FUJITSU COMPUTERS SIEMENS

Reliant UNIX ONLINE Documentation

Reliant UNIX 5.45

Network Administration

RM200, RM300, RM400, RM600

Edition September 2000

Identification: U42305-J-Z915-2-76

Copyright and Trademarks

1 Preface

The Reliant UNIX 5.45 base system includes functions that enable machines to be integrated in networks. These include, on the one hand, the functions of the TCP/IP protocol family that allow your machine to be connected to a local area network and, on the other, the Basic Networking Utilities (BNU) for communication between UNIX systems (e.g. the *uucp* command).

This manual is part of your operating system documentation. It describes the network administration tasks for the TCP/IP software, the Basic Networking Utilities and the transmission media for local networks.

1.1 Target group

The manual is aimed, first and foremost, at network administrators responsible for managing UNIX systems in TCP/IP networks. However, the manual will also be of interest to system administration staff responsible for networking their own machines and ensuring the smooth running of network applications.

To use the manual, you need to know how to work with UNIX at command level. Basic knowledge of data communication is also useful.

1.2 Summary of contents

This manual describes the network administration tasks that arise when using TCP/IP software on Reliant UNIX 5.45 and the Basic Networking Utilities (BNU).

The fundamentals of TCP/IP communications and the various functions involved are described in detail. You are given information on the tasks which arise during network administration, and solutions are provided in examples. The sections on error analysis and diagnostics describe problems that might occur, and you are given tips on how to solve them.

User functions which support communication between machines in the network are not dealt with in this manual. These are described in the "User Guide".

The "Network Programming Interfaces" manual is available for the programmers of network applications. This describes, among other topics, the TLI and socket interfaces.

For every Reliant UNIX 5.45 network manual, there is a reference manual containing complete descriptions of all the commands, functions, files, etc. in alphabetical order.

1.3 Readme and man files

Please consult the product-specific readme files for any functional changes or late amendments to the current product version that affect this manual.

Final changes affecting the manual are stored in what is known as a man file (supplementary to the manual). If this is the case, the information shipped with the product contains a note to this effect.

If the *SIreadmeM* readme package has been installed, the relevant readme or man files will be stored on your UNIX system under the */opt/readme/<productname>* directory.

Readme and man files are ASCII files which can be viewed using an editor or printed to any standard printer.

1.4 Changes from the previous version of the manual

- In Reliant UNIX V5.45 the TCP/IP protocol stack has been redesigned and standardized. As of Version 5.45, TCP/IP modules are based on 4.4BSD and these modules occur only once in the system. In special cases where the precise behavior before Reliant UNIX Version 5.45 is required, there is an option for switching the system back to the old two-way implementation (see the Section "Selecting the "enhanced" or the "fallback" protocol stack").
- Enhancing the protocol stack has led to a complete reorganization of the IP routing schema. This reorganized schema is no longer in the form of tables, but instead uses a routing tree structure (see Section "IP routing for the "enhanced" protocol stack solution").
- Parameters and measures for optimizing TCP/IP and NFS which influence performance are described in Chapter "Changing system limits and system parameters" and Section "Optimizing operation of NFS" respectively.
- Through use of the "enhanced" protocol stack, it is also possible from Version 5.45B to configure more than one network interface in the same IP network. This gives rises to certain peculiarities, which affect the IP routing and which are described in greater detail in the Chapter "Expanding your network" under the keyword Selective Net Routes.
- Other changes affect, among other things, Dual Homing for Gigabit Ethernet Controllers and the structure of the *ppd.config* file. These changes are described in the Chapter "Transmission media" in Section "Fault tolerance for Gigabit Ethernet Controllers ("Dual Homing")" and from Section "The structure of the ppd.config file" onwards.

1.5 Notational conventions

The following notational conventions have been used in this manual:

fixed-space font	System outputs, inputs, text on the screen, mask or file contents, program examples, names, descriptions, etc.
italics	File names, pathnames, commands, parameters, variables, etc.
CAPITALS	Variable types in programs and for the system as well as abbreviations etc.

Notes and warnings


This symbol draws your attention to particularly important information that you really must read.


This symbol indicates actions that may lead to loss of data or damage to a device.

References

References are specified in the following format:

"Manual title" and/or corresponding reference number in square brackets.

2 Introduction to network support using Reliant UNIX 5.45

This chapter provides basic information on the options offered by Reliant UNIX 5.45 for supporting network activities. The basic version of Reliant UNIX 5.45 contains the software components you will need to connect your system to a local area network on the basis of the TCP/IP protocols and establish links to other systems. This does not apply to connection via FDDI and Token Ring networks. To load and administer the Multibus II controller for FDDI (LCF) and Token Ring (LCT), you require the software product CMX.

It is, of course, also possible for systems running Reliant UNIX 5.45 to be connected to networks with different architectures. To do this, however, you will need additional network software. Options such as these are not dealt with in this manual. You should contact your Siemens customer service for more information on this subject.

