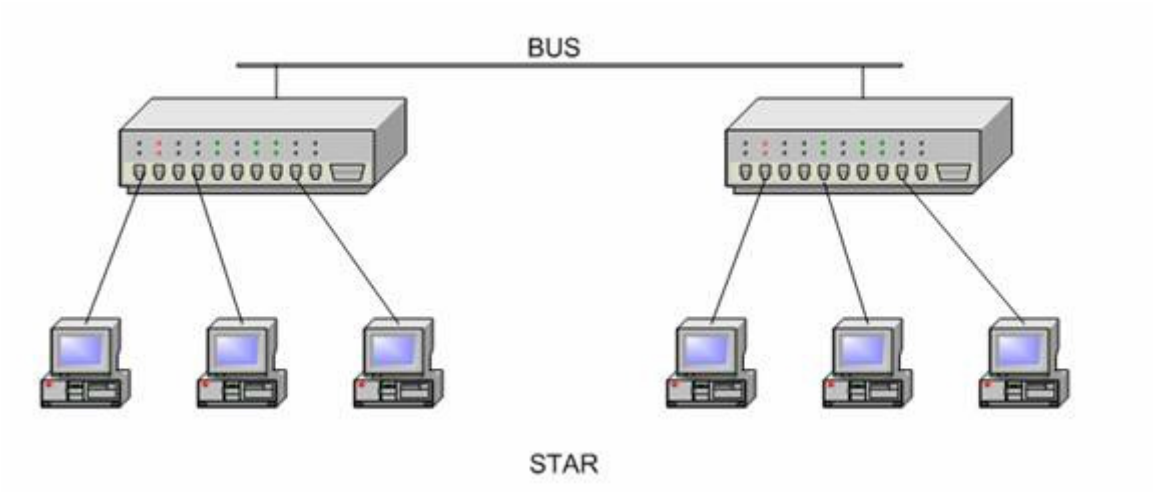


Figure 2-11: Hybrid Topology.

Mesh Topology

Wide Area Networks (WANs) are often configured in a mesh topology for the purpose of redundancy. In a mesh, a router or switch may have more than one connection to a different site. In this way, if one link fails, another path exists. Because of this, mesh topologies are also much more expensive, and seldom found between computers on LANs (although certain systems requiring high-availability may be configured in this way). A mesh in which every system has a connection to every other system is referred to as a full mesh. If fewer connections exist, it is simply known as a partial mesh. A good example of a technology that is often configured in a mesh is Frame Relay.

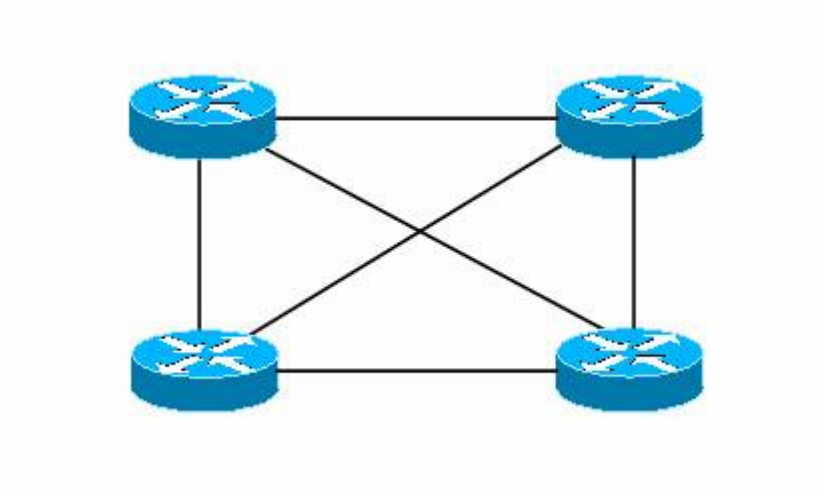


Figure 2-12: Full Mesh Topology.

LAN Technologies (Part 1): Ethernet

The term LAN technology refers to the Data Link and Physical layer technologies (or protocols) in use on a particular Local Area Network. In this section we'll take a look at the media access methods, addressing, frame types, media and performance issues associated with the three most popular LAN technologies – Ethernet, Token Ring, and FDDI.

Ethernet

Originally developed by Xerox in the 1970's, Ethernet has become the defacto technology standard for LANs today. Digital, Intel, and Xerox (DIX) standardized Ethernet in 1980, with the IEEE version finalized in 1982 with the 802.3 standard. Over time Ethernet has undergone a number of changes, both with respect to how devices are connected, and the ways in which data is framed. A solid understanding of Ethernet concepts is imperative to your success on the CCNA and CCDA exams.

Media Access - CSMA/CD

The media access method used by Ethernet is the contention-based Carrier Sense Multiple Access with Collision Detection (CSMA/CD). This name not only defines the technology, but also describes how it works. "Carrier Sense" means that different devices are listening to the media for

the opportunity to transmit. “Multiple Access” describes the media as being contention-based, in that it is shared amongst many computers. “Collision Detection” is an Ethernet feature whereby systems are capable of recognizing when a collision has occurred.

When a system uses collision detection techniques, there must be a way to try and avoid the same collision from happening repeatedly. CSMA/CD handles this by having systems back off for a random period of time after a collision occurs. If these collisions continue the system back off time will increase, considerably decreasing performance. Retransmission will be attempted up to 16 times before an error message will be passed to the upper-layer protocol in use. You may be familiar with some of the distance limitations imposed on Ethernet networks (we’ll look at those shortly). Understand, however, that the limitations exist not only because of signal attenuation, but also because as distances increase, the ability of CSMA/CD to properly detect collisions decreases. This is especially true when systems at opposite ends of a network attempt to communicate at the same time, sensing the media as available.

NOTE: *Remember that CSMA/CD is the media access method used on Ethernet networks. As such, hosts on a traditional Ethernet network “share” the media, making their transmissions susceptible to collisions. Equipment like switches and bridges help to reduce network collisions, and will be looked at in more detail in Chapter 3.*

Ethernet Addressing

All Ethernet network adapter cards are uniquely identified by a pre-assigned hardware (or MAC) address. A MAC address is a 48-bit address represented in hexadecimal format. The first 24 bits represent what is known as the Organizationally Unique Identifier (OUI), and represents a vendor code. The last 24 bits are assigned by the vendor and act as the unique identifier (and serial number) for a particular network card.

TIP: *Remember that the first 24 bits of a MAC address identify a manufacturer like Cisco, and the last 24 bits represent the unique serial number of a card. For a complete list of manufacturer OUI codes, see http://standards.ieee.org/regauth/oui/oui.txt.*

Hexadecimal is a numbering system that uses the numbers 0-9 and the letters A-F, where each hex digit represents 4 bits. As such, a MAC address will always be made up of 12 hex digits, in a format similar to 01-22-33-44-55-EF. Table 2-5 outlines the decimal value associates with each hexadecimal digit. Note that the highest valid hex digit is F – anything above that is not a valid character for a MAC address.

Table 2-5: Hexadecimal to decimal conversions.

Hex	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Dec	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

STUDY TIP: *You can easily convert between decimal and hexadecimal by using the scientific calculator included with Microsoft Windows operating systems. A great online tool for converting numbers between decimal, hexadecimal, and binary easily can be found at http://www.onlineconversion.com/base.htm.*

NOTE: *During both the CCNA and CCDA exams, you will not have access to any type of calculator. As such, it is extremely important for you to be able to convert between decimal, hexadecimal, and binary numbers on your own.*

Ethernet Frame Types

Four different frame types exist in the world of Ethernet, mainly a result of different implementations created for different purposes at different times. Ethernet frame types include Ethernet 802.3, Ethernet 802.2, Ethernet II, and Ethernet SNAP. When two systems need to communicate on an Ethernet network, they must be using a common frame format. The confusion as to when a given Ethernet frame type is used is generally a result of different vendors and standards bodies moving in different directions. So how do you know which frame type will be used on a given network? That depends. Sometimes you can configure the interface to use a certain frame type with a certain upper-layer protocol (such as IPX). In other cases, a vendor or organization will choose the frame type to be used with a particular upper level protocol. For example, TCP/IP will always attempt to use Ethernet II, as defined by the Internet Engineering Task Force (IETF). For their Cisco Discovery Protocol (CDP), Cisco uses Ethernet SNAP framing. For the most part, network equipment will be able to handle multiple frame types on a single interface. In all cases, the minimum Ethernet frame size is 64 bytes, while the maximum size is 1518 bytes. Anything smaller than 64 bytes is invalid and referred to as a "runt", while frames over 1518 bytes are also invalid and considered "giants".

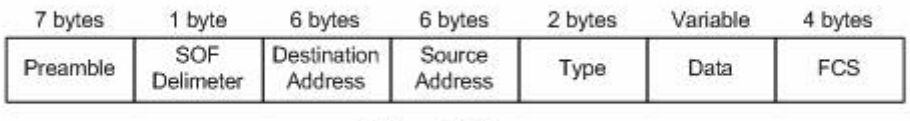
What's the difference between the various frame types? Different frame types may include header fields that were created to address different technical challenges. In all Ethernet frame types you'll find five main elements – a preamble, start of frame delimiter, header, data, and trailer. Each of these is described below:

- **Preamble**. The purpose of the 7-byte preamble is to mark the beginning of a frame and to enable synchronization between a sender and receiver.
- **Start of Frame Delimiter**. The 1-byte SOF field always ends in binary 11 to notify that the next bits represent the beginning of the destination MAC address.
- **Header**. At a minimum, the header will contain the source and destination MAC addresses (6 bytes each), as well as an extra 2-byte field. Various frame types use this extra field differently, as we'll discuss shortly.
- **Data**. The data portion houses everything that was encapsulated by the upper-layer protocols prior to being passed down for framing.
- **Trailer**. An Ethernet trailer consists of a Frame Check Sequence (FCS). This is where the Cyclic Redundancy Check (CRC) value is held that will be used to confirm that the frame has not been corrupted when it reaches its destination.

Now that we know the elements that are common to every frame, you need to be able to recognize the differences between the four main types.

Ethernet II

The Ethernet II frame type is by far the most simple. Those extra 2 bytes in the Ethernet header described earlier are used for a Type field in Ethernet II. The Type field simply identifies the upper layer protocol to which data should be passed. For example, a Type field of hex 0800 represents IP, while 8137 means that data is meant for IPX.



Ethernet II Frame

Figure 2-13: Ethernet II frame.

Ethernet 802.3

The Ethernet 802.3 frame was originally created by Novell for use with the IPX protocol, and was later standardized by the IEEE. Because these frames don't contain any LLC information, they are sometimes referred to as Ethernet RAW. These frames contain a 2-byte Length field instead of a Type field – they automatically assume that the upper-layer protocol is IPX, and do not work with other upper layer protocols.

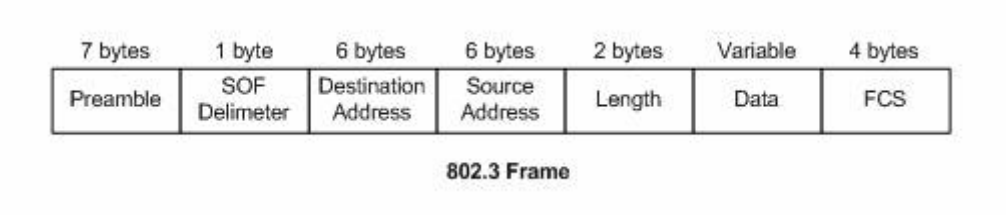


Figure 2-14: Ethernet 802.3 frame.

Ethernet 802.2 (SAP)

In order to provide a greater deal of flexibility with Ethernet framing, the IEEE defined what is known as the 802.2 Logical Link Control (LLC), the upper sub-layer of the Data Link Layer. At first glance an 802.2 frame may look like an 802.3 frame, since it has a length field. However, the first part of the data portion of an 802.2 Ethernet frame actually contains LLC information in the form of Source Service Access Point (SSAP), Destination Service Access Point (DSAP), and Control information.

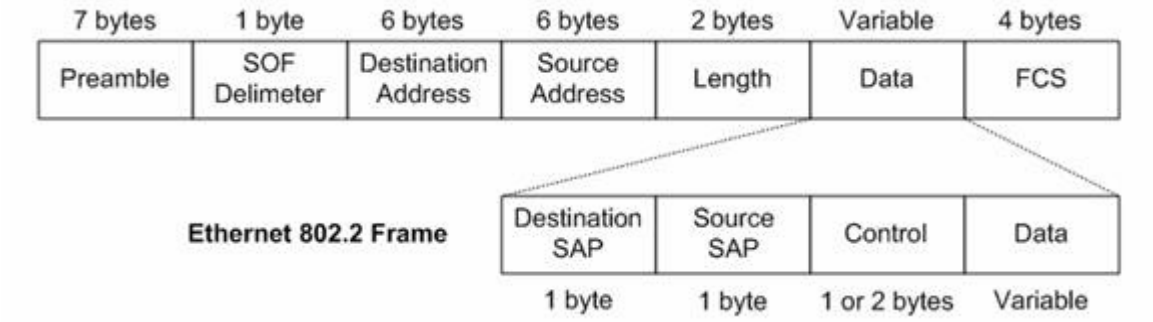


Figure 2-15: Ethernet 802.2 frame.

As you'll recall, the Logical Link Control SSAP and DSAP fields are used by the LLC sub-layer to interact with Network layer protocols. Examples of SAP codes include F0 for NetBIOS, 06 for IP, and E0 for IPX – again, all codes are represented in hexadecimal.

Ethernet SNAP

The final Ethernet frame type, Ethernet SNAP (which stands for Sub Network Access Protocol) was developed as a result of compatibility issues. Given that many vendors had been using the Ethernet II frame types for their upper layer protocols before 802.2 was standardized, they were

left with a 1-byte SAP field where they had previously used a 2-byte Type field. This made moving to the new standard difficult, so the IEEE came up with the Ethernet SNAP frame type. Ethernet SNAP allows a higher degree of flexibility for proprietary protocols. The Ethernet SNAP frame type is commonly used with AppleTalk.

If you look at an Ethernet SNAP frame, you'll notice that the SSAP and DSAP fields are always set to AA. These codes identify it as a SNAP frame. The command field always has a value of three, which specifies connectionless LLC service. Following this is the Organizationally Unique Identifier (OUI) field, which is used to define the organization that created the upper layer protocol. The Type field provides the same function as the Type field in an Ethernet II frame. When all is said and done, the SNAP information uses up 5 extra bytes of the data portion of an Ethernet frame, as shown in Figure 2-16.

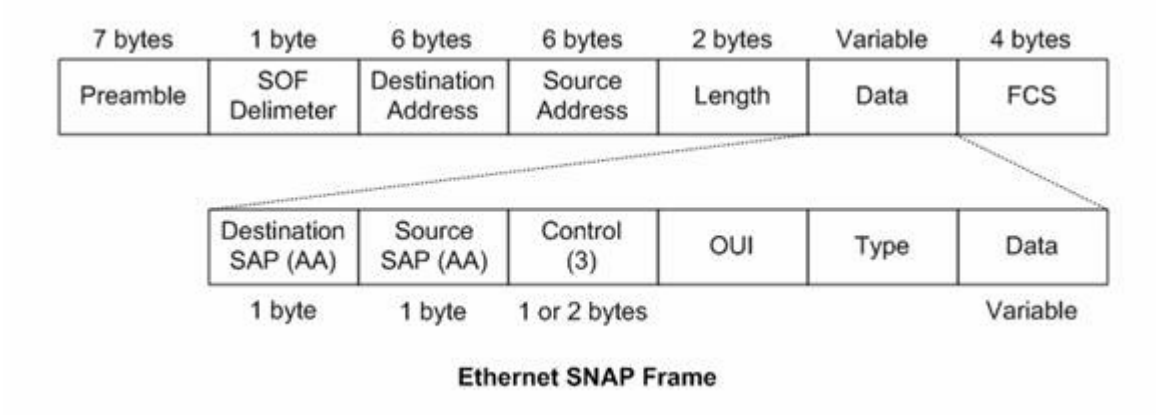


Figure 2-16: Ethernet SNAP diagram.

EXAM TIP: Remember that the Ethernet SNAP frame type can easily be identified by the AA value in both the SSAP and DSAP fields.

Physical Ethernet Standards

Up to this point we've mainly been looking at the Data Link layer elements of Ethernet. However, Ethernet standards also define Physical layer characteristics such as cable distances, media types, and just about anything to do with physical connectivity, including connectors. You may already be familiar with some of the different varieties of Ethernet. They're usually represented in a format such as 10BaseT or similar. Understanding the designations is critical, so we'll look at these first.

When you see Ethernet defined as 10BaseT, you're actually being provided with 3 pieces of information. In this example:

- "**10**" defines the maximum speed of transmission in Megabits per second.
- "**Base**" specifies that baseband transmission is used. Baseband transmission provides a single channel for digital transmission. In contrast, broadband transmission is analog and separates the cable into different frequency ranges or channels.
- "**T**" defines that this type of Ethernet runs over twisted pair wiring. On a 10BaseT network, the minimum cable standard is Category 3.

A variety of different Ethernet standards exist, a cross section of which are outlined below.

- **10Base2**. 10Mbps Ethernet that runs over ThinNet coaxial cable. Maximum segment length of 185 meters and a maximum of 30 connected systems per segment.
- **10Base5**. 10Mbps Ethernet that runs over ThickNet coaxial cable. Maximum segment length of 500 meters and a maximum of 100 connected nodes per segment.
- **10BaseF**. 10Mbps Ethernet that runs over fiber optic cabling for distances up to 2 kilometers in full duplex.
- **100BaseTX**. Fast Ethernet (100Mbps) that runs over Cat5 twisted pair wiring. Maximum cable length is 100 meters.
- **100BaseFX**. Fast Ethernet that runs over fiber optic cabling.
- **1000BaseT**. Gigabit Ethernet (1000Mbps) that runs over Cat5 twisted pair wiring. Maximum cable length is 100 meters.
- **1000BaseLX**. Long wave Gigabit Ethernet over fiber. If using multimode fiber, maximum distance is 550 meters. If single mode fiber, maximum distance of approximately 5 kilometers.
- **1000BaseSX**. Short wave Gigabit Ethernet over fiber. Uses multimode fiber to span distances up to 550 meters.

Ethernet also makes use of features at the Physical layer by auto-negotiating elements such as link speed and duplex type when a network card is plugged into a switch or hub. Originally defined in the IEEE 802.3u specification (Fast Ethernet), this is accomplished using something called Fast Link Pulses (FLPs), which are sent between the system and the connected port. For example, you may have a network card that supports both 10 and 100 Mbps speeds. However, if the hub only supports 10 Mbps, they will negotiate the connection to the common setting (in this case 10 Mbps). The same is true for negotiation of the duplex type used. When using half duplex, a system can be either sending or receiving data, but not both concurrently. In full duplex, systems can send at receive at the same time.

Note that when plugged into a hub, systems will always communicate using half duplex, since they share the media and only one system can communicate at any given time. When a system is plugged directly into a switch port, full duplex becomes possible. To that end, it is worth noting that when you connect a hub to a switch, all computers plugged into that hub will automatically use half-duplex, since they'll again be part of the same collision domain.

Ethernet Performance

Ethernet networks tend to be susceptible to performance problems as they grow, based on the CSMA/CD method of media access that they use. While implementing Layer 2 switching goes a long way towards better Ethernet performance, there are still a number of issues to consider when an Ethernet network begins to experience performance problems. Examples of reasons for congestion on Ethernet networks include having too many hosts on a given segment, not enough bandwidth available, broadcast storms, along with excessive broadcast or multicast traffic.

A few key metrics are used by Cisco to help decide when an Ethernet network is not performing at an appropriate level. These include:

- **Network utilization**. A network utilization of over 40% on shared Ethernet segments represents that the network is saturated.
- **Broadcasts/multicasts**. No Ethernet segment should have more than 20% combined broadcast and multicast traffic.
- **CRC Errors**. There should be less than 1 CRC error per MB of network traffic.
- **Collisions**. Less than 0.1% of network packets should be involved in collisions

While this data may seem difficult to obtain, a variety of network management tools can provide these particular metrics, along with many others. Examples include CiscoWorks and Cisco Netsys Performance Service Manager. These tools and others will be explored in more detail in Chapter 13.

LAN Technologies (Part 2): Token Ring

Token Ring was developed by IBM in the 1970's and was later the basis for the IEEE 802.5 specification. Over time Token Ring's popularity has declined considerably, in no small part due to advances in Ethernet technology. While the CCNA exam primarily concentrates on Ethernet, an awareness and basic understanding of Token Ring is still important for both the CCNA and CCDA exams.

Media Access – Token Passing

As described earlier, Token Ring systems access network media using a token passing method. The token is nothing more than a basic 3-byte frame that continuously circles the network when no data is being sent, forwarded from system to system. When one of these systems wishes to transmit, it seizes the token and produces a data frame, which it sends to the destination system. The destination system alters the frame to mark it as received, and then forwards it back to the sender. Once the sender verifies that the data reached its destination, it releases the token back onto the network. At any point in the process, a station can mark the reservation field in a data frame to reserve the new token once it is generated.

Token Ring offers a variety of features and benefits to with respect to reliability, fault management, and priority:

- **Priority**. Token Ring frames include a field that permits certain designated stations to gain access to the token more frequently. A system with an equal or higher priority than the system currently transmitting can choose to reserve (or seize) the token. Once the higher priority system obtains the token and has finished transmitting, it will set the priority back to normal, allowing other systems a chance to transmit.
- **Reliability**. Token ring networks are deterministic, meaning that the time between possible transmissions and delays is predictable. This can be an important consideration in environments where applications require regular access to the media (for example, voice and video).
- **Fault Management**. Token Ring includes 2 primary fault management techniques - monitoring and beaconing. One system on a Token Ring network is designated as the active monitor and watches the ring for issues like frames that continuously circle the network (which may happen if a sending system fails), removing them as necessary. Beaconing is a process by which a special "beacon" frame will be sent when a system encounters a critical network failure, attempting to reconfigure the network automatically to avoid the error.

Addressing

Token Ring addresses are universally administered by the IEEE in a manner similar to Ethernet addresses. They are also 48-bit addresses represented in hexadecimal, where the first 24 bits define a vendor, while the last 24 represent a unique identifier and serial number.

It is also possible to manually define the last 24 bits of a Token Ring MAC address using what is referred to as a Locally Administered Address (LAA). While possible, this involves additional administrative effort and isn't commonly used.

Framing

There are two primary frame types found on Token Ring networks – a token frame and a data frame. A token frame is the very simple 3-byte frame that is continuously passed between systems until one is ready to transmit. A token frame is made up of the fields shown in Figure 2-17.

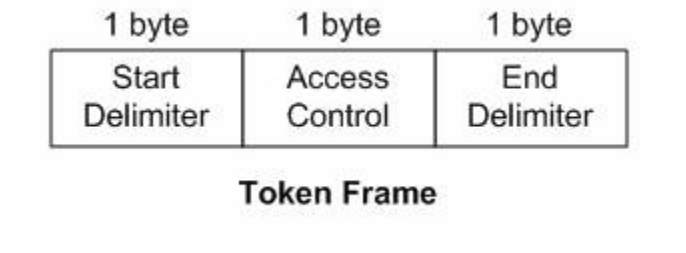


Figure 2-17: Token Frame.

Start Delimiter. This field is used to signal that a frame is being received.

b>Access Control. This field is used to provide information relating to token priority levels and reservation information, as well as to signify whether the frame is a token or contains data.

End Delimiter. This field specifies the end of a frame, and also identifies the last frame in a sequence.

Data frames are used for the purpose of transferring information, and can range in size from 54 to 4472 bytes. Larger sizes are possible, though generally not used on traditional Token Ring networks. A Token Ring data frame is shown in Figure 2-18, followed by an explanation of the various fields. Note that the Start Delimiter, Access Control and End Delimiter fields all serve the same purpose as those in a token frame, and are not explained again below.

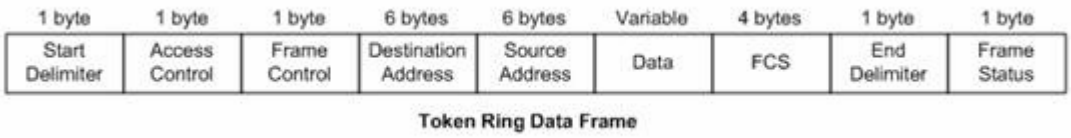


Figure 2-18: Token Ring Data Frame.

Frame Control. This field specifies whether the frame contains data meant for a particular end station, or control information that should be read by all stations.

Destination Address. Contains the MAC address of the receiving station.

Source Address. Contains the MAC address of the sending station.

Data. The actual data encapsulated by upper-layer protocols.

Frame Check Sequence. Contains the CRC value used to check whether frames have been corrupted in transit.

Token Ring Equipment and Standards

As describe earlier, a Token Ring network is logically connected as a ring, but physically connected as a star. Two main versions of Token Ring exist, one running at 4 Mbps and the other at 16 Mbps, primarily using Shielded Twisted Pair (STP) cabling (though UTP and fiber optics are also possible). End stations plug into hub-type devices known as Multi Station Access Units (MSAUs). The role of these devices is to provide systems with a connection point to the network, while providing access to the ring.

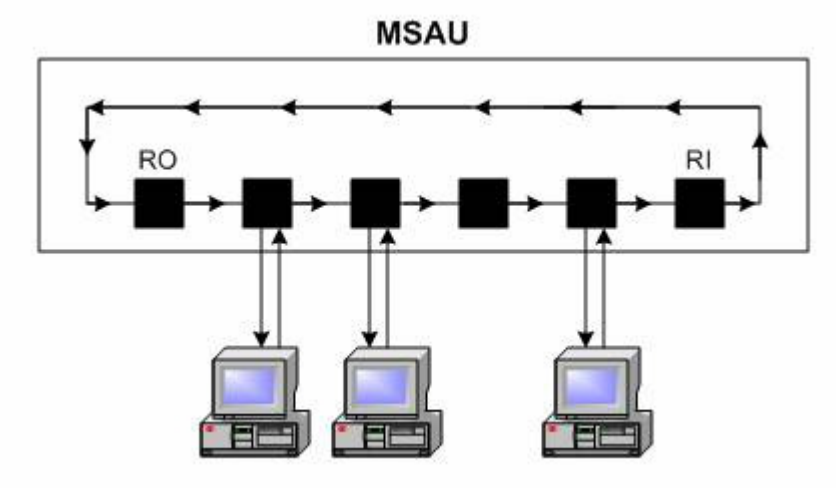


Figure 2-19: Multi Station Access Unit.

An MSAU not only provides connections for individual computers, but also allows interconnection with other MSAUs. It does this via two extra ports known as Ring In (RI) and Ring Out (RO). The RO port on one MSAU is connected to the RI port on others until the complete ring is defined. Figure 2-20 outlines a Token Ring network with two MSAUs.

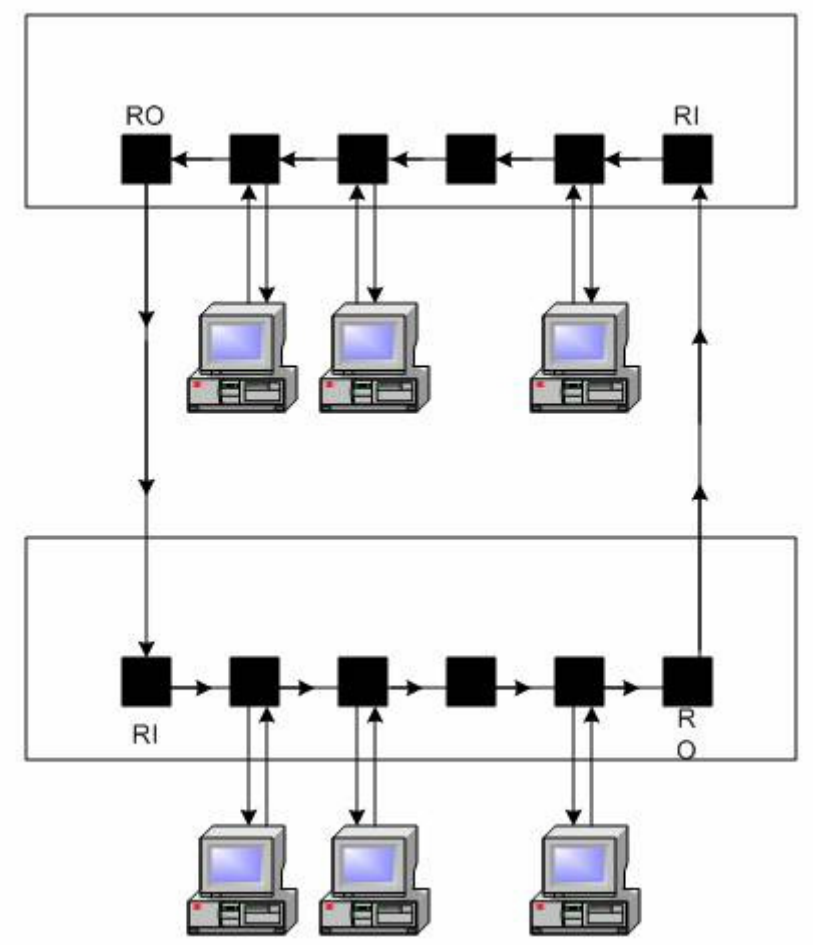


Figure 2-20: Token Ring network with MSAUs.

Token Ring Performance

A few key metrics are used by Cisco to help decide when a Token Ring network is not performing at an appropriate level. These include:

Network utilization. A network utilization of over 70% on shared Token Ring segments represents that the network is saturated.

Broadcasts/multicasts. No Token Ring segment should have more than 20% combined broadcast and multicast traffic.

CRC Errors. There should be less than 1 CRC error per MB of traffic.

Errors. Less than 0.1% of frames should be from errors not related to ring insertion. Ring insertion refers to a device being added to the ring.

LAN Technologies (Part 3): Fiber Distributed Data Interface (FDDI) and Chapter Summary

FDDI is a set of LAN standards developed in the 1980s; it is recognized by the ISO and is governed by the ANSI X3T9.5 standards committee. FDDI is not actually a single standard, but a collection of four standards that will be defined shortly. A FDDI network consists of a 100 Mbps dual-ring topology that runs over fiber optic cabling (copper is possible over shorter distances, and is referred to as CDDI). Because of the long distances that FDDI networks can span, it is often used for the purpose of creating and connecting a Metropolitan Area Network (MAN). The IEEE has defined MANs in their 802.6 standard. A basic knowledge of FDDI is only relevant to the CCDA exam.

FDDI is comprised of four standards specifications that exist at the Physical and Data Link layers of the OSI model. These include:

- **Physical Layer Protocol (PHY)**. Defines FDDI data encoding, clocking and framing.
- **Media Access Control (MAC)**. Defines frame formatting, token management, CRC calculations, and addressing functions.
- **Physical Medium Dependent (PMD)**. Defines elements of the physical media in use including bit rates, connectors, and power levels.
- **Station Management (SMT)**. Defines station configuration (including ring insertion and removal) as well as network management and fault tolerance features.

Media Access – Token Passing

Much like Token Ring, FDDI also employs token passing as its way to get data onto the network. When a station has the token it can transmit data, and once it receives the original data transmission sent, it releases the token onto the network again. The maximum data frame size on a FDDI network is 4500 bytes.

Depending on the type of fiber optic cabling in use, FDDI networks can span distances of 2-30 kilometers between devices. The maximum number of attached stations on a FDDI network is 500.

The equipment found on a FDDI network is unique when compared to Token Ring or Ethernet. Firstly, FDDI utilizes two rings, each of which passes data in a different direction (referred to as counter-rotating rings). The purpose of the two rings is to provide fault tolerance. When operating normally, one ring is active, and the other on standby. In cases where a cable breaks, the ring is wrapped on both sides of the failure to allow continued operation as shown in Figure 2-21.

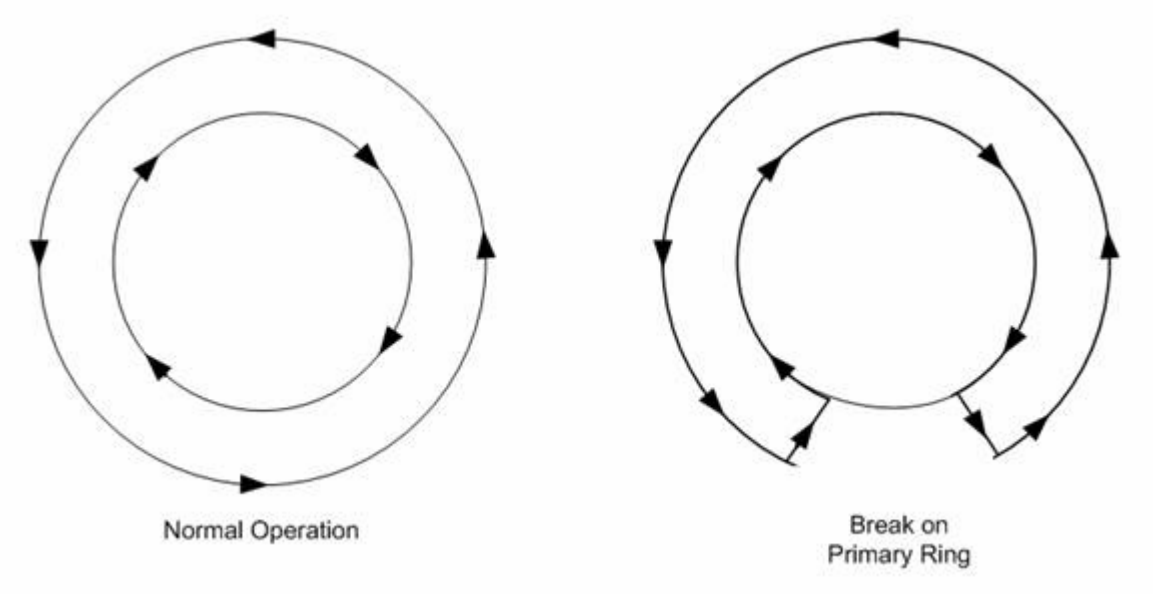


Figure 2-21: FDDI dual rings.

FDDI Equipment

Devices connect to FDDI networks in four different ways, depending on whether they are end stations or concentrators, and are described according to their attachments. A dual attached device will have 2 ports, one marked A and the other marked B. Equipment is cabled such that the B port from one station connects into the A port of another on the primary ring, while A attaches to B on the standby or secondary ring. The four FDDI device types are listed below:

- **Single-attachment station (SAS)**. An SAS attaches only to the primary ring through a concentrator (hub-type device). This allows the station to be powered down or disconnected without affecting the network.
- **Dual-attachment station (DAS)**. A DAS attaches to both the primary and backup rings. Because they connect directly to both rings powering down and/or removing these devices will impact the network.
- **Single-attachment concentrator (SAC)**. A SAC connects to only the primary ring.
- **Dual-attachment concentrator (DAC)**. A DAC connects to both rings, acting as a device into which single-attachment stations can be connected.

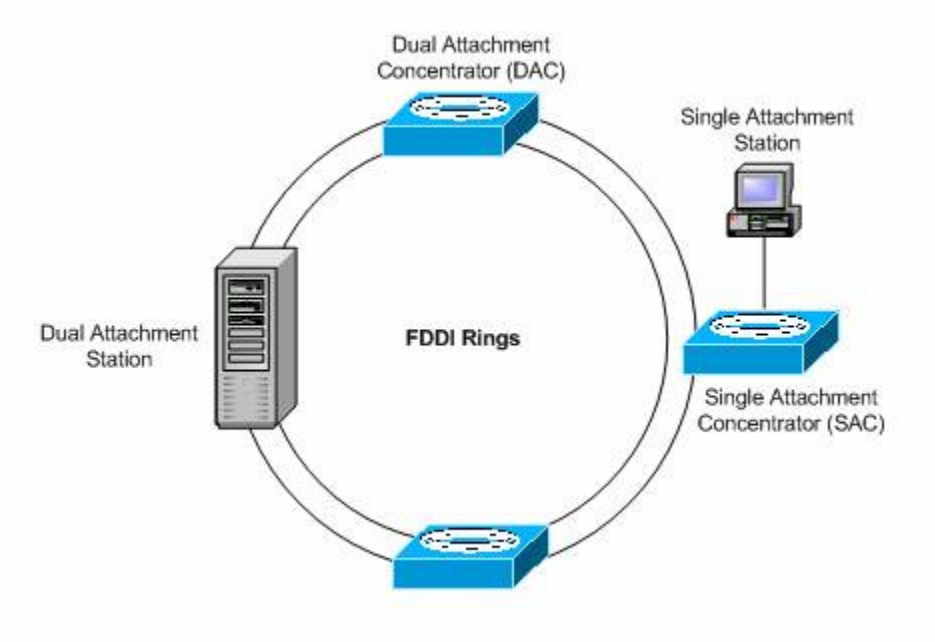


Figure 2-22: FDDI diagram with attached devices.

FDDI Framing

A FDDI data frame is comprised of nine different fields, including a preamble that marks the beginning of a frame. FDDI actually makes use of an encoding scheme that translates groups of 4 bits into 5-bit symbols, referred to as 4B/5B. These symbols are used to represent control, error, and data information.

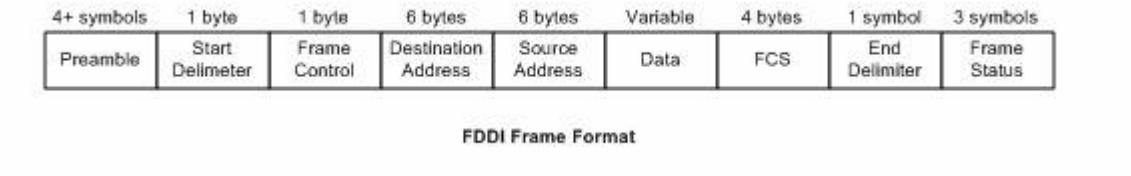


Figure 2-23: FDDI Frame format.

- **Start Delimiter**. Indicates the beginning of a frame.
- **Frame Control**. Used to determine whether the frame is a token or data frame.
- **Destination Address**. The destination MAC address for the frame.
- **Source Address**. The source MAC address for the frame.
- **Data**. The data encapsulated by upper-layer protocols.
- **Frame Check Sequence**. Contains the CRC value for the frame.
- **End Delimiter**. Used to indicate the end of a frame.
- **Frame Status**. Field used by the source station to ensure that the data was received at the destination.

FDDI Performance

Performance requirements for FDDI are in line with those listed in the token ring section. However, it should also be noted that Cisco considers anything more than 1 ring operation per hour (not related to ring insertion) to adversely affect performance.

Chapter Summary

We began this chapter with a look at the difference between Local and Wide Area Networks. This was followed by a look at the different types of equipment commonly found on networks including repeaters, hubs, bridges, switches and routers. Our look at equipment focused not only on their functions but also their relationship to the OSI model. We also covered how the equipment related to the concepts of broadcast and collision domains.

A look at network transmission methods provided a perspective on the three different mechanisms used to transmit data – unicasts, multicasts and broadcasts. This was followed by an overview of network cabling that included information on cable properties and maximum lengths. We also looked at the wiring of crossover and straight UTP cables to connect both similar and different port types.

A look at media access methods provided an overview of contention, token passing and polling, outlining the relative advantages and disadvantages of each in different environments.

Finally, a look at LAN technologies was used to gain an understanding of how data is passed using Ethernet, Token Ring and FDDI. This included a look at addressing, frame types, physical characteristics, and performance metrics.

Review Exercises

1. For each description provided on the left, enter the appropriate network equipment, technology, or term on the right.

a) A router exists at this layer of the OSI model.	
b) A switch segments a network into a many of these.	
c) Routers act as the demarcation point between these.	
d) This type of UTP cable should be used when connecting a router to a switch port.	
e) This type of network transmissions goes from one host to only one other host.	
f) This type of network cabling is used with Ethernet 10Base5.	
g) Ethernet segments are considered saturated at what percentage utilization?	
h) This is the number of bits present in an Ethernet MAC address.	
i) This type of UTP cable should be used to connect a switch to a hub.	
j) A router makes forwarding decisions based on addressing at this OSI layer.	

k) This is the media access method used by Ethernet.	
l) Hexadecimal digits can only include this range of valid characters.	
m) This is the minimum size of an Ethernet frame.	
n) Switches and bridges make forwarding decisions based on this specific address.	
o) A switch or bridge will build its MAC address table by looking at this specific address in a frame.	
p) This type of network cabling is not susceptible to electromagnetic interference (EMI).	
q) Both Token ring and FDDI use this media access method.	
r) This term is used to describe a FDDI station that connects to both rings.	
s) A Token ring station that detects a critical network failure will send out this type of frame.	
t) A Token ring network is considered saturated at this percentage of combined broadcast and multicast traffic.	
u) This is the length of a Token ring token frame.	
v) Token ring is often referred to using this IEEE specification.	
w) FDDI is often used in Metropolitan Area Networks, which are known by this IEEE specification.	

Answers to these review exercises can be found on the next page.

Answers to Chapter 2 Exercises
Answers to Question 1:
a) Network Layer
b) Collision domains
c) Broadcast domains
d) Straight-through cable
e) Unicast
f) ThickNet
g) 40%
h) 48
i) Crossover cable
j) Network Layer
k) Carrier Sense Multiple Access with Collision Detection (CSMA/CD)
l) 0 through F
m) 64 bytes
n) Destination MAC address
o) Source MAC address
p) Fiber optic cabling
q) Token passing

r) Dual-Attachment Station (DAS)
s) Beacon frame
t) 20%
u) 3 bytes
v) 802.5
w) 802.6

Review Questions (Multiple Choice)

1. Which of the following statements best describes a hub?

A. All connected systems are in the same broadcast domain, but different collision domains
B. All connected systems are in the same collision domain, but different broadcast domains.
C. All connected systems are in the same broadcast and collision domains.
D. All connected systems are in their own broadcast and collision domains.

2. At which layer of the OSI model does a switch exist?

A. Physical
B. Data Link
C. Network
D. Session

3. Every port on a switch defines a:

A. Collision domain
B. Broadcast domain
C. Broadcast and collision domain

4. Full duplex Ethernet communication is only possible when:

A. Systems are connected to different LAN segments
B. Systems are connected to a bridged segment
C. Systems are connected to their own switch port
D. Systems are running over a fiber optic connection

5. In the MAC address 01-77-45-E3-T5-99, which portion represents the Organizationally Unique Identifier (OUI)?

A. 01-77-45
B. 01-77-45-E2
C. 01-77-4
D. 01-77-45-E

6. What Class of IP address is reserved for multicast communication?

A. Class A
B. Class B
C. Class C
D. Class D

7. Category 5 UTP is made up of how many wire pairs?

A. 2

B. 3
C. 4
D. 8

8. Which of the following Ethernet standards run over coaxial cabling? (Choose all that apply)

A. 10BaseT
B. 10Base2
C. 1000BaseSX
D. 10Base5

9. Anything over this percentage of multicasts and broadcasts is considered to cause performance issues according to Cisco.

A. 10%
B. 20%
C. 30%
D. 40%

10. True or False. Fiber optic cables are not susceptible to crosstalk.

A. True
B. False

11. What type of cable should be used to connect a router to a switch?

A. Straight-through
B. Crossover

12. A switch builds its MAC table based on which of the follow?

A. Source MAC addresses
B. Destination MAC addresses

13. Which of the following are not examples of dynamic routing protocols? (Choose all that apply)

A. RIP
B. IPX
C. ICMP
D. OSPF

14. True or False. Routers separate broadcast domains.

A. True
B. False

15. A network topology in which every system has a connection to every other system is referred to as a:

A. Hybrid
B. Full Mesh
C. Bus
D. Ring

16. Which of the following is not a valid MAC address?

A. 001034E4FF65
B. 0233BEG77611
C. 0100EAFFFFF1
D. 01AAAA450101

17. This Ethernet frame type is characterized by its use of the code AA in the SAP fields.

A. Ethernet II
B. Ethernet RAW
C. Ethernet 802.2
D. Ethernet SNAP

18. True or False. A shared Ethernet segment running at 30% utilization is considered to be saturated.

A. True
B. False

19. True or False. Single mode fiber allows greater distances to be spanned than multimode fiber

A. True
B. False

20. True or False. STP cabling is less susceptible to EMI than UTP cabling.

A. True
B. False

21. If a frame enters a bridge and the MAC address is not found in the MAC address table, what will the bridge do with the frame?

A. Drop it
B. Forward it to all ports except the port it came in from
C. Hold it until the destination MAC address is discovered
D. Block it

22. True or false. A bridge will always forward all broadcast traffic to all ports.

A. True
B. False

23. Which of the following are examples of routed protocols? (Choose all that apply)

A. IP
B. IPX
C. RIP
D. OSPF
E. AppleTalk

24. What does the destination MAC address FF-FF-FF-FF-FF-FF represent?

A. A multicast destined for a select group of computers on an Ethernet network
B. A broadcast to all hosts in a broadcast domain
C. The MAC address of a particular host.

D. A frame destined for a single collision domain

25. If switches are used to replace hubs on a network, which of the following statements is true?

A. The number of broadcast domains will increase
B. The number of collision domains will increase
C. The number of collision domains will decrease
D. The number of broadcast domains will decrease

26. True or false. Collisions will not occur if all systems are plugged into individual switch ports.

A. True
B. False

27. How many bytes make up an Ethernet or Token Ring MAC address?

A. 4
B. 8
C. 6
D. 48

28. Token ring networks are considered saturated at what percentage utilization?

A. 20%
B. 30%
C. 70%
D. 90%

29. This type of FDDI station is connected to both rings, and acts as a type of hub for other stations.

A. DAS
B. SAS
C. DAC
D. SAC

30. How many repeaters are allowed on a 10Base5 network?

A. 3
B. 2
C. 4
D. 5

31. Which of the following represent FDDI specifications? (Choose all that apply)

A. MAC
B. PHY
C. LLC
D. SAP

32. Which of the following represent benefits of using fiber optics over copper-based LAN media? (Choose all that apply)

A. High bandwidth
B. Susceptible to EMI

C. Ability to span longer distances
D. Lower cost

Answers to Review Questions

1. C. When plugged into a hub, all systems are part of the same broadcast domain and the same collision domain.

2. B. A switch works at the Data Link layer of the OSI model.

3. A. Each port on a switch defines a collision domain.

4. C. Full duplex Ethernet communication is only possible between systems connected to their own switch ports.

5. A. The first 24 bits (or 3 bytes) is the Organizationally Unique Identifier in a MAC address.

6. D. The Class D IP address range is reserved for multicasting.

7. C. Cat5 UTP cable is made up of 4 wire pairs.

8. B and D. Both 10Base2 and 10Base5 run over coaxial cable.

9. B. Cisco recommends that a given segment have no more than 20% combined broadcast and multicast traffic.

10. A. Fiber optic cables are not susceptible to crosstalk.

11. A. Since routers and switch ports are wired differently, you would use a straight-through cable to connect them.

12. A. A switch builds its MAC address table based on source MAC addresses.

13. B and C. While IPX and ICMP can be routed, they are not dynamic routing protocols.

14. A. Routers are used to separate broadcast domains.

15. B. In a full mesh topology, every system has a connection to every other system.

16. B. The letter G is not a valid hexadecimal character.

17. D. An Ethernet SNAP frame uses the code AA in both the SSAP and DSAP fields.

18. B. A shared Ethernet segment is considered to be saturated at 40% utilization.

19. A. Single mode fiber is capable of spanning longer distances than multimode fiber.

20. A. Because of the additional shielding, STP cables are less susceptible to EMI than UTP.

21. B. A bridge will forward (or flood) frames destined to unknown MAC addresses to all ports.

22. A. A bridge will always forward all broadcast traffic to all ports.

23. A, B, and E. IP, IPX and AppleTalk are all examples of routed protocols.

24. B. The destination Ethernet address FF-FF-FF-FF-FF-FF represents a broadcast.

25. B. When switches replace hubs on a network, the number of collision domains increases.

26. A. If all systems are plugged into a switch, the collision domains will consist of only a single device, making collisions impossible.

27. C. A Token Ring or Ethernet address is 48 bits, or 6 bytes.

28. C. Token Ring networks are considered to be saturated at 70% utilization.

29. C. A Dual Attachment Concentrator (DAC) is connected to both rings and acts as a type of hub for Single Attached Stations.

30. C. 10Base5 follows the 5-4-3 rule, which allows 5 connected segments using 4 repeaters, and only 3 populated segments.

31. A and B. MAC and PHY are two of the four FDDI standards.

32. A and C. Fiber optic cabling provides higher possible bandwidth and the ability to span longer distances.

Exam Quick Review

LAN. Usually spans a very limited geographic distance, usually within a single building. LANs are capable of high-speed transfer using technologies such as Ethernet, Token Ring, and FDDI.

WAN. Usually spans a large geographic distance at lower speeds and interconnect LANs by way of connections to service providers. WAN technologies include Frame Relay, ISDN, dial-up, leased lines and more.

Network Equipment
Repeater. Physical layer device used to regenerate signal strength.

Hub. Physical layer device used for connectivity and signal regeneration. All connected systems are part of the same collision domain and broadcast domain. Can use half duplex communication only.

Bridge. Data Link layer device used to segment LANs into smaller collision domains. All systems remain part of the same broadcast domain. They inspect traffic and build a bridging table based on source MAC addresses encountered. Forwarding decisions are based on the destination MAC address of a frame. Broadcasts, multicasts, and frames for unknown hosts are flooded out of all ports.

Switch. Data Link layer device that microsegments a LAN into even smaller collision domains, although all systems are still part of the same broadcast domain. Same functions as a bridge, except with more ports. When systems are plugged directly into a dedicated port, full bandwidth and full duplex become possible. For example, 2 systems plugged into their own 10Mb ports can send and receive data to each other at a full 10Mbps.

Router. Used to make packet-forwarding decisions based on Network layer addressing such as IP, IPX, or AppleTalk. Entries for routes to networks are stored in its routing table. Can exchange information with other routers by using routing protocols such as IGRP, RIP, OSPF, and others. Routers split a network into smaller broadcast domains.

3 Types of network transmission:

- Unicast = one to one
- Multicast = one to many
- Broadcast = one to all

Network Cabling:

- UTP is susceptible to EMI, crosstalk.

- UTP Cat 3 uses 2 wire pairs, often used for telephony. Minimum for 10BaseT. Since only 1 pair is used, a system can only communicate in half duplex.
- UTP Cat 5 uses 4 wire pairs, minimum for 100BaseT, one pair used for sending, the other for receiving in full-duplex environments. Commonly used for Ethernet.
- STP. Extra shielding around each wire pair provides better resistance to EMI. Often used on Token Ring networks.
- Coaxial cable. ThickNet used in 10Base5. ThinNet used in 10Base2.
- Fiber optic cable. Can span much longer distances, not susceptible to EMI, higher possible speeds

Cable connections:

- Hub to hub = Crossover
- Hub to switch = Crossover
- PC to hub or switch = Straight-through
- Router to hub or switch = Straight-through

Media access methods

- **CSMA/CD**. All systems share the media, and can transmit if the line if clear. If two systems attempt to communicate at the same time, a collision occurs, and systems must wait and then retransmit. Used by Ethernet

- **Token Passing**. Deterministic access - a system requires access to the token frame to transmit. No collisions. Used by Token Ring, FDDI.

- **Polling**. A central master device controls access to the media. Popular in mainframe environments, this method is deterministic and collision free.

Network Topologies

- **Bus**. Usually found in coaxial Ethernet environments, all systems were connected to a single long segment of cable.

- **Star**. The most common LAN topology, formed when devices plug into a central device such as a hub or switch.

- **Ring**. A topology where systems connections form a logical ring in circuitry. Usually implemented as a physical star using MSAUs or concentrators.

- **Hybrid**. A combination of topologies, such as a star-ring or star-bus.

- **Mesh**. A topology where systems connect directly to each other. If every system has a connection to every other, it forms a full mesh. If not, a partial mesh.

LAN Technologies

Ethernet

- Uses contention-based CSMA/CD for media access. On shared segments, systems communicate in half-duplex. Full duplex is possible if systems are plugged into individual switch ports directly (NIC must support Full Duplex as well).

- Systems have a unique 48-bit MAC address. First 24 bits represent vendor OUI, last 24 identify a card uniquely. Ethernet broadcast address is FF-FF-FF-FF-FF-FF. Only valid Hexadecimal digits are those between 0 and F

- Minimum frame size of 64 bytes, maximum 1518 bytes.

- Ethernet II frames are used primarily by TCP/IP and have a 2-byte Type field that specifies the upper-layer protocol in use.

- Ethernet 802.3 sometimes referred to as RAW, have a 2-byte length field, and used only by Novell.

- Ethernet 802.2 SAP frames specify SSAPs and DSAPs for the Data Link layer to communicate with upper-layer protocols

- Ethernet SNAP frames are identified by AA in the SAP fields, used mainly with proprietary protocols such as Cisco's CDP.

- Ethernet Standards – usually referred to by their common definition – 10BaseT or similar. 10 represents speed in Mbps. Base refers to baseband communication. T refers to media, twisted pair. 100BaseT is fast Ethernet, requires Cat5 cabling.

- Ethernet systems can autonegotiate duplex and link speed using Fast Link Pulses (FLPs), and will used the highest common standard between host and port.

- Ethernet performance – 40% utilization is considered saturated, there should be no more than 20% combines broadcast/multicast traffic, no more than one CRC error per MB of traffic, and less than 0.1% of packets should be involved in collisions.

Token Ring

- Uses token passing as its media access media. Stations must have the token to transmit. Ability to set stations to a higher priority, granting them ability to transmit more frequently. Reliable and deterministic in terms of transmission rates. Fault management via Active Monitor and beaconing.

- Token ring MAC addresses are also 48-bit, same breakdown as Ethernet. It is possible to set the unique card addressing using Locally Administered Addresses (LAAs).

- Two frame types, token frames and data frames.

- When a station detects a network failure, it will send out a beacon frame and attempt to correct the problem.

- Minimum frame type of 54 bytes, usual maximum of 4472 bytes, with bigger sizes possible.

- MSAUs are used to connect systems to the ring, which is formed internally in circuitry. MSAUs can be connected using Ring In and Ring Out ports.

- Token Ring Performance – 70% utilization is considered saturated, and no more than 20% combines broadcast/multicast traffic. There should be less than 1 CRC error per MB of traffic, and less than 0.1% of frame should be soft errors not related to ring insertions.

FDDI

- Uses fiber optic cabling and a dual-ring topology for redundancy. Primary ring is active, secondary ring on standby. If a network failure occurs, the ring wraps to allow continued operation. Token passing media access. Distances spanned between devices ranges from 2-30 kilometers, depending on media used.

- Four physical layer FFDI standards – PHY, MAC, PMD, SMT

- Four main FDDI device types – Single Attachment Station (SAS) connects to concentrator. Dual Attachment Station (DAS) connects directly to both rings. Single Attachment Concentrator (SAC) connects only to the primary ring, while Dual Attached Concentrator (DAC) connects to both rings. Note that dual attachment devices cannot be powered down with affecting the network.

- FDDI Performance – there should be no more than 1 ring operation per hour not related to ring insertion.

Chapter 3: Layer 2 Switching

Layer 2 Switching

In Chapter 2 we covered a basic overview of bridging and switching functions, including a look at how these devices help to segment a network into a number of smaller collision domains. In this chapter we'll go many steps further, with a look at switching methods, loop avoidance, Virtual LANs, Spanning Tree Protocol, trunk connections, frame tagging, Token Ring switching, and Cisco Catalyst switch models.

For the purpose of the CCNA exam, you'll need to be familiar with the configuration of Cisco Catalyst 1900 series switches, the most basic manageable switches that Cisco offers. These particular switches have two different configuration modes - one menu-driven, the other command line-based. The 1900 comes with one of two software versions - the Enterprise version includes the command line interface (CLI), while the Standard Edition does not. In this chapter we'll stick to the initial configuration of a 1900 using a terminal session and the menu interface. Getting into the command line at this point would be premature, since we haven't examined the Cisco Internetwork Operating System (IOS) yet. For that reason, the 1900 configuration details can be found in Appendix A. By the time we get there, you'll almost be an IOS pro.

The material to be covered in this chapter includes:

- Cisco switching methods
- Redundancy and loop avoidance on bridged and switches networks
- Spanning Tree Protocol
- Virtual LANs (VLANs)
- Trunking and VLAN identification
- Cisco Catalyst 1900 initial configuration
- Layer 2 multicasting techniques
- Cisco switching equipment

Switching Reviewed

By this point, you should be familiar with the basic benefits that a switch provides. You know that a switch segments a network into smaller collision domains, and this helps to provide better performance and throughput. You also have an awareness of how a MAC address table is built, and how a bridge or switch makes forwarding/filtering decisions. Finally, you should recall that when a computer is plugged directly into its own switch port, full duplex communication becomes possible. The ability to communicate without collisions makes full bandwidth available to systems. For example, if two Fast Ethernet network cards are plugged into individual 100 Mbps switch ports, they become capable of simultaneously sending and receiving 100 at Mbps. While some people (including marketers, of course!) will suggest that this means the card is capable of 200 Mbps performance, this is stretching the truth. The card can only transmit or receive in either direction at a maximum of 100 Mbps, even if it can do both at the same time.

While a switch goes a long way towards providing better network performance, you have to remember that at the end of the day, it's still only a Layer 2 device. Because of that, all

broadcasts and multicasts will still be forwarded (or flooded) out all switch ports, along with frames whose destination address isn't yet known to the switch.

Does a switch allow networks to grow larger than when hubs are used? Certainly. However, in order to provide a higher level of performance and cut down on broadcast traffic, a large network will still need to be split into different broadcast domains. This still requires the use of routers – keep this in mind when you're thinking about network switching.

TIP: Remember that a switch or a bridge will create a larger number of smaller collision domains.

Cisco Switching Methods

When we looked at bridging in the previous chapter, I mentioned that a bridge would take the time to calculate the CRC value of every frame that it intended to forward. While this is true for a bridge, on Cisco switches you can configure the different methods in which switching will occur. Certainly a network will be more reliable if every frame has its CRC value calculated by the switch. However, this also introduces a degree of delay (or latency) to the network.

Network equipment today is more reliable than ever before. On a well-planned network using quality equipment, frame corruption shouldn't be a huge issue. Because of this, you might consider eliminating the CRC check on frames to take advantage of higher forwarding rates. This can be done on Cisco switches (including the 1900) by configuring the method by which the switch forwards frames. The three frame forwarding methods supported by Cisco include:

- Store-and-Forward
- Cut-Through
- FragmentFree

Store-and-Forward

Store-and-Forward switching is best described as traditional frame forwarding. When used, the switch will calculate the CRC value for each and every frame. When a frame enters a port, the switch will copy the entire frame onto its buffers and will calculate the CRC. If the CRC calculation shows the frame not to be corrupt, it will be forwarded to the destination port according to the information found in the switch's MAC address table. In situations where the frame is corrupt, it will be dropped. Because Ethernet frames can be different sizes, the latency introduced by the Store-and-Forward method will vary according to the size of a frame. Latency is the time it takes the device to send the packet from the source to its destination. Latency coupled with bandwidth defines the network speed and the capacity of that network. Store-and-Forward switching is the slowest method of the three outlined in this section, but is the default on high-speed Catalyst switches like those in the 5000 and 6000 series.

Cut-Through

Cut-Through switching is the fastest switching method offered by Cisco. Cut-Through doesn't copy the entire frame to its buffers, but instead begins the forwarding process immediately once the first six bytes of the frame have entered the switch (remember that the first six bytes provide destination MAC address information). Because CRC values aren't calculated, Cut-Through switches may very well forward corrupted frames. This is the tradeoff - you give up the added reliability of the additional CRC check, but gain in terms of faster frame forwarding rates. Another version of this switching method is referred to as adaptive Cut-Through. This involves a switch

using Cut-Through until an error threshold is encountered. Once reached, the switching method automatically changes to Store-and-Forward.

FragmentFree

FragmentFree is a variation on the Cut-Through method described previously. Instead of initiating the forwarding process after the first six bytes of a frame, a switch configured to use FragmentFree waits until the first 64 bytes have been received, making sure that they are properly formatted. If they are, the switch will forward the frame to the port on which the destination MAC address resides. FragmentFree is the default method used on Cisco 1900 series switches.

NOTE: Cisco Catalyst 1900 series switches supports both the FragmentFree and Store-and-Forward switching methods.

Redundancy and Loop Avoidance

When designing a switched or bridged network, you'll almost certainly need to consider redundancy. While network redundancy is a great idea in principle, there are issues that you'll need to be aware of. The biggest issue is that bridging redundancy exposes networks to a loop, and loops cause major problems if not dealt with properly.

The problems associated with network loops go back to the days when bridging was the primary way of segmenting a LAN. The idea was to have more than one bridge connecting two segments, in order to provide a redundant path should a link or bridge fail. The problem with having this redundancy in a bridged environment is that it may create loops, and network loops are capable of causing communication problems. In the case of a bridging loop, a network becomes susceptible to broadcast storms.

In this section you'll notice that I tend to refer to bridges instead of switches. The main reason is that a bridged network is easier to diagram than one using switches. For all intents and purposes, when describing loops the terms bridge and switch can be used interchangeably.

Broadcast Storms

A broadcast storm occurs when packets are continuously forwarded around a network, grinding it to a halt. This is most easily illustrated with a simple example. Consider the network in Figure 3-1, with two bridges (Bridge A and Bridge B) connecting network segments 1 and 2. At first glance, the network probably looks good – if one of the two bridges fails, a second redundant path to the other segment exists. However, this design quickly becomes problematic, as I describe in the steps below.

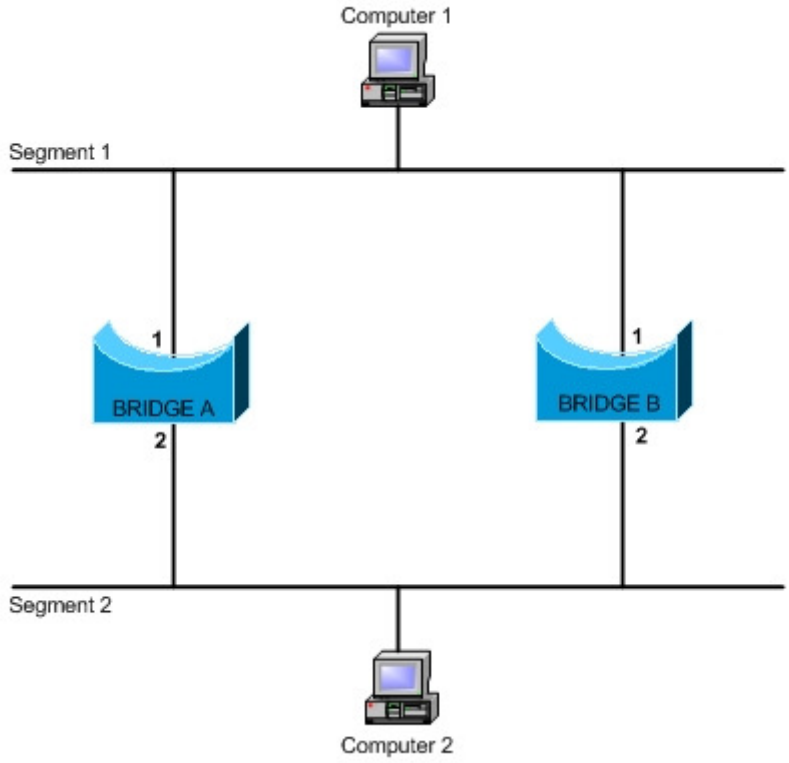


Figure 3-1: Bridged network with redundancy.

Imagine that Computer 1 wants to send a frame to Computer 2. Let's say that the bridges have already discovered that Computer 1 is on segment one, and that Computer 2 is on segment 2, and they have already added both to their MAC address tables. This is outlined in Figure 3-2.

Bridge A MAC Table			Bridge B MAC Table	
MAC	**Interface**		**MAC**	**Interface**
Computer 1	1		Computer 1	1
Computer 2	2		Computer 2	2

Figure 3-2: MAC Address tables for Bridge A and B

Remember that both segments are shared. As such, when Computer 1 sends a frame, both Bridge A and Bridge B will see it. Herein lies the first problem – both bridges see the frame and both will forward it. This means that 2 frames will be forwarded onto segment 2, even though only one was originally sent.

When those frames are forwarded, we have an even bigger problem. The frames will not only arrive at Computer 2 twice, but will also end up being encountered by both bridges on their segment 2 interfaces, as shown in Figure 3-3.

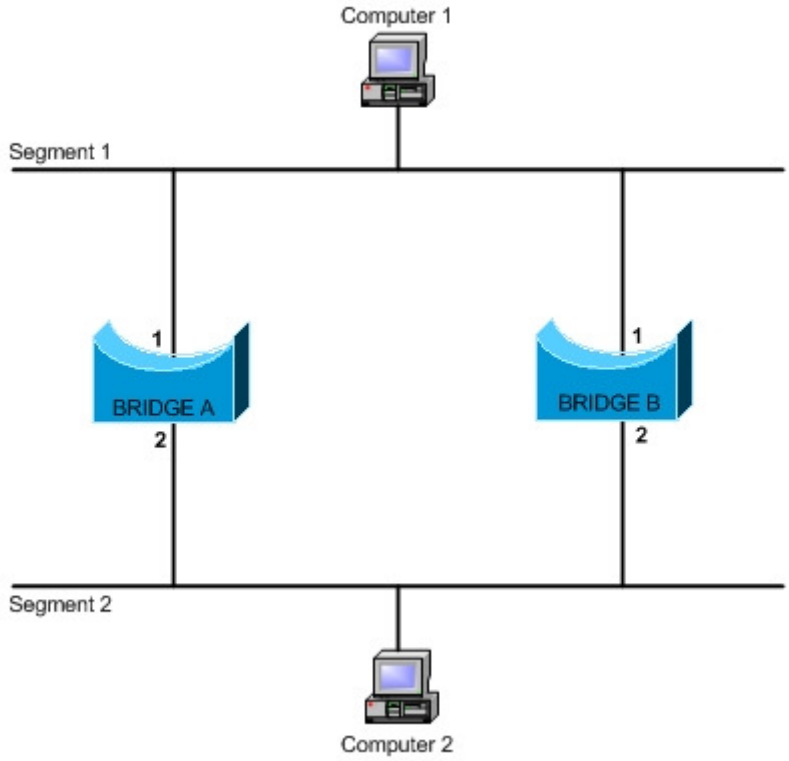


Figure 3-3: Frame forwarded by both Bridge A and Bridge B.

So why is this a problem? Well, you'll need to recall how a bridge builds its forwarding table. In this case, Bridge A will see the frame forwarded by Bridge B, and will look at the source address. The source address is still that of Computer 1. Because of this, Bridge A will now think that Computer 1 has been moved to segment 2, and will change its MAC address table. In the same way, Bridge B will see the frame forwarded by Bridge A, and will do the same. See the problem starting already?

Now let's say that Computer 2 responds. The bridges will not forward these frames, because they think that Computer 1 is also on segment 2. Since Computer 1 never gets a response, it sends the original frame again - a bigger problem. The bridges now believe that Computer 1 is again on segment 1 (by looking at the source MAC address), and change their MAC tables yet again. 2 frames again get forwarded, and so on. Ultimately this corrupts the MAC address table. Based on the quantity of traffic being moved (remember, it could be many systems transmitting), a broadcast storm takes over the network. All this happens simply because we wanted a little redundancy.

So how do you avoid this scenario? Am I saying that you can't have redundant links between switches? Not at all, but you will need a little help. In order to avoid the loops just described, you'll need to use something called Spanning Tree Protocol (STP), defined in standard 802.1d. It will eliminate network loops until an active link loses connectivity. When this happens, Spanning Tree will automatically reconfigure the network to use the redundant path.

Spanning Tree Protocol (STP)

Let me start by saying that there are still many network administrators out there who consider the words "Spanning Tree" to be evil. This stems from the fact that many networks were first segmented using bridges, and this is where Spanning Tree got its start. The main problem is that just like anything else; Spanning Tree doesn't work by magic. When a network is designed with redundancy and a link fails, it takes time to figure everything out and get back up to date - a process referred to as convergence. When a network using Spanning Tree does experience a failure, it can take anywhere between 30 seconds to over a minute for the network to converge. On a less-than-reliable large network, this can cause just as many headaches as it solves. To that end, many network administrators stopped using Spanning Tree and instead designed their networks loop-free, deciding that the lack of redundancy was better than dealing with convergence issues. In fact, once routing became a popular way to segment a network, many folks thought they had seen the end of it.

That was of course until Layer 2 switching became popular, reintroducing the need for the Spanning Tree Protocol on networks. The good news is that in the meantime, networks and equipment have gotten much more reliable. In that way, Spanning Tree isn't nearly as painful as it once might have been, although the protocol itself still works in the same way it used to.

Spanning Tree isn't nearly as difficult as most people make it out to be. At the most basic level, Spanning Tree's job is to eliminate loops in a bridged or switched network that are caused by redundant paths. It does this by learning the topology of the network, and then selectively blocking ports to eliminate any loops. This is where the 'tree' part comes in. Instead of a network with loops, with Spanning Tree what you end up with is a tree-like structure of branches. When a failure occurs, Spanning Tree recalculates the topology, and makes use of the redundant path. If you keep this in mind as you're trying to understand the protocol, you'll really find that Spanning Tree isn't that complex.

Tip: Spanning Tree's main purpose is to eliminate loops caused by redundant links on switched or bridged networks.

On most switches, Spanning Tree is turned on by default, mainly to help save you from yourself. You might accidentally introduce a loop to your network without really thinking about it when adding new equipment. To that end, Spanning Tree can also be turned off; although you'll want to be sure that you really understand your network connections before doing that. By default, an instance of Spanning Tree runs on each VLAN that you've configured, although it can be turned off on a per-VLAN or global basis. When turned on, a switch using 802.1d will communicate with other switches using Spanning Tree to calculate its configuration. A Spanning Tree port can be in one of four states. These include:

- **Listening**. In this state, a port is listening to Spanning Tree messages (BPDUs) and attempting to figure out how the network is configured.
- **Learning**. In this state, a port is adding addresses to its MAC table, but not yet forwarding frames.
- **Forwarding**. When in this state, a port is sending and receiving data as normal. During normal operation, a port will be in either a forwarding or a blocking state.
- **Blocking**. When in this state, a port will neither send nor receive data, but will listen to network messages relating to Spanning Tree. By default, all ports are in blocking mode when a switch is first powered on.

So how is the state of a port decided? Well, messages are passed between bridges or switches that are referred to as Bridge Protocol Data Units (BPDUs). BPDUs are very small frames sent using multicasts to let other switches know about the network topology with respect to Spanning Tree. We'll reference BPDUs often as we look at how the Spanning Tree topology is built.

Spanning Tree Protocol: Root Bridge

The first critical concept in understanding Spanning Tree is that of the Root Bridge. In any Spanning Tree instance, there is only one Root Bridge, and it must be elected. The Root Bridge is elected in the initial exchange of BPDUs between bridging devices. But how does the Root Bridge get elected? That's simple. In networks running STP, every bridge has a priority value associated with it. By default, the priority of all bridges is 32,768, unless changed by an administrator. The bridge with the highest priority gets to be the Root Bridge. But wait - you'll need to remember that the highest priority is the bridge with the lowest priority value. That is, a bridge priority of 1000 would beat the default priority of 32,768.

You have probably never touched bridge priorities. So if all the priorities are equal, who wins? That answer is the bridge with the lowest MAC address. All BPDUs contain a field called the Bridge ID (BID), which is actually made up of both the bridge's priority and its MAC address. In cases where all priorities are equal, the bridge with the lowest MAC address gets to be the Root Bridge. Consider Figure 3-4, where Bridge A will become the Root Bridge, based on its MAC address.

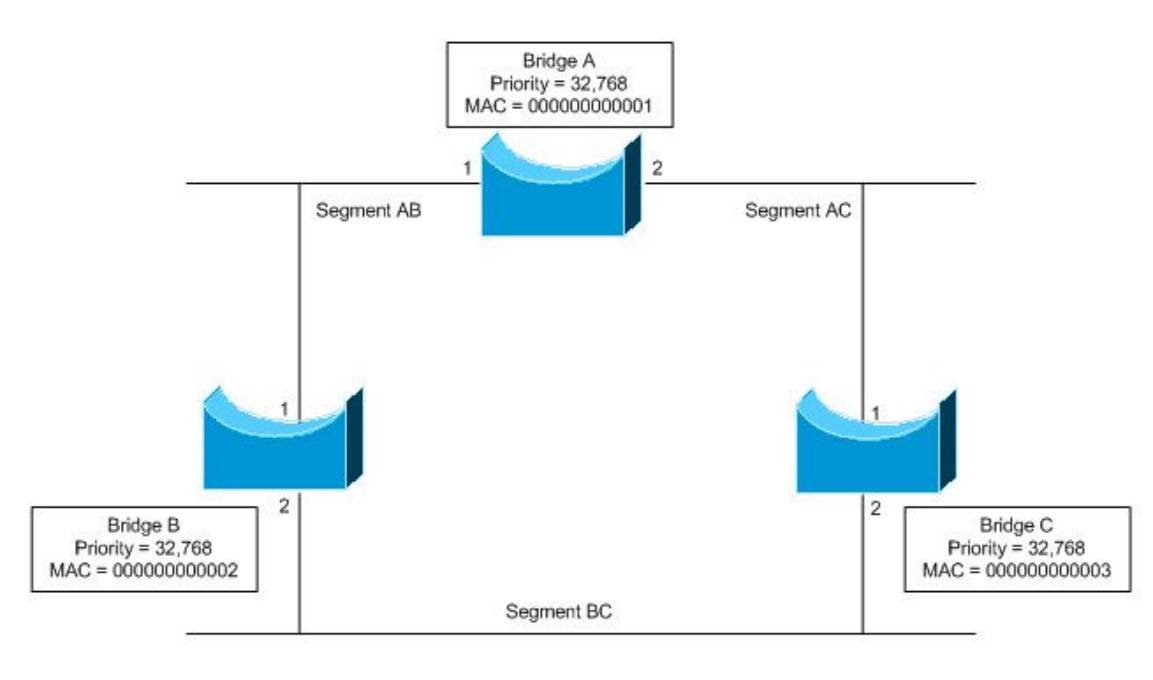


Figure 3-4: Root Bridge election using priorities and MAC addresses.

So why should you care about a Root Bridge? The answer is that all other bridging devices need to calculate a path to the Root Bridge in order to make decisions on which links will be used, and which will not. By calculating the distance to the Root Bridge, not only does STP work to build a loop-free topology, but also one that will have the shortest distance between devices. In the case of STP, "shortest" refers to the path that has the lowest aggregate cost to the root. We'll discuss aggregate costs in just a moment.

TIP: When bridge priorities are equal, the switch with the lowest MAC address will become the root bridge.

Spanning Tree Protocol: Root Ports

After the Root Bridge has been elected, it's time for bridges to designate what are known as Root Ports. Before we can look at how Root Ports work, you need to know something about port costs. Obviously some ports are faster than others, and usually a faster port will be used to interconnect switches. As such, ports have what is known as a cost value, based on their speed. The lower a cost value, the faster a port. Table 3-1 outlines cost values used in STP calculations. It's worth noting that when originally defined by the IEEE, 1 Gbps seemed like the fastest possible port speed. As such, there are two cost ranges that you'll find on switches. The first column shows the original IEEE Spanning Tree port costs, and the second shows the new cost numbers. The Cisco 1900 switch uses the original IEEE values by default. Note that port cost values can also be changed.

Table 3-1: Original and new IEEE port costs.

Port Speed	Original IEEE Port Cost	***New IEEE Port Cost***
1 Gbps	1	4
100 Mbps	10	19
10 Mbps	100	100

Remember that a cost is associated with a port. These cost values are used in calculating which port will be the Root Port for any given bridge. All Non-Root Bridges will have one Root Port.

Figure 3-5 outlines why designating a Root Port is important. In it, bridges are exchanging BPDUs to try and find the lowest cost to the Root Bridge. Note that Bridge A is the Root Bridge in this case. Because it is the Root Bridge, both of its ports have a cost of 0. In this example, all of the bridges are connected using 100Mbps links. The port cost outlined in the table for a 100Mbps link is 19 (using the new IEEE costs).

Let's walk through the process step by step. The Root Bridge will send out BPDUs with a cost of 0. These BPDUs will go to the 100Mbps port 1 on both bridges B and C. Since these ports have a cost of 19, the cost associated with port 1 on switches B and C reaching the Root Bridge is 19.

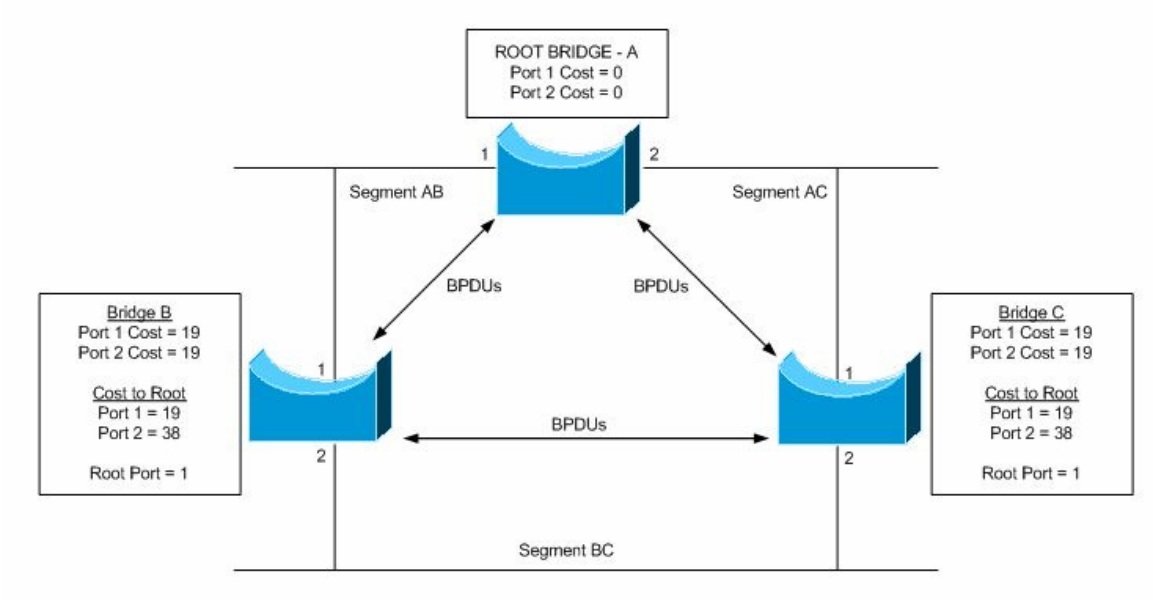


Figure 3-5: Port costs and Root Ports.

Now since B and C are also connected on segment BC with 100Mbps ports, these ports will also forward out BPDUs. Let's assume that B is sending a BPDU to C. In it, it will announce a cost of 19 to reach the root. When it reaches C, this bridge will add its port cost of 19 to the value in the BPDU. As such, bridge C knows that it can reach the Root Bridge using port 2 with an aggregate cost of 38, or it can reach the Root Bridge with a cost of 19 using port 1. For bridge C, the Root Port becomes port 1, as it does for Bridge B as well. For both bridges B and C, port 1 represents the lowest cost to the root.

To summarize, a Root Port is the port on a switch that has the lowest cost path to the Root Bridge.

Spanning Tree Protocol: Designated Port

You may have noticed that we haven't talked about loops yet. If you look back at Figure 3-5, a loop definitively exists. On each network segment, one port needs to be chosen at the Designated Port. The responsibility of the Designated Port is to act as the single interface to forward traffic destined for the Root Bridge. Recall that in our network example, 3 segments exist. Refer back to Figure 5 to review the costs associated with each port on our network.

To choose the Designated Port, another election needs to take place. Bridges compare their port costs to decide who gets to be the Designated Port for that segment. Consider each segment in Figure 3-5:

On segment AC, the Designated Port will be port 2 on bridge A. That's because port 2 on bridge A has a cost of 0, while port 1 on bridge C has a cost of 19.

On segment AB, the Designated Port will be port 1 on bridge A. Again, port 1 on bridge A has a cost of 0, while port 1 on bridge B has a cost of 19. Since their port cost is always 0, it should be clear that ports on the Root Bridge will always be Designated Ports for their connected segments.

On segment BC, there is a tie. Port 2 on each bridge has a cost of 19. As such, the Designated Port will be the switch with the lowest MAC address. In this case that's bridge B, so port 2 on bridge B will become the Designated Port.

Note that after all this is done, all traffic from segment BC will be forwarded out port 2 on switch B. Port 2 on switch C will be put into blocking mode, as shown in Figure 3-6. Notice also that there are no longer any loops on our network.

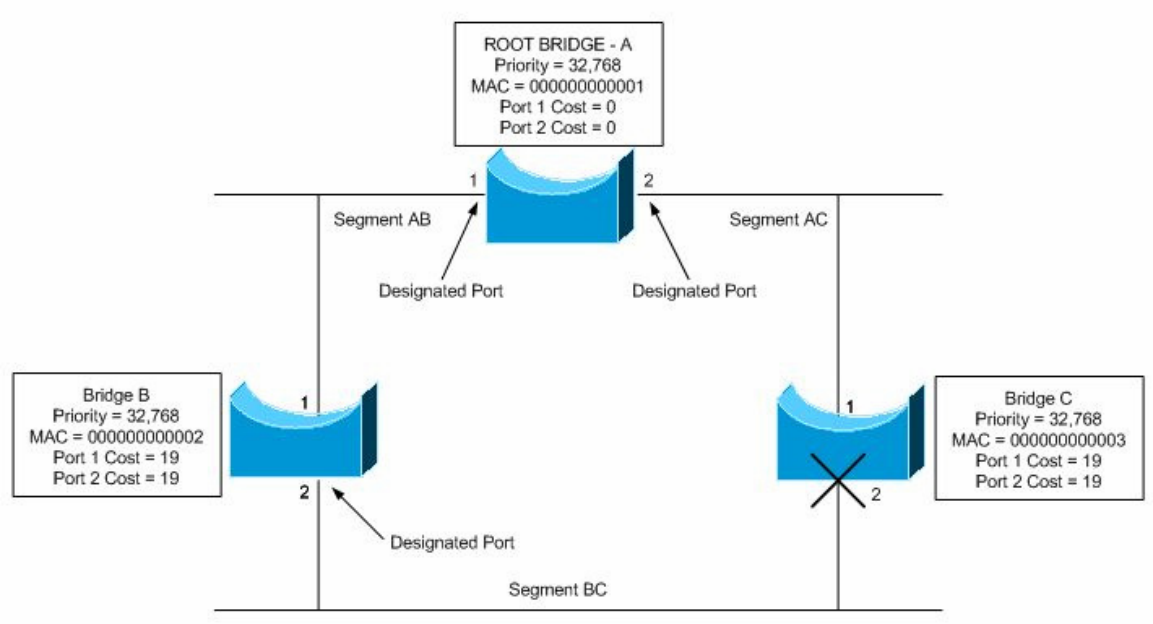


Figure 3-6: Designated Ports for each segment.

On any given segment, the bridge that has the current Designated Port is also known as the designated bridge.

Spanning Tree Protocol: Convergence

While our network is now loop free, STP keeps working away. BPDUs are still sent out at 2-second intervals to be sure that things are how they should be. If at any point a bridge doesn't hear from another bridge, the topology will need to be recalculated. For example, let's say that bridge B fails. Bridge C will stop receiving BPDUs on its blocked port. Once 20 seconds have passed without receiving a BPDU from Bridge B, Bridge C will:

First go into a listening state for 15 seconds. During the listening state, a bridge is examining BPDUs sent by other bridges.

Then go into a learning state for 15 seconds. During this time, the bridge is building the MAC address table for the connected segment. Remember that it was in a blocked state previous to this.

After these stages, Bridge C will become the new designated bridge for segment BC, and will begin forwarding frames. Note that convergence took about 50 seconds to complete - 20 seconds waiting for a BPDU, plus 15 seconds listening and 15 seconds learning. The time during which a switch is listening and learning is referred to as the Forward Delay.

While a number of BPDUs are passed back and forth while a Spanning Tree topology is being calculated, in truth calculating a Spanning Tree topology is really no more than a three-step process:

1. Elect the Root Bridge
2. Elect a Root Port on each non-Root Bridge
3. Elect one Designated Port on each network segment.

Once these steps are completed, a network should be loop free. However, you should also recall that while listening or learning, ports are not forwarding frames. A network is converged once all bridges have switched to a forwarding or blocking state.

Virtual LANs (VLANs)

While a switched network provides a number of great features including better performance, it also provides a greater degree of flexibility, scalability, and security. This is mainly accomplished through the use of Virtual Local Area Networks (VLANs).

Let's get started on the right foot. VLANs are nowhere near as difficult to understand as some people make them out to be. Remember a traditional LAN? All computers are connected together in a segment referred to as a broadcast domain. Systems are usually very close together, and connect to devices such as hubs or switches.

At the most basic level, a VLAN is nothing more than a broadcast domain, meaning that it is a Layer 2 entity. What tends to throw people is that a VLAN is actually a broadcast domain configured on switches on a port-by-port basis. Say what? Well, imagine a switch that has 10 ports, like the one in Figure 3-7. I could take the first 5 ports and make them part of one VLAN, and then take the next 5 and make them part of a different VLAN. What I've actually done is created two different broadcast domains. A broadcast made by a system on port 1 will only go to systems connected on those first 5 ports, and is not seen by those on the last six ports. In this way, I've used the switch to define two different broadcast domains.

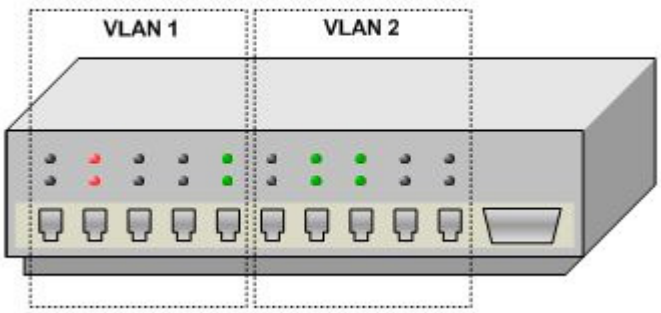


Figure 3-7: Switch with 2 VLANs.

But how do systems on different VLANs communicate with each other? You actually already know the answer to that: the same way systems on two different broadcast domains communicate, that is, through a router. So, in order for systems on one VLAN to communicate with those on another, some type of routing device must be involved. This could be a router, a Layer 3 switch, or a route switch module (RSM).

Because different VLANs are actually different broadcast domains, you'll need to be sure that you're connecting hosts to the correct ports. For example, if you were to connect two systems that are supposed to be part of the same IP subnet to ports from different VLANs, they would not be able to communicate. That is, even though the IP addresses assigned to the hosts may be correct, it's still possible that you've made them part of different VLANs by accident. In cases where two correctly-configured hosts connected to the same switch cannot communicate, be sure to check the VLAN membership of ports on the switch.

It is also possible for VLAN configurations to go beyond a single switch. Consider the Figure 3-8, where a crossover cable connects two switches. Notice that both switches contain ports on VLANs 1 and 2. In order for these VLANs to communicate properly across switches, we'll need to set up and configure a trunk connection, which we'll look at shortly.

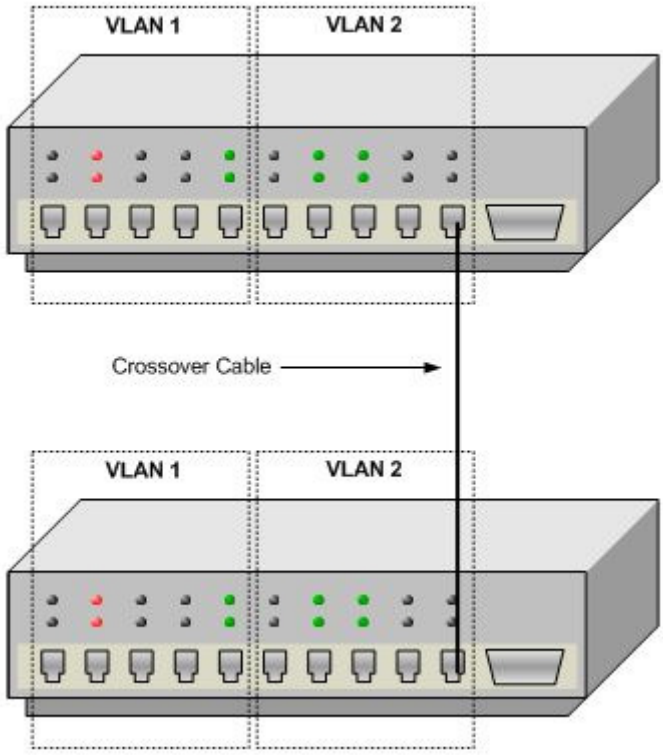


Figure 3-8: VLANs spanning more than one switch.

Benefits of VLANs

The main benefits of using VLANs are that they provide increased flexibility, scalability, and security:

- **Flexibility**. Because a VLAN assigns a user to a broadcast domain based on the port their system is plugged into, additions, moves, and changes are made simple. If you wanted a user to be moved to a different broadcast domain, all you would have to do is reconfigure the port they're connected to. For example, a user might have been working in Marketing and is now moving to Finance. Instead of moving their physical PC, simply modify the VLAN configuration of the port, changing it from the

Marketing VLAN to the Finance VLAN. This flexibility allows you to create logical (rather than physical) groupings of users.

- **Scalability**. VLANs aren't limited to a single switch. In fact, you can actually have VLANs that span an enterprise organization. For example, you might have groups of users on the 4th, 18th, and 42nd floors all being part of the same VLAN, even though they're connected to different switches. Taken a step further, VLANs can also be scaled beyond a single location, over a WAN link if necessary.
- **Security**. VLANs provide a greater degree of security than a traditional LAN. Consider the situation where a user from Human Resources might be connected to the same physical segment as a variety of other users. If plugged into a hub, other users could capture the information passed to the HR computer using a protocol analyzer program. By configuring all HR user systems to be in their own VLAN, their traffic remains separate and distinct from other network users. Also, traffic between VLANs can then be more easily controlled using router features such as access lists.

In the above explanation, the groupings of users to VLANs were based on functional departments. This is a popular way to divide traffic on a network, especially if departments have their own servers. Human Resources tends to be a particularly good example of a department that might require its own VLAN, based on the sensitive nature of the information they deal with. Separate VLANs are often defined for other purposes as well, including network management and monitoring.

Exam Tip: Remember that when you define additional VLANs, you are actually creating a larger number of smaller broadcast domains.

VLAN Types

VLAN membership can be configured in two different ways, known as static and dynamic:

- **Static VLANs**. With a static VLAN, an administrator defines VLANs on a switch and then assigns ports to them. This is the most common way in which VLANs are configured.
- **Dynamic VLANs**. A dynamic VLAN is one in which a switch port automatically configures itself to be part of a particular VLAN, based on the MAC address of a connected system. Think of a scenario where a laptop user uses different connections within an office building. In this case, she could plug into a given jack (which is connected to a switch) and automatically be made part of her native VLAN. In order to accomplish this, a management database needs to be created that maps MAC addresses to VLANs, which requires additional administrative effort. Cisco has a product that provides this functionality - VLAN Management Policy Server (VMPS).

Trunking and VLAN Identification

Setting up VLANs on a single switch is relatively simple. First you define different VLANs, and then make ports members of those VLANs. However, when you interconnect or link switches across a network (referred to as trunking), you'll need a way for switches to know on what VLAN a frame belongs. There are two main types of trunk links, as described below:

- **Access Link**. When a link connects a single VLAN between switches, and no traffic for other VLANs is passed over that link, it is considered an access link. The only traffic that moves across an access link is traffic belonging to the VLAN defined for the ports that are connected.

- **Trunk Link**. If a link connects two switches, and the switches have 2 or more VLANs defined, it wouldn't make much sense to set up a separate access link for each VLAN. Instead, it would be great if we could have traffic from multiple VLANs move across a single link. If a VLAN identification (frame tagging) technique is used, this is possible. The link is then known as a trunk link.

Consider Figure 3-9, which outlines both access and trunk links.

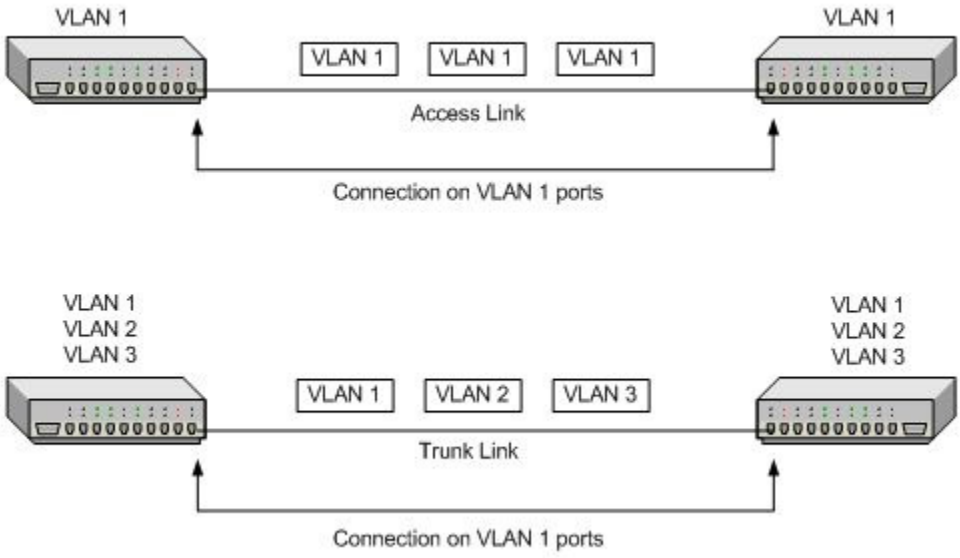


Figure 3-9: Access and Trunk Links.

Remember that switches are always connected together using a crossover cable.

If you remember back to Chapter 2, none of the Ethernet frames we looked at had any field used to identify the VLAN membership of a frame. In order for VLANs to work properly between switches, we'll need some way to be able to let switches know what VLAN a frame is meant for.

Enter frame tagging. Frame tagging is a technique where additional VLAN identification information is added to a frame. Two main protocols exist for the purpose of Ethernet frame tagging – Inter Switch Linking (ISL) and IEEE 802.1q. Both modify a frame in different ways to add VLAN identifiers. Once implemented, VLAN tagging allows ports on the same VLAN (but on different switches) to communicate as though they were part of a single physical switch.

Adding more information to a frame creates a slight dilemma. Remember that an Ethernet frame has a maximum size of 1518 bytes. How can we add information to a large frame without making it appear oversized and thus invalid to network devices? Well, we need to configure the ports that link switches to use a VLAN identification protocol. When configured with VLAN tagging, a switch port will tag a frame with VLAN information when sending it out a trunk port. This tagging will be stripped away by the switch at the receiving end of the link. In this way, end devices need not be aware that any special framing or tagging took place. It also helps avoid end systems seeing these frames as being invalid. A VLAN tagged frame has a maximum size of 1522 bytes. Figure 3-10 illustrates the process by which a frame is tagged to include VLAN identification information.

Note that the special tagging is added before it leaves the Switch 1 trunk port, and is removed once it enters the trunk port on Switch 2.

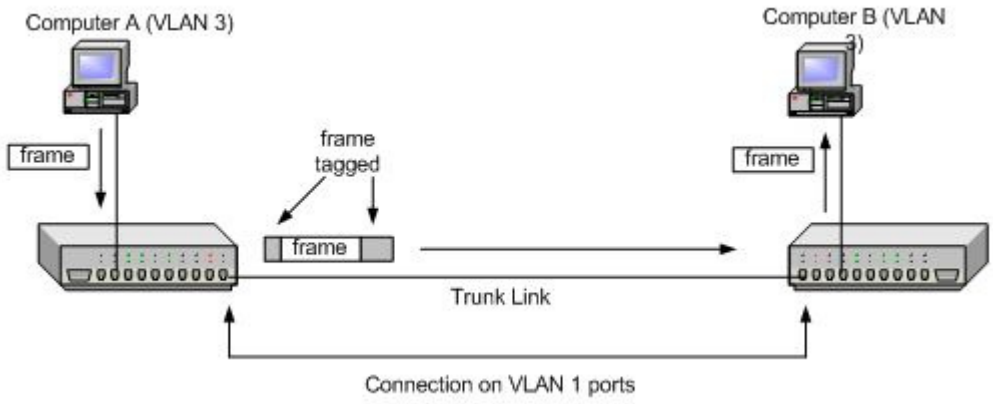


Figure 3-10: Frame tagging over a trunk link.

A number of different protocols exist for the purpose of adding VLAN identification to frames. These include:

- **InterSwitch Link (ISL)**. ISL is a Cisco proprietary VLAN identification protocol that can be used only on Fast Ethernet and Gigabit Ethernet trunk ports. Because the protocol is proprietary, it can only be used to trunk between Cisco devices. ISL actually re-encapsulates the entire original frame with a new header and a new CRC value.
- **IEEE 802.1q**. The IEEE 802.1q is the industry standard method of VLAN identification. This protocol doesn't entirely re-encapsulate a frame, but instead adds VLAN identification information into Ethernet frames. This in turn can make Ethernet frames as large as 1522 bytes. When you want to use VLAN identification on a network that includes equipment from different vendors, 802.1q should be used.
- **Dynamic Trunking Protocol (DTP)**. An enhancement of Cisco's Dynamic ISL (DISL) protocol, DTP dynamically negotiates both ends of a trunk link to use a common VLAN identification protocol, such as ISL or 802.1q.
- **FDDI 802.10**. While trunking protocols such as ISL are meant to create a trunk link between only two switches, 802.10 encapsulation allows VLAN tagging to be used on a shared FDDI backbone. It does this by adding a 4-byte Security Association Identifier (SAID) field to the FDDI frame header.
- **ATM Lane**. When Ethernet or Token Ring networks connect over ATM, LAN Emulation (LANE) must be used to emulate their native environments (since ATM doesn't support broadcasts, for example). In cases where VLANs are required over ATM connections, Emulated LANs (ELANs) need to be defined. Each ATM ELAN maps to a single VLAN.

For the purpose of the CCNA exam, you'll only need to understand the ISL VLAN identification method.

Exam Tip: VLAN tagging methods like ISL allow VLAN membership information to be transported with a frame across trunk links.

InterSwitch Link (ISL)

VLAN tagging is not a difficult concept – just remember that its purpose is to allow frames from multiple VLANs to be transferred across a trunk link and properly identified at the other end.