This chapter contains introductions to the following subjects:

- Internet communications
- Transmission media
- Communication and network administration functions
- User interfaces

2.1 Introduction to Internet communications

The term "network" implies two or more machines that are connected via communication lines. Data exchanged between two machines is sent/received via the communication line connecting both machines. Because different types of networking software and hardware need to interact to perform this function, network designers developed the concept of the communications protocol family (or suite). A network protocol is a set of formal rules describing how software and hardware should interact within a network in order to transmit information. The Internet protocol family is one such group of network protocols. It is centered around the Internet Protocol (IP). The other members of the Internet protocol family are the Transmission Control Protocol (TCP), User Datagram Protocol (UDP), Address Resolution Protocol (ARP), Reverse Address Resolution Protocol (RARP), and Internet Control Message Protocol (ICMP). Application protocols, such as TELNET, FTP (File Transfer Protocol) and SMTP (Simple Mail Transfer Protocol), are also included.

The entire family is popularly referred to as TCP/IP, reflecting the names of the two main protocols. This is also the term used in this document.

TCP/IP has been designed for use in heterogeneous networks, i.e. it can be used on many different types of machine, as well as in wide area networks (e.g. X.25-based networks) and local area networks (e.g. Ethernet-based).

TCP/IP was originally developed by the United States Department of Defense to run on the ARPANET, a packet-switching wide area network (WAN) first introduced in 1972. Today the ARPANET is part of a wide area network known as the DoD (Department of Defense) Internet, or, for short, the Internet. Many popular texts use the term Internet to describe both the protocol family and the wide area network of the DoD. In this manual, the term TCP/IP refers to the Internet protocol suite, and Internet refers to the network itself.

2.1.1 The Internet protocols

The TCP/IP protocol structure can be thought of as a series of layers, as shown in the table below.

Layer	Network services
Application	TELNET, FTP, SMTP
Transport	TCP, UDP
Internet	IP, ICMP
Data link	ARP, RARP, device driver (such as Ethernet ')
Network access	Cable or other medium (such as an Ethernet board)

In TCP/IP, a machine engaged in communication is referred to as a sender or a recipient. Every protocol layer on the sender has its peer protocol layer on the recipient. Each layer is required by design to handle communications in a pre-determined fashion.