Recall that ISL only works on 100Mbps ports and faster. As such, if your switch only has 10 Mbps ports available, using ISL will not be an option. One other limitation of ISL is that only ISL-aware devices will understand ISL frames – all others will not consider the frame to be valid.

Some network interface cards include ISL capabilities. If installed in a server, the server could then be part of two (or more) VLANs concurrently. This would allow systems from different VLANs to connect to the server without needing to route between different broadcast domains. In this way, the connection between the server and the ISL-configured switch port acts as a trunk link. In an even more common example, imagine if you connected a router to your switch in order to route between different VLANs. If that router had a 100Mb port that was ISL-capable (as many Cisco routers do), it could be connected to a trunk port on the switch, and provide routing between your VLANs. In this case, the router would add VLAN identification tags before forwarding a frame to the switch (and vice versa). The switch interface would strip away the tagging, and be sure that the frame is forwarded onto the proper VLAN.

Note that there is a downside to this configuration. By making a system (the router in this case) part of multiple VLANs, it will receive broadcast traffic from each of the VLANs for which its switch port is configured.

VLAN Trunking Protocol (VTP)

In large networks, configuring VLAN information on each and every switch would be incredibly time consuming. In order to deal with this issue, Cisco created a protocol referred to as the VLAN Trunking Protocol (VTP). VTP actually has very little to do with trunking. Instead, its responsibility is propagating information about the configuration of VLANs across trunk links. For example, let's say that you've defined a new VLAN on a switch, VLAN 99. Instead of having to manually create that VLAN on each and every switch, you could instead use VTP – it would automatically make VLAN 99 available on every switch after it was defined on the first. This information is sent to other switches in the form of VTP advertisements. These are multicasts that provide update information to neighboring switches over trunk links. VTP can be used to add, modify or delete VLANs across what is known as a VTP management domain. For example, you might add a new VLAN, change the name of a VLAN, or delete a VLAN that you no longer require.

By default, VTP is not configured on Catalyst switches. You first have to define what is known as a VTP management domain, the group of switches among which you want VLAN information passed and shared. You can actually define multiple VTP domains for different groups of switches that require different configurations. However, you can only make any given switch part of a single VTP management domain.

VTP Modes

Switches can operate in three different VTP modes, which affect the way in which they share or interact with VTP advertisements. Once configured, VTP information is stored in their VTP database. The three modes include:

- **Server**. By default, every Catalyst switch configured to use VTP will be configured as a server. In any VTP domain, at least one server must exist. When in server mode, a switch can be used to add, modify or delete VLAN related information, which will be passed to all other switches in the VTP management domain.

- **Client**. In client mode, a switch receives VTP advertisements and makes changes according to their contents. Note that a VTP client cannot change VLAN information.
- **Transparent**. In transparent mode, a VTP switch will forward VTP messages, but will not actually use the configuration information it receives. VLANs that are added, modified or deleted on a VTP switch in transparent mode will only apply to that particular switch.

  When a switch receives a VTP update, it checks the VTP domain name and revision number information stored in the advertisement. If the information is for a different VTP domain, it is ignored. If the revision number of the advertisement is below the number currently stored in its database, it is similarly ignored.

VTP Pruning

In some cases, it may not make sense for all traffic to be trunked to all switches. For example, consider the network diagram shown in Figure 3-11. In it, Switch C does not have any ports configured on VLAN99. As such, it doesn't make much sense for traffic destined for VLAN99 to be sent over the trunk link between switches A and C. In order to control traffic destined for VLAN99 from being forwarded to Switch C, you can enable VTP pruning. Once enabled, VTP pruning will stop unnecessary traffic from being forwarded to a switch with no configured ports on that VLAN. If VTP pruning were enabled in this example, traffic for VLAN 99 would not be forwarded to Switch C by Switch A, thus conserving bandwidth and switch resources.

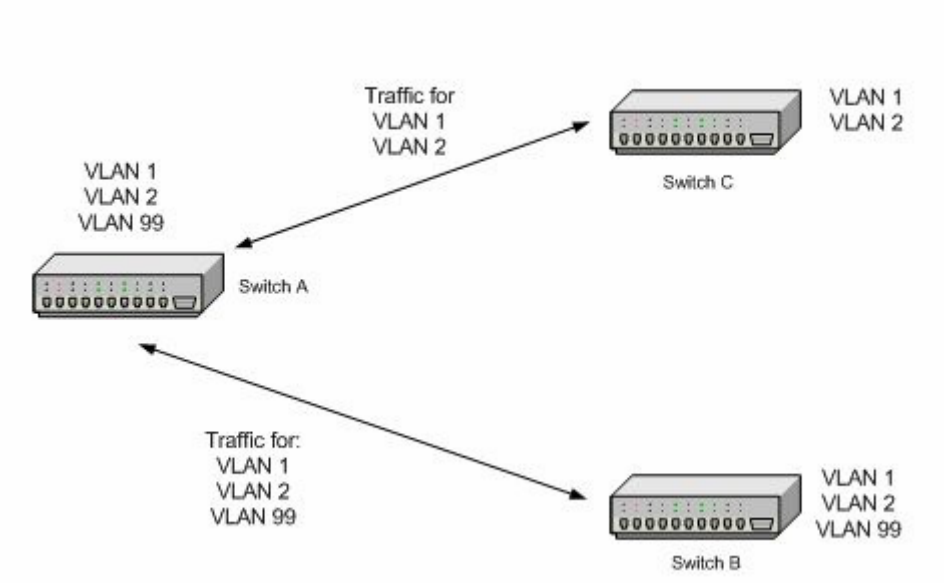


Figure 3-11: With VTP pruning enabled, traffic for VLAN99 is not forwarded to Switch C.

By default, all switches have VTP pruning disabled. When enabled, VTP pruning enables the function for the entire VTP management domain. If you do choose to use VTP pruning, note that you also have the option to go a step further and define which VLANs are eligible to be pruned and which are not.

Initial Configuration of a Cisco Catalyst Switch

In this section we'll take a look at the basic initial configuration of a basic Cisco switch, a 1900 series model. As mentioned at the beginning of the chapter, we'll leave the complete configuration to Appendix A, after we've learned more about the Cisco IOS. More advanced switch models and their configuration settings will be explored in more detail in upcoming chapters

To begin, you'll need to be familiar with the actual hardware. A 1900 comes in two main configurations with different port types and densities. The good news is that 1900s are easily recognized by their model number. For example:

- **1912**. Has 12 10Mbps Ethernet ports and 1 or 2 100Mbps FastEthernet ports.
- **1924**. Has 24 10Mbps Ethernet ports and 1 or 2 100Mbps FastEthernet ports.

The 1900 series also includes an Attachment Unit Interface (AUI) connector on the rear of the unit. This can be used for connections to 10Mbps networks including 10Base2, 10Base5, and fiber optics using an added transceiver. Additionally, the switch includes a console port, which we'll use

Powering the Switch

When you first power up a 1900 series switch, it goes through a Power-on-self-test (POST). If you're sitting in front of a 1900, you should notice that if the port lights (LEDs) are green, they are functioning correctly. Amber port LEDs suggest that a problem exists.

While the LEDs provide a quick visual indicator of whether a switch is functioning correctly, you'll need to create a terminal session with the switch to get at the configuration.

To do this, you have to connect the supplied console cable to the back of the switch, and the other end to a serial port on your PC (using the small terminal adapter provided). Initial configuration is done using a terminal emulation application such as HyperTerminal. HyperTerminal will need to be configured correctly to communicate with the switch. Figure 3-12 outlines the required settings. Note that for my PC, I am connecting to the switch using COM 3.

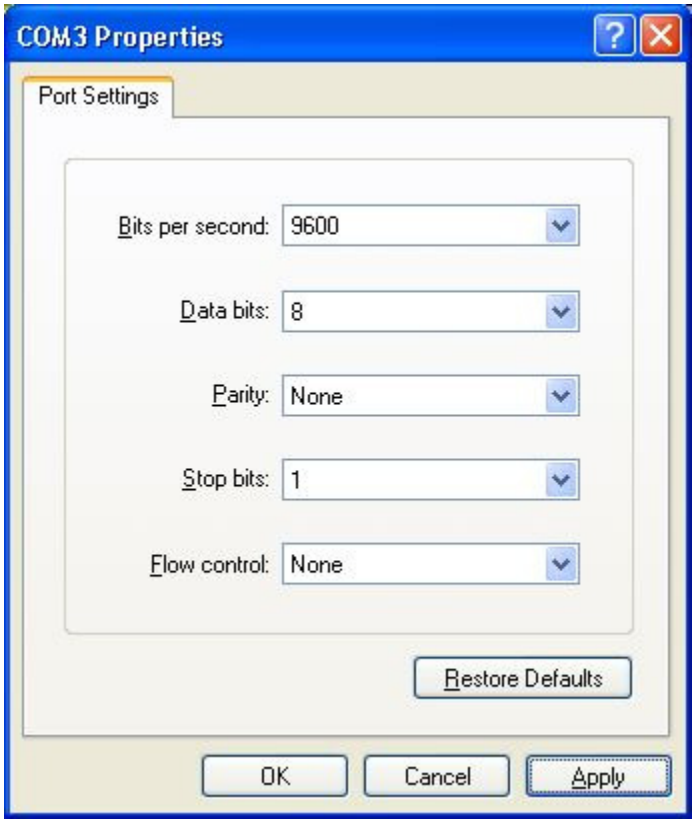


Figure 3-12: Configuration Settings for Console session.

After you've configured the correct configuration settings and pressed OK, you'll be prompted with the menu below.

Catalyst 1900 Management Console
Copyright (c) Cisco Systems, Inc.
1993-1999
All rights reserved.
Enterprise Edition Software

Ethernet Address: 00-50-F0-5F-25-00

PCA Number: 73-3122-01
PCA Serial Number: FAB03103IYN
Model Number: WS-C1912-A
System Serial Number: FAB0312S041
Power Supply S/N: APQ0252023A
PCB Serial Number: FAB03103IYN, 73-3122-01
---

1 user(s) now active on Management Console.

User Interface Menu

[M] Menus

[K] Command Line
[I] IP Configuration
[P] Console Password

Enter Selection:

Notice that the 1900 interface is menu driven to begin with. Two versions of software exist for the 1900 – the Standard Edition and the Enterprise Edition. In the Standard Edition, the switch can only be configured from the menus or later via a web browser. In the Enterprise Edition, the switch can also be configured from the traditional IOS command line. Looking at the information provided above, you notice that the switch is a model 1924, and is running Enterprise Edition software. If your switch were running the Standard Edition software, you wouldn't see the Command Line option (menu item K) shown.

Configuring a switch using the console port really isn't the most convenient way to do things regularly. Instead, you'll want to give the switch a password and IP address, which will allow you to handle further configuration via a telnet session.

To begin, I would suggest choosing the P option in the menu configuration and setting a password. For a 1900, the password must be between 4 and 8 characters, and is not case-sensitive. This password will enable privileged access to the switch, allowing changes to be made to the configuration. It's worth noting that the password is not encrypted. Pressing P will provide you with the following output:

The Management Console password can help prevent unauthorized accesses.
When specifying a password, use a minimum of 4 characters and
maximum of 8 characters. The password is case insensitive and
can contain any character with a legal keyboard representation.
For the user's protection, the password must be entered the same
way twice before it will be accepted.

Enter new password:

After setting the password, you'll be returned to the main menu. To set the IP configuration, press I:

Catalyst 1900 - IP Configuration

Ethernet Address: 00-50-F0-5F-25-00

---------- Settings ----------

[I] IP address 0.0.0.0
[S] Subnet mask 0.0.0.0
[G] Default gateway 0.0.0.0
[V] Management VLAN 1

[M] IP address of DNS server 1 0.0.0.0
[N] IP address of DNS server 2 0.0.0.0
[D] Domain name
[R] Use Routing Information Protocol Enabled

---------- Actions ----------

[P] Ping
[C] Clear cached DNS entries
[X] Exit to previous menu

Enter Selection:

If you choose I again, you'll be presented with the following:

Current setting ===> 0.0.0.0
New setting ===>

Simply type up the new IP address and press enter to change it.

Choosing any of the above menu options will allow you to configure that particular setting. At a minimum, you'll probably want to configure at least the IP address, subnet mask and default gateway for the switch. Once you've completed this, press X to return to the main menu. Note that configuration changes apply immediately.

Once the switch has a properly configured IP address and subnet mask, you can telnet into it. For the most part, the menu interface is very straightforward and shouldn't require a whole lot of explanation. In Appendix B we'll take a look at configuring the various elements described in this chapter from the command line.

Layer 2 Multicast Features

Please note that the material covered in this article is relevant to the Cisco CCDA exam only.

In Chapter 2 I provided a brief overview of the concept of multicast transmission. If you'll recall, a multicast is a type of transmission in which a single traffic flow is sent to multiple recipients – in other words, a one-to-many technique. In the world of TCP/IP, multicast transmissions are a Network layer concept, using the Internet Group Management Protocol (IGMP) to manage which systems will ultimately receive a multicast, and which routers will forward it. You may also recall from earlier that switches will, by default, forward all broadcast, multicast, and unknown destination frames to all connected ports. While this may not seem unreasonable at first glance, imagine a multicast being forwarded to literally hundreds of ports when only one or two hosts actually need the data being sent. Obviously this is wasteful, and some technique is required to both reduce the amount of work required by individual switches, and the number of systems that need to process unnecessary traffic. For the purpose of the Cisco CCDA exam, you will need to be familiar with some of the specific Layer 2 multicasting features that are supported by Cisco's line of Catalyst switches, and how these help to ensure more efficient and effective performance.

Multicasting Overview

Before getting into the details of Layer 2 multicasting protocols, let's first take a look at how IP multicasting works in a more generic sense. Although IP-based communications have not been looked at in detail yet, you have learned that multicasting is a one-to-many method of transmission. In order to facilitate this, a special class of IP addresses are designated or reserved for multicasts – Class D, or those addresses fall into the numerical range between 224 and 239 in

the first octet of an IP address. For example, the address 224.0.0.1 would be a valid multicast destination address. Later in this book I'll cover multicast addresses in more detail.

The primary multicast protocol used on the Internet is the Internet Group Management Protocol (IGMP). IGMP is a Network layer protocol whose primarily responsibility is managing the groups of hosts that which to receive a multicast, just like the name suggests. When a user opens a multicast-enabled application on their system, their computer sends an IGMP message to the local router. This message essentially tells the router that it should forward the requested multicast traffic onto this network. In turn, this router contacts its upstream router, telling that router to forward the particular multicast traffic requested. Overall, this ensures that forwarding a multicast to the entire Internet is unnecessary. Only those routers that actually have network hosts that "need" the multicast will have the traffic forwarded to them. This is illustrated in Figure 3-13.

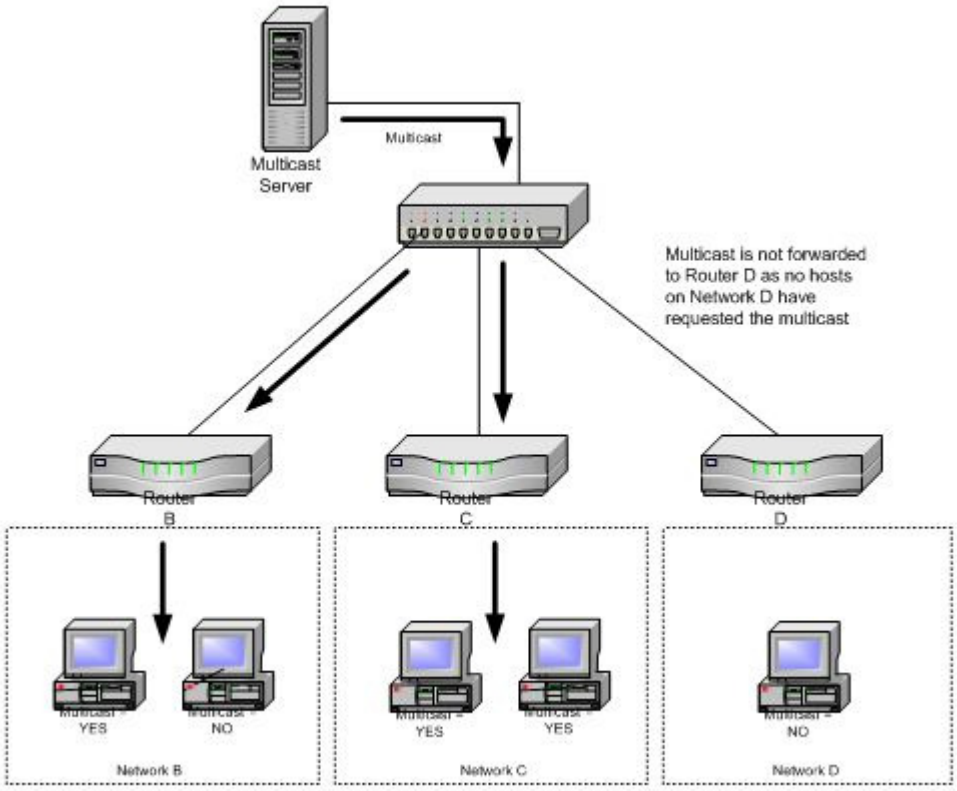


Figure 3-13: A single multicast is forwarded to necessary networks.

In fact, there may be many hosts on this network that wish to receive the same multicast transmission. In this case, as they open their multicast applications, they will simply begin processing the same frames forwarded by the router – again, the multicast is only sent once, but multiple hosts are listening and accepting the traffic. In order to reduce unnecessary traffic on the network, the local router will periodically send out what is known as an IGMP host membership query. If no systems respond to these queries, the router will ultimately stop forwarding this multicast traffic to the network.

Recall again that by default, a switch will forward multicast frames to all ports. What this means is that even though there are only perhaps two or three hosts that wish to receive the multicast, it

will still be forwarded to all systems, most of which will simply discard the traffic. This can place a heavy load on both the hosts and any switches, especially in cases where many different multicast streams need to be forwarded.

In order to deal with this traffic more effectively, Cisco Catalyst switches can use one of two different Layer 2 multicast features – the Cisco Group Management Protocol (CGMP) and IGMP Snooping.

Cisco Group Management Protocol (CGMP)

CGMP is a proprietary Cisco protocol and Layer 2 multicasting feature that works in conjunction with IGMP in order to make the forwarding of multicast frames more efficient. Because the protocol is Cisco specific, it only works on Cisco routers and switches. Basically, CGMP is a protocol used by a Cisco router to inform a Catalyst switch of the specific host that wishes to receive a multicast. With information about the specific hosts that wish to receive a multicast, the switch can then filter the multicast traffic, ensuring that only the hosts that require it actually receive it.

You should recall that switches make forwarding decisions based on MAC addresses rather than IP addresses. However, when a host wishes to join a multicast group for the purpose of receiving a transmission, it sends a membership report to the local router. When the router forwards the multicast onto the network, it cannot use the destination MAC address of the single host, because if this were the case, all other hosts who need the multicast would reject it. Instead, the router uses a specially created MAC address that is a variation on the multicast IP address. You don't to know the technical details of how this happens for the CCDA, but it is sufficient to say that the MAC address used unique identifies the multicast stream. When switches receive these frames, the MAC address shows them as being a multicast, and the frames will be forwarded out all ports. However, when CGMP is enabled on both switches and the local router, a different process takes place.

When the router receives the original IGMP message from a host requesting a multicast, this message includes the host's source IP address and MAC address. Based on this information, the router is able to send a CGMP message to switches, letting them know both the destination MAC address of the host, and the destination MAC address of the multicast. With this information, a switch is able to dynamically add another entry to its MAC address table, specifying that the port should receive traffic for both MAC addresses. Ultimately, this ensures that only the hosts that require the multicast stream are forwarded it. This eliminates the need for the switch to forward the multicast to all ports.

IGMP Snooping

IGMP Snooping is another Layer 2 function that helps to better manage the multicast traffic that a switch comes into contact with. Quite simply, not every environment is going to consist of Cisco equipment only. Routers and switches from other manufacturers may be present on the network, and these will be incapable of using the CGMP protocol. However, many vendors do implement a feature known as IGMP Snooping on their switches to help reconcile some of the multicast traffic issues mentioned earlier. A variety of Cisco Catalyst switch models do support IGMP Snooping, but it is important to recognize that at any given point in time, a Cisco switch can only be configured for either CGMP or IGMP Snooping – not both simultaneously.

As the name suggests, IGMP Snooping is a method that actually "snoops" or inspects IGMP traffic on a switch. When enabled, a switch will watch for IGMP messages passed between a host and a router, and will add the necessary ports to its multicast table, ensuring that only the ports that require a given multicast stream actually receive it. Unfortunately, IGMP Snooping suffers

from one major drawback, namely the need for the switch to inspect all IGMP traffic, on top of its other responsibilities. However, in environments that do not support CGMP, IGMP Snooping provides a solid alternative to having all multicast traffic flooded to all ports.

Cisco Catalyst Switch Models

For the purpose of the CCDA exam, you are expected to be familiar with the various models in the Cisco Catalyst line of switches. While Cisco doesn't expect you to be aware of every feature of every switch, it is important for you to have a solid foundation understanding of the various model series', their basic capabilities, and where they fit into the Cisco Hierarchical Network Design Model that was covered in Chapter 1. Since Cisco models can and do change, it is important for you to stay up to date on this information. For that reason, you should make a point of reading through the information posted on the Cisco website rather than relying solely on the information found here. It's quite possible that Cisco will release a new line of switching equipment, and then integrate questions that reference particular models into the CCDA exam. At the time this book was written, any of the Ethernet-related switches up to the 6500 series appear to be fair game for the exam.

Note: For more information on current Catalyst switch models, as well as the models listed here, visit http://www.cisco.com/en/US/products/hw/switches/index.html[/Note]

Cisco Catalyst 1900/2920 Series

Although the Cisco Catalyst 1900 and 2920 series of switches are no longer being sold by Cisco, you are still likely to come across these models in customer networks, hence the reason they are mentioned here. Used almost exclusively as a workgroup (access layer) switching solution, models in the 1900 and 2920 series provide both 10 Mbps and 100 Mbps switched Ethernet connectivity, using either Standard or Enterprise Edition software. The majority of the ports on these models are standard 10 Mbps RJ-45 ports, with one or two additional 100 Mbps ports using traditional Fast Ethernet or fiber optic connections.

Cisco Catalyst 2900 Series

The Cisco Catalyst 2900 series of switches are high-density fixed configuration switches primary used in wiring closets for access layer connectivity. The Catalyst 2980G model provides 80 10/100 Mbps Fast Ethernet ports, along with 2 Gigabit Ethernet uplinks (another model, the 2948, provides 48 10/100 Mbps Fast Ethernet ports). The Gigabit uplinks are provided in the form of GigaBit Interface Converter (GBIC) slots - these slots allow a variety of different gigabit interface modules to be slotted into the switch according to an environment's needs. Although the standard 2900 series models are Layer 2 switches only, Cisco has recently released a Layer 3 version of this line, allowing these models to provide both traditional Layer 2 switching, as well as Layer 3 switching (routing) functions. An example would be the Catalyst 2948G-L3.

Cisco Catalyst 2950 Series

The Cisco Catalyst 2950 series of switches are again primarily access-layer switches designed for connectivity to user workstations or servers. Models in this line typically provide 12, 24, or 48 10/100 Mbps Fast Ethernet ports, with higher-end models providing different combinations of gigabit port options. For example, the 2950-12 and 2950-24 models provide 12 and 24 10/100 Mbps Fast Ethernet ports respectively, with no gigabit ports. However, the 2950G-24 provides 24

10/100 Mbps ports, along with 2 GBIC slots for uplinks. Based on the flexibility and variety of models in this line, 2950's can effectively be used at the access, distribution, and core layers, depending upon the size and needs of a network. The higher-end models in this line provide enhanced security, availability, and quality of service (QoS) features allowing them to also be deployed as enterprise edge devices. Cisco-specific design methodologies, and concepts like the enterprise edge, will be look at in later chapters.

Cisco Catalyst 3550 Series

The Cisco Catalyst 3550 series of switches are primarily access and distribution layer models, although they can also be used to provide high-speed core layer connectivity for smaller networks. This series again provides a wide-variety of models with different port configurations as discussed shortly. 3550 series switches are stackable, and provide multilayer switching at Layers 2 through 4. In other words, these models are capable of switching Layer 2 and 3 traffic, as well as making forwarding decisions based on Layer 4 application information.

Capabilities of models in this line include advanced QoS features, high-performance IP routing, the ability to control and limit traffic rates, and more. The individual models in this line offer a diverse set of port configurations and combinations. For example, the 3550-12G model offers 10 GBIC slots and 2 10/100 Mbps Fast Ethernet ports for maximum flexibility, while the 3550-48 model provides a more traditional 48 10/100 Mbps Fast Ethernet ports and 2 GBIC slots.

Cisco Catalyst 4000/4500 Series

The Cisco Catalyst 4000 and 4500 series switches are modular models that use a chassis-based design to promote maximum flexibility. Both models provide multilayer switching capabilities at Layers 2, 3, and 4, and are a key component of Cisco's Architecture for Voice, Video, and Data (AVVID) that supports higher-end network services such as Voice over IP (VoIP), QoS, and bandwidth management. The chassis design of the 4003 and 4006 models provide 3 and 6 slots respectively for add-in modules such as a Supervisor Engine (for routing functionality) and a variety of different of different port combinations. For example, the Catalyst 4006 is capable of providing up to 240 10/100/1000 Mbps ports (as well as multiple depending upon the modules installed. Another key benefit of chassis-based switches is that they tend to reduce cost of ownership since individual modules can be added, removed, or changed as needs do.

The Catalyst 4500 series is Cisco's next generation of 4000 series switches available in 7-, 6-, and 3-slot chassis-based models. Cisco positions these models as enterprise access and distribution layer switches, as well as for branch office connectivity using optional WAN modules. Higher-end 4500 models are also aimed at services aggregation and subscriber access for service providers.

Cisco Catalyst 5000/5500 Series

The Cisco Catalyst 5000 and 5500 series switches also follow a chassis-based design, and office an incredibly wide variety of modular interface cards. These Layer 2 and 3 multilayer models support modules that include traditional Fast Ethernet, Gigabit Ethernet, Token Ring, Copper Distributed Data Interface (CDDI), Fiber Distributed Data Interface (FDDI), and ATM interfaces. The Catalyst 5000 provides 5 chassis slots, while the 5502 provides 2. In the 5500 series, the 5500 model provides 13 slots, while the 5509 provides 9. Based on the varied capabilities of these series' models they may be found deployed as a high-density access layer solution, a flexible distribution layer aggregation method, and of course as a high-speed core layer platform.

Cisco Catalyst 6500 Series

The Cisco Catalyst 6500 series of multilayer switches follow a chassis-based design similar to both the 4500 and 5500 series switches look at in the previous sections. Also providing an astounding range of modules that provide different port densities and service capabilities, this multilayer switch is considered another core part of Cisco's AVVID solution. Interesting modules available for the 6500 series include 10 Gbps Ethernet ports, Firewall and VPN modules, powered ports for IP telephones and more. Cisco positions the 6500 series as a solution that can be deployed in enterprises anywhere from the access through to the core layer, but various models in the 6500 are becoming increasing common in service provider networks due to their high speed and throughput capabilities.

CCNA Study Guide Chapter 3 Summary

By Dan DiNicolo, May 29th, 2006 Posted in **CCNA Study Guide Chapter 03.** Subscribe to our

RSS Feed

This chapter began with a review of Layer 2 switching. This included a look at how a switch segments a network into a number of smaller collision domains, as well as how a switch makes full duplex communication possible.

This was followed by a look at the different switching methods supported by Cisco including Store-and-Forward, Cut-Through, and the default for the Cisco 1900 series, FragmentFree.

A look at redundancy and loop avoidance explained the benefits of having multiple paths on a bridged or switched network, but also the potential problems this introduces, namely broadcast storms. The process by which a broadcast storm happens was also discussed step by step.

Loop avoidance for bridged networks was covered in an overview of the Spanning Tree protocol. The process by which switches elect a Root Bridge, choose Root Ports and Designated Ports were also illustrated with examples. The four Spanning Tree port states – Listening, Learning, Blocking, and Forwarding – were also discussed.

A look at VLANs provided an overview of the benefits they provide in segmenting a network into multiple broadcast domains in a switched environment. The VLAN Trunking Protocol (VTP) was introduced as a way to easily configure VLANs on switches across a VTP management domain. VTP Pruning was also discussed.

Trunking techniques included a look at tagging protocols such as Inter Switch Linking (ISL), a Cisco proprietary protocol for VLAN identification.

Chapter 4: Network Protocols

Introduction to Network Protocols

In Chapters 1 through 3 we mainly concentrated on Physical and Data Link layer technologies, protocols, and specifications. These characteristics of LANs (and WANs) are absolutely essential to understand. However, as we move closer to the world of routers and routing, we have to start looking at the protocols that ride on top of these technologies. While a variety of network protocols exist for the purpose of moving data across an internetwork, three in particular make up the bulk of the implementations that you'll come across. These include TCP/IP, IPX/SPX, and AppleTalk.

The first important thing to understand is that none of these protocols define a single entity. Instead, each represents what is known as a protocol suite. A protocol suite is actually a group of protocols that work together (at different layers) to make network communication possible. If you recall from Chapter 1, most network protocol suites do not map to the OSI model directly, but include protocols that can generally be mapped to defined OSI layers. Some suites (such as AppleTalk) map to the OSI model more clearly than others. In this chapter we'll not only look at the protocols that make up the TCP/IP, IPX/SPX, and AppleTalk suites, but also relate these protocols to the layers of the OSI model.

The material to be covered in this chapter includes:

- Transmission Control Protocol / Internet Protocol (TCP/IP)
- Internetwork Packet Exchange / Sequenced Packet Exchange (IPX/SPX)
- AppleTalk
- Scalability issues for Network Layer protocols

TCP/IP

The Transmission Control Protocol/Internet Protocol is by far the most popular protocol suite in use today, thanks to the growth of the Internet. TCP/IP has changed a great deal over the years, as different protocols were created or adapted to address different needs. For example, TCP was originally developed back in 1974, while TCP and IP came together in 1978. It wasn't until January 1st, 1983 that all systems on the Internet (then known as the ARPANET) officially had to use TCP/IP. The groups who set the direction for the Internet (and decide what will become a standard) are the Internet Engineering Task Force (IETF) and the Internet Engineering Steering Group (IESG).

Before we go any further, you should be aware of what are known as Requests For Comments (RFCs). Before anything becomes an Internet standard, it has to go through the RFC process. While all Internet standards are defined in RFCs, not all RFCs become standards. If you take the time to read RFC 1149 or 1438, you'll understand what I mean. It's also worth noting that RFCs can be superseded by newer versions. If a standard is changed, a new RFC document is created, and a new number is associated with it. An RFC contains the definitive information for any Internet standard. If you're just starting out in the world of networking, get in the habit of using RFCs when you need clarification. You can be sure that the information they provide outlines the facts and they are a freely available resource, too. Just make certain that you're reading the most current version.

Study Tip: One of the best resources for searching RFCs is http://www.rfc-editor.org/rfcsearch.html. The results supplied by this page outline RFC status information, as well as whether a particular RFC has been updated or is now obsolete.

A number of key protocols make up the TCP/IP suite, some of which we looked at briefly in Chapter 1. In this chapter we'll go into more detail on each. A solid understanding of the protocols that make up the TCP/IP protocol stack is not only imperative for the exams, but also in real life. Troubleshooting any network is relatively simple when you truly understand how communication processes take place.

TCP/IP and The OSI Model - Internet (Network) Layer Protocols

The TCP/IP protocol stack is comprised of protocols that exist at the upper three layers of the TCP/IP model (the lower layer is the Network Interface layer, and is responsible for network technologies that we've already looked at such as Ethernet, Token Ring, and so forth). The main protocols that can be found at each layer have different roles and responsibilities, and having an appreciation of how they work (and their purposes) is imperative. Figure 4-1 outlines the protocols that we're going to look at in this section, and how they map to the OSI model.

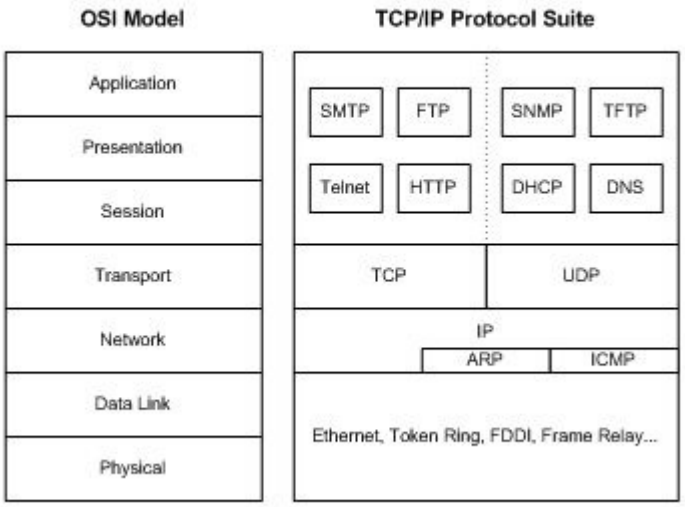


Figure 4-1: TCP/IP protocols and the OSI model.

Network (Internet) Layer

If you recall, the Internet layer's primary responsibilities are determining a path between networks (routing), as well as network addressing. The addressing that takes place at the Internet layer is often referred to as logical addressing. These addresses aren't "burned-in" like Ethernet MAC addresses, but instead are assigned by an administrator. The addressing protocol of the TCP/IP stack is the Internet Protocol (IP).

Note that TCP/IP routing protocols such as RIP, OSPF, and others also exist at the Internet layer. These will be looked at in Chapter 8, when routing is covered in detail.

Internet Protocol

IP is a connectionless protocol. As such, it doesn't request acknowledgements for data sent. Instead, it relies on upper-layer protocols (like TCP) to handle reliability functions. IP addresses are probably something that you're already familiar with. More likely than not, you've assigned an IP address to a computer for the purpose of connecting it to the Internet, your office, or something similar. An IP address is most often displayed in what is referred to as dotted-decimal notation, for example 172.16.0.1. IP addresses are made up of two main parts - the first uniquely identifies a network, and the second uniquely identifies a host on that network. Think of this as being similar to a street address. Your house (the host) has a unique address on your road (the network).

You may also be familiar with the fact that IP uses 32-bit addresses. But what does this mean? Simple, computers do just about everything in binary, representing information as a series of 0's and 1's. Since an IP address is 32 bits long, it can also be represented as a series of 32 1's and 0's. For example, the IP address 192.168.0.1 can also be represented as:

11000000 10101000 00000000 00000001

Don't worry just yet about how those numbers are generated. By the time you get through Chapter 5, I promise it will seem like the easiest thing in the world.

You're probably also familiar with something called a subnet mask. The purpose of a subnet mask is to define which portion of an IP address represents the network, and which portion represents the host. For example, if I have an IP address of 10.1.1.1 and a subnet mask of 255.0.0.0, it means that the first eight bits (referred to as the first octet) represent the network, while the last 24 bits represent a host. In this case, the network ID is 10, while the Host ID is 1.1.1. In Chapter 5 we'll delve much further into how masks work and how they're created. For now, it's important that you simply understand that every IP address requires a subnet mask to be able to distinguish between the network and host portions.

Every IP network requires a unique network ID, and each host on a network requires a unique host ID. That should be simple enough for now - an IP address simply gives us information as to the network on which a unique host can be found.

In order to communicate with a host on a different network, communication requires the use of routing. IP is what is referred to as a routed or routable protocol. That simply means that if a network is configured correctly, a host on one network will be able to communicate with a host on another, with routers acting as intermediaries deciding where a packet should be sent next. We'll take a look at routing protocols in detail in Chapter 8.

Data from upper layers is passed down to IP during the encapsulation process. When IP obtains data from upper-layer protocols, it adds an extra header prior to passing it down to lower layer technologies (like Ethernet) for framing. An IP header actually contains a variety of useful information that helps a packet move across a network. One note here - it has been my experience that people often choose to ignore the importance of the information found in headers when they're just starting out in networking. Be sure to take the time to examine the headers diagrammed in this book, and read the explanation of what the various fields are used for - ultimately, it will help you to better understand network communications.

Figure 4-2 outlines the fields found in an IP header, and their responsibilities.

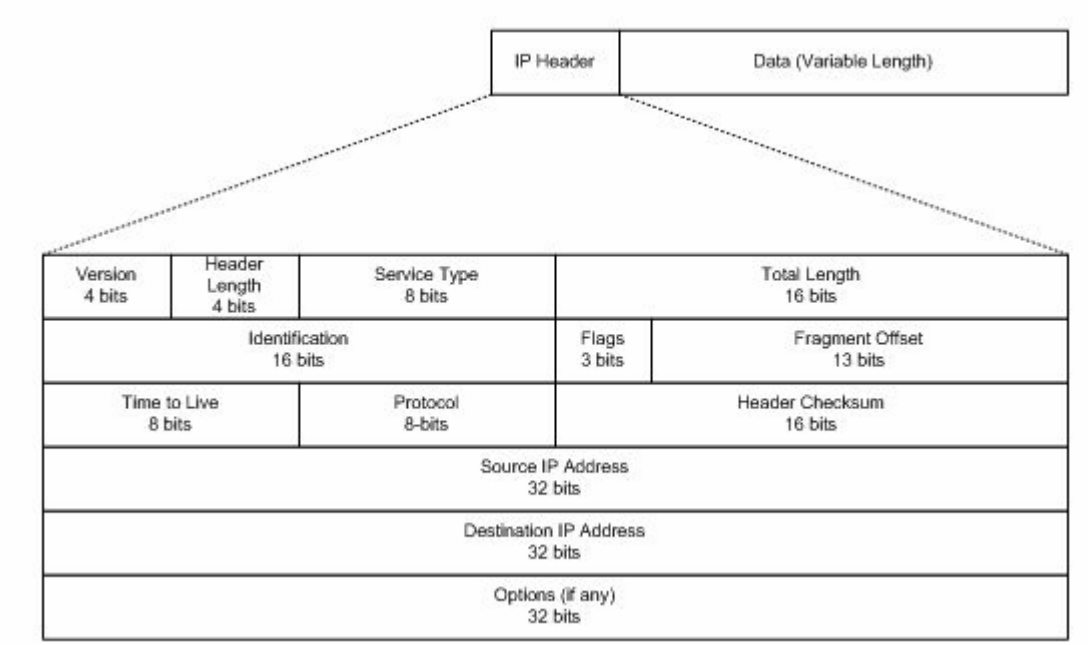


Figure 4-2: IP header.

The IP header fields are described below:

- **Version**. This field specifies the version of IP in use. The current version is IP version 4.
- **Header Length**. This field specifies the length of the IP header. It actually groups fields into 32-bit numbers. So, if the total header length were 160 bits, the Header Length field would specify 5 (160 divided by 32).
- **Service Type**. Often referred to as the Class of Service or DiffServ field, bits can be set here to define the priority level of a packet.
- **Total Length**. Specifies the total length of a packet, including the data field.
- **Identification**. A value to uniquely identify a particular packet
- **Flags**. This field specifies whether fragmentation is allowed to occur. In some cases, such as when a packet is routed from one network type to another, fragmentation (breaking the packet up into a number of smaller packets) may be necessary when the underlying technologies allow different maximum packet sizes.
- **Fragment Offset**. This field provides the information required to reassemble a fragmented packet.
- **Time to Live**. Specifies how long, in seconds, the packet is allowed to exist on a network. When a router processes a packet, it must decrement the TTL by 1. At a TTL of 0, a packet is removed from the network.
- **Protocol**. This field specifies the upper-layer protocol to which this packet belongs. For example, TCP is defined by protocol number 6, while UDP is protocol number 17.
- **Header Checksum**. A computed value that verifies the integrity of the IP header, similar to a CRC.
- **Destination Address**. The destination IP address for the packet.
- **Source Address**. The source IP address of the packet.
- **IP Options**. This field can be used to specify different options such as security restrictions, router alerts, path specification, and so on.

- **Data**. This field contains the encapsulated data passed down from upper-layer protocols and applications.

While you don't necessarily need to memorize the fields listed above for your exams, having an understanding of their purpose and functions will go a long way towards better understanding of IP communications.

Study Tip: For more information on Internet Protocol, see RFC 791.

Address Resolution Protocol (ARP)

ARP is another protocol found at the Internet layer of the TCP/IP model. As the name suggests, ARP is a protocol used to resolve addresses - in this case, finding the MAC address that corresponds to an IP address. Remember that data is encapsulated before being sent over the network. Even though a system may know the IP address it wants to ultimately send data to, it may not know the MAC address. On an Ethernet LAN, systems communicate directly using CSMA/CD, and as such must know the MAC address of the system that data must be sent to next.

Recall that MAC addresses are fixed. IP addresses, on the other hand, are not. Systems can be manually configured with an IP address, or they can obtain one using the Dynamic Host Configuration Protocol (DHCP). As such, it doesn't make much sense to have a static mapping between the two, since IP addresses may change. Instead, when a system needs to obtain the MAC address associated with an IP address, it sends out a broadcast message asking that the system with the specified IP address reply with its MAC address. Once it receives a reply, the answer is cached for a limited period of time (typically between 2 and 20 minutes) in the system's ARP table. Caching helps to ensure that ARP broadcasts don't get out of hand and overwhelm the network.

Since an ARP request is a broadcast, it will be seen by every system in the same broadcast domain. When a system comes across an ARP request, it will check to see if it is the intended recipient. If it is, the system will process the frame. If not, the frame is ignored.

When a system wishes to communicate with another system in a different broadcast domain, recall that the packets must be sent to a router. In this case, the ARP request isn't looking for the MAC address of the destination host (since the broadcast would never reach it), but instead the MAC address of the router interface to which the frame must first be forwarded.

The format of an ARP packet is relatively simple. Because ARP messages are created at the Internet layer, they only include IP information and framing for the underlying technology such as Ethernet. The structure of an ARP message is shown in Figure 4-3. Note that fields can be different sizes based on the technologies or protocols with which ARP is used.

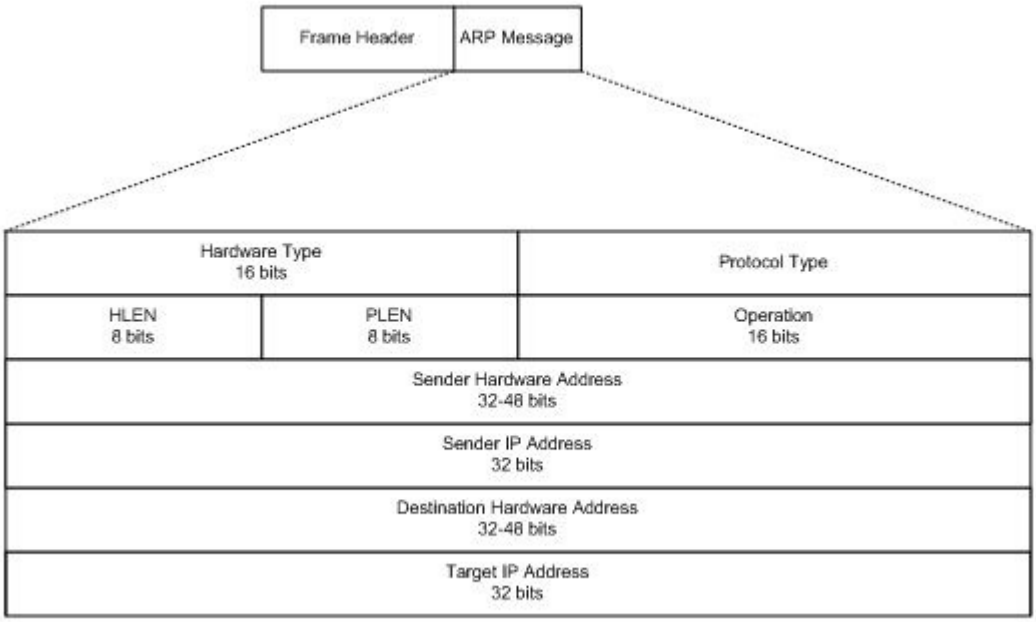


Figure 4-3: ARP Message.

The ARP message fields are described below:

- **Hardware Type**. Specifies the hardware interface type, for example Ethernet.
- **Protocol Type**. Specifies the upper-layer protocol in use, usually IP.
- **Hardware Address Length (HLEN)**. Specifies the length of the hardware address is use. This allows for flexibility with different network technologies.
- **Protocol Address Length (PLEN)**. Specifies the length of the address of the upper-layer protocol in use.
- **Operation**. Specifies the type of request, such as ARP or RARP.
- **Sender Hardware Address**. The MAC address of the sending system.
- **Sender IP Address**. The IP address of the sending system.
- **Target Hardware Address**. The MAC address of the target.
- **Target IP Address**. The IP address of the system whose MAC address is being sought.

You should also be aware of another protocol related to ARP, referred to as a Reverse ARP (RARP). RARP is used by diskless workstations to obtain an IP address when they boot, based on their MAC address. These requests are heard by a RARP server, who will allocate an IP address based on the sender MAC address contained in the request.

Study Tip: For more information on Address Resolution Protocol, see RFC 826

Internet Control Message Protocol (ICMP)

ICMP is yet another protocol at the Internet layer. Internet Control Message Protocol is used as an error reporting protocol in the TCP/IP suite. It is important to begin by noting that ICMP does nothing to make IP reliable. Instead, it simply reports on error situations that exist or occur. ICMP only sends error messages back to the originating host, and not intermediary devices. A good example of a type of error message sent by ICMP is a Source Quench message. These are used when a router receives more data than it can handle. Once its buffers finally fill, the router will send the source host a Source Quench message, effectively letting the source

know that it should send data at reduced rate. Note that the message doesn't include any information on packets that may have been lost. Instead, it simply makes the sender aware that a situation exists. It is still the responsibility of upper-layer protocols to ensure that data arrives at its destination.

You are probably more familiar with ICMP than you realize. Have you ever used the ping utility? If so, what you've actually done is sent out ICMP echo messages, and if the address that you attempted to ping was reachable, you received back ICMP echo replies. Ping is an example of a simple ICMP utility that provides information on whether or not hosts can be reached. If the host you're attempting to ping can't be reached, you'll receive an ICMP Destination Unreachable message. ICMP is also used to send messages that notify the sender that the Time to Live (TTL) on a packet has expired.

Consider the example in Figure 4-4, which shows a capture of an ICMP echo reply. Note that only the ICMP portion of the frame is expanded.

Ethernet II
Internet Protocol, Src Addr: 192.168.1.200 (192.168.1.200), Dst Addr: 192.168.1.21 (192.168.1.21)
Internet Control Message Protocol
Type: 0 (Echo (ping) reply)
Code: 0
Checksum: 0x3b5c (correct)
Identifier: 0x0200
Sequence number: 18:00
Data (32 bytes)

Figure 4-4: ICMP Echo Reply.

Remember that ICMP is simply a reporting protocol - it does nothing to actually remedy any errors that occur.

Study Tip: For more information on Internet Control Message Protocol, see RFC 792

TCP/IP and The OSI Model - Transport (Host-to-Host) and Application (Process) Layer Protocols

If you recall from Chapter 1, we've already spent some time looking at connection-oriented and connectionless protocols. At the Host-to-host layer of the TCP/IP model, two primary protocols exist - Transmission Control Protocol (TCP) and User Datagram Protocol (UDP).

Transmission Control Protocol (TCP)

TCP is the connection-oriented transport protocol in the TCP/IP protocol suite. TCP attempts to make connections reliable through the use of positive acknowledgement with retransmission - a system where acknowledgements are required for all data sent. If the sending host does not receive these acknowledgements, retransmission eventually occurs. Sequence numbers are assigned to segments to be sure that they are assembled in the correct order at the receiving system. Flow control is handled through buffering techniques such as the use of sliding windows.

When two systems wish to communicate via TCP, they first establish a session (remember the 3-way handshake?), also known as a connection or virtual circuit. Once data has been transmitted and the session is complete, it is closed via a modified 3-way handshake - the difference is that the connection needs to be closed (and acknowledged) independently by both systems. Figure 4-5 outlines the 4 steps required to close a TCP session. Closing a session involves sending FIN (no more data to send) and ACK messages. Notice that the extra step involved - the system that originated the end of the connection acknowledges that the other system is now closing the connection as well.

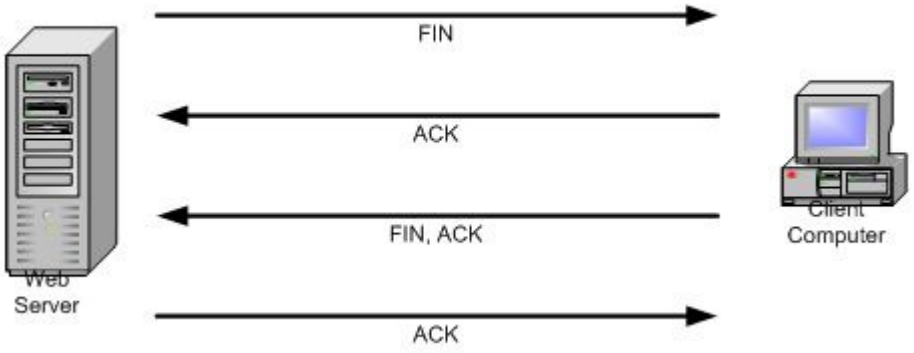


Figure 4-5: Ending a TCP session.

There can be a delay between the time when one system attempts to close a connection and the time that the entire connection is actually closed. This occurs because one end may close the connection, while the application on the other end might still be running. A good example is when a user accesses a website - the user may keep their browser running long after the web server has finished sending the requested data as has attempted to close the connection.

TCP communicates with upper-layer applications and protocols through the use of port numbers. When combined with an IP address, a port number defines a unique endpoint for communication. There are 65,536 port numbers available to TCP. How they are chosen requires a bit of discussion.

For the most part, server-side applications are defined by what are considered to be well-known port numbers. These port numbers are defined in RFC 1700 and fall into the 0-1023 range. For example, when a client opens a web browser and types in http://www.2000trainers.com, that name will be translated into an IP address using DNS, and then a connection will be made to port 80 on that IP address. TCP port 80 is the default port on which a web server waits for a connection. But what about the client side of the connection? Well, the client needs to specify a port on which it will send and receive data as well. These port numbers are usually in the range above 1024, and are dynamically assigned by the source host.

Consider the example packet capture in Figure 4-6. This request shows an HTTP connection between a client and a web server. Notice the TCP header portion of the frame. It specifies that the destination port is 80 (the web server) and the source port is 4773 (the client web browser).

Ethernet II
Internet Protocol, Src Addr: 192.168.1.21 (192.168.1.21), Dst Addr: 192.168.1.200 (192.168.1.200)
Transmission Control Protocol, Src Port: 4773 (4773), Dst Port: 80 (80), Seq: 1935801179, Ack:

2272844446
Source port: 4773 (4773)
Destination port: 80 (80)
Sequence number: 1935801179
Next sequence number: 1935801513
Acknowledgement number: 2272844446
Header length: 20 bytes
Flags: 0x0018 (PSH, ACK)
Window size: 17520
Checksum: 0xf4e3 (correct)
Hypertext Transfer Protocol

Figure 4-6: TCP header for an HTTP request.

Remember that a connection is uniquely identified by a combination of both the client and server port numbers and IP addresses. In this case the connection is uniquely identified as:

(192.168.1.21 port 4773) and (192.168.1.200 port 80)

Many simultaneous connections between different clients and a single server are possible. The server will keep track of where data needs to be sent according to the connection information.

Study Tip: Sometimes "seeing" connections can help you to better understand the network communication process. If you're running Windows on your workstation, open a command prompt and type netstat.exe to view current connection information.

Table 4-1 outlines some of the more common TCP port numbers that you should be familiar with.

Table 4-1: Common TCP port numbers.

Application	TCP Port Number
FTP	20, 21
Telnet	23
SMTP	25
HTTP	80
POP3	110

Study Tip: For a complete list of well-known port numbers see http://www.iana.org/assignments/port-numbers. This list obsoletes the previous list found in RFC 1700.

It's worth noting that some applications actually use more than one port number. With FTP, for example, port 21 is used as the control channel, where username and password information is passed when a connection is attempted. Port 20 is then used for the actual transfer of data.

Server side port numbers can usually also be custom configured. For example, you could set up a web server to answer on TCP port 2222. The only issue is that users would have to be aware of the port number and explicitly state it when attempting to connect from their web browser, which will automatically assume port 80 for an HTTP request. In this case, a user would need to connect to http://www.2000trainers.com:2222 - the number after the colon explicitly specifies the port number.

As you'll see shortly, a TCP header carries significantly more information than its UDP counterpart. This is a result of the extra fields that are required as part of making TCP reliable. It is also the reason why you'll hear it said that TCP as a protocol has higher overhead. The diagram below outlines the fields found in a TCP header.

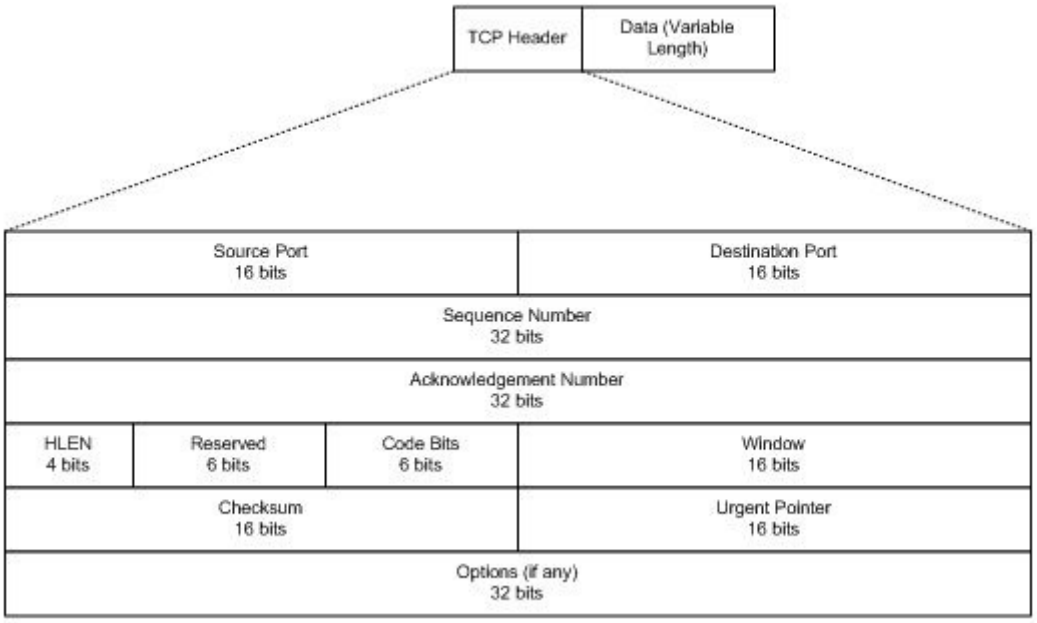


Figure 4-7: TCP header.

The TCP header fields are described below:

- **Source Port**. This field specifies the TCP port number of the system sending the segment.
- **Destination Port**. This field specifies the TCP port number of the destination system.
- **Sequence Number**. The ordered number of the segment that is used to ensure that data can be properly reassembled on the receiving system, since segments may arrive out of order.
- **Acknowledgement Number**. The acknowledgement number is used to let the other system know that a segment was received.
- **HLEN**. The header length field specifies the total TCP header length, again grouped into 32 bit numbers.
- **Reserved**. This field is always set to zero, and is reserved for future use.
- **Code Bits**. This field specifies the purpose of the segment. For example, it can be used to denote acknowledgements (ACK), closing a connection (FIN), a synchronization request (SYN), urgent data (URG) and so forth
- **Window**. Acts as the window size advertisement, specifying how much data this system is currently willing to receive.
- **Checksum**. A computed value that verifies the integrity of the TCP header, similar to a CRC.
- **Urgent Pointer**. When the URG code bit is set, this field specifies where within the segment the urgent data can be found.
- **Option**. If used, this field specifies the maximum segment size that can be received.
- **Data**. The encapsulated data passed down from upper-layer protocols and applications.

Study Tip: For more information on Transmission Control Protocol, see RFC 793

User Datagram Protocol (UDP)

In comparison to TCP, UDP is a very simple protocol. Recall that UDP is connectionless, and as such doesn't have any reliability mechanisms built in. Because of this, UDP relies on upper layer applications and services for reliability - UDP itself does nothing to attempt to make the communication process reliable. You should also remember that this isn't UDP's fault - as a protocol it was designed with speed in mind. If a reliable connection is required, applications should be programmed to use TCP.

Much like TCP, UDP also makes use of 65,536 port numbers to define communication endpoints. Similar to TCP, port numbers below 1024 are considered to be well defined and apply to servers, while client port numbers are usually dynamically assigned in the same way as with TCP. If you were to take a look at the list of port numbers on the IANA website, you would notice that both TCP and UDP port numbers are assigned to most services. For the most part, a service will regularly use one of TCP or UDP to communicate, and seldom both. Certain exceptions apply, such as with DNS. For regular DNS queries, UDP port 53 is used. However, for large queries and zone transfers, DNS uses TCP port 53.

Table 4-2 outlines some common UDP port numbers:

Table 4-2: Common UDP Port numbers.

Application	UDP Port Number
DNS	53
DHCP	68
TFTP	69
SNMP	161

Figure 4-8 outlines the fields found in a UDP header. While a TCP header is comprised of 192 bits of information, a UDP header is considerably smaller at only 64 bits (not including data, of course).

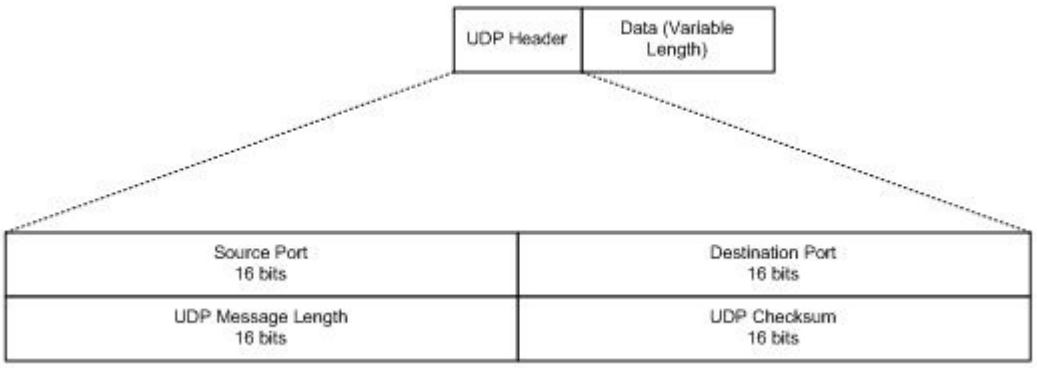


Figure 4-8: UDP Header.

Study Tip: For more information on User Datagram Protocol, see RFC 768

Application (Process Layer)

The Application/Process Layer is where TCP/IP applications and services reside. You're more than likely familiar with many of these, since you probably interact with many TCP/IP applications on a daily basis – a web browser using HTTP, or your email client connecting to a POP3 server are but two simple examples.

The list below outlines some of the more common Application layer protocols that you should be familiar with:

- **Telnet**. Telnet is used to create a terminal session with a remote host, providing command-line access to the target system running a telnet server (daemon).
- **FTP**. The File Transfer Protocol is used to reliably transfer files between an FTP client and server using TCP.
- **SMTP**. The Simple Mail Transfer Protocol is used for the exchange of email between systems.
- **DNS**. The Domain Name Service is a distributed database that is queried to resolve (or translate) names such as www.2000trainers.com to an IP address.
- **SNMP**. The Simple Network Management Protocol is a lightweight network protocol that allows information to be gathered about network devices. Examples include information about utilization, hardware configuration, and so forth.
- **TFTP**. The Trivial File Transfer Protocol is used to transfer files between a client and a TFTP server over UDP. You'll learn more about TFTP later, since it's the protocol used to transfer files to and from a Cisco router.

The Netware Protocol Suite

Similar to TCP/IP, the Netware or IPX/SPX protocol suite is actually made up of a number of protocols that serve different purposes in the network communication process. While the IPX/SPX suite has become less popular based on the widespread adoption of TCP/IP, it is still in use in many (especially larger) network environments.

The IPX/SPX suite comes from the world of NetWare, Novell's popular network operating system (NOS). Up until NetWare 4.x versions, Novell servers primarily relied on IPX/SPX as their native networking protocol suite. However, versions as early as 3.1x are capable of IP-based communication via a loadable module called NetWare/IP that encapsulates IPX/SPX traffic in UDP datagrams. Beginning with version 5.0, NetWare began using TCP/IP as its primary network protocol, although support for IPX/SPX continues into current versions. In order to reduce the complexity of networks, most companies now deploying NetWare are choosing to use TCP/IP only. However, the use of older IPX-based servers continues to make IPX/SPX a requirement on many networks. For the purpose of both the CCNA and CCDA exams, you are expected to be familiar with IPX/SPX, although you will definitely find a much larger emphasis placed on TCP/IP.

NetWare and the OSI Model

The NetWare protocol suite also follows a layered design that can be roughly mapped to the OSI model. Figure 4-9 outlines some of the more common protocols that make up the suite. We'll look at these in more detail in this section.

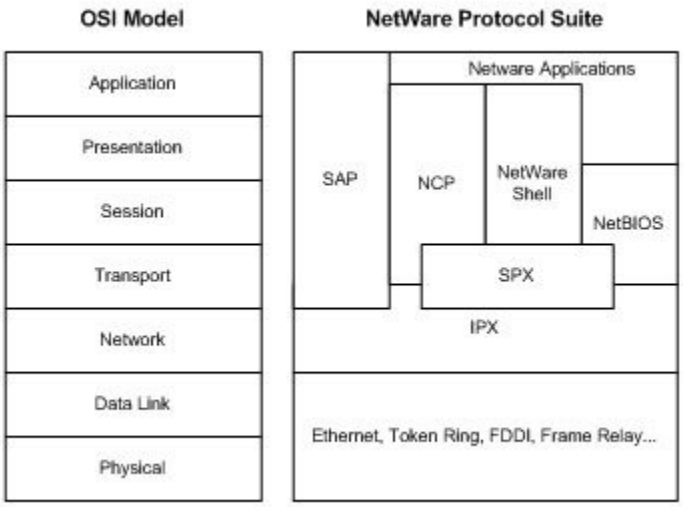


Figure 4-9: NetWare Protocol Suite and the OSI Model.

Physical and Data Link Layers - Frame Encapsulation

NetWare protocols can run over a variety of different network technologies, similar to TCP/IP. For the purpose of keeping things simple, in this section we'll concentrate on Ethernet. However, IPX can also be run over Token Ring, FDDI, ATM and a variety of WAN technologies. If you recall, Novell originally defined the 802.3 frame type to be used over Ethernet networks. These frames lacked a Type field, since all 802.3 frames were assumed to be destined for IPX at the Network layer. The 802.3 frame type is commonly referred to as Novell Ethernet 802.3 or Ethernet Raw. Different versions of NetWare use different Ethernet frame types as their default for encapsulating IPX packets. In NetWare 3.11 and earlier versions, the default frame type was 802.3. Beginning with Netware 3.12, the default frame type was changed to 802.2.

This presents a technical challenge, since different frame types are not interoperable. As such, a system configured to use only 802.3 frames cannot communicate with a system in the same broadcast domain that uses only 802.2 frames.

To help overcome these limitations, a Cisco router can be configured to use more than one Ethernet encapsulation type for IPX on a single interface. This allows a router to communicate with systems using different frame types. We'll look at the actual configuration of IPX on a router in Chapter 8. For now, it is sufficient to know the different frame types support by Cisco, and the terms used to describe them.

- **novell-ether**. Refers to 802.3 frames that lack an LLC header.
- **sap**. Refers to the IEEE standard Ethernet frame with an 802.2 LLC header.
- **arpa**. Refers to the Ethernet II frame type.
- **snap**. Refers to the Ethernet SNAP frame type.

Network Layer - Internetwork Packet Exchange (IPX)

IPX is the routable Network layer protocol of the Novell IPX/SPX protocol suite. Similar to IP, IPX is also connectionless, meaning that it leaves reliability to upper-layer protocols such as SPX. IPX addresses are used to uniquely identify hosts on an IPX network. While an IP address is comprised of 32 bits and is usually represented in dotted-decimal notation, an IPX address is an 80-bit address, represented in hexadecimal.

An IPX address is also made up of two parts - a network address and a host address. The network address is represented by the first 32-bits, while the unique host address is comprised of the final 48 bits. The example below outlines an IPX address.

0000000A.0101AE560163AC

An administrator defines the network portion of an IPX address. Leading 0's can be left off the network portion. Because of this, the network above can be more simply referred to as network A. Like IP networks, every IPX network requires a unique network address.

TIP: Remember that the network portion of an IPX address is represented in hexadecimal. Since each hex digit represents 4 bits, it can be comprised of 8 hex digits maximum. Don't forget that only the characters 0-9 and A-F are valid in hexadecimal.

The host address portion of an IPX address is actually incredibly simple - it's the MAC address of a given host. Since all MAC addresses are different, the combination of a network number and a MAC address uniquely identifies a host on a given IPX network. Note that this also eliminates the need for a protocol such as TCP/IP's ARP, since a host's MAC address is already contained within the full IPX address.

Figure 4-10 outlines the fields with an IPX header. Note that this particular diagram, field sizes are displayed in bytes rather than bits.

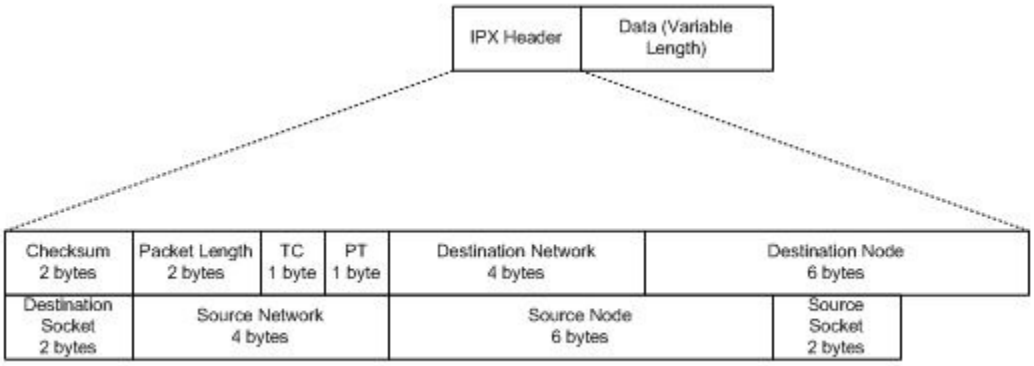


Figure 4-10: IPX Header.

The IPX header fields are described below:

- **Checksum**. A computed value that verifies the integrity of the IPX header, similar to a CRC. This was not used in versions of NetWare prior to 4.0, where it was set to all 1's.
- **Packet Length**. This field specifies the complete length of the IPX packet, including data.

- **Transport Control**. Similar to a TTL (Time-To-Live), when a router processes a packet, it must increment this TTL by 1. At 16, a packet is removed from the network.
- **Packet Type**. This field specifies the upper-layer protocol that data is to be passed to. Common hexadecimal values include 4 (SAP), 5 (SPX), 17 (NCP), 20 (NetBIOS)
- **Destination Network**. The 32-bit destination network address.
- **Destination Node**. The 48-bit destination node address.
- **Destination Socket**. This field specifies the software process address on the destination system, similar to a TCP port number.
- **Source Network**. The 32-bit source network address
- **Source Node**. The 48-bit source node address.
- **Source Socket**. Specifies the software process address on the sending system.
- **Data**. The encapsulated data passed down from upper-layer protocols and applications.

Routing protocols such as RIP and NLSP also exist at the Network layer of the NetWare protocol suite. These will be covered in detail in Chapter 8.

Transport Layer - Sequenced Packet Exchange (SPX)

Sequenced Packet Exchange (SPX) is the reliable transport protocol used by IPX when connection-oriented communication is required. Much like TCP, SPX uses sequence numbers and acknowledgements to be sure that data is reliably passed between hosts. Beginning with NetWare 4.0, a new (and backwards-compatible) version of SPX called SPX II was released, providing a sliding window flow-control mechanism. You should be aware that not all IPX network communication requires the use of SPX. IPX is also capable of interacting with some of the other upper-layer protocols directly.

Upper Layers

The IPX/SPX protocol suite isn't nearly as orderly as others when it comes to mapping upper layer protocols to the OSI model. Many of the protocols found above the network layer span multiple OSI layers between Transport and Application. The mapping of the suite to the OSI model at the beginning of this section helps to illustrate this. While the actual mappings may not be so clear-cut, the purpose of these protocols is what is most important. We'll take a look at a number of these here including SAP, NCP, NetWare Shell, and NetBIOS.

Service Advertising Protocol (SAP)

In older version of NetWare (those prior to version 4.0), servers made other systems aware of the services that they offered through the use of Service Advertising Protocol (SAP) broadcasts. For example, a NetWare server might be functioning as a both a file and print server. In this case, the server would broadcast SAP advertisements onto the network, making clients aware that these services are offered at its network address. SAP advertisements are also used by clients to find their nearest server at startup, in the form of a Get Nearest Server (GNS) request. SAP identifiers are hexadecimal numbers that represent the services. Common examples of SAP identifiers are listed in Table 4-3.

Table 4-3: Common SAP Identifiers.

Service	SAP Identifier (hex)
File Server	0x4

Gateway	0x6
Print Server	0x7

Servers send out SAP broadcasts every 60 seconds by default. Since these advertisements are broadcasts, they are limited in scope to the local broadcast domain. For this reason, clients in other broadcast domains will not hear these advertisements. To account for this, Cisco routers can be configured with access lists that can selectively propagate or ignore SAP advertisements from different systems. Similarly, when a local NetWare router is not available, a Cisco router can be configured to respond to GNS requests. Access lists and their configuration will be looked at in detail in Chapter 9.

Newer versions of NetWare that use eDirectory or NetWare Directory Services (NDS) rely on SAP broadcasts to a lesser degree, since the location of servers can be found via a query to an NDS/eDirectory server.

NetWare Core Protocol (NCP)

The NetWare Core Protocol (NCP) is a set of procedures followed in IPX-based communication. When a client using the NetWare Shell attempts to access file, print, or directory resources on a NetWare server, NCP processes the request.

NetWare Shell

In a Windows environment, the line between client and server roles is blurred. NetWare clients, on the other hand, are those running another operating system such as Windows 2000. The NetWare Shell is additional redirector software installed on a client to allow it to make requests to NCP on a NetWare server. For example, when a client wishes to access a network drive, the request is passed to the NetWare shell, which creates an NCP request. It then passes the NCP data to IPX to be forwarded over the network. In this way, a client application is unaware that the request is actually for a remote resource.

NetBIOS

NetWare also provides an emulated environment in which NetBIOS applications can be run. NetBIOS is a broadcast-based Session layer protocol developed by Microsoft and IBM that uses names instead of numerical addresses to represent clients. NetBIOS traffic can be encapsulated in IPX packets to allow NetBIOS-based communication across an internetwork.

The AppleTalk Protocol Suite and Network Protocol Scalability

The AppleTalk protocol suite was developed in the early 1980's by Apple Computer to facilitate networking on their Macintosh computers.

Although AppleTalk was the primary protocol suite used on Macintosh-based networks up until the late 1990's, Macintosh systems have supported TCP/IP since MacOS 8.1, and all relatively recent models use TCP/IP as their primary protocol. As such, many companies no longer route AppleTalk traffic within their networks, choosing to rely on TCP/IP instead. So why discuss AppleTalk here? Well, the reason is twofold. First, you should be aware that

although it is declining in popularity, many large networks (especially those in the academic world) still use AppleTalk as they continue to run applications that support no alternative protocols. Second, you may still come across AppleTalk-related questions on the CCDA exam. Although much less emphasis is now placed on AppleTalk concepts, there is still the chance that you'll come across questions that reference Macs, so it's better to be safe that sorry.

Tip: Remember that using newer Macintosh models (such as an iMac, G4, or post-MacOS 8.1 system) does not explicitly require a network to support AppleTalk.

Two versions of the AppleTalk protocol suite exist - the earlier version was known as AppleTalk Phase 1, while the current version is AppleTalk Phase 2. The main differences between the two versions relates to scalability, which will be discussed shortly. Much like Novell's IPX/SPX protocol suite, AppleTalk has also become less popular as networks are migrated to TCP/IP.

The AppleTalk protocol suite actually maps quite nicely to the OSI model. Figure 4-11 outlines how the core protocols in the suite map to the 7 OSI Layers.

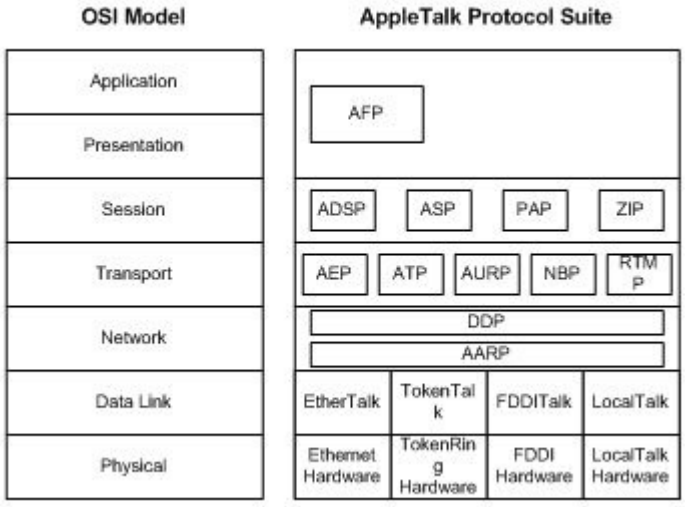


Figure 4-11: AppleTalk protocol suite and the OSI model.

AppleTalk Networks and Zones

Before getting into the protocols of the AppleTalk suite, we should first take a look at the concept of AppleTalk networks and zones. Two varieties of AppleTalk networks exist - these are known as extended and nonextended, and are described below:

- **Nonextended AppleTalk Network**. In AppleTalk phase 1, only nonextended networks existed. These are normally not used any more, mainly because they limit the network to a single physical segment. All nonextended networks are also limited to a single network number and a single AppleTalk zone.
- **Extended AppleTalk Network**. Available in AppleTalk Phase 2, an extended network allows multiple network numbers to exist, along with multiple AppleTalk zones. Because a given network number can only support a maximum of 253 nodes, it is also possible to configure multiple network numbers for a single physical segment on an extended network. This is referred to as a cable range. Almost all AppleTalk deployments today are based on extended networks.

But what is an AppleTalk zone? An AppleTalk zone is a logical grouping (or administrative unit) of AppleTalk resources, somewhat similar to a workgroup on a Microsoft network. With any Data Link protocol other than LocalTalk, zones can span multiple networks, or there can be multiple zones on a single network. On a LocalTalk network, only one zone can exist. Zones are commonly created according to functional areas or departments - for example, you might have a Finance zone or an IT zone. On a Macintosh computer, users can browse zone resources using the Chooser application.

Physical and Data Link Layers

Much like TCP/IP and IPX/SPX, AppleTalk can run over a variety of network technologies. Four main media-access specifications are implemented at the Data Link layer. These include:

- **EtherTalk**. To allow AppleTalk protocols access to Ethernet networks, the Data Link layer specifies a protocol referred to as the EtherTalk Link Access Protocol (ELAP). ELAP first encapsulates upper-layer data into a Ethernet SNAP frame, and into a standard 802.2 frame.
- **TokenTalk**. To allow AppleTalk protocols to access Token Ring / 802.5 networks, the Data Link layer specifies a protocol referred to as TokenTalk Link Access Protocol (TLAP).
- **LocalTalk**. LocalTalk is a proprietary protocol developed by Apple that was originally designed as a workgroup network technology. Local Talk uses a media access method somewhat similar to Ethernet referred to as Carrier Sense Multiple Access with Collision Avoidance (CSMA/CA). Unlike Ethernet, when nodes on a LocalTalk network wish to communicate, they first go through a handshaking process, which reserves the network for their exclusive use. Collisions can still occur if more than one pair of systems attempts to begin the handshake process at the same time. LocalTalk does not use static MAC addresses - the Data Link layer assigns these addresses dynamically. The protocol used at the Data Link layer is referred to as LocalTalk Link Access Protocol (LLAP)
- **FDDITalk**. To allow AppleTalk protocols to access FDDI networks, the Data Link layer specifies a protocol referred to as FDDITalk Link Access Protocol (FLAP).

Remember that the Data Link specification listed above are only used with the AppleTalk suite, and not when a Macintosh system uses other protocol stacks such as TCP/IP or IPX/SPX.

Network Layer

The AppleTalk Network layer includes protocols concerned with network addressing and routing. An AppleTalk network address is a 32-bit address and consists of three main parts - a network number, a node number, and a socket number. These are described below.

- **Network number**. A 16-bit number that uniquely identifies an extended or nonextended AppleTalk network.
- **Node number**. An 8-bit number that uniquely identifies a network node (host). A given network number is limited to supporting a maximum of 253 nodes.
- **Socket number**. An 8-bit number that uniquely identifies an upper-layer protocol interface for sending or receiving packets. Similar in function to a port number in TCP/IP.

AppleTalk addresses are usually displayed in dotted decimal notation. As such, node 100 on network 8 using socket 99 could be displayed as 8.100.99, or even 8.100, socket 99.

To reduce administrative effort, AppleTalk network addresses are dynamically assigned. When a node starts up, it gives itself a temporary address for the purpose of network communication. It

then uses the Zone Information Protocol (ZIP) to query a local router to find out the network numbers (cable range) available for its physical segment. After doing so, it assigns itself a node number, and broadcasts a message onto the network to see whether that node number is in use. If it isn't, the system will use that number. If it is, it will choose a different node number and try again.

The two main protocols found at this layer include the Datagram Delivery Protocol (DDR) and the AppleTalk Address Resolution Protocol (AARP). Network layer routing protocols used by AppleTalk will be discussed in Chapter 8.

Datagram Delivery Protocol (DDP)

The Datagram Delivery Protocol is the connectionless network layer protocol of the AppleTalk suite. It might be compared to IP or IPX, in that it makes reliable delivery the responsibility of upper-layer protocols. Two types of DDP packets exist, a long (extended) version and a short (nonextended) version. The short version is only used on nonextended networks, and does not include any network information (since only a single network can exist). The fields found in an extended DDP packet are shown in Figure 4-12.

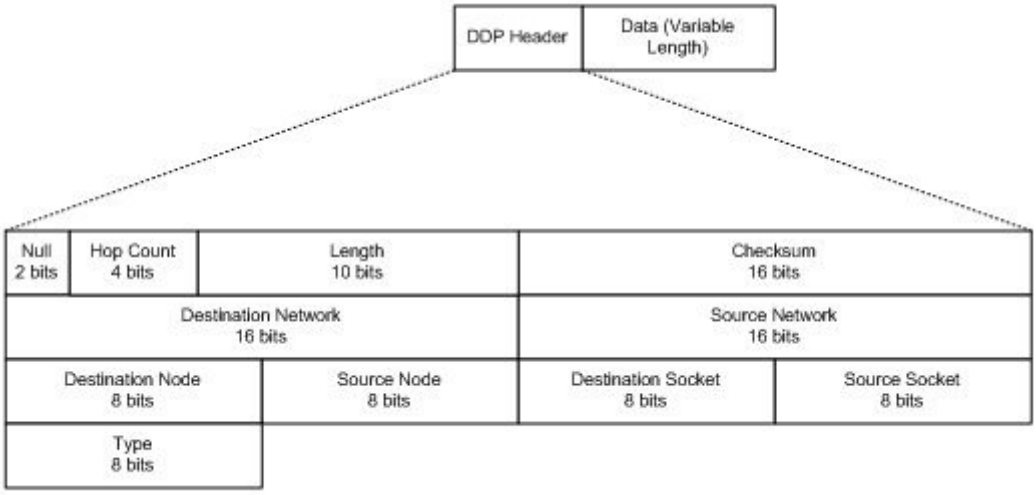


Figure 4-12: Extended DDP packet.

- **Null. The first two bits of a DDP packet are not used.**
- **Hop Count. Similar to a TTL value. For each router that a DDP packet crosses, this field is decreased by one. The maximum number of hops for a DDP packet is 15.**
- **Length. The total length of the DDP packet in bytes.**
- **Checksum. A computed value that verifies the integrity of the DDP header, similar to a CRC.**
- **Destination Network. The network number of the receiving system.**
- **Source Network. The network number of the sending system.**
- **Destination Node. The node number of the receiving system.**
- **Source Node. The node number of the sending system**
- **Destination Socket. The socket number of the receiving system**
- **Source Socket. The socket number of the sending system**
- **Type. This field specifies the upper-layer protocol to which the data in this packet should be passed.**

AppleTalk Address Resolution Protocol (AARP)

Much like the ARP protocol found in the TCP/IP suite, the job of the AppleTalk Address Resolution Protocol (AARP) is to map network addresses to physical (MAC) addresses. Similar to ARP, AARP also does this via broadcasts and temporarily caches entries that it has recently resolved. These are stored on each node in its Address Mapping Table (AMT).

Transport Layer

Five main protocols exist at the AppleTalk Transport layer. These include:

- **Routing Table Maintenance Protocol (RTMP)**. RTMP is used by AppleTalk routers to exchange, establish, and maintain routing table entries.
- **Name Binding Protocol (NBP)**. NBP is used to dynamically map AppleTalk resource names (such as shared folders or printers) to their network address. NBP allows resources to be accessed by name rather than network address.
- **AppleTalk Update-Based Routing Protocol (AURP)**. AURP allows AppleTalk networks to be connected over a WAN by tunneling AppleTalk through a TCP/IP network. This is accomplished by defining AURP tunnels between routers, which encapsulate AppleTalk traffic destined for a remote network in UDP headers. The UDP segments are then encapsulated for IP (and whatever network technology the WAN uses), and sent to the other end of the AURP tunnel where they are de-encapsulated and forwarded. Both point-to-point and multipoint AURP tunnels can be created. Note that the AppleTalk suite only defines Data Link interfaces for Ethernet, Token Ring, FDDI and TokenTalk. The Cisco implementation of AppleTalk also supports AppleTalk encapsulation over a variety of WAN technologies.
- **AppleTalk Transaction Protocol (ATP)**. ATP is the AppleTalk Transport protocol that handles transaction requests and responses between systems. An example of a transaction request would be a socket on a client system asking a socket on a server to perform an action, such as a time request. ATP on each system will not only be sure that for each request sent a response is received, but will also handle common Transport layer functions such as data segmentation, sequencing, and acknowledgements. ATP is mainly used for transferring small amounts of data, and can be used as the upper-layer protocol that brings reliability to DDP.
- **AppleTalk Echo Protocol (AEP)**. AEP performs a similar function to an ICMP echo request and reply. It is used to test for reachability and round-trip transmission times with another AppleTalk node. When used, the source node sends out an AEP request, and the recipient sends back an AEP reply.

Session Layer

Unlike TCP/IP and IPX/SPX, the AppleTalk suite has clearly defined Session Layer protocols. These include:

- **AppleTalk Data Stream Protocol (ADSP)**. ADSP establishes and maintains reliable full-duplex sessions between two AppleTalk sockets. It also handles flow control functions such as windowing. ADSP is symmetrical, meaning that both client sockets have equal access to the session.
- **AppleTalk Session Protocol (ASP)**. ASP is another Session Layer protocol that establishes, maintains, and closes connections between clients. ASP sessions are asymmetrical, with the client controlling session communication.
- **Printer Access Protocol (PAP)**. PAP is responsible for managing connection-oriented sessions to AppleTalk printers.

- **Zone Information Protocol (ZIP)**. The primary responsibility of ZIP is to maintain a network-wide mapping of networks (or cable ranges) to zone names. When a client starts and attempts to dynamically configure its address, it queries the local router using ZIP to find the valid network numbers on that segment.

Presentation and Application Layers

At the Presentation and Application Layers of the OSI model, one primary AppleTalk protocol exists, the AppleTalk Filing Protocol (AFP). The responsibility of AFP is to manage file system access between a client and AppleShare server on an AppleTalk network. AFP creates an abstraction whereby a client accesses network files as if they were stored locally. AFP uses ASP as its Session layer protocol.

Network Layer Protocol Scalability

Based on the varying amounts of unicast, multicast, and broadcast traffic generated by different Network Layer protocols, Cisco has developed a set of guidelines that outline the maximum number of hosts that should be configured as part of a single broadcast domain. Table 4-4 outlines the maximum number of hosts in broadcast domains running IP, IPX, AppleTalk, or NetBIOS only, as well as networks running a mix of these protocols.

Table 4-4: Maximum number of hosts per broadcast domain according to the Network Layer protocol is use.

Protocol	Maximum Hosts per Broadcast Domain
Internet Protocol (IP)	500
Internetwork Packet Exchange (IPX)	300
AppleTalk	200
NetBIOS	200
Mix of the above protocols	200

CCNA Study Guide Chapter 4 Summary

By Dan DiNicolo, June 2nd, 2006 Posted in **CCNA Study Guide Chapter 04.** Subscribe to our

RSS Feed

This chapter began with a look at the TCP/IP protocol suite. This included a look at the protocols that make up TCP/IP and the way in which they relate to the ISO OSI model.
A look at Network layer protocols provided an overview of IP packets and addressing, the way in which ARP is used to map IP addresses to MAC addresses, as well as how ICMP provides error reporting and diagnostic information.

Examination of protocols at the Transport layer provided an overview of both TCP and UDP, including their header structures. A closer look at TCP provided insight into port numbers, how connections are defined, how they are terminated. An introduction to UDP provided perspective on how it differs from TCP as a connectionless protocol with lower overhead.

Examples of common port numbers that act as interfaces to upper-layer protocols were also examined.

A look at the TCP/IP Application layer provided an overview of some common protocols and services found at this layer.

An examination of the NetWare protocol suite began with a look at the four Ethernet encapsulation types supported by Cisco and the terms used to refer to them on Cisco routers. A look at the Network layer provided an overview of IPX addressing and packet structure. At the upper layers, a variety of protocols were covered including the connection-oriented SPX. Other protocols discussed included NCP, NetWare Shell, NetBIOS and SAP, the Service Advertising Protocol.

Chapter 5: IP Addressing and Subnetting

IP Addressing and Subnetting

Chapter 5 of my free CCNA/CCDA Study Guide covers an essential subject area for CCNA and CCDA exam candidates: TCP/IP addressing and subnetting. Since I've already covered some of these topics in detail in previous tutorials, please use the links below to begin your journey into Chapter 5 of the book. Chapter 5 will continue exactly where the last article in this list (Introduction to Subnetting (Part 4) leaves off.

IP Addressing Basics (Part 1)

IP Addressing Basics (Part 2) - Classful IP Addressing

IP Addressing Basics (Part 3) - IP Addressing Rules

IP Addressing Basics (Part 4): Subnet Masks and Private IP Addressing

Introduction to Subnetting (Part 1)

Introduction to Subnetting (Part 2)

Introduction to Subnetting (Part 3)

Introduction to Subnetting (Part 4): Defining Valid IP Address Ranges

IP Addressing Basics (Part 1)

Whether you're preparing for the CCNA, CCDA, or even MCSE exams, you'll need to be familiar with IP addressing. Of course, just about everyone knows what an IP address looks like, but you need to go a step further. Specifically, you need to understand how IP addresses are constructed, what those numbers really mean, and how these addresses are used to facilitate communications on a TCP/IP network.

In previous articles we took a very basic and introductory look at the concept of an IP address. Recall that IP addresses are logical addresses made up of two parts – the first part represents a network, and the other, a unique host on that network. Before we get into the details of IP addressing, you'll first need to know a little more about how binary and decimal numbers relate to each other.

Remember that IP addresses are 32-bits in length, but are usually represented in dotted-decimal notation. As such, the address 192.168.2.200 can also be represented as:

11000000 10101000 00000010 11001000

But why should you care about the binary version? The answer is absolutely critical – in order to truly understand how IP addressing works, you must always take a look at elements such as addresses and subnet masks in binary form. After you've had lots of practice in binary, you'll get very good at "understanding" the numbers when you see them in decimal. Just remember that when starting off, it is important that you convert addresses to binary – doing so will ultimately unlock the secrets of subnetting, determining if hosts are on the same or different networks, and whether IP addresses are valid.

Having said that, our first logical step is learning how to do decimal-to-binary conversions and vice versa. Recall that a 32-bit address is actually grouped into four octets in its decimal form – each octet represents 8 bits in binary. The table below outlines the decimal and binary values for each of the 8 bits.

DECIMAL	128	64	32	16	8	4	2	1
BINARY	1	1	1	1	1	1	1	1

Notice that each of the eight binary digits has a single decimal value associated with it. An important pattern should be clear in the decimal row – moving from right to left, the value doubles in each successive column.

This is part of what makes binary numbering so useful – for any given decimal number, there is one (and only one), way to represent it in binary. After a few examples, understanding how binary numbering works will be clear.

We'll start with converting binary to decimal, since this is the easiest way to illustrate the conversion process. Imagine you wanted to convert the binary number 01010101 to decimal. Using the previous table, you'll notice that each binary "1" value corresponds to the decimal value above it. If we add together the decimal values that correspond to the 1s in the binary number, we'll have completed the conversion process. For example, the decimal value of the binary number 01010101 would be:

0 + 64 + 0 + 16 + 0 + 4 + 1 = 85

All I did was take the decimal value of each of the 1s in the binary number and replace them with their corresponding decimal value. In cases where the binary digit was 0, I did not add that value. Consider another example, the binary number 11100011. In this case, the corresponding decimal value would be calculated as:

128 + 64 + 32 + 0 + 0 + 0 + 2 + 1 = 227

I've left the 0 decimal values in my examples to act as placeholders – you should consider doing this until you feel comfortable that you remember the decimal values associated with each binary digit.

Converting decimal values to binary is a little different, but isn't terribly difficult either. To convert decimal to binary, we simply move from left to right, adding the decimal values together until we reach the number we're looking for. Consider the decimal number 68. If you wanted to calculate its binary version, you would begin adding across, beginning with the leftmost digit. In this case, the number would be:

0 + 64 + 0 + 0 + 0 + 4 + 0 + 0 = 68

In binary, this works out to 01000100. Notice that I replaced each non-zero decimal value above with a 1, and every unused value with a 0. Let's walk through the example step-by-step. We know that we're looking for the number 68. We must start at the left and add across to the right, or this will not work. Do we need the value 128 to get to 68? No, so we place a 0 as the first value. How about 64? Since 64 is less than 68, we do need that value, so I add it. Adding the next 3 values – 32, 16, and 8, would all bring us above 68, so we also replace them with 0s. The next value is 4. If we add 64 and 4, we reach my goal – 68. Therefore, we don't need to add the 2 or 1 values either – just replace them with 0s as well. When you look at the values we've added and compare them to our binary to decimal conversion table, it should be clear that 68 is 01000100 in binary.

>For the sake of clarity, let's try one more example. We'll convert the decimal number 177 to binary. Remember that you always begin adding from the leftmost bit to the rightmost. In this case, we end up with:

128 + 0 + 32 + 16 + 0 + 0 +0 + 1 = 177

Replacing the non-zero values above with 1s, 177 in decimal converts to 10110001 in binary. I would suggests trying a number of additional example on your own, just to be sure that you're clear on the concept – understanding these conversions will be critical once we get to subnetting.

You may also have figured out that these calculations can be easily accomplished using a scientific calculator, such as the one included with Windows. If you choose to use the calculator, switch it into scientific view; enter the value that you are looking for in decimal, and then click on the BIN radio button. One important note – if you use the calculator, understand that it will always leave the leading zeros off any binary conversion. Since we're looking for 8-bit values, you must add any dropped zeros to the converted value. For example, the calculator will display the binary version of 4 as 100. In order to use this value in our conversion process, you will need to add the 5 leading zeros. In this case, our answer becomes 00000100. The calculator will always remove digits that it does not consider "significant". Remember, it has no idea that we're looking for an 8-bit number.

Note: Remember that you will not be able to use the Windows calculator (or any calculator for that matter) during the CCNA and CCDA exams.

Coming Up

In the next article in the CCNA series we'll start to dig a little deeper with a look at Classful IP addressing – the place where you'll re-learn your A-B-Cs in a whole new way. Until then – stay tuned. If you're looking for more articles in the CCNA or CCDA series, click here.

IP Addressing Basics (Part 2) - Classful IP Addressing

In the last article in this series you learned about how all IP addresses are a product of binary numbering, and how to covert between decimal and binary values. If you missed that article, you can get to it by clicking here. This time around we're going to dig a little deeper into the A-B-Cs of the IP world with a look at classful addressing.

Since there are literally millions of IP addresses available, the IETF originally designated what are known as classes of IP addresses. The purpose of these classes was to break up the IP address space into ranges that accounted for networks of different sizes. The term "classful" is used to

describe addresses that are looked at according to their class. In reality, the world of IP addressing has changed such that classes of addresses are much less important than they used to be – later in the series, we'll take a look at classless addressing, including how and why it came about.

You'll definitely need to be familiar with classful addressing, since it forms the basis upon which IP addresses were originally defined, and is still a factor with routing protocols such as RIP version 1 and IGRP. Five different classes of addresses exist, and are distinguished according to the values found in their first octet. The Table below outlines each of the five ranges.

Class	First Octet Decimal Value	Network and Host Portions	Hosts Supported Per Network	Details
A	0-126	N.H.H.H	16,777,214	Intended for the largest networks only
B	128-191	N.N.H.H	65,534	Intended for medium sized organizations
C	192-223	N.N.N.H	254	Intended for small organizations
D	224-239	N/A	N/A	Reserved range used for multicasting
E	240+	N/A	N/A	Experimental range

The value of the first octet of an IP address holds the immediate answer to the class an address falls into. Notice that Class A addresses always begin with a value between 0 and 126. As such, the address 64.12.203.1 can safely be identified as Class A. From the table above, you should also note that in a Class A address, the first octet uniquely identifies the network (designated by the "N"), while the last three octets uniquely identify a host (designated by the "H") on that network. Only Class A, B, and C addresses are valid to assign to hosts. Class D addresses are used to support multicasting, while Class E addresses are reserved for experimental use.

You may have noticed that the first octet value of 127 is missing from the table above. What is the reason for this? The 127 range is actually reserved for diagnostic functions – for example, the address 127.0.0.1 is the loopback address. Ping that address, and you're actually testing the TCP/IP connectivity of the source machine.

But where did those 16,777,214 addresses come from? The answer to that lies in another small calculation. Since binary numbers can only have two values – 0 or 1 – the numbering system is referred to as Base 2. In order to calculate how many hosts a given network can support, you simply take the number of bits available in the host portion of an address, and calculate the exponential value using a base value of 2. In the case of a Class A address, there are 24 bits used to represent a host (the last 3 octets, 8 bits each). As such, a Class A network supports 2^{24} or 16,777,216 addresses. Notice that this number does not match the one in the table. This is because two addresses are never valid to be applied to a host – the host address of

all binary 0s, and the host address of all binary 1s. For this reason, you always subtract 2 from number of available host addresses. In this case, it brings us to a value of 16,777,214.

When the Internet Protocol was first defined, nobody ever thought that it would become as popular as it did – it was mainly designed for academic and military purposes, after all. The original idea behind classes of IP addresses was to broadly define groupings that would account for small, medium, and large organizations and their addressing needs. Notice the huge discrepancy between the ranges – while a Class B address supports 65,534 hosts, a Class C address supports only 254. So what happened when a company had only 2000 hosts? Well, in many cases they were allocated an entire Class B address range – not only is this wasteful, but it also ultimately resulted in changes being made to the way in which addresses are allocated. We'll look at how things have changed shortly.

But how many networks does each class support? For the Class A range, the answer is clear – 128, or the range from 0-127 (remember that the 127 address range is reserved for diagnostics). In the case of Class B and C networks, the answer is slightly more complex. The table below outlines how the number of available networks is defined. The leading bit value represents the binary digits that always start off the first octet of that class of address. For example, a Class A address always starts with its first digit set to binary 0 – this makes the maximum value that could appear in the first octet 127, or 01111111.

	Leading Bit Value	Remaining Network Bits	Number of Networks
Class A	0	7	$2^7 = 128$
Class B	10	14	$2^{14} = 16,384$
Class C	110	21	$2^{21} = 2,097,152$

Notice the Class B example – since a class B address always starts with the first two bits set to binary 10, the remaining bits in the network portion tell us how many networks are available. If the first two bits are reserved, that leaves 14 bits – remember that the first two octets (or 16 bits) represent the network portion of a Class B address. By calculating 2^{14}, we find that there are 16,384 Class B networks available. For a Class C address, the calculation would be 2^{21}, since the first three bits are fixed and the network is represented by the first 3 octets.

Because the first two octets of a Class B address represent a network uniquely, the IP addresses 131.107.2.200 and 131.108.2.200 are actually on different networks – the first is on network 131.107.0.0, and the second on network 131.108.0.0.

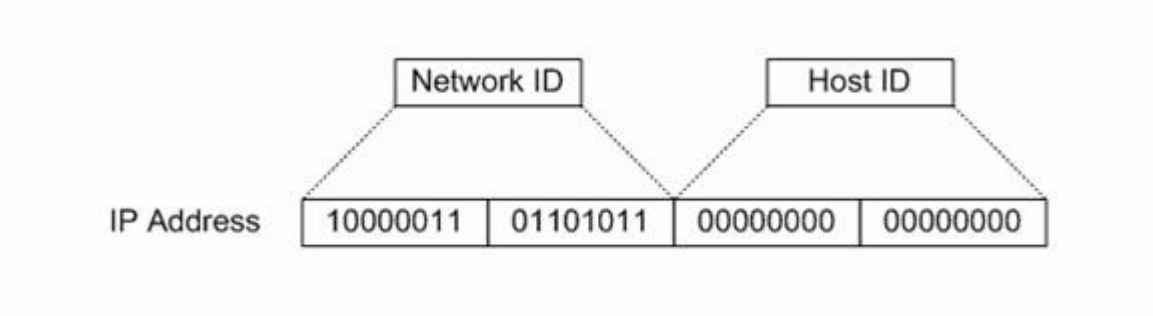
Coming Up Next

Now that you have a better idea of what classful IP addressing is all about, it's time to get into the meat and potatoes portion of this topic. Stay tuned for the next article in this series where we take a closer look at some of the rules governing IP addresses, including how to tell which addresses are valid, and which are not! Until then, all the best with your studies!

IP Addressing Basics (Part 3) - IP Addressing Rules

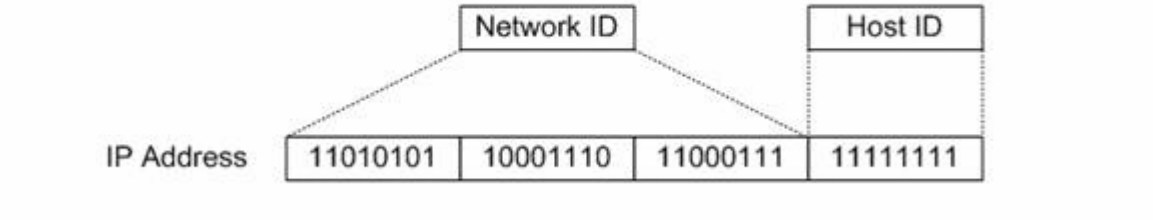
In the first article in this multi-part series you learned about how all IP addresses are a product of binary numbering, and how to covert between decimal and binary values. In the second, we dug into the concept of classful addressing. This article continues along the same path, looking at the rules that govern which IP addresses are valid, and which are not.

In order to be certain that IP addresses are valid, there are a number of rules that must be followed. The first rule of IP addressing is critical. The host portion of an address cannot be set to all binary 0s or all binary 1s. When the host portion of an address is set to all binary 0s, it is used as a way of referring to that particular network. For example, consider the Class B address shown in Figure 5-1, where the host portion is set to all binary 0s.



The IP address 131.107.0.0 is simply another way of saying “the 131.107 network”.

The requirement that the host portion of an address cannot be all binary 1s exists for a different reason. When the host ID is set to all 1s, it represents a broadcast on that particular network. For example, consider the Class C address in Figure 5-2, where the host portion is set to all binary 1s.



In this example, the address 213.142.199.255 represents the broadcast address for the 213.142.199.0 network. Any packets sent to this special address are destined for all hosts on the 213.142.199.0 network.

Recall how we subtracted 2 when attempting to figure out how many hosts a given network could support – this was to account for when the host portion is set to all binary 0s or 1s, as I just described.

The second rule that you need to remember is the use of all binary 0s or 1s in the network portion of an address. When the network portion is set to all 0s, it is interpreted to mean “this network”. For example, the address 0.0.12.145 would be interpreted as “host 12.145 on this network”. When the network portion is set to all ones, for example 255.255.1.2, this is the same as saying “host 1.2 on all networks”. For the most part, you will not be

manipulating the network portion of addresses in this manner – these designations will be used by the protocols, as per their programming.

The remaining rules are fairly simple. They include:

- The network ID of 127.0.0.0 is reserved for diagnostics and testing. The address 127.0.0.1 is referred to as the loopback address
- An IP address of all 0s (0.0.0.0) is used to represent the default route, or where all packets destined for unknown networks should be sent.
- An IP address of all 1s (255.255.255.255) is used to represent a broadcast to all hosts on a network.
- Network IDs of 224 and above in the first octet are not valid to assign to hosts, since Class D and E addresses are not valid for hosts.

The table summarizes the IP addressing rules that we've looked at in this article.

Rule	Purpose	Example
Host ID cannot be all binary 1s	This address represents a network broadcast	131.107.255.255
Host ID cannot be all binary 0s	This address identifies a network	131.107.0.0
Network ID cannot be all binary 0s	This address represents "on this network"	0.0.145.23
Network ID cannot be all binary 1s	The address represents " on all networks"	255.255.1.142
Network ID cannot be decimal 127	This address range is reserved for the loopback address	127.0.0.1
IP address cannot be all binary 0s	This address is used to represent the default route	0.0.0.0
IP address cannot be all binary 1s	This address is used to represent a broadcast	255.255.255.255
Network IDs of 224 and above in the first octet cannot be assigned to hosts	Class D addresses are reserved for multicasting, while Class E addresses represent an experimental range	224.0.0.1

As a quick test, see if you can answer the following question. Is the address 47.203.191.0 valid to assign to a host? You may not have thought so, but answer is yes. Why? The fact that the address ends in a decimal value of 0 doesn't make a difference. Since this is a Class A address, if you convert it to binary, you'll notice that the host portion (the last three octets) is neither all binary 0s nor all binary 1s. This is just another example of why it's so important to always consider addresses in binary. Try not to let assumptions based on what you see in decimal throw you off.

Up Next

With an understanding of the basic rules of IP addressing now on your side, it's time to move on to an introductory look at subnet masks. If you've ever been curious as to what subnet masks are or why they're necessary, the mystery will soon be unraveled. Until then, best of luck with your studies.

IP Addressing Basics (Part 4): Subnet Masks and Private IP Addressing

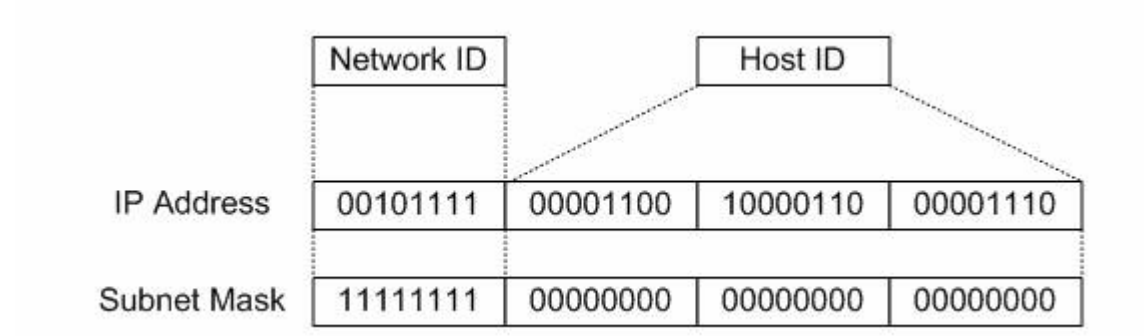
As I mentioned in previous articles in this series, subnet masks hold the key to understanding the breakdown between the network and host portions of an IP address. The reasons why the classes of IP addresses are split in the way they are is because of the subnet mask assigned to each class by default. The table below outlines the default subnet mask assigned to each class of IP address.

Address Class	Default Subnet Mask
Class A	255.0.0.0
Class B	255.255.0.0
Class C	255.255.255.0

Right off the bat, an association should become clear. Notice that for a Class A address, the default subnet mask occupies the entire first octet, with a value of 255. If you convert this subnet mask to binary, it becomes:

11111111 00000000 00000000 00000000

When the subnet mask in converted, the entire first octet is made up of binary 1s. The division between the network and host portion of a Class A address occurs where the 1s stop – in this case, right between the first and second octet. As such, we can say that when the default subnet mask is used, the address 47.12.134.14 is separated as shown below.

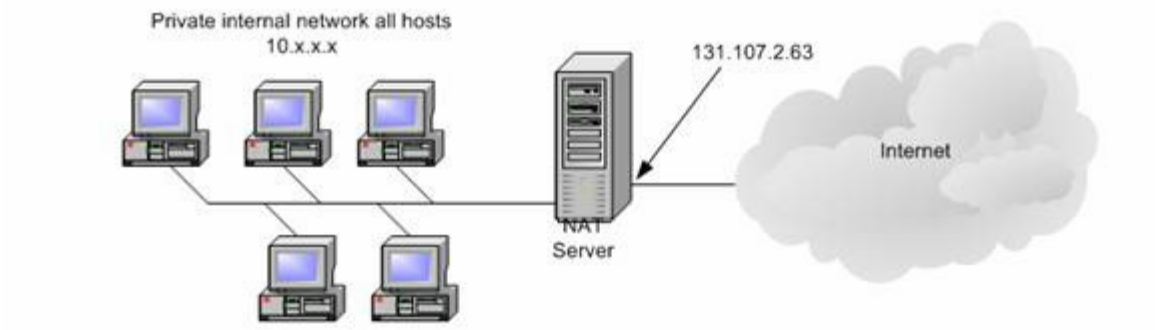


Notice the relationship between the subnet mask and the address. Think of a subnet mask as a cookie cutter that separates the network and host portions of an address. Without a subnet mask, you cannot make an educated statement about how the network and host portions of an address are defined. That may not be fully evident quite yet, given that we've only looked at classful addressing and default subnet masks. Later in this series, when we look at how custom subnet masks are defined, this distinction will become even more important.

Private IP Addresses

Based on the incredible growth of the Internet, it soon became evident that the IP address space would quickly become exhausted if the growth continued. To account for this, the IETF looked for ways in which the address space currently available could be extended. A future solution exists in the form of IP version 6 (or IPv6 for short), which uses a much larger 128-bit addressing scheme. In order to deal with the issue in the shorter term, it was decided that certain address ranges would be deemed "private". Private IP address ranges are defined in RFC 1918.

The idea behind private ranges of IP addresses is surprisingly simple – certain IP address ranges would be dedicated and limited to use for hosts on private networks. These addresses would no longer be considered valid (or be routable) on the public Internet. Instead, companies could allocate addresses in these ranges as they saw fit, with the address ranges open and available to everyone. Private IP addresses are a practical solution, since companies can use technologies such as Network Address Translation (NAT) or Proxy servers to connect their private networks to the public Internet. When these technologies are used, a single public IP address can be used to connect an entire organization to the Internet. NAT will be looked at in detail later in my CCDA series if you're curious about how it works. For the time being, the figure below shows a network that connects to the Internet using only a single public IP address.



To the outside world (the public Internet), all requests from within the company above appear to be coming from the single public IP address.

Three ranges of private IP addresses are defined in RFC 1918. The ranges defined have been allocated custom subnet masks that define their network and host portions. The table outlines the private IP address ranges defined in RFC 1918.

Network Address	Subnet Mask	Range of Addresses
10.0.0.0	255.0.0.0	10.0.0.1 – 10.255.255.254
172.16.0.0	255.240.0.0	172.16.0.1 – 172.31.255.254
192.168.0.0	255.255.0.0	192.168.0.1 – 192.168.255.254

Notice the second private network address, 172.16.0.0 with a subnet mask of 255.240.0.0. It's easy to immediately react and consider this to be a Class B address, based on the fact that the first octet value falls into the 128-191 range that we learned earlier. Instead, the private portion of this address range only goes as high as 172.31.255.254. Addresses beginning with network 172.32.0.0 are actually valid, public IP addresses. The 192.168.0.0 network uses a special subnet mask as well. In this case, its mask appears to be that of a Class B address. For the time

being noting the differences is enough – we'll explore how custom masks are defined beginning in the next article in this series.

But why would a company want to use private IP addresses? A big reason is because they offer a great degree of flexibility. A company can now pick one of these network IDs, and address internal hosts as they see fit. In the past, public IP addresses were allocated to companies, and later needed to be "rented" from ISPs. When private IP addressing is used, a company's need for public addresses is dramatically reduced, sometimes to only one or just a few IP addresses.

So which private range should a company use? Well, that's entirely up to them. Obviously the 10.0.0.0 network ID offers the greatest flexibility, based on the number of possible host addresses it provides. For small networks, the 192.168.0.0 range is probably most appropriate. At the end of the day, however, it's completely in the hands of the network administrator.

Tip: For a more detailed look at the private IP addressing ranges, see RFC 1918.

Coming up

Now that you understand the purpose of default subnet masks and private IP address ranges, it's time dig deeper into a topic that strikes fear in the heart of many a network administrator – defining custom subnet masks. To be honest, the whole deal is much easier than many people make it out to be. First we'll look at things the long way, ensuring that you actually understand why masks are defined the way they are (the real understanding part). Then, once you have the whole deal down pat, I teach you a couple of shortcuts that make the whole process quick and painless. Until next time, keep on learning!

Introduction to Subnetting (Part 1)

We've already looked at the basic idea of a subnet mask – we know that it is used to define the separation between the network and host portion of an IP address. The next step is looking at how we can define a custom mask to meet the requirements of a particular network. Before we go there, we need to define what subnetting is really all about. By the end of this series of articles, you'll understand how subnetting can be made easy - it's not as tough as you think, and is definitely well worth knowing.

If you're already familiar with Windows, you have probably experienced how some versions will automatically populate the subnet mask field after you've entered an IP address. By default, the subnet mask provided is usually the default mask for the class of address that you've entered. The default subnet mask is used in cases where you are not subnetting a network – meaning the entire address range is considered to be part of one big, single network.

When you custom subnet a network, what you are actually doing is logically breaking up the IP address space into a number of smaller (or sub) networks. Reasons for doing this vary, but generally fall into a three main categories. These include:

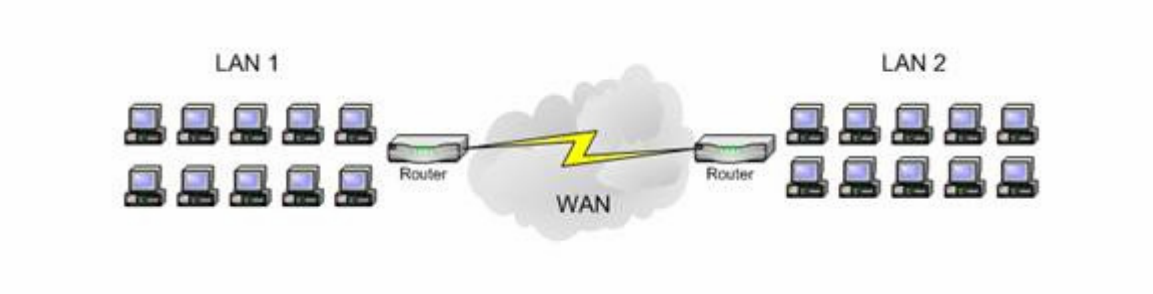
- **Facilitating better performance**. By breaking one big network up into smaller networks, traffic on each network is reduced, resulting in better performance.
- **Simplifying management**. In cases where a network is broken down into smaller networks, management complexity is reduced, making troubleshooting and monitoring easier.
- **Spanning geographical distances**. Because many network environments today consist of geographically dispersed but interconnected networks, subnetting allows remote

locations (and the WAN links between networks) to be considered as individual networks, resulting in better performance.

Defining a custom subnet mask isn't terribly difficult, but there are a number of factors that you will need to consider prior to getting started. Most of these factors center on properly characterizing the network, as well as accounting for future growth. In order to accomplish this, you need to be aware of the following:

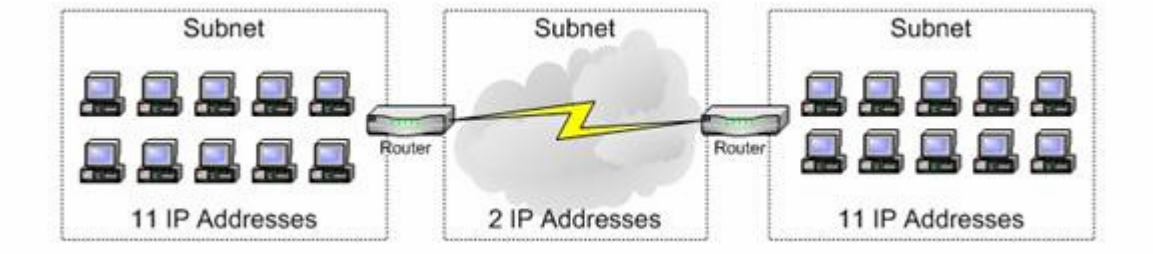
- Each subnet that you define will require a unique subnet ID, as will each WAN link.
- Each host and each router interface will require a unique IP address

Consider the simple network in the diagram below. Notice that it consists of two small LANs with a dedicated WAN link between the locations.



Given the scenario above, how do we calculate how many subnets we require? You should immediately assume one subnet for each LAN and one for the WAN link, or 3 total. Recall, however, that you also want to account for future growth. What if the company purchases another company, or decides to open additional offices? It's generally a good idea to account for more subnets than your immediate needs dictate.

Next, imagine that each office has 10 computers and a router for the WAN connection. In each office, it makes a basic requirement of eleven IP addresses, one for each computer, and one for the local router interfaces. Obviously this number doesn't account for growth, which should also be factored in. Don't forget the WAN link either – although it only connects two devices, it still requires its own subnet and two IP addresses. The figure below outlines the subnet and IP address requirements.



After we've gathered the data on the number of subnets and IP addresses required on each, we're ready to begin the real work – defining a custom subnet mask. That's exactly where we're going in the next article in this series, so stay tuned!

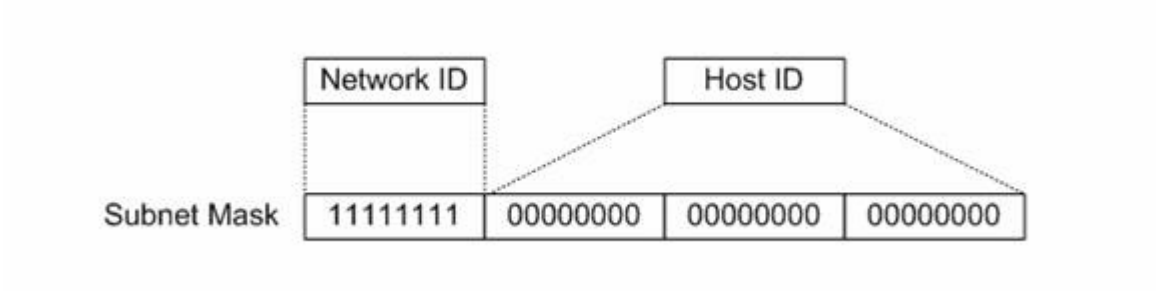
Introduction to Subnetting (Part 2)

In this article we're going to begin to look at the process for custom subnetting Class A, B, and C network addresses. The good news is that the process is all just mechanics – if you follow the steps, you will always be able to define a custom subnet mask that meets your requirements.

When defining a custom mask, we always start with the default mask for the class of address that we're working with. For a Class A address, the default mask is 255.0.0.0. Recall that when the default subnet mask is used, it means that no subnetting is currently taking place.

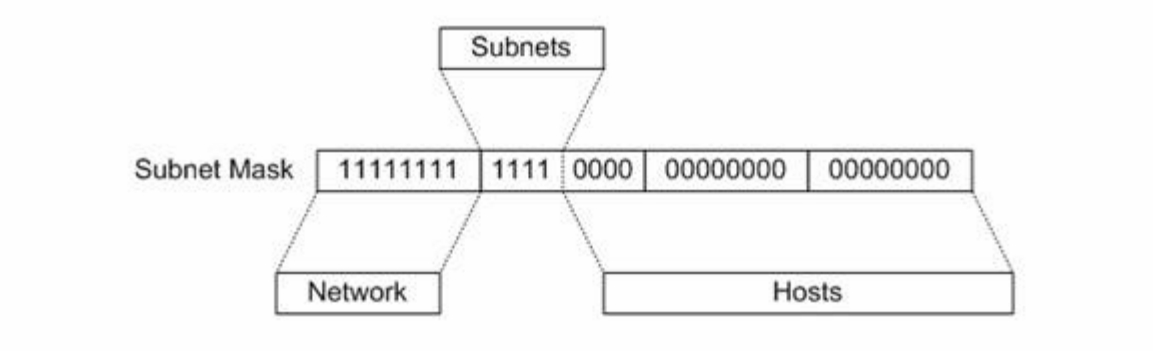
In order to define sub-networks, we'll need to play around with the value of the subnet mask. The extent to which the mask is changed will impact both how many subnets we can have, and how many host addresses will be available on each subnet. Creating a custom mask is accomplished by what I refer to as "stealing bits". We're literally going to create another division within the address space in order to identify our subnets.

Consider the basic example below. In it, the subnet mask clearly defines the boundary between the network and host portion of the IP address shown.



In order to define subnets, we'll need to extend the subnet mask. Notice that currently, the boundary between the network and the host portion of an address is defined by where the binary 1s stop – again, you really need to look at this in binary, or it really won't make sense. To create our custom mask, we're actually going to start replacing some of the subnet mask 0s with 1s, beginning on the left and moving to the right. In this way, we are "stealing" bits from the host portion of the address space, in what is referred to as "high order".

Take a look at another example below. Although I haven't associated it with any requirements yet, notice that I have taken the default subnet mask for a Class A address, and have changed it to a custom non-default value. The address now includes not only network and host portions, but also a section that will be used to identify subnets.



You have probably noticed that the divisions between the different portions of the address are no longer so clearly referenced. The subnet ID is defined by only 4 bits, meaning that it creates a division right in the middle of the second octet! This is an example of why looking at things in binary is so important – in decimal, it is very difficult to get a feeling for where (and how) the division occurs.

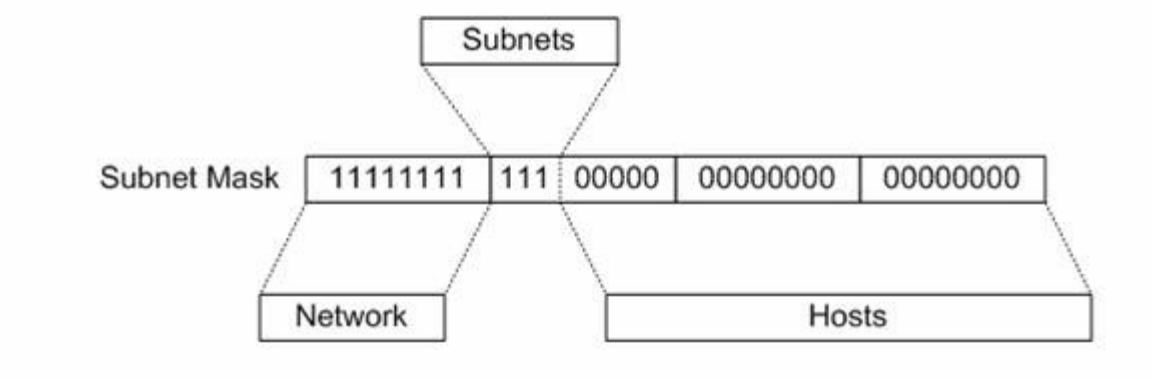
It's time to get some custom masks defined. Let's start off by going back to our original example. In it, we required at least 3 subnets, with an absolute minimum of 11 hosts per subnet. In order to come up with our custom mask value, we first need to decide which network address to use. I'm going to assume the private network address 10.0.0.0, which has a default subnet mask of 255.0.0.0.

In this example, we know that the first 8 bits (the complete first octet) define the network. By stealing bits from the host portion of the address (starting at the beginning of the second octet), our custom mask will begin to take shape. But how many bits should we use? Well, for that we have to go back to our requirements, and a tiny bit of math.

We know that we require at least 3 subnets at a minimum. In order to understand how many bits we need to use to define the subnet portion of the address, we need to do a simple calculation. Remember that binary is Base 2. If we use a very simple exponential calculation, we can figure out how many subnets a certain number of bits will provide. In the same way that we subtracted "2" from the number of available hosts on a network, we are also going to subtract 2 from the number of available subnets. Although this is not explicitly required by RFCs, it is recommended by Cisco, so this is the model that we will follow. This will ultimately remove the first and last subnets from our available subnets. In other words, we are removing the subnet defined by all binary 0s, and the subnet defined by all binary 1s. The reason for this is that some older routers will not recognize that these are valid subnets.

To find the number of available subnets, calculate the value $2^n - 2$, where n is the number of bits being "stolen". For example, using 1 bit is not enough, since 2^1-2 is 0. Using 2 bits would provide 2^2-2 subnets, or 2. Using 3 bits would provide 2^3-2 subnets, or 6. See a pattern developing?

From that quick look, it should be clear that using 3 bits will meet our immediate needs – after all, this allows for 6 subnets, and we only need 3. I'm not saying this is the best answer to account for growth, but it will work for the time being. Using 3 bits to define our subnet mask gives us the custom subnet mask illustrated below.



Notice that our custom subnet mask becomes 255.224.0.0. I simply converted the second octet from binary to decimal to get that value. But how do we know whether this custom mask supports enough hosts per subnet? Just look at how many host bits remain, according to the mask. In this case, the first 8 bits represent the network. The next 3 bits are used to define subnets. That leaves 21 bits for hosts – recall that the mask contains 32 bits in all. With 21 host bits, we can have 2^{21} - 2 hosts per subnet. Remember that the 2 bits are always subtracted when we calculate the number of available host addresses. Altogether, our 21 host bits provide an incredible 2,097,150 host addresses per subnet! If that seems like a bit much, not to worry. We can always "play" with the mask value to give us a subnet / host per subnet breakdown that better meets our requirements.

Coming up

In the next article in this series we'll continue on with a few more step-by-step examples of defining custom masks, including a look at subnetting class B and C addresses. Stay tuned!

Introduction to Subnetting (Part 3)

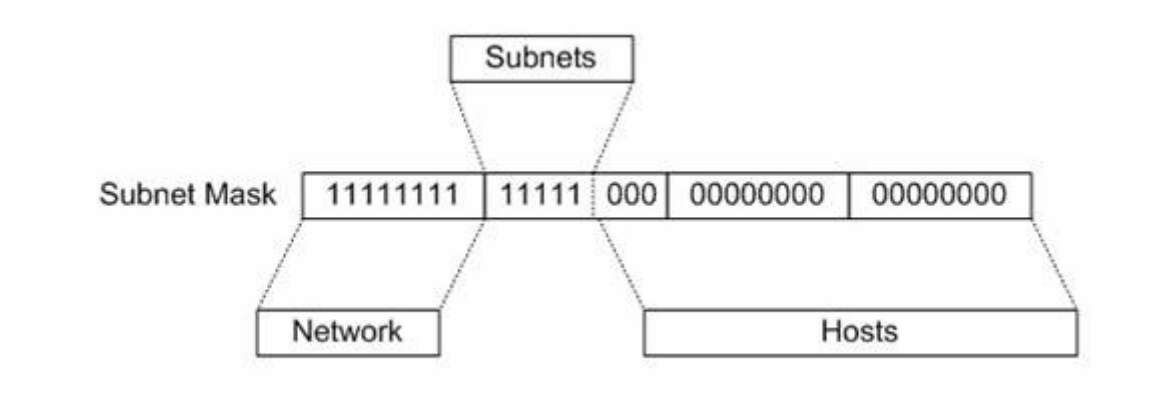
In the last article in my Cisco CCNA series we walked through an example of defining a custom subnet mask for a subnetted Class A network. In this article, we carry on with a look defining masks for Class B and C addresses. To get to part 1 of this series, click here. For part 2, click here instead.

The table below outlines the custom values that are possible for an octet when it comes to subnet masks. Notice that all of the binary values simply add 1s in high order – the binary 1s must be contiguous in this way.

Bits Used	Binary Value	Decimal Value
1	10000000	128
2	11000000	192
3	11100000	224
4	11110000	240
5	11111000	248
6	11111100	252

7	11111110	254
8	11111111	255

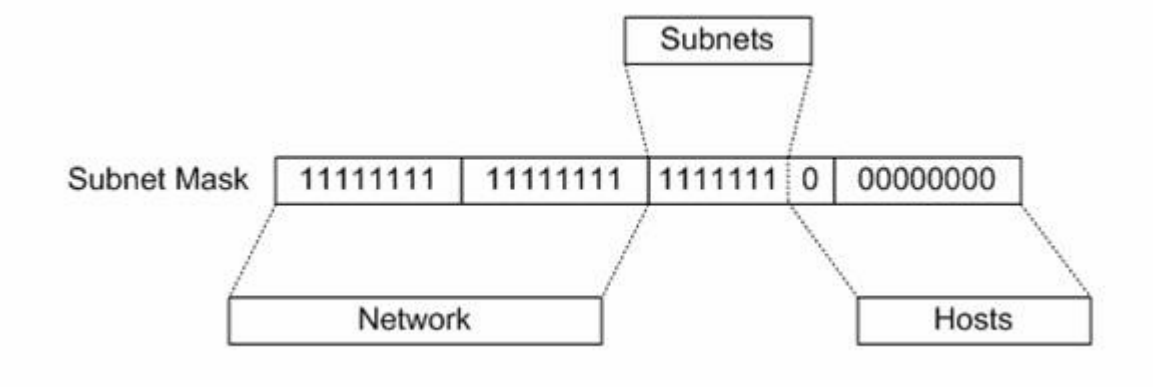
Since there are only eight custom values that can go into a mask, we are limited in terms of the number of subnets we can define. For example, consider a case where we need 17 subnets. How many bits would you use? Well, 4 would not be enough, since this provides only 2^4-2, or 14 subnets. The next choice is using 5 bits. Notice that this gives us 2^5-2, or 30 subnets. This is more than we need, but there isn't anything else between these two values. Not to worry, though. Using 5 bits, we not only meet our immediate need, but also account for 13 additional subnets worth of growth! Again assuming a Class A address, this gives us the breakdown illustrated below.



Using 5 bits gives us a custom mask of 255.248.0.0 after we convert the mask back to dotted-decimal notation. To calculate the number of hosts per subnet, just take the remaining bits (8 are used to define the network, and 5 the subnets, which leaves 19 host bits), and do the calculation. 2^{19}-2 equals 524,286 hosts per subnet.

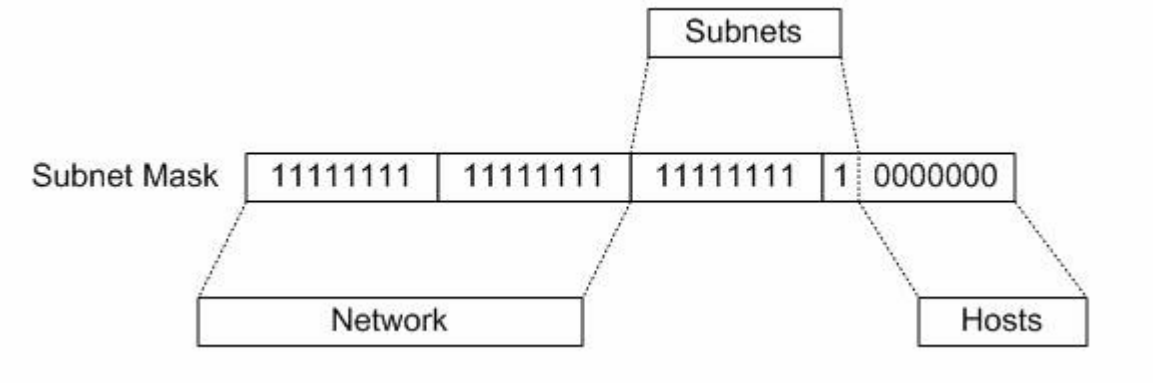
Now let's try creating a custom subnet mask for a Class B address. Remember that the default mask will be 255.255.0.0. So if subnetting a Class B address, we're going to begin defining our custom mask in the third octet. Say that we require at least 72 subnets to meet our current requirements. Let's walk through the steps below:

1. Since we know that we need at least 72 subnets, our first step is figuring out how many bits we'll need to steal from the host portion. We've already looked at an example with 5 bits, so we know that won't be enough – 2^5-2 provides only 30 subnets. How about 6 bits? That only provides 2^6-2, or 62 – again not enough. 7 bits will provide 2^7-2, or 126 subnets – this meets our requirement of 72, and also accounts for future growth.
2. Given that we need 7 bits, the next step is to define the custom subnet mask in binary. In this case, the first 16 bits will define our Class B network, the next 7 bits will define our subnets, and the remaining bits will define hosts, as shown below.



3. Still looking at the figure above, calculate how many host bits remain. In this case, there are 9 host bits available. This gives us a maximum of 2^9-2, or 510 hosts per subnet.
4. Convert the subnet mask to decimal and we're done – our custom mask is 255.255.254.0

But what if you need more subnets than 8 bits can provide? No problem – just steal more bits! For example, imagine that in the example above we actually required 300 subnets. Obviously 8 bits won't meet our needs, since 2^8-2 is only 254. But what about 9 bits? This will actually meet our needs – 9 bits provides 2^9-2 or 510 subnets. Don't get worried about spanning octets when you're subnetting. You just keep on stealing bits in high order. This is illustrated below.



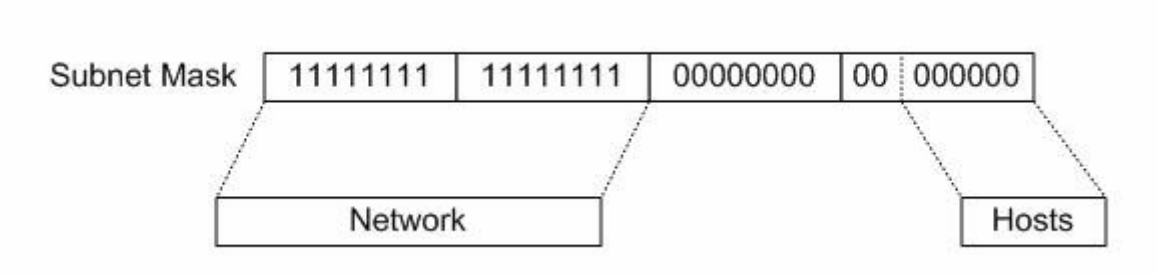
Notice that when converted back to decimal, our custom subnet mask becomes 255.255.255.128. This mask is absolutely valid, and leaves us a total of 7 host bits. This provides for a maximum of 2^7-2 hosts per subnet – 126. In other words, our mask provides 510 subnets total, with each supporting up to 126 hosts.

Remember that when defining a custom mask, you not only need to consider the number of subnets required, but also the number of hosts each subnet will need to support. Imagine if I came to you with a special requirement. I have a Class B address, want the maximum number of subnets possible, and each subnet must support 60 hosts maximum. This is not difficult to figure out, but the way in which we approach the problem is different. Since the requirement specifies that each subnet will only ever need to support a maximum of 60 hosts, that's where we'll start. Let's walk through this one step-by-step.

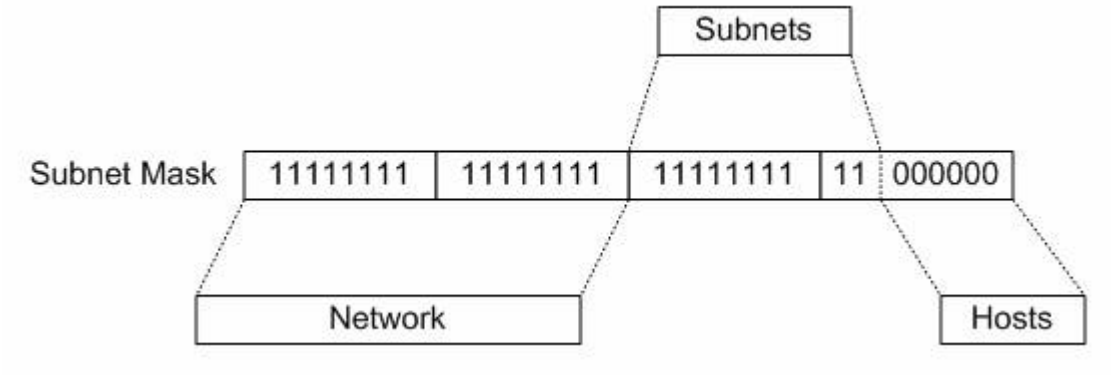
1. We know we need to support 60 hosts per subnet, so we should start by defining how many host bits we require. In this case, 6 bits will be enough, since 2^6-2 equals 62. Don't

forget to subtract the 2 whenever you're dealing with host bits – otherwise, your answer will be wrong.

2. Knowing that we need 6 bits for hosts, and that we're using a Class B address, simply draw out the diagram shown below. Notice that I've sectioned off the last 6 bits, since they're reserved for hosts.



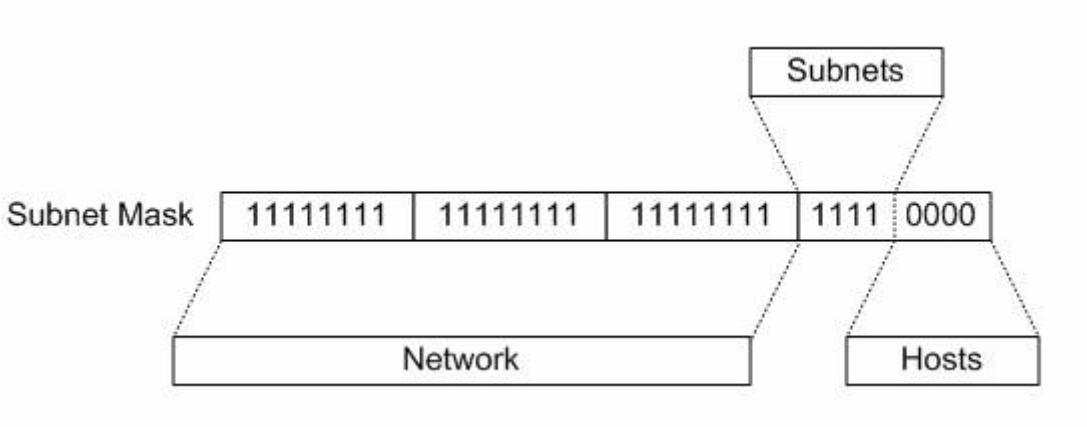
3. We want the maximum number of subnets possible, so we simply use all remaining non-network and non-host bits to define our subnets. Recall that this means changing the subnet bits to binary 1s. This is illustrated below.



4. If we convert our mask back to decimal, we come up with 255.255.255.192. This mask will provide the maximum number of subnets possible, given our requirement of supporting 60 hosts per subnet. In this case, we can have up to 2^{10}-2 (1022) subnets, and 2^{6}-2 (62) hosts per subnet.

Just for the sake of completeness, let's take a look at defining a custom subnet mask for a Class C address as well. Since Class C addresses support a fairly limited number of hosts to begin with, subnetting these addresses isn't terribly common. However, it is something that you need to know. Recall that the default mask for a Class C address is 255.255.255.0. In this example, let's assume that we want to have at least 12 subnets, with the maximum number of hosts per subnet possible.

The steps are the same as always. Since we've been given a requirement for a certain number of subnets, we need to figure out how many subnet bits we'll require. In this case we need at least 12 subnets, which means we'll need to "steal" at least 4 bits, since 3 will not be enough. 2^{4}-2 gives us a maximum of 14 subnets. This is illustrated in below.



Notice that because we're dealing with a Class C address, we're left with only 4 host bits. That means that our custom mask of 255.255.255.240 will provide us with a fairly limited 2^4-2, or 14 hosts per subnet. While this may not seem very practical, we have met the requirement of supporting at least 12 subnets given a Class C address range.

Up Next

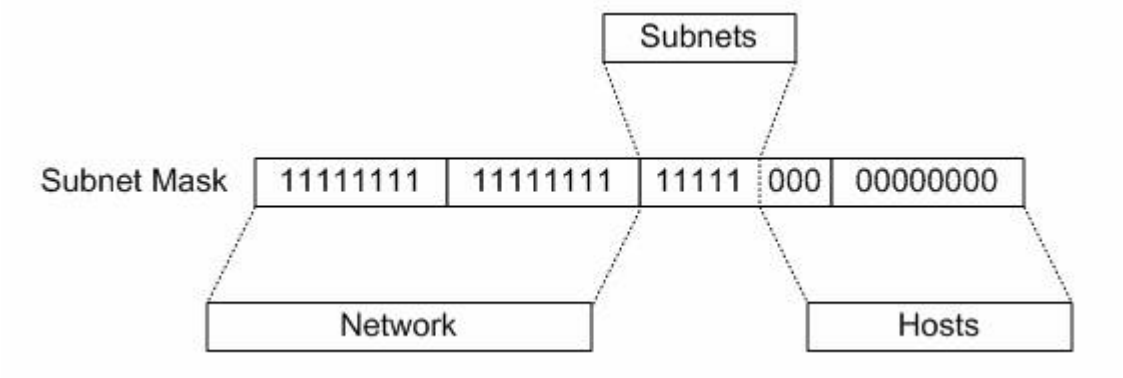
Now that you're familiar with defining custom subnet mask values, it's time to move on to the business of figuring out which addresses are valid on a given subnet, and why. It's all about defining IP address ranges, and it's coming up next. Until then, all the best.

Introduction to Subnetting (Part 4): Defining Valid IP Address Ranges

Once you've managed to define a custom subnet mask that meets your requirements, the next step is calculating the ranges of IP addresses that are valid for a given subnet. Remember that after a custom subnet mask is defined, you have actually turned one big network address space into a number of smaller sub-network address spaces. Because of this, certain addresses will no longer be considered local to one another.

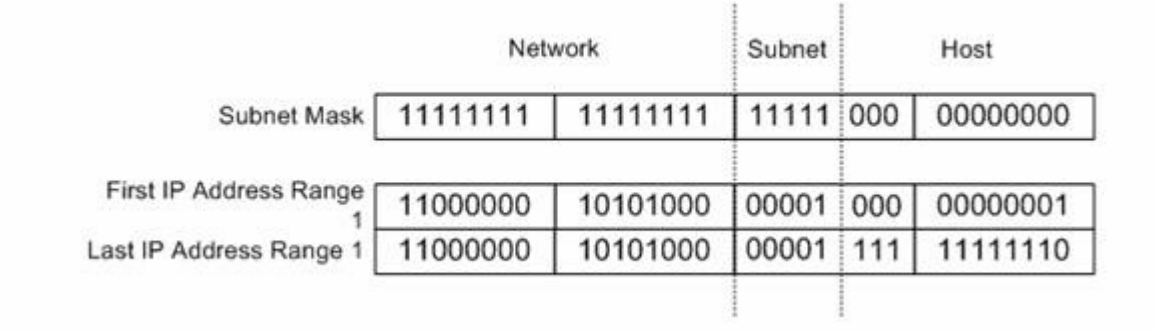
Calculating ranges of addresses isn't difficult, but we'll again need to go back to binary to understand how this is accomplished.

Consider the example of network 192.168.0.0 with a custom subnet mask of 255.255.248.0. Remember that this is one of the private address ranges outlined in RFC 1918, and the default mask is 255.255.0.0 in this case. To begin, we'll need to convert the subnet mask to binary, and define the network, subnet, and host portions, as shown below.



Notice that the first 16 bits represent the network, the next 5 define subnets, and the last 11 are used to uniquely identify hosts on a subnet.

Since 5 bits are used to define the subnet portion of the address, this gives us 2^5-2 or 30 possible subnets. Each subnet can support 2^{11}-2, or 1022 unique hosts. Calculating the range of available addresses for each subnet involves beginning with the first subnet that is not all binary 0s, and then looking at the additional subnets. In this case, the first subnet is defined as the one where all but the last bit is set to binary 0. The first host on the 00001 subnet will be the lowest host value that is not all 0s, since the all 0s address identifies the subnet. The highest address in a range will be the highest host value that is not all 1s, since that value is reserved for broadcasts. I know this can be a little confusing, so take a look at the example below.



If you take the addresses in Figure 5-17 and convert them back to decimal, you'll see that the range of IP addresses available on the first subnet are those from 192.168.8.1 up to 192.168.15.254.

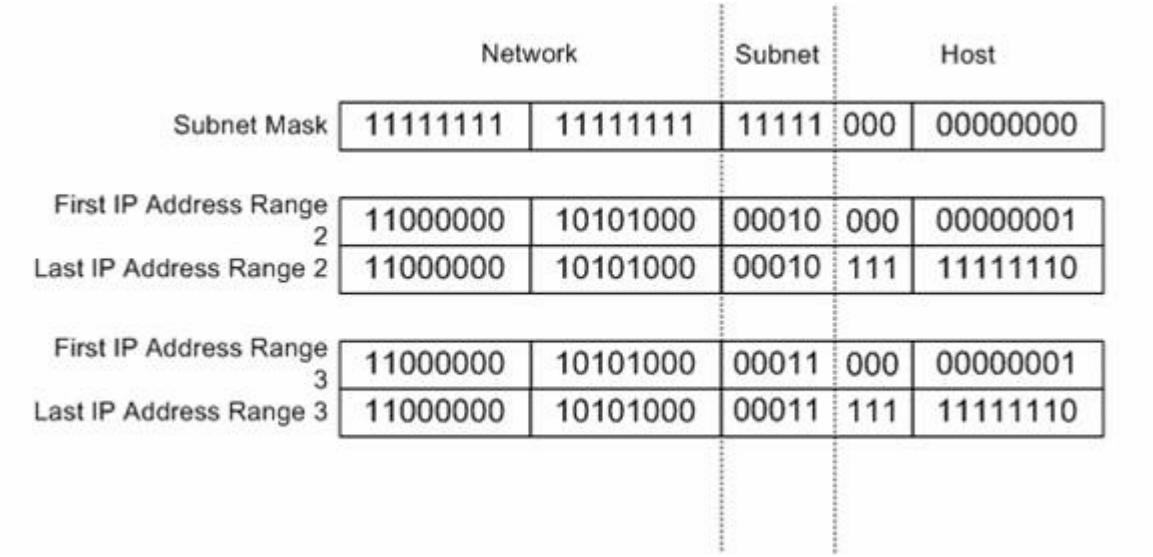
Calculating the next range of addresses requires that we change the subnet ID to its next lowest value. In this case, it would be 00010. To make things easier, consider this to be an isolated 5-bit binary number. The first subnet is 1; the next is 2, then 3, then 4, and so forth. In this way, the first five subnet IDs in our example would be those listed in the table below.

Subnet ID	Decimal Value
00001	1
00010	2
00011	3
00100	4

00101	5

Notice that each subnet simply increases by a value of 1 when we consider it as an isolated 5-bit number. This would continue all the way up to 11110, at which point we run out of subnets. Remember that according to Cisco, the subnet IDs of all binary 0s and all binary 1s should be avoided.

Our example continues by calculating the ranges for the second and third subnets. Notice in the figure below that the subnet ID has changed for each, but that we still always include the lowest and highest possible host values – the beginning and end of each range.



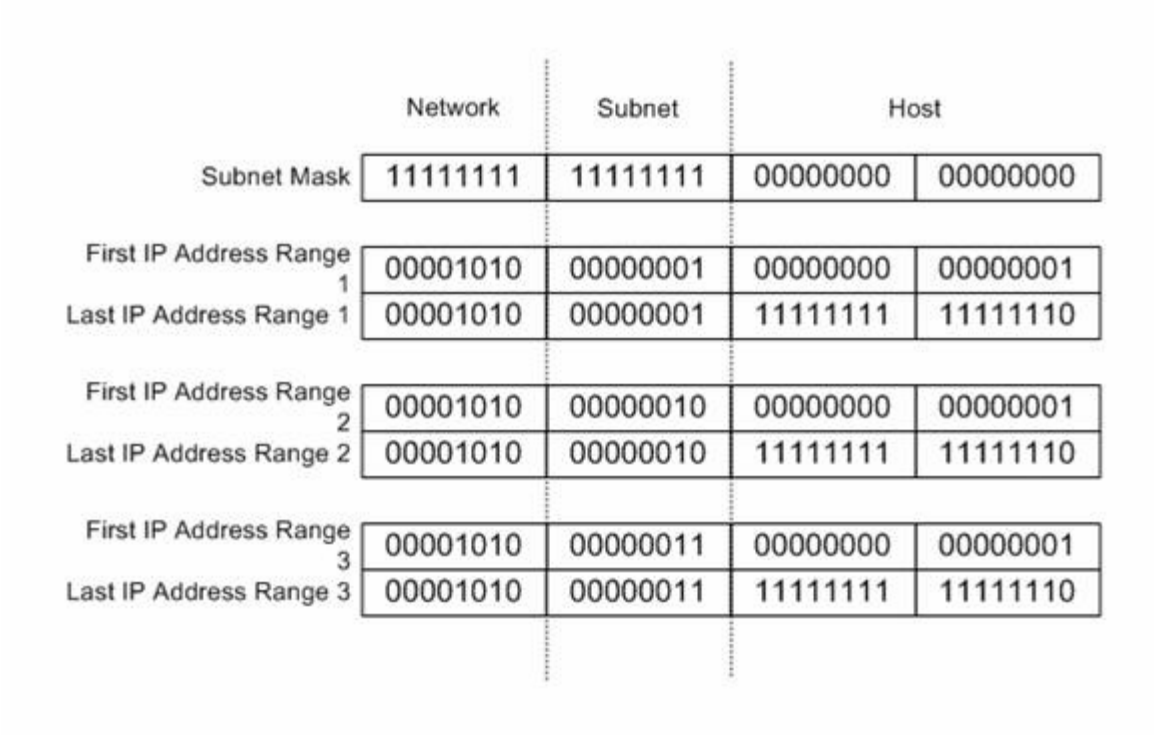
The table below outlines the first three address ranges, along with the subnet ID (when the host bits are all set to binary 0) and the broadcast address for that subnet (where the host ID is set to all binary 1).

Subnet ID	Address Range	Broadcast Address
192.168.8.0	192.168.8.1 – 192.168.15.254	192.168.15.255
192.168.16.0	192.168.16.1 – 192.168.23.254	192.168.23.255
192.168.24.0	192.168.24.1 – 192.168.31.254	192.168.31.255

If you look closely, you should notice a pattern developing – each new subnet (in this example) starts at a multiple of 8 in the third octet, and each ends just before the next multiple of 8. In this case, the fourth subnet would begin at the next multiple of 8, 192.168.32.1 and go up to just before the next multiple of 8 in the third octet. Since the next multiple of 8 is 40 (and represents the beginning of the next subnet), it must end at 39, or 192.168.39.254.

The pattern with multiples of 8 only occurs with a 248 custom mask value. With different masks, different multiples will become clear.

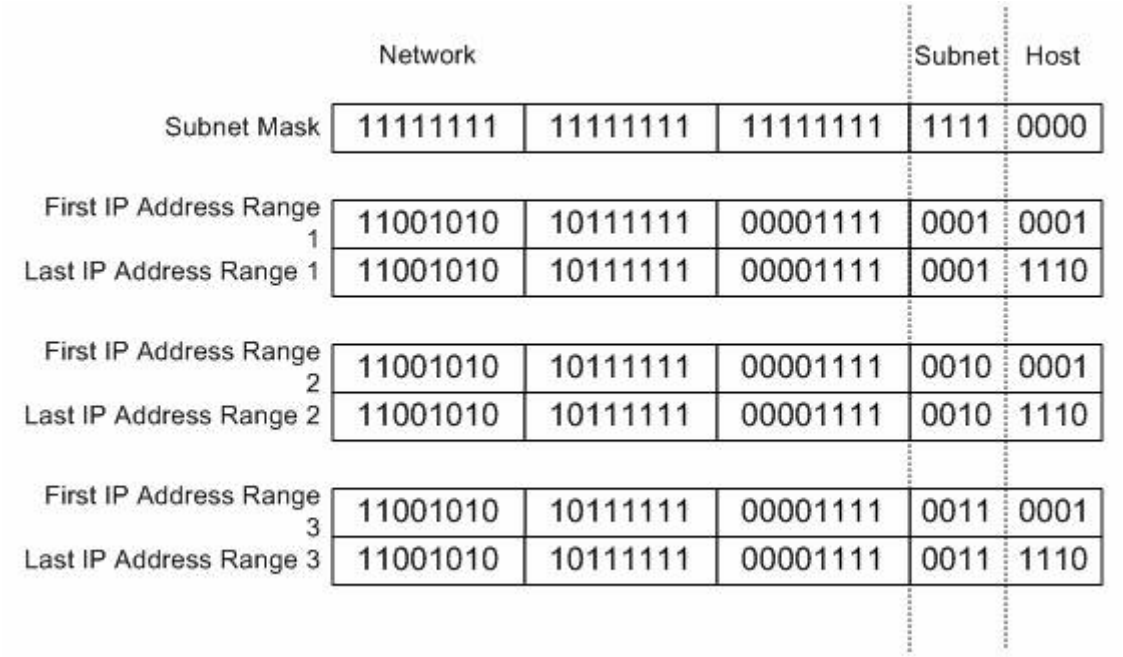
Let's try another example. We'll use the address 10.0.0.0 with a subnet mask of 255.255.0.0. With this example, there are 8 bits used for subnetting, leaving 16 bits for hosts. The figure below outlines the first three ranges of addresses in this example, beginning with the first subnet – 00000001 in the second octet.



In this case, the ranges move in multiples of 1 in the second octet – the first subnet is 1, the second is 2, the third is 3, and so forth. It would be safe to say that the fourth subnet will be the range between 10.4.0.1 – 10.4.255.254. Once you've found the multiple, determining the ranges is much easier. The first three subnet IDs, address ranges, and the broadcast address on each subnet are shown in the table below.

Subnet ID	Address Range	Broadcast Address
10.1.0.0	10.1.0.1 – 10.1.255.254	10.1.255.255
10.2.0.0	10.2.0.1 – 10.2.255.254	10.2.255.255
10.3.0.0	10.3.0.1 – 10.3.255.254	10.3.255.255

Some people find Class C address ranges a little trickier to deal with, but if you follow the rules, they are no more difficult that any other range. For example, let's say that we have the network address 202.191.15.0 with a subnet mask of 255.255.255.240. We know that subnetting occurs in the fourth octet in this example, and that we have 4 subnet bits and 4 host bits. The first subnet will again be lowest non-zero value, or 0001. The first host in each range will be the first non-zero value, and the last host the highest value that is not all binary 1s. The image below outlines the first 3 ranges in this case.



Notice that the values in the ranges look a little different when converted to decimal, as illustrated in the table below. This is because the subnet IDs end in non-zero values.

Subnet ID	Address Range	Broadcast Address
202.191.15.16	202.191.15.17 – 202.191.15.30	202.191.15.31
202.191.15.32	202.191.15.33 – 202.191.15.46	202.191.15.47
202.191.15.48	202.191.15.49 – 202.191.15.62	202.191.15.63

If you're confused by this example, it's probably because of the oddities that seem to occur in the ranges. For example, why is 202.191.15.16 not a valid IP address when a subnet mask of 255.255.255.240 is used? Look again at what happens when you convert this address to binary – the host portion is all binary 0s, which we already know is not allowed! The key to understanding address ranges is always the same – when in doubt, break everything down into binary. Looking at addresses in decimal only will often make things much more confusing than they need to be.

Stay tuned - in the next article in this series we'll walk through a full-blown subnetting example that will challenge you to piece together everything that you've learned in this series thus far!

Subnetting Challenge

To make things interesting, let's go through what I consider to be a tough subnetting question. To be honest, if you can follow this question and understand what you're doing, there isn't a subnetting question I can ask that you can't answer. We're going to go through it step-by-step, just to be sure that you're clear on determining the correct subnet mask, and the ranges of hosts on the first, second, and last subnets.

In this scenario, we're going to assume a network ID of 10.0.0.0. You should already be familiar with its default mask. Our requirement is to have at least 4000 subnets. Our goal

is to determine both the custom subnet mask, as well as the first, second, and last ranges of host IDs.

Let's start with the custom subnet mask. Since we need 4000 subnets minimum, we need to determine how many bits are required. It should already be obvious that we'll need to move beyond a single octet, since 8 bits will only provide a maximum of 256 subnets. If we try 10 bits, we come up with a value of 2^{10}-2 or 1022 subnets, obviously not enough. 11 bits provides 2046 subnets and 12 bits 4094 (2^{12}-2). Obviously 12 bits meets out requirement, so that's what we'll use.

Recall that the default subnet mask for the private 10.0.0.0 network address is 255.0.0.0. If we add an additional 12 bits, the mask becomes 255.255.240.0, as shown in Figure 5-21.

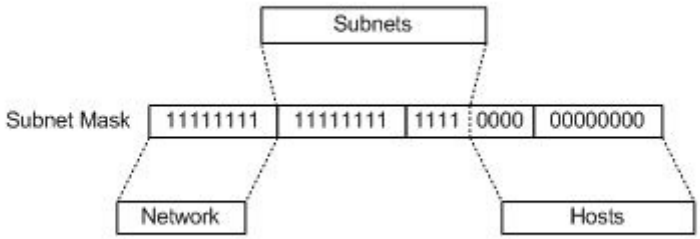


Figure 5-21: Network, subnet, and host portions based on the custom subnet mask 255.255.240.0.

Now that we know our custom mask, let's determine how many hosts we can support per subnet. From the diagram above, we have 12 host bits remaining, giving us a maximum of 2^{12}-2, or 4094 hosts per subnet.

Our next goal is determining the first valid range of IP addresses. Don't get thrown off by the fact that the subnet portion of the address spans one and a half octets. If we simply follow the same rules as before, we'll be fine. In this case, set all 12 subnet bits the lowest non-zero value, and determine the first and last valid host addresses, as shown in Figure 5-22.

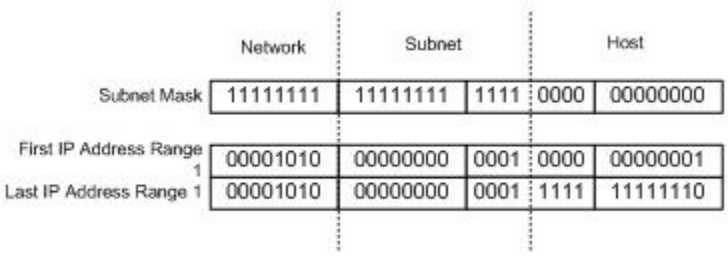


Figure 5-22: Defining the IP address range for the first subnet.

Notice that the first valid range of IP addresses on subnet 10.0.16.0 goes from 10.0.16.1 up to 10.0.31.254. The second subnet will start when we set the 12 bits of the subnet ID to 00000000 0010.

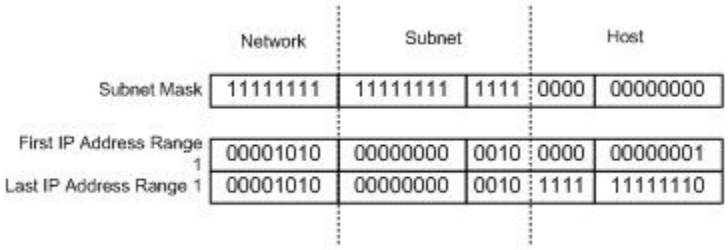


Figure 5-23: Defining the IP address range for the second subnet.

The second valid range of addresses becomes 10.0.32.1 to 10.0.47.254. The broadcast ID for this subnet is 10.0.47.255.

The subnet IDs will continue to increase until the last range is reached - remember that this occurs when the subnet ID is set to the highest value that is not all binary 1 values - in this case, 11111111 1110.

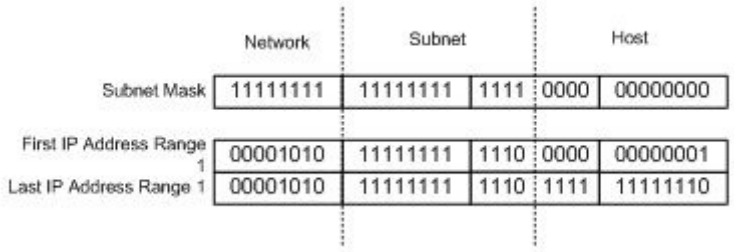


Figure 5-24: Defining the IP address range for the last subnet.

As such, our last valid subnet is subnet ID 10.255.224.0, with an address range from 10.255.224.1 up to 10.255.239.254. Remember that in between the three subnets that we've looked at, another 4091 subnets exist.

The Amazing Subnetting Shortcut Table

If you've been watching closely, you may have noticed the very clear pattern that develops when attempting to determine address ranges. Now that we have the long method out of the way, we can take a look at a quick shortcut. With any given subnet mask value, there is always an associated multiple at which new subnets will start. For example, if you are using the network ID 131.107.0.0, and a subnet mask of 255.255.252.0, subnetting is clearly taking place in the third octet. When a custom value of 252 is used in a mask, new address ranges always move in multiples of 4 in the third octet. For example, the first three ranges of addresses in this example would be:

- 131.107.4.1 - 131.107.7.254
- 131.107.8.1 - 131.107.11.254

- 131.107.12.1 - 131.107.15.254

Of course, we could calculate these the old fashioned way, breaking everything down to binary and then working from there. However, the pattern that exists can easily be put into a table to help you recall mask values and address range multiples quickly and easily, as illustrated in Table 5-12.

Table 5-12: Subnetting shortcut table.

Bit	1	1	1	1	1	1	1	1
Mask	128	192	224	240	248	252	254	255
Range Multiples	128	64	32	16	8	4	2	1

So what does this table tell you? Well, imagine that you were given the network ID 10.0.0.0 with a subnet mask of 255.224.0.0. If you used the table above, you could quickly determine the multiples at which new address ranges start. Simply look for the value associated with the custom mask in the Mask row - in this case 224. Now look in the Range Multiples row directly below 224, and you'll notice the number 32. When a custom subnet mask of 224 is used, new ranges always start at multiples of 32 in the second octet. So, in our example, the first three ranges of addresses would be:

- 10.32.0.1 - 10.63.255.254
- 10.64.0.1 - 10.95.255.254
- 10.96.0.1 - 10.127.255.254

While the table above works perfectly for Class A and B address ranges, it does present a small issue for Class C address subnetting. Recall that Class C addresses use only the last octet for subnetting. The table can still be used for these addresses, but with one small adjustment - you'll need to add 1 to the beginning of every range and subtract one from the end to account for the network ID and broadcast address.

For example, imagine the address 202.202.202.0 with a subnet mask of 255.255.255.240. In this case, according to our table, the ranges should begin at multiples of 16. This is still true, and the first three subnet IDs would be:

- 202.202.202.16
- 202.202.202.32
- 202.202.202.48

However, calculating the ranges means remembering that simple rule I just outlined. Add one to find the beginning of the range, and subtract one from what you might normally consider the end of the range. For example, the first range would go from 202.202.202.17 up to 202.202.202.30. This is because the "normal" end of the range (202.202.202.31) is actually the broadcast address for that subnet, where the host ID is set to all binary 1s. Altogether, the first three ranges in this example would be:

- 202.202.202.17 - 202.202.202.30
- 202.202.202.33 - 202.202.202.46
- 202.202.202.49 - 202.202.202.62

All things considered, the shortcut table provides a simple and effective way of recalling the address range starting points associated with a given custom subnet mask. It should be helpful as a quick reminder if jotted down before you write your exams. Just remember that when in doubt, always go back to the binary method.

Determining Whether Hosts are Local or Remote on a Subnetted Network

We spent time determining the valid ranges of addresses on a given subnet for a reason. Recall from our earlier look at TCP/IP communication that when a host wishes to communicate with another host, it must first determine whether the destination is local (on the same subnet) or remote (on a different subnet). In cases where hosts are local, they can communicate directly. In cases where the destination host is on a different network, the packets must be sent to a router, which will then forward them along on their journey to the destination network.

In order to determine whether a destination host is local or remote, a computer will perform a simple mathematical computation referred to as an AND operation. While the sending host does this operation internally, understanding what takes place is the key to understanding how an IP-based system knows whether to send packets directly to a host or to a router.

An AND operation is very simple - two binary digits are compared, and then based on their combination, a resultant value is formed. It is neither adding nor subtracting, so do not consider it as such. In the most simple terms, there are only three possibilities when ANDing two binary digits. Table 5-13 outlines these operations and their results.

Table 5-13: ANDing operations and resultant values.

AND Operation	Result
0 AND 0	0
0 AND 1	0
1 AND 1	1

Notice that when the binary digits 1 and 1 are ANDed, the result is 1, and that any other combination produces a result of 0.

The question now becomes how this is actually used. When a host wishes to figure out whether a destination host is local or remote, it goes through the following steps:

1. The host takes its own IP address and ANDs it with its own subnet mask, producing a result.
2. The host then takes the destination IP address and ANDs it with its own subnet mask, producing another result.
3. Finally, the host compares the two results. In cases where the ANDing results are identical, it means that the hosts reside on the same subnet. In cases where the results are different, it means that the destination host is remote.

Consider this example. Computer A has an IP address of 192.168.62.14 with a subnet mask of 255.255.248.0. It wishes to communicate with host 192.168.65.1. In order to determine whether

this destination is local or remote, it will go through the ANDing process. Its IP address and subnet mask are lined up in binary, and then vertically compared to find the AND result. The same is then done for the destination address, again using the subnet mask of the source host. This is illustrated in Figure 5-25.

Source IP	11000000	10101000	00111110	00001110	
Source Subnet Mask	11111111	11111111	11111000	00000000	
AND Result	11000000	10101000	00111000	00000000	= 192.168.56.0
Destination IP	11000000	10101000	01000001	00000001	
Source Subnet Mask	11111111	11111111	11111000	00000000	
AND Result	11000000	10101000	01000000	00000000	= 192.168.64.0

Figure 5-25: The ANDing process.

Notice that when the resulting AND values are converted back to binary, it becomes clear that the two hosts are on different networks. Computer A is on subnet 192.168.56.0, while the destination host is on subnet 192.168.64.0, which means that Computer A will next be sending the data to a router. Without ANDing, determining local and remote hosts can be difficult. Once you're very familiar with subnetting and calculating ranges of addresses, recognizing local and remote hosts will become much more intuitive.

Whenever you're in doubt as to whether hosts are local or remote, use the ANDing process. You should also notice that the ANDing process always produces the subnet ID of a given host.

Classless IP Addressing and Classless Inter-Domain Routing (CIDR)

By now you're familiar with the concept of classful addresses, where the default network and host portions of a network address can be easily identified by the value found in the first octet. While the classful system is simple and convenient, the scheme brings about some problems - mainly inefficiently large routing tables, and a wasteful use of the available 32-bit address space. In order to compensate for this, the idea of classless addressing was developed.

A classless address is just that - addressing without the common Class A, B, or C designations. Instead, classless addressing doesn't assume any class - it always includes the associated subnet mask (also referred to as the network "prefix") in order to determine the network portion of an address. Recall how classful addresses have a default subnet mask associated with them. In the classful world, routers could just assume the class of an address based on the network ID. In the classless world, subnet mask information must always be provided when routers exchange information with each other. Some routing protocols, such as the Border Gateway Protocol (BGPv4) and OSPF, support classless addressing. Others, like RIP version 1, do not. In Chapter 8 we'll look at routing protocols and their support for classless addressing in more detail.

In this section, we'll look at two classless addressing techniques that provide more flexibility in how IP addressing is defined - Classless Inter-Domain Routing (CIDR) and Variable Length Subnet Masks (VLSM). While CIDR and VLSM topics are mainly geared towards the CCDA exam, they are well worth knowing (especially CIDR notation) for the CCNA exam.

Classless Inter-Domain Routing

Recall that classful addresses are decidedly wasteful in the way they allocate addresses -even an entire Class B address range is too large for most companies. To make matters worse, a Class C block is so small that many companies would require many Class C ranges as a viable alternative. On the public Internet, routing tables were becoming extremely large, which in turn was affecting routing performance. A new solution was required to deal with the public address space more efficiently, and came about via a method referred to as Classless Inter-Domain Routing (CIDR).

CIDR is sometimes referred to as supernetting. While subnetting involves taking an address space and breaking it up into a number of smaller networks, supernetting is first and foremost a network aggregation scheme. For example, by supernetting four network addresses together, you can actually make them appear to be one network, as opposed to four. Again, this is all accomplished by playing with the subnet mask values.

CIDR has its own notation for dealing with subnet masks, and you may already be familiar with it. Instead of taking the time to say network 131.107.0.0 with a subnet mask of 255.255.0.0, CIDR notation truncates the subnet mask to what is known as "slash" notation. In this example, the network would be identified as 131.107.0.0/16. The "/16" value refers to the fact that the first 16 bits in the subnet mask are all set to values of binary 1.

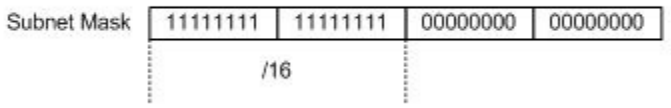


Figure 5-26: Determining CIDR notation based on a subnet mask.

This really is a more efficient way of referring to a network. For example, if we had a network address of 192.168.0.0 with a mask of 255.255.255.128, in CIDR notation it becomes 192.168.0.0/25. Again, the /25 means that the first 25 bits of the subnet mask are set to binary 1.

But what is supernetting really? Well, imagine a routing table on a large Internet backbone router. This router needs to know how to get to all public networks. If a certain company has been given many different network IDs (for example, an ISP might have 8 Class B network IDs), a minimum of 8 routing table entries would be required, just to be sure that these 8 networks could be reached. As routing tables get larger and larger, routing performance is definitely impacted. Supernetting provides a way to "collapse" or "summarize" those 8 entries into a single entry, by altering the subnet mask value.

Exam Tip: Supernetting allows contiguous network addresses to be summarized into a single routing table entry through the use of a custom subnet mask.

The first requirement to supernet addresses together is that network addresses must be contiguous. For example, you could supernet together networks 131.107.0.0 and 131.108.0.0, but not networks 131.107.0.0 and 145.36.0.0.

Supernetting is best illustrated with an example. Let's say that we wanted to aggregate the 8 network addresses between 131.0.0.0 and 131.7.0.0. In order to do this, we need to find a subnet mask value that makes all 8 network addresses appear to be on the same single network. Remember the ANDing process? By the time we're done, we should be able to AND together any two addresses in that range of 8 networks, and come up with the exact same value.

Recall that when subnetting, we stole bits from the host portion of an address, moving from left to right. When we're supernetting, we're going to do the opposite - instead of stealing bits from the host portion, we're going to steal low-order bits from the network portion. So this time, we're going to steal bits from right to left.

How many bits do we need to supernet 8 networks? Unlike with subnetting, we don't subtract 2 this time. In this case we need 3 bits, since 2^3 is 8.

Recall also that by default, a Class B address has a subnet mask of 255.255.0.0 (or /16). If we steal 3 bits from the network portion of the subnet mask, we end up with the mask shown in Figure 5-27.

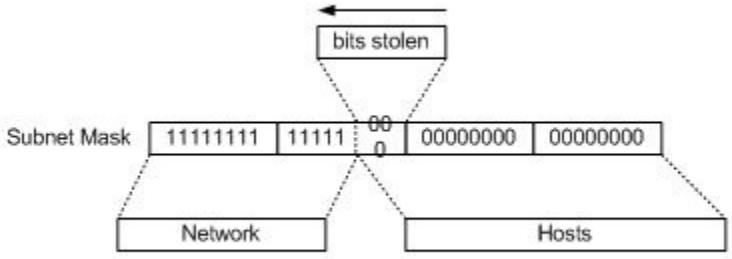


Figure 5-27: Network and host portions after using CIDR to create a /13 mask.

Supernetting really is no more difficult than that. Instead of stealing host bits, we steal network bits. If you look at Figure 5-28, notice that each of my 8 network IDs appear to be on the exact same network when I use a mask of 255.248.0.0, or /13. As such, the range can now be designated as 131.0.0.0/13. This value aggregates all addresses between 131.0.0.1 and 131.7.255.254.

/13 Subnet Mask	11111111	11111	000	00000000	00000000	
	Network					
131.0.0.0	10000011	00000	000	00000000	00000000	
131.1.0.0	10000011	00000	001	00000000	00000000	
131.2.0.0	10000011	00000	010	00000000	00000000	
131.3.0.0	10000011	00000	011	00000000	00000000	
131.4.0.0	10000011	00000	100	00000000	00000000	
131.5.0.0	10000011	00000	101	00000000	00000000	
131.6.0.0	10000011	00000	110	00000000	00000000	
131.7.0.0	10000011	00000	111	00000000	00000000	
Network ID	10000011	00000	000	00000000	00000000	= 131.0.0.0/13

Figure 5-28: Network portion of addresses after supernetting.

Think about why CIDR is so important. Small ISPs are the customers of larger ISPs, who may in turn be the customer of a Tier 1 network provider like WorldCom or AT&T. In cases like this, where a major ISP may have many smaller ISPs as customers, supernetting allows many networks to be reached through a single or smaller number of routing table entries. Just keep in mind that a large ISP is likely aggregating many Class B or C blocks of addresses together.

If you want a supernetting shortcut, just go back to the subnetting shortcut table that we looked at previously. Imagine that you wanted to supernet together 4 Class B addresses. You would simply look in the Range Multiples row, find the value of 4, and then look up. Supernetting 4 Class B addresses together is accomplished using a mask of /14, or 255.252.0.0. Just remember which way you're stealing bits!

It's really important to remember that only certain contiguous ranges will work together when supernetting. For example, if you had the network address 165.43.0.0 and wanted to supernet it with 8 other ranges, you would need to determine the correct 8 ranges that the 165.43.0.0 network can be aggregated with. The easiest way to do this is to AND the 165.43.0.0 network address with the 255.248.0.0 subnet mask (since we already know that supernetting 8 class B addresses uses this mask). The answer from the ANDing process will show you the beginning of the range.

Source IP	10100101	00101011	00000000	00000000	
Source Subnet Mask	11111111	11111000	00000000	00000000	
AND Result	10100101	00101000	00000000	00000000	= 165.40.0.0

Figure 5-29a: Calculating the network ID based on a custom subnet mask using ANDing.

There is also a very simple trick to bypass the need for binary conversions when performing AND operations. It is a very slow and cumbersome technique to convert everything to binary, run the AND operation, and then convert back to decimal. The following is a really simple shortcut using the Windows Calculator, since the AND operator works directly on decimal numbers. Simply

punch in 43, hit the AND operator, as shown in Figure 5-29b, and then 248 and [Enter] to instantly get 40. This makes mask calculations a lot easier.

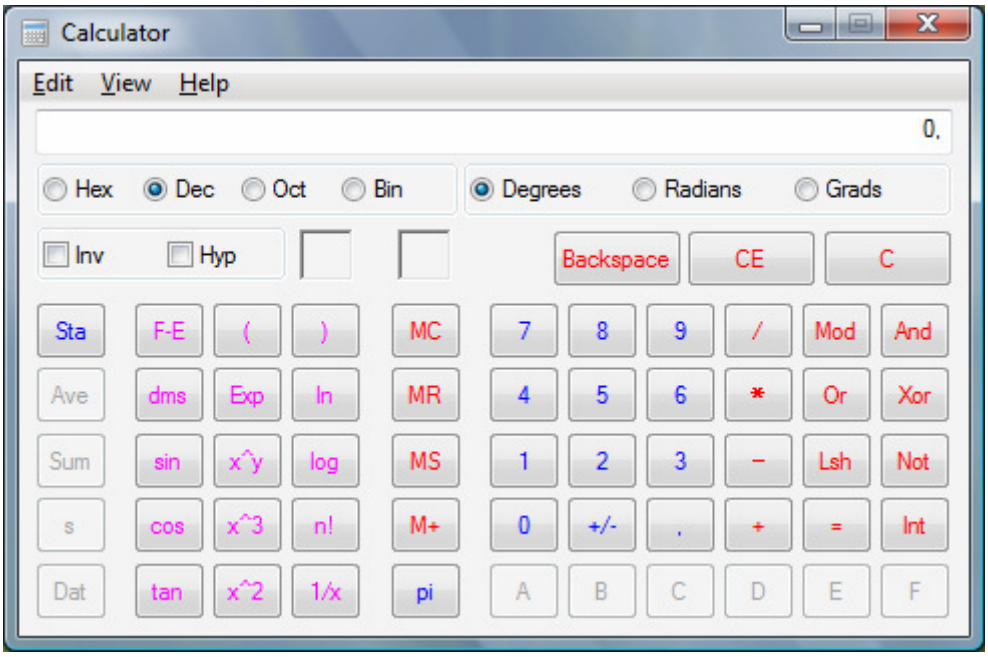


Figure 5-29b: Calculating the network ID using the Windows Calculator AND operator.

Since the network ID becomes 165.40.0.0 after the ANDing process, we now know that the 165.43.0.0 network ID must be aggregated with the 8 network IDs that span from 165.40.0.0 up to 165.47.0.0.

CIDR use is not limited to the public Internet. It can also be used on internal company networks to aggregate routes behind a smaller number of routing table entries. For example, imagine that the subnets 131.107.8.0/24 through 131.107.15.0/24 all exist behind one router. Instead of adding entries for all eight networks in the routing table, a single entry could be added for network 131.107.8.0/21 instead. In effect, the address 131.107.8.0/21 "summarizes" all eight networks into one network routing table entry. This technique, known as route summarization, is also used by some routing protocols to reduce the number of networks that they need to advertise to other routers. Route summarization is looked at in more detail in Chapter 8. Ultimately, techniques like route summarization make routing faster and more efficient, and are especially useful within large organizations.

Study Tip: For more information on Classless Interdomain Routing, see RFC 1519.

Understanding Variable Length Subnet Masks (VLSM)

One thing that you may have noticed during our look at subnetting was that each and every subnet defined was exactly the same size, based on a single subnet mask value. While this is simple, it is again very wasteful. In most organizations, LANs are not all the same size. While some subnets may have hundreds of hosts, others may have a very small number, perhaps 10 or

20. More obvious still is a WAN link connecting two locations, which only requires 2 addresses. In cases where you have subnetted a Class B network using a /20 mask, that leaves you with a fixed 4094 hosts per subnet. On a WAN link, that alone would mean wasting 4092 addresses!

While you may have plenty of subnets and address space for your smaller network, on very large networks proper address management can be crucial. Using Variable Length Subnet Masks (VLSM), this problem can be solved. VLSM allows you to use different subnet mask values throughout a network, in order to better account for how many hosts you have on each subnet. Like all classless addressing, VLSM relies on the associated subnet mask (prefix) information to be explicitly provided in order to determine the network portion of an address. While VLSM provides some very useful addressing capabilities, it also requires some careful planning.

At the most basic level, VLSM represents a hierarchical network addressing scheme. Think of it as subnetting multiple times within the same address range. For example, we can take the network 192.168.0.0 and subnet it into a number of large subnets. In our case, these will be used to define large geographic regions. Next, we look at those regions (or large subnets) individually, subnetting them further. This allows us to break up a region for the purpose of defining custom subnets that meet our size requirements. This additional subnetting can continue until you run out of subnets within a given range. A properly designed hierarchical addressing scheme both makes better use of the address space, and also provides for more optimized routing tables, as we'll see shortly. Figure 5-30 outlines the basic idea behind a hierarchical addressing arrangement. Note that I have only shown the breakdown of a single large subnet rather than both large subnets, in order to keep things clear.

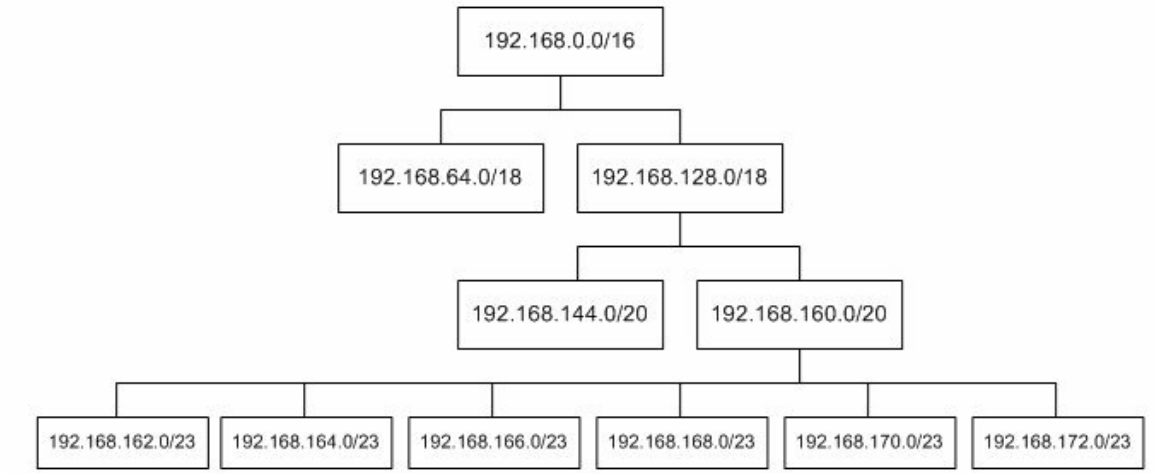


Figure 5-30: Hierarchical Addressing using VLSM.

Using VLSM addressing on a network is slightly more involved than using the same subnet mask throughout. You again need to properly characterize your network, defining how many WAN links, subnets, and hosts per subnet you are dealing with. The easiest way for us to begin is with a single network address. In this case we'll use the private address 192.168.0.0/16. I'm going to assume that our company has four main geographic regions, and that each region includes many smaller offices.

Let's start off by breaking our 192.168.0.0/16 network into 6 large subnets, since using the required 4 would only account for current regions, and would not allow for future growth. Doing

this is not in any way different than what we've done in the past. To get 6 large subnets, we'll need to steal 3 bits, since 2^3-2 gives us 6 subnets. In this case, our new mask becomes 255.255.224.0, or /19. Each of the new subnet IDs is listed in Table 5-14, including the location that it will be used for.

Table 5-14: Network IDs for main geographic areas.

Area / Location	Subnet ID
North America	192.168.32.0/19
South America	192.168.64.0/19
Europe	192.168.96.0/19
Asia	192.168.128.0/19
Future Range 1	192.168.160.0/19
Future Range 2 or WAN Links	192.168.192.0/19

Since we've used a /19 subnet mask, each of these ranges supports up to 2^{13}-2 or 8190 hosts as things presently stand. However, we're not done. Our next goal is to take one of the areas and further divide it into a number of smaller networks. For example, let's say that North America includes 16 offices. Of these, 3 are large, and need to support up to 400 hosts. 13 are smaller, and need to support 150 hosts maximum. In the future, it is possible that more large and small offices will be added.

In order to deal with these requirements, we need to subnet the North America network further. The easiest way to deal with this is to consider 192.168.0.0/19 to be just another network ID, and our starting point. To begin, we know that we need to immediately support 3 large offices, each with up to 400 hosts. For this example, I'm going to have us define one additional large office, in order to account for future growth. That gives us a requirement of 4 networks, and 400 hosts per network. Using the host portion, we would need 9 host bits in order to support our requirement, which would give us 2^9-2 or 510 hosts per subnet. In doing so, we are left with 4 bits to define subnets, as shown in Figure 5-31. Remember that our starting subnet mask is 255.255.224.0 or /19 in this case.

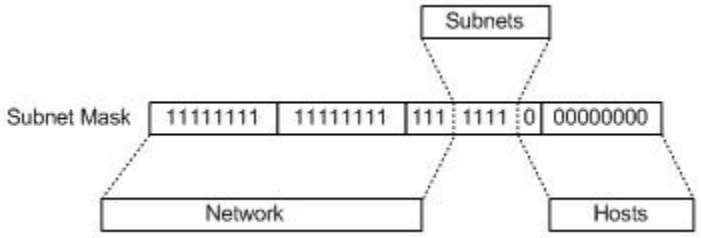


Figure 5-31: Dividing the 192.168.0.0/19 address range into additional subnets for large LANs.

Using the first 4 available ranges to define our four large LANs, we would allocate the subnet IDs as shown in Table 5-15.

Table 5-15: Subnet IDs for North America's large LANs.

Area / Location	Subnet ID	Address Range
Toronto	192.168.2.0/23	192.168.2.1 - 192.168.3.254

San Jose	192.168.4.0/23	192.168.4.1 - 192.168.5.254
New York	192.168.6.0/23	192.168.6.1 - 192.168.7.254
Future Growth	192.168.8.0/23	192.168.8.1 - 192.168.9.254

Notice from the table that many address ranges are still available - everything from 192.168.10.0 through 192.168.29.0 for North America. For that reason, we can again subnet the address range further to account for the smaller offices. For these smaller offices, we'll use a /24 subnet mask - this meets our requirement of supporting up to 150 hosts on each of the small LANs. In this case, we can support 254 (2^8) hosts on each. Table 5-16 outlines the 13 subnets for the small offices, along with a few defined for additional growth.

Table 5-16: Subnet IDs for North America's small LANs.

Area / Location	Subnet ID
Small Office 1	192.168.10.0/24
Small Office 2	192.168.11.0/24
Small Office 3	192.168.12.0/24
Small Office 4	192.168.13.0/24
Small Office 5	192.168.14.0/24
Small Office 6	192.168.15.0/24
Small Office 7	192.168.16.0/24
Small Office 8	192.168.17.0/24
Small Office 9	192.168.18.0/24
Small Office 10	192.168.19.0/24
Small Office 11	192.168.20.0/24
Small Office 12	192.168.21.0/24
Small Office 13	192.168.22.0/24
Small Office 14	192.168.23.0/24
Future Small Office	192.168.24.0/24
Future Small Office	192.168.25.0/24

Notice that even after dividing up the address space for our small offices, we still have additional address space left over - in this case, everything from 192.168.26.0 up to 192.168.29.0 is still unassigned. This space could be used to assign additional large or small subnets in North America, as need dictates. However, we still have one last requirement - WAN links.

Remember that a point-to-point WAN link still counts as a subnet, even if it is limited to only 2 hosts. In order to account for our WAN links, let's use the range 192.168.29.0/24, and subnet it further. In this way, we have left the entire address space between 192.168.24.0 and 192.168.28.0 for defining additional North America LANs if required.

Since we only require 2 addresses for each WAN link, we only need 2 host bits - 2^2-2 is 2, which meets our requirements. If we make all the remaining bits subnet IDs, we can support up to 2^6-2 or 62 WAN subnets within North America alone. This will easily meet our needs. Based on this scenario, our WAN links will use a subnet mask of 255.255.255.252. Table 5-17 outlines only the first 4 WAN links, since listing all 62 would serve little purpose.

Table 5-17: First 4 subnet IDs for North America WAN links.

Link	Subnet ID	Address Range
WAN Link 1	192.168.31.4/30	192.168.31.5 - 192.168.31.6
WAN Link 2	192.168.31.8/30	192.168.31.9 - 192.168.31.10
WAN Link 3	192.168.31.12/30	192.168.31.13 - 192.168.31.14
WAN Link 4	192.168.31.16/30	192.168.31.17 - 192.168.31.18

Obviously VLSM gives us better control of the size of our subnets, allowing us to allocate addresses in our chosen space more efficiently. But what other benefits does this scenario bring? Well, one is certainly smaller routing tables. For example, all networks in North America can now be reached from any other geographic region via a single routing table entry - 192.168.0.0/19. Once any data for North America is sent to this router, it then decides where the data needs to be forwarded next. If we weren't using CIDR and VLSM in this example, each router's routing table would need entries for every network in North America. By aggregating all the entries behind a single entry, routing performance will be greatly increased - especially on very large networks.

Internet Protocol Version 6 (IPv6)

It's not required knowledge for the CCNA exam, but CCDA candidates do need to be familiar with the ins and outs of IPv6. I covered the IPv6 material from Chapter 5 of my study guide a few month's back, so here's a list of direct links to those tutorials. That does it for Chapter 5 articles, stay tuned for Chapter 6!

Introduction to IPv6

Subnetting and Address Allocation on IPv6 Networks

IPv6 Operation

IPv6 Deployment Strategies

Introduction to IPv6

One new and prevalent topic on the 640-861 CCDA exam is Internet Protocol version 6 (IPv6). While you certainly are not expected to know and understand IPv6 to the same level of depth as IPv4, you will need to know the basics, including how IPv6 addresses are formed, displayed, subnetted, and allocated. Further to this, you will also need to have an awareness of the various IPv6 deployment strategies available, including the relative advantages and disadvantages of each method. Finally, you should be familiar with some of the changes to routing protocols introduced as a result of IPv6. It’s important to remember that this series of articles should not be considered the definitive, “all you will ever possibly need to know” source for IPv6 information – that would easily fill a dedicated book all by itself, and many IPv6 elements are still not completely finalized as standards. For these reasons, this section concentrates on the specific IPv6 concepts that you will need to be familiar with for the CCDA exam.

Introduction to IPv6

Before getting into the technical details of this new version of IP, a simple question must be answered – why does the world need a new version of IP at all? There are many answers to this question, but the most basic reason involves the rapid depletion of the IPv4 address space. If you'll recall from earlier in this chapter, IPv4 uses 32-bit addressing, and this limits the total number of IP addresses available for issue. When the Internet Protocol was first defined, nobody envisioned the phenomenal growth that defines the Internet we know today. As such, what was originally considered an almost endless supply of address space is now challenged as increasingly greater numbers of devices are connected to the Internet. Consider examples like cellular phones, PDAs, and other embedded systems, and it's easy to see how the need for more IP address space is upon us.

Note: Talking about IPv6 begs a simple question – whatever happened to IPv5? The answer to that is that IPv5 is actually defined, although for a different purpose. IPv5 defines an experiment protocol that was originally developed to provide quality of service (QoS) features on IPv4 networks. You don't need to know anything about IPv5 for the CCDA exam, but hopefully this helps to clear up what might have been a nagging thought. To that end, it's also an interesting little piece of tech trivia!

A number of techniques have been developed and implemented to help slow down the need to deploy a new version of IP, and of these, network address translation (NAT) is clearly the most popular. Earlier in this chapter we also looked at the three ranges of IP addresses set aside for private addressing. While this technique has reduced the need for public IP addresses, it is still not a proper solution, and does little to address the foreseeable growth of the Internet in the longer term. Quite simply, a larger IP address space is required, and IPv6 addresses this issue with a 128-bit address space. At this time, a 128-bit address space would be capable of providing more than a thousand IPv6 addresses for each man, woman, and child on the planet.

While the need for a larger address space was the primary consideration in defining a new version of IP, it was far from the only one. Over time, many lessons has been learned about how IP could be improved in terms of providing better performance and security features, as well as methods to simplify host configuration. Some of the major advantages that IPv6 provides over IPv4 include:

Larger address space. By using a 128-bit address space, the total number of IPv6 addresses available is approximately 3.4×10^{38}, or more than one thousand addresses for every person on the planet. In case you care, that's 340,282,366,920,938,463,374,607,432,768,211,456 (340 undecillion) IP addresses!

Simplified header format. IPv6 removes many of header fields previously found in the IPv4 header, including those associated with packet fragmentation, which is no longer supported. This helps to make packet processing on routers easier, and ultimately faster. Header extensions are also supported, allow for a range of options to be included in packets both now and in the future.

Hierarchical addressing format. IPv6 uses hierarchical addressing techniques that associates networks and allows them to be aggregated in a manner much more efficient than with IPv4. Ultimately, this will help to reduce the size of Internet routing tables.

Address autoconfiguration techniques. IPv6 introduces different autoconfiguration techniques to simplify the configuration of IPv6 interface addresses.

Mandatory support for IPSec. Unlike in IPv4 where IPSec support in the IP protocol is optional, IPSec support is mandatory in IPv6.

Elimination of the need for NAT. Based on the size of its address space, IPv6 eliminates the need for network address translation (NAT) devices. Outside of being a potential single point of failure, NAT devices break the end-to-end connectivity goal of IP.

Increased multicast address space. IPv6 provides a much larger multicast address space than IPv4, and improves network efficiency as a protocol by not supporting broadcast transmissions.

So when will IPv6 finally be used on the worldwide Internet? The answer is that it already is! A portion of the today's Internet, known as the 6bone, has been running IPv6 since 1996. Both companies and service providers already connect to this network as part of planning the eventual but inevitable migration to IPv6. To find out more about the 6bone, visit http://www.6bone.net.

IPv6 Addressing and Subnetting

It's easy to be a little intimidated when you come across an IPv6 address for the first time in its fully expanded form. First of all, at 128 bits in length, an IPv6 address is a full four times longer (in bits) than its IPv4 equivalent. Because of their length, IPv6 addresses are represented in hexadecimal rather than the common dotted decimal notation you are familiar with from IPv4. However, before you start worrying about endlessly typing (and calculating) hexadecimal characters on each and every host on your network, it's worth noting that IPv6 includes a variety of methods to make address deployment easier for administrators, features that I'll cover in the IPv6 operation section below.

Beyond handling addresses differently, IPv6 also changes the method by which a network is subnetted. If anything, this method is actually simpler than with IPv4, because an IPv6 address includes a section to define a subnet number directly within the address itself. Once you understand where that field is, determining the subnet to which a host belongs is as easy as just reading and matching the number listed.

In this section I'll walk you through some of the basic of IPv6 addressing and subnetting, including how IPv6 addresses are formatted and represented, how subnets are determined, how the address space itself is allocated, and the different types of addresses and transmission methods that exist.

IPv6 Address Format

As mentioned above, an IPv6 address is 128 bits in length, represented in hexadecimal. Much like a MAC address, an IPv6 address is broken down into 2-byte (16-bit) sections separated by a colon; the major difference being that an IPv6 address includes 8 of these sections rather than 3 with a MAC address. In fact, the standard configuration of an IPv6 address actually uses a system's MAC address as part of the interface ID of a host, as I'll explain shortly. The address below shows an example of a full 128-bit IPv6 address:

2031:0000:130D:0000:0000:08D0:875D:130A

The first thing to recognize about an IPv6 address is that it can actually be compressed quite easily. For example, in the address above, each 2-byte section that contains only 0s can be reduced to a single 0. By the same token, any 2-byte section that begins with a 0 can also have that leading 0(s) dropped. In other words, in compressed form the address just considered would be represented as:

2031:0:130D:0:0:8D0:875D:130A

Although this is a little better, there is still an even easier way to represent an IPv6 address. Because the IPv6 address space is so large, there's a good change that you're likely to find many 0s in any given address. So, in cases where more than more than one successive field contains 0s only, you can represent it with double colons (::) in the address. For example, the compressed address 2031:0:130D:0:0:08D0:875D:130A could be represented as:

2031:0:130D::8D0:875D:130A

When an IPv6 system comes across this double colon (::) within an address, it knows that it should include as many 0s as necessary to get the address back to 128 bits. However, it's very important to note than the double colons can only be included once within a given address – if it were included more than once, a system couldn't possible know where all the 0s were to be placed in the expanded address.

Tip: Remember that double colons (::) can be used as placeholders for contiguous 0 fields in an IPv6 address, but only once within any given address.

In some cases, using the pair of colons makes an address very small indeed. For example, the loopback address in IPv6 is 0000:0000:0000:0000:0000:0000:0000:0001. By using the double colon arrangement just learned this address can also be identified as:

::1

Overall, IPv6 addressing is probably neither easier nor harder than what you've come to know and love with IPv4 – it's simply different. As with anything new, it will just take a little time to understand and appreciate this new addressing format.

While this article gives you an introduction to IPv6, we're just getting started. In my next CCDA article I'll walk you through how subnetting occurs in this address space, and we'll explore different IP address allocation and transmission methods. Stay tuned!

Subnetting and Address Allocation on IPv6 Networks

In the previous article in my Cisco CCDA series, you were introduced to IPv6 and learned the basics of how addresses are formed and different notation methods. Continuing down the IPv6 path, this article takes a closer look at how subnetting takes place in the world of IPv6, and then outlines the different address allocation and transmission methods used on IPv6-based networks.

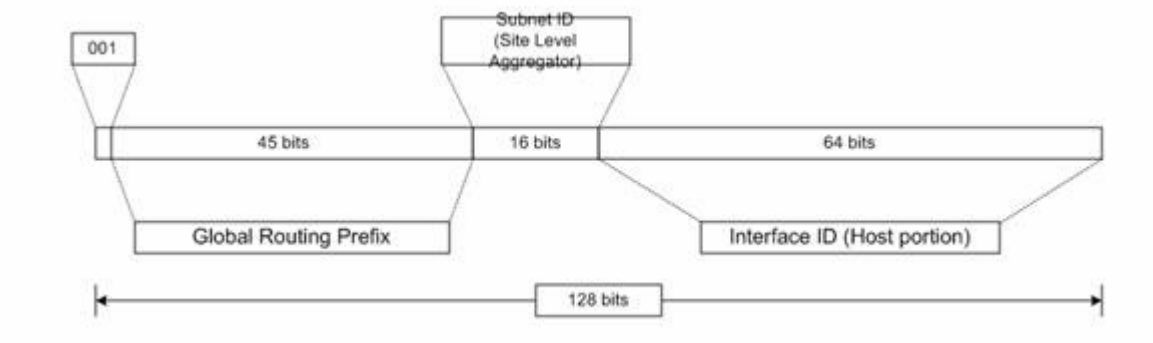
Subnetting the IPv6 Way

If you've been working with IPv4 for some time now, you may already be familiar with subnet masks and how they work to segment the IPv4 address space into subnets. In the IPv6 world, subnetting works somewhat differently, relying on a dedicated field within an IPv6 address. While the next section will look at the breakdown of the IPv6 address space in more detail, for now it's enough to say that an IPv6 unicast address includes a 16-bit field known as the Subnet ID or Site-Level Aggregator. Because this field is 16 bits in length, it gives companies the option of configuring up to 65535 individual subnets. The structure of an IPv6 global address is outlined below.

Tip: Remember that in IPv6, the Subnet ID (also know as the Site Level Aggregator field) is used to define individual subnets.

IPv6 Address Allocation and Transmission Methods

As part of developing the IPv6 address space, a number of the "problems" associated with IPv4 were taken into account. For example, IPv6 provides a much more organized hierarchical addressing scheme, addressing some of the limitations and problems associated with routing in the IPv4 world. The figure below outlines the major elements of a global IPv6 unicast address. To gain a better understanding of IPv6, it is worth knowing a little more about how these addresses are allocated in the real world.



First and foremost, the first three bits of the global IPv6 address space are set to use the prefix 2000::/3 (remember, this is not the decimal number 2000, but a series of four hexadecimal digits). Like the CIDR notation you are already familiar with, the /3 represents a mask that defines a portion of the address space. In this case, all IPv6 addresses that start with the binary values 001 (2000::/3) through 111 (E000::/3) are global addresses (with the exception of FF00::/8, which are addresses reserved for multicasts). Ultimately, these global addresses need to have a 64-bit interface identifier, as displayed in the previous figure. The 64-bit interface identifier is usually created by taking an interface's MAC address (which is 48 bits in length) and wedging another 16-bit hexadecimal string (FFFE) between the OUI (first 24 bits) and unique serial number (last 24 bits) of the MAC address. This format is known as extended universal identifier (EUI) 64 format, or EUI-64 for short. When all is said and done, the last 64 bits of an IPv6 global address represent an interface.

So what are the other parts of the address space used for? Well, so far we know that the last 64 bits of an IPv6 address represent a unique interface, while the 16 bits that precede that represent the Subnet ID. As such, the first 48 bits define what is known as the Global Routing Prefix, and since the global address space starts at 2000::/3, that leaves 45 bits to break up the Global Routing Prefix itself.

Without getting into too much detail here, the IPv6 address space is allocated by the Internet Assigned Numbers Authority (IANA). The IANA assigns addresses to the various registries, such as the American Registry for Internet Numbers (ARIN) in the Americas. A registry is given a /16 portion of the address space, such as the 2001:0400::/16 address space allocated to ARIN. From this allocated space, a registry such as ARIN would begin granting address space to ISPs with a /32 prefix. Then, individual ISPs would allocate this address space to organizations using a /48 prefix.

Once a company has been granted their address in the /48 space, they can begin to allocate this address space internally, segmenting the space into smaller subnets or networks by using the 16-bit Subnet ID field. From there, hosts are addressed using the remaining 64 bits of the address space as outlined earlier.

As if it we not enough that IPv6 introduces a whole new addressing scheme, this version of IP also introduces a new concept in terms of how hosts are individually addressed. For example, in the world of IPv4, a host usually had a single IP address assigned to it. In the world of IPv6, however, a host is assigned multiple types of addresses on a per-interface basis. These addresses include different types of unicast, multicast, and anycast addresses. One that you may find conspicuously absent is the famous broadcast – in fact, you might be happy to know that IPv6 doesn't support broadcasts at all.

Unicast Addresses

Like in the world of IPv4, a unicast transmission represents data meant for a single destination address only. However, IPv6 uses a few different types of unicast addresses for different purposes. These include global, site-local, link-local, and IPv4-mapped IPv6 addresses. Each is outlined below.

Global Unicast Address. Very similar in function to an IPv4 unicast address such as 131.107.1.100, these addresses include a global routing prefix, a subnet ID, and an interface ID as outlined earlier.

Site-Local Unicast Address. Very similar in function to the IPv4 private address space that includes ranges like 10.0.0.0/8, these addresses are meant for internal communications and are not routable on the public Internet. Site-local addresses start with the prefix FEC0::/10, and then include the appropriate subnet ID and interface ID as outlined earlier.

Link-Local Unicast Address. For certain communications that are meant to stay within a given broadcast domain, IPv6 uses link-local addresses. These addresses are used for features like stateless autoconfiguration, which will be looked at shortly. Link-local addresses start with the prefix FE80::/10, and then include an interface ID. Note that since these addresses never communicate outside of their local subnet, the subnet ID is not included.

IPv4-mapped IPv6 Address. For environments that are transitioning between IPv4 and IPv6, IPv6 provides another type of unicast address known as an IPv4-mapped IPv6 address. This addressing method is used on systems running both an IPv4 and IPv6 protocol stack. When used, a system will include its current 32-bit IPv4 address in the low-order bits of an IPv6 address, preceded by 16 bits set to FFFF, and the remaining bits set to 0. For example, a host with the IPv4 address 131.107.1.100 would use the address of 0:0:0:0:0:FFFF:131.107.1.100.

Anycast Addresses

IPv6 also defines an entirely new type of address and transmission, known as an "anycast". Simply put, an anycast address is a standard IPv6 global address that is assigned to a number of different interfaces on different systems. When a packet is destined for an anycast address, the "closest" device to the sender will process the packet. In this case, the concept of "closest" is defined by the routing protocols in use on the network. At this time, anycast addresses can only be used as a destination address, cannot be used as a source address, and are only assigned to routers. A common use in IPv6 is to apply the same anycast address to all routers interfaces that connect to the same subnet. The potential uses of anycast transmission methods are being

explored further by the IETF, and the technique is already finding its way into new technologies like Mobile IP.

Multicast Addresses

Must like the reserved Class D address space in IPv4, IPv6 dedicates some of its address space to multicast traffic, albeit a much larger portion. If you recall, a multicast transmission is one in which a single transmission is received by many systems, or a one-to-many technique. In IPv6, multicasts use the prefix FF00::/8. Common examples of multicast addresses used in IPv6 include the destination address FF02::1, which is used to send a multicast to all hosts on a given subnet. Similarly, the multicast address FF02::2 is used to communicate with all routers on a subnet. Later in this series you'll learn more about how some routing protocols use multicasts to facilitate inter-router communication.

Coming Up

Now that you're familiar with how subnetting occurs and the different transmission methods supported in IPv6, it's time to look at how IPv6 actually works. We'll get to that in the next article in this series, where we'll take a closer look at the "operation" of IPv6 communication. Until then, all the best!

IPv6 Operation

While IPv4 is both well understood and widely implemented in the world of internetworking, IPv6 introduces a number of changes to the operation of the Internet Protocol that you will need to be familiar with for Cisco's CCDA exam. In this article I cover some of the techniques that an IPv6 host uses to facilitate network communication, how some common IPv4 transmission mechanisms have changed, and ultimately, the different ways in which IPv6 addresses can be assigned to host interfaces.

Discovery Processes

On an IPv6 network, a number of important functions happen using discovery processes. These include the discovery of neighboring devices and routers, as well as the maximum transmission unit (MTU) that is supported between a source and destination host. Some of these concepts are similar to ones found on an IPv4 network, while others represent new ways of dealing with traditional IPv4 configuration issues.

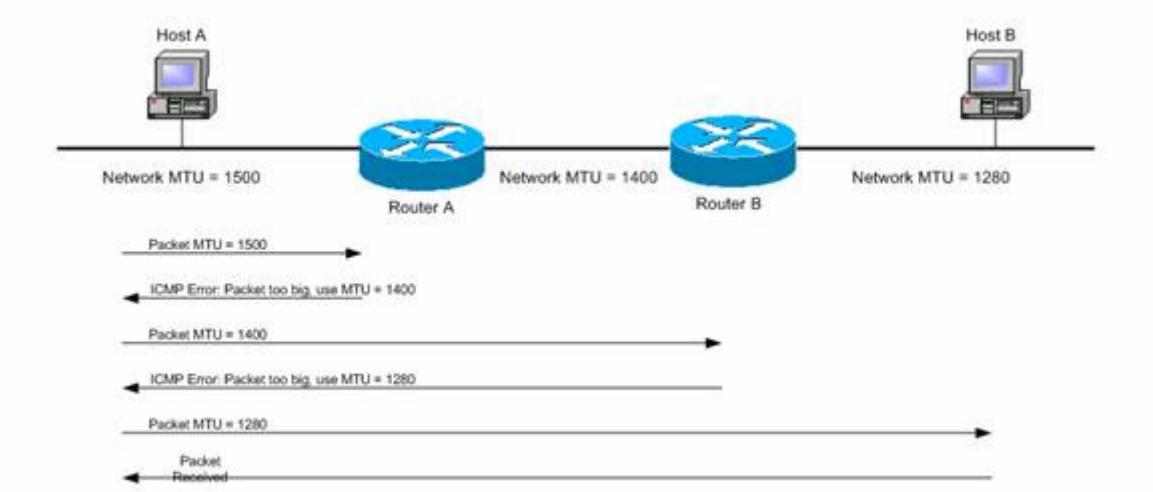
Neighbor discovery is the process by which an IPv6 node discovers the link-layer address of systems that it needs to communicate with on the local subnet, and is the method by which a node keeps track of local routers. This neighbor discovery process uses a new version of ICMP – ICMPv6. Ultimately, multicasts and anycasts are used for neighbor discovery functions on an IPv6 network. For example, recall the ARP function on an IPv4 network; a host would send out a broadcast requesting that the host with the specified IP send back its MAC address. In IPv6, ICMP multicasts are used to send out a request looking for the link-layer address associated with a known IPv6 address. This helps to reduce some of the traditional issues associated with broadcast traffic negatively impacting network performance.

Router discovery is a feature of IPv6 that allows an IPv6 node to discover the routers connected to its local link network. Although a similar feature exists in IPv4, it is rarely used with most administrators relying upon manually configured gateway addresses instead. Two main types of router discovery messages are used on IPv6 networks – router solicitations, and router

advertisements. A router advertisement is a multicast message periodically sent by an IPv6 router that allows a host to gather valuable information about the network. For example, a router advertisement could contain information about the address configuration method that should be used, the IP prefixes in use on the network (or any changes to them), which router should be considered the default router, and more. Router advertisements will be looked at in more detail in the section on addressing hosts.
In contrast, a node sends out a router solicitation method when it does not have a configured IP address at start up. The purpose of this multicast is to gather information about how the node should be configured, but without the need to wait for the next router advertisement. For example, the result of a router solicitation message could be a reply from a local router specifying that the host should auto-configure its IPv6 address, or perhaps that it should use DHCP instead.

Another very important discovery process on IPv6 networks is maximum transmission unit (MTU) discovery. In previous articles in the CCNA series, we looked at how a router was capable of fragmenting an IPv4 packet when the next network on the path to a destination used a smaller MTU size – for example, when forwarding data from a Token Ring segment to an Ethernet segment. While this method helped to avoid some of the issues associated with interconnecting different network types, it also slowed down the communication process, since a router was not only responsible for reframing and making forwarding decisions, but also fragmenting packets and subsequently reframing them all as well. In IPv6, routers no longer fragment any packets. Instead, the sending node uses a process referred to as MTU discovery to determine the larger possible MTU that is supported between itself and the destination host. If any fragmentation needs to take place, it must be done on the sending node – IPv6 routers stay out of this process completely, leading to greater routing efficiency.

To understand how MTU discovery works, consider the figure below. In it, we see a source host attempting to discover the biggest MTU possible between itself and the destination. In this case, the MTU between Host A and its local router is 1500 bytes, the MTU between Routers A and B is 1400, and the MTU between Router B and the destination is 1200.



To discover the MTU, Host A will sent out a packet to Host B attempting to use its 1500 byte MTU. At Router A, an ICMPv6 error message (packet too big) will be sent back saying that an MTU of 1400 should be used. Host A will then send out another packet with an MTU of 1400, which will be designated as too big by Router B, with a maximum MTU of 1280 specified. Since Host B is connected to a network with an MTU of 1280, Host A now knows that this is the MTU that it should use to communicate with Host B. Although this process may seen cumbersome, it's

worth noting that IPv6 specifies a minimum MTU size of 1280 bytes, and that 1500 is usually the default MTU configured on most internetworking equipment.

Addressing Hosts

Throughout this section there have been clues as to how hosts, and specifically interfaces, obtain their IPv6 addresses. The three methods by which an IPv6 host can obtain an IP address include stateless autoconfiguration, statefulautoconfiguration, or manually.
Stateless autoconfiguration is the easiest IPv6 address allocation method available. When used, stateless autoconfiguration uses the network prefix information contained in router advertisements as the first 64 bits of its addresses, and then appends its MAC address in EUI-64 format as the interface portion. This method is especially useful in environment where a DHCP server is neither configured nor present. On local networks without a router, a host using stateless autoconfiguration will use the link local network prefix and append to this its EUI-64 format MAC address.

For a higher degree on control over which addresses IPv6 interfaces use, statefulautoconfiguration can use another addressing method. When an IPv6 node sends out its router solicitation message at startup, the router can be configured to include whether a DHCP server should be used in its reply. If a DHCP server should be used, the node will used attempt to find a DHCP server through the use of multicasts. This is again an improvement over IPv4, where clients attempting to lease an IP address from a DHCP server use broadcast messages.

Finally, IPv6 addresses can also be configured manually. While generally not suggested for individual hosts, certain network nodes (such as routers) will require explicit configuration. Given the length and complexity of IPv6 addresses, it is generally best to use either stateful or stateless autoconfiguration for hosts to reduce potential errors and keep things simple.

Note: You may be curious about how DNS works in an IPv6 environment. Not surprisingly, the method is very similar to DNS in IPv4. However, when a host is attempting to obtain the IPv6 address associated with a fully qualified domain name (FQDN) or hostname, it sends a DNS query looking for the AAAA record associated with the host, rather than the standard A record used to resolve IPv4 addresses.

Coming up next

Now that you have a basic understanding of how IPv6 networks operate, it’s time to move on to a look at the different ways in which the IPv6 protocol can be deployed. That’s the focus of the next article in this series, so stay tuned!

IPv6 Deployment Strategies

One of the primary issues associated with deploying any new protocol on a network is the method in which the deployment will take place. While it would be nice if there were some global switch that could simply be “flipped” making such a change a one-step process, reality is unfortunately a little harsher. However, a number of different IPv6 deployment techniques are available to suit the needs of different environments. For the purpose of the CCDA exam, you should be familiar with the different IPv6 deployment methods outlined here, as well as the relative advantages and disadvantages of each.

Dual Stack Technique

The dual stack technique is likely to become quite common as companies make the transition

from IPv4 to IPv6. As the name suggests, this method involves running both IPv4 and IPv6 protocol stacks on network equipment such as hosts and routers until the transition to a purely IPv6 network can be completed. Under this scenario, Cisco routers on a network would be configured to route both IPv4 and IPv6 traffic, with hosts configured to use both protocol stacks as well. Cisco has already developed versions of its IOS software that support both protocol stacks simultaneously. Unfortunately, it will likely take quite some time before all of the applications that end users require access to support IPv6. As such, using a dual-stack technique would quite likely be a long-term arrangement, requiring two protocol stacks to be managed simultaneously, not to mention greater memory requirements on equipment like routers. However, the dual stack technique does provide an effective method for organizations to begin deploying and testing IPv6 throughout their networks.

Tunneling Techniques

A variety of different tunneling techniques already exist for the purpose of interconnecting IPv6 networks using the infrastructure provided by existing IPv4 networks. For example, imagine if a company were to deploy IPv6 in two of their offices. Using various tunneling methods, the company could interconnect these offices over an IPv4-based network such as an existing WAN connection or the Internet. In order to do this, the routers providing the interconnection between the IPv6 and IPv4 networks must be dual stack devices. The advantages of using tunneling include the fact that it allows service providers to begin providing end-to-end IPv6 connections, even before they have fully converted their infrastructure to supporting IPv6.

A number of different methods for tunneling IPv6 over IPv4 networks are currently in use including manually configured tunnels, generic routing encapsulation (GRE) tunnels, automatic IPv6 to IPv4 tunnels that use techniques similar to IPv4-mapped IPv6 addresses, and more. One inherent limitation of IPv6 tunneling techniques is that they do not support common Internet connection methods like network address translation (NAT).

Translation Techniques

Similar to many of the solutions proposed to extend the life of the IPv4 address space, a variety of proposal exist for the purpose of translating IPv6 to IPv4 addresses as a method of migrating networks to IPv6. The common thread with these solutions is that rather than use the dual-stack or tunneling techniques outlined earlier, a company would instead run IPv6 internally on its network, and then use some type of translation server or gateway for continued access to the IPv4 Internet.

Although this method would break the “end-to-end” connectivity that is a central premise of IPv6, it does present an interesting solution that many companies may choose to employ as a migration strategy. One such recommendation is outlined in RFC 2766 is known as Network Address Translation – Protocol Translation (NAT-PT). A variety of IPv6 translation RFCs are currently under consideration, but the ultimate standards are still largely yet to be determined.

Coming Up Next

As far as IPv6 and the CCDA exam is concerned, that’s all she wrote! This article (along with my 3 previous articles on IPv6) should provide you with the foundation knowledge you’re expected to be familiar with for the exam. That’s not to say that you shouldn’t dig deeper – IPv6 certainly won’t be “running” the Internet next week, but it is on its way. There’s plenty more to come in the CCDA series, so stay tuned. In the meantime, best of luck with your studies!

CCNA Study Guide Chapter 5 Summary

By Dan DiNicolo, June 2nd, 2006 Posted in **CCNA Study Guide Chapter 05.**

Subscribe to our RSS Feed

Chapter 5 began with a look at IP addressing basics. This included an overview of the binary-decimal conversion process. It continued with an introduction to classful IP addressing, where we outlined how the five classes of addresses are identified, and the number of hosts that each address range supports. The number of available Class A, B, and C networks were also calculated.

An overview of IP addressing rules outlined the eight key rules with respect to host addressing, such as the rule that mandates that the host portion of an address cannot be set to all binary 0s or all binary 1s. The concept and purpose of a subnet mask was introduced, including its relationship to IP addresses in defining the split between the network and host portion of an address. Default subnet mask values and when they are used were also examined.

A look at private IP addressing provided insight into the flexibility that using these addresses provides to organizations, as well as their purpose in helping to avoid the exhaustion of the current IPv4 address space. The special subnet masks associated with the three private ranges were also examined, along with the ranges of addresses that each supports.
An overview of subnetting provided insight into reasons for deciding to subnet a network. It continued by determining the subnet and host per subnet requirements that should be determined when characterizing a network. The process for determining custom subnet masks was covered in detail, including an overview of how bits are stolen from the host portion of the mask in order to allow subnets to be defined within the address space. Calculating the number of subnets and hosts per subnet available based on a given mask value was also looked at.

Next we looked at defining the ranges of addresses that were valid on a given subnet, including the process for determining the beginning and the end of a given range. This included an overview of ranges associated with Class A, B, and C addresses in custom subnetting situations. A subnetting challenge question was used to show how even questions that appear difficult are easy to solve if the rules outlined are followed.
The subnetting shortcut provided an overview of a quick and easy method of determining both custom subnet masks and their associated address ranges. It can be used as a simple reference for exam purposes.

The process of determining whether hosts are local or remote to one another was looked at next. We learned that the ANDing process provides a way to not only determine whether hosts are local or remote, but also to find the network ID associated with a given IP address.
A look at classless addressing provided insight into how addresses are now more commonly looked at in this manner rather than by class. An overview of CIDR provided insight to supernetting, address aggregation, and slash notation.

Chapter 6: Cisco Router Externals

Introduction to Cisco Routers - Router Externals - Ethernet and Serial Ports

In Chapters 1 through 5 we mainly concentrated on what I would consider essential networking concepts. Those concepts represent a huge portion of the information that you need to be familiar with to be successful on your CCNA and CCDA exams. However, doing well on the exams will also require knowledge of Cisco equipment, commands, and configuration. It's tempting to want to just jump right in and start configuring a router. Before we do that, we first need to take a closer look at router hardware and internals, and also learn a little bit about the Cisco IOS. Topics that we'll cover in this chapter include:

- Understanding a router's ports, LEDs, connections, and cables
- Understanding router memory and storage
- Investigating IOS versions
- An overview of Cisco router models

Because the Cisco 2500 series of routers are relatively inexpensive to obtain and still widely deployed, they represent a great platform to use when studying for the CCNA exam. This chapter focuses on the Cisco 2500 series models to explain concepts, but these concepts generally apply to almost all models universally.

Router Externals

The first thing that you'll notice when you pull a Cisco 2500 series router out of the box is obviously its physical elements. A Cisco 2501 includes not only Ethernet and serial ports, but also console and auxiliary ports. In this section we'll look at the purpose of each, their physical characteristics, and how devices are attached and cabled. Figure 6-1 outlines the location of the various ports on a Cisco 2501. Note that hardware ports are numbered nominally starting at 0. Therefore, on a system with only one Ethernet port, that port is referred to as Ethernet 0. On a Cisco 2501, the leftmost serial port is referred to as Serial 0, and the rightmost as Serial 1.

Figure 6-1: Cisco 2501 rear view.

The port numbering arrangement for newer Cisco router models (well, newer than the old workhorse 2500s) is somewhat different, and you should be aware of it. For example, a Cisco

2600 router follows the convention *interface slot/port* when referring to an interface. Fixed interfaces (those not part of add-in modules) are generally referenced using slot 0. So, if you were attempting to access the first fixed Fast Ethernet port, it would be port FastEthernet 0/0. If the model had a second such port, it would be FastEthernet 0/1. By the same token, if you added a four-port serial module to your 2600 router, those ports would be known as Serial 1/0 through 1/3 respectively. On the CCNA exam, it is very important that you reference ports correctly. If you're not sure of the port numbering method used on a particular model (especially during router simulation questions) you can always use the **show interfaces** command to confirm. This command and many others will be explained in detail in Chapter 7.

Ethernet Port

A Cisco 2501 includes a single 10Mb Ethernet port. While many Cisco router models now include an integrated 10/100 RJ-45 port, the 2500 series uses what is referred to as a generic attachment unit interface (AUI) DB-15 port instead. The name for this connector (DB-15) comes from the fact that it is physically shaped like the letter 'D' and uses a 15-pin connector.

The purpose of providing an AUI port instead of a fixed RJ-45 port is flexibility - AUI ports use an external transceiver, which allow different media types to be connected, according to your network needs. Different transceivers can be attached that allow twisted pair, coaxial, or fiber connections. In that way, a transceiver connects to the DB-15 port, and then provides a port to which an RJ-45 or BNC connection could be made, for example.

But what is a transceiver? Well, its name comes from what it does - *trans*mitting and *receiv*ing data. In most NICs that you'll come across today, the transceiver is built directly into the network card. With older network cards (like those found on Ethernet 10Base5 networks) the transceiver was usually an external device.

When connecting a Cisco 2501's Ethernet connection to a common RJ-45-based network, a transceiver attaches to the AUI port, and then a patch cable connects the transceiver to a switch or hub. Remember, a router connecting to a switch or hub always uses a straight-through cable, as shown in Figure 6-2.

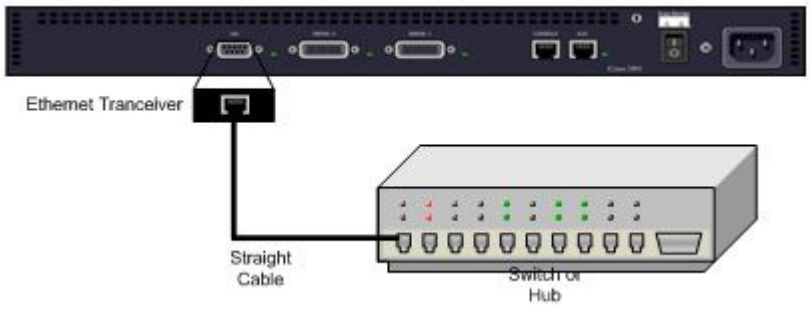


Figure 6-2: Connection between a Cisco 2501 and a switch or hub via an Ethernet transceiver.

Serial Ports

A Cisco 2501 includes two synchronous DB-60 serial ports, which are used for the purpose of WAN connections. Synchronous communication relies on the use of timing in order to control (or

synchronize) the transmission. Before synchronous systems communicate, they agree on parameters such as the time interval that will be used between sending bits of data.

A router is referred to as Data Terminal Equipment, or DTE. When DTE devices connect to a data network such as a service provider WAN link, they usually require an external device to handle their serial transmission timing, also referred to as clocking. In order to accomplish this, they connect to devices referred to as Data Communications Equipment (DCE). The function of the DCE is to provide access to the actual data network and provide clock synchronization. Two common examples of DCE equipment are modems and Channel Service Units / Data Service Units (CSU/DSU).

A CSU/DSU looks somewhat similar to an external modem, and it many ways, performs similar functions. The exception is that a modem converts digital signals to analog, while a CSU/DSU uses digital signals throughout. The DSU portion is responsible for timing, and actually connects to the DTE (in this case the router) via its serial port. The CSU portion is responsible for terminating the service provider's link, and handles transmitting and receiving data over the WAN link. Figure 6-3 shows the connection between a Cisco 2501, CSU/DSU (which is assumed to have an EIA/TIA-232 interface), and a service provider's data network.

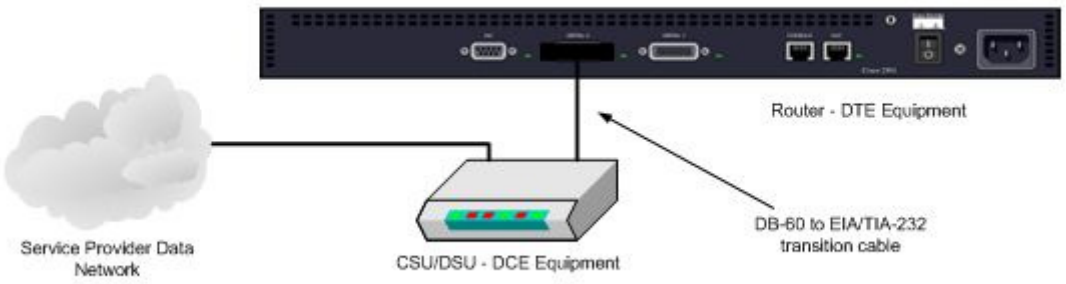


Figure 6-3: CSU/DSU connection to a serial port on a Cisco 2501.

A variety of Physical Layer standards are supported over synchronous serial interfaces to connect to different types of DCE equipment. Some of the different signaling standards and connectors that might be found on DCE equipment include EIA/TIA-232, EIA/TIA-449, V.35, X.21, and EIA-530. Cisco and a variety of other vendors manufacture "transition" cables capable of connecting a router's DB-60 DTE port to DCE equipment using these different standards. Table 6-1 outlines each standard, and for reference purposes provides information on the associated connector and connection properties.

Table 6-1: Physical layer signaling standards for connecting to DCE devices.

Signaling Standard	Connector	Properties
EIA/TIA-232	DB-25	Formerly known as RS-232, this standard supports transmissions of up to 64kbps
EIA/TIA-449	DB-37	A faster version of the EIA/TIA-232 standard, not widely adopted. Speeds up to 2Mbps.
V.35	34-pin	Speeds between 48kbps and 4 Mbps

X.21	15-pin	Commonly used in the UK to connect to the public data network. Common speed of 64kbps, up to 2Mbps
EIA-530	DB-25	Similar to EIA/TIA-449, but uses a smaller DB-25 connector. Speeds up to 4Mbps possible.

WAN protocols and connectivity will be looked at in more detail in Chapter 11.

Router Externals - Console Ports, Auxiliary Ports, Power and LEDs

Console Port

In order to perform the initial configuration of a Cisco router, you will need to create a terminal session with the router via its console port. On a Cisco 2501, the console port uses an RJ-45 connection and is an asynchronous serial port that uses the common EIA/TIA-232 communications standard. Asynchronous serial differs from synchronous in that it doesn't require synchronized clocking between the two systems. Instead, systems use what are known as start and stop bits to mark the beginning and end of a character.

While the console port may look like it should be used for a standard Ethernet connection, do not make this mistake - it is not an Ethernet port, and a straight-through or crossover cable will not provide a proper connection. Instead, connections to this port are made using what is referred to as a rollover cable. This cable is usually included when you purchase the router, but we'll look at how to create our own rollover cable shortly.

The physical connection to the console port is made via a standard PC COM port. Since the COM port on a PC is usually accessed via a DB-9 or DB-25 serial interface, a small RJ-45 to DB-9 (or DB-25) adapter is required. Again, this adapter is usually included with the router, and is labeled "TERMINAL". Once the rollover cable is connected to both the PC COM port and the console port on the router, the router can be configured using a terminal emulation program like Windows HyperTerminal. The communications settings are the same as those looked at in Chapter 3 for connecting to the console port on a Cisco 1900 switch. As a reminder, the settings remain:

- Bits per second - 9600
- Data bits - 8
- Parity - None
- Stop bits - 1
- Flow Control - None

We'll discuss the initial connection to and configuration of a Cisco 2500 router in Chapter 7.

Rollover Cable

In order to connect to the console port, you're going to require what Cisco calls a rollover cable. While these are usually provided with a router, they are also exceptionally simple to create. All that you require is a reasonable length of twisted pair cabling, some RJ-45 connectors, and the ability to remember what rollover means.

The pinouts of a rollover cable are simple to remember because they relate directly to the cable name - the ends of the cable are actually "rolled", with the pins directly opposite one another in terms of their connections. In that way, pin 1 on one end connects to pin 8 on the other, while pin 2 connects to pin 7, and so forth. There is no need to pay attention to colors when creating a rollover cable. Just ensure that the cables on one end are in reverse order on the other end. Table 6-2 outlines the complete pinouts for a rollover cable.

Table 6-2: Rollover cable pinouts.

Pins - End 1	Pins - End 2
1	8
2	7
3	6
4	5
5	4
6	3
7	2
8	1

As a best practice, make a point of identifying the various cables you create to avoid confusion. Use different colored cables if possible or at least consider adding letters such as "X" (for crossover) or "R" (for rollover) to the cable ends using a permanent marker.

Auxiliary Port

The purpose of the auxiliary port is for connecting to an external modem. Once configured, this modem can be used as a backup demand-dial connection to another location, or as a way to dial in to the router for troubleshooting purposes should regular connectivity fail. Unlike the console port, the auxiliary port supports hardware flow control, which ensures that the receiving device receives all data before the sending device transmits more. In cases where the receiving device's buffers become full, it can pass a message to the sender asking it to temporarily suspend transmission. This makes the auxiliary port capable of handling the higher transmission speeds of a modem.

Much like the console port, the auxiliary port is also an asynchronous serial port with an RJ-45 interface. Similarly, a rollover cable is also used for connections, using a DB-25 adapter that connects to the modem. Typically, this adapter is labeled "MODEM".

LEDs and Power

On any given Cisco router, the LED lights provide quick insight into whether the system or ports are functioning correctly. On a Cisco 2500 series router, each port (with the exception of the console and auxiliary ports) has an associated LED that indicates port activity, in a fashion similar to what you might expect to see on a switch or hub. Another LED, to the right of the auxiliary port, is referred to as the "System OK" LED. When this LED is lit, it indicates that the router is functioning correctly. Figure 6-4 outlines the location of the LEDs on a Cisco 2501.

Figure 6-4: Cisco 2501 LEDs.

In a normal working state, port LEDs will flicker to indicate activity. In cases where the port LED is not active, it suggests either a connectivity issue, or potentially a problem with the port.

Obviously a router needs to be powered. In cases where the router needs to be opened in order to change or upgrade components, not only should it be powered down using the power switch, but the Alternating Current (AC) cable should also be disconnected. Cisco routers are usually also available in Direct Current (DC) versions, since many telecommunications carriers require DC-powered devices. In cases where DC devices need to be powered down, the power switch should be turned off, and then power should be disconnected at the circuit breaker.

Router Internals: Getting to Know RAM, ROM, NVRAM, and Flash Memory

The internal components of the router are where the real magic takes place. Think of a Cisco router are being really no more than a specialized computer running a custom operating system. In this case, it is a computer optimized to provide routing and related functions. Instead of relying on a hard disk for storage, a Cisco router relies on different types of memory, each with a different purpose. In this section we'll take a look at each of these different storage areas, and the functions they are responsible for.

There are four main memory areas within a Cisco router that you'll need to be familiar with - Flash, RAM, ROM, and NVRAM.

Flash Memory

Flash memory is implemented on a Cisco 2500 using two Single Inline Memory Module (SIMM) slots that hold erasable programmable read-only memory (EPROM). Flash memory is used to store and run the Cisco IOS software. When a system is powered down, the contents of Flash memory are not lost. However, its contents can be upgraded by "flashing" the chip, a concept that we'll look at in more detail in Chapter 12. While a router is running, the contents of Flash are set to a read-only mode.

Flash memory for a Cisco 2500 series router ranges in size from a minimum of 4MB up to a maximum of 16MB. You might consider adding additional Flash memory to meet the space requirements of the IOS version that you have chosen to run. For a Cisco 2501, the base IP version of IOS 12.0 requires a minimum of 8MB of Flash memory. If your Cisco 2501 shipped

with only 4MB of Flash, you would require at least one additional 4MB SIMM. For IOS versions with more advanced feature sets, it is not uncommon to require at least 16MB of Flash.

When installing or upgrading Flash using multiple SIMMs, it is important to note that they must be the same size. For example, if you already have 4MB of Flash and wish to upgrade, you can either replace the 4MB SIMM with an 8MB SIMM, or simply add a second 4MB SIMM. You cannot mix and match SIMMs with different storage capabilities. As such, you cannot have one 4MB and one 8MB SIMM installed at the same time - their storage capabilities must equal.

RAM

Random Access Memory (RAM) represents the non-permanent or volatile working area of memory on a Cisco router. When the router is powered down, the contents of RAM are lost.

By default, RAM is broken up into two main areas - Main Processor Memory, and Shared I/O Memory. Main Processor Memory is where the routing table, ARP tables, and running configuration are stored. Shared I/O Memory is used as a buffer location for temporarily storing packets prior to processing. Most Cisco 2500 routers will have 2MB of RAM soldered to the system board (this amount, however, depends on the revision number of the router), along with one SIMM slot to add additional RAM. The maximum amount of RAM that can be added to a Cisco 2500 is 16MB. If 16MB is added, that provides a maximum of 18MB of available RAM. In cases where a RAM SIMM is installed, its capacity will be used as Main Processor Memory, while the onboard RAM (2MB) will be used as Shared I/O memory. If no SIMM chip is present, that 2MB of on-board RAM will be split between both areas, providing each with 1MB of working space. This should be avoided for performance reasons.

ROM

In older Cisco router models, Read-Only Memory (ROM) chips were used to store the IOS software. In newer models, this is no longer the case. As mentioned previously, the IOS image is now stored in Flash memory (it can also be stored on a TFTP server, as I'll discuss in the next chapter). ROM is now used as the memory area from which a Cisco router begins the boot process, and is made up of a number of elements. These elements are implemented via microcode, a set of programming instructions that are contained in ROM.

- **Power-on Self Test (POST)**. When the router is powered up, microcode stored in ROM performs a POST sequence. This is used to ensure that elements such as the CPU, memory, and interfaces are capable of functioning correctly.
- **Bootstrap Program**. The bootstrap program is used to initialize the CPU and boot functions of the router. The bootstrap program is responsible for locating and loading the router's IOS.
- **ROM Monitor**. A special diagnostic environment used for the purpose of troubleshooting or special configuration. For example, this mode can be used to transfer an IOS image over a console connection.
- **RxBoot**. When a valid IOS image cannot be found in Flash or on a TFTP server, this limited IOS version is loaded for the purpose of installing a new IOS image into Flash. It is also sometimes referred to as the boot loader, boot image, or helper image. The command set provided is only a subset of normal IOS commands.

On Cisco 2500 series routers, ROM is 2MB in size. In cases where ROM needs to be upgraded (which is rare), the actual chips needs to be replaced on the router's motherboard. When a router is powered down, the contents of ROM are always retained.

NVRAM

Non-Volatile Random Access Memory (NVRAM) is used as the storage location for the router's startup configuration file. After the router loads its IOS image, the settings found in the startup configuration are applied. When changes are made to a router's running configuration, they should always be saved to the startup configuration (stored in NVRAM) or they will be lost when the router shuts down. Remember that the running configuration is stored in RAM, which is erased when the router is powered down. On a Cisco 2500 series router, NVRAM is a relatively tiny 32KB in size.

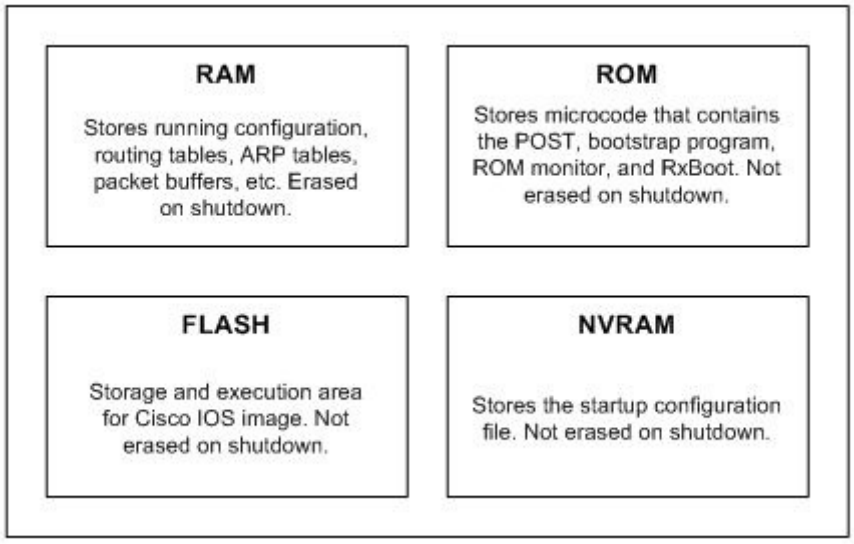


Figure 6-5: Cisco router memory elements and purposes.

Cisco IOS Versions

Before we get into the configuration of a Cisco router in Chapter 7, we should first understand a little more about the Cisco IOS. This is an area that many people find confusing, especially when just starting out. Between the various IOS versions and release codes, it's easy to understand why.

Note: You are not explicitly required to be familiar with the specifics of IOS versions for either the CCNA or CCDA exams. This information is provided for practical purposes only, and to give you some insight into how IOS images are named.

The Cisco IOS is specialized operating system software that was originally designed for Cisco's routers, but is now also used on much of their switching equipment. The key thing to understand is that Cisco IOS software is specific to the particular hardware platform that you are using. For example, while a release such as IOS version 12.0 will work across many platforms, the specific images provided for the Cisco 2500 series are not for use on 2600 series routers.

Cisco breaks its IOS software releases into groupings referred to as "feature sets". Many different feature sets will be available for a given hardware platform, and allow different capabilities

depending upon the requirements of a given environment. For example, the basic IP feature set image does not include IPSec encryption capabilities, but the IP Plus IPSec 56 feature set image does. In any network environment, required features will vary according to the protocols, security requirements, and functionality needed. When purchasing or upgrading the IOS for a router, it is important to be sure that you have chosen a feature set image that meets the needs of your environment. Of course, the IOS feature set can always be changed (at a cost) if your requirements change.

Cisco releases three major types of IOS versions - major releases, early deployment (ED) releases, and general deployment (GD) releases.

A major release is developed to ensure high quality and stable software for customer networks. Once issued, no new features are ever added to a major release - changes are limited to bug fixes, which are implemented in the form of maintenance updates. For example, you may come across IOS software version 12.1(3). In this case, the number 12.1 identifies the major release. The bracketed 3 identifies the fact that it is the third maintenance release. When any new bug fixes are applied and integrated into a release, a new maintenance number will be associated with it. In this case, the next would be 12.1(4). Maintenance releases are intended to add to a major release's stability. They tend to be issued approximately every 8 weeks in the early portion of the major release's lifecycle.

In order to satisfy customers who require access to new or emerging features and functions, Cisco also releases what are referred to as early deployment (ED) releases. These releases include all of the functionality of the associated major release, but also include additional features (and potentially bug fixes for those features). ED releases always end with a capitalized letter - in most cases, the letter "T". 12.1(3)T represents major release 12.1, maintenance release 3, and an ED release. In any given maintenance release, new features may also be added. A variety of trailing capitalized letters can refer to an ED release. For example, releases ending in "X" represent a one-time-only release.

The final status that Cisco applies to an IOS version is general deployment. While a major release will be deployed to customers along with updated maintenance versions, a product is not considered to be certified for General Deployment (GD) until it has been qualified through extensive exposure to customer networks, analyzed by Cisco engineering groups, and customer feedback has been assessed. A cross-functional team at Cisco decides when the IOS version receives GD status. In the 12.1 IOS major release, everything from maintenance release 13 forward was granted GD status. Simply, GD status means that the stability of the release is considered proven by Cisco and validated by customer experience.

Tip:

In practicing for your CCNA or CCDA exam, you should generally try to work from a router using a Cisco 12.x IOS version.

Cisco Router Models

For the purpose of the CCDA exam, you are expected to be familiar with the various models in Cisco's line of popular routers. While you're not expected to be aware of every feature of every router, it is important for you to have a solid foundation understanding of the various model series' and their basic capabilities. Since Cisco router models can and do change, it is important for you to stay up to date on this information. For that reason, you should make a point of reading through the information posted on the Cisco website rather than relying solely on the information found here. It's quite possible that Cisco will release a new line of routers or new model numbers,

and then integrate questions that reference those particular models into the CCDA exam. At the time this book was written, any of the routers up to the 7600 series appear to be fair game for the CCDA exam.

Note: For more information on current Cisco router models, as well as the models listed here, visit http://www.cisco.com/en/US/products/hw/routers/index.html

SOHO Series

Cisco's SOHO series routers are aimed at small and home office environments using Digital Subscriber Line (DSL) connections. The ports on this model are fixed, meaning that module slots are not provided. Its NAT capabilities allow multiple users to share a single connection. Models in the SOHO 70 and SOHO 90 lines include different combinations of Ethernet and DSL ports, primarily to meet the needs of home and small offices utilizing broadband Internet connections like ADSL and cable. Features found in these models include built-in firewall capabilities, as well as management via a standard web browser.

Cisco 700 Series

The Cisco 700 series routers are aimed at providing always-on ISDN Internet access to small and home offices with up to 5 users. Some models in the series provide integrated 4-port hubs, as well as analog telephone ports to allow fax or modem lines to share the ISDN Basic Rate Interface (BRI) connection. No modular slots are provided.

Cisco 800 and 1000 Series

The Cisco 800 and 1000 series routers are aimed at small offices and telecommuters. Models in these series provide single or multiple 10BaseT Ethernet ports, along with the different types of WAN connections including ISDN, DSL, serial (up to 512Kbps), analog dial-up, and even an additional Ethernet WAN port to support external cable or DSL modems. All ports on the 800 and 1000 series routers are fixed.

Cisco 1600 Series

Cisco 1600 series routers are aimed at providing flexible connectivity solutions for smaller branch offices and small businesses. All 1600 models include at least one 10BaseT Ethernet port and one fixed WAN interface (ISDN, or serial). All models also support the ability to add a single modular WAN interface card (WIC), which can add additional ports for X.25, ISDN, Frame Relay, T1/E1 connections, and more. Other options include the ability to add an integrated CSU/DSU as well as firewall and VPN software. The module cards used in the 1600 series are capable of being used in many of the other routers looked at in this section, as outlined shortly.

Cisco 1700 Series

Cisco 1700 series routers are aimed at small and medium-sized business and small branch office environments, proving a wide range of routing, firewall, and VPN functions. Both the 1720 and 1751 models provide a single 10/100 Ethernet port. While the 1720 provides 2 modular WAN interface slots, the 1751 provides 2 modular WAN slots and one voice slot. This allows models like the 1751 to be used for voice, fax, and data applications simultaneously. The newer Cisco

1760 model is a standard 19-inch rack-mounted unit that includes four modular slots. WAN interface cards that can be added include serial, ISDN BRI, DSL, X.25. The WAN interface cards used into the Cisco 1700 series can also be used in the Cisco 1600, 2600, 3600, and 3700 series routers, extending their flexibility and helping to protect equipment investments.

Cisco 2500 Series

One of the most popular routers in Cisco product line, the Cisco 2500 series is aimed at branch office applications. The 2500 series comes in a variety of different port configurations, and can include fixed Ethernet, serial, asynchronous, Token Ring, and ISDN ports. Models in the Access Server line (2509 and 2511) include 8 or 16 asynchronous ports to allow the aggregation of multiple analog or digital telephone lines in environments where higher speed service such as T1 may not be available. These interfaces can also be used to provide dial-in access to external users.

Cisco has also released modular versions in the 2500 series, such as the 2524 model - it provides 3 modular WAN interface slots, allowing a mix of ISDN BRI, serial, and integrated CSU/DSU modules to be added. Table 6-3 outlines some of the more popular models in the Cisco 2500 series.

Tip: If you're planning on purchasing a Cisco 2500 series router on an online auction site, be sure to pay attention to the model number. For example, Cisco 2502 and 2504 models do not include Ethernet ports, which may impact your ability to connect them to your test network.

Table 6-3: Popular Cisco 2500 router models.

Model	LAN/WAN Ports
2501	1 Ethernet, 2 Serial
2502	1 Token Ring, 2 Serial
2503	1 Ethernet, 2 Serial, 1 ISDN BRI
2504	1 Token Ring, 2 Serial, 1 ISDN BRI
2505	1 Ethernet, 2 Serial, and 8-port integrated hub
AS2509	1 Ethernet, 2 Serial, 8 Asynchronous Serial
AS2511	1 Ethernet, 2 Serial, 16 Asynchronous Serial
2524	1 Ethernet, up to 3 WAN modules.

Note: Cisco recently announced that most of the models in its 2500 series have reached EOS (end-of-sale) status and are no longer orderable. This currently does not apply to the various access server (AS) models in the line.

Cisco 2600 Series

The Cisco 2600 series of routers are considered to be the logical replacements for companies that might have relied upon models in the 2500 series in the past. These models are primarily positioned as multiservice branch office routers capable of supporting a variety of applications including analog and digital dialup access services, voice and video integration, VPN access, routing between VLANs, firewall security, and more. The Cisco 2600 series supports one of the widest ranges of modules in the router product line, with over 70 different modules and interfaces available to provide everything from integrated Layer 2 switch ports up to ATM connectivity. As mentioned previously, these modules can also be used in the 1600, 1700, 3600, and 3700 product lines.

Popular models in the 2600 series include the 2621XM and the 2691XM. The 2621XM provides either one or two 10/100 Fast Ethernet ports, along with one network module and two WIC slots. The 2691XM provides similar connectivity options, but provides a total of three WIC slots and faster packet forwarding rates.

Cisco 3600 Series

The Cisco 3600 series is in many ways similar to the 2600 series, with the notable exception that models in this line provide higher performance and port densities. The 3600 series of routers are primarily aimed at larger branch office environments.

Two of the most popular models in this line are the Cisco 3620 and 3660. The 3620 provides two network module slots that support a range of different LAN and WAN technologies including Fast Ethernet, Token Ring, serial, and ISDN interfaces to name but a few. The 3660 further expands the possible port densities available by providing six network module slots, 2 advanced integration module (AIM) slots, and redundant power supplies.

Cisco 3700 Series

The Cisco 3700 series is a key component of Cisco's Architecture for Voice, Video and Data (AVVID). Models in the 3700 series are aimed at enterprise branch offices looking for integrated LAN and WAN connectivity options with a wide range of supported add-on modules. The two main models in this line are the 3725 and 3745. The 3725 provides two integrated Fast Ethernet ports, two AIM slots, three WIC slots, two network module slots, and one high-density service module slot. The 3745 model is similar, with the exception that it provides four network module slots and two high-density service module slots.

One way in which Cisco positions models in the 3700 series is as a branch office telephony gateway and VoIP solution platform. For example, both models support 16- or 36-port EtherSwitch modules that provide Layer 2 switching and support inline power, which is used to power IP phones over an Ethernet connection.

Cisco 7500 Series

Cisco 7500 series of multiprotocol routers are aimed at large enterprise environments as well as service provider networks. These modular routers utilize a very different architectural model that the other models looked at here, relying on both Route Switch Processors (RSPs) and Versatile Interface Processors (VIPs) to carry out routing and packet switching functions. On a Cisco 7500 model, the RSP represents the main system processor that looks after tasks like sending and receiving routing protocol updates, managing routing tables, monitoring the status of various interfaces, and so forth. A VIP, on the other hand, is a processor dedicated to packet switching tasks associated with one or more interfaces. Because a VIP has dedicated processing capabilities, it is capable of making multi-layer switching and routing decisions based on information provided by the RSM. Furthermore, VIP cards are also capable of handling tasks like queuing, encryption, traffic analysis, and more. In many ways, with its own processor, memory, and software that is basically a subset of the IOS, a VIP card is very similar to a dedicated multilayer switching device. However, VIPs still ultimately take direction from RSPs. This distributed system architecture ultimately results in much higher system performance overall. Ultimately, Cisco considers their 7500 series routers to be the premier solution for companies looking for high-end voice, video, and data routing performance.

The three main models in the Cisco 7500 line include the 7505, 7507, and 7513. The 7505 supports one RSP and four VIP slots, while the 7507 provides two RSP and five VIP slots. The ultra-flexible 7513 also supports two RSPs, but permits up to 11 VIPs to be added.

CCNA Study Guide Chapter 6 Summary

By Dan DiNicolo, June 7th, 2006 Posted in **CCNA Study Guide Chapter 06.** Subscribe to our

RSS Feed

Chapter 6 began with a look at the external features of a Cisco 2500 series router. This included an overview of each port, including its numbering, physical connectors, and characteristics. An overview of the router's Ethernet port provided insight into AUI connections, transceivers, and their purpose.

A closer look a router's serial ports included an explanation of synchronous communications, DTE and DCE devices, and the physical layer communications standards that are used to connect between these devices. A basic introduction to CSU/DSUs provided insight into its role in the WAN communication process.

The purpose of both the asynchronous console and auxiliary ports was looked at next, including the ways in which these ports are connected to devices using rollover cables and adapters. The pinouts for a rollover cable were also looked at, in order to allow you to create your own as necessary. A review of the router's LED lights provided insight into the types of information that these provide.

A look at a router's internal elements focused on the four main memory storage areas used on Cisco routers, including Flash, RAM, ROM, and NVRAM. The information contained within each was examined, as were their associated characteristics.

An overview of the Cisco IOS provided insight into features sets and the different types of IOS releases provided by Cisco. The ability to distinguish a major release from an early deployment release was discussed, as were the purposes of maintenance and general deployment releases.

Chapter 7: IOS Commands and Configuration

IOS Commands and Configuration: Introduction and the Cisco Router Boot Process

In Chapter 6 we learned about router interfaces, hardware, memory and the IOS. In this chapter, it's finally time to get down to the initial configuration of a Cisco router. Although you aren't really expected to be familiar with router configuration issues for the CCDA exam, you should still understand the majority of the concepts covered in this chapter. On the CCNA exam, you will not only need to remember commands and their purpose, but will also need to be able to issue them in questions where you will be working from a simulated router command line environment.

The topics that we'll cover in this chapter include:

- The router boot process
- Using the System Configuration Dialog to configure a router
- Understanding the command-line environment and prompts
- Setting router passwords
- Configuring router banners
- Configuring router interfaces
- Configuring host names and name resolution
- Using telnet to access and configure routers
- Backing up and restoring the Cisco IOS and configuration files
- Using the Cisco Discovery Protocol (CDP) to gain information about neighboring devices

At the end of the chapter you will use many of the commands learned to walk through the configuration of a router from start to finish, using a set of hypothetical requirements. More advanced topics, including the configuration of routing, access lists, WAN protocols, and IOS troubleshooting will all be looked at in upcoming chapters.

Cisco Router Boot Process

The process that a Cisco router goes through on boot is not terribly unlike that of any computer. A familiarity with the process will help you to better understand how a router's various configuration elements relate to one another, and will ultimately be required knowledge in troubleshooting situations.

To begin, we'll assume that we're powering up a Cisco 2501 for the first time. At this point, the router should include a valid IOS image in Flash memory, but the router should not yet be configured. The sequence of events that a router goes through during the boot process includes:

1. **POST**. When first powered up, a router will carry out a power-on self-test (POST). Recall that the POST is used to check whether the CPU and router interfaces are capable of functioning correctly.

2. **Execute bootstrap to load IOS**. After a successful POST, the router will execute the Bootstrap program from ROM. The bootstrap is used to search Flash memory for a valid Cisco IOS image. If one is present, the image is loaded. If an image cannot be found, the router will boot the RxBoot limited IOS version found in ROM.

3. **IOS loads configuration file**. Once the IOS image is loaded, it will search for a valid startup configuration in NVRAM. If a valid startup configuration file cannot be found, the router will load the System Configuration Dialog, or what is sometimes called setup mode. This mode allows you to perform the initial configuration of the router.

   If these three steps listed seem a little too simple, not to worry - they are the default steps of what I consider to be a "proper" or "good" boot. In Chapter 13 we'll take a closer look at troubleshooting the boot process when things go wrong.

Configuring a Cisco Router Using the System Configuration Dialog

After the IOS loads, it looks for a valid startup configuration file in NVRAM. In cases where this file is not found, the router will enter what is known as the System Configuration Dialog. This environment is almost like a wizard, in that it will prompt you with questions relating to the configuration of the router. It can be used to configure basic settings only, or more advanced parameters, depending on your requirements. For the most part, the configuration that can be accomplished from this environment is somewhat limited, and not nearly as extensive as what can be done directly from the command line.

To begin our configuration, we'll need to connect a rollover cable between the console port on the router and the COM port on a PC. Use a program such as Windows HyperTerminal (or your preferred terminal emulation program) to create a terminal session with the router, as outlined in Chapter 6. Once the router boots, you'll notice a range of messages that relate to the bootstrap program, the router's IOS version, interfaces, memory, and more. When the messages are done, you are presented with the System Configuration Dialog Utility, as shown below.

```
--- System Configuration Dialog ---
Continue with configuration dialog? [yes/no]: y
At any point you may enter a question mark '?' for help.
Use ctrl-c to abort configuration dialog at any prompt.
Default settings are in square brackets '[]'.
Basic management setup configures only enough connectivity
for management of the system, extended setup will ask you
to configure each interface on the system
Would you like to enter basic management setup? [yes/no]:
```

Notice the structure of the questions. You are first asked if you are interested in continuing with the initial configuration dialog. I chose "yes" by typing the letter "y" (you could also fully type "yes") and then pressing Enter. Doing so provides you with information about obtaining help, aborting the configuration (pressing CTRL+C), and default settings.

The second paragraph outlines the two different modes that can be used to configure the router from this environment – basic management setup and extended setup. Basic management setup will provide you with a fairly limited set of choices, such as setting a router's hostname, passwords, and an interface IP address. The idea is that after doing this, you would move on to the command line environment to configure more advanced features. Once you're familiar with IOS commands, you will probably prefer using the command line for configuration.

If you want to get at more advanced configuration elements from the System Configuration Dialog, you would choose "no" when asked whether you would like to enter the basic management setup. This brings you into the extended setup. Extended setup gives you access to a wider range of configuration choices, including configuring protocols, interfaces, the asynchronous port (if you have a modem attached), passwords, and so forth. For the purpose of our look at the System Configuration Dialog, we'll use the extended setup.

```
Would you like to enter basic management setup? [yes/no]: n
First, would you like to see the current interface summary? [yes]: y
Interface    IP-Address    OK?  Method Status                Protocol
Ethernet0    unassigned    NO   unset  administratively down  down
Serial0      unassigned    NO   unset  administratively down  down
Serial1      unassigned    NO   unset  administratively down  down
```

Notice that when asked if I wanted to see an interface summary, I entered "y" for yes. Because the default value shown in the square brackets is already [yes], I could have simply pressed Enter to accept that value. At this point, the interface summary reminds us that no settings have really been configured. We'll get to that, as we work through the rest of the extended setup.

The next section allows us to configure what are known as global parameters. The first option involves giving the router a hostname, followed by the configuration of various passwords.

```
Configuring global parameters:
Enter host name [Router]: toronto-1
The enable secret is a password used to protect access to
privileged EXEC and configuration modes. This password, after
entered, becomes encrypted in the configuration.
Enter enable secret: cisco
The enable password is used when you do not specify an
enable secret password, with some older software versions, and
some boot images.
Enter enable password: cisco2
The virtual terminal password is used to protect
access to the router over a network interface.
Enter virtual terminal password: cisco3
```

The first prompt asked for a hostname. I chose toronto-1, which will ultimately be shown at the router's command prompt, as we'll see shortly. This is followed by the configuration of passwords. The first password required is what is known as the "enable secret". This password will ultimately be encrypted, such that it cannot be read in the router's configuration file as plain text. Enable passwords are used to access what is known as privileged EXEC mode, where router configuration takes place. We'll take a look at privileged EXEC mode in detail later in this chapter.

The second password you are asked to provide is what is known as the “enable password”. This is different from the enable secret, in that it is not encrypted. The enable password is used with older IOS versions, as well as older boot images found in ROM. When both an enable secret password and an enable password are configured, only the enable secret password is used. Technically these cannot be the same – if you enter the same value for the enable password as you did for the enable secret, you will be prompted to change it. The truth of the matter is that if you enter the same password again, it will be accepted. I would suggest choosing different values, if only for the sake of not having the “real” enable password appear in the configuration file in plain text.

The final entry from the System Configuration Dialog shown above asked for a virtual terminal password. This password will be required to access the router via a telnet session. Again, my suggestion is to choose a different password than your enable secret password. Passwords will be looked at in more detail once we get to the command line.

The extended setup continues by asking whether you want to configure a variety of routing or routed protocols on the system. For the purpose of this illustration, I’m going to stick to configuring IP only.

```
Configure SNMP Network Management? [no]: n
Configure DECnet? [no]:
Configure AppleTalk? [no]:
Configure IPX? [no]: n
Configure IP? [yes]: yes
Configure IGRP routing? [yes]: no
Configure RIP routing? [no]:
Configure bridging? [no]:
```

The next step in the process depends on your router model. If it had an ISDN BRI interface (such as on the 2503, for example) you would be prompted to configure its properties. Since there are so many possibilities based on different hardware models, we’ll jump straight to the interface configuration section.

```
Configuring interface parameters:
Do you want to configure Ethernet0  interface? [yes]: y
Configure IP on this interface? [yes]: y
IP address for this interface: 192.168.1.46
Subnet mask for this interface: 255.255.255.0
Class C network is 192.168.1.0, 24 subnet bits; mask is /24
Do you want to configure Serial0  interface? [no]: y
Configure IP on this interface? [no]: y
Configure IP unnumbered on this interface? [no]: n
IP address for this interface: 192.168.2.1
Subnet mask for this interface [255.255.255.0] : 255.255.255.0
Class C network is 192.168.2.0, 24 subnet bits; mask is /24
Do you want to configure Serial1  interface? [no]: n
```

The extended setup just walked us through a very simple method of configuring IP addresses and subnet masks for the interfaces of our Cisco 2501. I also chose not to configure the Serial1 interface – we’ll leave that for the command line. After choosing not to configure that last interface, we are immediately presented with the router’s brand new running configuration, based on the settings entered during the setup process.

```
The following configuration command script was created:
hostname toronto-1
enable secret 5 $1$x73p$iIK9q8cRLza30wwITfd28.
enable password cisco2
```

```
line vty 0 4
password cisco3
no snmp-server
!
no decnet routing
no appletalk routing
no ipx routing
ip routing
no bridge 1
!
interface Ethernet0
ip address 192.168.1.46 255.255.255.0
no mop enabled
!
interface Serial0
no shutdown
ip address 192.168.2.1 255.255.255.0
no mop enabled
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
interface Serial1
shutdown
no ip address
dialer-list 1 protocol ip permit
dialer-list 1 protocol ipx permit
!
end
[0] Go to the IOS command prompt without saving this config.
[1] Return back to the setup without saving this config.
[2] Save this configuration to nvram and exit.
Enter your selection [2]: 2
```

The last choice to be made involves deciding whether or not we wish to save the configuration that we've entered. Remember that it can always be changed later. Choosing option 2 here will save the running configuration that was just displayed to NVRAM, ultimately making it the router's startup configuration. We'll look at the details of configuration files later in the chapter. For now, notice the interface configurations as well as the encrypted enable secret password shown above.

Introduction to the IOS Command Line Interface and Working with Configuration Files

Do not fear the command line! While you may initially feel uncomfortable working without a graphical interface, I promise that the command line really isn't any more difficult. In fact, easy-to-use help is available every step of the way. As long as you can remember a few simple commands and techniques, you'll never be at a loss for how to configure a Cisco router.

Logging In and Logging Out

For the time being, we're going to continue to access the router via a console connection. We'll get into the details of connecting via a telnet session a little later in the chapter. After connecting, you'll be presented with the message below.

```
toronto-1 con0 is now available
Press RETURN to get started!
```

The message makes us aware that we are connected to the console port, also known as con0. After pressing Enter, we'll be in what is known as user EXEC mode. You can always identify the mode you are in by the prompt you are presented with. In this case, the prompt appears as:

```
toronto-1>
```

Notice that the prompt displays the hostname that we configured when walking through the extended setup. In this case, it ends with a > sign, which designates that we're in user EXEC mode (or just "user mode" for short). Your capabilities in user mode are fairly limited, allowing you to view information such as statistics, issue pings, show system hardware and software status, and so forth. In order to get at the configuration of the router, we'll need to be in what is known as privileged EXEC mode (also known as "privileged mode"). To access privileged mode, you need to issue the **enable** command.

```
toronto-1>enable
Password:
toronto-1#
```

Because we set the enable secret password to cisco, this is the password that we enter to access privileged mode. The password does not appear on the screen while you are typing, nor are characters designated with asterisks – this is for security purposes. Notice how the prompt has changed. Instead of the > character, the privileged mode prompt is designation by the # sign.

Exam Tip: Don't forget that the > prompt signifies user EXEC mode, while the # prompt signifies privileged EXEC mode.

Once you have finished configuring your router, you will want to exit privileged mode. Doing this is quite intuitive; the command you need to enter is simply **disable**.

```
toronto-1#disable
toronto-1>
```

Notice that issuing the **disable** command returns us to user mode, as shown by the > prompt. In order to log out of the router completely, you have the choice of issuing either the **logout** or **exit** command. These commands can also be issued directly from privileged mode, allowing you to log out of the router in a single step.

```
toronto-1>logout
toronto-1 con0 is now available
Press RETURN to get started!
```

Using the **logout** or **exit** command brings us right back to where we started.

Configuration Files

In order to get a sense for what is really happening when you configure a router, you will need to understand the difference between a router's running configuration and its startup configuration. In Chapter 6 we looked at the contents of RAM and NVRAM. A router's running configuration is stored in RAM. When you make changes to the configuration of a router, this is almost always what you are changing. However, remember that the contents of RAM are lost when a router is powered down. As such, if you want to save any changes that you've made, you will need to copy them to NVRAM, where the startup configuration is stored. The startup configuration that is saved in NVRAM is the one that will be applied to the router if it is rebooted.

Before getting into the details of how to save a router's configuration, we should first know how to view both the startup and running configurations. Both can be accessed from privileged EXEC mode using the show command. A variety of different configuration elements can be viewed using the show command, many of which we'll look at as the chapter progresses.

To view the current running configuration of a router, use the **show running-config** command. To view the startup configuration, use the **show startup-config** command.

```
toronto-1#show running-config
Building configuration...
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service tcp-small-servers
!
hostname toronto-1
!
enable secret 5 $1$RATF$Rr5XuMrPDNBHSqnvJLVwl/
enable password cisco2
!
ip subnet-zero
!
!
process-max-time 200
!
interface Ethernet0
 ip address 192.168.1.46 255.255.255.0
toronto-1#show startup-config
Using 739 out of 32762 bytes
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
service tcp-small-servers
!
hostname toronto-1
!
enable secret 5 $1$RATF$Rr5XuMrPDNBHSqnvJLVwl/
enable password cisco2
!
ip subnet-zero
!
!
process-max-time 200
!
interface Ethernet0
ip address 192.168.1.46 255.255.255.0
```

Since we haven't really configured any major settings yet, I cut the output of both commands to save space. You may have noticed that the output from both commands looks fairly similar. That's because we haven't made any configuration changes yet. If we had, the running configuration would include settings not yet saved to the startup configuration.

In order to save the running configuration to the startup configuration, use the **copy running-config startup-config** command.

```
toronto-1#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
toronto-1#
```

Notice the format of the command – it is telling the router to copy the settings found in the running configuration to the startup configuration. In other words, the format of the copy command is:

copy *from to*

But why is this important to remember? Because if you issue the command in the reverse order – copy startup-config running-config – you are actually telling the router to overwrite the current running configuration with the settings found in the startup configuration! Take time to think before issuing these commands, and be sure to remember that you are always specifying the source first, followed by the destination. We'll look at the configuration files in much more detail later in the chapter. For now, just remember that in order to save configuration changes, you'll need to copy them to the startup configuration stored in NVRAM.

IOS Shortcuts and the Terminal History Command

A number of shortcuts exist within the IOS command line interface. We've already seen that we could simply enter **y** or **n** to represent answering yes or no from within the System Configuration Dialog setup modes.

Before getting into the various shorthand versions of commands that can be used in the IOS, let's look at the editing commands available. Editing commands (sometimes referred to as "hot keys") allow you to quickly navigate the command line. For example, these can be used to move the cursor forward or back a word, move the cursor to the beginning or end of a line, and so forth. There are a number of these editing commands available, and to remember them, I suggest creating associations between the shortcuts and what they do. For example, pressing **CTRL+A** will move your cursor to the beginning of a line. Associate the command with the fact that the letter A begins the alphabet, and it's easier to remember. In the same way, **CTRL+E** will bring you to the end of a line – just associate E with "end". Table 7-1 outlines the primary editing commands available on a Cisco router. You'll need to be familiar with what each key or combination does, so be sure to practice using these! Of course, they'll also come in very handy when trying to configure a router quickly in real life.

Table 7-1: Cisco router editing commands.

Command	What It Does
Delete	Removes the character to the right of the cursor

Backspace	Removes the character to the left of the cursor
Up Arrow	Allows you to scroll forward through previous commands
Down Arrow	Allows you to scroll backwards through previous commands
Ctrl+P (or up arrow)	Displays the last command entered
Ctrl+N (or down arrow)	Displays previous commands entered
Ctrl+A	Moves the cursor to the beginning of the current line
Ctrl+E	Moves the cursor to the end of the current line
Ctrl+F	Moves forward one character
Ctrl+B	Moves backwards one character
Esc+F	Moves forward one word
Esc+B	Moves backwards one word
Ctrl+R	Redisplays a line (starts a new line, with the same command shown)
Ctrl+U	Erases a line
Ctrl+W	Erases a word
Tab	Completes a partial command
Ctrl+Z	Exits configuration mode, returning you to privileged EXEC mode

One very helpful shortcut listed in Table 1 is the Tab key – instead of having to type out a complete command, you can instead type just enough of it such that the command is not ambiguous. For example, if you were to type **sh** and press the Tab key, you would be presented with the completed command **show**. However, typing **s** alone and pressing tab won't do the same. Many commands start with the letter s, and the router wouldn't be able to determine which command you were referring to. A good way to get used to entering partial commands is to hit the tab key after entering the first few letters at the command line – this will give you a feel for how much of the command needs to be entered in order for it not to be considered ambiguous.

Similarly, commands can be issued in shorthand. For example, if you grow tired of entering the complete command **enable** to enter privileged mode, you could simply type **en** and press enter, as shown below.

```
toronto-1>en
Password:
toronto-1#
```

However, notice what happens when you only type **e** and press enter in the same scenario:

```
toronto-1>e
% Ambiguous command:  "e"
toronto-1>
```

My suggestion is that you first familiarize yourself with the complete commands we look at. Next, focus on using the Tab key to complete them. Finally, figure out the shorthand versions that can be used to represent them.

Terminal History

The commands for moving back and forth on the command line are pretty self-explanatory. However, some of the commands listed in Table 1 are especially useful. For example, the up and down arrows allow you to scroll through previous commands, much like you might be familiar with from the Windows or Linux command line.

If you do choose to scroll through commands regularly, you may want to control how many commands are stored in the router's command history. Conversely, you might feel that the command history feature represents a security threat – another user could easily view recently issued commands if you were to step away from the router momentarily without logging off. In that case, you can also turn the command history off.

Configuring terminal history settings can be accomplished from both user mode and privileged mode. By default, the command history feature is enabled, and is set to show the last 10 commands entered. To verify this, use the **show terminal** command. I've eliminated most of the command's output in the example below to show only the data relevant to terminal history settings.

```
toronto-1>show terminal
History is enabled, history size is 10.
```

In order to actually view the command history, use the **show history** command:

```
toronto-1>show history
show int e0
show int s1
show int s0
show history
help
show history
show sessions
sh ver
sh term
sh history
toronto-1>
```

You can also set the command history to a larger or smaller size. To set the command history to buffer the last 20 commands issued, use the **terminal history size** command.

```
toronto-1>term his size 20
```

Notice that in the example above, I entered shorthand versions of the first two parts of the command. Instead of entering **terminal history size 20**, I simply truncated it to a shorter version that still provided enough information to not be considered ambiguous by the router.

Finally, if you want to turn the terminal history feature off, enter the command:

```
toronto-1>terminal no history
```

Terminal history can easily be re-enabled by issuing the command **terminal history**.

Using the Cisco IOS Command Line Help Feature

The most wonderful thing about Cisco's IOS command line interface is that help is available every step of the way. There are a few tricks to using the help system effectively. The most basic element that you'll need to be familiar with is the help command itself – the question mark.

From the command line, you can always get information on available commands by entering **?**. This command is sensitive to where you happen to be in the environment – for example, the commands that are available to you from user mode are different than those available in privileged mode, which in turn are different from those available when configuring an interface. We'll look at configuration modes shortly. For now, we'll start off by getting a sense of how to use the help command effectively.

Notice that when we press **?** from privileged mode, we're presented with a list of available commands.

```
toronto-1#?
Exec commands:
<1-99>          Session number to resume
access-enable    Create a temporary Access-List entry
access-profile   Apply user-profile to interface
access-template  Create a temporary Access-List entry
bfe             For manual emergency modes setting
cd              Change current directory
clear           Reset functions
clock           Manage the system clock
configure       Enter configuration mode
connect         Open a terminal connection
copy            Copy from one file to another
debug           Debugging functions (see also 'undebug')
delete          Delete a file
dir             List files on a filesystem
disable         Turn off privileged commands
disconnect      Disconnect an existing network connection
enable          Turn on privileged commands
erase           Erase a filesystem
exit            Exit from the EXEC
help            Description of the interactive help system
lock            Lock the terminal
login           Log in as a particular user
--More--
```

The list displayed only includes commands up to **login** – this is a function of my screen display size. Can you see the **--More--** entry at the bottom of the list? You can display the remaining commands, one screen at a time, by pressing the spacebar. You can also view additional commands one at a time by pressing the Enter key. Each command also provides a short explanation of what it is used for.

Let's say that we decide to enter the **copy** command. After pressing Enter, the output tells us that the command is incomplete. In other words, it needs additional information in order to do anything useful.

```
toronto-1#copy
% Incomplete command.
toronto-1#
```

To find out what we should enter next, we can again use the help command. In this case, enter **copy ?** (be sure to leave a space between copy and the question mark). There is no need to press Enter – after typing **?**, the router will provide output immediately.

```
toronto-1#copy ?
/erase          Erase destination file system.
flash:          Copy from flash: file system
flh:            Copy from flh: file system
ftp:            Copy from ftp: file system
null:           Copy from null: file system
nvram:          Copy from nvram: file system
rcp:            Copy from rcp: file system
running-config  Copy from current system configuration
startup-config  Copy from startup configuration
system:         Copy from system: file system
tftp:           Copy from tftp: file system
```

The help function presents us with a list of additional commands that can be used with **copy**. To determine what should come next, just add one of the additional commands, leave a space, and again enter the question mark.

```
toronto-1#copy flash: ?
flash:          Copy to flash: file system
ftp:            Copy to ftp: file system
lex:            Copy to lex: file system
null:           Copy to null: file system
nvram:          Copy to nvram: file system
rcp:            Copy to rcp: file system
running-config  Update (merge with) current system configuration
startup-config  Copy to startup configuration
system:         Copy to system: file system
tftp:           Copy to tftp: file system
```

Using the help function one last time yields the output shown below.

```
toronto-1#copy flash: tftp: ?
<cr>
```

The **<cr>** output represents a carriage return, and means that the next step is simply hitting Enter to execute the command. Do not bother hitting enter this time – we haven't learned anything about copying to a TFTP server yet, so you're better off just erasing the line using a shortcut just recently learned. In case you forgot, try **Ctrl+U**.

Help can actually be used in different ways. We've already seen that leaving a space after a command and then entering the question mark provides us with a list of the possible commands that are expected next. However, the question mark can also be used as a type of lookup character. For example, entering a letter or series of letters directly followed by the question mark (no space) provides a list of commands that start with those characters. Consider the output from entering **c?**.

```
toronto-1#c?
cd    clear  clock  configure  connect
copy
```

This provides a list of all of the commands starting with the letter C that are available from the current prompt. Going a step further, you could enter the first two (or more) letters directly followed by the question mark. In this case, let's enter **cl?**.

```
toronto-1#cl?
clear  clock
```

This time, the only commands listed are those starting with the letters CL. Using the help command will also give you an idea of how truncated a command can be without being considered ambiguous. For example, it should be obvious that entering only **cl** would not be enough to use as a shortcut for clock, since the router won't know whether we had meant to enter **clock** or **clear**. Add one additional letter, however, and we have a unique entry.

```
toronto-1#clo
% Incomplete command.
```

Notice what just happened – the router recognizes that I'm entering the **clock** command, but tells me that it's incomplete. Stepping back to what you just learned, you could find out more about the additional commands required by the clock command by entering **clo ?**.

```
toronto-1#clo ?
set  Set the time and date
```

Don't be bashful. Whether on an exam or in real life, there is never anything wrong with using the help commands. If you're ever in doubt, just remember that the help feature is there for a purpose – nobody can remember everything!

Working in the Cisco IOS Global Configuration Mode

One thing that you'll definitely need to be familiar with when attempting to configure a Cisco router is the different configuration modes available. We've already discussed the basic idea behind user mode and privileged mode. However, when it comes down to actually configuring almost all settings, you will need to be in what is known as global configuration mode. From this mode, you will also be able to access the configuration of individual interfaces, and eventually routing protocols as well. In this section we'll take a look at accessing the different configuration levels and some associated commands.

Global Configuration Mode

In order to make any configuration changes of substance to a Cisco router, you'll need to access global configuration mode. Three main global configuration modes are available, and allow you to change the configuration stored in different locations. Noting the differences between the three is extremely important. The three main modes include:

- **Configure Terminal**. This mode is by far the most common one you'll interact with. **Configure terminal** allows you to make changes to the running configuration of the router, which you should recall is stored in RAM. These settings are the ones currently being used by the router.
- **Configure Memory**. This mode provides you with access to changing the startup configuration of a router, which is stored in NVRAM. If you choose this mode, the startup configuration file will be loaded into RAM, where it can then be changed. It is generally a good idea to back up the current running configuration to a TFTP server prior to issuing this command – we'll look at how that is accomplished later in the chapter.

- **Configure Network**. This mode allows you to change a configuration file that is stored on a TFTP server. Again, this file would first be loaded into RAM, then allowing you to make changes. Similarly, a backup of the current running configuration should first be performed.

For the most part, you will be making changes to a router's running configuration, so you'll probably be using the configure terminal or config t option. To access global configuration mode, simply enter **configure terminal** from the privileged mode command line.

```
toronto-1#configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#
```

Notice that the prompt has changed – the new prompt specifies that we are in global configuration mode. You should also pay attention to the instructions – the section that says End with CNTL/Z actually means that to exit global configuration mode, you should press **Ctrl+Z**. This command was also covered in the IOS shortcuts we looked at earlier.

While entering the complete command configure terminal isn't too tough, you can also access global configuration mode using shorthand. In this case, try **con t**.

```
toronto-1#con t
% Ambiguous command:  "con t"
```

Again we end up with the ambiguous command message. But why? To find out, enter **con?** at the prompt:

```
toronto-1#con?
configure  connect
```

Notice that both the commands **configure** and **connect** start with the letters CON. Let's try going a step further, adding another letter:

```
toronto-1#conf t

Enter configuration commands, one per line.  End with CNTL/Z.

toronto-1(config)#
```

It worked! The command **conf** is not ambiguous, nor is using the letter T to represent terminal. Just to confirm, take a look at the options available for **conf ?**.

```
toronto-1#conf ?
memory            Configure from NV memory
network           Configure from a TFTP network host
overwrite-network  Overwrite NV memory from TFTP network host
terminal          Configure from the terminal
<cr>
```

Don't worry about the **overwrite-network** option. It allows you to overwrite the startup configuration file stored in NVRAM with one stored on a TFTP server. Looking at the end of the output above, notice that one of the options is a carriage return **<cr>**. If you type **conf** and then press Enter, you will be asked where you want to configure from, with **terminal** being the default option. Pressing Enter again will automatically choose the **terminal** option.

```
toronto-1#conf
Configuring from terminal, memory, or network [terminal]?
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#
```

To return back to privileged mode, press **Ctrl+Z**. This will produce the **^Z** characters shown. Ctrl commands do not require you to press Enter.

```
toronto-1(config)#^Z
toronto-1#
```

Interface Configuration Mode and Line Configuration Mode

Configuring router interfaces involves accessing their own specific configuration environments. Each interface is configured individually, according to its name and numeric identifier. To begin, you need to enter global configuration mode. Then, you'll need to access a specific interface using the **interface** command. The output below demonstrates entering global configuration mode, and then attempting to determine the command to access our router's Ethernet interface.

```
toronto-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#interface ?
Async             Async interface
BVI               Bridge-Group Virtual Interface
Dialer            Dialer interface
Ethernet          IEEE 802.3
Group-Async       Async Group interface
Lex               Lex interface
Loopback          Loopback interface
Null              Null interface
Port-channel      Ethernet Channel of interfaces
Serial            Serial
Tunnel            Tunnel interface
Virtual-Template  Virtual Template interface
Virtual-TokenRing  Virtual TokenRing
```

Notice the option for Ethernet. To continue, enter **interface ethernet ?**.

```
toronto-1(config)#interface ethernet ?
<0-0>  Ethernet interface number
```

The command requires that we specify the number of the Ethernet interface that we wish to configure. Remember that some routers will have more than one Ethernet interface. Our Cisco 2501 has only one Ethernet interface, nominally numbered 0. If we issue the command **interface ethernet 0** and press Enter, notice how the command prompt again changes.

```
toronto-1(config)#interface ethernet 0
toronto-1(config-if)#
```

The prompt now shows **toronto-1(config-if)#**. This designates that we are in interface configuration mode, although you should note that the prompt doesn't actually identify the

interface – that part you'll need to remember on your own. Accessing a serial interface would have been just as simple – the command would be **interface serial**, followed by the nominal interface number.

You can also use shortcuts to access interfaces. For example, we could have accessed interface Ethernet 0 using the command **int e0**.

```
toronto-1(config)#int e0
toronto-1(config-if)#
```

The shorthand version is a great deal quicker – I'm sure that's ultimately what you will end up using.

Depending upon the router model, accessing interfaces can be slightly different. Some of the routers that we looked at in the Chapter 6 are modular. In these cases, interfaces are numbered according to both their slot and interface number. For example, on a Cisco 2600 router, the first Fast Ethernet interface would be known as FastEthernet0/0. The numbering designates the interface's slot/port number, and would be accessed as shown below.

```
Router(config)#int fastethernet 0/0
```

To exit the configuration of an interface and return to global configuration mode, use the **exit** command. You can also exit global configuration mode completely by pressing **Ctrl+Z**.

```
toronto-1(config-if)#exit
toronto-1(config)#
```

We'll explore the actual configuration of router interfaces later in the chapter. For now, we'll continue with our look at the different configuration modes.

Line Configuration

Interface configuration does not include the console or auxiliary ports. These interfaces, along with settings related to telnet sessions, are configured using the line command. Again accessed via global configuration mode, these ports and settings are also configured using their names, though with a slightly different syntax.

- **Console**. The configuration of a router's console port is accessed using the command **line console 0**. Accessing the line configuration for the console port allows you to configure settings such as a password.
- **Auxiliary**. The auxiliary port's configuration is accessed using the command **line auxiliary 0**.
- **Virtual Terminals**. Telnet ports are also known as virtual terminals. Different routers will allow varying numbers of simultaneous telnet sessions, depending upon their IOS version. Telnet connections occur over an existing hardware port, such as a correctly configured Ethernet or serial interface. However, properties relating to telnet sessions are set by configuring virtual or vty ports. The command to configure virtual terminal ports is **line vty**, followed by the number (or numbers) of the ports that you wish to configure.

To access a line configuration, you must already be in global configuration mode. In the example below, we are accessing the line configuration of the console port.

```
toronto-1#conf t
```

```
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#line console 0
toronto-1(config-line)#
```

Notice the prompt has changed to **toronto-1(config-line)#**. Again, it's up to you to remember which line you just accessed. We'll look at the actual configuration of line properties (and specifically setting passwords) shortly.

Routing Protocol Configuration Mode

The last configuration mode that you will eventually come across is the one associated with configuring routing protocols. We'll look at routing protocols and their configuration in detail in Chapter 8. For now, it's enough to simply be familiar with the prompt associated with configuring routing protocols.

```
toronto-1#conf t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#router rip
toronto-1(config-router)#
```

We have now seen six different router prompts, four of which fall within global configuration mode. Table 7-2 reviews each of the prompts we've seen so far, and the mode or level that it designates.

Table 7-2: Cisco IOS prompts and modes.

Router Prompt	Purpose / Meaning
Router>	User EXEC mode
Router#	Privileged EXEC mode
Router(config)#	Global configuration mode
Router(config-if)#	Interface level of global configuration mode
Router(config-line)#	Line level of global configuration mode
Router(config-router)#	Routing protocol level of global configuration mode

Exam Tip: For the purpose of your CCNA exam, it's important to recognize the differences between the various prompts, as well as which commands can be entered at each.

Configuring Cisco Router Passwords

At the beginning of this chapter we configured our initial passwords using the System Configuration Dialog. In both real-life and on the exams, however, you will need to know how to configure passwords from the command line. Remember that by default, a router will usually have no passwords associated with it (some models do ship with default factory passwords, usually **cisco**), so this is something that you'll definitely want to change. There are 5 main passwords associated with a Cisco router. These include:

- **Enable password**. The enable password is used to restrict access to privileged EXEC mode on a Cisco router. Recall that enable passwords are not encrypted, meaning that they can be read in plain text via the configuration files from privileged EXEC mode. The enable password was used by older IOS versions, but has been superceded by the enable secret password, which is encrypted.
- **Enable secret password**. The enable secret password also provides access to privileged EXEC mode on a Cisco router, but is stored in encrypted form using the Message Digest 5 (MD5) algorithm. On any Cisco router beyond IOS version 10.3, the enable secret password should always be used. In fact, you should probably ignore the enable password completely in favor of enable secret password. Again, when both are configured, only the enable secret password can be used to access privileged mode.
- **Console password**. A console password is used to restrict access to a router's physical console port. If a password is not associated with the console port, anyone can walk up to the router, plug in a rollover cable and create a session, gaining access to at least user EXEC mode.
- **Auxiliary password**. Much like the console port, a password can also be used to restrict access to the auxiliary port, which may be configured to allow access via an external modem. Whether you're using it or not, it's always a good idea to set a password on this port.
- **Telnet password**. As mentioned earlier, a Cisco router allows telnet sessions via what it considers to be virtual terminals. On a Cisco router running Standard Edition IOS software, a maximum of 5 virtual terminals are provided, named vty 0 through 4. On Enterprise Edition IOS versions, the number of possible virtual terminals is much higher, depending upon the version and platform.

Although the enable secret password is the only one encrypted by default, any of the passwords above can be encrypted as required. We'll explore this after we learn how to assign passwords to interfaces.

Exam Tip: Remember that when both an enable and enable secret password are configured, the enable secret password always takes precedence.

Assigning Enable Passwords

The first important step in configuring your Cisco router is setting a password to control access to privileged mode. Without one, your router's configuration is fair game to anyone with a rollover cable and only a tiny bit of know-how. Recall that the enable secret password always takes precedence over the unencrypted and less secure enable password. In fact, Cisco recommends not using the **enable password** command at all.

All Cisco passwords are configured from global configuration mode, although the console, auxiliary, and vty ports are configured at the line level. To set the enable password on your Cisco router, simply issue the **enable password** command.

```
toronto-1(config)#enable password cisco99
```

Obviously, this will set the enable password to cisco99. To set the enable secret password, use the **enable secret** command:

```
toronto-1(config)#enable secret cisco100
```

Assigning Console, Auxiliary, and Virtual Terminal Passwords

A console password is configured from global configuration mode, at the console line level. The output below outlines each step from privileged EXEC mode forward.

```
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#line console 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco1
```

Notice the series of commands above. First, global configuration mode was accessed, and followed by entering the line console 0 level. The command **login** specifies that we are requiring users to be authenticated to access this port. If we later changed our mind, we could remove the requirement by using the **no login** command. Finally, the password was set using the **password** command along with the password itself – in this case, cisco1. When setting passwords, they appear on the screen in plain text. During the login process, they are not visible.

Setting an auxiliary password follows the same steps, with the exception that the auxiliary line must be accessed.

```
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#line aux 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco1
```

In this case, I went back to shorthand and used **aux 0** instead of typing out **auxiliary 0**. You should make a point of using shorthand where possible to save time and avoid typing errors.

Configuring virtual terminal passwords for telnet sessions works a little differently. If you do not set vty passwords on the router, you will not be able to make a telnet connection to it – this is obviously a security feature. In order to set telnet passwords, you will need to follow the configuration listed below. In this example, we are going to configure all 5 virtual terminals to use the same password. Note the syntax used.

```
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#line vty 0 4
toronto-1(config-line)#login
toronto-1(config-line)#password cisco1
```

Pay particular attention to the third line of output above. The line level accessed was for all five virtual terminal lines, numbered 0 through 4. By accessing the line level for all 5 simultaneously, we have made 5 telnet sessions possible using the password cisco1. To allow only a single telnet session, the command would be **line vty 0**. The 4 other sessions would then not be accessible.

Encrypting Passwords

Normally, passwords other than enable secret will appear in our configuration files in plain text. Even though you need to be in privileged mode to view the configuration files, encrypting all

passwords is still a good idea. Eventually we'll back up the configuration files to a network server, which means that other people may have the ability to access and view them.

The command used to manually encrypt passwords is **service password-encryption**. You can encrypt any password manually by first issuing this command from global configuration mode, and then changing passwords as you normally would. Once complete, enter the **no service password-encryption** command. In the example below, we have encrypted both the auxiliary and console port passwords.

```
toronto-1(config)#service password-encryption
toronto-1(config)#line con 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco1
toronto-1(config-line)#line aux 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco1
toronto-1(config-line)#exit
toronto-1(config)#no service password-encryption
```

After completing the steps listed, you can view the encrypted versions of the passwords by using the **show running-config** command. I have again truncated the output to show only the pertinent information.

```
toronto-1#show run
Building configuration...
Current configuration:
line con 0
password 7 01100F17580457
login
transport input none
line aux 0
password 7 03075218050070
login
transport input all
line vty 0 4
password cisco1
login
!
end
```

Notice that both the console and auxiliary passwords have been encrypted. The vty password has not, since we didn't specify it while configuring the encrypted passwords. Viewing the startup configuration at this point would still show unencrypted versions of all these passwords. Why? Because we haven't saved our changes to the startup-configuration, of course!

```
toronto-1#sh star
Using 790 out of 32762 bytes
line con 0
password cisco1
login
transport input none
line aux 0
password cisco1
login
transport input all
line vty 0 4
password cisco1
login
```

```
!
end
```

Remember that in order to save our changes, we need to save the running configuration to the startup configuration. In simple to remember shorthand, you can simply enter the command **copy run star**.

Tip: You may have noticed that the number "7" precedes a password encrypted by the **service password encryption** command. These passwords are encrypted using a weaker algorithm than **enable secret**, and can easily be cracked with a variety of utilities or Perl scripts available online, assuming a person has access to the encrypted string from the configuration file.

Configuring Router Banner Messages

If you have used Windows or other operating systems in a corporate environment, you are likely familiar with the concept of a login message or banner. This is a message presented to users, usually before they attempt to log in. The main reason for these banners is to provide users with information, perhaps a warning message that makes them aware of security restrictions on this particular equipment. While this message won't do anything to actually stop them from attempting to log in, it does help to cover things from a legal perspective. Many operating systems and applications provide users with a "welcome" prompt. Believe it or not, there are precedents where hackers have claimed that they were "in the right" for hacking into private systems – the message did welcome them, after all.

Regardless of how ridiculous you may think this is, it's still a good idea to set up banner messages on your systems. At the very least, you are making people who attempt to connect aware that unauthorized users should not be accessing the system.

A variety of different banners can be created on a Cisco router, but the most popular is the "message of the day" or MOTD banner. Users will be presented with this banner every time they attempt a connection via the console port, auxiliary port, or a telnet session. MOTD banners are configured from global configuration mode. In the example below, we're going to configure our router to display a simple message that states that only authorized users are allowed access.

```
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#banner motd ?
LINE  c banner-text c, where 'c' is a delimiting character
toronto-1(config)#banner motd #Authorized Access Only!#
```

The command used to configure a MOTD banner is **banner motd**. Notice that we followed the command with a question mark, in order to determine how to complete it. In this case, the output tells us that we need to start and end the banner message with a "delimiting character". A delimiting character is one that cannot be used within the actual message. For example, you might choose to use the **$** or **#** signs, with the same character appearing at the beginning and end of the message. Logging out and attempting to access the router again presents us with the MOTD banner.

```
Authorized Access Only!
User Access Verification
Password:
```

Tip: Remember that an MOTD banner must begin and end with the same delimiting character, and that this character cannot be used within the actual banner message.>

Configuring Router Interfaces

In order to get our router to do anything truly useful, we'll need to configure its interfaces. Recall that on a Cisco 2501, we have one Ethernet interface and two serial interfaces. We'll start off by configuring these interfaces with IP addresses, and follow up by configuring IPX addresses.

Ethernet and serial interfaces are configured from the interface level of global configuration mode. In order to configure interface Ethernet 0, we'll need to access that interface.

```
cisco2501#config t
Enter configuration commands, one per line.  End with CNTL/Z.
cisco2501(config)#int e0
```

Our next step will be assigning interface Ethernet 0 an IP address. This couldn't be easier – we'll simply use the **ip address** command.

```
cisco2501(config-if)#ip address 192.168.1.46
% Incomplete command.
```

After entering the **ip address** command along with the address we want to configure, we're faced with the incomplete command message. Obviously that means that we've forgotten something. Use the question mark to figure out what that something is.

```
cisco2501(config-if)#ip address 192.168.1.46 ?
A.B.C.D  IP subnet mask
```

It's now clear that we're supposed to also add a subnet mask value following the IP address. In this case, we'll use a mask of 255.255.255.0.

```
cisco2501(config-if)#ip address 192.168.1.46 255.255.255.0
```

The address is now set, but even if you tried to ping it from another system, you wouldn't receive a reply. Why not? Because even though it has been configured, the interface is still not "turned on". In order to make the interface active, you will need to enter the **no shutdown** command.

```
cisco2501(config-if)#no shutdown
```

Assuming that you have a straight cable connecting the Ethernet 0 interface to a hub or switch, you should now be able to ping it from other systems.

Tip: As a best practice, issue the **show interface** command after configuring an interface to be sure that it is functioning correctly.

Configuring our router's serial interfaces is really no different – access the serial interface that you wish to configure, and allocate an IP address and subnet mask using the same command. Remember that the **no shutdown** command will also need to be issued. The output below shows the configuration of IP address 192.168.2.1 on interface serial 0.

```
cisco2501(config)#int serial 0
```

```
cisco2501(config-if)#ip address 192.168.2.1 255.255.255.0
cisco2501(config-if)#no shutdown
```

A variety of other properties can be configured for interfaces beyond their addresses and shutdown properties. To get a listing on the commands available, use the question mark from the **router(config-if)#** prompt.

It's usually a good idea to also add a description to interfaces. For example, we could add a description to the serial 1 interface, mentioning that it connects the Toronto router to our Montreal location. This is also accomplished from the interface level of global configuration mode.

```
cisco2501(config)#int s1
cisco2501(config-if)#description WAN link Toronto to Montreal
cisco2501(config-if)#^Z
cisco2501#show run
Building configuration...
Current configuration:
interface Serial1
description WAN link Toronto to Montreal
ip address 192.168.2.1 255.255.255.0
```

I once again truncated the output from the **show run** command, limiting it to the pertinent information about the serial 0 interface. Notice the description is now included in the interface section. This provides a quick and easy way to reference what a given interface connects to.

Configuring Serial Interfaces in Lab Environments

In a real-life network, your serial interfaces will almost certainly be configured as DTE interfaces. Recall that a CSU/DSU usually handles the clocking for a synchronous serial interface. If you're working in your own lab, however, you may be connecting the serial ports of two routers directly using what is known as a DCE-to-DTE crossover cable. These cables allow you to simulate a serial WAN connection without requiring a CSU/DSU or similar device. This is shown in figure 7-1.

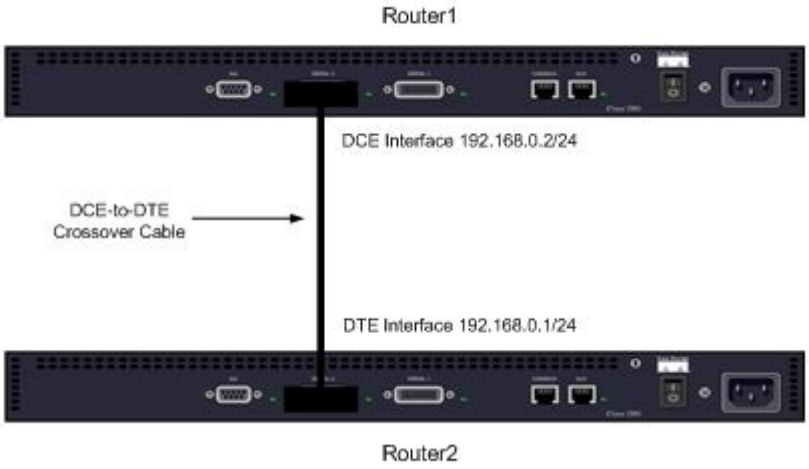


Figure 7-1: Connecting the serial ports of two routers using a DCE to DTE crossover cable.

The main issue with connecting your serial interfaces in this manner is the fact that one of the devices will need to be configured as DCE in order to provide the timing mechanism required. The DCE-to-DTE crossover cable will have two different DB-60 interfaces – one marked DTE, and the other marked DCE. The router connected to the DCE end of the cable will need its serial interface configured as a DCE device.

For the sake of argument, we're going to assume that we are simulating a WAN link between two routers, Router1 and Router2. Their serial interfaces have already been configured as per the IP address settings shown in Figure 1. On the DCE device, we need to configure the serial interface using the **clock rate** command.

```
Router2(config-if)#clock rate 56000
%Error: This command applies only to DCE interfaces
```

Notice what just happened. I accidentally attempted to configure the clock rate on Router2 (the DTE device) instead of Router1. Router2 makes me aware of my mistake by letting me know that I have attempted to configure the wrong device – it recognizes its cable connection as DTE. Attempting the configuration again on Router1 yields the following results.

```
Router1(config-if)#clock rate 56000
Router1(config-if)#
```

Because we didn't get an error message, the command completed successfully. Notice that we set the clock rate to 56000, a value represented in bits per second. A variety of clock rates are possible, and can be viewed by entering the question mark after the **clock rate** command.

If you're looking for a quick way to check whether a serial interface is connected to the DCE or DTE end of a crossover cable, use the **show controllers** command. The first few lines of output are shown below.

```
Router2#show controllers serial 0
HD unit 0, idb = 0x710CC0, driver structure at 0x716140
buffer size 1524  HD unit 0, V.35 DTE cable
```

Interface Bandwidth

One other interface configuration command that you'll need to be familiar with is the **bandwidth** command. The **bandwidth** command is used to specify the configured bandwidth of an interface, such as serial 0. It's worth noting that this command doesn't change the actual bandwidth of a physical interface, but instead communicates the bandwidth available on an interface to upper-layer protocols. For example, serial interfaces have a default bandwidth of 1.544 Mbps on a Cisco router. However, if the interface has only a 64 Kbps circuit attached, you would issue the **bandwidth** command on the interface specifying this. This number would then be used by certain routing protocols (such as IGRP) that use bandwidth as one of their metrics in determining the best route to a destination. While IGRP will be looked at in more detail in Chapter 8, for now it's important to simply know how to configure the bandwidth value for a serial interface. The **bandwidth** command is always followed by a value expressed in kilobits per second (Kbps). This is important to remember, since it is easy to confuse the **bandwidth** and **clock rate** commands.

```
Router1(config)#int s1
Router1(config-if)#bandwidth 64
```

In the example above, the bandwidth on the interface S1 has been set to 64 Kbps.

Tip: Remember that the command **clock rate** is two words, expressed in bits per second; the **bandwidth** command is a single word, expressed in kilobits per second.

Configuring IPX and Using the Show Interface Command

Configuring IPX addresses and related settings isn't really any more difficult than IP, although there are a few extra steps involved.

The first step in configuring a router to use IPX involves enabling IPX routing globally. This is done from global configuration mode, using the **ipx routing** command.

```
cisco2501#config t
Enter configuration commands, one per line.  End with CNTL/Z.
cisco2501(config)#ipx routing
```

The next step is configuring an interface with an IPX network number. Remember that IPX addresses are 80 bits in length. The first 32 bits represent the network, and the last 48 bits are assigned automatically as per the interface MAC address. Because of this, we actually don't need to specify the complete address – only the network number, which will ultimately have the interface MAC address appended to it. In this case, we'll configure interface Ethernet 0 to be part of IPX network 101A. Recall that IPX addresses are listed in hexadecimal, and that any leading 0s can be left off the address.

```
cisco2501(config)#int ethernet 0
cisco2501(config-if)#ipx network 101A
```

The interface is now configured to run IPX. However, you may recall from Chapter 4 that IPX can be configured to use different incompatible frame types. Table 7-3 outlines the four IPX frame types used in Ethernet environments. The default is **novell-ether** (the Ethernet 802.3 frame type), unless you specify otherwise.

Table 7-3: Novell Ethernet frame types and their Cisco IOS encapsulation names.

Novell Ethernet Frame Type	Cisco Encapsulation Name
Ethernet 802.3	novell-ether
Ethernet 802.2	sap
Ethernet II	arpa
Ethernet SNAP	snap

In order to set the frame type for an interface, add the **encapsulation** command when configuring the interface network number, as shown below.

```
cisco2501(config-if)#ipx network 101A encapsulation sap
```

In this case, we set the interface encapsulation to Ethernet 802.2, or **sap**. An IPX interface can actually be configured for multiple frame types. However, different frame types use different

network numbers, since they cannot communicate with one another. To add an additional IPX frame type to an interface, use the **ipx network** command, followed by the **secondary** keyword.

```
cisco2501(config-if)#ipx network 101B encapsulation arpa secondary
```

In this case, we configured interface Ethernet 0 to use **arpa** (Ethernet II) encapsulation for another IPX network, 101B.

Tip: Don't forget that the default encapsulation on Ethernet interfaces configured for IPX is 803.3 (**novell-ether**).

Show Interface

In order to truly understand your interfaces, you'll need to make use of the **show interface** command. By specifying an interface along with the command, you'll be provided with statistical information, the interface's MAC address, logical addresses, encapsulation, and most importantly, whether the interface is functioning correctly or not.

The **show interface** command is issued from the privileged EXEC mode prompt. In order to view the configuration of our router's Ethernet 0 interface, use the **show interface e0** command.

```
Router#show interface e0
Ethernet0 is up, line protocol is up
Hardware is Lance, address is 00e0.f751.d6af (bia 00e0.f751.d6af)
Internet address is 192.168.1.46/24
MTU 1500 bytes, BW 10000 Kbit, DLY 1000 usec, rely 255/255, load 1/255
Encapsulation ARPA, loopback not set, keepalive set (10 sec)
ARP type: ARPA, ARP Timeout 04:00:00
Last input 00:00:00, output 00:00:00, output hang never
Last clearing of "show interface" counters never
Queueing strategy: fifo
Output queue 0/40, 0 drops; input queue 0/75, 0 drops
5 minute input rate 2000 bits/sec, 3 packets/sec
5 minute output rate 1000 bits/sec, 2 packets/sec
15248 packets input, 1718569 bytes, 0 no buffer
Received 5405 broadcasts, 0 runts, 0 giants, 0 throttles
0 input errors, 0 CRC, 0 frame, 0 overrun, 0 ignored, 0 abort
0 input packets with dribble condition detected
12122 packets output, 4045434 bytes, 0 underruns
0 output errors, 0 collisions, 4 interface resets
0 babbles, 0 late collision, 1 deferred
0 lost carrier, 0 no carrier
0 output buffer failures, 0 output buffers swapped out
```

It's important to be familiar with the information provided by the show commands. When looking at the output of the **show interface e0** command, a variety of information is provided, including the interface IP address, MAC address, statistics, and so forth. However, the first line contains information that is absolutely critical – it tells us about both the physical and data link characteristics of the interface.

The first portion, **Ethernet 0 is up**, refers to the Physical layer characteristics of the interface. In this case, it means that the link is receiving a carrier detect signal – in other words, our Ethernet interface is plugged into a working switch or hub. If the message had been **Ethernet 0 is down**, that would generally indicate a physical connectivity problem, such as the cable being disconnected.

The second portion, **line protocol is up**, refers to the Data Link layer characteristics of the interface. Keepalives are the messages sent over a link to ensure that a given interface is usable. If keepalives are functioning correctly, the line protocol is considered to be up.

There are four possible entries that you may come across when using the show interface command. Each is outlined below. Although this example uses an Ethernet interface, it could just as well be the output for a serial interface.

- **Ethernet0 is up, line protocol is up**. Both the Physical and Data Link characteristics of the interface are functioning correctly.
- **Ethernet0 is down, line protocol is down**. This message usually represents a physical interface problem. For example, the cable may be disconnected. This problem can also occur if this interface is connected to another router whose interface has been shut down using the shutdown command.
- **Ethernet0 is up, line protocol is down**. In this case, Physical layer connectivity is obviously not the issue. The line protocol being down is usually related to either a clocking issue (such as with keepalives) or a mismatch between the frame types being used on connected devices. For example, one router being configured to use ARPA frames, and another to use SNAP.
- **Ethernet0 is administratively down, line protocol is down**. This output means that a local interface has been manually shut down using the shutdown command. In the example below, the shutdown command is issued for interface serial 0, followed by the **show int s0** command.

```
Router(config)#int s0
Router(config-if)#shutdown
Router(config-if)#^Z
Router#show int s0
Serial0 is administratively down, line protocol is down
```

Tip: The show interface command should always be one of the first commands issued when attempting to troubleshoot interface connectivity or communication issues. Be sure that you feel very comfortable understanding the interface status information provided by the first line of this command's output.

Configuring Hostname Resolution Settings on a Cisco Router

During the initial System Configuration Dialog, we gave our router a hostname of toronto-1. This can be changed using the **hostname** command. For example, to change the hostname of this router to cisco2501, enter the following from global configuration mode:

```
toronto-1(config)#hostname cisco2501
cisco2501(config)#
```

Notice that the command prompt name immediately changes to **cisco2501**. The hostname associated with the router is there to give you perspective on which router you are connected to. Unless you have an entry set up on a DNS server that maps this name to one of the router's IP addresses (or appropriate host file entries), you still won't be able to telnet into the router using its hostname.

By default, a Cisco router will always assume that any unrecognized command is the name of a host that you wish to initiate a telnet session with. Because of this, it will attempt to resolve the name to an IP address using DNS. For example, consider what happens when I enter “helpme” at the prompt and press enter.

```
cisco2501#helpme
Translating "helpme"...domain server (255.255.255.255)
% Unknown command or computer name, or unable to find computer address
cisco2501#
```

If you want to avoid this frustrating and somewhat annoying action, you can always configure the router to not perform a DNS lookup on unrecognized commands using the **no ip domain-lookup** command.

```
cisco2501(config)#no ip domain-lookup
cisco2501(config)#^Z
cisco2501#helpme
Translating "helpme"
% Unknown command or computer name, or unable to find computer address
cisco2501#
```

You will probably get to a point where you’ll want to configure a router to resolve names, since these are generally easier to remember (and to input) than IP addresses. If you decide to do this, you have two choices – you can either configure your router to use DNS, or you can use a locally configured hosts table. If you’re familiar with the HOSTS file from UNIX or Windows environments, this is almost exactly the same – a group of static name-to-IP address entries that you manually define.

To configure a router to use a local hosts table, you will need to be in global configuration mode. In the example below, I have created entries for 2 different routers, named accra and montreal, using the **ip host** command.

```
cisco2501(config)#ip host accra 192.168.1.45
cisco2501(config)#ip host montreal 192.168.36.2
cisco2501(config)#^Z
cisco2501#show hosts
Default domain is not set
Name/address lookup uses static mappings
Host                 Flags      Age Type   Address(es)
accra                (perm, OK)  0   IP    192.168.1.45
montreal               (perm, OK)  0   IP    192.168.36.2
```

The **show hosts** command is used to view the hosts table. The table shows us that the entries are permanent, along with hostnames and associated IP addresses. To be honest, creating a hosts table on each an every router would be painful – you are much better off using DNS if it’s available.

Configuring a router to use a DNS server to resolve hostnames isn’t much more difficult. Just remember that entries for the hosts and their associated IP addresses need to be entered in DNS prior to the router being able to resolve them. There are a couple of steps involved in setting up a router to query DNS. As a first step, we need to reinstate the **ip domain lookup** command that we turned off earlier.

```
cisco2501(config)#ip domain-lookup
cisco2501(config)#ip name-server 192.168.1.100
cisco2501(config)#ip domain-name 2000trainers.com
```

So what just happened? Well, we reinstated domain lookup to begin with. The second step set the IP address of the DNS server that the router will query. The final command set the domain name of the router to 2000trainers.com. This domain name will be appended to hostnames when we don't provide a fully qualified domain name (FQDN). For example, an attempt to resolve the hostname accra would be sent to the DNS server as a request to resolve accra.2000trainers.com.

Backing Up and Restoring Router Configuration Files

Although we've taken a reasonable look at configuring router settings and interfaces, we've only spent a little bit of time on what is perhaps the most important topic of all – knowing how to backup and restore router elements such as configuration files and IOS images.

Backing up and restoring configuration files and IOS images requires you to remember the simple command learned earlier – copy *from to*. In all of the examples that we're going to look at here, you will need to remember how the copy command is structured. In fact, you should make a point of reading the command in that way as well. For example, the command **copy running-config startup-config** should actually be read as "copy settings from the running configuration to the startup configuration". In other words, save the configuration currently stored in RAM to NVRAM, where it will be found the next time the router boots. It may sound simple, but it's also important.

Configuration Files

When you make changes to a router's configuration by using **configure terminal**, you are actually changing the settings stored in RAM – the running configuration. To save these settings to NVRAM, you copy them to the startup configuration, overwriting what was stored there previously. Remember that the copy commands need to be issued from privileged EXEC mode. The example below saves the running configuration settings to the startup configuration.

```
cisco2501#copy running-config startup-config
Destination filename [startup-config]?
Building configuration...
```

You can also enter **copy run star** to achieve the same goal. If you reverse the command, you are copying the contents of the startup configuration into RAM, replacing the current running configuration.

```
cisco2501#copy star run
```

You aren't limited to moving configurations between RAM and NVRAM alone – in fact, you can also save both to a TFTP server. But why would you do this? Mainly to have a backup copy of the files, just in case. For example, it's always a good idea to copy the startup configuration to a TFTP server prior to changing the configuration of a router. That way, if you mess something up, you can always just restore an old configuration file.

Any TFTP server software can be used, including the TFTP server software that can be downloaded from the Cisco website. After installing it on your laptop or a network server, you will ultimately reference it by its associated IP address.

To save the startup or running configuration to a TFTP server, enter the commands **copy star tftp** or **copy run tftp** respectively. The example below shows the running configuration being saved to a TFTP server.

```
cisco2501#copy run tftp
Address or name of remote host []? 192.168.1.21
Destination filename [running-config]?
!!
1136 bytes copied in 5.100 secs (227 bytes/sec)
```

Notice that the command asked us to specify the location of the TFTP server (the address of the remote host). In this case, the TFTP server was running on my desktop, as shown in Figure 7-2. The output shows us not only that the TFTP server is receiving a file called running-config from the IP address of the router, but also points out that the transfer was successful. In larger environments, consider using a destination filename that uniquely identifies the router that the file is associated with. In this case, I simply chose the default name, running-config.

Figure 7-2: Cisco TFTP server output.

Going a step further, you can also restore a configuration file to the router from a TFTP server.

```
cisco2501#copy tftp run
Address or name of remote host []? 192.168.1.21
Source filename []? running-config
Destination filename [running-config]?
Accessing tftp://192.168.1.21/running-config...
Loading running-config from 192.168.1.21 (via Ethernet0): !
[OK - 1136/2048 bytes]
1136 bytes copied in 5.572 secs (227 bytes/sec)
```

This time, we entered the command **copy tftp run**, since we wanted to copy the running configuration file from the TFTP server to the running configuration of the router. We also had to specify the IP address of the TFTP server, and the name of the file to restore.

One last note as far as the configuration files are concerned. During your studies (and in real life) there will be times when you just want to kill the entire configuration of the router, reboot, and start from scratch. This is easy enough to accomplish. In fact, it only takes one command.

Effectively, what you want to do is erase the startup configuration stored in NVRAM. If you do this and reboot the router, it's just like starting anew – when the router does reboot, you'll be presented with the System Configuration Dialog. While practicing for your exams, it is definitely in your best interest to go through the router configuration process many times, or until you feel completely comfortable with it. To erase the startup configuration, use the **erase startup-config** command.

```
cisco2501#erase startup-config
Erasing the nvram filesystem will remove all files! Continue? [confirm][OK]
Erase of nvram: complete
```

It's always a good idea to backup your startup configuration to a TFTP server before issuing this command, just in case you want to return to where you were previously.

Backing Up and Restoring Cisco IOS Images

The **copy** command isn't limited to simply backing up or restoring configuration files. It can also be used to copy your current IOS image to a TFTP server, or to apply a new IOS image to your router. Recall that the Cisco IOS image is stored in Flash memory. Before deciding to install a new IOS image, be sure that your router has enough Flash memory to support the image. Images from different feature sets may require more Flash memory than you currently have installed.

To back up the current IOS to a TFTP server, use the **copy flash tftp** command.

```
cisco2501#copy flash tftp
Source filename []? d1205.bin
Address or name of remote host []? 192.168.1.21
Destination filename [d1205.bin]?
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
3289170 bytes copied in 47.668 secs (69982 bytes/sec)
```

While the copy operation to the TFTP server is completing, the exclamation points shown will scroll across the screen – I have eliminated many of these in order to save space. You will notice that the command requires us to provide information on the image filename and the IP address of the TFTP server. If you're not sure of the name of the image stored in Flash, use the **show flash** command.

```
Router#show flash
System flash directory:
File  Length   Name/status
1   3289170  d1206.bin
[3289236 bytes used, 905068 available, 4194304 total]
4096K bytes of processor board System flash (Read/Write)
```

Since this router has only 4 MB of Flash, we are obviously limited to an image that will fit within that space. The contents of Flash can also be viewed using the **dir** command. You will always need to know the name of the images you want to back up or restore. You should also ensure that the TFTP server is available and has enough room to store the image prior to attempting a backup.

Copying a new IOS image from a TFTP server to Flash involves issuing the command **copy tftp flash**.

```
cisco2501#copy tftp flash
Address or name of remote host []? 192.168.1.21
Source filename []? d1206.bin
Destination filename [d1206.bin]?
Accessing tftp://192.168.1.21/d1206.bin...
Erase flash: before copying? [confirm]
Erasing the flash filesystem will remove all files! Continue? [confirm]
Erasing device... eeeeeeeeeeeeeee ...erased
Erase of flash: complete
Loading d1206.bin from 192.168.1.21 (via Ethernet0): !!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
[OK - 3289170/6578176 bytes]
Verifying checksum...  OK (0xB6BD)
3289170 bytes copied in 89.272 secs (36956 bytes/sec)
```

Notice that Flash memory was erased completely before the new image was copied over to it. Otherwise, there would not have been enough space to complete the copy process. You'll also need to issue the **reload** command after updating the IOS. Finally, use the **show version** command to ensure that the router is now running the IOS version that you intended to install. **Show version** also provides a quick and easy way to learn more about your router's hardware, software, and memory.

```
Cisco2501#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-D-L), Version 12.0(5), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Tue 15-Jun-99 20:08 by phanguye
Image text-base: 0x0000144C, data-base: 0x00637308
ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
BOOTFLASH: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTWARE (fc1)
Router uptime is 2 days, 8 hours, 9 minutes
System restarted by reload
System image file is "flash:d1206.bin"
cisco 2500 (68030) processor (revision A) with 16384K/2048K bytes of memory.
Processor board ID 02285256, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
1 Ethernet/IEEE 802.3 interface(s)
2 Serial network interface(s)
32K bytes of non-volatile configuration memory.
4096K bytes of processor board System flash (Read/Write)
Configuration register is 0x2102
```

On some routers, including those in the Cisco 2500 series, IOS upgrades should be done from the limited IOS version stored in ROM. Depending upon your model and IOS version, you may be prompted to boot into the ROM-based image after issuing the **copy tftp flash** command, or you may need to change what are known as configuration register settings. Configuration register settings will be looked at in detail in chapter 13.

Managing Cisco Routers Via Telnet and HTTP

While the initial configuration of your Cisco router using the console port and a rollover cable may be necessary, you'll eventually want to access routers on your network using telnet sessions. Since telnet is an IP-based application, your routers will need to be configured with at least one valid and reachable IP address to use this method. Also remember that in order to connect to a router using telnet, that router will need a virtual terminal (vty) password configured. If not, any connection attempts will be refused. Notice what happens when we attempt to telnet into the accra router at IP address 192.168.1.45.

```
cisco2501#telnet 192.168.1.45
Trying 192.168.1.45 ... Open
Password required, but none set
[Connection to accra closed by foreign host]
cisco2501#
```

Using telnet to connect to routers is much faster than connecting via the console port. If you recall, back in the hostname section of this chapter we added an entry to our hosts table that resolved the name accra to its IP address. Because of that, we can easily connect to the accra router by simply entering **accra** at the prompt. By the same token, we could just as easily enter the IP address without the **telnet** command preceding it. The router will assume that we're trying to telnet if we don't provide any additional information.

```
cisco2501#accra
Trying accra (192.168.1.45)... Open
User Access Verification
Password:
```

Before issuing the command, I set a vty password on the accra router – notice it prompts us for a password rather than refusing the connection this time.

Telnetting from a client machine to a telnet server is known as a forward telnet session. However, when you connect from a telnet server to another telnet server, it is known as a reverse telnet session. In general, this detail isn't terribly important, but I thought I should mention it since you may come across the term in the Cisco documentation.

After we have connected to the accra router via telnet, we still have the ability to get back to the prompt of the cisco2501 router using a special key sequence. By pressing **Ctrl+Shift+6** together and then the letter **X**, we return to the original prompt. The telnet session to the accra router is still open – we've just left it temporarily, perhaps to open yet another reverse telnet session to another router. The ability to switch between connections is much more convenient than constantly logging off and back on.

However, having multiple telnet session open can also be a little confusing. So how can you recall sessions that you've initiated and access them again? Well, you should start off with the **show sessions** command. This will list the currently active sessions, as shown below.

```
Cisco2501#show sessions
Conn Host          Address      Byte  Idle Conn Name
*  1 192.168.1.45  192.168.1.45   0     0   192.168.1.45
Cisco2501#
```

The asterisk shown above is used to designate the last session accessed. To reconnect to this telnet session, you can either press Enter twice (this will always access the last session), or enter the number associated with the Conn header – in this case 1, followed by the Enter key.

To end a telnet session, you have two main options. From within a session, just type **exit**.

```
accra>exit
[Connection to accra closed by foreign host]
cisco2501#
```

If you want to close a session without actually being in it, use the **disconnect** command. For example, to close that accra session from the cisco2501 command prompt, we would enter **disconnect** followed by the associated connection number.

```
cisco2051#disconnect 1
Closing connection to 192.168.1.45 [confirm]
cisco2501#
```

While configuring a router using telnet may be common, it is also possible to configure your router via a web browser. Although it's disabled by default, your router has its own mini HTTP server built in. This provides yet another way to gain access to the router for the purpose of issuing commands. To enable the HTTP server, use the command **ip http server** from global configuration mode.

```
cisco2501(config)#ip http server
cisco2501(config)#
```

After doing this, open your web browser and point it to one of your router's IP addresses. While the browser interface may not be pretty, it's worth being aware of its existence. In general, it really provides no more functionality that what is available in a telnet session, although it does offer the ability to issue commands using hyperlinks. For the most part, I suggest that you keep the HTTP server turned off, since it offers another point of access for potential hacking. After taking a look, the HTTP server can be turned off using the **no ip http server** command.

IOS Diagnostic Utilities - Ping and Traceroute

The Cisco IOS provides two primary utilities that you'll need to be familiar with for the purpose of testing connectivity – **ping** and **traceroute**.

Ping

You are probably familiar with the ping utility from Windows or Linux. The version included with the Cisco IOS provides significantly enhanced functionality, and can be used to test connectivity for a variety of different protocols including IP, IPX, AppleTalk and more. To get a sense of the functions provided by ping, issue the **ping** command followed by the question mark.

```
cisco2501#
ping ?
WORD       Ping destination address or hostname
appletalk  Appletalk echo
decnet     DECnet echo
ip         IP echo
ipx        Novell/IPX echo
tag        Tag encapsulated IP echo
<cr>
```

Notice the range of protocols that ping can work with. In fact, the list can be even longer depending on the protocols supported by your IOS version. At the most basic level, ping sends out echo request messages and expects to receive back echo replies. It is important to be clear about the information that a ping provides. For example, if you can ping an IP host on a different network, it suggests that both hosts have TCP/IP correctly initialized and configured, and that routing between the networks is also configured correctly. In cases where you cannot ping a remote host, don't jump to the conclusion that the remote host is unavailable or misconfigured – though it might be, the problem may also be a configuration issue with the source host, or potentially some routing-related (or physical connectivity) issue between the two. As a general rule, use the following steps to determine the source of connectivity issues between your PC and a remote system:

1. Assuming that your IP address, subnet mask, and default gateway are correct, attempt to ping a host on a different subnet. If this fails, one possibility is that routing is not configured correctly.

2. If pinging a remote host fails, attempt to ping your default gateway. If this fails, it may indicate that TCP/IP is not configured correctly on your local router interface, on your host PC, or that the router interface has not been enabled with the **no shutdown** command.

3. If pinging your default gateway fails, try pinging your host's configured IP address. If this fails, it can may mean that you have configured your host PC's IP address incorrectly, or that TCP/IP is not properly installed or initialized on the host system.

4. If pinging the host's IP address fails, try pinging the loopback address – 127.0.0.1. If this fails, it generally indicates that TCP/IP is not properly installed or initialized on your host system.

To test IP connectivity, use the **ping** command followed by a hostname or IP address.

```
cisco2501#ping accra
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.209, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/4/4 ms
```

The ping was successful in this case, as illustrated by the five exclamation points and the final statement. In cases where a ping fails, you'll see a message similar to the one shown below.

```
cisco2501#ping 192.168.1.234
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 192.168.1.234, timeout is 2 seconds:
.....
Success rate is 0 percent (0/5)
```

When the exclamation points are replaces by dots, it means that for whatever reason, the destination host did not respond successfully. Again, this could suggest a range of issues including misconfiguration, physical network issues, routing problems, and so forth.

An extended ping allows a higher degree of control than the default ping settings, including the ability to change the repeat count, size of the datagrams, and so forth. The example below outlines an IPX ping using the extended ping interface.

```
cisco2501#ping
Protocol [ip]: ipx
Target IPX address: 101A.0060.5cc4.f41b
```

```
Repeat count [5]:
Datagram size [100]:
Timeout in seconds [2]:
Verbose [n]:
Type escape sequence to abort.
Sending 5, 100-byte IPXcisco Echoes to 101A.0060.5cc4.f41b, timeout is 2 seconds
:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 4/5/8 ms
```

Although generally a TCP/IP utility, ping works with a number of protocols beyond IP and IPX. For a complete list of the protocols that ping supports on your router, issue the command **ping ?**.

Traceroute

Another useful utility is testing connectivity, especially in routed environments, is traceroute. While ping tests for basic connectivity with another host, traceroute will show you the path that a packet takes (in terms of crossing intermediate routers) between a source and destination. Since we haven't set up routing yet, traceroute won't provide us with much useful information. In a routed environment, traceroute provides valuable information because it helps to indicate at which point in a packet's travels a failure is occurring. Issues might include an intermediate router being offline, or physical connection problems.

Traceroute works by sending groups of 3 UDP datagrams to the destination address specified, with varying time to live (TTL) values. For example, imagine there are three routers between our system and the destination host that we're to determine the path to. Traceroute will send out 3 UDP datagrams with a TTL of one. When these hit the first router in the path, their TTL will be decremented by one, causing the packets to expire. ICMP "time exceeded" messages will be sent back to the source host. It will then send out another 3 UDP datagrams with a TTL of 2, which will exceed their TTL at the second router. This process continues until the destination host is reached. The cumulative information provided shows the path to the destination. If the process fails at any point, this indicates or suggests a problem area between the source and destination. Traceroute is an exceptionally simple and powerful troubleshooting tool in routed environments. To use it, simply enter **traceroute** followed by the destination IP address or hostname.

```
cisco2501#traceroute 192.168.1.209
Type escape sequence to abort.
Tracing the route to 192.168.1.209
1 192.168.1.209 4 msec 40 msec *
```

As I mentioned previously, traceroute doesn't provide very much information on our network yet. Once some routing is configured, we'll be able to see multiple hops in the path to a destination.

Working with the Cisco Discovery Protocol (CDP)

Cisco Discovery Protocol (CDP) is a proprietary protocol developed by Cisco that provides a quick and easy way to find out about neighboring Cisco devices on your network. Enabled by default, CDP provides a variety of information on neighboring Cisco routers and switches. CDP messages use the Ethernet SNAP frame type.

To view information about neighboring Cisco devices, use the **show cdp neighbors** command.

```
cisco2501#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
S - Switch, H - Host, I - IGMP, r - Repeater
Device ID    Local Intrfce     Holdtme    Capability  Platform  Port ID
accra          Eth 0          146          R        2500      Eth 0
```

The information provided shows that our Cisco 2501 has received CDP announcements from one other piece of equipment – in this case, another Cisco 2500 router. Notice the capability codes listed first. These provide you with an overview of the different types of equipment that can be found via CDP – R designates a router, S a switch, and so forth. These codes are then associated with the devices found, under the Capability header.

Obviously a variety of information is provided by the **show CDP neighbors** command, but what does it all mean? Each section is looked at below.

§ **Device ID**. The hostname of the neighboring device.

§ **Local Interface**. The interface on which this router received information about the neighboring device.

§ **Holdtime**. The amount of time the router will store this information before dropping it from memory, if additional CDP packets are not received.

§ **Capability**. The type of device that announced itself using CDP.

§ **Platform**. The hardware platform of the neighboring equipment.

§ **Port ID**. The port from which the CDP packet was sent on the neighboring device.

To obtain more detailed information on any CDP neighboring device, use the **show cdp neighbor detail** command.

```
cisco2501#show cdp neighbor detail
-------------------------
Device ID: accra
Entry address(es):
IP address: 192.168.1.45
Novell address: 101A.0060.5cc4.f41b
Platform: cisco 2500,  Capabilities: Router
Interface: Ethernet0,  Port ID (outgoing port): Ethernet0
Holdtime : 178 sec
Version :
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-D-L), Version 12.0(5), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Tue 15-Jun-99 20:08 by phanguye
```

This command provides six primary pieces of information about neighboring devices including its hostname, logical addresses (CDP will provide one logical address per protocol), platform, connected interface, CDP holdtime, and IOS version. If you're looking for a simple way to determine an IP or IPX address of a neighboring device, this command is also the way to go.

By default, CDP packets are sent out every 60 seconds. The default holdtime is set to 180 seconds – again, this is the length of time that the device will hold information provided by CDP if an update is not received.

To view the CDP holddown and timer values currently configured on your router, use the **show cdp** command.

```
cisco2501#show cdp
Global CDP information:
Sending CDP packets every 60 seconds
Sending a holdtime value of 180 seconds
```

To reconfigure either value, use the **cdp timer** and **cdp holdtime** commands, as shown below. We'll set the holdtime on the router to 200 seconds, and the timer value to 70 seconds. Remember that in order to change just about any setting, you will need to be in global configuration mode.

```
cisco2501#config t
Enter configuration commands, one per line.  End with CNTL/Z.
cisco2501(config)#cdp holdtime 200
cisco2501(config)#cdp timer 70
```

In some cases, you may not want CDP enabled on a given interface. To turn it off for a single interface, access the interface's configuration and issue the **no cdp enable** command.

```
cisco2501(config)#int e0
cisco2501(config-if)#no cdp enable
```

In the example above, I disabled CDP, but only on interface Ethernet 0. If you change your mind and want to re-enable CDP on an interface, use the **cdp enable** command. To globally enable or disable CDP, use the **cdp run** or **no cdp run** commands respectively from global configuration mode.

```
cisco2501(config)#cdp run
cisco2501(config)#no cdp run
```

To view the CDP properties of all interfaces on your router, use the **show cdp interface** command.

```
cisco2501#show cdp interface
Ethernet0 is up, line protocol is up
Encapsulation ARPA
Sending CDP packets every 60 seconds
Holdtime is 180 seconds
Serial0 is down, line protocol is down
Encapsulation HDLC
Sending CDP packets every 60 seconds
Holdtime is 180 seconds
Serial1 is down, line protocol is down
Encapsulation HDLC
Sending CDP packets every 60 seconds
Holdtime is 180 seconds
```

Configuring a Cisco Router: Practice Run

In order to be successful on your CCNA exam especially, you will definitely need to be familiar with the commands listed in this chapter, as well as those found in upcoming chapters. To get a sense for how easy it is to configure a Cisco router, we're going to walk through a scenario, beginning with a blank startup configuration. Our goal is to configure the settings that meet the requirements outlined below.

If you already have a configuration on your router, back it up by saving the running configuration to a TFTP server before getting started. The next step will be clearing your startup configuration by using **erase startup-config**. For the purpose of this example, we'll handle the entire configuration from the console.

The bullet points below outline the required configuration for our router:

- Hostname of toronto-1
- Enable secret password of cisco99
- Console password of cisco, should be encrypted
- Auxiliary password of cisco, no encryption
- Allow 3 virtual terminal connections with a password of cisco, no encryption
- Interface Ethernet 0 IP address 192.168.1.99/24
- Interface Ethernet 0 IPX network 101a with 802.2 encapsulation
- Interface Serial 0 IP address 192.168.2.99/24
- Interface Serial 1 IP address 192.168.3.99/24 configured as a DCE interface with a clock rate of 56000
- Set a description on interface serial 1 of "WAN link Toronto-Montreal"
- Terminal history size of 15
- Message of the day banner "Unauthorized Access Prohibited"
- View CDP neighbors, and then set the holdtime interval to 200 and the CDP timer to 70.
- Back up the current IOS image to a tftp server at address 192.168.1.21
- Save the current running-configuration to NVRAM
- Back up the new startup configuration to the same TFTP server.

One note – we are not going to use the System Configuration Dialog to configure the router. It's the command line interface for us! When prompted with the System Configuration Dialog, enter no, and also be sure to terminate auto-install if prompted.

For the purpose of the example, let's follow the configuration exactly as it appears in the bullet points listed above.

```
Router con0 is now available
Press RETURN to get started!
Router>enable
Router#config terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)#hostname toronto-1
toronto-1(config)#enable secret cisco99
toronto-1(config)#service password-encryption
toronto-1(config)#line con 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco
```

```
toronto-1(config-line)#exit
toronto-1(config)#no service password-encryption
```

Notice the **service password-encryption** command is completed with the "no" version of the command. This meets the requirement of encrypting the console password only.

```
toronto-1(config)#line aux 0
toronto-1(config-line)#login
toronto-1(config-line)#password cisco
toronto-1(config-line)#line vty 0 2
toronto-1(config-line)#login
toronto-1(config-line)#password cisco
```

When switching between line configurations, there is no need to exit back to the main global configuration mode prompt. Instead, simply enter the command to access the next line or interface you need to configure.

```
toronto-1(config-line)#interface ethernet 0
toronto-1(config-if)#ip address 192.168.1.99 255.255.255.0
toronto-1(config-if)#no shutdown
toronto-1(config-if)#exit
```

Don't forget to issue the **no shutdown** command after configuring the properties of an interface!

```
toronto-1(config)#ipx routing
toronto-1(config)#int e0
toronto-1(config-if)#ipx network 101a encap sap
```

Configuring the router for IPX required first issuing the **ipx routing** command from global configuration mode. The Ethernet encapsulation type specified was 802.2, which is designated by the keyword **sap**.

```
toronto-1(config-if)#int serial 0
toronto-1(config-if)#ip address 192.168.2.99 255.255.255.0
toronto-1(config-if)#no shutdown
toronto-1(config-if)#int s1
toronto-1(config-if)#ip address 192.168.3.99 255.255.255.0
toronto-1(config-if)#clock rate 56000
toronto-1(config-if)#description WAN link Toronto-Montreal
toronto-1(config-if)#no shutdown
toronto-1(config-if)#^Z
```

Configuring the serial interfaces was fairly straightforward – just don't forget the **clock rate** command on the DCE interface.

```
toronto-1#terminal history size 15
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#banner motd #Unathorized Access Prohibited#
toronto-1(config)#^Z
```

The **terminal history size** command is issued from privileged EXEC mode, not global configuration mode. After configuring it, I moved back into global configuration mode to enter the banner MOTD message – remember to use the same starting and ending delimiter characters!

```
toronto-1#show cdp neighbors
Capability Codes: R - Router, T - Trans Bridge, B - Source Route Bridge
S - Switch, H - Host, I - IGMP, r - Repeater
Device ID    Local Intrfce     Holdtme    Capability  Platform  Port ID
accra          Eth 0          152           R        2500     Eth 0
toronto-1#config t
Enter configuration commands, one per line.  End with CNTL/Z.
toronto-1(config)#cdp holdtime 200
toronto-1(config)#cdp timer 70
toronto-1(config)#^Z
```

The CDP commands are fairly straightforward as well. The next step is backing up the IOS image to the specified TFTP server.

```
toronto-1#copy flash tftp
Source filename []? d1206.bin
Address or name of remote host []? 192.168.1.21
Destination filename [d1206.bin]?
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
3289170 bytes copied in 82.572 secs (40111 bytes/sec)
```

I omitted some of the exclamation points from the output above to save space. The next step is saving the running configuration to the startup configuration, followed by saving the startup configuration to the TFTP server.

```
toronto-1#copy run star
Destination filename [startup-config]?
Building configuration...
toronto-1#copy star tftp
Address or name of remote host []? 192.168.1.21
Destination filename [startup-config]?
!!
825 bytes copied in 0.232 secs
```

And we're done! When issuing commands, it may take some time to become comfortable remembering where they should be issued from. Don't worry or get too frustrated – you can always use the question mark to obtain help when memory fails you. As a best practice, make a point of viewing the startup configuration prior to shutting down or rebooting the router. You'll want to ensure that you saved your changes, after all.

CCNA Study Guide Chapter 7 Summary

By Dan DiNicolo, June 27th, 2006 Posted in **CCNA Study Guide Chapter 07.** Subscribe to our

RSS Feed

Chapter 7 began with a look at the basics of the boot process of a Cisco router. Next, the initial configuration of a Cisco router from the System Configuration Dialog prompts was explored. Extended setup mode was used to assign the router its initial configuration parameters including a hostname, passwords, and interface IP addresses.

Configuration from the IOS command line was looked at next, beginning with the process of logging into and out of a router. The differences between user and privileged EXEC mode was discussed, including how they can be identified by their associated command prompts.

A look at the configuration files on a Cisco router outlined the differences between the running configuration stored in RAM and the startup configuration stored in NVRAM. The syntax for the copy command was also looked at, including its use in saving the running configuration to the startup configuration. An overview of IOS shortcuts explained navigation techniques that can be used from the command line, as well as the use of truncated or shorthand commands. A look at the terminal history command explained how to change the buffer size, or disable it within a session.

The IOS help function was explored next, including the ways in which it can be used to find or complete commands.

A look at IOS configuration modes introduced global configuration mode, as well as the difference between the configure terminal, memory, and network options. Various levels of global configuration mode were also explored, including those associated with interface, line, and routing protocol configuration. The router prompts associated with each were also explained.

The configuration of passwords was also looked at, including the purpose and configuration of enable, enable secret, console, auxiliary, and virtual terminal passwords. The service password-encryption command was also discussed as a way of encrypting passwords not usually encrypted by default. The ability to configure a logon banner was also explored, using the banner motd command.

The interface configuration section outlined the procedure for setting IP addresses on interfaces, as well as the importance of the no shutdown command. An overview of configuring serial interfaces as DCE for lab environments was also discussed. The procedure for enabling and configuring IPX was also looked at, including the configuration of Ethernet encapsulation settings for IPX. The show interface command was discussed as a way to gain information about an interface, including its physical and data link characteristics.

A look at hostnames explained how to change the hostname on a Cisco router, configure a hosts table, and finally how to configure a router to use a DNS server for name resolution.

The copy command was then looked at in more detail. It explained the backup and restoration of configuration files and IOS images, both between memory areas and via a TFTP server. A look at telnet explained the basics of initiating, disconnecting, and switching between sessions. Configuration of the router via a web browser was also briefly explained. Diagnostic utilities were looked at next, including both ping and traceroute.
A look at the Cisco Discovery Protocol followed, including how it can be used to gain valuable information about neighboring Cisco devices.

Chapter 8: Understanding and Configuring Routing

Understanding and Configuring Routing

In Chapter 7, you learned the basics of configuring a Cisco router from the IOS command line. After completing important initial tasks like setting passwords and configuring interfaces, it's time to get the router to start serving its real purpose – routing traffic between networks. In order to do that, you need to take a look at more than just the commands required to configure routing and associated protocols. More importantly, you need to begin by understanding how routing works on a conceptual level and defining what it is that you're actually trying to accomplish.

For the purpose of both the CCNA and CCDA, you'll need to understand how routing works, the differences between routing protocols, and finally how routing is configured. The topics that I will cover in this chapter include:

- An introduction to routing
- The difference between routed and routing protocols
- Understanding how network communication occurs in a routed environment
- Static IP routing
- Distance vector versus link state routing protocols
- Dynamic IP routing with the Routing Information Protocol (RIP)
- Dynamic IP routing with Interior Gateway Routing Protocol (IGRP)
- Default routing
- Routing IPX traffic
- The IPX Routing Information Protocol (RIP) and Service Advertising Protocol (SAP)

- Other routing protocols including RIP v2, RIPng, OSPF, OSPFv3, EIGRP, NLSP, RTMP, and AURP
- An overview of the differences between classful and classless routing protocols
- Route redistribution and route summarization
- Layer 3 switching

Many of the routing protocols listed above are not required knowledge for the CCNA exam. However, you will definitely need to be familiar with these concepts for the CCDA exam.

Introduction to Routing

You may recall from previous chapters that in order for systems in one broadcast domain to communicate with systems in another broadcast domain, a router must be involved. A router usually acts as the demarcation point between broadcast domains. Remember that a broadcast domain is a Layer 2 concept. Routing, on the other hand, happens at Layer 3 of the OSI model – the Network Layer. We'll look at how these two layers interact in a routed environment shortly.

The act of routing is primarily concerned with moving traffic between networks. This can be as simple as having two networks directly connected via a single router. More commonly, we will need to be concerned with moving traffic between networks across a larger internetwork – in other words, there may be many routers between a source and destination network. In order to get traffic from one network to another, we will need to somehow make routers aware of where they should next forward traffic, in order for it to reach its final destination. This can be done in different ways, each with associated pros and cons. Ultimately, the way in which we choose to configure our routers will depend on our goals in a specific network environment.

At the most basic level, a router does no more than make decisions about getting to a destination network. In that way, routing isn't much different than driving a car between your home and work – there may be many ways to get to work, and ultimately you have to choose a path. At any given intersection, you have to make a decision. Should you turn the car and take a different path, or should you carry on and take what you usually consider to be the best route? Many factors will likely impact your decision. For example, you might consider the most scenic route, the one with the highest speed limit, or simply may be trying to avoid roadwork. Regardless of the route you choose, you ultimately make a decision at each major intersection on your drive. In the same way, routers are like intersections on a network – at each point, a decision has to be made as to where to send data next. There may be many ways to get to a destination, but ultimately the goal of routing is to determine the best path. What constitutes the "best" path depends on how your network is configured, as you will see shortly.

Routed Versus Routing Protocols

Before we take a look at a how a path is determined, it's important to be able to differentiate between a routed and routing protocol. Although the names are similar, there is a big difference between the two. A routed protocol is a Network Layer protocol that is used to move traffic between networks. IP, IPX, and AppleTalk are all examples of routed protocols. Routed protocols allow a host on one network to communicate with a host on another, with routers forwarding traffic between the source and destination networks. They are characterized by logical addressing (such as an IP or IPX address) that not only identifies a source or destination host, but also the network (or subnet) on which they reside. In contrast, a protocol like NetBEUI does not use any logical addressing, and isn't routable. Why is that? Because when a router comes across a

NetBEUI packet, it has no way of determining where the destination host resides, since a NetBEUI packet does not include a logical destination address, only a name. The protocol cannot be routed; this means that communication between NetBEUI hosts is limited to occurring within a single non-routed network. Obviously that limits NetBEUI's usefulness on a large internetwork.

Routing protocols serve a different purpose. Instead of being used to send data between source and destination hosts, a routing protocol is used by routers to exchange routing information with one another. For example, if we want our routers to dynamically "learn" about networks from one another, we configure them with a common routing protocol such as RIP or IGRP. Routers use routing protocols to exchange information about the networks they are aware of. In other words, routing protocols allow routers to "talk" to one another. This doesn't mean that we need to configure routing protocols on every internetwork – there are other options, such as statically defining paths to destination networks on each router. If that sounds like a lot of work, you're right. Once a network moves beyond a few routers, you will definitely need to consider adding one or more routing protocols. A variety of routing protocols exist beyond RIP and IGRP, including OSPF, EIGRP, AURP and others. The reason for choosing one over another will be influenced by a number of factors, including the size of a network, required performance, and the routed protocol(s) in use.

Communications in Routed Network Environments

In order to appreciate the role of a router, we need to take a look at how communication happens between hosts on different networks in a routed environment. For the purpose of these illustrations, I'm going to assume that our network is running the most popular routed protocol suite, TCP/IP. We'll begin with a look at a simple internetwork consisting of two networks, both connected to a single router. This will be followed by a look at a larger internetwork, where multiple networks are connected via many routers.

Routed Network – A Simple Example

Take a look at Figure 8-1, where a single router connects to networks 192.168.0.0/16 and 10.0.0.0/8. By this point, you should recognize these network addresses as two of the private IP address ranges that are not valid on the public Internet. The router is a simple 2-port Ethernet router, with interface E0 connected to the 192.168.0.0 network, and interface E1 connected to the 10.0.0.0 network. Both interfaces on the router have already been assigned their IP addresses, 192.168.0.1 and 10.0.0.1 respectively. Using an IP address that ends in ".1" for router interfaces is fairly common. It is not a rule, but rather a convention. You may or may not choose to follow this convention, but it does make the IP address of a router interface easier to remember.

Figure 8-1: A simple routed internetwork.

Believe it or not, without configuring anything else on this router, it is already capable of routing data between network 192.168.0.0 and network 10.0.0.0. How is this possible? Well, the first thing you need to understand is that after assigning a Cisco router IP addresses, it knows about the networks that it is directly connected to, and the command **ip routing** is enabled by default. In other words, the router is aware that interface E0 is connected to the 192.168.0.0 network, and that interface E1 is connected to network 10.0.0.0. A router will automatically add both of these networks to its routing table. If our router receives a packet destined for the 192.168.0.0 network on interface E1, it will know to forward it out interface E0. In the same way, interface E0 knows that any traffic destined for network 10.0.0.0 should be forwarded out interface E1. A router always knows how to get to the networks to which it is directly connected, without any additional configuration. Things get a little more complex when a router needs to get to networks that are not directly connected.

Each network in this example also has a single host configured, as shown in Figure 8-2. Notice that each host has an IP address, subnet mask, and default gateway value assigned. The default gateway IP address is that of the local router interface on the host's network. Hosts will forward traffic to the router when they calculate that the destination host is not on their local network.

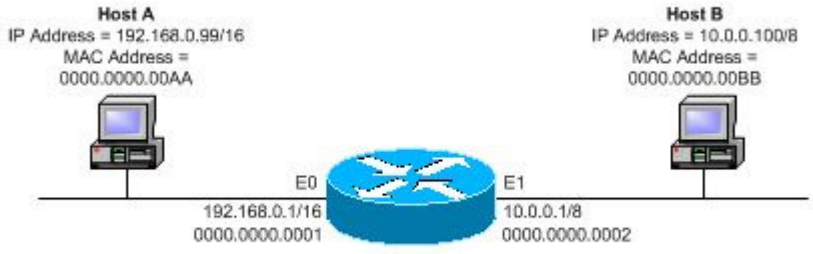


Figure 8-2: Simple routed network with 1 host on each network.

In this example, each network is a Layer 2 broadcast domain, running Ethernet. Both networks are also unique Layer 3 IP networks, as shown by their logical layout. On most (but not necessarily all) networks, each broadcast domain will be associated with a unique IP network (or subnet) address.

There are cases, however, where the preceding statement is not necessarily true. For example, a company may run out of IP addresses and choose to map two subnets to a single broadcast domain. Making things work would then involve adding a second IP address to the local router

interface. Even through hosts will be in the same broadcast domain, the router will still need to route packets between the different subnets. I didn't say it was efficient, but it is something that you may come across.

For the purpose of illustration, I'm going to assume that Host A wishes to communicate with Host B. In order for these two hosts to communicate, our router will obviously need to be involved. Let's take a look at how the communication process occurs when Host A attempts to create an HTTP session with Host B.

1. The first step in the process is the creation of an HTTP request at the Application Layer. In this case, Host A is requesting a web page from Host B. After formatting the HTTP request, the data is passed down to the Transport layer. Remember, our interaction is happening over TCP/IP.
2. Once the Transport Layer on Host A receives the data, it will add header information that will include the source and destination TCP ports. In this case, the destination TCP port will be 80, and the source port some number above 1024. Once this is complete, the segment will be passed to the Network Layer.

3. At the Network Layer, the IP header will be added. This header will include a variety of information, but most importantly the source and destination IP addresses. The source address is 192.168.0.99, and the destination address is 10.0.0.100. The next step is passing the packet down to the Data Link layer.
4. Recall the ANDing process looked at in Chapter 5. This is the process that a host uses to determine whether a destination host is local or remote. In this case, the ANDing results will be different, which means that the destination is on a remote network. When a destination is remote, the packet cannot be sent directly to that host. Instead, it must be first sent to a router.
5. Pay particular attention to this step. Host A knows that it needs to send this packet to its local router, but the router is not the ultimate destination IP address – 10.0.0.100 (Host B) is. In order to get this data to the router, our host must frame the packet with the destination MAC address – in this case, the MAC address associated with the router's E0 interface. To obtain this MAC address, it will send out an ARP request looking for the MAC address associated with 192.168.0.1. Once the router responds, Host A will frame the packet. In this case, the source address is the MAC address of Host A, and the destination address is the MAC address of the router's E0 interface. This is illustrated using the simplified frame shown in Figure 8-3. Once this step is complete, the frame will be sent across the network as a series of bits.

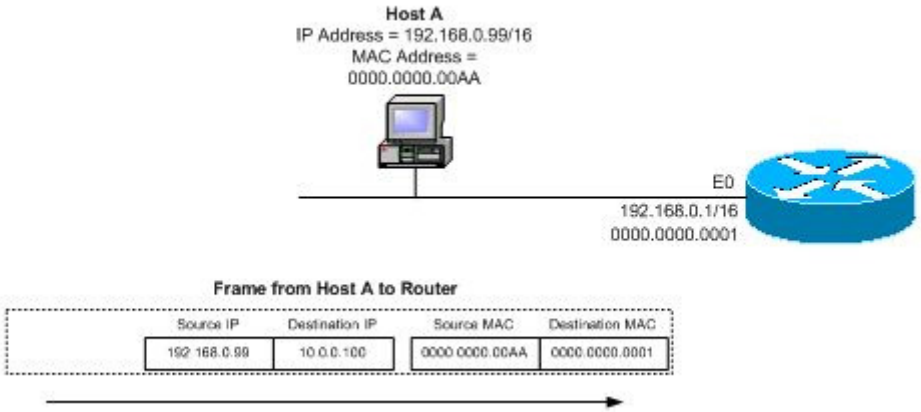


Figure 8-3: The source and destination MAC and IP addresses in the frame sent from Host A to the router.

6. Ultimately, this frame will reach interface E0 on the router. The router will notice that the frame is destined for its MAC address, and as such it will process the frame. After calculating the frame's CRC, it will strip off the frame header and pass the resulting packet up to the Network layer.

   At this point, the router will notice that the destination IP address is not it's own. When a router receives a packet that is not destined for it specifically, it looks in its routing table to see whether it has a route defined to the destination network, as shown in Figure 8-4. In this case, the router sees that it is directly connected to network 10.0.0.0/8, and also determines that it should forward the packet via interface E1. Before it can forward the packet, however, it has to pass it back down to the Data Link Layer for re-framing.

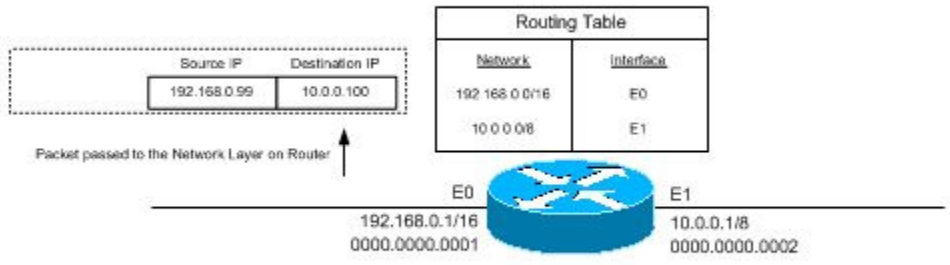


Figure 8-4: Packet passed to the Network Layer on the router, routing table consulted.

7. At the Data Link layer, the router will re-frame the packet with a new header, meaning that source and destination MAC addresses will need to be added. In this case, the source MAC address is now the MAC address of interface E1 on the router. The destination MAC address will be obtained via an ARP request to IP address 10.0.0.100. Once Host B replies with its MAC address, this will be added to the frame as the destination MAC address, as shown in Figure 8-5. The router will then calculate a new CRC for the frame, and forward it through interface E1 as a series of bits.

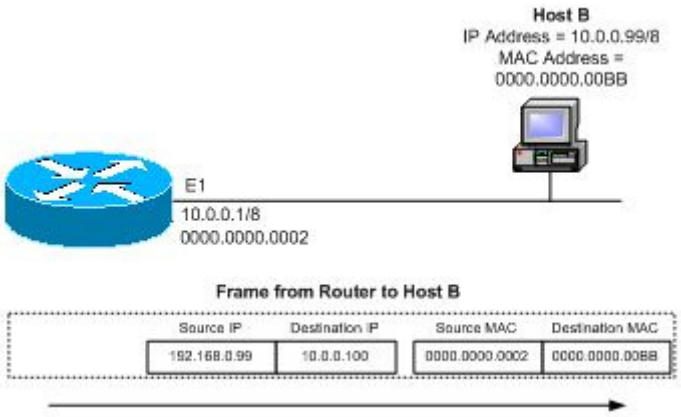


Figure 8-5: The packet is reframed by the router and sent to Host B.

8. Eventually the frame will arrive at Host B. By inspecting the destination MAC address, Host B will recognize that the frame is meant for it, calculate the CRC, strip off the framing, and pass it up to the Network Layer.
9. At the Network Layer, Host B will also recognize its IP address as the destination. This means that it has to process the packet further. In the IP header, it will see that the data should be passed to TCP at the Transport layer.
10. At the Transport Layer, Host B will look at the destination TCP port, and recognize that the data contained in the segment should be sent to TCP port 80. This is the port on which the web server application is listening for connections.
11. The web server on Host B will process the HTTP request contained in the data. Ultimately it will need to send a reply, where the whole process happens again, in reverse.

If those twelve steps seem like an awful lot of work to pass data between two hosts, you're right. However, this is the way things work when you want to communicate between hosts on a routed network. Think of it this way – we are effectively using a Layer 3 protocol (IP) to communicate between different Layer 2 networks. In this case, both of those networks were Ethernet, but that won't always be the case. For example, what if Host B had resided on a Token Ring network? In that case, our router would require one Ethernet interface and one Token Ring interface. When the data was sent to the router from Host A, it would have been framed for Ethernet. After the router stripped off the framing and passed the data up to the Network layer, it would determine that the packet needed to be forwarded out the Token Ring interface, where it would need to be framed for Token Ring. As even this very basic example suggests, a router is doing a great deal of work to every packet it encounters – not only does it have to make a forwarding decision based on the destination IP address, but it also has to reframe every single packet that it forwards on its way. By this point, you should be recognizing some of the reasons as to why routing is typically much slower than switching.

You should also recognize that although packets are re-framed at the router, the actual source and destination IP addresses never change. In fact, the only thing that the router touches for certain in the IP header is the time-to-live (TTL) value. Recall that a router will always decrement the TTL on an IP packet that it forwards by one, which ultimately ensures that packets don't end up being forwarded around a network forever. Once its TTL expires, a packet is discarded.

A router may further alter an IP packet under special circumstances, such as when the maximum transmission unit (MTU) of connected networks is different. For example, imagine that the

networks connected were Ethernet and Token Ring. Token Ring has a much larger maximum frame size than Ethernet. When a frame is received on a Token Ring interface, the packet it contains may be much larger than what Ethernet can handle (recall that the MTU of an Ethernet frame is only 1518 bytes). In this case, the router will "chop up" or fragment the packet into a number of smaller packets, then reframe each and send them on their way. Again, it becomes clear that there is more to what a router does than initially meets the eye.

Understanding Routed Networks – A More Complex Example

A router knows about the networks to which it is directly connected. However, it's up to you to enable a router to find out about networks to which a direct connection does not exist. As I mentioned earlier, a router primarily finds out about remote networks in one of two ways – via a static routing table entry, or through the use of a dynamic routing protocol like RIP or IGRP. Again, this is best illustrated by an example. In Figure 8-6, our network includes four networks and three routers.

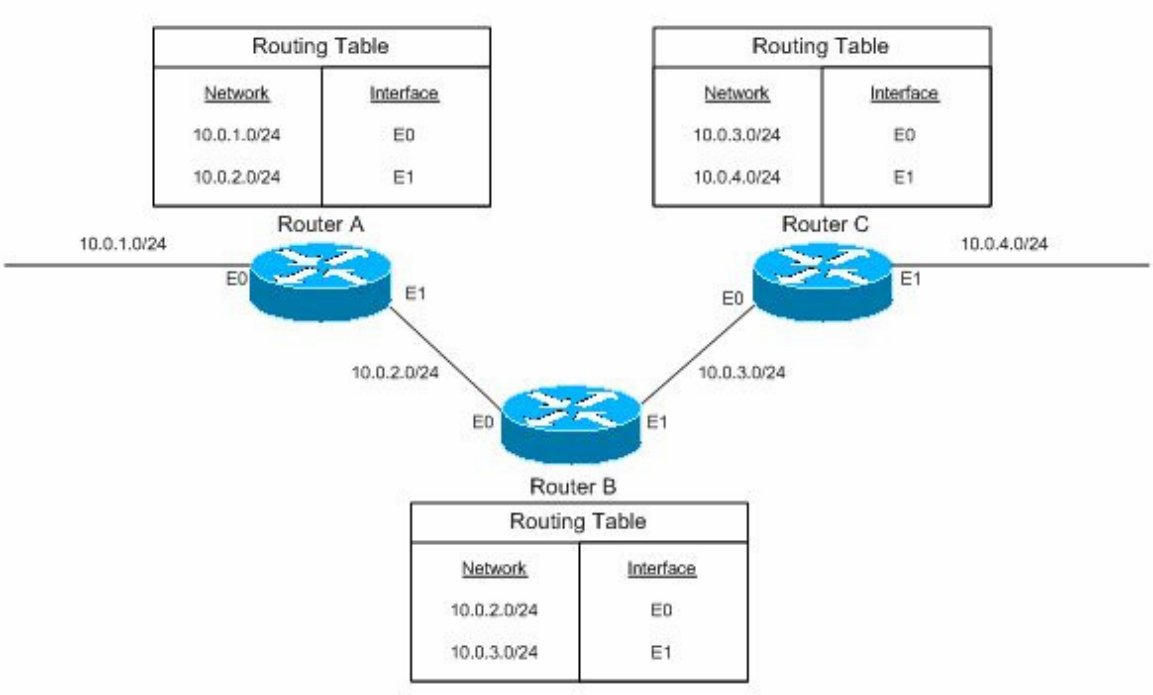


Figure 8-6: Routing tables showing directly connected networks only.

For this example, I'm again going to assume that all of our connections are via Ethernet ports. Remember that all 3 routers known about their directly connected networks. Given the current configuration, Router A can route packets between network 10.0.1.0/24 and network 10.0.2.0/24. Router B can route packets between networks 10.0.2.0/24 and 10.0.3.0/24. Finally, Router C can route packets between 10.0.3.0/24 and 10.0.4.0/24. From the routing tables shown in Figure 6, it should be clear that each router is unaware of the two networks to which it has no direct connections. For example, Router A doesn't know how to reach either network 10.0.3.0/24 or 10.0.4.0/24.

First off, let's be clear on one thing – when a router doesn't know how to get to a destination network, it will drop any packets destined for that network and send an ICMP destination unreachable message to the host. For example, if Router A receives packets destined for network 10.0.3.0/24, it will drop them as things currently stand. The router won't guess or make any assumptions on its own, which is good news (later in the chapter we'll discuss how this can be changed through the use of default routes). This also means that it's up to you to make the whole network route – it actually isn't that big a deal in this case, as you'll see shortly.

For now, I'm going to assume that we're setting up static routing on our simple network. If you think about things, we really need to accomplish three main goals:

- Configure Router A with a way to get to networks 10.0.3.0/24 and 10.0.4.0/24.
- Configure Router B with a way to get to networks 10.0.1.0/24 and 10.0.4.0/24.
- Configure Router C with a way to get to networks 10.0.1.0/24 and 10.0.2.0/24.

Let's start with Router A. We need to tell Router A how to get to networks 10.0.3.0/24 and 10.0.4.0/24. This will involve adding static entries to Router A's routing table. The key thing to remember is that routers will always communicate directly with a host on a connected network, or pass a packet on to the next "hop", which is another router. There is no magic flying across networks or avoiding routers in the path between a source and destination host. Keep in mind everything the router does will happen one hop at a time.

Take another look at Figure 6. The key to configuring routing is to figure out what the next hop address is. First, we are trying to determine where Router A should send packets destined for network 10.0.3.0. Obviously it cannot communicate with network 10.0.3.0/24 directly, since it doesn't have an interface connected to this network. As such, Router A will need to forward packets destined for network 10.0.3.0/24 to Router B – in this case, interface 10.0.2.2. The address 10.0.2.2 is our next hop address on the journey to network 10.0.3.0/24. Once packets arrive at 10.0.2.2, Router B will be able to forward them to network 10.0.3.0/24 – a directly connected network. Similarly, we need to tell Router A how to get to network 10.0.4.0/24. Since it is in the same direction, we will also add a static route that sends packets destined for network 10.0.4.0 to the next hop – which is also 10.0.2.2. Since network 10.0.4.0/24 isn't directly connected to Router B, it will also need a route defined to this network.

Routing table entries will also need to be added to Router B and Router C. In this case, the entries below need to be added, as also shown in Figure 8-7.

- Router B needs a static route defined to network 10.0.1.0/24. The next hop address to reach this network would be 10.0.2.1. A route will also need to be defined to network 10.0.4.0/24, with a next hop address of 10.0.3.2.
- Router C needs static routes defined to networks 10.0.1.0/24 and 10.0.2.0/24. In both cases, the next hop address will be 10.0.3.1.

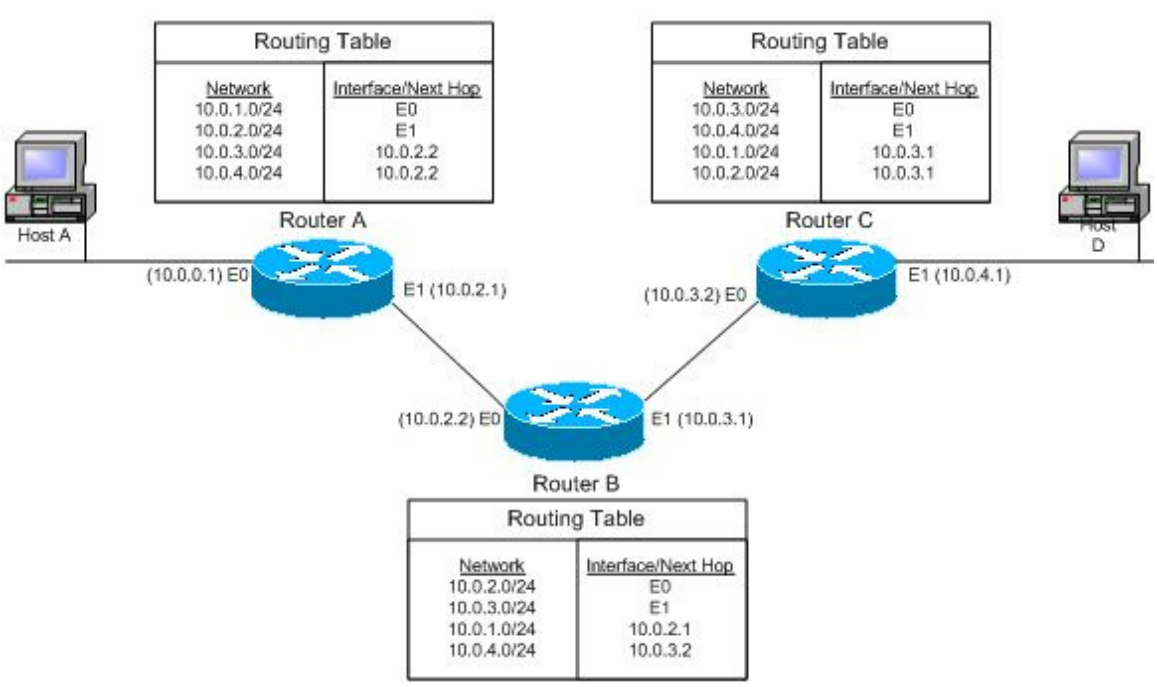


Figure 8-7: Network with complete routing table entries.

Notice that by adding just two static routing entries on each router, we end up with a fully routed network. Obviously, if there were more networks involved, additional routing table entries would be required. To get a feel for exactly what is happening on this network, let's assume that Host A (from Figure 7) wishes to communicate with Host D. I'm not going to point out every single detail of the communications process this time, but all the same things that happened in our original example do happen again here.

1. In this example, Host A wishes to communicate with Host C. Host A will create an IP packet with a source address of 10.0.1.99 and a destination address of 10.0.4.101. Since Host C is remote, the packet will be framed, with the destination MAC address being that of Router A's E0 interface.
2. Once the frame reaches Router A, it will calculate the CRC, strip off the framing, and pass the packet up to the Network Layer. In consulting its routing table, Router A will find that the next hop address to reach network 10.0.4.0 is 10.0.2.2. Router A will decrement the IP packet's TTL by 1, and will reframe the packet. In this case, the source MAC address will be MAC address of Router A interface E1 and the destination MAC address will be that of Router B interface E0. The MAC address of Router B interface E0 will be found with an ARP request.
3. Once the frame reaches Router B, a similar process occurs. The CRC will be calculated, the framing will be stripped off, and the packet will be passed to the Network Layer. In consulting its routing table, Router B will find that the next hop address to reach network 10.0.4.0 is 10.0.3.2. Router B will again decrement the IP packet's TTL by 1, and will reframe the packet. In this case, the source MAC address will be MAC address of Router B interface E1, and the destination MAC address will be that of Router C interface E0.
4. The frame is now forwarded to Router C interface E0. Router C will calculate the CRC, strip off the framing, and pass the packet up to the Network Layer. Here it will consult its routing table, and will see that network 10.0.4.0 is directly connected on interface E1. Router C will also decrement the TTL by 1, and then reframe the packet. This time, the source MAC address will be Router C's E1 interface, and the destination MAC address

will be that of Host D, which will have been found by an ARP request. The frame is then forwarded to Host D.

5. Upon receipt at Host D, the frame is processed, and ultimately the data it contains makes its way to the appropriate application. The reply from Host D to Host A undergoes the same process, in reverse.

In a nutshell, simple routing is no more complex than this example. If we had wanted to, we could have configured all three routers with a dynamic routing protocol, and this would have allowed them to exchange routing table information without our intervention. While this may immediately sound like a great idea, there are some downsides, as we'll look at a little later in the chapter. For now, it's most important that you understand the purpose of a routing table, how basic routing decisions are made, what happens when IP packets cross a router, and the constant re-framing that occurs on a packet's journey.

Understanding and Configuring Static Routing

In our second example we looked at configuring routing statically. Static routing really involves nothing more than telling a router about a non-connected network, and the next-hop address to reach it (or local interface on which to forward the packet). While it may be a little more work than using dynamic routing, static routes are a quick and effective way to configure routing, especially on small networks.

On larger internetworks that consist of many routers and networks, static routing will probably not be the best option. This is because of what transpires when a router or network becomes unreachable – with static routing, a router won't be able to do anything to find another path through the internetwork. With dynamic routing, when a path becomes unavailable, a router will find out about it from other routers, and will attempt to use a different path, if one exists. That's not to say that dynamic routing protocols are the solution to all of life's routing problems. In fact, they can sometimes cause just as many problems as they solve, as we'll see a little later in this chapter.

Even before our router is configured with static routes, it has a routing table. Each routed protocol (like IP or IPX) maintains its own routing table. To view the IP routing table on a Cisco router, use the **show ip route** command, as shown below.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
C    10.0.0.0/8 is directly connected, Serial0
C    192.168.1.0/24 is directly connected, Ethernet0
```

Before any static (or dynamic) routes exist, a routing table will show entries for only directly connected networks. In this case, the output of **show ip route** shows two connected networks, 10.0.0.0/8 and 192.168.1.0/24. In both cases, the fact that they are directly connected is clear according to the message following the network ID. However, the letter "C" that precedes the entries also identifies the networks as being directly connected. The codes shown at the

beginning of the command output describe how a router knows about a network in its routing table.

Adding a static route isn't terribly difficult at all. The command to add a static IP route is **ip route**. The complete syntax of the command is:

ip route [destination network] [mask] [next hop address OR exit interface] [administrative distance] [permanent]

Not all of the options listed above need to be entered. In this case, we'll enter only the destination network, subnet mask, and next-hop address information. We'll discuss the other options shortly. Let's assume that our network is configured as shown in Figure 8-8. In it, RouterA is directly connected to networks 192.168.1.0/24 and 10.0.0.0/8. We'll need to add a static route on RouterA, telling it that network 172.16.0.0/12 can be reached via the next hop address of 10.0.0.2.

Figure 8-8: Router A and Router B interconnecting 3 networks.

```
RouterA#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#ip route 172.16.0.0 255.240.0.0 10.0.0.2
RouterA(config)#^Z
RouterA#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
C    10.0.0.0/8 is directly connected, Serial0
C    192.168.1.0/24 is directly connected, Ethernet0
S    172.16.0.0/12 [1/0] via 10.0.0.2
```

Notice that a new entry has been added to the routing table, and is preceded by an "S". This designates the route as static. In this case, network 172.16.0.0/12 is accessible via the next-hop address, which is interface S0 on Router B - 10.0.0.2. If you're setting up static routes and you notice that the route just added doesn't appear after issuing the **show ip route** command, it likely means that the next-hop address you specified cannot be contacted. In order for the route to appear in the table, the next-hop address must be accessible. If you want to add a static route and have it appear in the routing table regardless of whether it's available, add the **permanent** keyword to the end of the **ip route** command, as shown below.

```
RouterA(config)#ip route 172.16.0.0 255.240.0.0 10.0.0.2 permanent
RouterA(config)#^Z
RouterA#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
```

```
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
S    172.16.0.0/12 [1/0] via 10.0.0.2
C    192.168.1.0/24 is directly connected, Ethernet0
```

In the example above, I unplugged the S0 interface on RouterB. Notice how network 10.0.0.0/8 no longer appears as directly connected, since the address 10.0.0.2 cannot be reached. However, the use of the **permanent** keyword keeps the new entry for network 172.16.0.0/12 in the routing table, even though the next hop address is unavailable.

After adding the static route to network 172.16.0.0/12 on RouterA, the next step is to add a static route to network 192.168.1.0/24 on RouterB.

```
RouterB#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#ip route 192.168.1.0 255.255.255.0 10.0.0.1
RouterB(config)#^Z
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
C    10.0.0.0/8 is directly connected, Serial0
C    172.16.0.0/12 is directly connected, Ethernet0
S    192.168.1.0/24 [1/0] via 10.0.0.1
```

In order to remove a static route, use the “no” version of the **ip route** command, followed by the network address and subnet mask of the route you wish to remove. For example, to remove the static route to network 192.168.0.0/24 from RouterB, you would enter:

```
RouterB(config)#no ip route 192.168.0.0 255.255.255.0
```

Recall that another option for the ip route command allows you to add what is known as an administrative distance. An administrative distance is simply a number that designates how reliable or “trustworthy” the information about a particular route source is considered to be. For example, a directly connected interface is always considered the most trustworthy, and has an administrative distance of 0. A static route is also considered trustworthy, since an administrator will have manually defined it, after all. Static routes are assigned an administrative distance of 1 by default. The lower an administrative distance, the more trustworthy a route is considered to be. Consider the routing table entry below. The administrative distance of this route is 1, as shown by the value in the first portion of the square brackets.

```
S    192.168.1.0/24 [1/0] via 10.0.0.1
```

Routes learned from dynamic routing protocols are also assigned default administrative distances. But how are administrative distances used? Well, assume that an administrator has defined a static route to a network, and a route to the same network is also learned from a routing protocol. In cases where this happens, the router will “keep” or use the entry with the lower administrative distance. By default, a static route will always beat an entry learned dynamically. Table 8-1 provides a listing of the default administrative distances associated with routes learned in different ways.

Table 8-1: Default Administrative Distances.

Routing Protocol or Source	Default Administrative Distance
Directly connected interface	0
Static Route	1
EIGRP	90
IGRP	100
OSPF	110
RIP	120
Unknown	255

To set the administrative distance of a static route to 50, use the ip route command as follows:

```
RouterB(config)#ip route 192.168.0.0 255.255.255.0 10.0.0.1 50
RouterB(config)#^Z
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
C    10.0.0.0/8 is directly connected, Serial0
C    172.16.0.0/12 is directly connected, Ethernet0
S    192.168.1.0/24 [50/0] via 10.0.0.1
```

While static routing may be a great deal of work to configure and maintain, there are two scenarios where static routing will commonly be used. The first is on “stub” networks, a network from which there is only one exit router to the rest of an internetwork. The second is in demand-dial routing environments, which are looked at in more detail in Chapter 11.

Tip: When a destination network with the same cost is learned from different methods (such as static routing or different routing protocols), only the entry with the lowest administrative distance will appear in the routing table.

Introduction to Dynamic Routing Protocols

In the real world, and especially in larger environments, companies usually rely on dynamic routing protocols to keep routing tables updated. Quite simply, it is easier to configure all routers to use a dynamic routing protocol and have them learn about unconnected networks from other routers, rather than statically defining a route to each and every network on every router. Dynamic routing protocols also provide a degree of fault tolerance. If a network or router fails on a network using dynamic routing protocols, other routers will find out about it, either through updates from neighboring routers, or via the absence of updates messages. In contrast, with only static routes defined, neighboring routers do not exchange information. When a network or router fails in a static routing environment, other routers do nothing to compensate for the failure.

Certainly dynamic routing protocols have their advantages, but they also have drawbacks. For one, the update messages that are passed between routers running dynamic routing protocols add traffic to the network. Secondly, some routing protocols also increase resource usage on the router, since they calculate the best route to a network from updates received. Finally, it takes time for routers to find out about a network that is unavailable, and for all routers to have consistent routing table information. Once all routers have knowledge of all available networks, the network is referred to as being converged.

Not all routing protocols are equal in these respects. Some converge faster, but use up more of a router's processing resources. Others go easy on resource usage, but cause more network traffic. Some make routing decisions based on very basic information, while others take into account a variety of factors. Using a routing protocol almost always involves some type of tradeoff, and deciding which works "best" will largely depend on the characteristics of the network you are dealing with. Routing protocols generally fall into two main categories – distance vector and link state.

Distance Vector Routing Protocols

Distance vector routing protocols create their routing tables based on the networks to which they are directly connected, and the information passed to them by neighboring routers in the form of routing table updates. The term "distance-vector" actually describes the way in which these protocols work. The "vector" is simply a combination of two pieces of information that describe how to get to a network – the direction, and the distance.

For example, a router may find out that a previously unknown network is accessible via its E0 interface (the direction), with a metric of 2 hops (the distance). A metric is simply the piece – or pieces – of information that a router uses in determining which path is the "best" in reaching a destination network. In the case of a protocol like RIP, the metric is the number of routers that need to be crossed in getting to a destination, also called the hop count. In other distance-vector protocols, like IGRP, the default metric is a calculation based on network bandwidth and delay. While we'll look at both shortly, for now it's enough to say that a metric is used by a routing protocol to find the best path to reach a destination network.

Distance-vector protocols work on a principle that is sometimes called "routing by rumor". This is because a router running a distance-vector protocol learns about other networks based on the information provided by neighboring routers. In other words, the knowledge it obtains is not first-hand. The way in which a router learns from neighboring routers is by routing table updates. At certain intervals, a router sends out its complete routing table, which will be received by other routers running the same routing protocol. These other routers look at the information contained in the routing table update, and add routes to their own tables for networks they didn't previously know about.

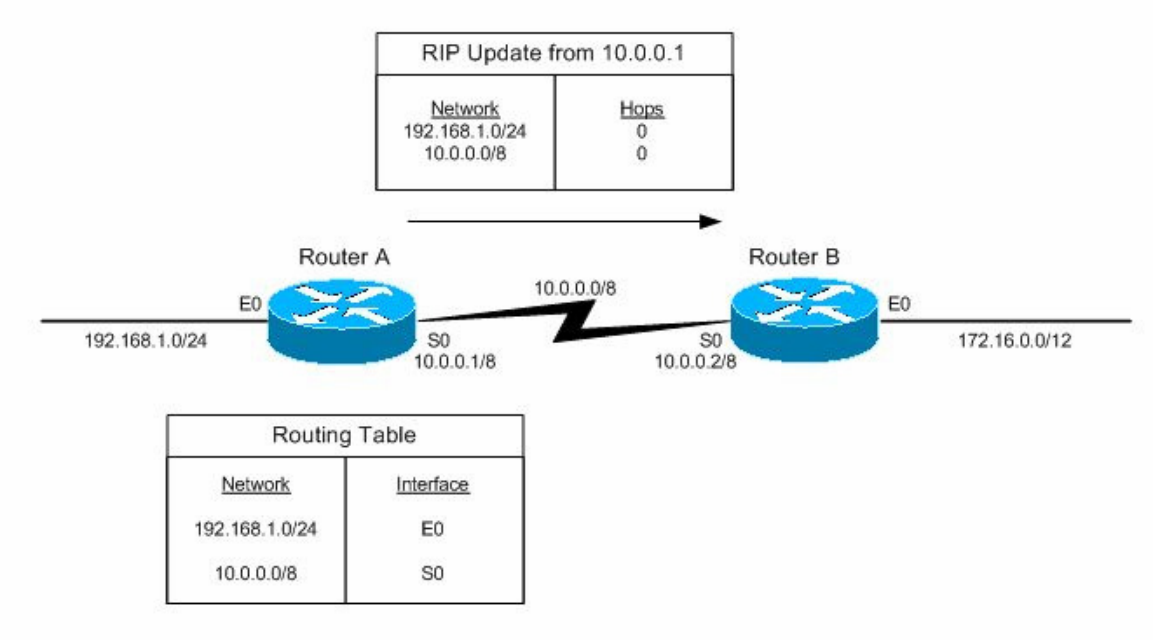


Figure 8-9: RIP update broadcast from Router A.

Consider Figure 8-9, in which Router A sends out its routing table, which is ultimately received by Router B. Router A knows about networks 192.168.1.0 and 10.0.0.0. Router B doesn't yet have an entry for network 192.168.1.0, so it adds it to its routing table with a hop count of 1, as shown in Figure 10. When Router B needs to reach network 192.168.1.0, it will forward the packets to Router A. In the same way, Router B will also send out updates to Router A. Through these, Router A will learn that network 172.16.0.0 is available via router B with a hop count of 1. Recall that the hop count is the number of routers that need to be traversed in order to reach a network. Because Router B only needs to go through one other router to reach network 172.16.0.0, the hop count is 1. Hop count is the metric used by RIP. In other words, RIP will choose the best path to a network based on the lowest number of hops.

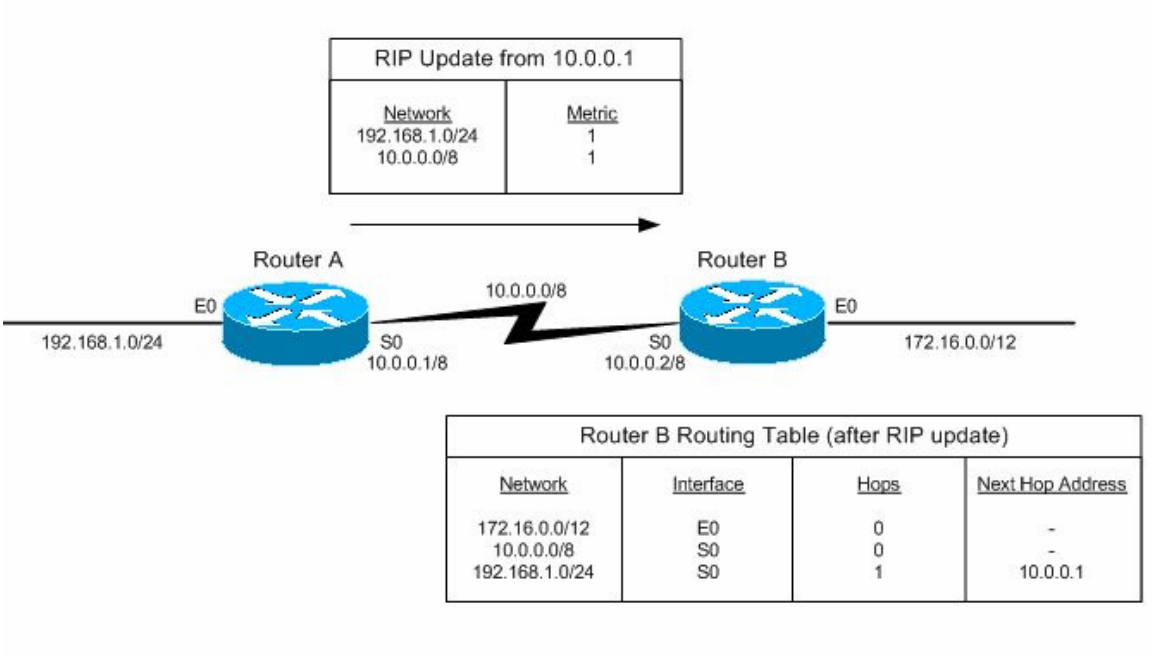


Figure 8-10: Routing table on Router B after RIP update.

If Router C (running the same distance vector routing protocol) were added to our network as shown in Figure 8-11, it would also begin exchanging updates with its neighbors. In this case, it would send a routing table update to Router B, letting it know about network 192.168.2.0. Router B will add this to its routing table with a hop count of 1. In Router B's next update to Router A, this network would also be included in the table. Router A would thus add it to its routing table, with a hop count of 2. Once all the updates were complete, and all routers knew about all networks, the network is considered to have converged. At that point, the routing tables would appear as shown in Figure 8-12.

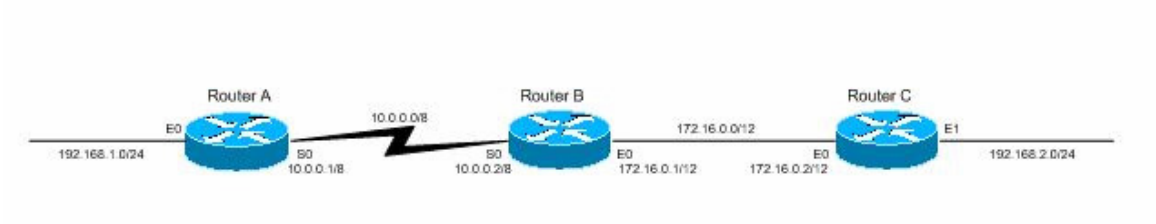


Figure 8-11: Router C and network 192.168.2.0 added to the existing network.

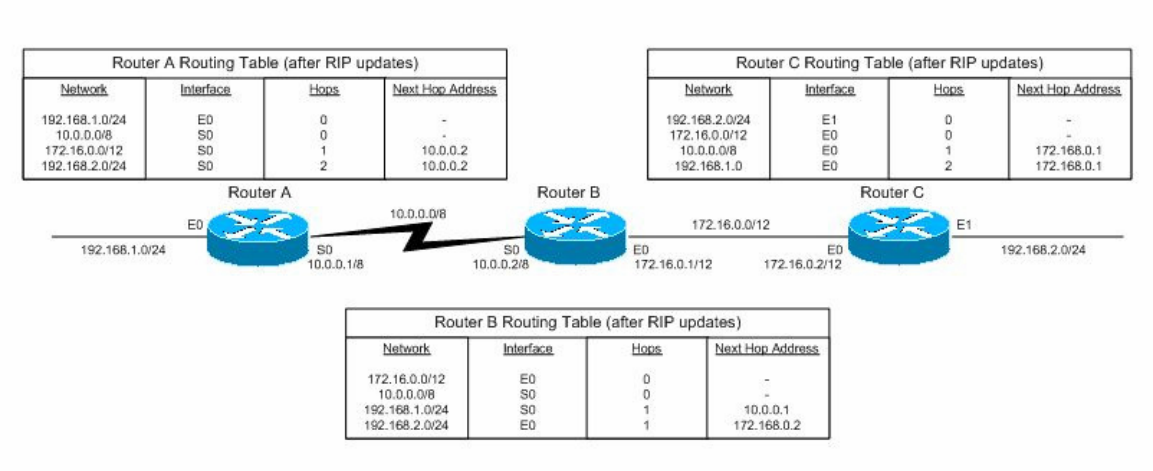


Figure 8-12: Routing tables after the network has converged.

Multiple Paths

On any internetwork, it is possible that more than one path to a destination may exist, with the same distance. For example, consider Figure 8-13. This network consists of three networks. From Router A, network 172.16.0.0 can be reached in two possible ways – both via Router B and Router C. In both cases, the hop count to network 172.16.0.0 is identical – 1 hop. A distance vector protocol like RIP decides which route to take based on hop count, but can also load-balance over routes with the same hop-count in a round-robin fashion. In this case, some packets will be sent to network 172.16.0.0 via Router B, and some via Router C. It's worth remembering, however, that hop count is the only metric used by RIP. For example, the link between A and C is only 64kbps, while the link between A and B is a T3 line. Even though AB is a much faster link than AC, Router A still sees both paths as equal. Such is how RIP sees the world.

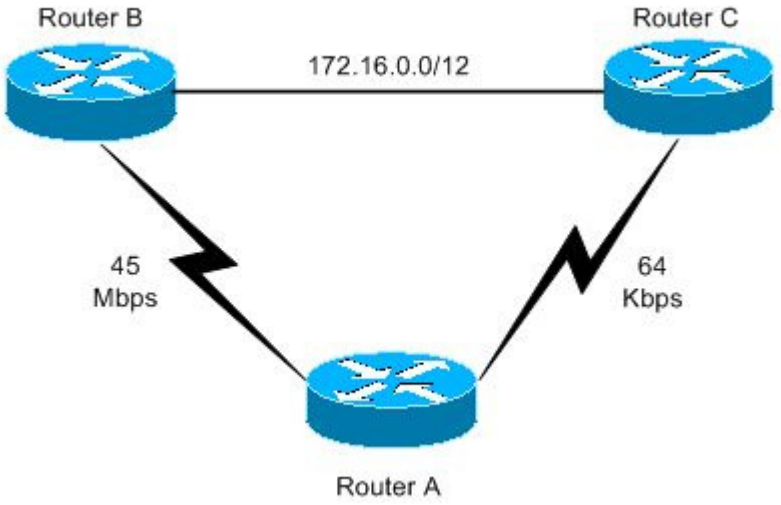


Figure 8-13: Network 172.16.0.0 is available to Router A via two paths, both with a hop count of 1.

Routing Loops

One of the main issues with distance-vector routing protocols is that they are susceptible to routing loops – a direct result of their slow convergence times. A routing loop can occur in the distance vector world because of the way routers exchange information. For example, let's say that we have a network as shown in Figure 8-14. Three routers exist in this example, connecting a total of four networks. We'll begin with a network that is fully converged – that is, all routers are aware of all networks, as shown in the diagram. On a fully converged network, everything works well.

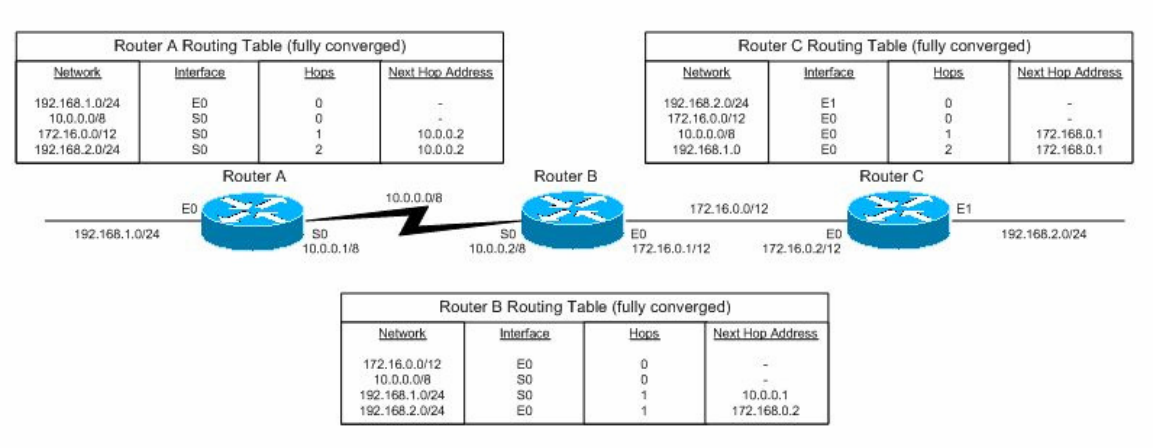


Figure 8-14: Fully converged network, prior to routing loop.

Routing loops become a potential issue when our network experiences a problem. For example, imagine that network 192.168.2.0 experiences a failure – maybe the switch it was connected to malfunctioned, or somebody simply disconnected the cable. At any rate, Router C recognizes that network 192.168.2.0 is unavailable, and passes this information to Router B in its next routing table update. Once the update arrives, Router B removes the entry for network 192.168.2.0 from its routing table. So far, things are going well. However, there is one little problem – Router A is also sending out routing table updates, and is telling Router B that network 192.168.0.0 is available through it, with a hop count of two, as shown in Figure 8-15. See a problem developing?

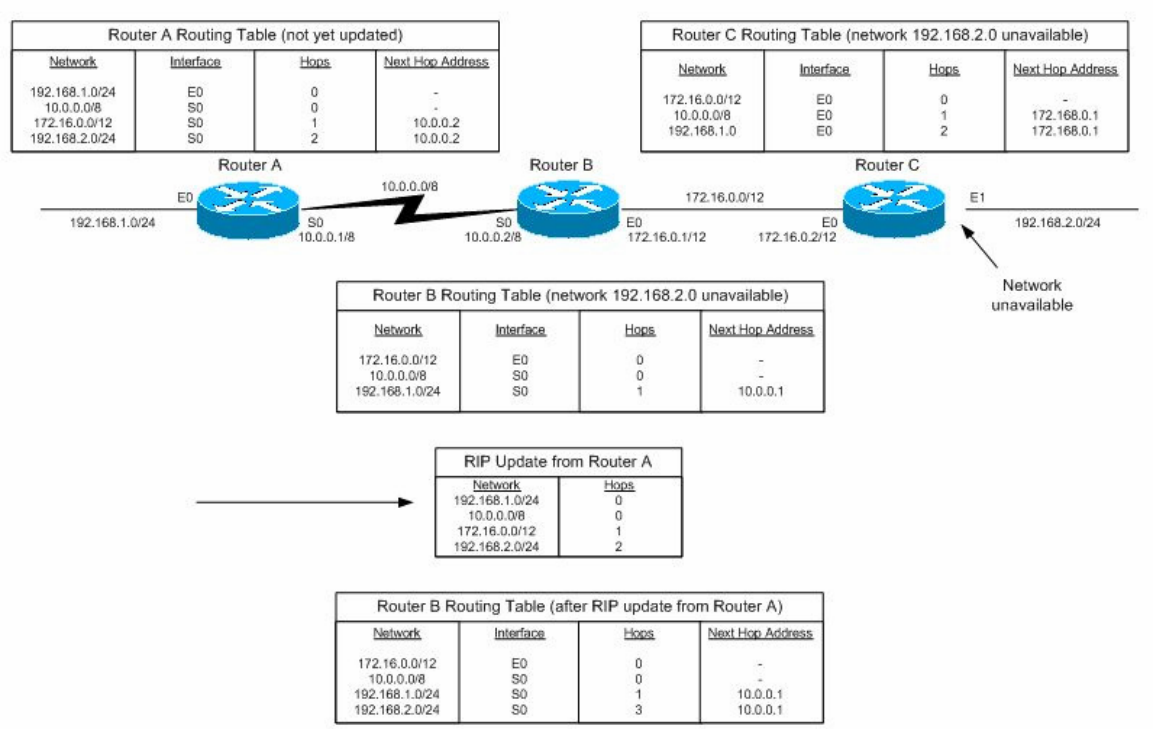


Figure 8-15: Update from Router A creates a routing loop between Routers A and B.

Now we have a network where Router B thinks it can get to network 192.168.2.0 via Router A. Without anything else occurring, think about what happens here. When Router B gets a packet destined for network 192.168.2.0, it will send it to Router A. Router A will look at the packet, and will send it back to Router B – that is the direction of network 192.168.2.0 as far as Router A is concerned, after all. The packet will actually end up being passed back and forth forever and ever. This is certainly not a very comforting thought.

The problem just described is a directly related to slow convergence. Router C did its job in getting information about the unavailable network to Router B, but unfortunately Router A also sent an update to Router B prior to the “correct” information arriving at Router A. On a larger network, the problem would be even worse. Remember that distance vector routing protocols are not terribly discerning – they simply trust their neighbors to provide them with correct information. Thankfully, distance vector routing protocols take a number of steps to try to avoid the routing loops just described.

Avoiding Routing Loops

Distance vector routing protocols use four main techniques in attempting to prevent routing loops. These include:

- **Defining Infinity**. Instead of allowing a packet to just loop around an internetwork endlessly, distance-vector protocols define what is considered to be infinity by specifying a maximum hop count. For example, RIP considers any route that takes more than 15 hops to reach to be unreachable. Remember that TTL value that gets decremented by 1 each time a router is crossed? Once the TTL reaches 0, the packet is discarded. While

this technique doesn't eliminate routing loops on its own, it does ensure that a packet doesn't loop around an internetwork forever.

- **Split Horizon**. In looking at the routing loop example, you should have noticed that Router A ultimately let Router B know about a route to network 172.16.0.0, even though it originally learned that route from Router B. Split horizon is a rule that specifies that a router can never send information about a route back to the router that originally supplied the information. Had split horizon been used in our example, Router A would not have included information about network 172.16.0.0 in its update to Router B.
- **Route Poisoning**. Usually used in conjunction with split horizon, route poisoning involves explicitly poisoning a routing table entry for an unreachable network. In our example, once Router C learned that network 172.16.0.0 was unavailable it would have immediately poisoned the route to that network by setting its hop count to the routing protocol's infinity value. In the case of RIP, that would mean a hop count of 16.
- **Triggered Updates**. Obviously the interval between routing table updates is part of the problem that leads to routing loops. Instead of relying on regular periodic updates, distance vector protocols will send out a triggered update when a metric to reach a network increases. In our example, Router C would immediately send out an update about network 172.16.0.0 not being available to Router B, who would then immediately send out an update to Router A. While triggered updates cause a little more network traffic in the short term, they go a long way towards faster convergence on a distance-vector network.
- **Holddowns**. Holddowns are a technique used to ensure that a route recently removed or changed is not reinstated by a routing table update from another router. Even with the use of triggered updates, it will still take some time for the information to reach all routers on the network. As such, there is still the possibility that a not-yet-current routing table update will attempt to reinstate a route that is unreachable. Holddowns make a router wait a period of time before accepting an update for a network whose status or metric has recently changed, giving a network the opportunity to converge.

Tip: Remember that split horizon prevents a router from sending information about a network back to the router that originally supplied the route.

Link State and Hybrid Routing Protocols

Link State Protocols

Link State Protocols work differently than distance-vector protocols. Where a distance-vector protocol periodically sends out its entire routing table to neighboring routers, a link state protocol instead builds a topology database that give it perspective on finding the best or "shortest" loop-free path to other networks. A link state protocol will usually build 3 databases. The first database is known as the neighbor or adjacency database, and keeps track of other routers on directly connected networks. The second database is the link state or "topology" database. This database keeps track of the "state" of the links on other routers on the network. Link state routers periodically send out what are known as "hello" messages, which are sent to neighboring routers as a type of keepalive message. An OSPF router will also periodically send out link state advertisements (LSAs), which are flooded across an internetwork. These messages contain information on the router's active links, its IP address, subnet mask, and the routers it knows about. The information stored in the topology database is ultimately used to calculate the shortest path to a destination network. It is also used to create the final database, the routing table.

Although the flooding of these LSA messages sounds dangerous, in reality their scope is limited, as we'll see when we look at OSPF. Ultimately, a link state router stores the complete topology of the internetwork in its database built on first-hand knowledge rather than the rumors passed by neighbor. Another efficiency is found in link state protocols because they don't actually broadcast out their routing tables, or take nearly as long to converge. On the downside, link state protocols typically involve more planning to implement than their distance vector alternatives. Common examples of link state protocols include Open Shortest Path First (OSPF) and Netware Link State Protocol (NLSP). Both of these will be looked at in more detail later in the chapter.

Hybrid Protocols

Some protocols can't be so clearly defined as either distance-vector or link state. A great example is Cisco's Enhanced IGRP (EIGRP), which is more accurately defined as a hybrid between the two. While distance vector protocols typically rely on the broadcast of complete routing tables, EIGRP instead only sends out specific updates when the metric to a route changes. Like link state protocols, EIGRP also maintains a database of neighboring routers and the network topology.

Routing Information Protocol

The Routing Information Protocol (RIP) is a simple distance-vector protocol. In order to avoid confusion, you should be aware that two versions of RIP exist – RIP version 1 (RIP), and RIP version 2 (RIPv2). For the purpose of the CCNA exam, you need to be familiar with RIP version 1, which is what we'll look at in this section. RIPv2 will be looked at a little later in the chapter.

The first thing that needs to be made clear is that RIP is a classful protocol. What that means is that RIP will always assume the class of an address according to the standard Class A, B, and C designations. Ultimately, this means that RIP is expecting that every single host in a network is using the same subnet mask. This does not mean that RIP won't work on a subnetted network; only that all of the subnet masks are assumed to be the same. Techniques like VLSM won't work with RIP, mainly because when RIP sends out its routing table updates, it doesn't include any subnet mask information.

Recall that RIP uses hop count as its metric in determining the best route to a network. Remember that RIP will always take the path with the fewest hops, regardless of factors like bandwidth, delay, or otherwise.

The maximum diameter of a RIP network is 15 hops. That means that a maximum of 15 routers can be traversed between a source and destination network before a network is considered unreachable. At the 16th hop (which RIP considers to be "infinity"), the TTL reaches 0, and a packet is discarded. Obviously this makes RIP a poor choice for very large internetworks.

RIP is also incredibly chatty. By default, a RIP router will broadcast out its complete routing table every 30 seconds, regardless of whether anything has changed. That's not terribly efficient, and causes a great deal of unnecessary traffic. Remember that broadcasts go to all hosts in a broadcast domain, meaning that even regular computers will have to process RIP packets to some degree before discarding them. RIP may not be pretty, but it is simple.

There are a few different timers that you should be familiar with on a RIP network. There include:

- **Route Update Timer**. The route update timer controls how often a router will broadcast routing table updates. As mentioned, a RIP router will broadcast its complete routing table every 30 seconds by default.
- **Route Timeout Timer**. The route timeout timer specifies the amount of time that will pass before a router will mark a network as unavailable, and is set to three times the update interval (180 seconds) by default. For example, if a router doesn't hear about network 172.16.0.0 from any other router for 180 seconds, it will mark the route as invalid. After marking a route as invalid, the information will be sent to other routers via a triggered update.
- **Route Holddown Timer**. The route holddown timer specifies the length of the holddown timer that will be used when RIP receives information about an unreachable routing table entry. The default holddown timer is 180 seconds.
- **Route Flush Timer**. After marking a route as invalid, a RIP router will not immediately remove the route from its routing table. Instead, it will wait until the flush timer (sometimes called the garbage collection interval) has expired. This gives the router time to let other routers know about the invalid route before removing the entry from its routing table. The default flush timer is 240 seconds.

While RIP may not be the most efficient or effective routing protocol for use on large networks, it still does the job. A big reason why RIP is so popular is because of how easy it is to set up. With as few as two commands, you can have your router fully configured for RIP.

Configuring RIP Routing

In this section we're going to configure a simple network with two routers to run RIP. Our network in consists of three subnets, 10.0.10.0/24, 10.0.20.0/24, and 10.0.30.0/24, as shown in Figure 8-16. Our goal is to ultimately have Router A learn about network 10.0.30.0 from Router B, and Router B learn about network 10.0.10.0 from Router A using RIP.

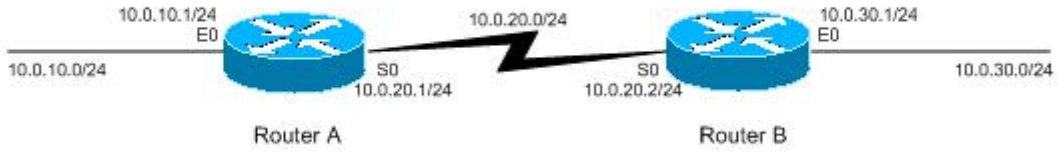


Figure 8-16: Network configuration for distance vector protocol configuration examples.

If you're trying to configure a router to use RIP to exchange routing table information, a first important step is to remove any static routes that you may have defined. Remember those administrative distances that we looked at earlier? If you have a static route defined on Router A that provides information on how to get to network 10.0.30.0, any information that Router A receives about network 10.0.30.0 via RIP will be ignored. A static route has an administrative distance of 1, while RIP's administrative distance is 120. When a route with a higher administrative distance is received, it is ignored, since a more trustworthy routing table entry already exists.

The command to turn on RIP routing is simple – from global configuration mode, simply issue the command **router rip**, as shown below.

```
RouterA(config)#router rip
RouterA(config-router)#
```

Notice how the prompt changes. At the most basic level, the **router rip** command makes this router capable of sending and receiving RIP updates. However, in order for this router to send out any information of use, we have to tell it which network(s) to advertise. In this case, the network we want to advertise is 10.0.0.0. You don't need to specify either the subnet mask or specific subnet addresses – because RIP is classful, it will automatically assume that you meant all networks starting with 10. In order to make RIP announce all of the 10.0.0.0 subnets, enter the **network 10.0.0.0** command, as shown below.

```
RouterA(config-router)#network 10.0.0.0
RouterA(config-router)#
```

That's literally all it takes to set up RIP. Of course, we'll also need RIP configured on Router B, which involves following the same steps.

```
RouterB(config)#router rip
RouterB(config-router)#network 10.0.0.0
```

In order to confirm that RIP is properly exchanging information between our routers, let's take a look at the routing tables on Router A.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
10.0.0.0/24 is subnetted, 3 subnets
C       10.0.10.0 is directly connected, Ethernet0
R       10.0.30.0 [120/1] via 10.0.20.2, 00:00:22, Serial0
C       10.0.20.0 is directly connected, Serial0
```

Notice in the routing table above that Router A learned about network 10.0.30.0 from RIP, as designated by the R at the beginning of the entry. The administrative distance is 120, and the number of hops to reach the network is 1, as designated by the entry [120/1]. Finally, the next-hop address is 10.0.20.2, which is accessible via interface Serial0.

An easy way to test whether RIP is working on both routers is a simple ping. If Router A can successfully ping IP address 10.0.30.1, it means that Router B also has a route (learned from RIP) back to network 10.0.10.0.

```
RouterA#ping 10.0.30.1
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echos to 10.0.30.1, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/38/48 ms
```

Remember that many routing protocols are also able to load-balance across multiple paths. RIP can load balance traffic across up to 6 paths, as long as the hop counts to the network in question are equal.

Monitoring RIP

Monitoring RIP

Once RIP is up and running, it can largely be left alone. However, there are a number of commands that you should be familiar with in order to gain information about the status of RIP, or for troubleshooting purposes.

Of these commands, the most basic and useful is the **show ip protocols** command. The command will provide you with information about the interfaces on which RIP is configured, the sources of routing information, and the timer values configured.

```
RouterA#sh ip protocols
Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 26 seconds
Invalid after 180 seconds, hold down 180, flushed after 240
Outgoing update filter list for all interfaces is
Incoming update filter list for all interfaces is
Redistributing: rip
Default version control: send version 1, receive any version
Interface        Send  Recv   Key-chain
Ethernet0        1     1 2
Serial0          1     1 2
Routing for Networks:
10.0.0.0
Routing Information Sources:
Gateway         Distance      Last Update
10.0.20.2            120      00:00:19
Distance: (default is 120)
```

Notice that this router is receiving updates from address 10.0.20.2, the near interface on Router B.

In order to gain more information about the actual contents of the RIP traffic moving between systems, use the **debug ip rip** command. This particular command is like a toggle switch – debugging information that shows the contents of RIP updates sent and received will continue to appear on-screen until you turn it off with the **no debug ip rip** command.

```
RouterA#debug ip rip
RIP protocol debugging is on
22:48:34: RIP: received v1 update from 10.0.20.2 on Serial0
22:48:34:      10.0.30.0 in 1 hops
22:48:38: RIP: sending v1 update to 255.255.255.255 via Ethernet0 (10.0.10.1)
22:48:38:      subnet 10.0.20.0, metric 1
22:48:38: RIP: sending v1 update to 255.255.255.255 via Serial0 (10.0.20.1)
22:48:38:      subnet 10.0.20.0, metric 1
RouterA#no debug ip rip
```

Note that the **debug ip rip** command shows the RIP updates both sent and received on a particular interface, along with associated subnet and metric information. Remember than a metric of 16 suggests an unreachable network in RIP.

Although you don't need to know how to do this for the exam, it's always nice to know how to change the default timers used by a protocol. The command to do so is **timers basic**, and is issued from the routing protocol configuration level. One important note here – if you do decide to

change the timers on one router, you should also change the timers on every router. Otherwise, you'll be dealing with one very unpredictable network potentially susceptible to routing loops.

```
RouterA(config)#router rip
RouterA(config-router)#timers basic ?
<0-4294967295>  Interval between updates
RouterA(config-router)#timers basic 30 ?
<1-4294967295>  Invalid
RouterA(config-router)#timers basic 30 180 ?
<0-4294967295>  Holddown
RouterA(config-router)#timers basic 30 180 180 ?
<1-4294967295>  Flush
RouterA(config-router)#timers basic 30 180 180 240
```

I used the help function to walk through the **timers basic** command step by step to show you the order of the entries – update, invalid, holddown, and flush. Ultimately, the same command can be used to set IGRP timers.

Interior Gateway Routing Protocol (IGRP)

The Interior Gateway Routing Protocol (IGRP) is another distance vector protocol commonly used on Cisco networks. Developed by Cisco, IGRP is proprietary and provides a number of advantages over RIP on a network comprised of Cisco routers. Firstly, the maximum hop count allowed by IGRP is much greater than RIP – up to 255 hops are supported, although by default IGRP is set to allow only 100. Secondly, where RIP uses only the very basic metric of hop count to make routing decisions, IGRP actually supports a variety of metrics including bandwidth, delay, reliability, load, and MTU. The default metrics used by IGRP are bandwidth and delay.

Tip: Remember that the default metrics used by IGRP are bandwidth and delay. These values are used to calculate what is known as the **composite metric**.

The metrics used by IGRP almost immediately make it a better choice routing protocol than RIP, since decisions can be made based on the most efficient path to a network rather than the one with the fewest hops. IGRP uses what is known as the composite metric in determining the best path to a network. The composite metric is actually a calculation that takes into account both the minimum bandwidth between a router and a destination, and the delay value of an interface. While you don't actually need to know how the composite metric is calculated, I think it's still worth knowing where the numbers come from. Ultimately, the path with the lowest composite metric will be used by IGRP. To view the bandwidth and delay numbers used in the calculation of the composite metric, use the **show interface** command followed by the interface that the update is received on.

```
RouterA#show int s0
Serial0 is up, line protocol is up
Hardware is HD64570
Description: WAN link Toronto-Montreal
Internet address is 10.0.20.1/24
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
```

Like RIP, IGRP routers also periodically broadcast their routing tables to exchange information with neighboring routers. The intervals involved in the exchanges are:

- **Route Update Timer**. The route update timer controls how often a router will send out routing table updates. An IGRP router will send out updates every 90 seconds by default.
- **Route Invalid Timer**. The route invalid timer specifies the amount of time that will pass before an IGRP router will mark a network as unavailable. By default this is set to three times the update timer, or 270 seconds.
- **Route Holddown Timer**. The route holddown timer specifies the length of the holddown timer that will be used when IGRP receives information about an unreachable routing table entry. The default holddown timer for IGRP is 3 times the update interval plus 10 seconds, or 280 seconds total.
- **Route Flush Timer**. The amount of time that will pass before IGRP completely removes an entry from the routing table. By default, this is set to seven times the update interval, or 630 seconds.

Configuring IGRP

The configuration of IGRP is only slightly different than that of RIP. The major difference is that IGRP routers are made part of what is known as an autonomous system (AS), a grouping that defines routers that should exchange routing tables. For example, if a router is made part of AS 100, it will only exchange IGRP routing information with other routers that are part of IGRP AS 100. Many IGRP autonomous systems can be defined within an internetwork, allowing you a more granular level of control over which routers exchange routing table information with one another.

Similar to RIP, you must also specify the networks to be included in IGRP updates. In this configuration, we are going to add IGRP AS 100 to our existing RIP network. This will also help to show what happens when two routing protocols are used on the same network. To review the network, look back at Figure 16. To add IGRP routing, use the **router igrp** command, followed by the autonomous system number.

```
RouterA#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#router igrp 100
RouterA(config-router)#network 10.0.0.0
```

Of course, IGRP also needs to be enabled on Router B:

```
RouterB#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#router igrp 100
RouterB(config-router)#network 10.0.0.0
```

A look at the routing table on Router A shows that that network 10.0.30.0 has been added, but via IGRP instead of RIP.

```
RouterA#show ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is not set
10.0.0.0/24 is subnetted, 3 subnets
C       10.0.10.0 is directly connected, Ethernet0
```

```
I      10.0.30.0 [100/8576] via 10.0.20.2, 00:00:01, Serial0
C       10.0.20.0 is directly connected, Serial0
```

Notice that the listing for network 10.0.30.0 begins with an I, which designates that the route was found using IGRP. The RIP route is no longer available. How does this happen? This is due to that fact that IGRP has a lower administrative distance than RIP. In cases where a route is found by a protocol with a lower administrative distance, the routing table will only show the more trustworthy route. In this case, both RIP and IGRP are announcing the same route, but IGRP's administrative distance of 100 beats RIP's administrative distance of 120. The administrative distance and composite metric are both listed in the entry as well – [100/8576], where 8576 is the composite metric.

Tip: All static and dynamic routing table entries are followed by a combination of the route's administrative distance and metric in square brackets. For example, the output [100/8576] represents the IGRP administrative distance (100) followed by the IGRP composite metric (8576).

Like RIP, IGRP is also capable of load balancing over up to 6 links, including links with uneven costs.

Monitoring IGRP

Information about IGRP messages can be obtained with the **debug ip igrp** command. Unlike RIP, the command requires additional information. Two types of debugging information are available for IGRP – information about IGRP events, and information about IGRP transactions. The **debug ip igrp events** command provides summary information about the IGRP updates being sent and received, as well as the number of routes that the update contains. The **debug ip igrp transaction** command actually shows the routes included in the update.

```
RouterA#debug ip igrp events
IGRP event debugging is on
23:18:27: IGRP: received update from 10.0.20.2 on Serial0
23:18:27: IGRP: Update contains 1 interior, 0 system, and 0 exterior routes.
23:18:27: IGRP: Total routes in update: 1
23:19:24: IGRP: sending update to 255.255.255.255 via Ethernet0 (10.0.10.1)
23:19:24: IGRP: Update contains 2 interior, 0 system, and 0 exterior routes.
23:19:24: IGRP: Total routes in update: 2
RouterA#no debug ip igrp events
IGRP event debugging is off
RouterA#debug ip igrp transactions
IGRP protocol debugging is on
23:37:20: IGRP: sending update to 255.255.255.255 via Ethernet0 (10.0.10.1)
23:37:20:      subnet 10.0.30.0, metric=8576
23:37:20:      subnet 10.0.20.0, metric=8476
23:37:20: IGRP: sending update to 255.255.255.255 via Serial0 (10.0.20.1)
23:37:20:      subnet 10.0.30.0, metric=1100
23:38:39: IGRP: received update from 10.0.20.2 on Serial0
23:38:39:      subnet 10.0.30.0, metric 8576 (neighbor 1100)
RouterA#no debug ip igrp transactions
IGRP protocol debugging is off
```

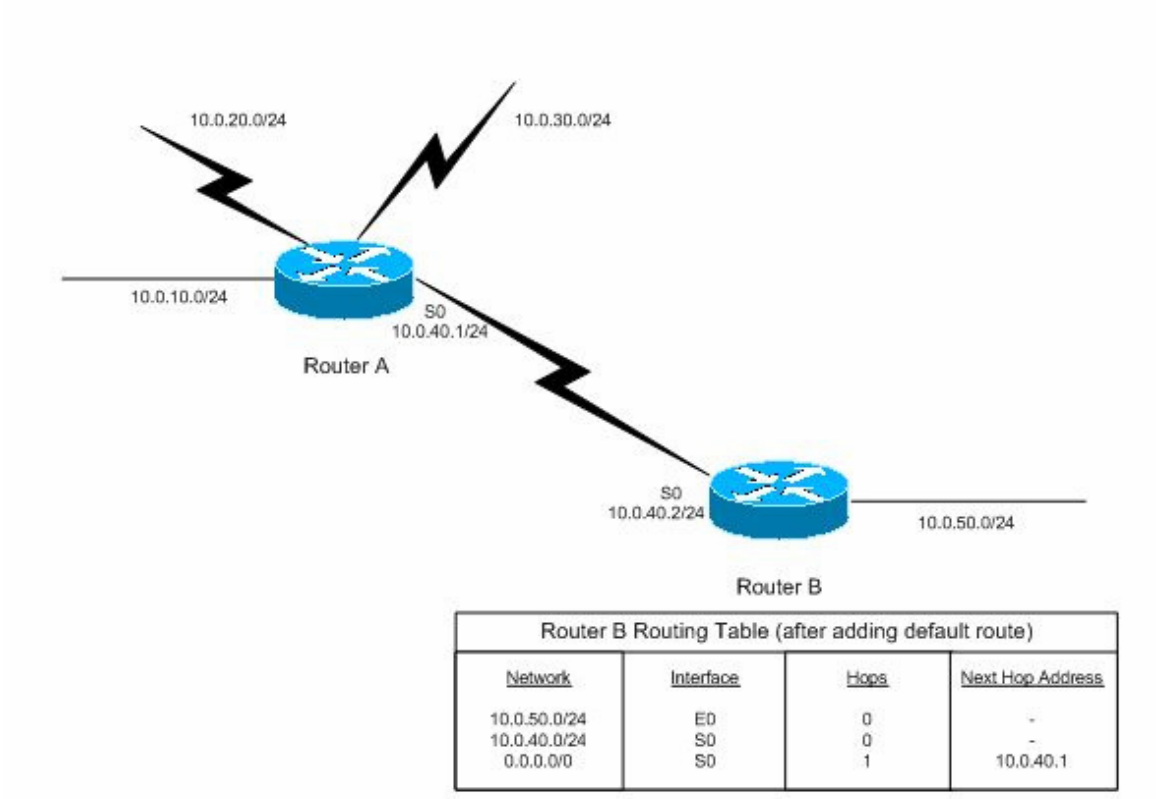
To get a quick overview on the status of the IP routing protocols running on your system, use the **show ip protocols** command. Notice that for Router A, we are presented with information about both of our configured protocols, RIP and IGRP.

```
RouterA#sh ip protocols
```

```
Routing Protocol is "rip"
Sending updates every 30 seconds, next due in 6 seconds
Invalid after 180 seconds, hold down 180, flushed after 240
Outgoing update filter list for all interfaces is
Incoming update filter list for all interfaces is
Redistributing: rip
Default version control: send version 1, receive any version
Interface        Send  Recv   Key-chain
Ethernet0        1     1 2
Serial0          1     1 2
Routing for Networks:
10.0.0.0
Routing Information Sources:
Gateway         Distance      Last Update
10.0.20.2            120      00:00:06
Distance: (default is 120)
Routing Protocol is "igrp 100"
Sending updates every 90 seconds, next due in 33 seconds
Invalid after 270 seconds, hold down 280, flushed after 630
Outgoing update filter list for all interfaces is
Incoming update filter list for all interfaces is
Default networks flagged in outgoing updates
Default networks accepted from incoming updates
IGRP metric weight K1=1, K2=0, K3=1, K4=0, K5=0
IGRP maximum hopcount 100
IGRP maximum metric variance 1
Redistributing: igrp 100
Routing for Networks:
10.0.0.0
Routing Information Sources:
Gateway         Distance      Last Update
10.0.20.2            100      00:01:07
Distance: (default is 100)
```

Default Routing

In the same way that you can set a default gateway on your desktops and servers, you can also set a default gateway of sorts on your Cisco router – the gateway of last resort. While the name is different, the function is the same. Remember that when a router comes across a packet destined for an unknown network, it automatically drops the packet. If we specify a gateway of last resort, a router will forward traffic for networks that it doesn't know about to the destination router address we specify.



Router B Routing Table (after adding default route)

Network	Interface	Hops	Next Hop Address
10.0.50.0/24	E0	0	-
10.0.40.0/24	S0	0	-
0.0.0.0/0	S0	1	10.0.40.1

Figure 8-17: Router B configured with a default route, also known as a gateway of last resort.

Consider Figure 8-17, in which Router A is connected to many different networks. Router B, on the other hand, is only connected to two networks. In order to allow Router B to get to all of the other networks shown, we would either need to configure a routing protocol (like RIP or IGRP), or define static routes to each network. In this particular scenario, it might actually be easier to use default routing to allow Router B to reach those three networks. Remember that Router B already knows about networks 10.0.40.0/24 and 10.0.50.0/24, since they are directly connected. In order to have Router B forward all traffic destined for other networks to Router A, we should configure a gateway of last resort on Router B. This involves a single routing table entry, with a destination network of 0.0.0.0. If you recall from Chapter 5, this address literally means “all networks”. In other words, we are saying that all other networks can be reached via Router A. When Router B attempts to route packets, it will first look in its routing table for the destination network. If it doesn’t find an entry, it will forward packets to the default route specified. To configure the gateway of last resort on Router B, use the **ip route** command, as shown below.

```
RouterB(config)#ip route 0.0.0.0 0.0.0.0 10.0.40.1
RouterB(config)#^Z
RouterB#sh ip route
Codes: C - connected, S - static, I - IGRP, R - RIP, M - mobile, B - BGP
D - EIGRP, EX - EIGRP external, O - OSPF, IA - OSPF inter area
N1 - OSPF NSSA external type 1, N2 - OSPF NSSA external type 2
E1 - OSPF external type 1, E2 - OSPF external type 2, E - EGP
i - IS-IS, L1 - IS-IS level-1, L2 - IS-IS level-2, * - candidate default
U - per-user static route, o - ODR
Gateway of last resort is 10.0.20.1 to network 0.0.0.0
10.0.0.0/24 is subnetted, 2 subnets
C       10.0.50.0 is directly connected, Ethernet0
```

```
C       10.0.40.0 is directly connected, Serial0
S*   0.0.0.0/0 [1/0] via 10.0.40.1
```

Notice that the static route entry used 0.0.0.0 for both the destination IP address and subnet mask, followed by the address of the next hop router. In this case, all traffic not destined for networks 10.0.50.0/24 and 10.0.40.0/24 will be forwarded to 10.0.40.1, interface S0 on Router A. The default route also appears in our routing tables as both the gateway of last result and as a static route. In this scenario, Router A would still need a routing table entry that defines how to reach network 10.0.50.0/24.

Default routes are commonly used when you are routing public IP addresses to the Internet. If you didn't define a default route, you would literally need to define a next hop address for every network on the Internet!

Routing IPX

Recall from Chapter 7 that IPX functionality is enabled on a router through the use of the **ipx routing** command. After issuing this command, you can assign IPX addresses to interfaces by providing a network number, and optionally (on Ethernet, Token Ring, or FDDI interfaces) specifying an encapsulation type.

Setting up IPX routing is exceptionally simple. What I didn't tell you in the last chapter is that after issuing the ipx routing command, your router is already running an IPX routing protocol – IPX RIP. Though much like the RIP protocol looked at previously, IPX RIP and IP RIP are not compatible – they are different routing protocols, even if they exhibit similar characteristics. For example, both define infinity as 16 hops, meaning that the maximum diameter of an IPX network running IPX RIP is 15 hops. IPX RIP update packets are broadcast every 60 seconds by default.

One major difference between the IP and IPX versions of RIP is the metric used. IPX RIP uses something referred to as a "tick" as its metric. A tick is the expected delay in reaching a network, and is defined as $1/18^{th}$ of a second. In cases where the tick count between two networks is equal, hop count is used as the tiebreaker.

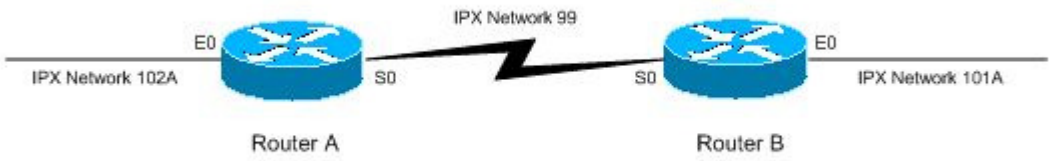


Figure 8-18: IPX network configuration.

For the purpose of illustrating IPX routing, we're going to configure our routers as per Figure 8-18. In this scenario we have two routers and three networks. The steps below outline the configuration of Router A.

```
RouterA#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#ipx routing
RouterA(config)#int s0
RouterA(config-if)#ipx network 99
RouterA(config-if)#int e0
RouterA(config-if)#ipx network 102A
```

The configuration of Router B is similar:

```
RouterB#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterB(config)#ipx routing
RouterB(config)#int s0
RouterB(config-if)#ipx network 99
RouterB(config-if)#int e0
RouterB(config-if)#ipx network 101A
```

With only the few steps shown, we have enabled IPX on the routers, given the appropriate interfaces IPX addresses, and even enabled IPX RIP routing – recall that it is enabled automatically as part of the **ipx routing** command. The next step is taking a look at our IPX routing tables. This is done using the **show ipx route** command, which is shown on Router A below.

```
RouterA#sh ipx route
Codes: C - Connected primary network,    c - Connected secondary network
S - Static, F - Floating static, L - Local (internal), W - IPXWAN
R - RIP, E - EIGRP, N - NLSP, X - External, A - Aggregate
s - seconds, u - uses, U - Per-user static
3 Total IPX routes. Up to 1 parallel paths and 16 hops allowed.
No default route known.
C        99 (HDLC),          Se0
C       102A (NOVELL-ETHER),  Et0
R       101A [07/01] via       99.0060.5cc4.f41b,   13s, Se0
```

Notice that an IPX routing table has a slightly different format, but still displays similar information to an IP routing table. It shows that networks 99 and 102A are directly connected, and also shows a route to network 101A via IPX address 99.0060.5cc4.f41b, which is the S0 interface on Router

B. To check and see whether Router A can route all the way to network 101A properly, we can use an IPX ping, specifying the IPX address of interface E0 on Router B.


```
Router#ping ipx 101a.0060.5cc4.f88b
Type escape sequence to abort.
Sending 5, 100-byte IPXcisco Echoes to 101a.0060.5cc4.f88b, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 32/35/36 ms
```

Since our IPX ping worked, Router A and Router B are both configured with complete information about the IPX internetwork.

If the fact that the **ipx routing** command enables IPX RIP routing wasn't enough of a surprise for you, you should also know that it enables the propagation of Service Advertisement Protocol updates as well. If you recall from Chapter 4, SAP advertisements are sent out by Netware servers periodically to announce services that they have available. By default, these announcements are sent out every 60 seconds. But why do routers forward them? Well, since they're broadcasts, SAP messages usually wouldn't leave the broadcast domain, which means that client systems wouldn't be able to find out about services offered by Netware servers on different networks.

On most networks running Netware, SAP isn't nearly as important as it once was. For example, on networks that run NDS, clients usually query the NDS directory to find out about servers as necessary. However, with older versions of Netware, clients relied on SAP announcements to find file servers, print servers, and so forth. Whether you need SAP advertisements propagated through your network (or to what extent), will usually depend upon the version of Netware you are running.

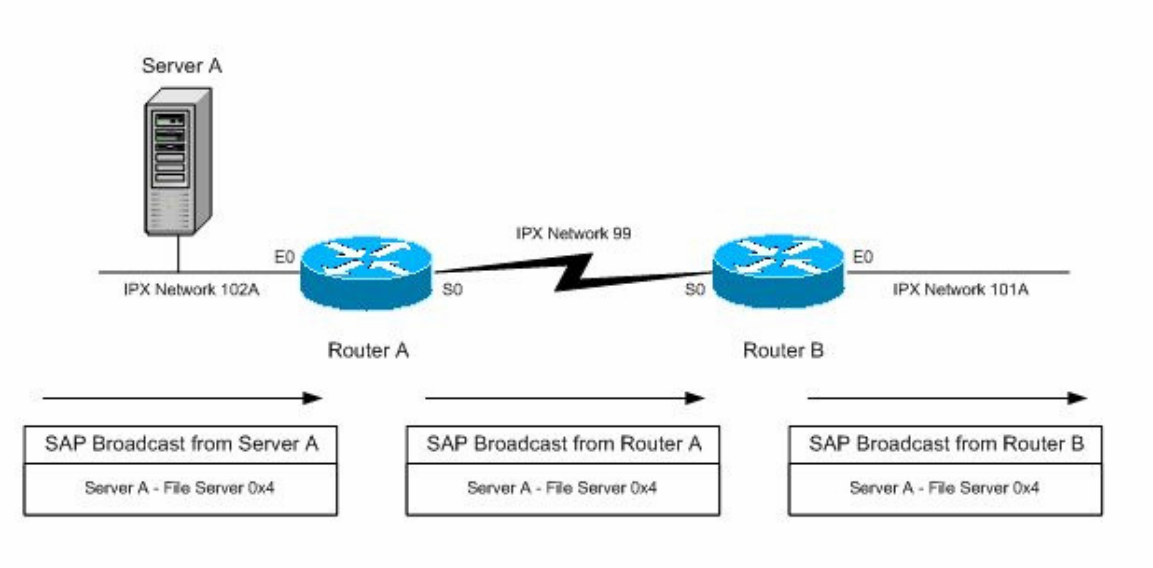


Figure 8-19: SAP broadcast sent by Server A, and then propagated by Routers A and B.

When Cisco routers forward SAP broadcasts to another router, they are simply making that router aware of servers available on other networks - all for the benefit of clients. Clients will find out about the servers on other networks via the SAP broadcasts sent out every 60 seconds by the router. Consider the network shown in Figure 8-19. In it, Server A is sending out SAP broadcasts that are ultimately forwarded to Router B. Router B makes clients on Network 101A aware of the fact that Server A exists and is offering file and services. The command to view the SAP table on a router is **show ipx servers**.

```
RouterB#show ipx servers
Codes: S - Static, P - Periodic, E - EIGRP, N - NLSP, H - Holddown, + = detail
1 Total IPX Servers
Table ordering is based on routing and server info
Type Name        Net      Address        Port Route Hops Itf
P    4 ServerA   FD819A6B.0000.0000.0001:0451 1/01  1    Et0
```

In SAP broadcasts, the services offered by servers are designated by hexadecimal identifiers – for example, the hex value of 4 represents a file server, while 7 represents a print server.

While the actual configuration of basic IPX routing is exceptionally simple, there are still a few commands that you'll want to have in your back pocket for the purpose of understanding a little more about the routing table updates and IPX traffic on your network. The **show ipx traffic** command provides you with information on the IPX traffic that has been sent

and received by a router. I've truncated the output to show only basic information provided by the command below.

```
RouterA#show ipx traffic
System Traffic for 0.0000.0000.0001 System-Name: RouterA
Rcvd:   996 total, 153 format errors, 0 checksum errors, 0 bad hop count,
932 packets pitched, 305 local destination, 0 multicast
Bcast:  301 received, 5367 sent
Sent:   5375 generated, 0 forwarded
0 encapsulation failed, 0 no route
SAP:    4 Total SAP requests, 0 Total SAP replies, 0 servers
4 SAP general requests, 0 ignored, 0 replies
0 SAP Get Nearest Server requests, 0 replies
0 SAP Nearest Name requests, 0 replies
0 SAP General Name requests, 0 replies
0 SAP advertisements received, 0 sent
0 SAP flash updates sent, 1 SAP format errors, last seen from 101A.00e0.
9872.90da
RIP:    5 RIP requests, 3 RIP replies, 2 routes
50 RIP advertisements received, 5354 sent
1 RIP flash updates sent, 0 RIP format errors
```

Like the commands used to view information on the status of IP RIP and IGRP, there are also debug commands to view IPX RIP and SAP status information. These commands include:

- **Debug ipx routing activity**. This command will show the full information on IPX RIP messages being broadcast from interfaces, including the routes contained in the update. The command debug ipx routing events shows only limited information, such as when an update has been sent or received.
- **Debug ipx sap activity**. This command displays detailed information of SAP broadcasts sent and received, including the server information they contain. Using the command debug ipx sap events provides only limited information on the actual updates.

To turn debugging off, you can use either the "no" version of these commands, or the **undebug all** command, which turns debugging off globally.

To view information on all the protocols that your router is currently routing, you can use the **show protocols** command. It will display not only whether routing for a protocol is enabled, but also the status and addresses of interfaces. The command provides a quick and effectively way of gathering important router status information in a single command.

```
RouterA#show protocols
Global values:
Internet Protocol routing is enabled
IPX routing is enabled
Ethernet0 is up, line protocol is up
Internet address is 10.0.10.1/24
IPX address is 102A.00e0.f751.d6af
Serial0 is up, line protocol is up
Internet address is 10.0.20.1/24
IPX address is 99.00e0.f751.d6af
```

Routing Information Protocol Version 2 (RIPv2)

RIPv2 is the newer, enhanced version of the RIP routing protocol, and is specified in RFC 1723. In many ways, this newer version is still very similar to its predecessor – it is still a distance vector protocol that uses hop count as its metric (the hop count limit is still 15), and still has a default administrative distance of 120. However, version 2 also introduces a number of features not found in the original version. Firstly, RIPv2 is classless; this means that it can be used on networks that employ variable-length subnet masking (VLSM). This is possible because RIPv2 includes the subnet mask associated with a destination network in its routing table updates. Where routing table updates were broadcast in RIP version 1, RIPv2 instead uses multicasts to send updates – specifically, a router will send updates to the multicast address 224.0.0.9.

RIPv2 is also capable of employing authentication between neighboring routers. This is another feature not found in the original version. You may be asking why authentication might be an issue when it comes to routing table updates. Remember that a RIPv1 update was no more than a broadcast, and that routers completely trust the information provided by neighbors. Now, imagine how easy it would be to anyone to set up another RIP router on a network (even versions of Windows can be configured as a RIP router), and begin broadcasting all sorts of incorrect routing table information! It certainly wouldn't take long to really mess up those RIP routing tables. With RIP version 2, authentication can be enabled on any router interface using either plain text or MD5 authentication. If authentication is enabled, a router will only accept updates from routers whose updates contain the correct authentication string.

RIP for IPv6 Networks (RIPng)

Although RIPv2 represents a significant improvement over the original version, RIPv2 is still a routing protocol used for IPv4 networks only. Because of this, a new version of RIP, referred to as RIPng or RIP version 3 has been developed in order to support this popular distance vector routing protocol on IPv6 networks. In case you're curious, the "ng" in RIPng stands for "next generation".

RIPng functions in a manner almost identical to RIPv2, though with a couple of key differences. The first is that instead of using IPv4 addresses in its update messages, RIPng uses IPv6 addresses and prefixes. The second change is that when a RIPng router needs to communicate with other RIPng routers, it uses a special multicast address (FF02::9) as the destination address.

As of this writing, RIPng was still not a finalized Internet standard. It is currently a proposed standard in the RFC process, but Cisco already supports the protocol in their IPv6 IOS images. For the CCDA exam, you should have a basic awareness of this new version, mainly why it needed to be developed in the first place.

Open Shortest Path First (OSPF) Routing Protocol

OSPF is a scalable, industry standard link state protocol used on IP networks and defined in RFC 2328. Because link state protocols build and maintain a topology database for a network based on first-hand knowledge rather than simply relying on the "hearsay" of neighbors, they tend to be more efficient in determining the most efficient route to a network. OSPF uses what is known as the Dijkstra algorithm to determine the shortest path between a router and a destination network. The metric used by OSPF to determine the best route to a network is interface cost, which is

calculated based on bandwidth. For those interested, the actual calculation is 10^8 divided by the interface bandwidth in bits per second, which you might remember can be found using the show interface command. For example, the cost associated with a T1 line would be 10^8/1544000, which equals 64.

Like RIPv2, OSPF is also classless, meaning that it supports the use of VLSM addressing. OSPF routers exchange information by flooding link state advertisement (LSA) packets throughout a network. These packets include information on the current state of the router's links, interfaces, and cost. Ultimately, this information is used by an OSPF router to build its adjacency, topology, and routing tables. Every 10 seconds, on a broadcast network, an OSPF router also sends out a "hello" message, letting adjacent routers know that it is still around.

While a simple distance vector protocol like RIP or IGRP may require little more than being enabled in order to make routing decisions, an OSPF implementation is usually designed according to a hierarchy, where different routers are given different roles. A proper OSPF design also helps to ensure that those LSA packets just mentioned don't overwhelm routers. Instead, an OSPF network is divided into what are known as areas, with a group of areas forming what is known as an OSPF autonomous system (AS). Figure 8-20 shows a network made up of 4 areas, all within autonomous system 100.

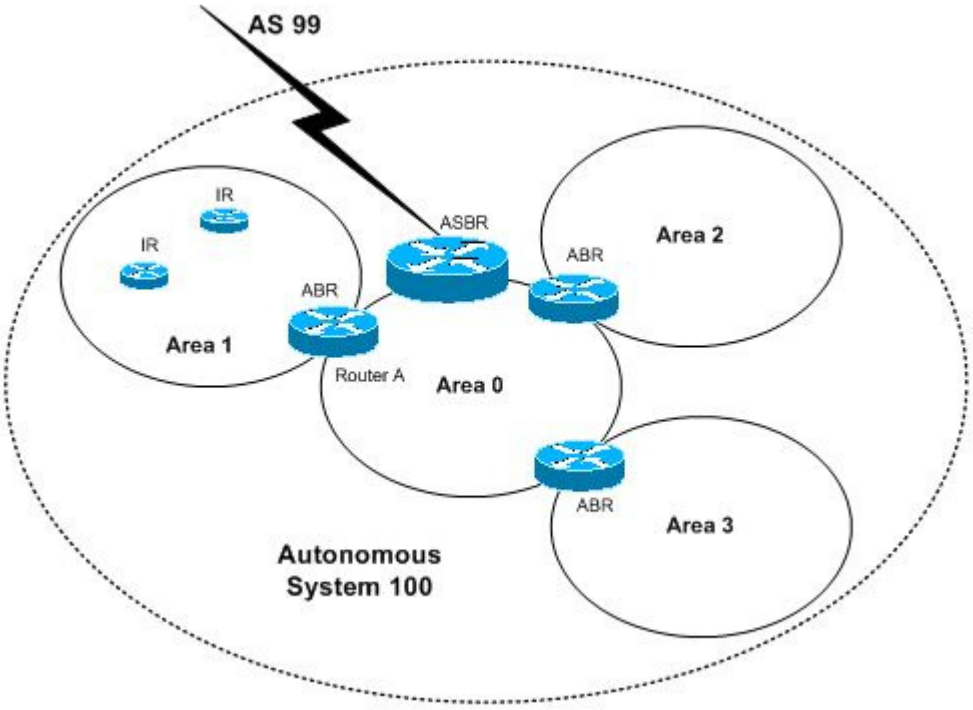


Figure 8-20: OSPF autonomous system including multiple areas, IRs, ABRs, and an ASBR.

It's easy to get confused by OSPF terms when learning them for the first time. The bullet points below outline all of the key terms that you should be familiar with.

- **Area**. An OSPF area is nothing more than a grouping of routers (running OSPF) amongst whom link state advertisements will be flooded. The idea is to try to reduce the number of LSAs that are sent throughout an internetwork by keeping the majority of LSAs within an area.

- **Backbone Area**. The backbone area should be the first area defined in an OSPF network, and is the area to which all other areas usually connect. The backbone area is known as Area 0. The backbone area is used to move OSPF routing information between areas.
- **Autonomous System**. An OSPF autonomous system is nothing more than a collection of OSPF areas amongst which OSPF routing information is shared. An autonomous system is assigned a number, similar to the one we used when looking at IGRP.
- **External Network**. Any networks running other routing protocols, or other OSPF autonomous systems, are considered external networks.

Routers within an Area exchange LSAs with one another to build their link state databases. All OSPF routers in the same area have the same link state database. The LSAs flooded within an area are known as Type 1 or router LSAs – they contain information on the links, interfaces, and costs for routers within an area. Any router whose interfaces only connect to a single area is known as an Internal Router (IR).

If you take another look at Figure 8-20, you'll notice that Router A connects to both Area 1 and Area 0. This makes Router A what is known as an Area Boundary Router (ABR). It connects to (at least) two areas, and maintains a link state database for both. The job of an ABR is important – it summarizes the routes within an area, and passes this information to other ABRs, who will then pass the information within their connected areas. The route summarization done by the ABRs helps to make routing tables much smaller than they would otherwise be – instead of passing information about 8 individual networks, the information might be summarized into a single routing table entry. For example, if an ABR had networks 192.168.8.0/24 through 192.168.15.0/24 configured within a connected area, it could summarize these routes via a single advertisement for network 192.168.8.0/21. For a review of how route summarization works, take a look back at the CIDR section of Chapter 5.

The final type of router in an OSPF implementation is an Autonomous System Boundary Router. The job of an ASBR is to act as a gateway to networks running other routing protocols, or different OSPF autonomous systems. This is often referred to as redistribution. For example, the ASBR in Figure 8-20 is connected to both AS 99 and AS 100. It would be the job of the ASBR to summarize the information learned about AS 100 and redistribute it into AS 99 and vice versa. This ultimately allows routers that are part of AS 99 to know about AS 100, without actually requiring any of the other routers within one AS to talk to routers in another AS directly. Ultimately, OSPF allows you to design your network hierarchically, and obtain a higher degree of routing efficiency.

OSPF for IPv6 Networks

In much the same way that a new version of RIP has been defined for use on IPv6 networks, a new version of OSPF is in the works as well. Referred to as either OSPF for IPv6 or OSPF version 3, this new version is fundamentally very similar to the IPv4 version of the protocol, but which a few changes. The most basic of these is the fact that the addressing portions of an LSA packet have been changed to use IPv6 addresses and prefixes, as you may well have guessed. Other major changes include OSPF running on a per-interface rather than a per-subnet basis, and the removal of all authentication functions from the protocol. The IPv6 authentication header now handles all authentication responsibilities instead.

For the purpose of the CCDA exam, Cisco doesn't expect you to know all of the inner workings of the new version of OSPF, mainly because it hasn't been fully standardized

yet (at the time of writing). OSPF version 3 is still only a proposed standard going through the RFC process. However, all signs point to it soon becoming the standard for running OSPF on IPv6 networks.

Enhanced Interior Gateway Protocol (EIGRP)

While IGRP might be a better solution than RIP when it comes to scalability, EIGRP takes things many steps further. First of all, EIGRP is classless, meaning that it supports the use of VLSM. Unlike IGRP, EIGRP supports the routing of multiple protocols, including IP, AppleTalk, and IPX. EIGRP is usually described as a hybrid protocol, meaning that it displays characteristics of both a distance vector and link state protocol.

EIGRP uses the same metrics as IGRP in making its routing decisions – bandwidth, delay, reliability, load, and MTU. The default metrics used are again the same, bandwidth and delay. However, for a more granular level of control, EIGRP multiplies each of the metrics by 256 before performing the calculation of the composite metric. EIGRP was designed to make much better use of bandwidth, and to allow routers to have a much better awareness of neighboring routers.

Instead of sending its entire routing table out at regular intervals, an EIGRP router instead sends out only partial updates, and even then, only when a route changes. Obviously this makes better use of the available network bandwidth. An EIGRP router also has a more complete view of the network than a typical distance vector protocol – it not only maintains its own routing table, but also keeps a copy of the routing tables of neighboring routers. When an EIGRP router cannot find a route to a network based on all the information it currently has, it sends out a query to other routers, which is propagated until a route is found.

EIGRP operates through the use of four key technologies:

- **Neighbor Discovery**. Similar to link state protocols, EIGRP routers also periodically send out “hello” packets, letting neighboring routers know that they are functioning and available. On a LAN or point-to-point links, these message are sent out as multicasts every 5 seconds. On a multipoint network (like Frame Relay) with speeds lower than T1, these packets are unicast every 60 seconds. As long as these “hello” packets are received, an EIGRP router assumes that its neighbors are available for the purpose of exchanging routing table information. If three “hello” periods pass without receiving a “hello” message, a router will consider its neighbor unavailable and make the necessary routing table changes. On a LAN, this can happen in as little as 15 seconds (3 times the “hello” message interval).
- **Reliable Transport Protocol**. The Reliable Transport Protocol is responsible for ensuring that EIGRP updates actually reach neighboring routers, in the correct order. EIGRP updates are sent out as multicasts to address 224.0.0.10. When a neighboring router receives an update, RTP requires that an acknowledgement be sent. This is different than many routing protocols, which send update traffic in a connectionless manner.
- **Diffusing Update Algorithm**. DUAL is the protocol used by EIGRP to ensure fast convergence and that the most efficient loop-free route advertised by neighbors is the one added to a router’s routing table. DUAL uses the lowest calculated metric to determine the best path to a destination, referred to as the feasible distance. Routers that advertise a lower metric to the destination than the feasible distance are known as feasible successors, and are ultimately used as the next hop router to which packets will

be sent. When a topology change occurs, an EIGRP router will use the route provided by the next most feasible successor as the next hop. In cases where all metrics are higher than the feasible distance, the EIGRP router must recompute the route.

- **Protocol-Dependent Modules**. Because it is capable of routing multiple protocols (IP, IPX, and AppleTalk), EIGRP implements what are known as protocol-dependent modules. For example, the IP EIGRP module will automatically redistribute IGRP routes into EIGRP and vice versa. Similarly, AppleTalk EIGRP will redistribute routes into and out of AppleTalk RTMP.

EIGRP offers greater flexibility, reliability, and better convergence times than a traditional distance-vector protocol. One limiting factor is that EIGRP is proprietary to Cisco – as such, EIGRP is limited to networks running Cisco equipment.

Netware Link State Protocol (NLSP)

NLSP is a link state protocol designed by Novell that addresses the limitations inherent in the distance vector IPX RIP protocol (such as the maximum hop count of 15), as well as the Service Advertisement Protocol (SAP). The idea was to create a more scalable, reliable, and efficient routing protocol for IPX networks that could replace both RIP and SAP. NLSP in many ways exhibits characteristics similar to those found in OSPF. However, the original design of NLSP was based on the ISO's Intermediate System to Intermediate System (IS-IS) link state routing protocol. The metric used by NLSP is a cost value based on the speed of a connected link.

Like OSPF, NLSP maintains topology, adjacency, and routing databases. The topology data is an NLSP router's map of the network, and the adjacency database stores information on known and working neighboring routers. The topology database is built from link state packets (LSPs) sent between routers, while the adjacency database is maintained based on the periodically multicast "hello" messages sent by NLSP routers. If a "hello" messages are not received from a neighbor, the adjacency is removed from the database, as the neighbor is assumed to not be available. NLSP routers only send out updates when routing (or SAP) information changes, rather than at periodic intervals. The maximum hop count on an NLSP network is 127.

NLSP uses a hierarchical routing model. While the methods used by NLSP may seem similar to those used by OSPF, in reality they are much more similar to those found in the IS-IS routing protocol. Three main concepts that you should be familiar with in an NLSP implementation are that of an area, a routing domain, and a global internetwork. The terms outlined below are shown in Figure 21.

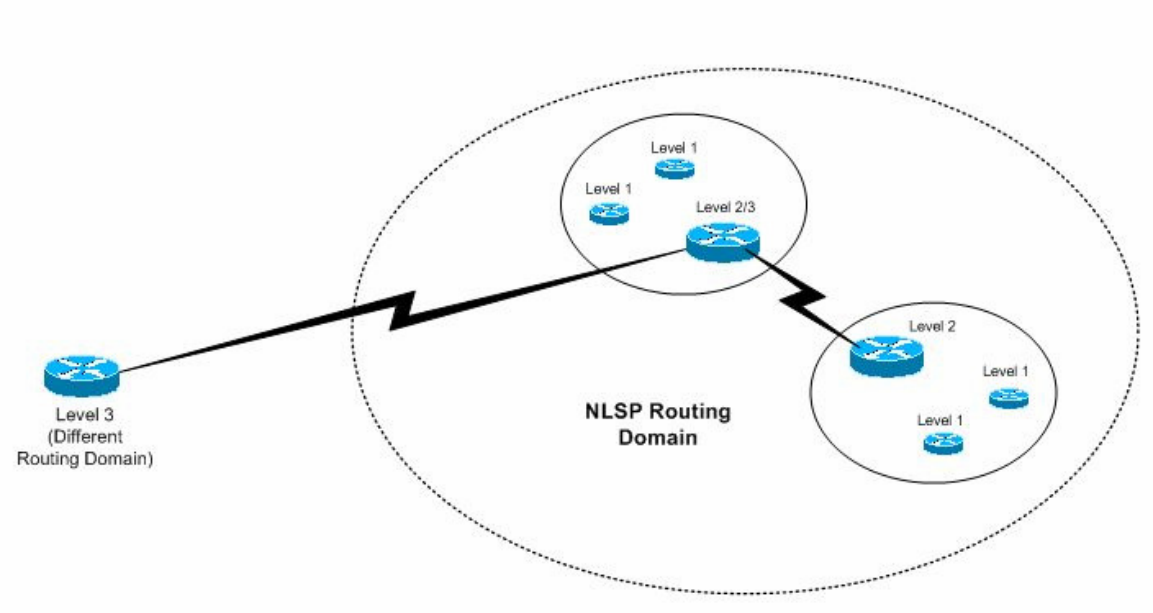


Figure 8-21: NLSP routing domain showing Level 1, 2, and 3 routers.

- **Area**. An NLSP area is similar to what you are familiar with from OSPF. All routers within an NLSP area have the same area number, and exchange LSPs with one another. Ultimately, all routers within an NLSP area have the same topology database. Routers within an area are known as Level 1 routers.
- **Routing Domain**. A routing domain is a collection of areas that all belong to the same organization. You can consider a routing domain to be similar to an autonomous system (AS). Level 2 routers are used to interconnect different areas within a routing domain, and only advertise the areas they are connected to, rather than their entire link state database.
- **Global Internetwork**. A global internetwork is what is formed when routing domains are interconnected, whether within or between organizations. When connecting routing domains to form a global internetwork, Level 3 routers are used, which behave similar to Level 2 routers in that they only announce their area information, rather than their entire link state database.

NLSP also provides better performance across WAN links, through the use of IPX header compression and the use of multicast addressing.

Routing Table Maintenance Protocol (RTMP)

The primary routing protocol used in AppleTalk environments in the distance-vector Routing Table Maintenance Protocol (RTMP). Like RIP, RTMP uses hop count as its metric, and supports a maximum of 15 hops. By default, RTMP updates are broadcast every 10 seconds.

The information included in an RTMP routing table includes the destination network address (or cable ranges), the interface on which packets should be forwarded, the next hop address, the hop count to the destination network, and whether the current state of a route is good, bad, or suspect.

AppleTalk Update-Based Routing Protocol (AURP)

The AppleTalk Update-Based Routing Protocol (AURP) is different than a traditional routing protocol in that its primary job is allowing physically distant AppleTalk routers to communicate over a TCP/IP-based WAN. When AURP is used, AppleTalk traffic doesn't need to be routing over the WAN separately from IP. Instead, a WAN can be limited to routing IP traffic only.

AURP is made up of two primary components – exterior routers, and AURP tunnels. An exterior router is one which functions as a normal AppleTalk router (running RTMP, for example) for local networks, and then connects to other AppleTalk routers across a TCP/IP WAN. Communication across the TCP/IP WAN is accomplished through the creation of an AURP tunnel, which ultimately connects exterior routers for the purpose of exchanging AppleTalk traffic between networks. The AppleTalk traffic is encapsulated in UDP datagrams for the purpose of transmission over the WAN. This is shown if Figure 8-22.

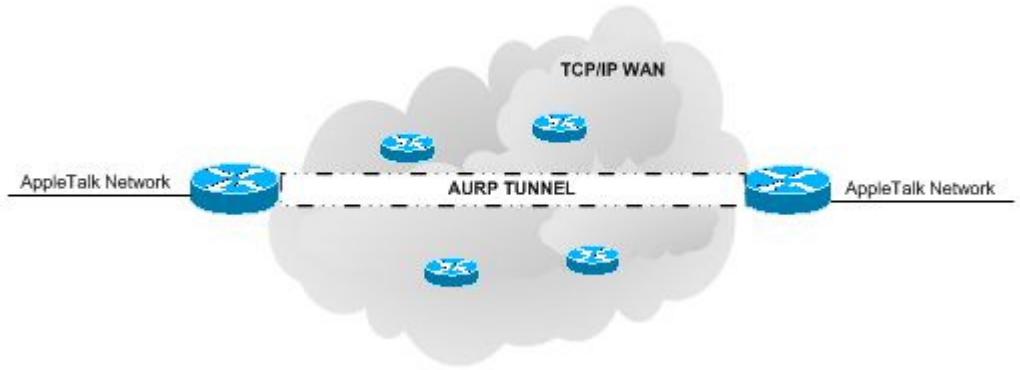


Figure 8-22: AURP Tunnel connecting AppleTalk networks across a TCP/IP WAN.

But why is AURP necessary? One reason is that RTMP limits a packet to 15 hops before a network is considered unreachable. On a very large network, this may not be sufficient. If AURP tunnels are used, an exterior router will consider another exterior router only one hop away, regardless of how many routers are actually traversed by the encapsulated packet. In that way, many hops can be crossed without the TTL in the AppleTalk packet being decremented by more than one. This ultimately circumvents the TTL restrictions enforced by RTMP. Also, AURP only sends out routing table updates when a route changes, rather than every 10 seconds, as is the default with RTMP.

Classful Versus Classless Routing Protocols

This chapter has contained a great deal of information about both classful and classless routing protocols. It is very important that you remember the differences between both, as well as the category into which the various IP routing protocols looked at in this chapter fall. You've already learned that classful routing protocols must use the same subnet mask consistently throughout a

network, a result of the fact that these protocols do not transmit subnet mask or network prefix information with their updates. However, classful routing protocols also fall prey to another limitation, namely the fact that all common networks must be must contiguous when a classful routing protocol is used.

What do I mean by contiguous? Well, consider the network illustrated in Figure 8-23. It consists of three networks total – 10.1.0.0/16, 10.2.0.0/16 and 192.168.0.0/16. However, you'll also notice that the only connection between the two networks starting with "10" is via the 192.168.0.0/16 network. In other words, these networks are not contiguous. This causes a problem when classful routing protocols are used, since classful routing protocols summarize major network information at network boundaries.

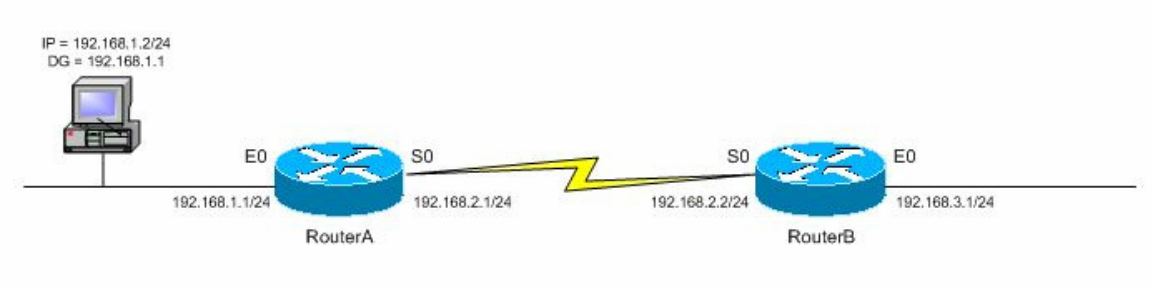


Figure 8-23: Eight networks summarized into a single routing protocol advertisement.

In this case, Routers A and B represent major network boundaries. Router A is connected to network 10.1.0.0/16 and network 192.168.0.0/16. Similarly, Router B is connected to network 10.2.0.0/16 and network 192.168.0.0/16. When Router A attempts sends an update using a classful routing protocol like RIP, it will be aware of the fact that as a boundary between two different major networks (10.0.0.0/8 and 192.168.0.0/16), it should advertised network 10.0.0.0/8 to Router B – a summarized route. In the same way, Router B would advertise network 10.0.0.0/8 to Router A. As you can see, neither router ends up with the correct information necessary to route to the discontiguous networks. In a situation like this, a classless protocol would need to be used, since subnet mask information would be provided with the updates.

Just for the sake of review, remember that OSPF, EIGRP, and RIPv2 are all considered classless protocols – they do transmit subnet mask information in their updates, can use variable-length subnet masks, and network need not necessarily be contiguous. Examples of classful protocols include RIPv1 and IGRP. These protocols must use the same subnet mask throughout the network since this information is not contained in updates, and networks must always be contiguous.

Route Summarization and Redistribution

Two concepts that you will definitely need to know for the CCDA exam are route summarization and route redistribution. While the technical details of their implementation are mainly associated with the CCNP core exams, a CCDA is expected have an awareness of both concepts, including their impact on network design.

Route summarization is something that you already have an awareness of having read Chapter 5. If you recall from our look at Classless Interdomain Routing (CIDR), it is

possible to represent many individual subnets or networks in a single routing table entry by allocating a custom subnet mask. Sometimes referred to as supernetting, route summarization simply involves collapsing a number of contiguous routing table entries into a single entry instead. This not only saves routing table space, but also makes routing more efficient.

In the world of routing protocols, route summarization refers to the process by which a protocol (like OSPF) will summarize multiple entries into a single routing table entry at a boundary on the network. For example, consider an internetwork where networks 10.0.8.0/24 through 10.0.15.0/24 all exist behind Router A. Instead of Router A advertising each of these eight networks to a neighboring router, it would make much more sense to summarize the routes into a single routing table entry, and forward information about one network only.

Networks 10.0.8.0/24 through 10.0.15.0/24 can be summarized into a single routing table entry or advertisement – 10.0.8.0/21. If you are having trouble remembering how I came up with the new network prefix of /21, I would strongly suggest reviewing the CIDR section of Chapter 5. For those looking for a quick refresher, eight networks need to be summarized in this example. By stealing 3 bits from the network portion of the existing subnet mask, I am capable of summarizing 8 addresses, since 2^3 equals 8.

Route redistribution is another concept that you will need to be familiar with. Under normal operation, most routing protocols follow a “ships in the night” method of operation – RIP routers talk to RIP routers, OSPF routers talk to OSPF routers, and never the twain shall meet. In other words, RIP and OSPF will not share information about the networks they are individually aware of by default. By the same token, if a router is configured for two different EIGRP autonomous systems (like EIGRP 99 and EIGRP 100) they will not share information either – they are separate and distinct processes.

However, there may also be cases where you have a router connected to one network running a routing protocol like EIGRP, while also connected to another network running OSPF. By default, these two routing protocols will not exchange information. However, it is possible to make the two routing protocols share information, if that is your goal. For example, it’s possible to “inject” the routes that OSPF has learned into the EIGRP routing process, or vice versa. So, if I were to redistribute OSPF information into EIGRP, EIGRP would then pass the information learned from OSPF to all other EIGRP routers. It’s worth noting that this process is one-way by default. In other words, redistributing OSPF into EIGRP does not redistribute what EIGRP has learned into OSPF automatically – both types of redistribution would need to be explicitly configured, which ultimately gives a network administrator a higher degree of control over the process.

Some protocols do redistribute between each other automatically. The best example of this is when a router is running both IGRP and EIGRP. If both routing protocols are given the same autonomous system number (for example IGRP 1000 and EIGRP 1000), then the two will redistribute information automatically.

Tip: Route summarization and route redistribution typically occur at the distribution layer of the Cisco Hierarchical Network Design model.

Layer 3 Switching

During your career in internetworking, you will continuously hear talk of equipment functioning at different “layers”. As a recurring theme since Chapter 1, you should now be aware that the layers being referred to are those in the OSI model. While the concept of a switch or a bridge as a Layer 2 device or a router as a Layer 3 device should now seem elementary, it’s easy to become confused by the mish-mash of marketing lingo that pervades the industry. What is a Layer 4

switch, for example? Well, it depends on whom you ask. Ultimately, vendors tend to use the different layers of the OSI model to represent intelligent decision-making features in their equipment. In some cases, as with Layer 3 switching, the term used represents a clearly defined and valid function. In others, like Layer 4 switching, what the term actually means can be a little less clear.

At the most basic level, the role of a Layer 3 switch is more or less identical to that of a router. Recall that a Layer 2 switch makes forwarding decisions based of the destination MAC address of a frame. In the same way, a Layer 3 switch is also capable of carrying out the functions of a router, making forwarding decisions based on the destination IP address of a packet. For all intents and purposes, a Layer 3 switch is basically a traditional Layer 2 switch that is also capable routing traffic between different subnets or networks. The big difference with a Layer 3 switch is usually speed, namely the speed at which it is capable of routing. Recall that Layer 2 switching is typically a much faster operation than routing, if only because there is less work involved in the forwarding process. With a Layer 3 switch, routing can often occur at close to the same forwarding rates as those associated with Layer 2 switching.

Note: The processes that are being looked at in this section relate to a multilayer switch (such as a Cisco Catalyst) also handling a routing function. Cisco routers also have ways of speeding up the process of switching packets between their interfaces more quickly. These processes are looked at later in this chapter.

For better or for worse, Layer 3 switching is implemented in many different ways by different vendors. Even in Cisco's line of Catalyst switches, many different methods are used depending upon the hardware series and model. The different methods used by Catalyst switches will be looked at shortly. For now, let's reexamine the process by which a traditional router forwards traffic.

Imagine a very simple network that consists of two hosts that are part of different VLANs on the same switch. Each VLAN is connected to a different port on a traditional router. Recall that even though the two hosts are connected to the same switch, routing is necessary in order for them to communicate, since they are members of different VLANs. Host A will create a packet listing itself as the source IP address, and Host B as the destination IP address. It will then frame the packet, listing itself as the source MAC address, and router interface E0 (in this example) as the destination MAC address. Once complete, the frame is forwarded across the network to the router.

When the frame arrives at the router's E0 interface, the CRC is calculated, the MAC framing is stripped away, and the packet is passed to the Network Layer. At this layer the router calculates the IP checksum, and then examines the routing table to determine where the packet should be forwarded next. After decrementing the packet's TTL by 1, the router reframes the packet with its E1 interface as the source MAC address, and the MAC address of Host B as the destination. Once complete, the packet is forwarded back to the switch, and ultimately to Host B. All this work for just one routed packet!

Ultimately, every routed packet is forwarded in this manner on a traditional IP network. Even though many packets may ultimately need to be forwarded between the same two hosts, this process must still be completed. As you'll see shortly, Layer 3 switching techniques go a long way towards reducing the amount of overhead associated with routing packets between networks.

To understand how a typical Layer 3 switch functions, imagine that the router from the previous example has been replaced by a Layer 3 switch. In this example, Host A is connected to one port on the switch, but is part of VLAN1. Host B is connected to another port on the same switch, but is part of VLAN2.

When the two hosts in this example attempt to communicate, the initial process is very similar. The exception is that the router interfaces that the hosts communicate with are virtual, or internal to the switch (in this example, the interfaces are designated using familiar names like E0 and E1). Host A will still forward the packet it has created to the router portion of the Layer 3 switch. The Layer 3 switch will still look in its routing table to determine where the packet should be forwarded next. In this case it will be forwarded out another virtual interface, and ultimately to the port where Host B resides. Note that in this case, the Layer 3 switch has still calculated the CRC, stripped away the framing, calculated the IP header checksum, determined where the packet should be forwarded next, reframed it, and sent it on its way. Some Catalyst switch models (such as a Catalyst 5000 with a Route Switch Module (RSM) installed) still forward all packets in this way. In other words, some Layer 3 switches simply add the traditional functions of a Cisco router within their hardware. The process is a little faster than using an external router, but typically not by much.

However, in other Catalyst switch models, the packet forwarding can occur at speeds closer to traditional Layer 2 switching. For example, some models support a “shortcut” feature, where after routing the first packet in the traditional manner, a Layer 3 switch gets smart and uses pattern matching (via specialized hardware) to speed up the process of forwarding additional packets. In this example, the switch would recognize that it passed a frame up to Layer 3, only to have it forwarded back to itself. Instead of sending all other packets in this session from Host A to Host B up to Layer 3, the switch would instead rewrite the frame header automatically at Layer 2, making it appear as though the packet has passed through the router. In fact, once the process is complete, it is impossible to distinguish a packet that has been forwarded in this manner from one that has been routed traditionally.

In order to better understand this process, consider another example using the same network. After the first packet in the stream between Hosts A and B has been routed the old fashioned way, the switch caches the appropriate information it needs to rewrite the packet. In this case, it knows that when it receives packets with the source MAC address of Host A, a destination MAC address of the virtual router interface, and the source and destination IP addresses of Hosts A and B respectively, it can rewrite the header using the shortcut information it has gathered by pattern matching. In this case, it will add new source and destination MAC addresses to the frame, decrement the TTL, recalculate the IP header checksum, and then switch the packet out the port that Host A is attached to – all without sending the packet up to Layer 3. When a Layer 3 switch functions in this manner, routing takes place at speeds closer to traditional Layer 2 switching.

Layer 3 Switching Hardware

In the previous section, I mentioned that different Catalyst switches perform routing functions in different ways. Before asking why all Catalyst switches don’t use the same techniques, it’s worth noting that different methods evolve over time, and some cost more to implement than others. In some cases, companies may simply wish to add routing capabilities to an existing investment, such as a Catalyst 5000. In others, the company might need the fastest routing performance possible across a campus network. In both cases, a realistic analysis of needs along with budget considerations will dictate the best solution. The following sections outline three different methods commonly used to integrate routing capabilities with different Catalyst switches.

The Route Switch Module (RSM)

The Cisco Catalyst 5000 series switches have long been a staple of many server rooms and wiring closets in the corporate world. Although originally only a Layer 2 switch, one method that

can be used to add routing functionality to these models is the addition of a Route Switch Module (RSM). An RSM is essentially a Cisco router on an add-in module that plugs directly into the backplane of a Catalyst 5000 switch. Just like a Cisco router, an RSM runs Cisco's IOS software, and is configured in a very similar manner. The main benefit of using an RSM is that it eliminates the need for an external router, thus speeding up the routing process. However, an RSM is still a router in the traditional sense. Although the fact that packets do not need to be passed to an external device makes the process a little faster, routing is still occurring using the IOS software, using traditional routing methods. As such, a Catalyst 5000 with an RSM installed can still be considered a Layer 3 switch, but will not provide the dramatic increases in speed that other methods listed here do. More than anything, the RSM provides convenience, adding routing capabilities to a Layer 2 switch.

The NetFlow Feature Card (NFFC)

For vastly improved routing performance on a Catalyst 5000 router, another alternative is to install a NetFlow Feature Card (NFFC). The NFFC is a specialized piece of hardware that acts as a pattern-matching engine for the purpose of rewriting the subsequent packets in a routed transmission at Layer 2. This results in vastly improved routing performance suitable for campus environments.

The technical details of Multilayer Switching (MLS) are required knowledge for the CCNP/CCDP exam "Building Cisco Multilayer Switched Networks". Fortunately, you are not responsible for knowing the operational details of MLS for the CCDA exam. However, there are a couple of concepts that you should be familiar with, namely the basic operation of MLS.

MLS uses three primary components in order to facilitate higher-speed Layer 3 switching performance. These are listed below, along with explanations of their purposes.

- **MLS Route Processor**. In a Layer 3 switch, the MLS Route Processor takes on the role of a router. Ultimately it makes routing decisions for the network, using the information stored in its routing table. Even when a packet is switched using a rewrite process at Layer 2, it is ultimately the Route Processor that made the initial decision on how this should occur.
- **MLS Switching Engine**. The MLS switching engine is simply a switch that includes an NFFC. This switch will build CAM table entries for the various MLS Route Processors that it knows about, and build cache entries for the shortcut switching methods discussed earlier.
- **Multilayer Switching Protocol**. The Multilayer Switching Protocol is a lightweight protocol that runs on an MLS Route Processor, allowing it to communicate with the MLS Switching Engine.

When a multilayer switch boots up, the Multilayer Switch Protocol on the MLS Route Processor sends hello packets to the NFFC, identifying VLANs and MAC addresses used by the router. As the NFFC forwards initial packets to the route processor (known as candidate packets) it creates a partial shortcut entry. When the packet is ultimately forwarded back to the NFFC from the router, the NFFC checks its shortcut table, notices a partial entry for the original packet that was forwarded to this router, and creates a full shortcut entry for the flow. In essence, the NFFC has noted that a packet forwarded to the route processor was passed back to it, and it will subsequently handle the inline rewrite of matching packets automatically at Layer 2 until the flow either times out (has not been used for a period of time) or the route processor lets it know of a topology change.

While this is a simplified view of what can be a very detailed process, it helps you to get the picture – an NFFC can significantly speed up the routing process at Layer 2 by rewriting packets, even though the ultimate routing decision is still made by the route processor at Layer 3.

Multilayer Switch Feature Card (MSFC)

In the case of a Catalyst 6000 or 6500 series switch, multilayer switching functions in much the same fashion manner as the example illustrated in the NFFC section just considered. However, there are some differences in terms of the hardware used. In the case of a Catalyst 6000 or 6500, three main hardware components are involved, as outlined below.

- **Catalyst Supervisor**. The Catalyst Supervisor contains the CPU and ASICs used to carry out Layer 2 switching functions.
- **Policy Feature Card (PFC)**. Much like an NFFC, a Policy Feature Card provides the "shortcut" caching services that allow routed packets to be written without each packet needing to be passed to the route processor. The PFC is typically provided as a daughtercard on the same board as the MSFC.
- **Multilayer Switch Feature Card (MSFC)**. The MSFC effectively provides the high-performance routing functions for a multilayer Catalyst 6000 or 6500. The MSFC route processor is another daughtercard on an MSFC module.

Tip: Remember that in a Layer 3 switch, a route processor still makes the forwarding decisions. The inline rewrites of subsequent packets are handled by the shortcuts created by an NFFC or PFC.

Layer 4 Switching

Now that you're familiar with Layer 3 switching, you're probably curious about what Layer 4 switching represents. Well, the answer isn't as difficult as you might have imagined. Quite simply, a Layer 4 switch is typically just a Layer 3 switch that is also capable of making decisions based on Layer 4 information. Layer 4 (the Transport Layer) carries information about the source and destination TCP and UDP ports in use, which generally represent unique applications. Because of this, a Layer 4 switch is capable of making forwarding decisions according to the applications in use. For example, an administrator might choose to prioritize VoIP traffic through the use of Quality of Service (QoS) features, granting VoIP applications more bandwidth. Conversely, the Layer 4 port information could also be used to route the packets from certain applications along a different path than other traffic. Ultimately, a Layer 4 switch gives administrators a higher level of control over how bandwidth is used within a network.

Tip: A current list of TCP and UDP port number assignments can always be found at http://www.iana.org/assignments/port-numbers.

Re-examining the Cisco Hierarchical Network Design Model

Now that you have an awareness of the purpose and features of Layer 3 switching, it's time to take a step back in time, and reexamine the Cisco hierarchical network design model. Back in Chapter 1 you learned about this model that serves as the basis for Cisco's network design methodologies. As presented in Chapter 1, the basic functions of the access, distribution, and

core layers have not changed. However, for the CCDA exam, it's important that you have a more realistic understanding of the types of hardware typically found at each layer.

Access Layer

You may recall that the primary function of the access layer is to provide workgroup connectivity to a network. It is at the access layer that collision domains are created by connecting end devices (like user PCs) into their own dedicated switch ports. At this layer, traditional Layer 2 switching is almost always used. Layer 2 switches are generally much less expensive, which is important especially considering the large number of devices that typically connected at the workgroup level.

The primary functions of the access layer include:

- Microsegmenting a VLAN into a larger number of smaller collision domains, providing increased bandwidth, and making full-duplex transmission possible when devices are connected to a dedicated switch port.
- Providing workgroup or departmental connectivity to the distribution layer.
- Extending security policies (like access lists) from the distribution layer and implementing features like MAC address security.
- Providing remote locations with access the corporate network via technologies like demand-dial routing or other WAN technologies.

Tip: Remember that in almost all cases, Layer 2 switching is used at access layer.

Distribution Layer

If you look back to Chapter 1, you'll notice that the design model illustrated used routers rather than switches at the distribution layer. While this was a common setup on networks of old, most companies now implement routing functions at the distribution layer using Layer 3 switches. Quite simply, these models tend to be more flexible (providing both switching and routing services) and are usually capable of much higher frame forwarding rates than traditional routers.

Some of the primary functions of the distribution layer include:

- Defining broadcast domains and routing between VLANs.
- Implementing network access lists, security, Quality of Service (QoS), firewall, and network address translation functions.
- Interconnecting and translating between different media types.
- Redistribution of routing protocols and the aggregation of addressing information using techniques like route summarization.
- Connecting the access layer to the network core/backbone.

Tip: Because the distribution layer needs to carry out a routing function, devices used here are almost always Layer 3 switches, although traditional routers are also sometimes used.

Core Layer

In our original look at the core layer back in Chapter 1, the devices used were Layer 2 switches. While this is a very common way to provide high-speed switching across the network backbone, it

is also not uncommon to find Layer 3 switches at the core layer, especially in campus environments.

Some of the primary functions of the core layer include:

- Providing high-speed service with as little latency as possible.
- Providing reliability to ensure that network connectivity continues in the event of a device failure.

Although many companies choose to deploy Layer 2 switching at the core of a network, Layer 3 switching provides a number of potential advantages. These include:

- Better control of broadcast and multicast traffic.
- Ability to scale the core of a network to much larger sizes.
- Flexible topologies without the need to worry about spanning tree loops.
- Load balancing of traffic over multiple paths, along with fast convergence.

Remember that when designing the core network, it should not include any services or features that will slow down traffic – common examples would be access lists or other security-related policies. Also, the core layer is usually "upgraded" by replacing existing equipment with faster models rather than adding more devices and increasing the network diameter – a constant diameter is always preferred. Any routing protocols used at the core layer should have the fastest convergence times possible in order to minimize potential delays due to route failures. As a general rule, OSPF and EIGRP tend to be the best choices here, especially on larger networks.

Tip: On smaller networks, Layer 2 switching is commonly used at the core layer. On larger networks Layer 3 switching is generally recommended. You should be able to recognize the relative advantages and disadvantages of each method for the CCDA exam.

Router Switching Methods

At the beginning of this chapter we took a look at the process by which a router will accept a frame on one interface, strip away its framing, determine where it should be sent next according to its routing table, reframe the packet, and ultimately forward the frame to the next hop or destination host. The manner in which this process occurs differs based on the switching path method employed by the router. In this case, "switching path" refers to the manner in which a router will accept a packet on one interface, process it, and ultimately forward it out another interface. There are 8 main switching path methods used on Cisco routers that you should be familiar with, as listed below. Note that not all methods are supported on all routers – in fact, some of the faster methods are available only on very high-end models.

- **Process Switching**. With this switching method, incoming packets are copied to the router's buffers, associated with a destination network according to a routing table entry, encapsulated, and then forwarded out the appropriate interface. The router's CPU processes every packet in process switching.
- **Fast Switching**. Fast switching handles the first packet in a stream just like process switching, but then creates a fast switching cache against which following packets are compared. Subsequent packets in the same stream have their incoming frame header stripped off and compared to the first packet. When a match is found, the header appended to the first frame is appended to subsequent frames prior to forwarding. This method helps to eliminate the need for routing table lookups for each packet in the same

stream, increasing router throughput. Fast switching is the default method on lower-end Cisco routers.

- **Optimum Switching**. Though it works in a manner similar to fast switching, optimum switching is faster due to the optimized cache lookup process that it uses. Optimum switching is the default method used on Cisco 7500 series routers.
- **Silicon Switching**. This method uses a dedicated processor known as a silicon switch processor (SSP) module to cache packet switching information. This method allows the switching process to take place without interrupting the router CPU. Silicon switching is available on Cisco 7000 series routers only.
- **Autonomous Switching**. Another very fast switching method found on Cisco 7000 series routers, autonomous switching allows the ciscoBus (cBus) controller to switch packets without the need for CPU intervention.
- **Distributed Switching**. This switching method allows the switching function to take place locally via a route cache stored on a Versatile Interface Processor (VIP) card. This eliminates the need to use the router CPU to perform switching functions.
- **NetFlow Switching**. This switching method allows you to collect detailed statistics on the traffic switched through the router for the purpose of accounting, planning, and network management. Because of the overhead associated with gathering this data, NetFlow switching is generally the slowest switching method.
- **Cisco Express Forwarding (CEF)**. An increasingly popular switching method aimed primarily at high-performance IP backbone switching. Less CPU-intensive than the fast switching method looked at first, CEF uses two main components to make switching decisions – a forwarding information base (FIB) and adjacency table. The FIB contains next-hop information for all IP networks in the routing table, and the adjacency table stores information on associated Layer 2 addresses. With this information, CEF is capable of switching packets faster than through the use of some of the multilayer shortcut switching methods looked at early in this chapter. CEF is no longer limited to Cisco routers; it is also now commonly found in Catalyst multilayer switches.

CCNA Study Guide Chapter 8 Summary

By Dan DiNicolo, July 15th, 2006 Posted in **CCNA Study Guide Chapter 08.** Subscribe to our

RSS Feed

This chapter began with an introduction to routing, including a look at the difference between routed (IP and IPX) and routing (RIP and IGRP) protocols. The communication process on a simple one-router network was described step-by-step, and included an overview of the different processes that take place when a host on one network attempts to communicate with a host on another. A more complex example, involving multiple routers and networks demonstrated the way in which routers will forward packets to the next-hop address on their journey between a source and destination host on different networks.

A look at static routing provided an overview of how this technique allows an administrator to manually define the path that a router will choose in reaching a network. This included a look at the IP routing table, and the syntax of the ip route command. The concept of administrative distances was also introduced, in order to illustrate how a router makes decisions on which routing table entry to use when a path to the same network is learned in different ways. For example, a static routing table entry will always be considered more trustworthy than a route learned from a routing protocol by default.

A look at dynamic routing demonstrated how routers are capable of exchanging routing table information with one another, both through the use of distance vector and link state routing protocols. The major differences between both types of routing protocols were outlined, as was the concept of a hybrid protocol. The danger of routing loops was also discussed, including a look at the techniques used by distance vector protocols to avoid these loops.

An overview of the distance-vector Routing Information Protocol outlined its major characteristics, timer values, and finally the commands to configure and monitor RIP on a Cisco router. The Interior Gateway Routing Protocol was also discussed, including an overview of its configuration. Default routing was looked at as a way to configure a gateway of last resort, the next-hop router to whom a router will send packets destined for unknown networks.

IPX routing was looked at next, including the fact that IPX RIP and SAP messages are broadcast between routers automatically after issuing the ipx routing command. The IPX routing table was looked at, as were the commands to monitor IPX RIP and SAP broadcast traffic. The show protocols command outlined a quick an effective way to obtain information on a router's interfaces, including their current state and configured addresses.

Chapter 9: Cisco IOS Access Lists

Introduction to Chapter and IOS Access Lists

In Chapter 8 we walked through the essentials of configuring a Cisco router for both static and dynamic routing. While the ability to route protocols like IP and IPX might be the central purpose of a router, a Cisco router is actually capable of much more. Cisco's IOS includes the ability to "filter" network traffic based on source or destination address, protocols, port numbers, and more. In Cisco's world, the ability to filter network traffic is accomplished through the use of access lists. Access lists can be defined for a variety of protocols, and ultimately allow you to control the types of traffic that will be allowed in or out of a router interface. For example, you might block a group of hosts from accessing a certain internal server, or limit systems to passing only HTTP traffic to another network.

While the concept of an access list may be simple, the actual implementation of access lists involves some careful planning. It is said that Cisco receives more support calls about misconfigured access lists than anything else. For the purpose of both the CCNA and CCDA exams, you will need to understand the essentials of access lists, including the different types that exist, their capabilities, and how rules are applied and evaluated. The topics that we'll cover in this chapter include:

- An introduction to access lists
- Configuring standard and extended IP access lists
- Configuring standard and extended IPX access lists
- Configuring SAP access lists

The successful use of access lists involves both understanding how they are evaluated, and properly planning the implementation.

Introduction to Access Lists

At the most basic level, an access list is no more than a list of packet filters applied to a router interface. Access lists inspect network packets based on criteria such as source address, destination address, protocols, and port numbers. The rules specified in an access list are then used to either permit or deny the traffic. For example, an access list entry might specify that traffic from network 192.168.25.0/24 should be denied. When network traffic is encountered on the interface, with the access list applied, the router will inspect the packet. If the source address is from the range listed, the packet will be denied, and dropped.

TIP: Cisco generally recommends that access lists be implemented at the distribution layer. Although they are sometimes used at the access layer as well, it is strongly recommended that you avoid implementing access lists at the core layer, since they slow down packet forwarding rates due to the inspection process that takes place.

In order to appreciate access lists, you must keep a few key things in mind. The first is that an access list is nothing more than a series of packet filtering rules. However, this list of rules does nothing until it is applied to a router interface. That's important to remember – first you define an

access list, and then apply it to an interface. Access lists can be applied to an interface to control inbound traffic, outbound traffic, or both. After being applied to an interface, an access list can still have packet-filtering rules added to it. It's also important to keep in mind that access lists only filter traffic that moves through a router. As such, access lists do nothing to filter traffic that remains local.

While an access list that specifies a single rule may be simple, things get a little more complex as additional rules are added. An access list can be made up of many rules, each of which specifies whether certain types of traffic should be permitted or denied. In order to completely understand how access lists filter traffic, you'll need to understand how rules are evaluated, the different types of access lists that exist, and how access lists are applied to interfaces.

Access List Evaluation Rules

An access list can be (and usually is) made up of more than just one packet-filtering rule. Rules in an access list are evaluated from the start of the list, in sequential order. The evaluation process occurs only until the conditions of one of the rules are met. After that, no further rules are looked at.

If you've ever done programming, think of an access list as being similar to an "if-else" loop. A router will inspect traffic it receives on an interface, and compare it to the configured access list, starting from the first rule. Let's say that two simple rules exist in an access list, as shown in Figure 9-1.

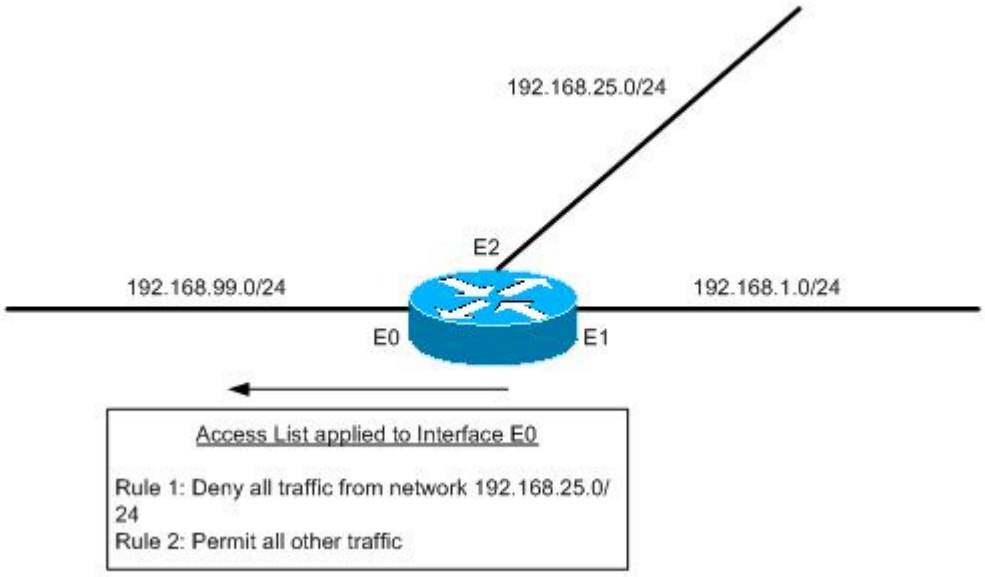


Figure 9-1: Simple access list that filters all traffic exiting router interface E0.

This access list has been configured to filter traffic as it exits interface E0, onto network 192.168.1.0/24. When RouterA receives traffic to be forwarded out interface E0, it will evaluate it against the access list. In this case, let's assume that the traffic has a source address of 192.168.1.100. The first rule specifies that all traffic from network 192.168.25.0/24 should be denied. Obviously the rule doesn't affect this packet. Because of this, the router looks at the next rule, which specifies that all other traffic should be allowed to pass. Our packet meets this criteria – it is part of "all other traffic", after all. The router will forward this packet.

However, when traffic is reaches interface E0 from address 192.168.25.3, the router will evaluate the first rule and find a match – this traffic is from network 192.168.25.0/24. Because of this, the action specified in the rule should be carried out, and no other rules will be evaluated. In this case, the rule specifies that traffic from 192.168.25.0/24 should be denied, so the packet is dropped.

Getting back to my "if-else" comment from earlier, think of it like this. An access list will inspect traffic according to the rules specified, moving from top to bottom. If the packet matches the criteria specified in the rule, the associated action (permit or deny) will be carried out. If not (else), the next rule will be evaluated. This will continue until the end of the access list is reached.

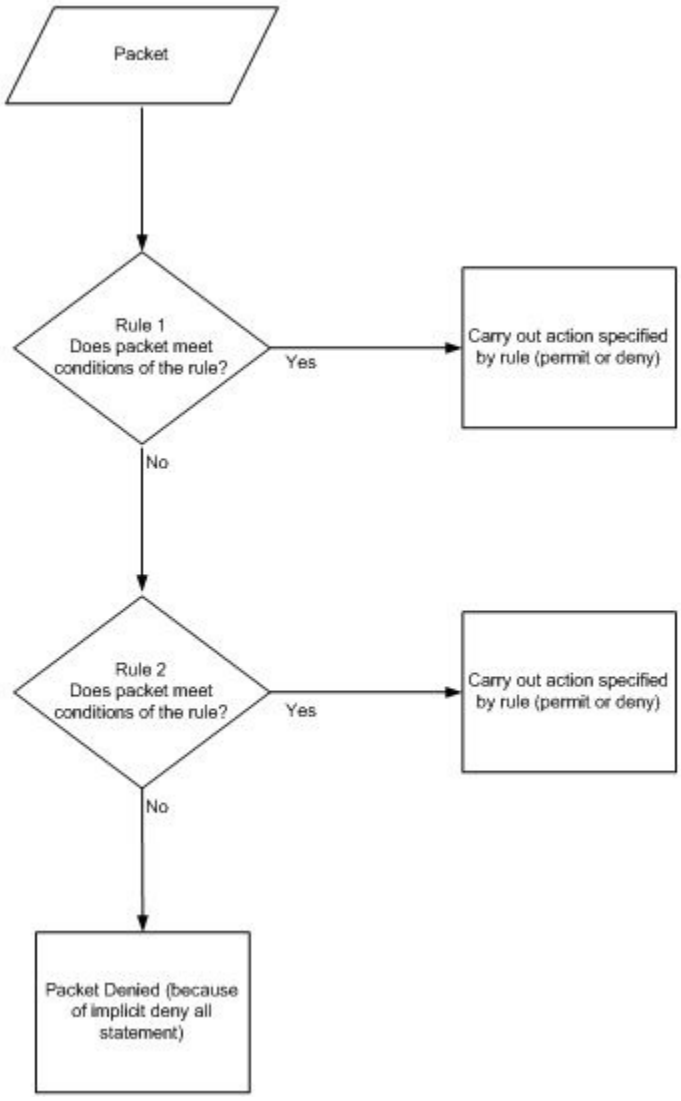


Figure 9-2: When a packet encounters an interface with an access list applied, it will be evaluated until a match is found.

Tip: Access lists are evaluated starting with the first rule, and only until a match is made.

The second critical thing to understand about access lists is how they end. Every single access list ends with what is known as an "implicit deny all" statement. This deny statement, which is not visible when viewing the list, specifies that all traffic that does not match any of the rules in the access list should be denied, and as such, dropped.

This is critical – on every access list you look at, always remember that it ends in a "deny all" statement, even though you don't see it. Since access lists are evaluated sequentially from the beginning of the list, the purpose this serves is to deny any traffic that you haven't explicitly permitted. I can't overstate the importance of remembering that all access lists end with this "phantom" statement.

Tip: All access lists end with a hidden statement that denies all traffic.

For example, consider the simple access list applied to interface E0, shown in Figure 9-3. It includes three rules, some of which permit traffic, and some of which deny it. Our phantom "deny all" is there, even though we don't see it. This particular access list filters all traffic as it attempts to exit interface E0 on RouterA. This is known as an outbound access list.

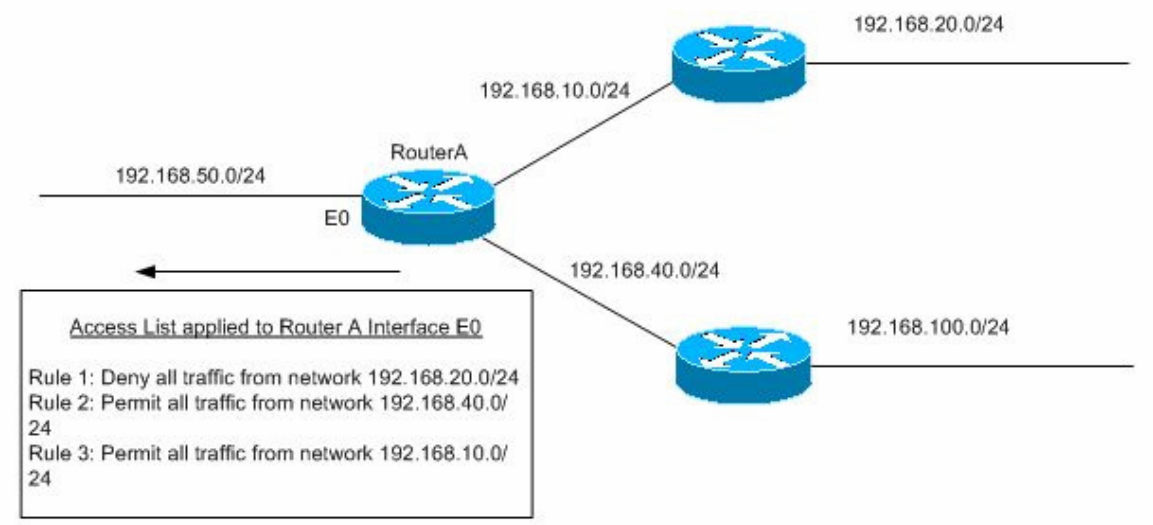


Figure 9-3: Access list with 3 explicit rules that filter all traffic exiting RouterA interface E0.

In this case, when a packet with a source address of 192.168.10.13 reaches interface E0, it will be permitted, since the first rule doesn't apply, and the second rule permits access from systems on network 192.168.10.0/24. When a packet from source address 192.168.20.3 reaches interface E0, it will be denied (according to the first rule), and dropped. Notice what happens when a packet from source address 192.168.100.13 reaches interface E0: the first rule doesn't impact this packet, nor do the second and third. So what happens? The packet is denied and dropped, because of the "invisible" deny statement at the end of list. Don't underestimate the importance of this – forgetting about the implicit deny statement is probably the cause of more access list "problems" than all others combined.

The question then becomes what to do if you only want to deny traffic from certain source addresses. Consider the simple access list shown in Figure 9-4. The second visible rule specifies that all traffic should be permitted.

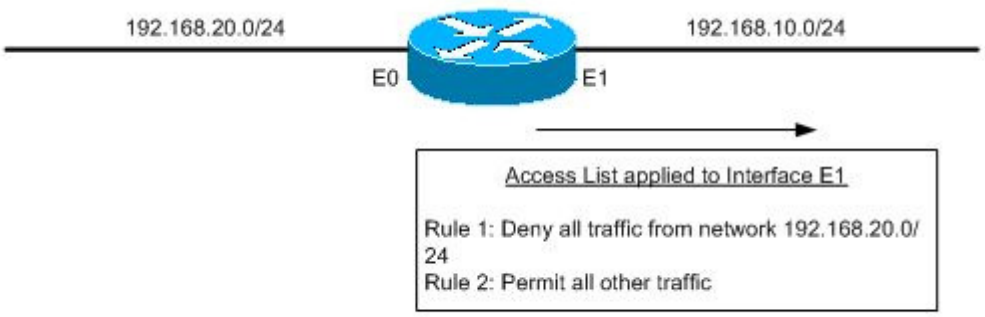


Figure 9-4: When an access list ends with an explicit statement permitting all traffic, the implicit deny all statement will never been reached.

In this case, traffic that isn't filtered by the first rule would move on to the second, and then be allowed to pass. The implicit deny at the end of the access list would never be reached, since the second rule says that all traffic should be permitted to pass. We'll look at this in more detail later in the chapter, when we start defining real access lists.

Types of Access Lists

Two major types of access lists exist in the Cisco IOS – standard and extended. Standard access lists provide basic filtering capabilities. For example, a standard IP access list only allows the source address of a packet to be used in filtering decisions. Extended access lists allow filtering to be accomplished in a more granular way. For example, an extended IP access list allows packets to be filtered according to source address, destination address, protocol type, port numbers, and so forth. The decision on whether to use standard or extended access lists will depend upon what it is that you are trying to accomplish.

For example, if you simply want to deny access to certain hosts based on their IP address, a standard IP access list would suffice. However, if your needs were more specific, and you wanted to block a certain group of hosts from accessing telnet on a particular server, an extended IP access list would be required.

In the Cisco IOS, access lists are identified numerically. This number not only uniquely identifies an access list, but also specifies the type of list, based on the numeric range it falls into. Table 9-1 outlines the numeric ranges associated with different types of access lists. Depending on the protocols supported by your IOS, the list may differ, but the number ranges remain the same.

Table 9-1: Access lists and their associated numeric ranges.

Number Range	Type of Access List
1-99	IP standard access list
100-199	IP extended access list

200-299	Protocol type-code access list
300-399	DECnet access list
400-499	XNS standard access list
500-599	XNS extended access list
600-699	AppleTalk standard access list
700-799	AppleTalk extended access list
800-899	IPX standard access list
900-999	IPX extended access list
1000-1099	IPX SAP access list
1100-1199	Extended 48-bit MAC address access list
1200-1299	IPX summary address access list

Creating an access list with the number “87” would identify the list as an IP standard access list. The number “907” would identify an IPX extended access list. Access lists do not need to be numbered in any specific order – just be sure to give lists a unique number (in the proper range) that you’ll remember.

Applying Access Lists to Router Interfaces

After an access list has been created, it ultimately needs to be applied to an interface in order to filter traffic. Access lists can be applied in one of two ways – inbound or outbound. Differentiating between the two and understanding both is critical.

- **Inbound**. When an access list is applied to an interface as inbound, the access list is evaluated when a packet attempts to enter that interface. In other words, an inbound access list controls which types of traffic are allowed to enter the router through that particular interface.
- **Outbound**. When an access list is applied to an interface as outbound, the access list is evaluated when a packet attempts to exit an interface. In other words, an outbound access list controls which types of traffic are allowed to exit the router through that interface.

I understand that the difference between configuring an access list as inbound or outbound can be a little confusing. It’s actually easier to understand with an example. Consider Figure 9-5. In it, our router has three interfaces (E0, S0, and S1), each connected to a different network. For the purpose of illustration, I’m going to add one very simple access list to the router, on interface E0.

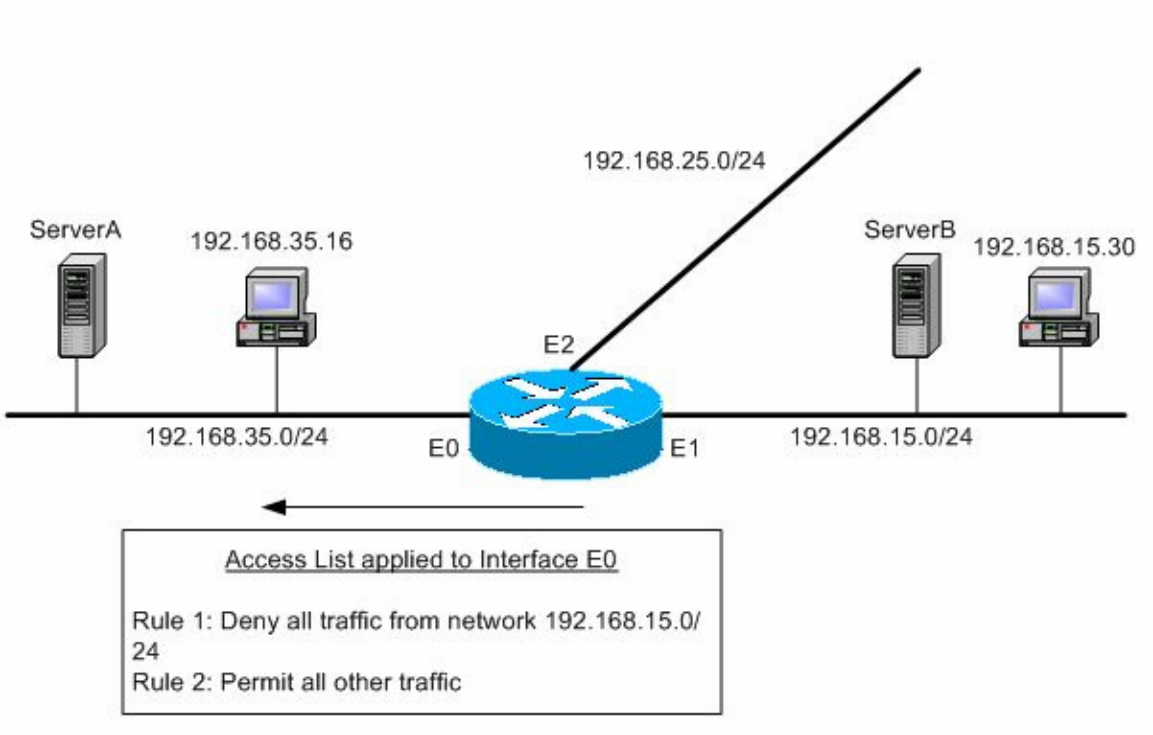


Figure 9-5: Router connecting three networks, with an outbound access list applied to interface E0.

Notice that the access-list is applied as an outbound access list on interface E0. It denies access to hosts from network 192.168.15.0/24, and permits all other traffic.

Let's look at the impact of this outbound access list. It specifies that traffic from hosts on network 192.168.15.0/24 should be denied. The steps below outline what will happen when host 192.168.15.30 attempts to communicate with a server on network 192.168.35.0/24.

1. Host 192.168.15.30 will recognize that the server it wants to communicate with is on a remote network, and will forward packets via its default gateway, router interface S1.
2. The router will use its routing table to determine that these packets should be forwarded out interface E0.
3. Recall that interface E0 has an outbound access list applied. It will be evaluated for all packets that attempt to exit interface E0. In this case, the source address of the packets (192.168.15.30) matches the first rule specified, and according to the access list, the packets are dropped.

In this case, the access list isn't evaluated until traffic tries to exit interface E0. Had a host on network 192.168.25.0 attempted to access a server on network 192.168.35.0, it would have been permitted to pass through the outbound access list. It would not have been affected by the first rule, while the second rule permits all other traffic to pass.

However, this outgoing access list does a little more than just stop systems on network 192.168.15.0 from initiating communications with hosts on network 192.168.35.0. For example, if a host on network 192.168.35.0 attempted to communicate with a server on network 192.168.15.0, the following would happen.

1. Host 192.168.35.16 would recognize that the host it wants to communicate with is on a remote network, and would forward packets via its default gateway, router interface E0.
2. The router will use its routing table to determine that these packets should be forwarded out interface S1. Note that the access list on E0 didn't apply – these packets are entering, not exiting, interface E0. The access list is outbound only.
3. The packets will reach the server at 192.168.15.200, who will send a reply. The reply will reach the router, who will check its routing table, and determine that the packets are to be forwarded out interface E0.
4. Unfortunately, because the source address in the reply is coming from network 192.168.15.0, the router will drop the packets, since the outbound access list denies traffic from network 192.168.15.0, according to the first rule.

As you can see, even one simple access list can have many repercussions. The outbound access list applied to interface E0 stopped traffic from network 192.168.15.0 from reaching network 192.168.35.0. However, this means that all traffic with a source address of 192.168.15.0 is blocked, even when it's a reply from a request that originated on network 192.168.35.0.

Going a step further, let's add an inbound access list on interface E2. The access list will also include two rules. The first will deny hosts on network 192.168.25.0 from accessing telnet services on any network, while the second will permit all other types of traffic to pass, as shown in Figure 9-6.

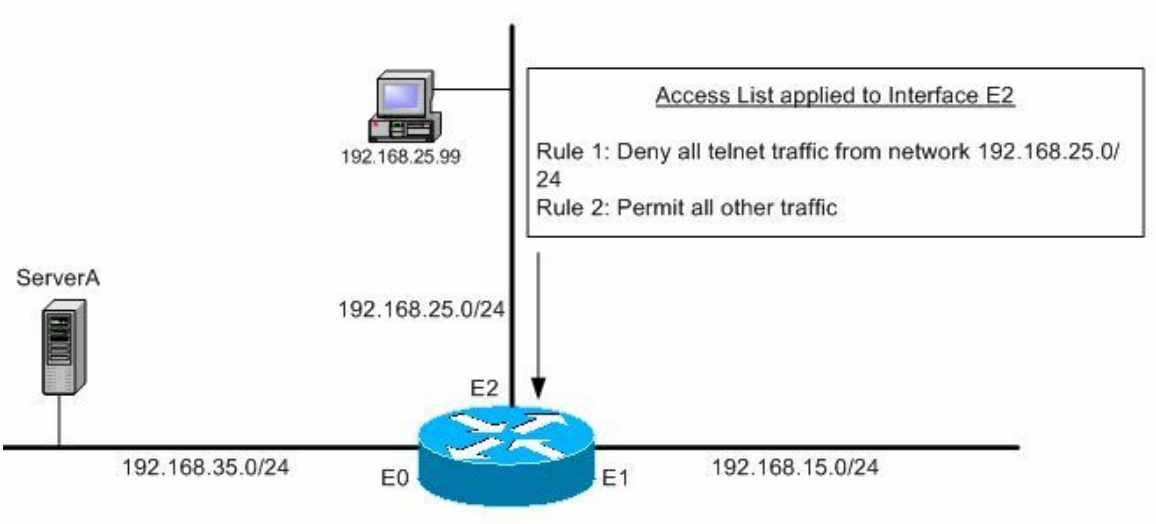


Figure 9-6: The inbound access list on interface S1 denies telnet traffic from hosts on network 192.168.25.0/24, but allows all other traffic.

In this example, imagine that host 192.168.25.99 tries to initiate a telnet session with Server A.

1. Host 192.168.25.99 will recognize that Server A is on a different network, based on the destination IP address. It will send the packets to its configured default gateway, interface E2.
2. Since traffic is attempting to enter interface E2, the inbound access list will be evaluated. In this case, the first rule specifies that any telnet traffic originating in network 192.168.25.0/24 should be denied. The router will drop the packets immediately.

Even though telnet traffic is denied via the inbound access list, all other traffic is still allowed to pass. If host 192.168.25.99 attempted an HTTP connection with ServerA, the rules in the access list would be evaluated. The first rule wouldn't apply (since this isn't telnet traffic), and the second rule permits all other traffic to pass. Again, recognize the importance of that second rule – if it didn't exist, all traffic from network 192.168.25.0/24 would be denied. While the first rule denies telnet traffic only, without the second rule, the implicit "deny all" at the end of the access list would stop all traffic from network 192.168.25.0/24 from being allowed to enter interface E2. If that's your goal, it's probably a better idea to just shut down the interface. Both would have the same end result. Every access list requires at least one permit statement to do anything useful.

For any interface (or subinterface), only one inbound and outbound access list can be applied per protocol. In other words, an interface could have one IP and one IPX access list applied inbound, and one IP and one IPX access list applied outbound. You could not have 2 IP access lists applied inbound on the same interface. However, you can have multiple access lists associated with a single physical interface through the use of subinterfaces. For example, an inbound IP access list could be applied to both interface s0 and interface s0.1.

Tip: For any interface (or subinterface), only one inbound and outbound access list can be applied per protocol.

One important note before we actually begin implementing access lists. Any access lists applied to interfaces will not impact traffic that originates from the router where the access lists are defined. For example, configuring an outbound access list that denies telnet traffic will not deny a telnet session initiated from the router from leaving that interface.

Standard IP Access Lists

As mentioned earlier, a standard IP access list provides basic packet filtering abilities, based on the source IP address of a packet only. As a general rule, apply standard IP access lists close to the destination network to which you wish to permit or deny access. Consider Figure 9-7 - in this simple network, we wish to deny hosts on network 10.1.10.0/24 from accessing network 10.1.30.0/24. If an outbound access list were placed on RouterA interface E1, this would stop hosts on network 10.1.10.0/24 from accessing both networks 10.1.20.0 and 10.1.30.0. By placing the outbound standard access list on RouterB interface E1 instead, hosts on network 10.1.10.0/24 are only denied access to network 10.1.30.0.

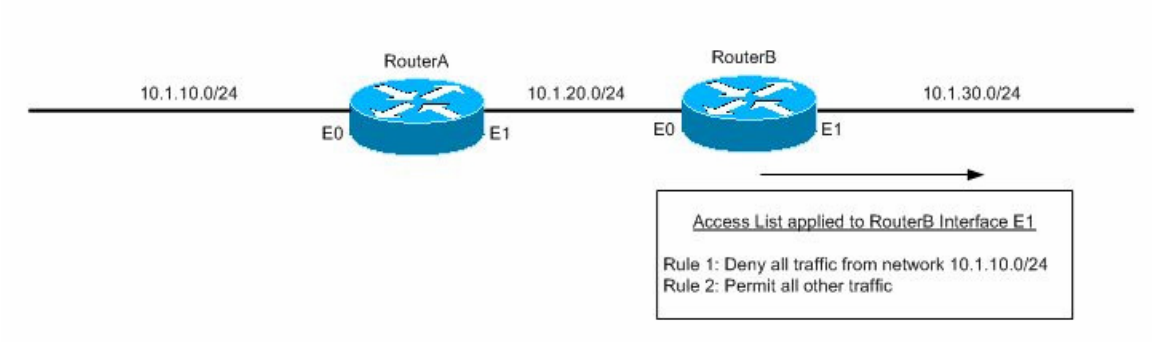


Figure 9-7: The outbound access list applied to RouterB interface E1 denies access to hosts from network 10.1.10.0/24.

Recall that standard IP access lists fall into the numerical range 1-99. Implementing a standard access list involves two major steps. The first step involves adding rules to an access list. The second involves applying the access list to an interface.

IP standard access lists are created from global configuration mode, using the access-list command. The syntax of an access list can be a little confusing, so we'll walk through our examples using the help feature of the Cisco IOS. The complete syntax for an IP standard access list is:

access-list *access-list-number* {**deny** | **permit**} *source [source-wildcard]* **log**

The access-list command is used to specify the creation of a new access list entry. The *access-list-number* is the number of the access list to which this rule will be added. It's worth noting that if an access list of this number doesn't exist, it will be created. If it does exist, the command will add another rule to the bottom of that access list. Then the deny or permit entry does what their names suggest – outlines the action to be carried out when the criteria specified in the rule is met. A permitted packet is allowed to pass, while a denied packet is dropped. The *source* section allows you to specify the source address of the host to which the rule applies. This might be a single host, or a group of hosts if a customized *source-wildcard* mask is specified (wildcard masking will be looked at in the next section). The log command is optional – if specified, the router will literally display a message on the console each time an access list entry is matched. While it may be interesting to see, I generally suggest leaving it turned off, especially on a production router.

The best way to begin is by creating an access list based on a scenario. In Figure 9-8, our network consists of two subnets. Our goal is to deny traffic from host 192.168.1.100 from reaching network 192.168.2.0. All other traffic should be allowed to pass.

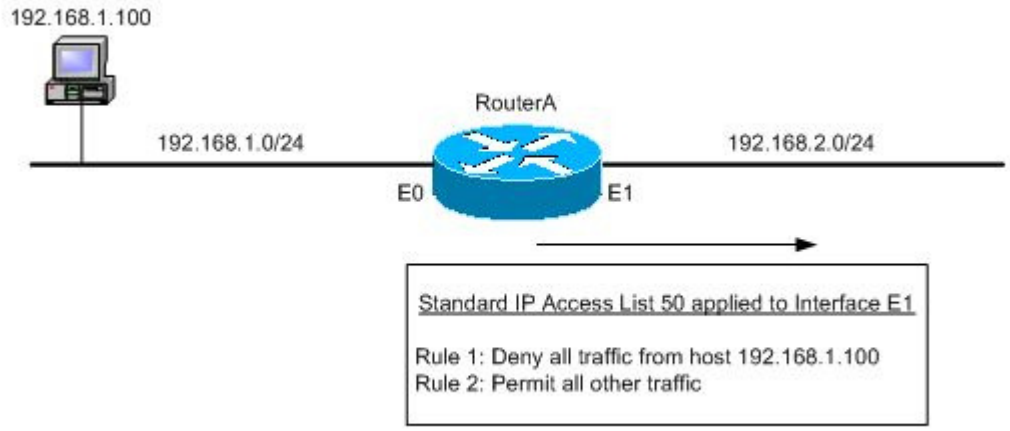


Figure 9-8: An outbound access list will be applied outbound on interface E1.

We'll give the access list a number of 50, which I chose randomly. Remember that access lists are defined from global configuration mode.

```
RouterA(config)#access-list ?
<1-99>      IP standard access list
<100-199>   IP extended access list
<1000-1099> IPX SAP access list
<1100-1199> Extended 48-bit MAC address access list
```

```
<1200-1299>  IPX summary address access list
<1300-1999>  IP standard access list (expanded range)
<200-299>    Protocol type-code access list
<2000-2699>  IP extended access list (expanded range)
<300-399>    DECnet access list
<600-699>    Appletalk access list
<700-799>    48-bit MAC address access list
<800-899>    IPX standard access list
<900-999>    IPX extended access list
```

By using help, we know that the access-list command expects an access list number next. We're using the number 50. Requesting help again yields the following:

```
RouterA(config)#access-list 50 ?
deny    Specify packets to reject
permit  Specify packets to forward
```

The next step is specifying whether this rule permits or denies traffic. The goal of our first rule is to deny access to host 192.168.1.100, so we'll choose deny.

```
RouterA(config)#access-list 50 deny ?
Hostname or A.B.C.D  Address to match
any                  Any source host
host                 A single host address
```

After specifying our deny statement, we are prompted to identify the host (or hosts) for which we wish to deny access. Notice that possible entries include a hostname (this would require a configured hosts table or DNS lookup), the IP address of the host, or use of the any keyword. Choosing any would deny access to all hosts in this case. Denying access to a single host can be accomplished in two different ways. The first method is by using the host keyword, as shown below.

```
RouterA(config)#access-list 50 deny host 192.168.1.100
```

Another way to accomplish the same thing is to specify what is known as a wildcard mask. For a single host, the wildcard mask is all 0s, as shown below. We'll look at how to specify wildcard masks for groups of hosts shortly.

```
RouterA(config)#access-list 50 deny 192.168.1.100 0.0.0.0
RouterA(config)#
```

Pressing Enter completes the command (though the log keyword could have also been added). This adds a single rule to standard IP access list 50. We still need to add a second rule, specifying that all other traffic is permitted. This command is very straightforward:

```
RouterA(config)#access-list 50 permit any
```

The second rule simply specifies that all traffic is allowed to pass. It's important to remember that this is the second rule on the list – rules are sequentially added to an access list, in the order they are specified. If we were to add another rule, it would be the third on the list, and so forth. There is no way to change the order of rules in the Cisco IOS. However, one popular way to edit the order of access lists is to save your configuration to a TFTP server, and then edit the saved configuration file in a text editor.

It's also important that you think very carefully about the order in which you add access list entries. You want to make sure that the entries permit or deny traffic as per your intentions. For that reason, it is extremely important that more specific entries always be added towards the beginning of an access list. If not, a packet that you intended to deny may in fact be permitted by a more "generous" preceding entry. Remember – access lists are only evaluated until a match is found. If we had created our access list in the reverse order (with the access-list 50 permit any rule added first), the second rule would never be evaluated - all traffic would match the first rule.

To view the access lists that have been defined on a router, use the show run command. Though I've truncated the output below, access list entries appear towards the bottom of the configuration file.

```
RouterA#sh run
Building configuration...
Current configuration:
!
access-list 50 deny   192.168.1.100
access-list 50 permit any
```

With our simple standard IP access list defined, the next step is applying it to an interface. The command to apply a standard IP access list to an interface is ip access-group, followed by the number of the access list and a direction (inbound or outbound). Access lists are applied to an interface from interface configuration mode, as shown below.

```
RouterA#config t
Enter configuration commands, one per line.  End with CNTL/Z.
RouterA(config)#int e1
RouterA(config-if)#ip access-group ?
<1-199>      IP access list (standard or extended)
<1300-2699>  IP expanded access list (standard or extended)
WORD         Access-list name
RouterA(config-if)#ip access-group 50 ?
in   inbound packets
out  outbound packets
RouterA(config-if)#ip access-group 50 out
```

Recall our original scenario. Our goal was to deny host 192.168.1.100 access to network 192.168.2.0/24. By configuring the access-list as outbound on the router's E1 interface, only traffic from host 192.168.1.100 is denied the ability to exit interface E1. You might wonder why we didn't just apply the access list as inbound on interface E0. If we had, it would have denied host 192.168.1.100 access to all networks, and not just network 192.168.2.0/24.

To view the IP access lists applied to an interface, use the show ip interface or show running-config commands. The show ip int e1 command is shown below, truncated to only include relevant output.

```
RouterA#sh ip int e1
Ethernet0 is up, line protocol is up
Internet address is 192.168.2.1/24
Broadcast address is 255.255.255.255
Address determined by setup command
MTU is 1500 bytes
Helper address is not set
Directed broadcast forwarding is disabled
Outgoing access list is 50
Inbound  access list is not set
```

To view all access lists applied to a router, use the show access-lists command. To view a particular IP access list, use the show ip access-list command, followed by the access list number. Examples of both are outlined below.

```
RouterA#show access-list
Standard IP access list 50
deny   192.168.1.100
permit any
RouterA#show ip access-list 50
Standard IP access list 50
deny   192.168.1.100
permit any
```

Because we only have one access list defined at this point, the output of both commands looks similar. However, the first command will ultimately list all access lists (including IPX or otherwise), while the second shows only the contents of a specific IP access list.

To remove an access list from an interface, you use the “no” version of the ip access-group command from interface configuration mode.

```
RouterA(config-if)#no ip access-group 50 out
```

To entirely remove an access-list from the router, use the no access-list command, followed by the number of the access list that you wish to delete from global configuration mode.

```
RouterA(config)#no access-list 50
```

Using Wildcard Masks

In the standard IP access list that we looked at in the previous article, you learned how to define a rule that would permit or deny access to a single host. In reality, you will probably wish to permit or deny access to a range of hosts rather than just one. Perhaps you’ll want to control access for all of the hosts on a subnet, or maybe just a subset of hosts. Either way, the ability to control access for a group of hosts is accomplished using what is known as a wildcard mask.

A wildcard mask is different than a subnet mask. Defining a wildcard mask is really no more difficult, but the approach is somewhat different. The purpose of the wildcard mask is to specify which group of addresses an access list entry should apply to. For example, imagine that we wanted to create a standard IP access list that would deny inbound access on interface E0 to all hosts on network 192.168.20.0/24. The access list entry would be:

```
RouterA(config)#access-list 40 deny 192.168.20.0 0.0.0.255
```

I know that the wildcard mask looks a little confusing. In this case, the wildcard mask is 0.0.0.255. The binary 0s in the mask tell the router that the associated bits in the source address must match exactly. Since the first 3 octets in the wildcard mask are set to binary 0, the router knows that the first three octets must match 192.168.20. The binary 1s in the mask tell the router to match any possible value. Since the last octet in the wildcard mask is all binary 1s, the router knows that any value in the last octet is a match. So, any source address beginning with 192.168.20 would match the access list, and in this case, be denied.

In the same way, consider the example below. It tells the router to permit traffic from hosts with any address that starts with 10.10. Since the last two octets are masked, any source address

starting with 10.10 will be a match. In this case, all hosts from 10.10.0.1 to 10.10.255.254 would be permitted access.

```
RouterA(config)#access-list 40 deny 10.10.0.0 0.0.255.255
```

Recall that a single host can be specified in an access list using either the host command, or a wildcard mask of all 0s. The wildcard mask of all 0s means “match all octets exactly”. In the example below, only host 192.168.1.100 would be denied access.

```
RouterA(config)#access-list 40 deny 192.168.1.100 0.0.0.0
```

Things get a bit trickier when you want to block only a certain range of hosts, but not necessarily an entire network or subnet. Before looking at the masking, there is an important rule to remember. When you wish to block only a certain group of hosts, the multiple and starting values must be powers of 2. For example, you can block 2, 4, 8, 16, 32, 64, or 128 hosts, but not, for example, 26. If you think back, this is similar to the rules associated with subnetting.

Defining a custom wildcard mask is different than subnetting, in that those binary ones move in the opposite direction. Remember that the 1s in a wildcard mask specify the bits that the router should pay attention to in the source address. For example, imagine if we wanted to deny access to a group of 4 addresses starting with 192.168.1.4. In effect, that means that we want to deny access for addresses between 192.168.1.4 and 192.168.1.7, or 4 addresses total. In this case, the wildcard mask would be 0.0.0.3. At this point, I agree that number looks confusing. A quick look at things in binary will help to make things clearer.

Remember that we want to start at 192.168.1.4 and include 4 addresses. The wildcard mask specifies the bits that the router should consider when attempting to determine which hosts to permit or deny. In this case, the mask is 0.0.0.3, which is the equivalent to the following in binary:

00000000 00000000 00000000 00000011

Notice that only the last two bits are set to 1. What this means is “start at 192.168.1.4, and include any values for those last two bits”. Figure 9-9 outlines what all of the possible values would be for those last two octets.

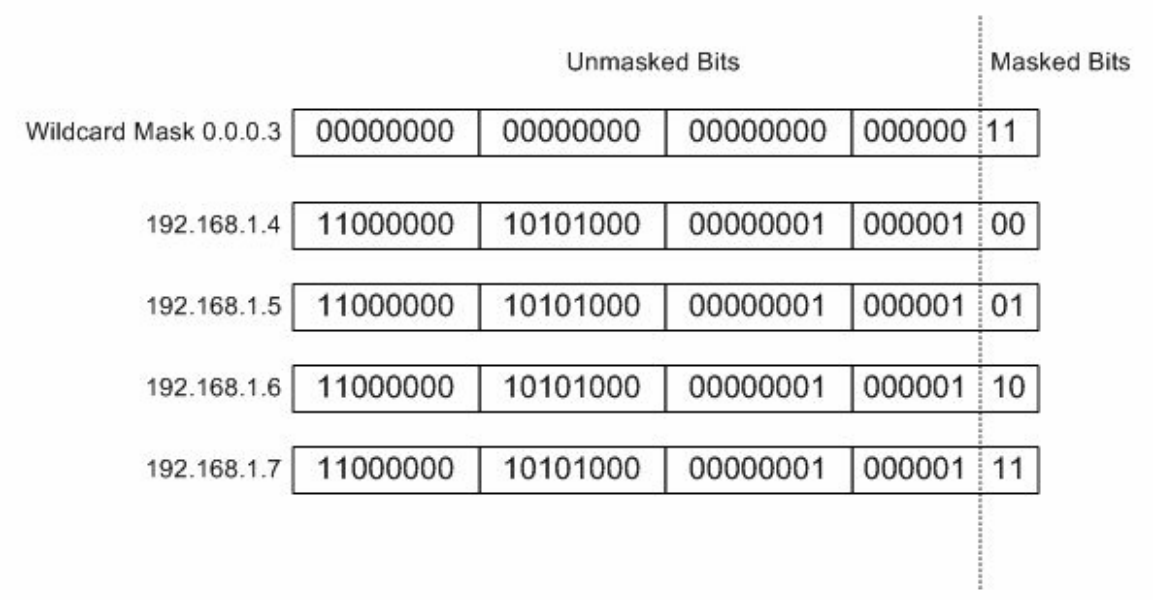


Figure 9-9: Matching address values for when a source address of 192.168.1.4 is specified with a wildcard mask of 0.0.0.3

The only possible addresses that apply are 192.168.1.4, 192.168.1.5, 192.168.1.6, and 192.168.1.7. In other words, our wildcard mask tells the router that starting with address 192.168.1.4, allow the last two address bits to be set to any value, and use this as the range to permit or deny traffic.

If you're looking for an easier way to remember wildcard mask values, you're in luck. Recall what I said earlier. When grouping ranges of addresses, they must be a valid power of 2. The associated wildcard mask number will always be one less than the value of the range. So, if you want to group 128 addresses together, the associated wildcard mask value would be 127. To group 64 addresses, the wildcard mask value would be 63, and so forth. These values are outlined in Table 9-2.

Table 10-2: Address groupings and associated wildcard mask values.

Address Grouping	128	64	32	16	8	4	2	1
Wildcard Mask	127	63	31	15	7	3	1	0

Let's take a closer look at the table. Notice that for a single address, the wildcard mask value is 0. This is consistent with our example of creating a wildcard mask for a single host of 0.0.0.0.

Let's say that we want to group together 8 addresses starting at 192.168.20.17. Unfortunately we can't. Why? Because you cannot start with the number 17 – the range must begin at a power of 2. So, let's start at 192.168.20.16 instead. If we want to deny access to a group 16 addresses, then the wildcard mask will be one less – 15. In this case, the mask would be 0.0.0.15. This would deny access to all hosts between 192.168.20.16 and 192.168.20.31. If we wanted to add this entry to a standard IP access list, the command would be:

```
RouterA(config)#access-list 40 deny 192.168.20.16 0.0.0.15
```

The table above can also be used to block bigger ranges. For example, imagine if we wanted to permit access for 32 subnets, those from 192.168.32.0 up to 192.168.63.0. In this case, the wildcard mask must begin in the third octet. Because of this, the wildcard mask becomes 0.0.31.255. Notice that the third octet value is one less than the grouping, and that the last octet is 255. This tells the router that this access list entry applies to hosts with any value between 32 and 63 in the third octet, and any value in the fourth octet. The access list entry would be:

```
RouterA(config)#access-list 40 permit 192.168.32.0 0.0.31.255
```

As you'll see shortly, wildcard masks can be used in both standard and extended IP access lists.

Extended IP Access Lists

Unlike standard IP access lists (which only allow you to filter packets based on their source IP address), extended IP access lists allow a much more granular level of control. Extended IP access lists allow filtering not only on source addresses, but also on destination addresses, protocols, and even applications, based on their port number. For example, you might choose to permit or deny a group of hosts from accessing a particular server, limit access to a telnet server to only a single host, or similar. Recall that extended IP access lists are identified through their use of the 100-199 numerical range.

The syntax of an extended IP access list is similar to that of a standard IP access list, though obviously a little longer, based on the additional filtering options available.

access-list access-list-number { **deny** | **permit**} **ip**|**tcp**|**udp**|**icmp**} *source [source-mask] dest [dest-mask]* **lt**|**gt**|**eq**|**neq** *dest-port**log***

Although the command looks quite complex, you won't be required to remember every option. The syntax of the command is actually the same as the syntax of a standard IP access list up until the permit or deny statement. That statement is followed by the protocol type you wish to specify, and then the source and destination addresses. The end of the statement allows you to specify the port number(s) for which the rule applies.

For example, let's say that we want to deny all hosts on network 192.168.20.0/24 from accessing ServerA via telnet, as shown in Figure 9-10. The command to add this entry to access list 102 is shown below, using help to walk through the command step-by-step.

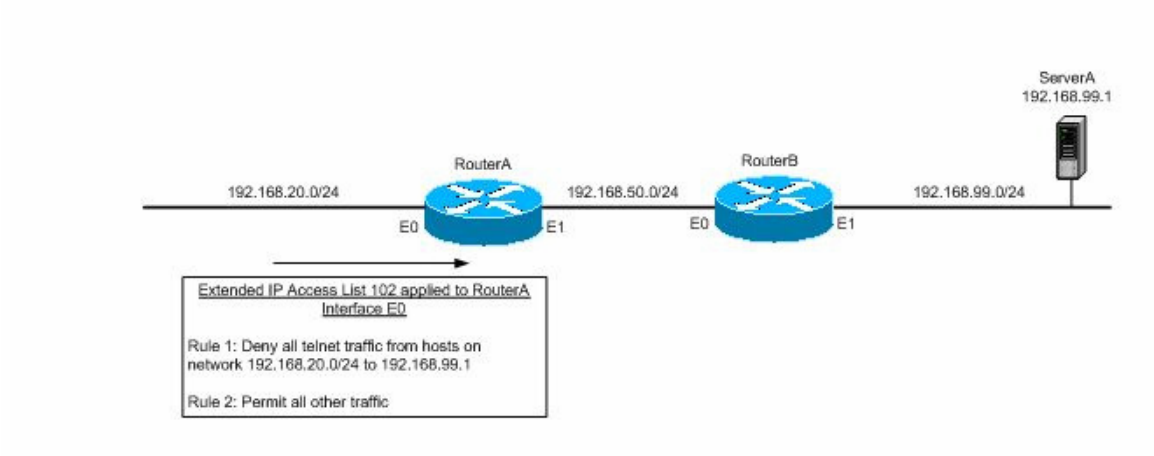


Figure 9-10: An inbound extended IP access list on RouterA interface E0 denies hosts on network 192.168.20.0/24 from accessing ServerA via telnet.

```
RouterA(config)#access-list 102 deny ?
<0-255>  An IP protocol number
ahp      Authentication Header Protocol
eigrp    Cisco's EIGRP routing protocol
esp      Encapsulation Security Payload
gre      Cisco's GRE tunneling
icmp     Internet Control Message Protocol
igmp     Internet Gateway Message Protocol
igrp     Cisco's IGRP routing protocol
ip       Any Internet Protocol
ipinip   IP in IP tunneling
nos      KA9Q NOS compatible IP over IP tunneling
ospf     OSPF routing protocol
pcp      Payload Compression Protocol
pim      Protocol Independent Multicast
tcp      Transmission Control Protocol
udp      User Datagram Protocol
```

The help function lists the different protocols on which packets can be filtered. Because telnet functions over TCP port 23, we'll choose tcp as our protocol.

```
RouterA(config)#access-list 102 deny tcp ?
A.B.C.D  Source address
any      Any source host
host     A single source host
```

The next step is specifying the source host address, followed by a wildcard mask. In this case, we want our list to apply to all hosts on network 192.168.20.0/24, so our wildcard mask is 0.0.0.255.

```
RouterA(config)#access-list 102 deny tcp 192.168.20.0 0.0.0.255 ?
A.B.C.D  Destination address
any      Any destination host
eq       Match only packets on a given port number
gt       Match only packets with a greater port number
host     A single destination host
lt       Match only packets with a lower port number
neq      Match only packets not on a given port number
range    Match only packets in the range of port numbers
```

```
RouterA(config)#access-list 102 deny tcp 192.168.20.0 0.0.0.255 192.168.99.1 ?
A.B.C.D  Destination wildcard bits
```

The next step is specifying our destination host address. I chose to enter the destination IP address followed by the wildcard mask of 0.0.0.0. Recall that the host keyword can also be used when specifying a single host.

```
RouterA(config)#$ 102 deny tcp 192.168.20.0 0.0.0.255 192.168.99.1 0.0.0.0 ?
ack          Match on the ACK bit
eq           Match only packets on a given port number
established  Match established connections
fin          Match on the FIN bit
gt           Match only packets with a greater port number
log          Log matches against this entry
log-input    Log matches against this entry, including input interface
lt           Match only packets with a lower port number
neq          Match only packets not on a given port number
precedence   Match packets with given precedence value
psh          Match on the PSH bit
range        Match only packets in the range of port numbers
rst          Match on the RST bit
syn          Match on the SYN bit
tos          Match packets with given TOS value
urg          Match on the URG bit
<cr>
```

Notice the $ sign that appears next to the prompt above. This is simply a placeholder that makes you aware that the command entered is too long to appear on a single line.

One additional caveat at this point – if we had pressed enter after entering the destination wildcard mask, our access list entry would be accepted. However, it would also deny all TCP traffic from network 192.168.20.0/24 to host 192.168.99.1. The last step in configuring our extended access list is specifying the TCP port number (or protocol name) that we wish to deny - in this case port 23. By using the eq (equal to) operator, we can specify that this access list entry applies to port 23 only. You can also specify certain protocols by name, as shown below.

```
RouterA(config)#$t tcp 192.168.20.0 0.0.0.255 192.168.99.1 0.0.0.0 eq ?
<0-65535>    Port number
bgp          Border Gateway Protocol (179)
chargen      Character generator (19)
cmd          Remote commands (rcmd, 514)
daytime      Daytime (13)
discard      Discard (9)
domain       Domain Name Service (53)
echo         Echo (7)
exec         Exec (rsh, 512)
finger       Finger (79)
ftp          File Transfer Protocol (21)
ftp-data     FTP data connections (used infrequently, 20)
gopher       Gopher (70)
hostname     NIC hostname server (101)
ident        Ident Protocol (113)
irc          Internet Relay Chat (194)
klogin       Kerberos login (543)
kshell       Kerberos shell (544)
login        Login (rlogin, 513)
lpd          Printer service (515)
nntp         Network News Transport Protocol (119)
pim-auto-rp  PIM Auto-RP (496)
```

```
pop2        Post Office Protocol v2 (109)
pop3        Post Office Protocol v3 (110)
smtp        Simple Mail Transport Protocol (25)
sunrpc      Sun Remote Procedure Call (111)
syslog      Syslog (514)
tacacs      TAC Access Control System (49)
talk        Talk (517)
telnet      Telnet (23)
time        Time (37)
uucp        Unix-to-Unix Copy Program (540)
whois       Nicname (43)
www         World Wide Web (HTTP, 80)
RouterA(config)#$tcp 192.168.20.0 0.0.0.255 192.168.99.1 0.0.0.0 eq 23
```

After issuing the completed command, our extended IP access list now includes a single entry, which denies hosts on network 192.168.20.0/24 from sending telnet traffic destined for host 192.168.99.1 through the router. Remember, however, that the access list doesn't actually filter traffic until applied to an interface. Also recall that all access lists end with the implicit deny statement. As such, we should add an entry that allows all other traffic to be forwarded by the router, and apply the list to an interface. To allow all other traffic to pass, we will need to add another entry to access list 102.

```
RouterA(config)#access-list 102 permit ip any any
```

Notice the syntax of the command. It adds an entry to extended IP access list 102, telling it to permit all IP traffic from any source to any destination. This is a common statement, and will meet our needs. To view access list 102, use the show ip access-list 102 command.

```
RouterA#show ip access-list 102
Extended IP access list 102
deny tcp 192.168.20.0 0.0.0.255 host 192.168.99.1 eq telnet
permit ip any any
```

Even though we didn't specify a protocol name in our original access list entry, the router still recognizes TCP port 23 as being a telnet port.

Our final step is applying this access list to an interface, and specifying whether it will be applied inbound or outbound. Extended IP access lists should always be applied close to the source network rather than the destination. This helps to ensure that unnecessary traffic does not need to traverse a large portion of the network prior to being blocked. In this case, we'll apply access list 102 as an inbound access list on port Ethernet0, using the ip access-group command.

```
RouterA(config-if)#ip access-group 102 in
```

The show ip access-list command will show us all IP access lists defined on the router, including how many times each condition listed has been matched.

```
RouterA#sh ip access-list
Extended IP access list 102
deny tcp 192.168.20.0 0.0.0.255 host 192.168.99.1 eq telnet
permit ip any any (404 matches)
deny tcp 192.168.20.0 0.0.0.255 host 192.168.99.1
```

Standard IPX Access Lists

In the same way that access lists can be used to permit or deny IP-based traffic from passing through a router, IPX access lists control the flow of IPX traffic. A standard IPX access list is a little different than a standard IP access list. The standard IPX variety allows traffic to be filtered based on both source and destination addresses, rather than just source addresses alone.

To define a standard IPX access list, you can also use the access-list command from global configuration mode. Recall that the numeric range of standard IPX access lists is 800-899. While IPX access lists can also define individual hosts (or ranges of hosts with wildcard masks), they are more commonly implemented by specifying source and destination network numbers. Consider Figure 9-11. The small network depicted consists of one router connecting three networks – 101A, 101B, and 101C. Our goal is to deny traffic originating from network 101A from reaching network 101B, while allowing all other traffic to pass. Our access list number for this example will be 850.

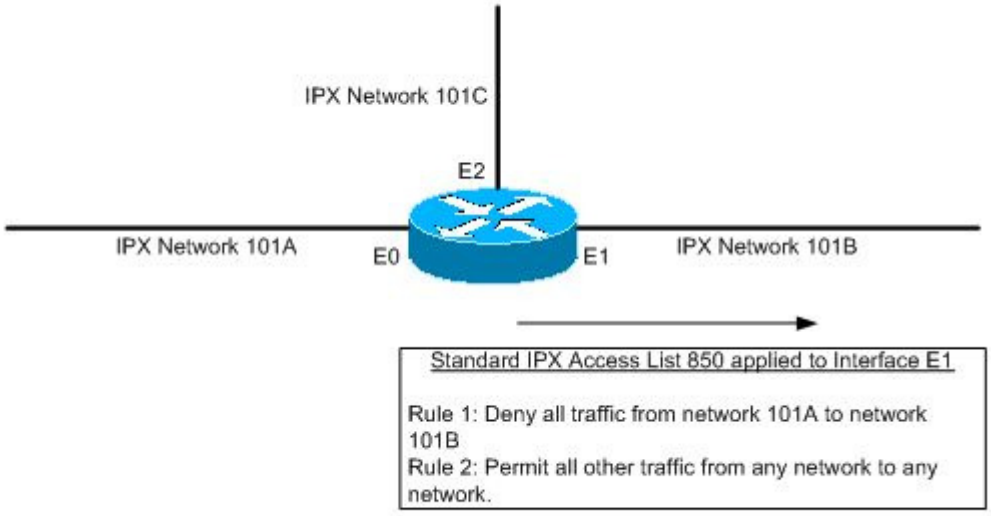


Figure 9-11: An outbound standard IPX access list applied to the router's E1 interface denies traffic from network 101A from reaching network 101B.

```
RouterA(config)#access-list 850 deny 101A 101B
```

That was certainly simple enough. The statement above identifies the access list entry as belonging to standard IPX access list 850, which denies traffic from source network 101A from reaching the destination network (101B). Our next step involves permitting all other traffic, which is imperative since all access lists end with the implicit "deny all" statement. I have used the help function to demonstrate how the syntax of an IPX access list differs when you wish to specify that "any" traffic should be allowed to pass.

```
RouterA(config)#access-list 850 permit ?
-1          Any IPX net
<0-FFFFFFFF>  Source net
N.H.H.H      Source net.host address
RouterA(config)#access-list 850 permit -1 -1
```

Notice that standard IPX access lists don't use the any keyword to reference all hosts. Instead, their syntax differs in that –1 is used to represent any IPX network. The statement above would

be the same as saying “permit any traffic from any network”. Again, this access list does nothing until applied to an interface using the access-group command. In this case, the syntax varies slightly, since it is an IPX access list that we’re dealing with.

```
RouterA(config)#int e1
RouterA(config-if)#ipx access-group 850 out
```

To view all IPX access lists defined on a router, use the show ipx access-list command. To view only a specific access list, follow the command with the access list number is question.

```
RouterA#sh ipx access-list
IPX standard access list 850
deny 101A 101B
permit FFFFFFFF FFFFFFFF
```

The IPX access lists associated with a given interface can be viewed using both show run and show ipx int followed by the interface number. I’ve truncated the output below to show only the relevant information.

```
RouterA#show ipx int e1
Ethernet1 is up, line protocol is up
IPX address is 101B.0060.5cc4.f41b, NOVELL-ETHER [up]
Delay of this IPX network, in ticks is 1 throughput 0 link delay 0
IPXWAN processing not enabled on this interface.
IPX SAP update interval is 60 seconds
IPX type 20 propagation packet forwarding is disabled
Incoming access list is not set
Outgoing access list is 850
```

Remember that any given interface may have only one incoming and one outgoing access list assigned per protocol.

Extended IPX Access Lists

In the same way that extended IP access lists give you a more granular level of control over IP traffic, extended IPX access lists allow you a finer level of control over IPX traffic. Extended IPX access lists not only allow you to filter traffic based on source and destination IPX addresses, but also IPX protocols and socket numbers. A variety of different IPX protocols and sockets exist, many of which were looked at in Chapter 4. The following example shows an extended IPX access list that denies all standard IPX ping traffic from moving from network 101A to network 101B. Recall that extended IPX access lists use the numerical range 900-999.

```
RouterA(config)#access-list 900 deny ?
<0-255>  Protocol type number (DECIMAL)
any      Any IPX protocol type
ncp      NetWare Core Protocol
netbios  IPX NetBIOS
rip      IPX Routing Information Protocol
sap      Service Advertising Protocol
spx      Sequenced Packet Exchange
```

After the deny statement, an extended IPX access list expects an IPX protocol type to be entered. For the purpose of illustration, I have chosen the entry for any, which would be similar to choosing ip in an extended IP access list.

```
RouterA(config)#access-list 900 deny any ?
<0-FFFFFFFF>  Source net
N.H.H.H       Source net.host address
any           Any IPX net
log           Log matches against this entry
<cr>
RouterA(config)#access-list 900 deny any 101A ?
<0-FFFFFFFF>  Source Socket HEXIDECIMAL
all           All sockets
cping         Cisco ipx ping
diagnostic    Diagnostic packet
eigrp         IPX Enhanced Interior Gateway Routing Protocol
log           Log matches against this entry
ncp           NetWare Core Protocol
netbios       IPX NetBIOS
nlsp          NetWare Link State Protocol
nping         Standard IPX ping
rip           IPX Routing Information Protocol
sap           Service Advertising Protocol
trace         Trace Route packet
<cr>
```

After the source address has been entered, the access list expects a source socket to be specified. An IPX socket is similar to a TCP or UDP port. In this case, I want to deny standard IPX pings from network 101A to network 101B, so I chose nping (which is socket number 9086, incidentally), as shown below.

```
RouterA(config)#access-list 900 deny any 101A nping ?
<0-FFFFFFFF>  Destination net
N.H.H.H       Destination net.host address
any           Any IPX net
log           Log matches against this entry
<cr>
```

The destination network is specified next, followed by the destination socket number.

```
RouterA(config)#access-list 900 deny any 101A nping 101B ?
<0-FFFFFFFF>  Destination Socket HEXIDECIMAL
all           All sockets
cping         Cisco ipx ping
diagnostic    IPX Diagnostic packet
eigrp         IPX Enhanced Interior Gateway Routing Protocol
log           Log matches against this entry
ncp           NetWare Core Protocol
netbios       IPX NetBIOS
nlsp          NetWare Link State Protocol
nping         Standard IPX ping
rip           IPX Routing Information Protocol
sap           Service Advertising Protocol
trace         IPX Trace Route packet
<cr>
RouterA(config)#access-list 900 deny any 101A nping 101B nping
```

After the command has been entered, this access list consists of one deny statement. Recall that we'll still need some type of permit statement to allow all other traffic to pass. The statement below will allow all other traffic to pass through the access list.

```
RouterA(config)#access-list 900 permit any any all any all
```

If you take a look at the previous example, the meaning of the statement above should become clearer. The first two any statements represent protocols and source networks. The first all statement represents all sockets. The final any all statement is equivalent to saying "any destination network, all sockets". Don't forget to apply the access list to an interface – in this case, we'll apply it close to the source network (that's how extended access lists are usually applied), as an inbound access list on interface E0.

```
RouterA(config-if)#ipx access-group 900 in
```

To view all of the ipx access lists defined on the router, use the show ipx access-list command.

```
RouterA#show ipx access-list
IPX standard access list 850
deny 101A 101B
permit FFFFFFFF FFFFFFFF
IPX extended access list 900
deny any 101A 9086 101B 9086
permit any any all any all
```

For the most part, you probably shouldn't concentrate on memorizing the syntax of extended IPX access lists. However, you should be familiar with the ways in which they are capable of filtering traffic (source address, destination address, protocol, and socket numbers) and the numeric range (900-999) that they are identified by.

IPX SAP Access Lists

You might recall from Chapter 8 that the ipx routing command also initiates the broadcasting of IPX SAP updates between routers. The information contained in SAP updates can be filtered using IPX SAP access lists, which use the numerical range 1000-1099. By properly implementing these access lists you can control the extent to which certain SAP broadcasts are propagated through an IPX network. Much like standard access lists, IPX SAP access lists should usually be implemented close to the source of SAP updates, in order to reduce unnecessary network traffic.

IPX SAP access lists are implemented using the access-list command. They specify the source network from which the broadcasts originate, the type of SAP broadcast we want to filter, and optionally, the name of the server whose broadcasts we wish to filter. Let's walk through an example step-by-step to see how these access lists are implemented. Figure 9-12 outlines a network that includes three IPX networks – 101A, 101B, and 101C. Our goal is to stop print server updates from ServerB from being added to the SAP table on RouterA.

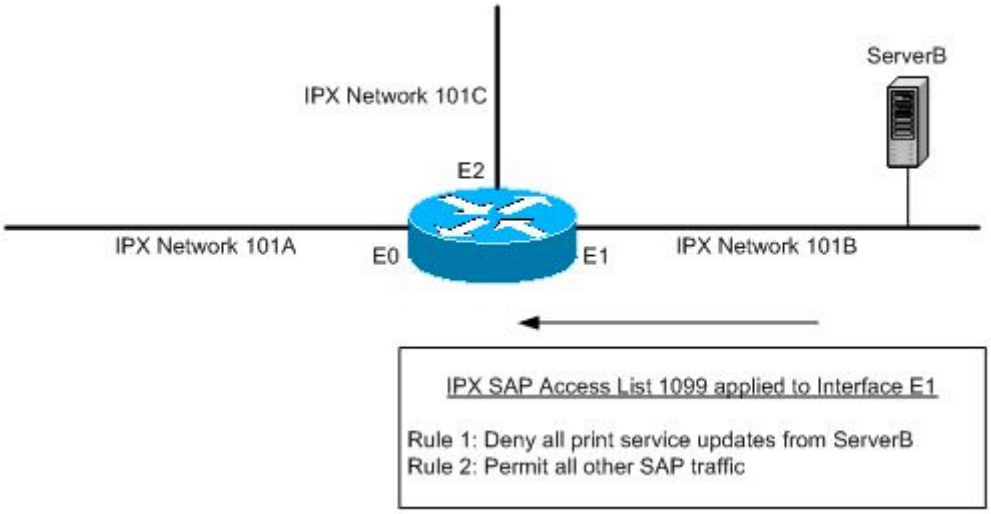


Figure 9-12: An inbound IPX SAP access list will ultimately deny ServerB's print services broadcasts from being added to the router's SAP table.

```
RouterA(config)#access-list 1099 deny ?
-1            Any IPX net
<0-FFFFFFFF>  Source net
N.H.H.H       Source net.host address
RouterA(config)#access-list 1099 deny 101B ?
<0-FFFF>  Service type-code (0 matches all services)
N.H.H.H   Source net.host mask
<cr>
```

The command above specifies that we wish to deny IPX SAP traffic from network 101B. If we had pressed enter here, all IPX SAP broadcast traffic from network 101B would be denied. Instead, we'll carry on to specify the service type code that we wish to deny. Recall from previous chapters that the type code associated with print servers is 7.

```
RouterA(config)#access-list 1099 deny 101B 7 ?
WORD  A SAP server name
<cr>
RouterA(config)#access-list 1099 deny 101B 7 ServerB
```

By specifying ServerB, we only deny SAP updates announcing print services from that particular server, as opposed to all servers on network 101B. After the command is issued, remember that the access list still ends in an implicit "deny all" statement, and still needs to be applied to an interface.

We'll add a permit statement that allows all other IPX SAP traffic. This statement is fairly simple, as shown below.

```
RouterA(config)#access-list 1099 permit -1
```

The –1 in the permit entry is the equivalent to saying "allow all IPX SAP traffic from any network".

The syntax to apply an IPX SAP filter to an interface is different than the access-group command that we used previously. Two different statements can be used to apply an IPX SAP access list to an interface, each with different results.

- **ipx input-sap-filter**. This command will apply an IPX SAP access list to an interface, and stops incoming SAP updates from being added to a router's SAP table.
- **ipx output-sap-filter**. This command also applies an IPX SAP access list to an interface, but stops filtered entries from being broadcast from that interface.

In this case, since we only have a single router and don't want the information about ServerB's print services added to the router's SAP table , we'll use an input filter on interface E1. If we had wanted the SAP broadcasts about print services on ServerB to be denied to only network 101A, an output filter on interface E0 would have been more appropriate.

```
RouterA(config-if)#ipx input-sap-filter 1099
```

To view the IPX SAP access lists associated with an interface, use the show ipx interface command, as shown below. The output has been truncated to show only relevant information.

```
RouterA#sh ipx int e1
Ethernet0 is up, line protocol is up
IPX address is 101B.0060.5cc4.f41b, NOVELL-ETHER [up]
Delay of this IPX network, in ticks is 1 throughput 0 link delay 0
IPXWAN processing not enabled on this interface.
IPX SAP update interval is 60 seconds
IPX type 20 propagation packet forwarding is disabled
Incoming access list is 900
Outgoing access list is 850
IPX helper access list is not set
SAP GNS processing enabled, delay 0 ms, output filter list is not set
SAP Input filter list is 1099
SAP Output filter list is not set
```

CCNA Study Guide Chapter 9 Summary

By Dan DiNicolo, August 13th, 2006 Posted in **CCNA Study Guide Chapter 09.** Subscribe to our

RSS Feed

Chapter 9 began with an overview of the purpose of Cisco IOS access lists, and their role in filtering network traffic. This included a look at the order in which access lists are evaluated, the different types of access lists that exist (standard and extended), as well as the different ways in which access lists are applied to router interfaces (inbound and outbound). The implicit "deny all" statement at the end of every access list was also discussed.

A look at standard IP access lists outlined their ability to filter traffic based on the source IP address of a packet. The access-list command was introduced for the purpose of adding entries to an access list, as was the access-group command, which is used to standard and extended access lists to an interface. A look at wildcard masking explained how groups of computers could be specified within an access list. An overview of extended IP access lists provided perspective on the more granular level of filtering control that these access list provide – by source and

destination IP address, as well as by protocol and port number. The ability to monitor IP access lists was examined by looking at the command used to view access lists defined on the router, as well as those applied to interfaces.

IPX standard and extended access list were looked at next. Standard IPX access lists can filter traffic by source or destination address. Extended IPX lists provide a finer level of control, allowing traffic to be filtered by protocol and socket number if necessary. An overview of IPX SAP access lists showed how SAP traffic could be filtered on a router according to a source address, service type code, and even a server name. The difference between the input-sap-filter and output-sap-filter commands used to apply SAP access lists to interfaces was also discussed.

Chapter 10: Router Troubleshooting

Troubleshooting and Disaster Recovery

It's a fact of life that no matter how carefully you manage your equipment, something is bound to go wrong at some point. While Cisco has a great track record of providing stable equipment with a solid operating system, there are still times when something will go wrong. When problems do occur, fixes are usually issued by Cisco as an updated IOS release.

On a day-to-day basis, especially while studying, many of the problems that you will come across will be related to managing the IOS image and passwords. For example, it's important to be familiar with how to gain access to a router when you've forgotten (or perhaps were never told) one of the required passwords. Along the same lines, you will need to know what to do in cases where your router's IOS image is corrupted or missing.

Troubleshooting and password recovery of a Cisco router requires an understanding of the different working environments provided by the router, and how to reach them. In order to successfully access these environments, you'll need to be familiar with what is known as the configuration register and how it impacts the operation of your router. Not only does this setting control how a router boots, but also the ability to issue break sequences, configure console port speeds, and so forth.

The topics that we'll cover in this chapter include:

- The Cisco router boot process
- Router environments
- Understanding configuration register values
- Changing configuration register values from different environments
- Password recovery on Cisco routers
- Restoring missing or corrupted IOS images

The Cisco Router Boot Process

Way back in Chapter 6, we took an introductory look at how a Cisco router boots by default, according to factory-configured settings. However, you will also need to know how to alter settings such that a router can boot into different environments, if necessary. For example, if a router password is lost or misplaced and you can't log into the router, you will need to alter the boot process such that settings stored in the startup configuration file are ignored. Doing so requires that you change the configuration register settings on your router. This presents a problem – if you can't log on to the router, how you change its configuration? The answer involves understanding a little bit more about two different operating environments – ROM Monitor and RxBoot (Boot Image) mode.

ROM Monitor Mode

As its name suggests, ROM Monitor is stored in ROM and is implemented in firmware. ROM Monitor mode is actually the bootstrap program that we talked about in previous chapters. The bootstrap program is responsible for initializing hardware and loading the Cisco IOS. As such, it is the first thing loaded by the router at power up.

The bootstrap program isn't limited to just these tasks. It is also capable of providing a command-line environment that can be used to perform certain configuration tasks, such as downloading software over the console port, recovering a lost password, or changing the configuration registers that control the way in which a router will boot. ROM Monitor mode is accessed by issuing what is known as a break sequence, either when the router is booting, or potentially during normal operation.

In order to access ROM Monitor mode using a break sequence, you need to be physically connected to the console port. This is for security purposes, since we'll see that this mode can be used to change the configuration register of the router. Issuing the break sequence on a router involves pressing a certain key combination, which differs depending on the terminal emulation program that you are using to access the router via the console port. Table 10-1 outlines the break sequences for common terminal emulation software.

Table 10-1: Common break sequences for terminal emulation programs.

Program	Break Sequence
HyperTerminal	Ctrl+Break (Windows 2000) Ctrl+6+Break (Windows 95)
Minicom (Linux)	Ctrl+a, f
ProComm Plus	Alt-b

The information in Table 10-1 is provided mainly for references purposes – you certainly won't need to remember the break sequences for all software. Since most users will use the version of HyperTerminal included with Windows, it is worth noting that the break sequence may not work with the version included with Windows NT.

Tip: An updated free version of HyperTerminal for personal use can be downloaded from the Hilgraeve website at http://www.hilgraeve.com/htpe/

To access ROM Monitor mode, issue the break sequence in the first 60 seconds of the router boot process. In other words, power-cycle (reboot) the router, and while connected to the console port in HyperTerminal, press Ctrl+Break. Issuing the break sequence and accessing ROM Monitor mode will generally provide output similar to what is shown below.

```
19:44:26: %SYS-5-RELOAD: Reload requested
System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
Copyright (c) 1986-1995 by cisco Systems
2500 processor with 16384 Kbytes of main memory
Abort at 0x10EA880 (PC)
>
```

After issuing the break sequence, you are presented with the abort message, followed by a prompt. The prompt will differ depending on the type of router you are working with. On a Cisco

2500, the ROM Monitor prompt is simply a flex bracket, as shown in the output above. On a Cisco 1600 or 2600, the ROM Monitor prompt is a little easier to recognize:

```
rommon 1>
```

There's no need to worry about the number following the rommon> prompt. It will simply increment by 1 each time you press enter. The syntax of commands issued from ROM Monitor mode differs depending on whether you're working from the > or rommon> prompt, as we'll see a little later in the chapter. For now, it's enough to know how to access ROM Monitor mode and be able to recognize the associated prompts.

The Boot Image

Recall from Chapter 6 that the read-only memory (ROM) on a Cisco router also includes a limited IOS version that can be used to boot the router in cases where an IOS image is not present. This limited IOS version is commonly referred to as the boot image (or RxBoot) and provides an environment from which you can access a TFTP server to install a new IOS image to your router. If you ever accidentally erase the contents of flash, you'll definitely need to be familiar with the boot image.

There are two common methods of purposely accessing a router's boot image stored in ROM. The first is to add the boot system rom command to a router's startup configuration. However, it's much more common to access RxBoot by changing the value of the router's configuration register, as we'll explore shortly. If your router automatically loads the boot image, that usually indicates that you need to change the configuration register on your router, or that a valid IOS image could not be found in Flash memory or on a network TFTP server. In all cases, accessing the boot image will require that you reboot the router.

You can recognize that the router has loaded the boot image according to the prompt displayed. The prompt when the boot image is loaded will be:

```
Router(boot)>
```

No matter how badly you have misconfigured a router, you should always be able to access the boot image since it is stored in ROM. If you can't, that likely indicates a hardware-related issue.

Boot Options

By default, a Cisco router uses a configuration register setting that tells it to boot using the first IOS image stored in Flash memory. If a valid IOS image isn't found in Flash, it will then attempt to "netboot" using an IOS image stored on a TFTP server. If even this fails, the default configuration register tells the router to boot using the image stored in ROM. This default behavior of the router can also be changed by specifying boot system commands in a router's startup configuration file.

For example, it's quite possible that your system has enough Flash memory to store multiple IOS images. If this is the case, you can specify the image that should be loaded when the router boots by using the boot system command, followed by the IOS filename of the image you wish the router to load. The boot system commands also allow you to configure your router for fault-tolerant booting. For example, you can specify multiple boot system commands that will form the order in which your router will attempt to load an IOS image. The following example shows a router configured to boot from an image in Flash, followed by an image stored on a TFTP server, and if neither is available, into the boot image stored in ROM.

```
Router(config)#boot system flash c2500-js-l.120-7.T
Router(config)#boot system tftp c2500-js-l.120-7.T 192.168.1.100
Router(config)#boot system rom
```

After issuing the boot system commands, they can be viewed using show run. Don't forget that changes need to be saved using copy run star.

```
Router#sh run
Building configuration...
Current configuration:
!
version 12.0
service timestamps debug uptime
service timestamps log uptime
no service password-encryption
!
hostname Router
!
boot system flash d2500.bin
boot system tftp d2500.bin 192.168.1.100
boot system rom
enable secret 5 $1$.vqP$WjeX3EZhdUnY80YHXbeNh0
```

Understanding Configuration Registers

When it comes time to troubleshoot or initiate password recovery procedures on a router, one element that you'll definitely need to be familiar with is the software configuration register. The configuration register is a 16-bit number, represented in hexadecimal, which controls everything from the way in which a router boots to whether or not it will process the contents of the startup configuration file. By default, the configuration register on a router is set to a value of 0x2102. The "0x" portion represents the value as hexadecimal, while the "2102" portion is the hexadecimal representation of the 16-bit value.

In order to appreciate what the configuration register value does, you'll need to know a little more about how the number is defined. The configuration register is made up of 16 bits, numbered 15 through 0 moving from left to right, as shown below. Recall that each hexadecimal digit is made up of 4 bits. Figure 10-1 shows the default configuration register, 0x2102, represented in hexadecimal and binary, with the associated bit numbers labeled.

Configuration Register Value (hexadecimal)	2				1				0				2			
Configuration Register Value (binary)	0	0	1	0	0	0	0	1	0	0	0	0	0	0	1	0
Bit Number	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0

Figure 10-1: Configuration register values in hexadecimal and binary, including bit numbers.

Each bit, or group of bits, provides special instructions that control certain properties of the router. For example, bits 0 through 3 control how the router boots. By manipulating these values, you can configure the router to boot from Flash, ROM, or into ROM Monitor mode. Bit 6 controls whether or not the startup-config file is processed when the router boots. As you'll see later in the chapter, this bit serves an important purpose if you've forgotten (or don't know) the enable secret password.

Table 10-2 outlines the purpose of the bits in the configuration register. Changing some of these values is much more common than others – I will show some examples shortly. For now, it's important to understand the purpose of each bit. Remember that a bit can have only two possible values – 0 or 1. "Setting" a bit involves making its value 1 rather than 0. The table below describes what happens when individual bits are changed. Hexadecimal values will be looked at shortly.

Table 10-2: The purpose of configuration register bits.

Bit Number(s)	Purpose
0 through 3	These four bits form what is known as the boot field. Depending on their value, they tell the router boot using the commands found in the startup configuration file, or into the boot image or ROM Monitor
6	This bit is used to control whether or not the startup-configuration file stored in NVRAM should be ignored when the router boots.
7	This bit is used for OEM testing.
8	This bit controls whether the break sequence is enabled or disabled.
10	When this bit is set, the IP broadcast address uses all 0s instead of all 1s.
11-12	These two bits control the speed of console connections. Recall that the default console speed in 9600 bps.
13	Bit 13 is used to control whether the router will boot from ROM (the boot image) if a network boot fails.
14	When this bit is set, IP broadcasts do not include network numbers.
15	When bit 15 is set, the router displays diagnostic messages, and the startup configuration file stored in NVRAM is ignored.

I'm the first to admit that looking at configuration register settings can be a little confusing, so let's take a closer look. The best place to start is with the default configuration register value of 0x2102.

Tip: Be sure to take the time to familiarize yourself with configuration register values, both for the CCNA/CCDA exams, and for troubleshooting purposes in real life.

Take a look at the boot field alone (bits 0 through 3), as shown in Figure 2. Since the field is made up of 4 bits, it can hold any hexadecimal value from 0 to F. Remember how hexadecimal values are converted – each group of 4 binary digits is converted to a single hex digit. For example, if the bits are set to 1010, that equals 10 in decimal, or A in hex. If you've forgotten how to convert binary to decimal, recall that 1010 is the equivalent to saying 8+0+2+0, which equals 10. In hex, the letter A represents the number 10.

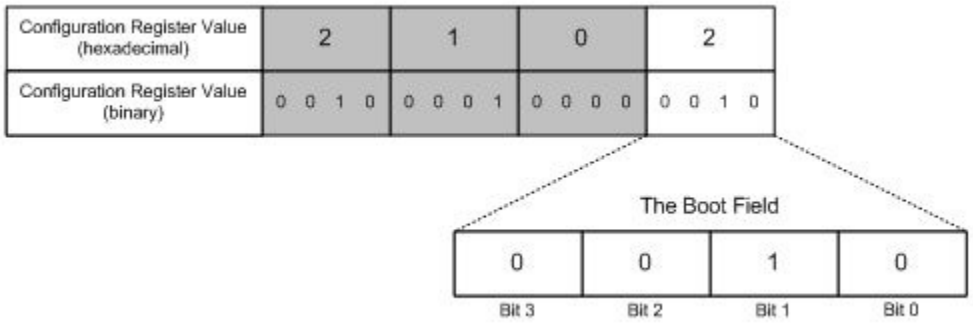


Figure 10-2: Bits 0 through 3 form the boot field of the configuration register.

Bits 0 through 3 form the last hex digit of a configuration register, as shown in Figure 10-2. The value found in this hex digit controls where the router will boot from. Table 10-3 outlines the values associated with booting from different locations.

Table 10-3: Meanings associated with different boot field values.

Hex Value	Purpose
0	When the boot field is set to a hex value of 0, the router will boot into ROM Monitor mode. An example would be a configuration register of 0x2100.
1	When the boot field is set to a hex value of 1, the router will boot using the boot image stored in ROM (RxBoot). An example would be a configuration register of 0x2101.
2-F	When the boot field is set to any value between 2 and F, the router will boot using the boot system commands found in the startup configuration file stored in NVRAM. Examples would be a configuration register of 0x2102, 0x2108, 0x210F, and so forth.

The next important bit to consider is bit number 6, as shown in Figure 10-3. Bit 6 controls whether or not the router will ignore the contents of the startup configuration file stored in NVRAM during the boot process. By default, bit 6 is not set, meaning that the router will indeed process the contents of NVRAM.

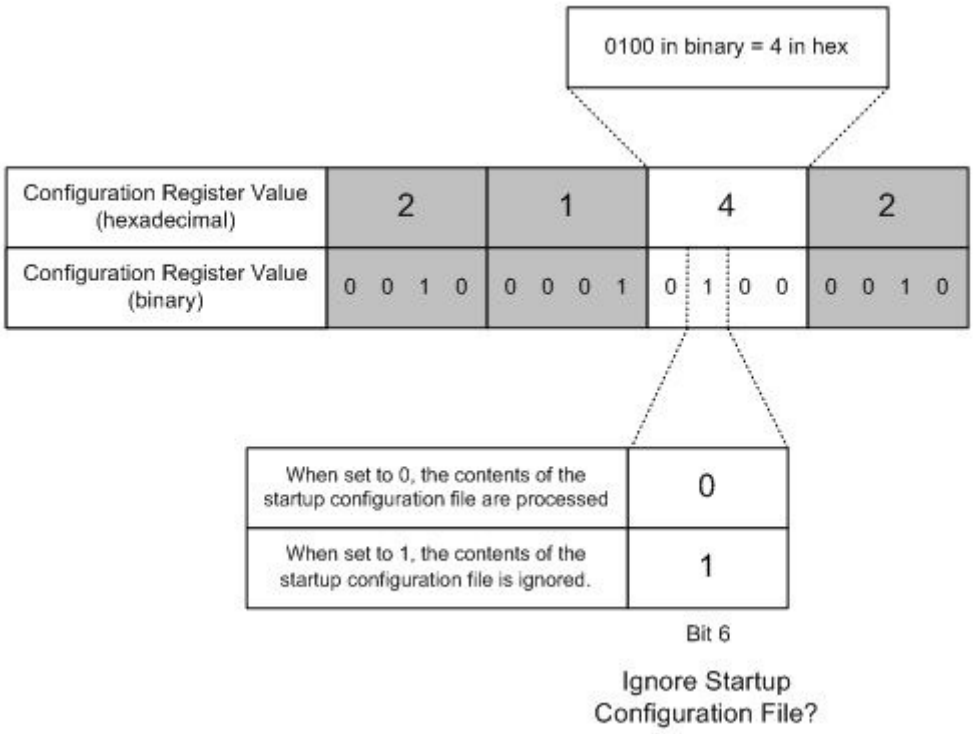


Figure 10-3: When bit 6 is set to binary 1 the contents of NVRAM are not processed.

In Figure 10-3, you also see the change that takes place in the configuration register value when bit 6 is set. This changes the configuration register value to 0x2142, which tells the router to ignore the contents of NVRAM when booting. If you do this, the router won't have a startup configuration, and you'll be presented with the System Configuration Dialog that we looked at in Chapter 6. By ignoring the startup configuration file, you then have the opportunity to set a new router password. We'll walk through the procedure a little later in the chapter.

Going back to the default configuration register of 0x2102, take a look at bit 8, as shown in Figure 10-4. Bit 8 is used to control whether the break sequence for a router is enabled or disabled while the router is running. By default, the break sequence is disabled, because that bit 8 is set to binary 1. What that means is that you cannot issue the break command during normal router operation. Recall that the break sequence is used to enter ROM Monitor mode.

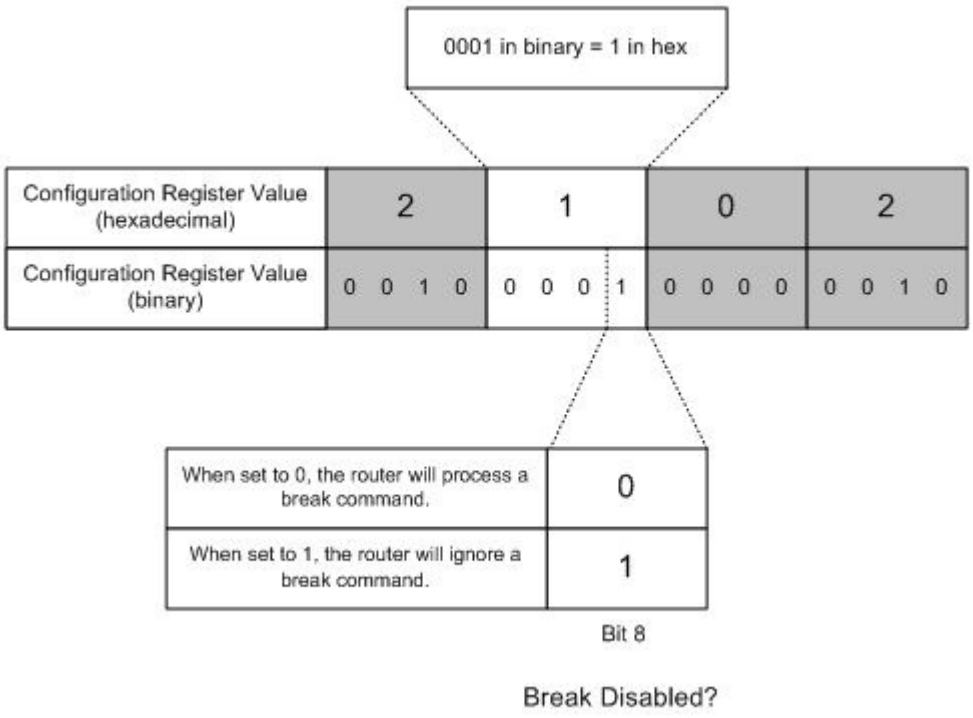


Figure 10-4: When bit 8 is set to binary 1 the break command is disabled.

Although the break sequence is turned off by default, this still doesn't stop you from entering the break sequence when a router is rebooting. As a general rule, you have anywhere between the first 15 and 60 seconds of a reload to issue the break command and enter ROM Monitor mode. Bit 8 only controls whether the break sequence can be entered during normal router operation. If the configuration register were set to 0x2002, the break command could be issued from the console at any time.

Bits 11 and 12 are used to control the console port line speed. By default, both of these bits are set to 0, which sets the line speed to 9600 bps. Table 10-4 outlines the line speeds associated with different values for these two bits. You don't need to remember these values for your exams, but they are worth noting in case you're having trouble making console connections.

Table 10-4: Different console port line speeds can be configured by changing bits 12 and 11.

Line Speed	Bit 12 Value	Bit 11 Value
9600	0	0
4800	0	1
2400	1	1
1200	1	0

The last critical bit to be familiar with is bit number 13, as shown in Figure 10-5. When bit 13 is set, a router will boot into the image stored in ROM (RxBoot) after netboot fails. Recall that when boot system commands are not stored in a router's startup configuration file, the router will first try

booting from Flash, then a TFTP server, and finally the boot image stored in ROM. If bit 13 is set to 0, the router will continue to attempt to load a configuration file from a TFTP server indefinitely.

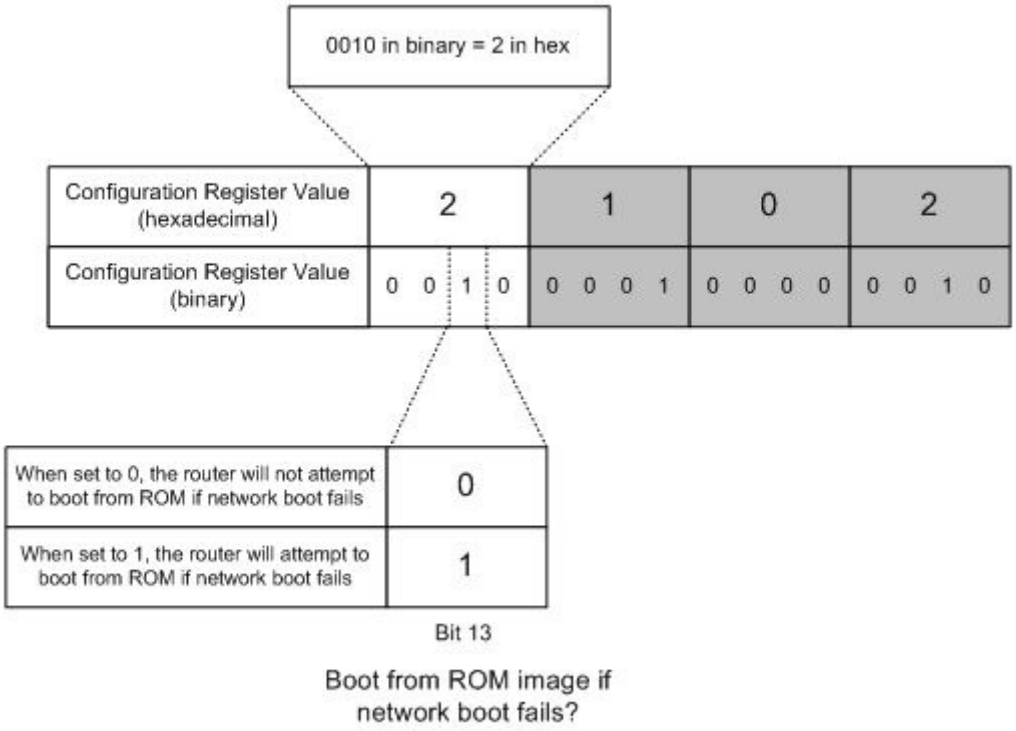


Figure 10-5: When bit 13 is set to binary 1 a router will attempt to boot from ROM if an image cannot be located on a TFTP server.

Changing Configuration Register Settings

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 10.** Subscribe to our

RSS Feed

Now that you understand a little more about configuration register settings, let's take a look at how current register settings can be identified, and then changed.
The easiest way to determine a router's current configuration register setting is by using the show version command. The configuration register setting will appear at the end of the command's output, as shown below.

```
Router#show version
Cisco Internetwork Operating System Software
IOS (tm) 2500 Software (C2500-D-L), Version 12.0(5), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Tue 15-Jun-99 20:08 by phanguye
Image text-base: 0x0000144C, data-base: 0x00637308
ROM: System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
```

```
BOOTFLASH: 3000 Bootstrap Software (IGS-RXBOOT), Version 10.2(8a), RELEASE SOFTW
ARE (fc1)
Router uptime is 2 hours, 50 minutes
System restarted by power-on
System image file is "flash:d1205.bin"
cisco 2500 (68030) processor (revision A) with 16384K/2048K bytes of memory.
Processor board ID 02265778, with hardware revision 00000000
Bridging software.
X.25 software, Version 3.0.0.
1 Ethernet/IEEE 802.3 interface(s)
1 Serial network interface(s)
1 PCbus interface(s)
32K bytes of non-volatile configuration memory.
4096K bytes of processor board System flash (Read/Write)
Configuration register is 0x2102
```

A router's configuration register settings can be changed from global configuration mode using the config-register command, followed by the new value. For example, let's say that we wanted to configure our router such that the break command is enabled at any time. That would involve setting the configuration register value to 0×2002, changing the value of bit 8 from a 1 to a 0.

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#config-register ?
<0x0-0xFFFFFFFF> Config register number
Router(config)#config-register 0×2002
Router(config)#^Z
Router#show version
Cisco Internetwork Operating System Software
IOS ™ 2500 Software (C2500-D-L), Version 12.0(5), RELEASE SOFTWARE (fc1)
Copyright (c) 1986-1999 by cisco Systems, Inc.
Compiled Tue 15-Jun-99 20:08 by phanguye
Image text-base: 0×0000144C, data-base: 0×00637308
Configuration register is 0×2142 (will be 0×2002 at next reload)
```

The output of the show version command has been truncated. Notice that the configuration register value is still 0×2102, but will be 0×2002 at the next reload. It is not necessary to issue the copy run star command in order to save changes to the configuration register value.

In cases where you cannot access a router because a valid IOS image cannot be found (or you've forgotten the router's password), you will need to change the configuration register value from ROM Monitor mode. Recall that to access ROM Monitor mode, you'll need to be connected to the router via the console port to issue the break sequence. The sequence of commands to change the configuration register will differ, depending upon whether your ROM Monitor prompt is > (like on a 2500) or rommon 1> (like on a 2600).
To change the configuration register value from ROM Monitor mode on a Cisco 2500, reboot the router and issue the break sequence (Ctrl+Break if using HyperTerminal). This will provide you with the > prompt, as shown below.

```
19:44:26: %SYS-5-RELOAD: Reload requested
System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
Copyright (c) 1986-1995 by cisco Systems
2500 processor with 16384 Kbytes of main memory
Abort at 0x10EA880 (PC)
>
```

In the Cisco 2500's ROM Monitor mode, the command o shows the current configuration register settings, as well as the purpose of the various bits.

```
>o
Configuration register = 0x2002 at the last boot
Bit# Configuration register option settings
15 Diagnostic mode disabled
14 IP broadcasts do not have network numbers
13 Boot default ROM software if network boot fails
12-11 Console speed is 9600 baud
10 IP broadcast with ones
08 Break disabled
07 OEM disabled
06 Ignore configuration disabled
03-00 Boot file is cisco2-2500 (or 'boot system' command)
>
```

The command to modify the configuration register value is o/r, followed by the new register value. After pressing enter, use the command i to initialize the router. The i command reloads the router, using the new configuration register setting. In this case, a configuration register value of 0×2102 ensures that the router will boot as per the factory default settings.

```
>o/r 0x2102
>i
```

If you issue the break sequence on a Cisco 2600 router, you'll be presented with the rommon> prompt. Although the commands from this prompt are a little different, they achieve the same result. In the example below, we're setting the configuration register to 0×2142, which tells the router to ignore the contents of the startup configuration file, since bit 6 has been set. The command to change the configuration register from the rommon> prompt is confreg, followed by the new register value. To reload the router, issue the reset command.

```
rommon 1>confreg 0x2142
rommon 2>reset
```

After issuing the reset command, the router will reboot using the new configuration register value of 0×2142.

Common configuration register settings and their meanings:

0×2102 The default configuration register setting. The break command is disabled, the contents of NVRAM are processed, and the router will boot according to the commands stored in NVRAM

0×2101 The break command is disabled. The router will process the contents of NVRAM, but will boot into the RxBoot image stored in ROM.

0×2100 The break command is disabled, and the router will boot into ROM Monitor mode.

0×2142 The break command is disabled, and the router will ignore the contents of NVRAM during the boot process.

0×2002 The break command is enabled, but otherwise the router will boot normally.

Cisco Router Password Recovery

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 10.** Subscribe to our

RSS Feed

If one thing is for certain, it's that at some point you'll forget the password that you assigned to a router, or be asked to configure a router whose password you cannot be provided with. The good news is that with physical access to the console port, you're in luck. Now that you know about configuration register settings, you know that you can change its setting to ignore the startup configuration file on a router, thus allowing you to bypass any passwords. Once you reboot the router, you can then change (and save) new passwords.

For the purpose of this example, let's assume that we're using a Cisco 2600 router. The steps on a Cisco 2500 are similar, with the exception of the ROM Monitor mode commands.

The first step is to access ROM Monitor mode and changing the configuration register setting to 0×2142, such that the router will ignore the contents of the startup configuration file. After rebooting the router, issue the break sequence, and then enter the confreg and reset commands.

```
rommon 1>confreg 0x2142
rommon 2>reset
```

The router should now reload, ignoring the contents of the startup configuration file. Press Enter to access user mode, and then enter privileged mode using the enable command. Notice that no password is required.

```
Press RETURN to get started.
Router>enable
Router#
```

Now that we've accessed privileged mode, the next step is to overwrite the current running configuration with the information stored in the startup configuration. Loading this configuration into RAM will allow us to change the password, as well as to save it.

```
Router#copy star run
```

Be very careful not to mistakenly issue the copy run start command - that would cause you to lose all configuration settings stored in the startup configuration file.

Even though we've copied the startup configuration into RAM, all of the interfaces are still shutdown. To verify this, use the show run command. As such, your last steps will be to access all interfaces and issue the no shutdown command. Our main goal is still to change the enable secret password, so that's our next step.

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#enable secret cisco
```

After setting our new enable secret password (and any other passwords that we may need changed), issue the no shutdown command for all interfaces, and then change the configuration register back to the default value (or whichever value you require) using the config-register command. In this case, we'll set the register back to 0×2102, and then issue the all-important copy run star command to save our changes.

```
Router(config)#config-register 0x2102
Router(config)#^Z
Router#copy run star>/code>
```

And there you have it. You don't even require a reboot at this point (assuming that you remembered to issue the no shutdown command for all necessary interfaces). The next time the router does reboot, its configuration register will be set to 0×2102. This means that it will not ignore the startup configuration file, allowing you complete and normal access using our newly configured password.

Restoring a Missing or Corrupt IOS Image

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 10.** Subscribe to our

RSS Feed

There may also come a time where you router doesn't have a valid IOS image in Flash memory - perhaps because you have accidentally erased Flash, upgraded your router's Flash memory, or the image contained in Flash has become corrupted. This leaves you with a couple of options, one of which is to use ROM Monitor mode to download a new IOS image to the router over its console port. Unfortunately, at a speed of 9600 bps, this will take a fairly long time. A more common way to restore an IOS image to Flash is to use the boot image (RxBoot) to download a new image to the router from a TFTP server.

This means that you'll need a properly configured TFTP server that contains a valid IOS file to begin with. The next step involves booting the router into the boot image, which may mean that you will need to first change the router's configuration register via ROM Monitor mode. After booting into the boot image, you can then download an IOS image to your router via TFTP.

For the purpose of this illustration, we'll assume that our router is a Cisco 2500. Our goal is to install a new IOS image on a system that currently has no IOS image in Flash. We'll start by accessing ROM Monitor mode to change the router's configuration register. Recall that the configuration register value to boot the system from the boot image in ROM ends with the hex digit 1. As such, we'll change the register to 0×2101.

```
19:44:26: %SYS-5-RELOAD: Reload requested
System Bootstrap, Version 5.2(8a), RELEASE SOFTWARE
Copyright (c) 1986-1995 by cisco Systems
2500 processor with 16384 Kbytes of main memory
Abort at 0x10EA880 (PC)
>o/r 0x2101
>i
```

After issuing the i command, the router will reset and boot from the boot image, as specified by the new configuration register value. Once it finishes booting, you should find yourself at the Router(boot)> prompt.

The first step is entering enable mode. Although the router doesn't have a valid IOS image in Flash, it may still have a valid startup configuration file in NVRAM. As such, you may still be prompted for a password. To avoid the startup configuration file as well, you would need to set the configuration register to 0×2141.

It is also worth testing whether or not the router can ping the TFTP server you intend to download the new IOS image from. If it can't, you may need to change (or add) an IP address of a router interface.

```
Router (boot)>en
Password:
Router(boot)#ping 192.168.1.21
Type escape sequence to abort.
Sending 5, 100-byte ICMP Echoes to 192.168.1.21, timeout is 2 seconds:
!!!!!
Success rate is 100 percent (5/5), round-trip min/avg/max = 1/200/1000 ms
```

In this case, the ping was successful, meaning that the router is capable of communicating with the TFTP server. The next step is issuing the copy tftp flash command. This prompts us for the IP address of the TFTP server, as well as the name of the IOS image that we wish to download to Flash. If Flash memory already contains IOS images, they may need to be erased as well.

```
Router (boot)#copy tftp flash
Address or name of remote host [255.255.255.255]? 192.168.1.21
Source file name? d2500.bin
Destination file name [d2500.bin]?
Erase flash device before writing? [confirm] yes/no yes
```

I've saved some space by not showing the output of the command - those exclamation points that you usually see when copying files to or from the router. Once the copy operation is complete, there is still an additional step to undertake – changing the configuration register back to our original value, 0×2102. Follow this up with the reload command, being sure to think carefully about whether you want to save any changes that you have made to the running configuration file. If your startup configuration file already contains correct settings, make a point of not saving any changes when prompted.

```
Router(boot)#configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(boot)(config)#config-register 0x2102
Router(boot)(config)#^Z
Router(boot)#reload
System configuration has been modified. Save? [yes/no]
```

With the router configuration register changed back to 0×2102, the router should boot into the recovered IOS image normally, using the startup configuration file stored in NVRAM.

CCNA Study Guide Chapter 10 Summary

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 10.** Subscribe to our

RSS Feed

Chapter 10 began with a look at the boot process of a Cisco router, including an overview of the different environments into which a Cisco router can be booted – ROM Monitor mode, the boot image stored in ROM, or a normal IOS image stored in Flash memory or on a TFTP server. The break sequences to access ROM Monitor mode for different applications were also discussed, as

were the boot system commands that can be added to a router's startup configuration file to control the startup environment.

This was followed by a look at a router's configuration register value, and the impact that this value has on how a router behaves during the boot process. A variety of different elements were looked at in this section, including how to enable or disable the break sequence, bypass the startup configuration file and, most importantly, the values associated with the boot field.

The act of changing a router's configuration register was looked at next. This included a look at the commands to change the register value from the regular IOS image, as well as from ROM Monitor mode, on both Cisco 2500 and 2600 routers. While both routers accomplish the same goal in two steps, the commands used in ROM Monitor mode are different.

The chapter ended with a look at troubleshooting, both for the purpose of password recovery and restoring a missing or corrupted IOS image. The configuration registers played a key role in both processes, allowing the startup configuration file to be bypassed in the case of changing a password, while the boot image was used to download a new IOS image from a TFTP server.

Chapter 11: Wide Area Network (WAN) Technologies

Wide Area Network (WAN) Technologies

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

By and large, the LAN marketplace is dominated by one technology – Ethernet. When it comes to interconnecting multiple locations, however, a variety of different Wide Area Network technologies are available. The decision to use one of these technologies over another is usually based on a number of factors including cost, performance, reliability, and availability. In some cases, a company's WAN will not be based on a single technology, but many. For example, an organization may choose to connect some offices in a permanent fashion, while having other offices connect using various dial-up techniques. In this chapter, you'll take a look at many of the most common WAN technologies that you'll need to be familiar with for the CCNA and CCDA exams, including those based on point-to-point, circuit switched, and packet-switched connections.

The material to be covered in this chapter includes:

- Introduction to WAN technologies
- High Level Data Link Control Protocol (HDLC)
- Point-to-Point Protocol (PPP)
- Integrated Services Digital Network (ISDN)
- Frame Relay

Introduction to WAN Technologies

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

A Wide Area Network (WAN) is best described as a data network that covers a relatively broad geographic distance. Unlike a LAN, which is usually localized within a relatively small area such as a single office, building, or small campus, a WAN will typically span distances of anywhere from a few, to thousands of kilometers. Besides distance, LANs and WANs are generally differentiated by network ownership. With a LAN, companies typically own and operate all of the equipment that interconnects devices – switches, routers, wiring, and so forth. This is usually not the case with a WAN, where a service provider (such as a local telecommunications carrier) generally owns the network links and switching equipment.

In order to interconnect geographically dispersed locations, companies will usually provision services from a telecommunications carrier, generally renting or leasing links on a monthly basis.

The speed and cost of these links can vary greatly, depending upon bandwidth requirements, distances to be spanned, and available technologies. As a general rule, WANs are implemented at speeds much slower than their LAN counterparts, with higher bandwidth requirements significantly increasing monthly costs. This is a function of simple economics – providers have made huge investments in building their networks and are doing their best to make money. For companies, choosing the most appropriate WAN technology involves finding a cost-effective solution that meets their performance, reliability, and scalability requirements.

WAN Technologies and the OSI Model

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

WAN technologies are considered to exist and function at the three lower layers of the OSI model – Physical, Data Link, and Network. While not all WAN technologies have elements that function at the Network Layer, some (like X.25 and ISDN) do. The figure below provides an overview of how the WAN technologies that you'll look at in this chapter map to the OSI model.

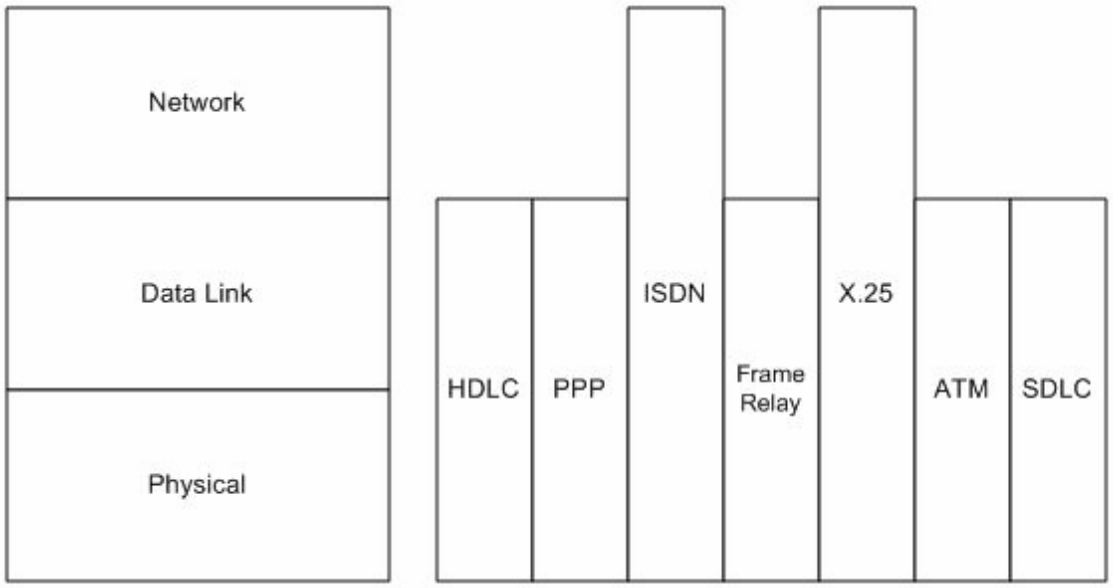


Figure: A high-level overview of how various WAN technologies map to the Physical, Data Link, and Network layers of the OSI model.

WAN Connectivity

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Anyone familiar with the concept of a WAN has no doubt seen the “cloud”. In just about any network diagram that includes WAN links or a connection to the Internet, the shape of a cloud will usually appear. In the case of a WAN, the cloud represents the switching equipment and links of the service provider network, over which data will travel between locations. The cloud is just an abstraction, meant to symbolize the fact that you don’t necessary know (or care about) the path that data takes through this equipment in reaching its destination. In other words, data leaves one site, enters the cloud, and then somehow reaches the other. The path over which our data travels within their network cloud is the responsibility of the service provider.

While functions within the service provider network aren’t usually of concern to the customer, there are still some terms and concepts relating to them that you should be familiar with. Consider the network outlined in the figure below. In it, a company with two locations has WAN link connecting to their two offices. At this point, the actual WAN technology in use isn’t important.

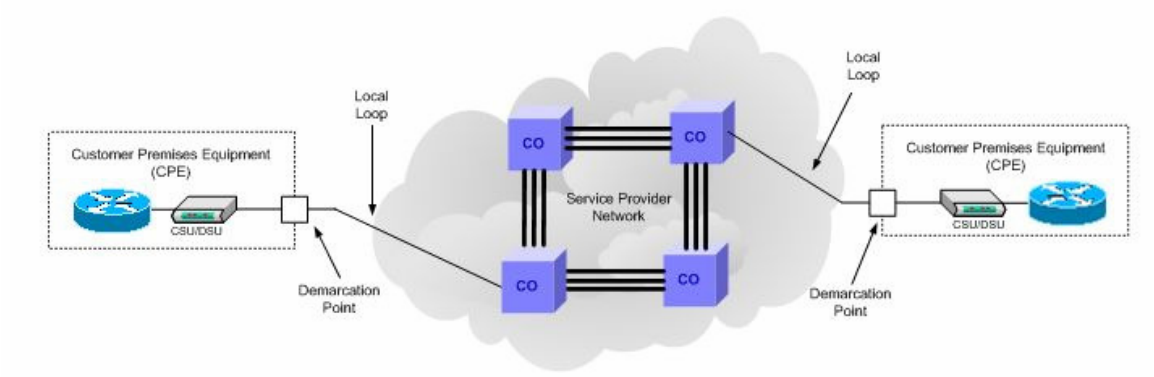


Figure: Network diagram showing the physical connections to and within a service provider’s network.

The physical connection between the locations shown in Figure 11-2 consists of a number of elements. At both company offices, the service provider will install a connection point (usually in the form of an RJ-45 jack) that physically connects a circuit to their nearest switching office. This jack is known as the demarcation point, and represents the point at which the service provider’s responsibility is said to end – in order words, the provider will ensure that the link functions correctly up to that point. The other end of this link ultimately connects to the service provider’s nearest switching facility, also known as the Central Office (CO). In general, a CO will be located within approximately 5 kilometers of the customer premises. These links (between a CO and customer premises) are part of what is known as the “local loop” or “last mile”. The local loop may consist of a variety of technologies, including fiber optics, traditional twisted pair wiring, and more.

Central Offices act as distribution points within a service provider network. In any given city, there will be many COs, interconnected to one another using high-capacity trunk links (usually fiber optics). While these links will typically connect COs to one another, some COs also serve as interconnection points to other service providers, such as national or international carriers. WAN connections that need to use the facilities of multiple service providers generally incur higher monthly fees than those that stay within a single provider’s network.

On the customer’s side of the demarcation point is where you will find what is referred to as Customer Premises Equipment (CPE). The term CPE is often used quite loosely, but traditionally refers to equipment that is owned and operated by the customer for the purpose of connecting to the service provider’s network. However, the term CPE can also be used to describe just about any piece of equipment that resides at the customer’s location. Furthermore, many companies choose to lease equipment (such as routers or DCE devices) from their service providers – this

equipment is still considered to be CPE.
Before physically connecting to a service provider network, a company needs to determine the type of WAN service or connectivity that they require. For example, a customer may want a dial-up link between locations, or an “always-on” dedicated connection. Three main WAN connectivity techniques exist, each with relative advantages and disadvantages in terms of speed, performance, and cost. These include point-to-point, circuit switched, and packet switched links.

Point-to-Point WAN Links (Leased Lines)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

The terms “point-to-point link” and “leased line” refer to the same thing when describing WAN connections – a dedicated link through the service providers’ network. This link is not shared with other customers, and provides a dedicated circuit between two locations. This isn’t to say that the link is a single physical wire. In reality, it is a permanently established circuit through the service provider’s switched network, similar to what is illustrated in the figure below.

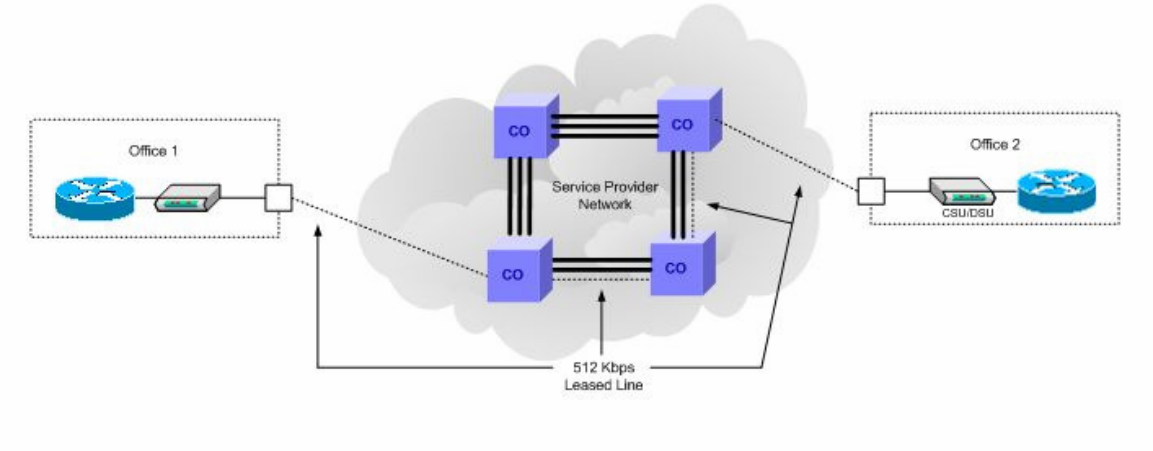


Figure: Two offices connected through a service provider’s network using a leased line.

A leased line is a good choice for customers who require consistent bandwidth between two locations. The speed of these dedicated links can vary – for example, a customer may choose a 64K leased line, or one at T1 speeds or higher. Leased lines are generally more expensive than other WAN connections, since the link is permanent and dedicated to the customer, regardless of the extent to which they use their allocated bandwidth. In other words, if a customer pays for a 128 Kbps leased line and then averages only 64 Kbps usage, they still pay for the full bandwidth that the link provides. Prices for leased line vary according to required bandwidth, and sometimes distance between locations. Leased lines will typically use either the Point-to-Point Protocol (PPP) or High Level Data Link Control (HDLC) protocol to encapsulate and send data between locations.

Circuit Switching

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Circuit switching is a WAN connectivity technique that allows circuits to be created across a network on demand, and then terminated once they are no longer required. The perfect example of circuit switching in action is a normal telephone call – after you pick up the phone and dial a number, a circuit is created between your phone and the phone of the person you are calling. The circuit is static, meaning that the path over which data (or voice) travels is the same for the duration of the call. Once you hang up, the circuit is terminated.

Circuit switching is commonly used to interconnect networks in cases where a permanent connection is not required. For example, a company might need to transfer data to a branch office just once or twice a day. In cases such as this, the cost of a permanent link wouldn't be justified. While beneficial in cases where data traffic requirements are low, circuit switched connections generally provide slower throughput rates than other technologies. Examples of circuit switched WAN technologies include traditional analog dialup links using modems and ISDN connections.

Packet Switching

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

A third and increasingly common WAN connectivity technique is known as packet switching. Unlike with leased lines, where customers pay for a dedicated link and consistent bandwidth, packet switching allows a service provider's network resources to be shared amongst many customers, which in turn reduces costs. This makes packet switching a great choice for companies whose WAN traffic is variable or "bursty" in nature.

On a packet switching network, companies still connect to the provider network as they normally would, but instead of provisioning a dedicated circuit between locations, they share bandwidth will all other customers. The theory is that at any given time, a company will not be using its fully allocated bandwidth, based on the variable nature of data traffic. This allows other companies to make use of the available bandwidth, which in turn ensures makes more efficient use of the service provider's network. Because the service provider doesn't have to provision a physical end-to-end circuit for a packet switched customer, they are able to offer the service at a lower price.

A packet switched network provides a great example of the service provider "cloud" in action. When companies connect to the cloud, the service provider generally guarantees the minimum average bandwidth they will have access to, while allowing their traffic to "burst" to higher speeds if excess bandwidth is available on the shared network. On a packet switching network, individual packets are sent from one location into the "cloud". These packets may take different paths to reach their destination, as per available bandwidth and network resources. When they arrive at their destination, they are reassembled in the correct order. In order to connect company offices, the service provider defines what are known as a "virtual circuits" between locations. Virtual circuits will be looked at in more detail later in the chapter. Common examples of packet switching WAN technologies include Frame Relay, X.25, and ATM.

WAN Hardware

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Back in Chapter 6 I discussed the difference between Data Terminal Equipment (DTE) and Data Communications Equipment (DCE). You should recall that DTE equipment is usually the source or destination of a network communication session, such as a router, computer, or terminal. In order for DTE equipment to establish communication over a service provider's data communications network, DCE equipment is required. Common examples of DCE equipment include:

Modems. A modem provides the ability to connect a DTE device to a service provider's analog communications facilities. It does this by modulating the digital signal output by a DTE device into the analog signals used on the local loop portion of the Public Switched Telephone Network (PSTN). At the receiving end, a modem demodulates the analog signal back to its digital form. While modems can be used to form a WAN connection between locations, they are typically relegated to "backup duty" because of their relatively slow (56K or less) connection speeds.

Terminal Adapters. A terminal adapter is used in ISDN communications to connect a DTE device to a service provider's ISDN network. Terminal adapters are similar in function to a modem, in that they are used to dial into a network. However, they do not convert digital signals to analog or vice versa, since an ISDN networks are completely digital.

CSU/DSUs. Channel Service Unit / Data Service Units are the modem-like devices that act as an intermediary between DTE equipment (such as a router) and the service provider's digital circuit. The CSU/DSU handles the sending and receiving of data over the service provider's circuit, as well as clocking functions.

Carrier Line Systems and Speeds

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

When provisioning WAN circuits from a service provider, it's important to have an understanding of the various terms used to describe circuit speeds and their groupings. In North America, most digital links are grouped according to what is known as the digital signal standard. The base digital signal standard is known as DS0, and represents a digital channel with 64K of bandwidth, the same amount used for a typical voice call. The North American T, European E, and Japanese Y carrier systems use the DS0 channel as their base multiple in link calculations.

For example, a T1 line is made up of 24 DS0s or channels, for a total aggregate bandwidth of 1.544 Mbps. If you do the multiplication, you'll notice that 24 channels of 64 Kbps yields only 1.536 Mbps – the "missing" 8 kilobits are used to channelize a T1 line. The list below outlines some of the common speeds associated with carrier lines in North America, Europe, and Japan.

Carrier line designations, DS0s, and speeds for North America, Europe, and Japan:

T1	24 DS0s	1.544 Mbps
T3	672 DS0s	44.736 Mbps
E1 (Europe)	30 DS0s	2.048 Mbps
E3 (Europe)	480 DS0s	34.064 Mbps
Y1 (Japan)	30 DS0s	2.048 Mbps

It is also possible to provision “fractional” service from most service providers. For example, not every company requires (or can afford) a full T1 link between locations. In order to better meet their customer’s need, most providers offer what is known as fractional T1 service. This allows customers to lease only a portion of a T1 line, usually in multiples of 64K. As such, a company could rent 4 channels of a T1 link, providing them with 256 Kbps of bandwidth in total.

High Level Data Link Control Protocol (HDLC)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

The High Level Data Link Control protocol (HDLC) is the default encapsulation used on the synchronous serial interfaces of a Cisco router. You’ll recall that synchronous serial interfaces require an external clocking device (such as a CSU/DSU) in order to synchronize the sending and receiving of data. HDLC is a superset of the Synchronous Data Link Control (SDLC) protocol that was originally developed by IBM for use in SNA environments. SDLC and SNA will be looked at in more detail later in this chapter.

HDLC is a Data Link layer protocol used to encapsulate and transmit packets over point-to-point links. It handles the transfer of data in full duplex, as well as link management functions. As an OSI standard, many vendors implement the HDLC protocol in their equipment. Unfortunately, these implementations are usually not interoperable. The reason is that when the HDLC frame format was defined, it did not include a field to identify the Network layer protocol that it was framing. As such, the OSI version of HDLC assumes that any link using HDLC is running only a single Network layer protocol like IP. Of course, many networks run IP, IPX, and other Layer 3 protocols simultaneously. This has led vendors (including Cisco) to implement HDLC using a proprietary frame format that includes a type code field, thus allowing the Network layer protocol within a frame to be properly identified.

The Cisco HDLC frame is illustrated in the figure below.

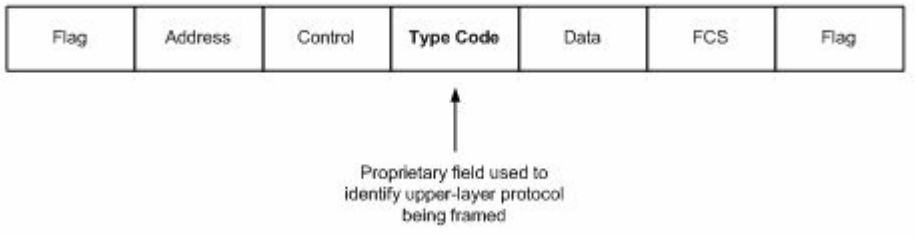


Figure: The Cisco HDLC frame format contains a Type Code field not found in the ISO standard HDLC frame.

Because of the proprietary nature of vendor HDLC implementations, you should only use HDLC framing on point-to-point links when the router at each end of a link is from the same vendor. In cases where you want to connect equipment from different vendors over a leased line, the Point-to-Point Protocol (PPP) should be used. Always remember that the router on both sides of a point-to-point link must be using the same data framing method in order to communicate.

Because HDLC is the default encapsulation method for synchronous serial interfaces on a Cisco router, it doesn't require any explicit configuration. To view the current encapsulation type used on a router serial interface, use the show interface command. The example below shows a router using HDLC encapsulation on interface S0.

```
RouterA#show int s0
Serial0 is up, line protocol is up
Hardware is HD64570
Internet address is 131.107.2.200/28
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation HDLC, loopback not set, keepalive set (10 sec)
```

If the encapsulation method used on a serial interface is ever changed, you can switch back to HDLC by issuing the command encapsulation hdlc from interface configuration mode.

```
RouterA#config t
Enter configuration commands, one per line. End with CNTL/Z.
RouterA(config)#int s0
RouterA(config-if)#encapsulation hdlc
```

Point-to-Point Protocol (PPP)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

The Point-to-Point Protocol is another Data Link layer protocol, and one that you may already be familiar with. While it can be used to encapsulate data over dedicated leased lines, PPP is also commonly used for dial-up connections to corporate networks and the Internet. As such, PPP works over both synchronous and asynchronous serial interfaces. PPP is an IETF-standard protocol capable of encapsulating a variety of upper-layer protocols including IP, IPX, and AppleTalk to name but a few. It has largely replaced an earlier standard known as the Serial Link Internet Protocol (SLIP). One of the main disadvantages of SLIP is that it only supports point-to-point connections that use IP at the Network layer.

PPP offers a wide variety of configurable options that make it a robust choice for encapsulating data over leased lines. First and foremost, PPP supports authentication, which can be used to confirm the identity of equipment or users at either end of a point-to-point connection. Both the Password Authentication Protocol (PAP) and the Challenge Handshake Authentication Protocol (CHAP) can be used for authentication on a Cisco router. PPP also supports a variety of data compression techniques including Stacker, Predictor, and Microsoft Point-to-Point Compression (MPPC). Finally, PPP provides the ability to combine multiple synchronous and asynchronous serial links such that they work as a single logical connection, a technique referred to as PPP Multilink.

PPP is comprised of multiple protocols at the Data Link Layer, each with different areas of responsibility. It can also run over a variety of different physical layer standards, although

EIA/TIA-232 (formerly RS-232) is the most commonly used. The ways in which the various PPP protocols and standards map to the OSI model is shown in the figure below.

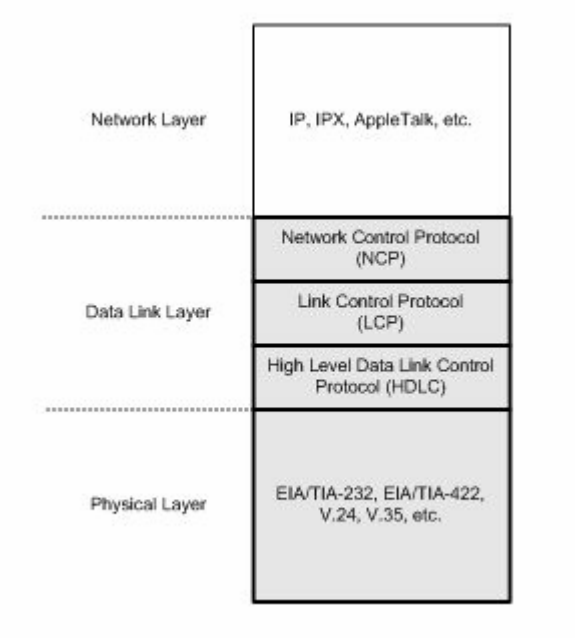


Figure: Point-to-Point Protocol (PPP) protocols and standards as they relate to the OSI model.

The three protocols used by PPP at the Data Link layer are HDLC, the Link Control Protocol (LCP), and various implementations of the Network Control Protocol (NCP). Each is described below.

- HDLC. HDLC is the data framing method used over PPP links. In the case of PPP, the OSI standard version is used rather than the Cisco proprietary version. This standardization helps to ensure that PPP implementations by different vendors can communicate properly.
- LCP. The Link Control Protocol is used for establishing, testing, configuring, and terminating PPP connections. PPP options such as authentication, compression, and multilink are all configured by LCP. The three main types of LCP frames used on a PPP connection are link-establishment, link-maintenance, and link-termination
- NCP. Network Control Protocol frames are used to negotiate and configure the Network layer protocols that can be used over a PPP session. For example, there are specialized NCPs for IP (IPCP), IPX (IPXCP), AppleTalk (ATCP), and others. NCPs allow PPP to work in conjunction with many Network layer protocols over the same link.

There are four main steps in involved in establishing, maintaining, and terminating a PPP session, as outlined below.

1. The first step in establishing a PPP session between devices involves both sending LCP link-establishment frames for configuration and testing purposes. These frames also define which options a given PPP host is using, such as compression, authentication, and multilink. If authentication is defined and required (PPP authentication is optional), it will take place during this phase.

2. The second step is optional, and uses LCP frames to test the quality of a link. The information gathered can be used to determine whether the link is capable of handling various upper-layer protocols.
3. In step 3, NCP frames are sent over a link to establish which individual Network layer protocols need to be configured. For example, a link may need to be configured to use IP, IPX, AppleTalk, and so forth.
4. When a PPP session needs to be terminated, LCP link-termination frames are used to close the connection. The third LCP frame type (link-maintenance) is used to manage and debug PPP connections.

Configuring PPP Connections

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

The configuration of PPP on Cisco routers isn't difficult, but can vary depending upon which options you choose to implement. For example, configuring PPP can be as simple as specifying that the appropriate serial interface on each router should use PPP encapsulation. This is accomplished by issuing the encapsulation ppp command from interface configuration mode.

```
RouterA#config t
Enter configuration commands, one per line. End with CNTL/Z.
RouterA(config)#int s0
RouterA(config-if)#encapsulation ?
atm-dxi ATM-DXI encapsulation
frame-relay Frame Relay networks
hdlc Serial HDLC synchronous
lapb LAPB (X.25 Level 2)
ppp Point-to-Point protocol
smds Switched Megabit Data Service (SMDS)
x25 X.25
RouterA(config-if)#encapsulation ppp
```

After configuring interface S0 on RouterA to use PPP, notice the truncated output of the show int s0 command shown below. While the encapsulation is set to PPP, the status of the port shows that Serial 0 is up, but the line protocol is down. This is because the other end of the link on my network is still configured to use HDLC encapsulation. Recall from Chapter 7 that frame type mismatches will result in this message. Notice also that the LCP status shows a REQsent message (if properly connected, this would be open), and that the NCP used to configure IP (IPCP) is also closed.

```
RouterA#show int s0
Serial0 is up, line protocol is down
Hardware is HD64570
Internet address is 192.168.2.200/28
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation PPP, loopback not set, keepalive set (10 sec)
LCP REQsent
Closed: IPCP
```

After changing the encapsulation type on the other end of the link to PPP, the output of the show int s0 command on RouterA displays the following:

```
RouterA#show int s0
Serial0 is up, line protocol is up
Hardware is HD64570
Internet address is 192.168.2.200/24
MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec, rely 255/255, load 1/255
Encapsulation PPP, loopback not set, keepalive set (10 sec)
LCP Open
Open: IPCP
```

Now that both ends of the link are configured to use PPP encapsulation, the line protocol has changed to up; LCP is open, as is the NCP for IP (IPCP). IP traffic should now be able to move across the link without a problem. If additional protocols like IPX or AppleTalk were also configured on both interfaces, their associated NCPs would also be listed.

PPP Authentication

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Cisco routers support two main authentication methods on PPP links – PAP and CHAP. One benefit of configuring PPP authentication is that it allows routers to be sure of the identity of the router at the other end of a link. PPP authentication is optional, and is often not configured on dedicated PPP links like leased lines. Recall, however, that PPP is the standard protocol used for dial-up connections. As such, a company might configure a demand-dial connection (using modems or ISDN) between two locations. In such a case, using PPP authentication would be a very good idea, in order to verify the identity of a router attempting to make a connection.

Although two authentication choices exist for PPP connections, you should make a point of avoiding the Password Authentication Protocol (PAP). PAP sends username and password information across a link in plain text, meaning that this information could be captured and read by a protocol analyzer like Ethereal or Sniffer.

A much better choice for PPP authentication is the Challenge Handshake Authentication Protocol (CHAP). CHAP uses a “challenge” technique to authenticate hosts, rather than requiring that username and password information be passed along with a connection request. This makes it a much more secure authentication method. The process begins with the authenticating router sending out a challenge request. When received by the router who initiated the connection, it will calculate a value for the challenge using a one-way MD-5 hash function, with the configured password as its input (you’ll look at configuration shortly). This hashed value will be sent back to the authenticating router. If that router has calculated the same hash value, the remote router will be successfully authenticated, and the connection is permitted. CHAP also supports mutual authentication, allowing routers at either end of a PPP connection to authenticate each other.

The configuration of PAP or CHAP requires that both routers be configured with a hostname, as well as an appropriate username and password combination for authentication purposes. The network used in this example is illustrated in the figure below.

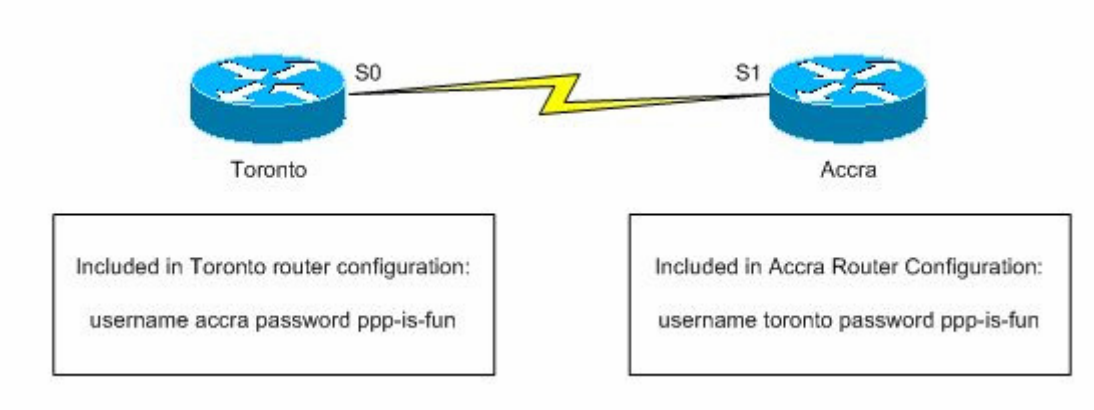


Figure: Both the Toronto and Accra routers need to be configured appropriately to use PPP authentication.

Both routers need to be configured properly in order for PPP authentication to work. For example, the username specified on the Toronto router should be the hostname of the connecting router (Accra in this case), and the passwords for both systems must be identical. The configuration of Toronto router would be as follows:

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname toronto
toronto(config)#username accra password ppp-is-fun
```

Similar steps need to be taken on the Accra router, although the username specified this time would be Toronto.

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#hostname accra
accra(config)#username toronto password ppp-is-fun
```

After these steps have been completed, an authentication mechanism needs to be specified on the appropriate PPP interfaces. For example, the configuration of CHAP on the Toronto router's S0 interface would be:

```
toronto(config)#int s0
toronto(config-if)#ppp authentication chap
Similarly, the configuration of CHAP on the Accra router would be as follows:
accra(config)#int s1
accra(config-if)#ppp authentication chap
```

The command to use PAP authentication is ppp authentication pap. In cases where both methods are configured on a PPP interface, the router will attempt to use the first authentication method specified, and will only use the second in cases where the first method fails. Again, PAP authentication should generally be avoided as a security best practice.

Integrated Services Digital Network (ISDN)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Integrated Services Digital Network (ISDN) was originally conceived as an all-digital replacement for the existing phone network. ISDN is capable of moving digital voice, data, video, and more over existing telephone wires. Unfortunately, the deployment of ISDN hasn't been nearly as comprehensive as service providers and various industry committees once thought it would be. Challenges in deploying ISDN have included different standards early in its development, the large investment required by service providers in new Central Office switching equipment, and the emergence of newer, faster technologies like DSL. Having said that, ISDN service certainly has a place in the world of Wide Area Networking, and is a popular choice for dial-on-demand and backup connections.

The world of ISDN is full of terms and acronyms. For example, different types of equipment are designated by codes, depending upon their function and whether or not they are ISDN-capable. There are still other codes used to reference the various ISDN interfaces between equipment. Finally, there are codes that reference the various ISDN standards and protocols.

ISDN Equipment

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

ISDN network equipment can generally be broken down into two groups – devices that understand and are capable of communication via ISDN, and those that do not natively support it. For example, a regular analog phone would not be capable of communicating over the ISDN network without some type of intermediate device converting its signals to the digital standard used by ISDN. However, some devices are already ISDN-capable, such as routers that ship with an ISDN interface. In the word of ISDN, devices are assigned a code that describes their capabilities.

The first type of device that you should look at is known as Network Termination 1 (NT1). In North America, an NT1 device sits at the customer premises, and connects to the service provider's network, as shown in the figure below. The local loop wiring between the service provider and a customer is based on 2 wires. The purpose of an NT1 is to convert this 2-wire interface into the 4-wire interface used by ISDN. Once installed, the NT1 can be used to connect up to 8 additional ISDN-aware devices on the customer network. As such, any customer (in North America) that wants ISDN connectivity must have a local NT1 device. In Europe, NT1 equipment is located at the service provider's facilities. Many Cisco router models provide an ISDN interface with a built-in NT1, allowing them to connect directly to a service provider's ISDN network.

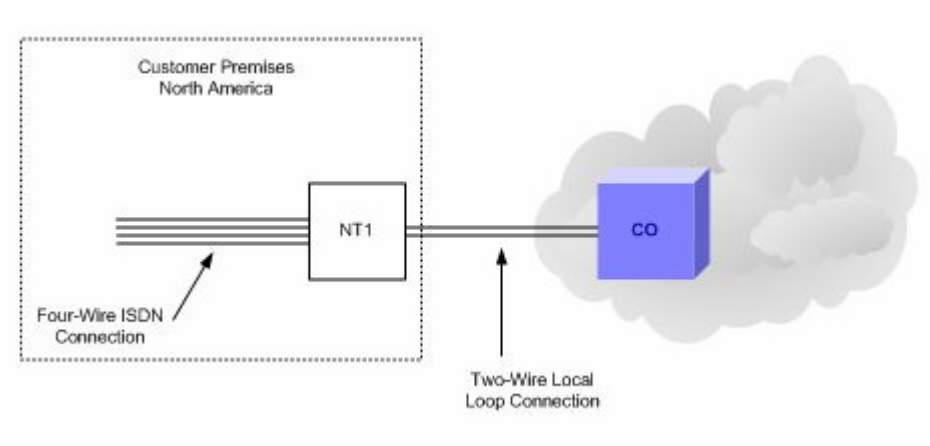


Figure: NT1 devices are located at the customer premises in North America and convert the standard 2-wire local loop interface to the 4-wire connection used by ISDN.

The next devices that you need to be aware of are known as Terminal Equipment (TE). Two types of Terminal Equipment exist, known as TE1, and TE2. TE1 devices are those designed to work with ISDN, using the 4-wire interface. An example might be an ISDN telephone, or a router with a built-in ISDN interface.

TE2 devices are those that pre-date ISDN standards, or are simply not ISDN capable. Examples would include a traditional analog phone, fax, or even a computer. In order for TE2 equipment to connect to the ISDN network, an additional piece of equipment is required. This device is referred to as a Terminal Adapter (TA). A Terminal Adapter is often an external device that connects to TE2 equipment over a serial port. It can also take the form of an expansion card, to install within a PC, for example. In either case, TE2 equipment always needs a TA to connect to an ISDN network. The figure below illustrates a small network with both TE1 and TE2 equipment connecting to an NT1 device.

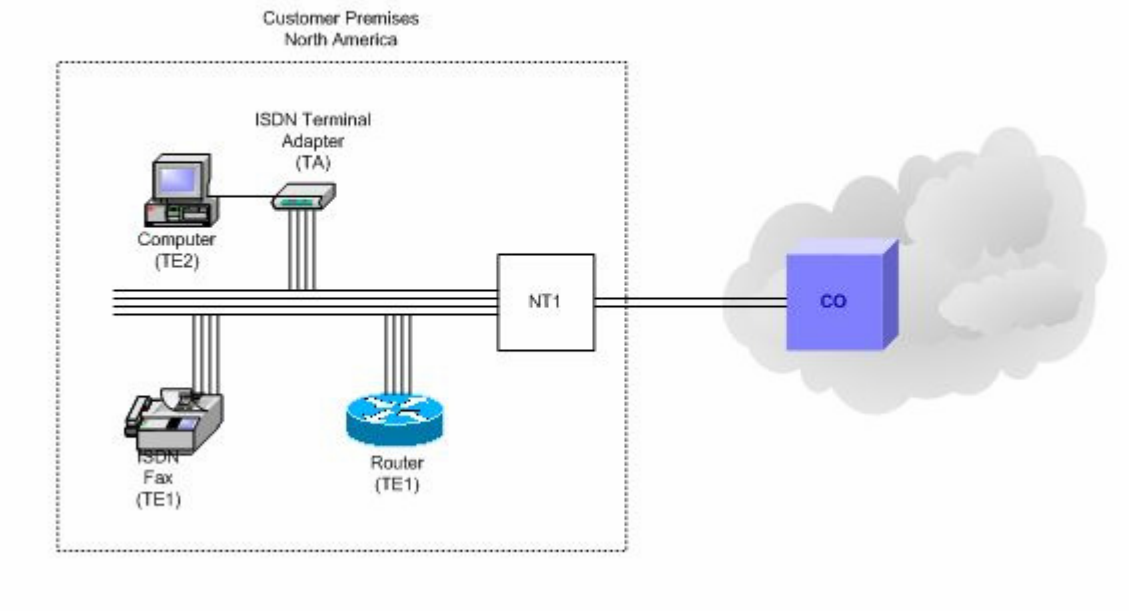


Figure: TE1 devices can connect directly to the 4-wire ISDN connection. TE2 devices require a Terminal Adapter (TA) to connect.

Another ISDN device that you should be familiar with is known as Network Termination 2 (NT2). An NT2 device is much less common, but is used to provide switching, concentrating, and multiplexing services for ISDN lines. NT2 devices typically take the form of Private Branch Exchange (PBX) equipment. In reality, most NT1 devices also include NT2 capabilities in their design. The placement of an NT2 device is illustrated in the figure below.

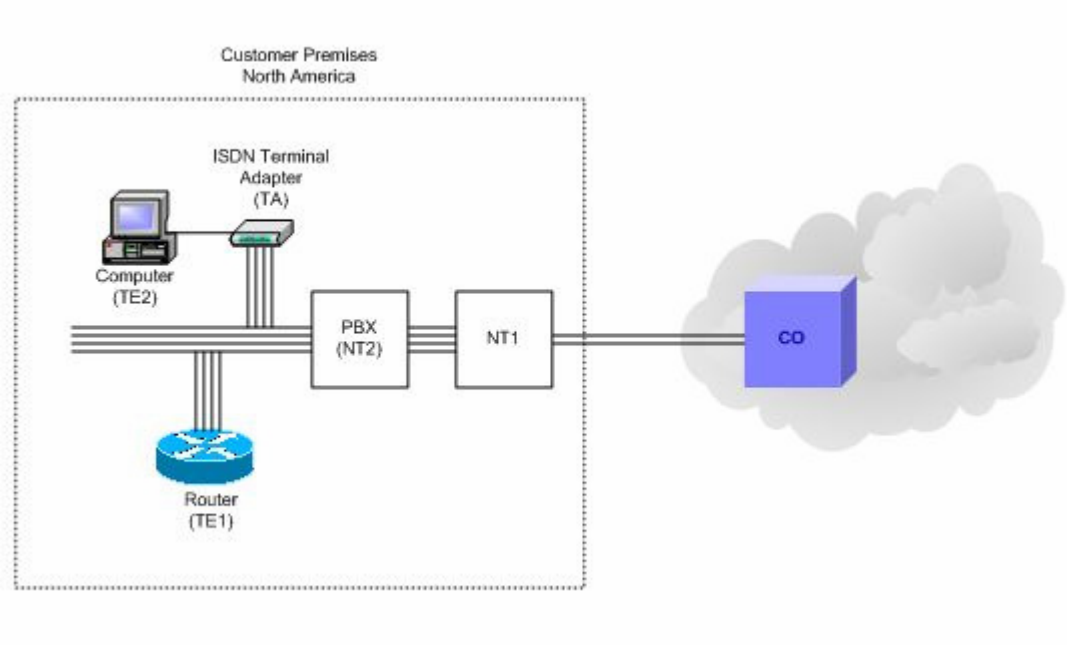


Figure: Placement of an NT2 PBX device at a customer premises in North America.

ISDN Reference Points

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

ISDN also specifies what are known as reference points between different types of ISDN equipment. These are letters that are used to define logical interfaces between equipment like an NT1 and TE1 device, for example. The letters R, S, T, and U identify the different reference points, but I've always found it easier to remember them using the term "RUST". The reference points described below are also illustrated in the figure below.

- The R reference point is used to describe the logical connection between non-ISDN devices (TE2) and a Terminal Adapter TA. If it helps you remember, just think of TE2 equipment as being "Retro", since it predates ISDN standards.
- The U reference point is used to describe the logical connection between the service provider's network and an NT1 device. Since an NT1 device is only found at the customer premises in North America, I always find it easiest to remember by designating it "United States".

- The S reference point is used to describe the logical connection between ISDN-aware equipment (like TE1s and TAs) and an NT1 device. Because TE1s and TAs are ISDN-aware, I use the term “Standard” to remember these connections.
- The T reference point is used to describe the logical connection between an NT1 and an NT2 device. In my mind, this has always looked like a type of trunk link, so I remember the interfaces using the term “Trunk”.

As mentioned earlier, many vendors implement both NT1 and NT2 capabilities with a single device, and their interfaces are electronically identical. As such, the S and T reference points are often referenced by the combination S/T.

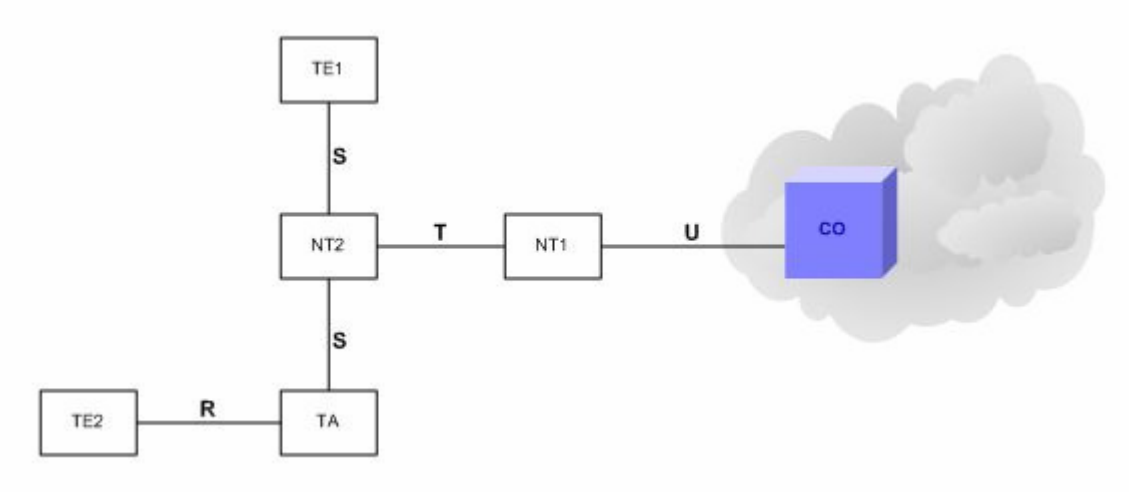


Figure: ISDN Reference Points

ISDN Protocols and Standards

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Having looked at the various codes used to describe ISDN equipment and references, now is a good time to get the remaining codes out of the way – those associated with ISDN protocols and standards. Different codes are used by the ITU-T to differentiate between standards and protocols that relate to the ISDN network, ISDN concepts, and ISDN switching and signaling. These include:

E-series. E-series ISDN standards relate to the telephone network. For example, standard E.164 describes international ISDN addressing.

I-series. I-series ISDN standards deal with ISDN concepts and interfaces. For example, standard I.112 outlines vocabulary terms of ISDN.

Q-series. Q-series ISDN standards relate to ISDN switching and signaling. For example, standard Q.921 describes the details of the Link Access Procedure on the ISDN D channel, LAPD.

ISDN BRI and ISDN PRI Services

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

There are two main services associated with ISDN – Basic Rate Interface (BRI) and Primary Rate Interface (PRI). Both services consist of multiple channels over which data can be sent (known as B channels) and also include a signaling channel (the D channel). The D channel is used for control and signaling purposes, such as setting and tearing down ISDN call. Referred to as "out-of-band" signaling, this method ensures that other ISDN calls do not interfere with existing connections, that bandwidth on the B channels is reserved for data only, and ultimately results in quicker call setup and teardown.

Basic Rate Interface (BRI) ISDN

Basic Rate Interface ISDN is made up of two 64 Kbps B channels that are used for sending and receiving data in full duplex, and one 16K D channel for signaling. In total, an ISDN BRI interface provides 144K of bandwidth (64+64+16). ISDN BRI is often referred to as 2B+D. Many Cisco router models include built-in BRI interfaces, but they can also be added using modular WAN interface cards.

Primary Rate Interface (PRI) ISDN

For companies with higher bandwidth requirements, ISDN Primary Rate Interface (PRI) service exists. In North America, PRI service consists of 23 64 Kbps B channels, and one 64Kbps D channel, for a total possible bandwidth of 1.544Mbps (T1 equivalent). In Europe, PRI service consists of 30 B channels and 1 D channel, for a total bandwidth of 2.048Mbps (E1 equivalent). ISDN PRI interfaces are typically implemented as modular WAN interfaces on Cisco routers, although some models do include built-in PRI ports.

ISDN and the OSI Model

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

ISDN maps to the three lower layers of the OSI model – Network, Data Link, and Physical. However, different protocols exist at the Data Link and Network layers for B and D channels, as illustrated in the figure below. The functions handled at each OSI layer are described below.

- Physical Layer. The ISDN Physical layer is concerned with the actual sending and receiving of bits over variety of interfaces. For example, the I.430 standard is responsible for providing communication over S/T reference points, while the ANSI T1.601 standard defines communication over U interfaces in North America.
- Data Link Layer. ISDN D channels use the Link Access Procedure for D Channels (LAPD – Q.921) to frame signaling and control data at Layer 2. On the B channel, data can be framed in a variety of ways, including via PPP and HDLC.
- Network Layer. ISDN D channels handle call setup, termination, and maintenance at the OSI Network layer using the Q.931 protocol, which implements common signaling

standards. On B channels, ISDN uses common Network layer protocols like IP, IPX, AppleTalk, and so forth.

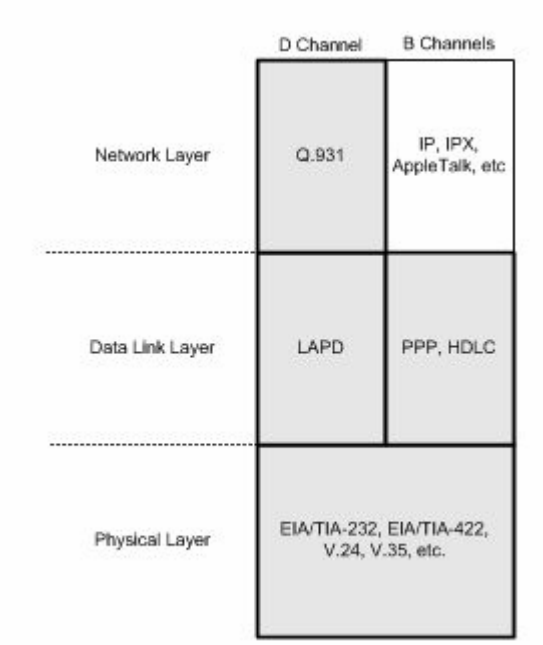


Figure: ISDN protocols and their relationship to the OSI model for B and D channels.

Configuring ISDN

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

For the purpose of your exams, you will need to be familiar with the configuration of ISDN. While many Cisco router models include an integrated ISDN BRI interface, many do not. In cases where an ISDN BRI port is not provided, you have two options. First, you can add a modular BRI port to your router using a WAN interface card. If one is not available (or your router's chassis isn't modular), you also have the option of using one of your existing synchronous serial interfaces in conjunction with an ISDN Terminal Adapter (TA).

If you're located in North America, it's generally a good idea to purchase a router whose BRI interface already includes an NT1 (this is usually marked next to the interface). If your model includes only an S/T interface, remember that you will require an external NT1 device. In Europe, this is obviously not an issue – you'll simply need an S/T interface, as NT equipment is located at the service provider's office.

BRI ports are referenced in the Cisco IOS using the same numbering scheme as other interfaces – the first BRI port will be bri0, the second bri1, and so forth. Configuring a router for ISDN is not terribly difficult, but does require that you have information provided by your service provider. For example, you'll need to know the ISDN switch type that you're connecting to, and potentially any Service Profile Identifier (SPID) numbers assigned to you. I'll discuss both in detail shortly.

To begin, I'm going to assume that your ISDN connection will be permanent. In other words, you are not using demand-dial routing (DDR) to connect locations. We'll look at DDR connections a little later in this section.

The first step in configuring ISDN on a Cisco router involves specifying the ISDN switch type in use at the service provider's facilities. A variety of different ISDN switch types exist, and you certainly don't need to memorize the keywords associated with them all. However, you do need to contact your service provider to find out which switch type they are using in order for your router to connect to their facilities properly. If a switch type is configured from global configuration mode, it will apply to all ISDN interfaces on a router. If configured from interface configuration mode, it will apply to that particular interface only. In this example, you'll configure our router to use a National ISDN-1 switch, identified by the keyword basic-ni1 (note that the last character in this command is the number "1" rather than a lowercase letter "l"). Common ISDN switch types are listed below. The router used in this example is a Cisco 1604, a model with a built-in BRI interface that includes an NT1 – in other words, it has a "U" interface.

Common ISDN switch types:

AT&T Basic Rate Switch basic-5ess

Nortel DMS-100 Basic Rate Switch basic-dms100

NET3 ISDN and Euro ISDN Switch (BRI) basic-net3

National ISDN-1 Switch (BRI) basic-ni1

AT&T 4ESS (PRI) primary-4ess

AT&T 5ESS (PRI) primary-5ess

Nortel DMS-100 (PRI) primary-dms100

```
Cisco1604#config t
Enter configuration commands, one per line.  End with CNTL/Z.
Cisco1604(config)#isdn switch-type basic-ni1
```

After configuring the ISDN switch type, the next step involves configuring the BRI interface. To access interface configuration mode, use the command int bri0.

```
Cisco1604(config)#int bri0
Cisco1604(config-if)#
```

The default encapsulation type used on an ISDN BRI interface is HDLC. PPP encapsulation can also be used on ISDN interfaces by issuing the command encapsulation ppp.

```
Cisco1604(config-if)#encapsulation ppp
```

Many service providers also require that Service Profile Identifier (SPID – literally pronounced "spid") numbers be specified in order to establish an ISDN connection with their switching equipment. A SPID is not a phone number, but is usually derived from the associated local dial number of the ISDN line. The purpose of a SPID is not only to identify you to the service provider's ISDN switch, but is also used to define any additional services that you may have subscribed to (such as call waiting, etc). Not all service providers use SPIDs, so you'll need to

find out if yours does. If a SPID is not configured when required, you won't be able to connect to the provider's network. As a general rule, a SPID will be provided for each ISDN B channel, if necessary.

SPID numbers are added from interface configuration mode, using the commands isdn spid1 and isdn spid2 respectively. Many service providers also require that a local directory number (LDN) be provided with the SPID in order to use both B channels. If required, the LDN directly follows the SPID number.

```
Cisco1604(config-if)#isdn spid1 41655512120101 5551212
Cisco1604(config-if)#isdn spid2 41655512130101 5551213
```

Having configured your SPIDs, there are only two steps left – assigning the interface an IP address, and issuing the no shutdown command.

```
Cisco1604(config-if)#ip address 192.168.1.101 255.255.255.252
Cisco1604(config-if)#no shutdown
```

Configuring ISDN Dial-on-Demand Routing (DDR)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

In cases where a branch office doesn't require a permanent connection to a company's head office, ISDN is often the best choice amongst WAN technologies. Because ISDN is circuit switched, a branch office router using ISDN could dial into a head office, as necessary, and then terminate the connection once a data transfer is complete. This can lead to significant cost savings, especially when compared to the expense associated with permanent connections or leased lines.

Demand-dial Routing (DDR) is the term used to describe such a connection. Setting up DDR involves configuring branch office routers to dial into a central corporate office (or vice versa), allowing data to be routed between locations. DDR isn't limited to ISDN – it can also be configured using slower technologies like traditional analog phone lines. However, the speeds at which connections are made with ISDN (often 1-2 seconds versus almost 30 seconds for analog calls) make it the better choice for demand-dial routing.

Perhaps the most important consideration when configuring a DDR connection is determining which types of traffic will initiate the connection. For example, will any traffic destined for the remote network initiate the connection, or will only certain types of traffic, such as HTTP or SMTP initiate it? These are very important decisions. If not specified, any type of traffic destined for the remote network could initiate the connection, which might it turn leave it almost permanently connected. This would certainly have an impact on any expected cost savings. The solution to this issue involves defining what traffic the router will find "interesting". If correctly configured, a router will only initiate a DDR connection when it comes across interesting traffic destined for the remote network.

A second key element to consider when configuring DDR is how routing will take place. As a general rule, DDR connections should use static or default routing only. If a dynamic routing

protocol like RIP were to be used on a DDR link, update traffic would reinitiate connections every 30 seconds. Remember that static routes will need to be defined to all networks (subnets) that have to be reached by the branch office. In cases where the branch office is a stub network (as shown in the figure below), a default route is usually the best option.

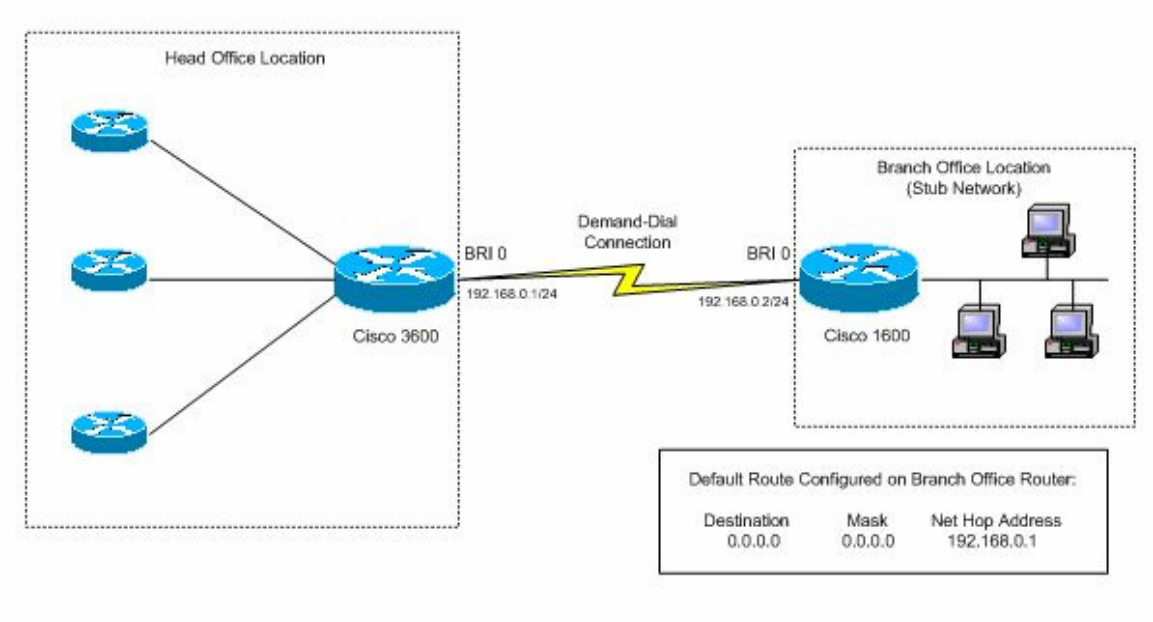


Figure: For DDR connections from a stub network, using a default route is usually the easiest configuration option.

For this example, let's assume that our network is configured as shown in the figure below. Our goal is to configure the branch office ISDN router such that it will initiate a connection to our head office location, but only for SMTP and HTTP traffic. Conversely, the head office ISDN router could also be configured to initiate connections to the branch office if necessary.

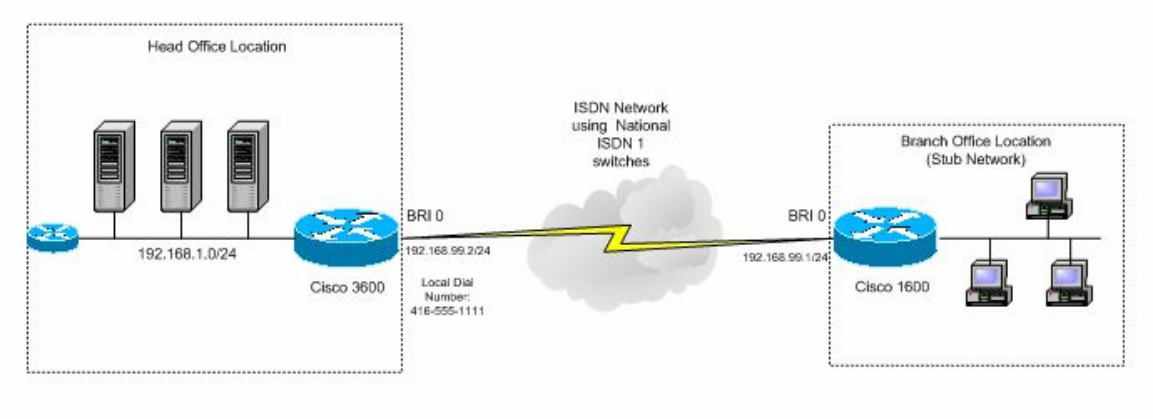


Figure: Network diagram for the ISDN configuration example.

As a first step, you'll configure the branch office router with an IP address on its BRI interface, define the ISDN switch type in use at the local CO, and configure a static route to the head office

location. You'll also configure the interface to use PPP encapsulation, which will allow you to add CHAP authentication (a very good idea) for these DDR connections.

```
Branch1604#config t
Enter configuration commands, one per line. End with CNTL/Z.
Branch1604(config)#isdn switch-type basic-ni1
Branch1604(config)#int bri0
Branch1604(config-if)#ip address 192.168.99.1 255.255.255.0
Branch1604(config-if)#encapsulation ppp
Branch1604(config-if)#exit
Branch1604(config)#ip route 192.168.1.0 255.255.255.0 192.168.99.2
```

Notice that I chose to use a static route to connect to the head office location. Since the branch office is a stub network, it would also have been reasonable to create a default route, using the destination network address 0.0.0.0.

The next step involves specifying which traffic the router will consider interesting. This is accomplished using the dialer-list command. Recall that our goal was to limit DDR connections to SMTP and HTTP traffic only. In order to accomplish this, you need to use an extended IP access list. For illustration purposes, initially configure the router to view all IP traffic as interesting:

```
Branch1604(config)#dialer-list 1 protocol ip permit
Branch1604(config)#int bri0
Branch1604(config-if)#dialer-group 1
Branch1604(config-if)#exit
```

The dialer-list command specified above tells the router that all IP traffic is "interesting", and should initiate the link. However, this dialer list does nothing until actually applied to an interface using the dialer-group command from interface configuration mode. To remove this dialer list from the interface, use the no dialer-group 1 command.

```
Branch1604(config-if)#no dialer-group 1
```

In order to narrow the list of traffic that the router finds "interesting", you can use access lists. Recall that in order to filter traffic according to port number, an extended IP access list would be required. This is illustrated below.

```
Branch1604(config)#access-list 150 permit tcp any 192.168.1.0 0.0.0.255 eq 25
Branch1604(config)#access-list 150 permit tcp any 192.168.1.0 0.0.0.255 eq 80
Branch1604(config)#dialer-list 1 list 150
Branch1604(config)#int bri0
Branch1604(config-if)#dialer-group 1
```

With the "interesting" traffic now specified, you still need to configure the number to be dialed. The command to do this is dialer map, followed by the ip address of the remote router, its hostname, and the phone number to connect to the remote location.

```
Branch1604(config-if)#dialer-map ip 192.168.99.2 name Cisco2620 4165551111
```

One additional capability that you might be interested in is the ability to define when the second BRI interface should be connected. This is accomplished using the dialer load-threshold command. The number specified after the command is an integer value between 1 and 255, where the number specified is used as a percentage. For example, if you were to configure the interface with a load threshold of 255, the router wouldn't bring up the second B interface until the load on the first had reached 100% utilization. If a lower number such as 128 were specified, the

second B channel would be connected once utilization on the first had reached just over 50%. An associated direction is also specified with the command – for example, you can configure the load-threshold to consider only inbound, outbound, or traffic in both directions in its calculations. To specify both, the either keyword is added to the command as shown below.

```
Branch1604(config-if)#dialer load-threshold 128 either
```

By default, a router will terminate a demand dial connection after 120 seconds have passed without it coming across any interesting traffic. This number can be changed using the dialer idle-timeout command, and specifying a new value in seconds.

```
Branch1604(config-if)#dialer idle-timeout 90
```

Although not explicitly required, it's always a good idea to configure CHAP authentication on DDR links for the extra security it provides. In order to do this, you need to specify a username and password from global configuration mode, following the exact same steps used in the PPP section of this chapter. Configuring this router to use CHAP will force authentication when the other router initiates a session. Similar commands would need to be issued on the 2620 router at the head office location to force CHAP authentication when our 1604 router attempts to connect. As the last step, don't forget to issue the no shutdown command on the bri0 interface!

```
Branch1604(config)#username Cisco2620 password isdn-is-fun
Branch1604(config)#int bri0
Branch1604(config-if)#ppp authentication chap
Branch1604(config-if)#no shutdown
```

Monitoring ISDN

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

There are a variety of commands available for the purpose of monitoring ISDN connectivity, as outlined below.

- show isdn active. This command displays information about the current ISDN call, including the number dialed and whether a call is in progress.
- show isdn status. Displays the status of all ISDN interfaces, providing information relating to OSI layers 1 through 3.
- debug isdn q921. Displays link access information about ISDN D channels at the Data Layer. To make this command easier to remember, just relate the "2" in the q921 keyword as the associated OSI layer.
- debug isdn q931. Displays information about ISDN call setup and teardown on ISDN D channels at the Network layer. To make this command easier to remember, just relate the "3" in the q931 keyword as the associated OSI layer.
- debug dialer. Displays information relating to call setup and teardown.

Frame Relay

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Frame Relay is a packet-switching technology that exists at the Data Link layer of the OSI model and is one that has become increasingly common as a WAN solution since the early 1990s. Unlike with leased lines and circuit-switched networks, the available bandwidth on a provider's Frame Relay network is shared amongst many subscribers. This sharing of resources leads to significantly lower costs than traditional leased lines.

Many people tend to be confused by packet-switching technologies like Frame Relay. Mostly this is a result of trying to understand how data actually gets from one location to another. On packet-switched networks (like Frame Relay), data streams are separated through the use of "virtual" rather than dedicated hardware circuits. In other words, a logical path is defined between endpoints, through a provider's packet-switched network. Many virtual circuits will be defined for different customers, and will be multiplexed over the shared physical links of the network. As an example, consider the figure below. It shows two different companies, each connecting two offices over a provider's Frame Relay network. Notice that between Frame Relay switches X and Y, both of their virtual circuits traverse a common physical link. The data that one company passes between their own offices is completely separate from the data of the other company – all data stays within each company's dedicated virtual circuit only.

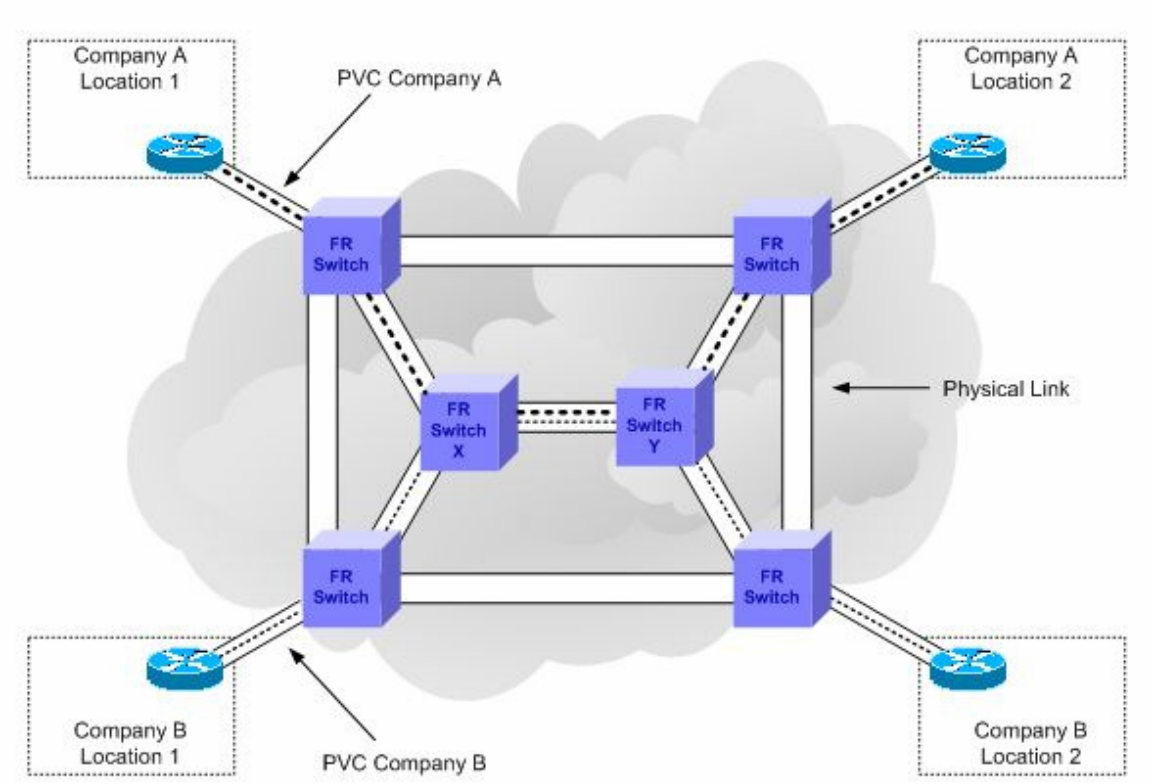


Figure: PVCs of two different companies traveling over the same Frame Relay network, and at times, common links.

Two main types of virtual circuits can be defined on a Frame Relay networks – permanent virtual circuits (PVCs) and switched virtual circuits (SVCs). A PVC functions somewhat similar to a

leased line, in that a service provider defines a path through the packet switched network to each customer location. In cases where companies wish to have “always-on” connectivity between locations using Frame Relay, PVCs are usually defined.

A switched virtual circuit (SVC) functions somewhat differently, almost like a circuit-switched connection. SVCs are not permanent, and can instead be created as required across a packet-switched network. For example, an SVC could be created between a company’s head office and a remote location. For the duration of the connection, data would travel across the path defined by the SVC. However, if the circuit was terminated and a new SVC established at a later time, data might travel over a completely different path.

Frame Relay networks are referred to as being non-broadcast multi-access (NBMA). What this means is that, by default, broadcast traffic will normally not be passed over a virtual circuit without explicit configuration. This is an important consideration when dealing with the use of broadcast-based routing protocols like RIP or IGRP in a Frame Relay environment. You’ll look at how broadcast traffic can be handled on Frame Relay networks later in this section.

Frame Relay Equipment

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

In order to connect to a Frame Relay network, both DTE and DCE equipment needs to be located at the customer premises. This DTE equipment is usually a router, whose serial interface connects to a DCE device. In the past, customers required a completely separate DTE device known as a Frame Relay Access Device (FRAD) to connect to a Frame Relay network. However, almost all routers sold today (with an appropriate serial interface) are capable of handling Frame Relay encapsulation and communication. The DCE device is usually a CSU/DSU that provides clocking functions and the connection to the provider’s physical circuit. Ultimately, the physical link from the customer premises connects to the Frame Relay switching equipment of the service provider. This switching equipment is not the responsibility of the customer. The figure below illustrates the interconnections of equipment on a Frame Relay network.

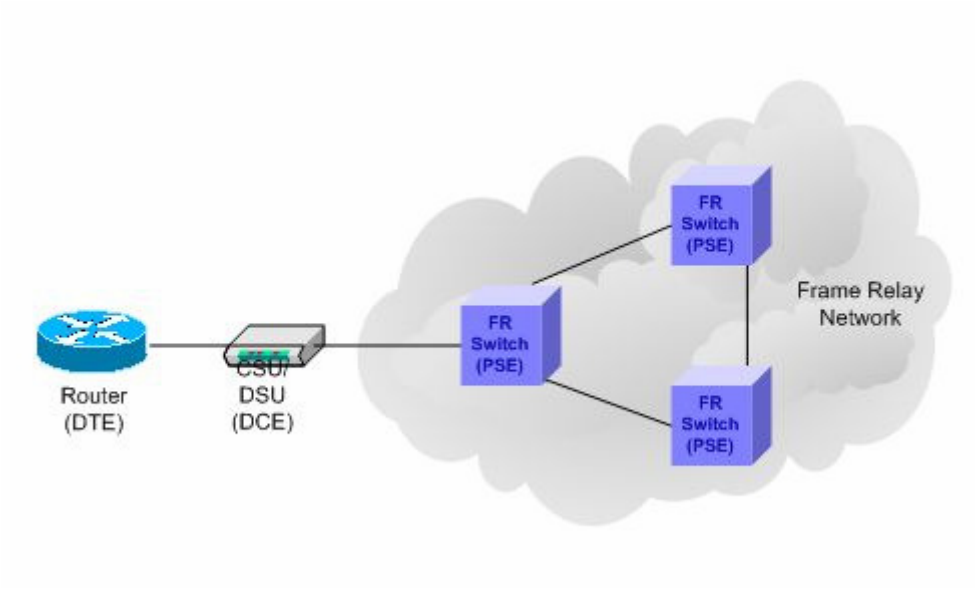


Figure: Connections through a Frame Relay network are made using DTE, DCE, and PSE equipment.

Committed Information Rate (CIR)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

When a company provisions a leased line from a service provider, that link provides a fixed, dedicated amount of bandwidth. For example, a company might implement a dedicated 256 Kbps link between two locations. Things work a little bit differently in the world of Frame Relay. Instead of providing fixed bandwidth over a dedicated circuit, bandwidth is allocated using what is referred to as a Committed Information Rate (CIR).

A CIR is an average transmission capacity agreed upon for a virtual circuit that connects one company location to another. For example, a company may provision a circuit with a CIR of 64Kbps. What this means is that the company is basically guaranteed that they will always have at least 64 Kbps of bandwidth available to them. However, because the physical circuits of the Frame Relay network are shared amongst many subscribers, there will often be excess bandwidth available at any point in time. This allows a customer's traffic to actually "burst" to higher speeds, as available network bandwidth permits. The speed to which data can burst is referred to as the Committed Burst Information Rate (CBIR). For example, a virtual circuit with a CIR of 64 Kbps may have a CBIR of 128 Kbps.

The fact that companies can potentially use more bandwidth than they have paid for makes Frame Relay an attractive option, but it obviously presents an issue. What would stop one company's traffic from monopolizing available bandwidth? This issue is dealt with by designating certain frames "discard eligible" (DE). For example, let's say that a company has paid for a CIR of 64Kbps. When sending data, all frames that are within this 64 K bandwidth allotment are sent out normally. However, once the CIR is met, any additional frames that need to be sent have a special bit (known as the DE bit) in their frame header set to 1. The DE bit marks the frame as eligible to be discarded, should the network be congested. This is illustrated in the figure below.

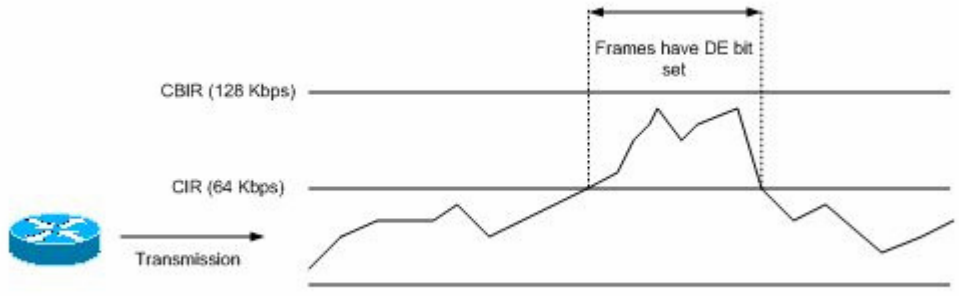


Figure: Frames sent at rates above the CIR will have their discard eligibility (DE) bit set.

Another benefit of Frame Relay is that CIRs need not necessarily be the same throughout a customer's network. For example, a company may have two branch office locations connected to

their head office, as illustrated in the figure below. In this example, the PVC between the Montreal branch office and the head office has a CIR of 64Kbps, while the PVC connecting the head office to the much smaller Utah office has a CIR of only 16Kbps. CIR values are usually made available in multiples as small as 4 or 8 Kbps. In fact, it is also possible to provision Frame Relay services with a CIR of 0 – while this connection would be valid; all frames sent over the PVC with this CIR would be marked discard eligible. In reality, many Frame Relay networks have enough excess capacity that a CIR of 0 is not necessarily a bad choice for connecting to small offices, or as an interim solution while a more reasonable CIR is being determined.

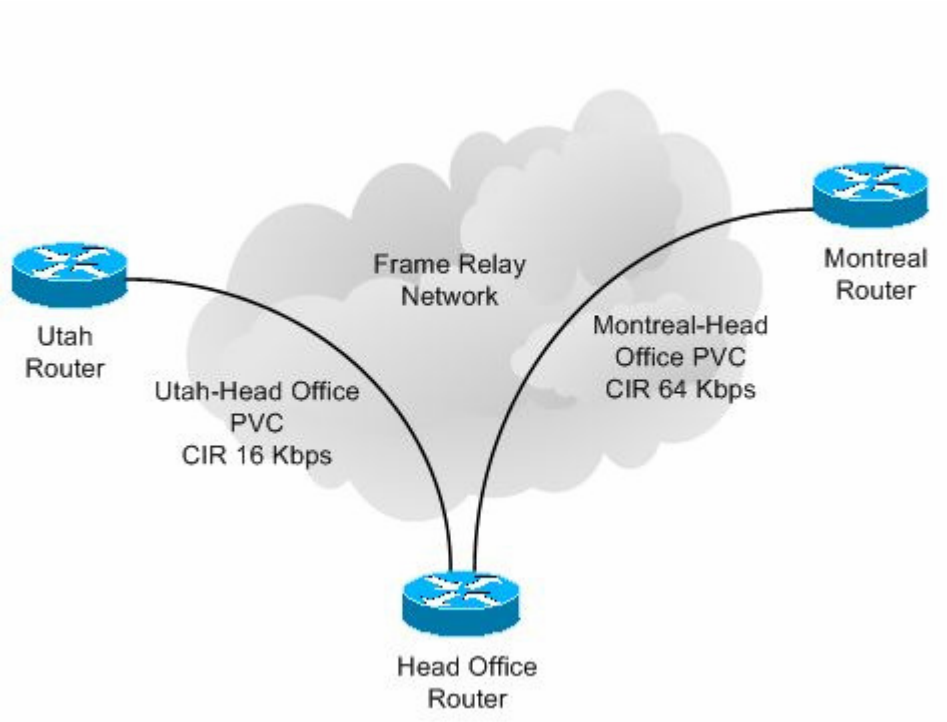


Figure: PVC connections across a Frame Relay network can have different CIRs.

Data Link Connection Identifiers (DLCIs)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

In order to ensure that data reaches its proper destination through a packet-switched network, virtual circuits somehow need to be identified. On a Frame relay network, virtual circuits are identified using what are known as Data Link Connection Identifiers (DLCIs).
A DLCI is a 10-bit number that is stored in the header of a frame that traverses a Frame Relay network. DLCI numbers are issued by the service provider and uniquely identify a virtual circuit. These numbers are said to be only locally significant, in that they may not be the same (or consistent) throughout a Frame Relay network. This concept can be confusing and is best illustrated with an example.

Imagine that you were to provision a Frame Relay PVC between two locations from a service provider. The provider would issue two DLCI numbers – one for each location. While these numbers could be the same, there is no guarantee that will be the case, and it doesn't really matter anyhow. The DLCI number is only used to ensure that the provider's switching equipment can properly identify the customer's virtual circuit at any point in the network. So, although your virtual circuit (VC) may be identified as DLCI 99 between your Toronto router and the first Frame Relay switch it encounters, intermediate switches may use different DLCI numbers to identify the same virtual circuit. At each switch, the DLCI stored in the header will be translated if necessary. As such, the very same virtual circuit may be identified using DLCI 101 at the New York office. In this way, the DLCI value stored in the header is only locally significant. This is illustrated in the figure below.

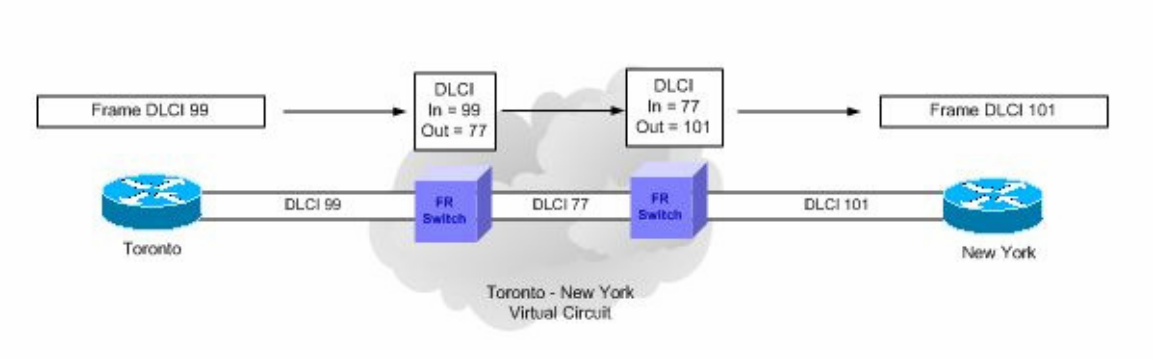


Figure: A single PVC is usually identified by different DLCI numbers across a Frame Relay network.

While less common, it is also possible for DLCI numbers to have global significance in a service provider's network. When globally significant, each customer location would be identified by a single DLCI value. While this makes a Frame Relay network easier to manage, it also seriously limits the number of possible PVCs that can be configured.
Because a DLCI number is a 10-bit value, up to 1024 (210) DLCI numbers (0-1023) can be used to identify virtual circuits. However, some of these numbers are reserved for management functions. On a Cisco router, up to 992 DLCIs can be defined, ranging from DLCI 16 up to 1007.

Frame Relay Encapsulation

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

In order for upper-layer data to be transmitted across a Frame Relay network, it obviously needs to be framed. Cisco routers support two different Frame Relay encapsulation types, known as Cisco and IETF. By default, a Cisco router will use Cisco encapsulation on Frame Relay interfaces, unless IETF is explicitly specified. IETF encapsulation is usually used when connecting both Cisco and non-Cisco devices across a Frame Relay network. However, another issue at hand is the framing method in use by your service provider. As a general rule, it's always a good idea to obtain this information before beginning the configuration process.

Frame Relay Local Management Interface (LMI)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

The Local Management Interface (LMI) is a set of extensions to the Frame Relay protocol that were designed to provide information about the status of Frame Relay networks, and extend the technology's capabilities. LMI is primarily concerned with diagnostic functions. For example, LMI is used to send keepalive messages between a router and a Frame Relay switch. These messages verify the status of PVCs, and help to ensure that data can be transferred between Frame Relay DTE and DCE equipment. The LMI extensions also allow DLCI values to be made globally significant, and bring multicast capabilities to Frame Relay networks.

Both ANSI and the ITU-T have also developed LMI standards. As of IOS version 11.2, a Cisco router interface using Frame Relay encapsulation will auto-sense the LMI type used by a Frame Relay switch. However, it is still possible to configure the LMI type manually. The three different LMI types supported on a Frame Relay interface include:

- Cisco. This is the "Group of Four" LMI type, and is the default used on Frame Relay interfaces prior to IOS version 11.2. It uses DLCI 1023 to exchange status information.
- ANSI. This is the LMI type defined by ANSI. It uses DLCI 0 to exchange status information.
- Q.933a. This is the LMI type defined by the ITU-T. It uses DLCI 0 to exchange status information.

A router and its local Frame Relay switch must be using the same LMI type in order for diagnostic messages to be passed between them.

Frame Relay Interfaces and Connections

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Back in Chapter 7 you looked at the concept of subinterface. Subinterfaces allow you to configure a single physical interface in such a way that it can act as multiple interfaces, each of which can have different properties. When configuring Frame Relay on a router, each virtual circuit is usually assigned to its own serial subinterface. This not only allows multiple virtual circuits to function across a single physical interface, but also allows the properties of each subinterface to be configured individually.

As an example, consider the figure below. In it, a company has one head office and three branch locations. Each of the branch offices connects to the head office location using a dedicated PVC. Router interface S0 at the head office is divided into 3 subinterfaces, each of which would be configured with an appropriate IP address and its associated DLCI number. In order to simplify

the management of subinterfaces (and make things easier to remember), subinterfaces are usually given a number that maps directly to a PVC's DLCI number. This is not a requirement, but rather an easy way to keep track of which PVC is associated with a given subinterface. For example, DLCI 23 at the head office would be configured on subinterface S0.23.

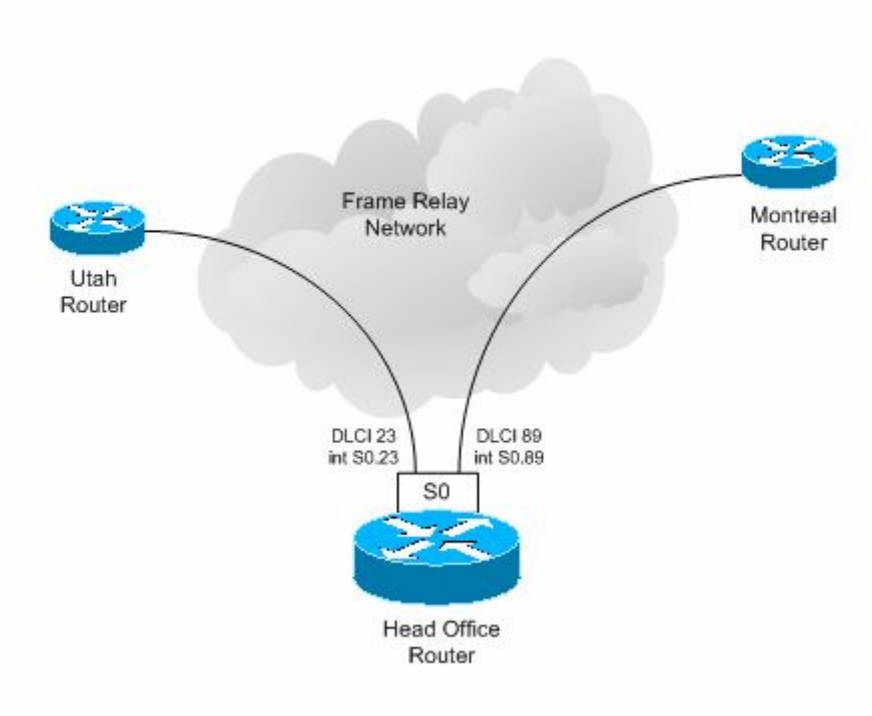


Figure: Subinterface numbers that correspond to DLCI values simplify subinterface management.

In order for a Frame Relay network to function correctly, interfaces will still need to be configured according to a logical addressing scheme. I'm going to assume that our network runs only IP in this example, but it could be running other Network layer protocols like IPX or AppleTalk as well. The issue at hand is how logical addressing should take place over the Frame Relay network. For example, each PVC could be treated as a point-to-point network and, as such, would require its own IP subnet. However, if multiple PVCs are connected into a single router in a hub-and-spoke type configuration, it's also possible to configure the connections as a single subnet instead – this configuration is known as multipoint. Point-to-point connections are generally more common when connecting locations by Frame Relay, but either configuration is valid. Both are illustrated in the figure below.

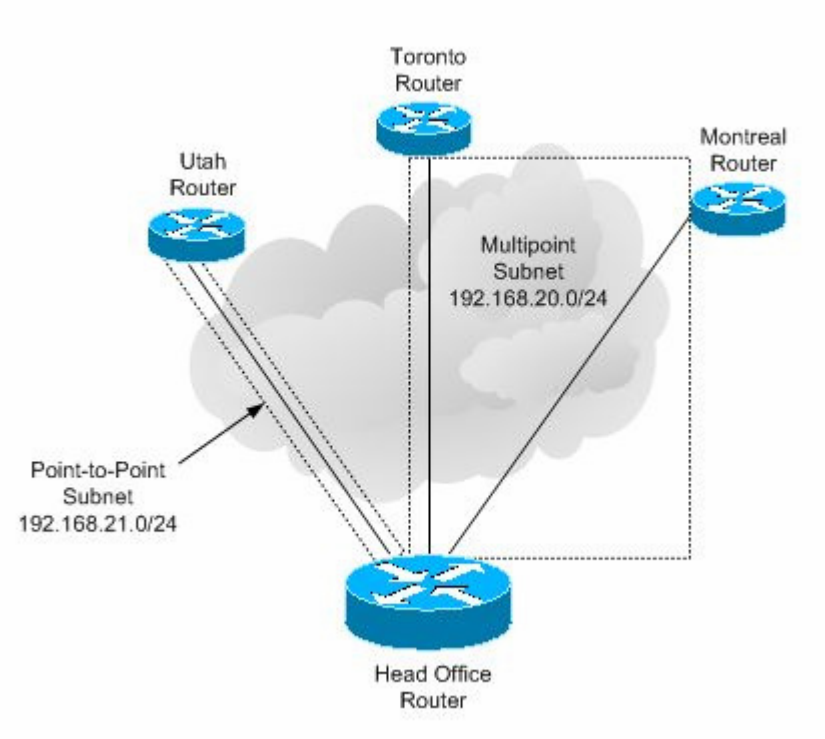


Figure: PVCs connecting locations can be made part of either point-to-point or multipoint links.

Another important reason for the use of subinterfaces on a Frame Relay network relates to issues associated with distance vector routing protocols. If you'll recall, split horizon prevents a router from sending routing table updates that were received on a given interface back out the same interface. As such, if you had the multipoint configuration shown in the figure below, where the head office location does not use subinterfaces, routing table updates from Router B could never reach Router C or vice versa. Quite simply, Router A would not be allowed to forward updates received from B to C over the same interface.

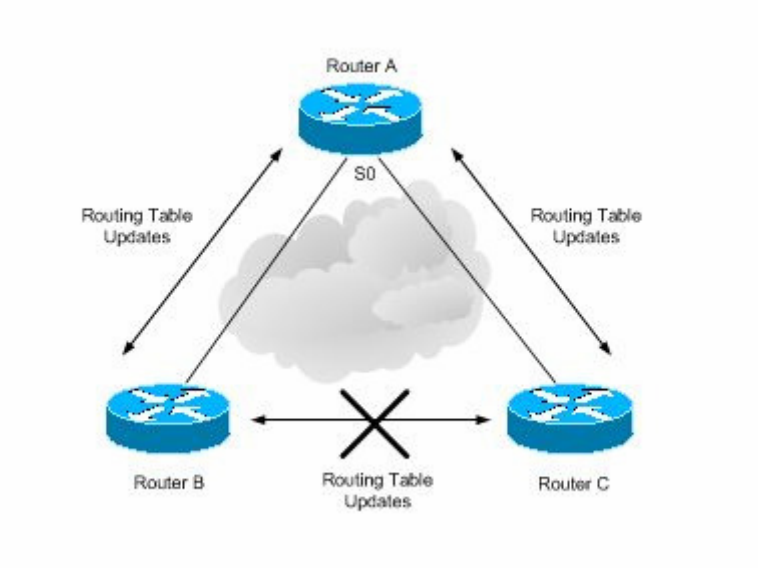


Figure: Without the use of subinterfaces, routing table updates from Router B could never reach Router C via Router A because of split horizon rules.

This is not an issue when subinterfaces are used, since the routing protocol will consider subinterfaces as distinct and separate. If subinterfaces were used on Router A in Figure 21, routing table updates could pass between all three routers via Router A without issue.

Frame Relay Congestion Management

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Back in the CIR section of this chapter, you looked at how the discard eligibility (DE) bit is used to "mark" frames that are transmitted above the CIR on a Frame relay virtual circuit. Making certain frames discard-eligible provides a way to determine which frames should be discarded first if a Frame Relay switch is experiencing congestion. However, Frame Relay also uses two additional techniques to help make end devices aware of congestion on the Frame Relay network – Forward Explicit Congestion Notification (FECN) and Backwards Explicit Congestion Notification (BECN).

- FECN. If a Frame Relay switch is experiencing congestion, it will set what is known as the FECN bit in the Frame Relay header. This bit is used to notify the receiving device that the network is experiencing congestion and that it should expect some delay in the receipt of frames. Ultimately, this information should be passed to higher-layer protocols for processing.
- BECN. By the same token, a switch experiencing congestion will also set the BECN bit in frames that are being sent back to the original sender. BECN alerts the sender to a congestion situation, effectively letting the sender known that it should throttle back the rate at which it sends data. In the same way, this information should be passed to higher-layer protocols for processing.

Frame Relay Communications

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Now that you've taken a look at the key concepts associated with Frame Relay, let's take a look at an example of how communication occurs across a Frame relay network. I am going to assume that the network is configured as shown in the figure below, and that Computer1 at the New Orleans location ultimately wants to communicate with Server3 in the Austin office. Both PVCs are configured as point-to-point links.

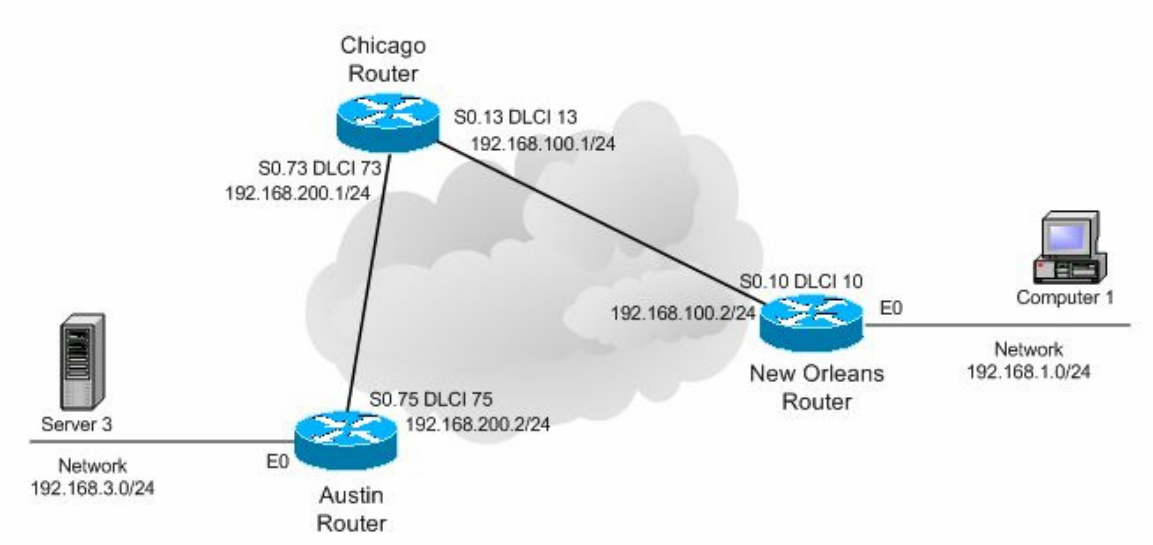


Figure: Example network used to describe the communication process across a Frame Relay WAN.

The steps below outline how an IP packet from Computer 1 will ultimately reach Server 3.

1. Through the ANDing process, Computer 1 will recognize that Server 3 is on a remote network. As such will ARP for the MAC address of its default gateway, interface E0 on the New Orleans router. The IP packet will be framed for Ethernet and sent to the router's E0 interface.
2. After receiving the Ethernet frame, the New Orleans router will calculate the CRC to ensure it is not corrupt, strip off the header, and will pass the packet up to the Network layer. At this point, it will look in its routing table to see where it should forward the packet next. In this case, the destination would be 192.168.100.1, the router at the Chicago office.
3. Before forwarding the packet, the New Orleans router will obviously need to reframe it to travel over the Frame Relay network. In order to do this, it must identify the DLCI of the virtual circuit that connects to the Chicago location. Let's assume that the router is configured correctly, and knows address 192.168.100.1 is accessible via DLCI 10. The router will reframe the packet, and forward it out subinterface S0.10.
4. Ultimately, this frame will reach the switching equipment of the service provider. Because DLCIs are usually only locally significant, the DLCI number in the frame header may, in fact, be changed many times as it moves between the switching equipment of the service provider. You can't tell for certain which mappings the provider uses to identify the virtual circuit within its own network. However, you do know that upon exiting the last switch before the Chicago router, its DLCI value should be set to 13, as per Figure 22.
5. Upon reaching the Chicago router's S0.13 interface, the frame's CRC will be recalculated, the framing striped off, and the packed will be passed to the Network layer. At this point, the Chicago router will consult its routing table, look at the destination address, and will determine that the packets destined for network 192.168.3.0/24 should be forwarded to address 192.168.200.2. I will again assume that the router is configured correctly, and knows that address 192.168.200.2 is accessible via DLCI 73. The router will reframe the packet, and forward it out subinterface S0.73.
6. This frame will also traverse the provider's Frame Relay network through a variety of switches, where the DLCI information stored in the frame header may again change many times. Again, the final DLCI number in the header once it reaches the Austin router should be 75, as per our diagram.

7. Upon reaching the Austin router's S0.75 interface, the frame's CRC will be recalculated, the framing will be stripped away, and the packet will be passed to the Network layer. At this point, the router will search for the destination network address is its routing table, and will determine that the packet should be forwarded out interface E0. The router will ARP for the MAC address associated with Server C (if the MAC address is not already stored in its ARP cache), and once obtained, will reframe the packet for Ethernet and forward it out interface E0.
8. Upon receiving the Ethernet frame, Server C will identify the MAC address as its own, recalculate the CRC, strip off the framing, and then pass the packet up to the Network layer for further processing.

Obviously there are many steps involved when Computer 1 wishes to communicate with Server 3. You may be asking yourself why the company doesn't simply add another PVC between the New Orleans and Austin offices. While entirely valid, the need for another PVC would largely depend on the volume of data that would be passed between these two offices regularly. Don't forget that provisioning another PVC would incur additional costs, which may or may not be justifiable. In most cases, companies will configure PVCs between a branch office and a central location, rather than between branch offices directly. However, additional PVCs connecting locations (forming a mesh) does provide an additional degree of fault tolerance.

Frame Relay Configuration

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Having looked at both Frame Relay concepts and an example of the communication process, let's take a look at configuring a Cisco router to use Frame Relay. Our network in this example is configured as illustrated in the figure below. Notice that the PVC between Chicago and Toronto is a point-to-point interface, while the PVCs between Chicago and San Jose, as well as Chicago and Miami are part of a multipoint configuration. Your goal is to properly configure Frame Relay on the Chicago router.

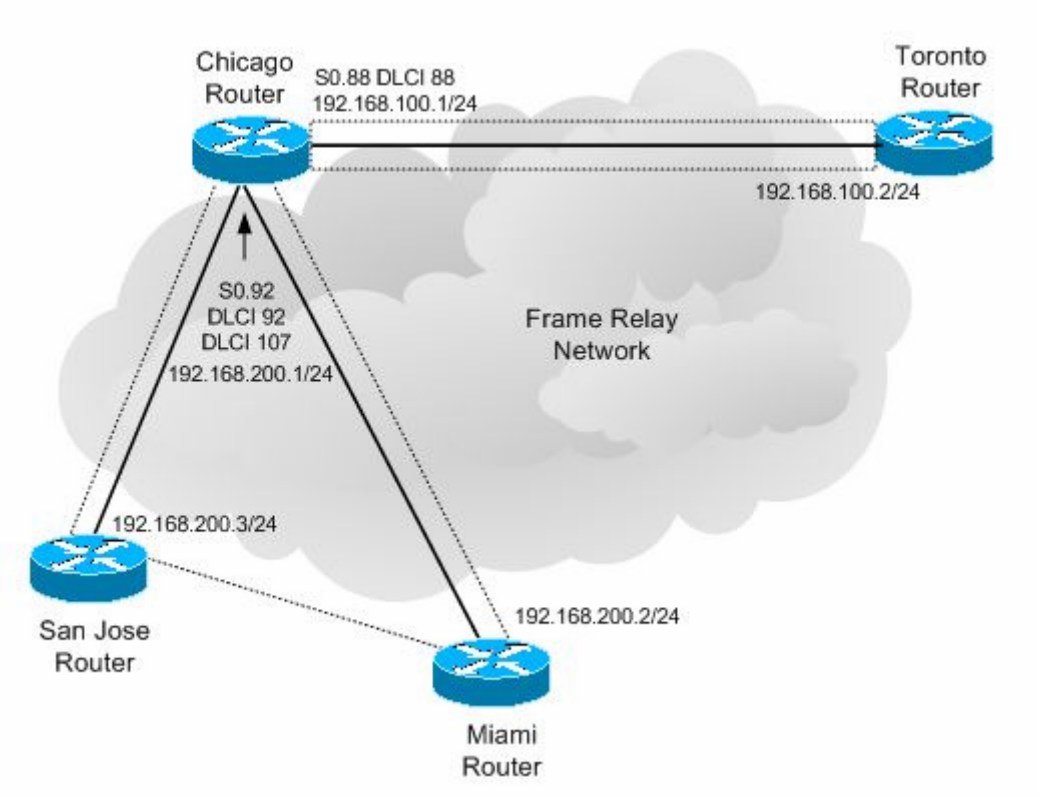


Figure: Frame Relay network used in configuration example.

You'll begin by configuring settings on the Chicago router's S0 interface. This includes the encapsulation method to be used, as well as the LMI type. After these steps are complete, you will configure two subinterfaces, one for DLCI 88 (the PVC to Toronto), and one other for DLCIs 92 and 107 (the PVCs to Miami and San Jose respectively).

```
Chicago#config t
Enter configuration commands, one per line. End with CNTL/Z.
Chicago(config)#int s0
Chicago(config-if)#encapsulation ?
atm-dxi ATM-DXI encapsulation
frame-relay Frame Relay networks
hdlc Serial HDLC synchronous
lapb LAPB (X.25 Level 2)
ppp Point-to-Point protocol
smds Switched Megabit Data Service (SMDS)
x25 X.25
Chicago(config-if)#encapsulation frame-relay ?
ietf Use RFC1490/RFC2427 encapsulation
Chicago(config-if)#encapsulation frame-relay
Chicago(config-if)#frame-relay lmi-type ?
cisco
ansi
q933a
Chicago(config-if)#frame-relay lmi-type cisco
Chicago(config-if)#exit
```

By default, the encapsulation method used on a Frame Relay interface will be Cisco, unless IETF is specified. Recall that it isn't usually necessary to specify an LMI type, since the interfaces will auto-sense the LMI type in use by the local Frame Relay switch as of IOS version 11.2.

With the basic interface settings completed, you can move on to the configuration of the subinterfaces. You'll begin with the subinterface that will be used to connect to the Toronto office, S0.88. This link represents a point-to-point connection, as shown below.

```
Chicago(config)#int s0.88 ?
multipoint Treat as a multipoint link
point-to-point Treat as a point-to-point link
Chicago(config)#int s0.88 point-to-point
```

Your next step involves configuring an IP address on subinterface S0.88, as well as its DLCI number.

```
Chicago(config-subif)#ip address 192.168.100.1 255.255.255.0
Chicago(config-subif)#frame-relay interface-dlci 88
```

At this point, configuration of a point-to-point Frame Relay interface on the Chicago router is complete. Of course, communication will not be possible until the Toronto router is also configured, and the no shutdown command is issued on interface S0. DLCI to IP address mappings are not configured on point-to-point Frame Relay interfaces, since only one other possible communication endpoint exists on the virtual circuit.

The configuration of multipoint interfaces is slightly different, and is the reason why I included both types in this example. The encapsulation method and LMI type have already been specified on interface S0, so there is no need to issue them again (remember that subinterfaces inherit settings from their "parent" interface). Because the connection between Chicago, Miami, and San Jose is multipoint (a single subnet), you can configure both virtual circuits on a single subinterface. You'll use the DLCI number for the PVC to Miami as our subinterface number, if for no other reason than the fact that it's the lower of the two.

```
Chicago(config)#int s0.92 ?
multipoint Treat as a multipoint link
point-to-point Treat as a point-to-point link
Chicago(config)#int s0.92 multipoint
Chicago(config-subif)#ip address 192.168.200.1 255.255.255.0
Chicago(config-subif)#frame-relay interface-dlci 92
Chicago(config-fr-dlci)#exit
Chicago(config-subif)#frame-relay interface-dlci 107
Chicago(config-fr-dlci)#exit
```

Communication over a multipoint interface requires that you somehow map remote IP addresses to the local DLCI number of the corresponding virtual circuit. By default, a multipoint interface will automatically map this pairing of information using inverse ARP. However, it is also possible (and generally recommended) to create static mappings. This is accomplished using the frame-relay map command, as outlined below.

```
Chicago(config-subif)#frame-relay map ip 192.168.200.2 92 broadcast
Chicago(config-subif)#frame-relay map ip 192.168.200.3 107 broadcast
```

If you take a closer look at the commands above, you'll notice that they create a mapping between the IP address of the remote router and the local DLCI number of the appropriate virtual circuit. When the Chicago router needs to reach address 192.168.200.2 (the Miami office), it now

knows that frames must be assigned to the virtual circuit identified by DLCI 92. When traffic needs to reach address 192.168.200.3 (the San Jose office), the virtual circuit identified by DLCI 107 should be used. You may have also noticed the use of the broadcast keyword at the end of the frame-relay map commands. The broadcast keyword tells the router to forward broadcasts (like routing protocol updates) across the PVC as required. This is necessary because by default, Frame Relay networks are non-broadcast multi-access (NBMA). As such, they would not forward routing table updates from broadcast-based protocols like RIP or IGRP without the broadcast keyword configured. For practical purposes, it's worth noting that broadcasts will be forwarded across point-to-point Frame Relay interfaces by default as of IOS 11.1

Monitoring Frame Relay Communications

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

There are a variety of commands available for the purpose of monitoring Frame Relay on Cisco routers. Some of the commands that you should be familiar with include:

- **show frame-relay lmi**. This command provides detailed statistics on LMI traffic passed between a router and its local Frame Relay switch, as well the LMI type is use.
- **show frame-relay map**. This command shows a listing of the Network layer address to DLCI mappings defined on a router, along with their current status.
- **show frame-relay pvc**. This command provides a listing of all PVCs configured on the router, and their associated local DLCI numbers. The command also provides traffic statistics, such as the number of frames sent, received, dropped, marked discard eligible, or with their FECN or BECN bit set.
- **show interface**. This familiar command provides general interface data such as the encapsulation type in use, as well as information relating to LMI.

CCNA Study Guide Chapter 11 Summary

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 11.** Subscribe to our

RSS Feed

Chapter 11 began with an introduction to WAN technologies, including how they map to the lowest three layers of the OSI model. Different WAN concepts and connectivity techniques were also introduced, including point-to-point, circuit-switched, and packet-switched connections. A look at different carrier lines and speeds compared the differences between the T and E carrier systems used in North America and Europe, respectively.

Point-to-point or leased-line connections were looked at next, including concepts and the configuration of both HDLC and PPP on Cisco routers. While HDLC is the default encapsulation used on synchronous serial interfaces on Cisco routers, PPP provides a standard method of

encapsulating data over point-to-point links, and includes the ability to configure additional features like mutual authentication using CHAP or PAP.

ISDN was introduced next as a circuit-switching technology that uses all-digital links between locations. The configuration of both dedicated and demand-dial ISDN connections were explored, as was a breakdown of ISDN services, protocols, equipment, reference points, and standards.

This was followed by a detailed overview of Frame Relay, a popular packet-switching technology used to interconnect locations using virtual circuits. This included a look at not only the Frame Relay communication process, but also the ways in which virtual circuits are identified, provisioned, and configured on Cisco routers. The configuration of Frame Relay for both production and lab environments was also discussed.

Chapter 12: Network Address Translation

Network Address Translation

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

In Chapter 5 we took a look at how companies have moved to using private IP addresses on their internal networks. The reason for this transition is twofold. Firstly, the rapid growth of the Internet has led to a serious reduction in the number of public addresses available in the IP version 4 address space. While this is being addressed by a new version of IP (IPv6), the wide-scale deployment of IPv6 is likely to take many years to occur. The second reason for the increased use of private addresses is the benefit that they provide from a security and administration point of view. Not only do they allow administrators more flexibility in terms of addressing, these addresses are not routable on the public Internet, providing an additional layer of security for internal systems. The private internal IP address ranges specified in RFC 1918 include:

10.0.0.0/8

172.16.0.0/12

192.168.0.0/16

In order for hosts using private addresses to access the Internet, they require an intermediary device to process their requests. This is usually accomplished through the use of Network Address Translation (NAT), where requests from internal clients for resources on the public Internet are “translated”, such that they appear to have been initiated from a valid public Internet address.

Consider the example network illustrated in the figure below. A company has a small private network with hosts addressed in a private range – 192.168.1.0/24. The router in the illustration is acting as a NAT device, and has one public IP address configured on its S0 interface. When internal hosts make a request for Internet resources, these requests are sent to the router, which is configured as the clients’ default gateway. The router, seeing that the request is for an external Internet address, will “translate” the packet, such that the source address and port number are changed to the public address associated with its S0 interface. The router will store a mapping in its NAT table that keeps track of which client initiated the request, so that the subsequent reply can be forwarded to the correct host.

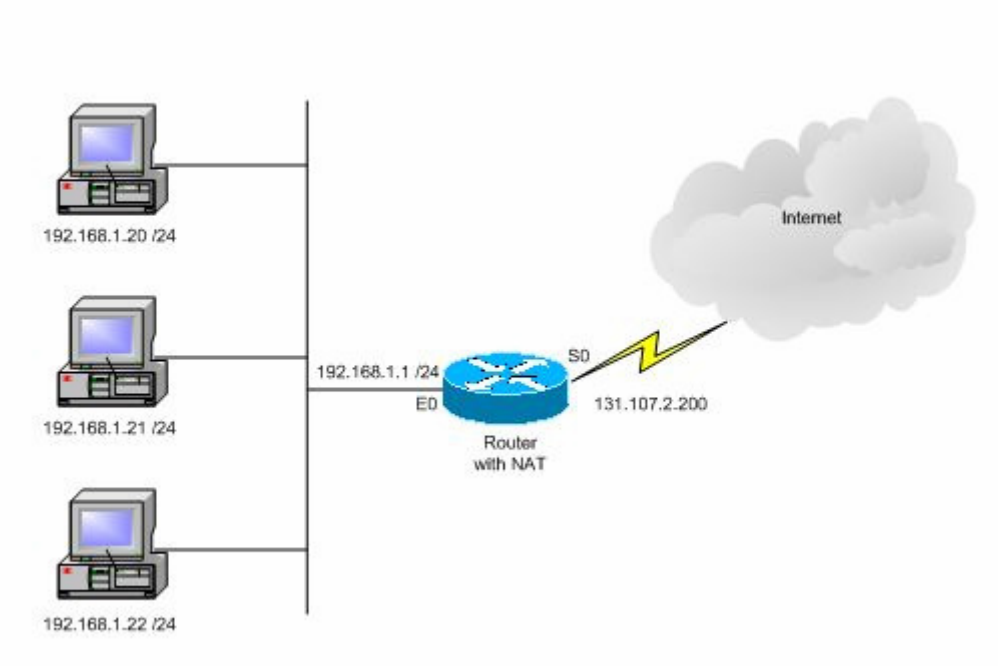


Figure: Internal clients with private addresses gain access to the Internet through the NAT-enabled router.

Before looking at the different ways in which NAT can be implemented on a network, we should first look at what it is that we want to accomplish with NAT. For example, is our goal only to allow internal clients to access the public Internet, or do we also want to allow Internet systems to be able to gain access to certain internal servers? By default, NAT will act as a type of firewall, blocking all requests that do not originate from the internal private network. This allows internal clients to access Internet resources, but stops Internet clients from accessing our internal LAN. In cases where you have a server on your private network that must be accessible from the Internet (such as a web or mail server), NAT must be explicitly configured to forward these requests. If not, all requests that originate from the public Internet will be dropped.

There are a number of different ways in which NAT can be configured. The three most popular NAT implementation techniques are static NAT, dynamic NAT, and what is known as overloading. These techniques can be used individually, or in combination with one another.

Static NAT

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

A static NAT implementation is one in which each private internal IP address is mapped to unique public external IP address. This technique involves defining a static NAT table on the router that maps each internal private address to its external public counterpart. Consider the example illustrated in the figure below. It shows a small network consisting of five client systems, each configured with a private address in the 192.168.1.0/24 range. The router is configured for NAT, and has five external public addresses. The NAT table shown in the example illustrates the mapping between the private and public addresses.

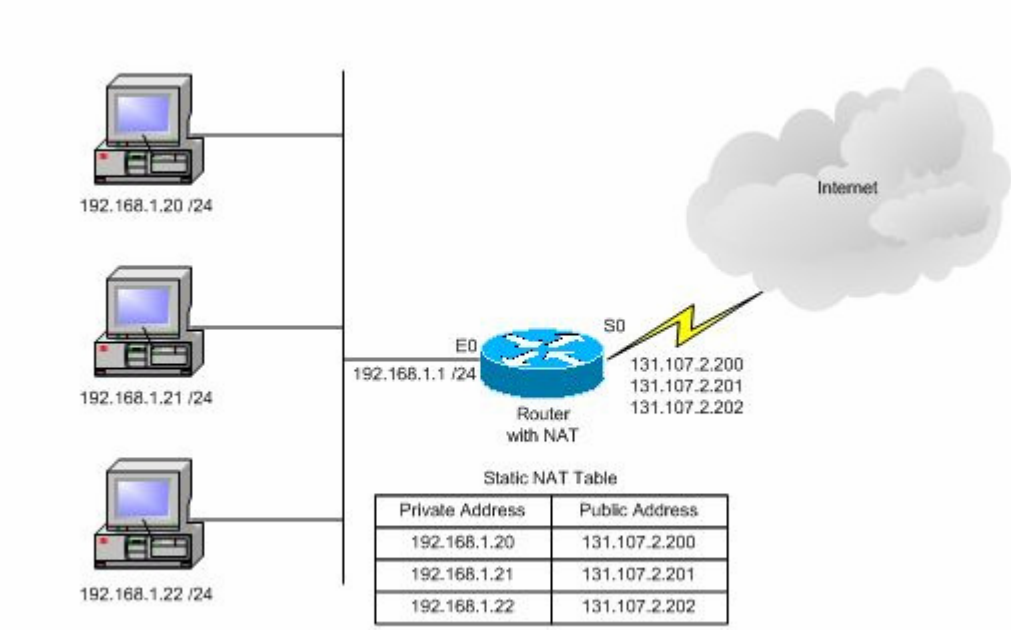


Figure: With static NAT, each internal private address that requires access to the Internet is mapped to a dedicated public IP address.

With static NAT, when client 192.168.1.12 attempts to access an Internet resource, the request will be forwarded to its configured default gateway, 192.168.1.1. When the router receives this packet, it will change the source address to 131.107.1.46, as per the information stored in the NAT table. When the destination web server receives the request, it considers it to have originated from 131.107.1.46. This is also the address to which the subsequent reply will be sent. Once received by the router, it will check its NAT table, and will again translate the packet such that its destination address is changed to 192.168.1.12. The packet will then be forwarded to the internal client.

Companies generally don't implement static NAT for the purpose of allowing internal hosts to gain access to the Internet. It is simply too time consuming to build the NAT table, and companies often do not have an available public IP address for each and every internal host. Instead, static NAT is most often used in order to allow Internet hosts to gain access to internal servers. This will be discussed shortly.

Dynamic NAT

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

Dynamic NAT works slightly differently in that private and public addresses are not mapped on a one-to-one basis. Instead, a range of public IP addresses is configured on the NAT device, and private internal clients will be mapped to an available address as necessary. The NAT table is built dynamically, avoiding the need for mappings to be statically defined. The address translation function that occurs is similar to that with static NAT, with the obvious exception that address mappings may change.

NAT Overloading Port Address Translation (PAT)

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

Overloading is a very popular NAT technique, and is sometimes referred to as Port Address Translation (PAT). Instead of requiring multiple public IP addresses, overloading instead uses a single (or small number) of public address, and differentiates between sessions according to port number. When a client on the internal network wishes to access the Internet, it forwards the request to its configured gateway, the router running NAT. The router will translate the source address and port number of the packet to use the router's public IP address and the same port number (if not already in use by another client), and will forward the "new" packet to the destination host. NAT mappings are stored in the router's NAT table, as shown in the table below.

Internal Source IP	Internal Source Port	External Source IP	External Source Port
192.168.1.54	4085	131.107.2.200	4085
192.168.1.59	5519	131.107.2.200	5519
192.168.1.201	5519	131.107.2.200	4712

Table: The router's NAT table shows a session mappings from three different internal clients.

When host 192.168.1.54 attempts to access the web server at address 204.242.16.4, the request is first passed to the NAT server, where the source address and port number are translated, and a mapping is added to the NAT table. To the external web server, the request appears to be coming from address 131.107.2.200, TCP port 4085. The web server will send its reply to this address and port number. Once received by the router, it will look in its NAT table, and discover that since the packet's destination is address 131.107.2.200 TCP port 4085, it should be forwarded to internal host 192.168.1.12, TCP port 4085. The process is illustrated step-by-step in the figure below.

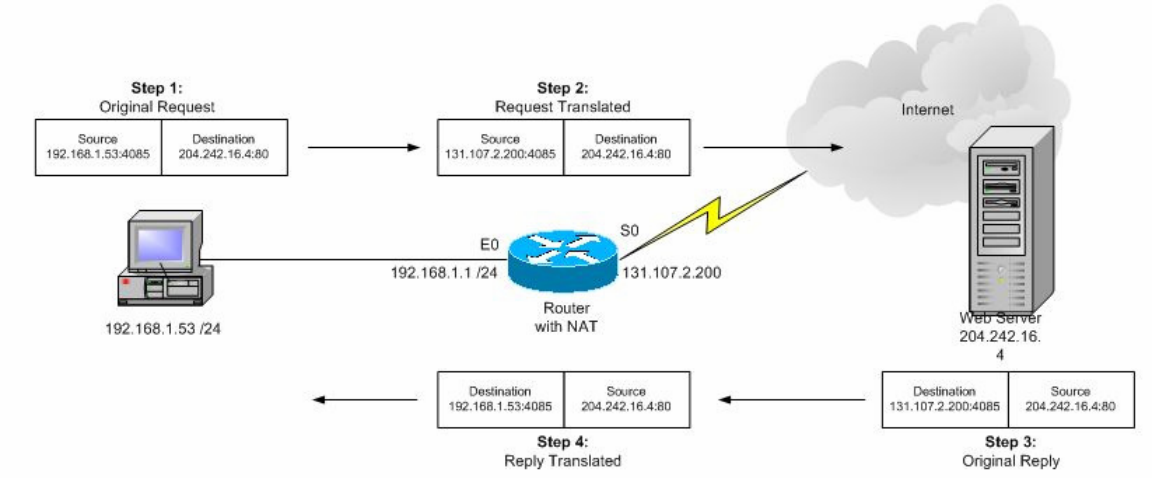


Figure: The steps involved when an internal client forwards a request to the Internet through a NAT router.

The overloading technique is obviously a very efficient way to implement NAT, since it requires only a single public IP address at a minimum. With thousands of TCP and UDP port numbers available, the technique is capable of supporting many internal clients using private addressing.

NAT Inbound Mapping

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

While NAT is most commonly looked at as a way to allow internal clients to gain access to the Internet, it can also be used to allow external Internet hosts to gain access to resources on a private network. Recall that by default, a NAT server will drop all packets that are not replies to requests that were originated from the internal private network. However, it is also possible that your company has servers on its internal privately addressed network that need to be accessible from the Internet – both mail and web servers are good examples. In order to accomplish this, companies will most commonly use what is known as an inbound static mapping. This technique takes requests that are made to a certain ports on the external public interface of the NAT router, and statically maps them to an address and port number on the private network. If multiple public IP addresses are available, individual public addresses can be mapped to internal private addresses on a one-to-one basis.

Imagine that a company wishes to host its web server internally. In order for Internet clients to access our server, it will need to be accessible using a public IP address. In this example, the web server has a private address, 192.168.1.10, and is waiting for connections on the default HTTP port, TCP 80. This is illustrated in the figure below.

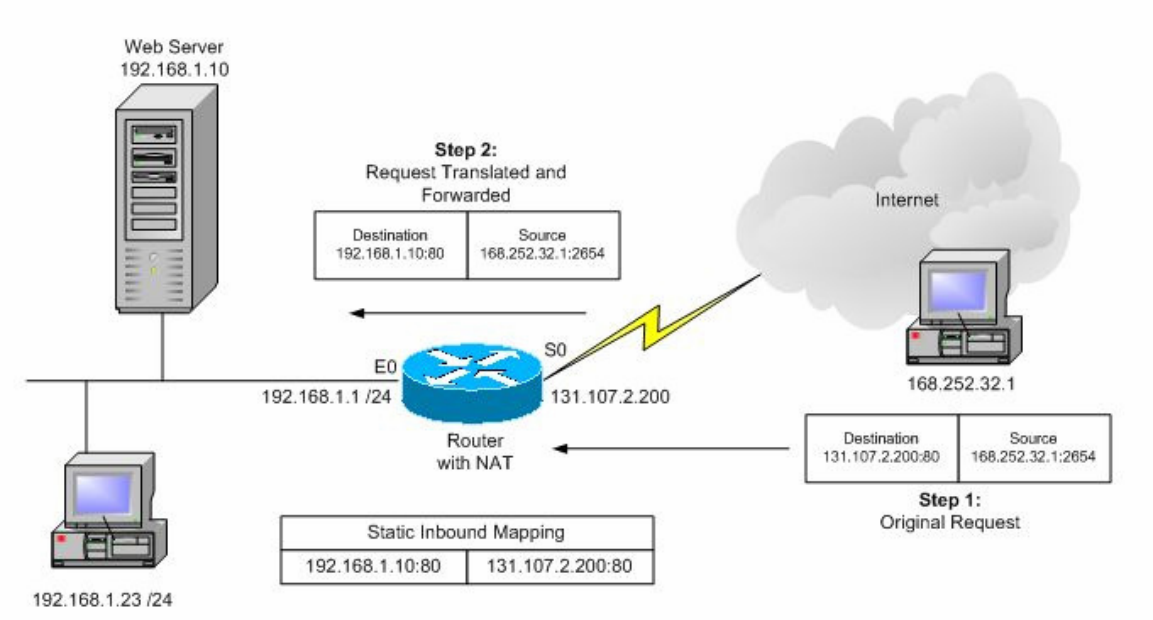


Figure: A web server hosted on the internal private network can be accessed from the Internet if an inbound static mapping is defined.

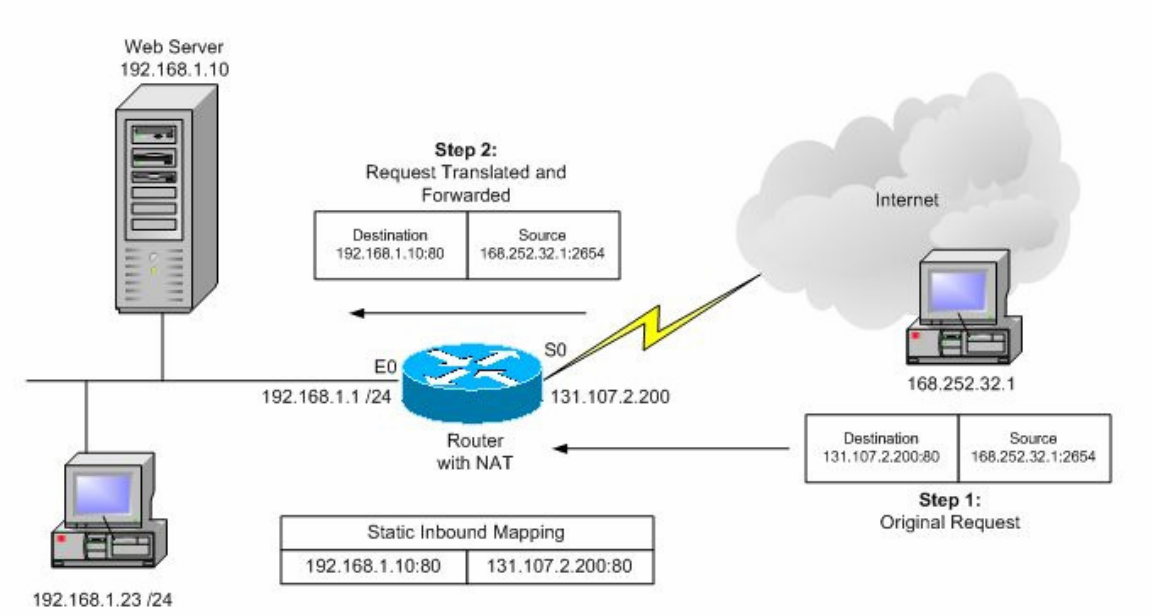
In order to allow Internet hosts to access the HTTP server, we will need to create an inbound static mapping. This will involve configuring NAT such that when it receives a request on the router's public interface that is destined for TCP port 80, it will forward the request to the web server at 192.168.1.10, port 80. To the outside world, it appears as though our web server can be found at the public address. In reality, these requests are being translated by NAT and forwarded to the designated address and port on the internal network. This allows us to host services on the internal network, without external clients being any the wiser as to the true location of a server.

Configuring NAT

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

For the purpose of this example, I'm going to assume that we're using NAT overloading (PAT) to allow our internal clients to access the Internet through a single public IP address. We'll also create a static inbound mapping to allow external clients to gain access to a web server on our private network. The network properties used in this example are the same as those found in this figure.



Our first step involves configuring interface Ethernet 0 with its private IP address, and designating it as the internal NAT interface. The ip nat inside command designates an interface as internal.

```
Router#config t
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)#int e0
Router(config-if)#ip address 192.168.1.1 255.255.255.0
Router(config-if)#ip nat inside
```

The next step involves configuring Serial 0 with an IP address, and specifying it as the external NAT interface. External NAT interfaces are defined with the ip nat outside command.

```
Router(config-if)#int s0
Router(config-if)#ip address 131.107.2.200 255.255.255.240
Router(config-if)#ip nat outside
```

Depending on the NAT technique being used, a range of IP addresses could be configured as part of the NAT "pool". Because we're using NAT overloading, this "pool" will only consist of a single address – 131.107.2.200. The subnet mask associated with an address pool is specified with the prefix command.

```
Router(config)#ip nat pool Toronto 131.107.2.200 131.107.2.200 prefix 28
```

After the pool is defined, it needs to be configured for overloading. This is accomplished using the command shown below, which defines an access list. In this example, access list 88 allows us to control which addresses can access the Internet via NAT.

```
Router(config)#ip nat inside source list 88 pool Toronto overload
Router(config)#access-list 88 permit 192.168.1.0 0.0.0.255
```

Assuming that internal clients are configured with addresses in the 192.168.1.0/24 range, and that their default gateways are set to 192.168.1.1, they should now be able to access the Internet through the router's NAT implementation.

In order to allow external clients to access the web server on our internal private network, we'll create a mapping that tells NAT to forward all requests to address 131.107.2.200 port 80 to the internal address 192.168.1.100, port 80. This is accomplished using the command shown below.

```
Router(config)#ip nat inside source static tcp 192.168.1.100 80 131.107.2.200 80
```

Once implemented, NAT statistics can be viewed using the show ip nat statistics command, while address translations can be viewed using show ip nat translations.

CCNA Study Guide Chapter 12 Summary

By Dan DiNicolo, January 8th, 2007 Posted in **CCNA Study Guide Chapter 12.** Subscribe to our

RSS Feed

Chapter 12 explored concepts relating to Network Address Translation (NAT), outlining how it can be used to allow privately addressed internal hosts to access Internet resources. Three different NAT techniques were looked at, including static NAT, dynamic NAT, and overloading (PAT). The ability to create mappings to allow Internet hosts to gain access to privately addressed internal servers was also discussed.

Dan DiNicolo is a freelance author, consultant, trainer, and the managing editor of 2000Trainers.com. He is the author of the CCNA Study Guide found on this site, as well as many books including the PC Magazine titles Windows XP Security Solutions and Windows Vista Security Solutions. Click here to contact Dan.

http://www.2000trainers.com/

All Practice Exams:

CCNA Practice Exam 1

CCNA Practice Exam 2

CCNA Practice Exam 3

CCNA Practice Exam 4

CCNA Practice Exam 5

CCNA Practice Exam 6

http://www.2000trainers.com/tutorials/practice-exams/