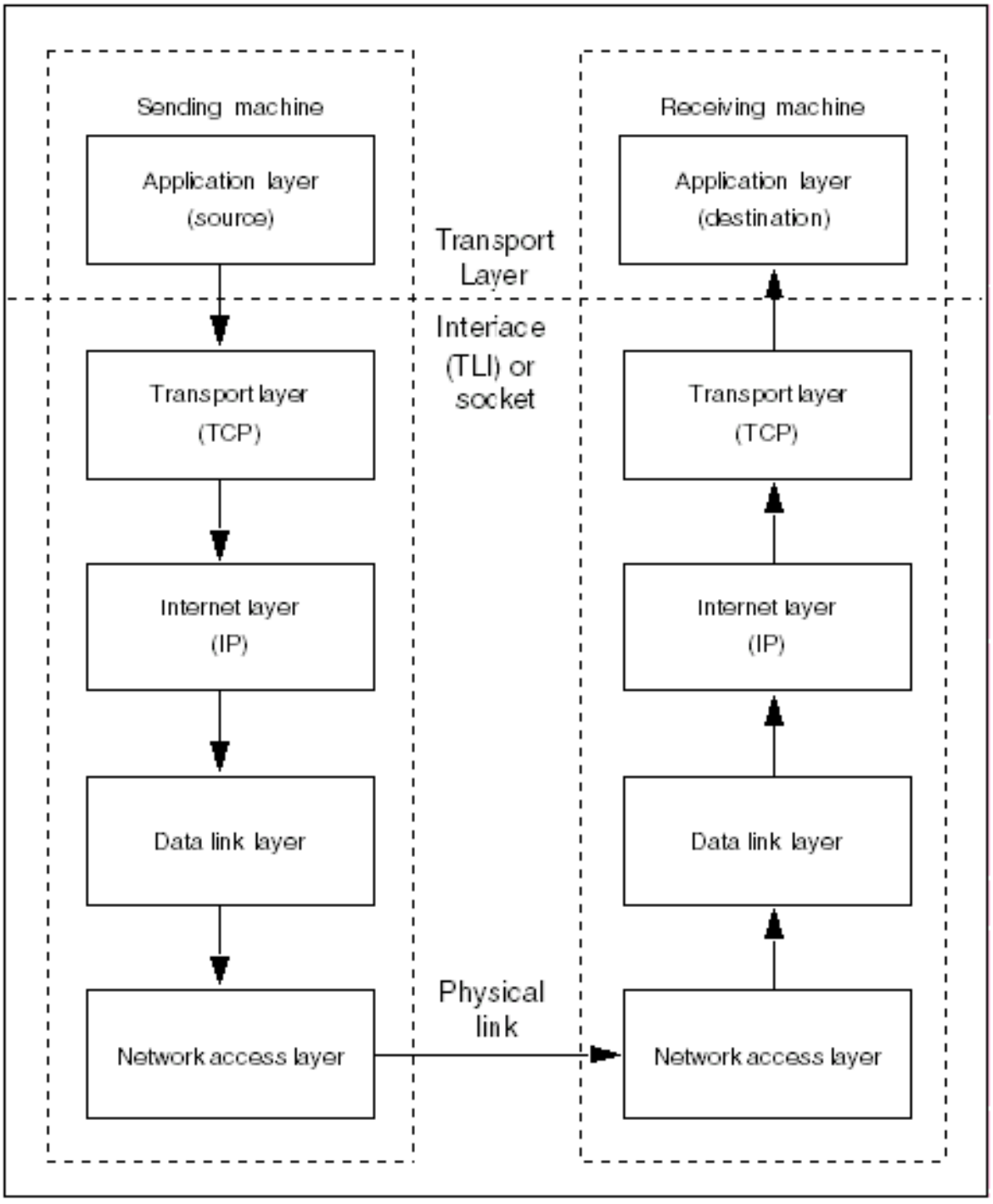


Figure 1: Interaction between sender and recipient

Each protocol formats communicated data and appends or removes information from it. Then the protocol passes the data to a lower layer on the sending machine or to a higher layer on the receiving machine (see Figure "Interaction between sender and recipient").

The Transport Layer Interface (TLI) and the socket interface are the interfaces between the application layer and the transport layer. Any application written for the TLI or the socket interface can run in a TCP/IP network. For more information on the TLI and sockets, please refer to the "Network Programming Interfaces" manual.

The next subsections briefly explain how each protocol layer handles message transfer. For more detailed information, refer to the description of the relevant protocol in the "Networking Reference Manual".

Network access layer

This layer defines how data is transmitted via a line or other medium. The protocols of this layer send and receive data in the form of packets. A data packet contains a source address, the transmission itself and a destination address.

Ethernet', FDDI and Token Ring' are examples of protocols on this layer. Ethernet is an example of a packet-switched network; its communication channels are occupied only for the duration of the transmission of a packet. The telephone network, on the other hand, is an example of a circuit-switched network. The next section contains information on the transmission media supported.

Data link layer

The data link layer is responsible for the addressing of the systems connected to the network by means of MAC addresses. The MAC address uniquely identifies a LAN controller (see also the Section "Machines in the TCP/IP network").

Two additional TCP/IP protocols exist between the Internet and data link layers:

- ARP (Address Resolution Protocol) maps known Internet addresses to MAC addresses.
- RARP (or Reverse ARP) is the counterpart to ARP. It maps known MAC addresses to Internet addresses.

Internet layer

IP (Internet Protocol) and ICMP (Internet Control Message Protocol) are the protocols on this layer.

IP is responsible for machine-to-machine communication and transmission routing, i.e. it determines the path a transmission must take, based on the receiving machine's Internet address. The section entitled Internet addresses discusses Internet addresses in detail. IP also provides transmission formatting services; it assembles data for transmission into an Internet datagram. If the datagram is outgoing (received from a higher-layer protocol), IP attaches an IP header to it. This header contains a number of parameters, including the Internet addresses of the sending and receiving machines. Refer to the description of *IP(7)* in the reference part of the manual.

Error or control messages are sent to other machines via ICMP. ICMP also controls Internet software communication between machines. Refer to the description of *ICMP(7)* in the "Networking Reference Manual" for more information about ICMP.

Transport layer

The TCP/IP transport layer protocols TCP and UDP enable communication between processes running on separate machines.

TCP (Transmission Control Protocol) enables applications to talk to each other via "virtual" circuits, as though they had a physical circuit between them. TCP is a connection-oriented protocol that can recognize transmission errors and is therefore very reliable: if the data transmitted by the sender does not arrive at the recipient in the specified order, this is identified as an error and corrected.

UDP (User Datagram Protocol) is the alternative protocol available at the transport layer. UDP is a connectionless datagram protocol. Datagrams are groups of information transmitted as a unit to and from the upper layer protocols on sending and receiving machines. UDP datagrams use port numbers to address sending and receiving processes. However, unlike with TCP, there is no mechanism for detecting transmission errors, incorrect datagram order or data loss. Errors must be dealt with on the application layer.

The application determines whether TCP or UDP is used. For example, if the user invokes *telnet*, the application passes the user data to TCP. The application TFTP, on the other hand, uses UDP as the transport protocol.

For more information on these protocols, refer to the descriptions of *TCP(7)* and *UDP(7)* in the "Networking Reference Manual".

Application layer

The TCP/IP family of protocols provides a whole series of application layer protocols for handling communication and administration functions in the Internet. They include protocols for:

- establishing sessions with remote systems - *telnet* and *rlogin*
- transferring files - *ftp* and *rcp*
- distributed file systems - NFS (Network File System)
- network administration - NIS (Network Information Service) and DNS (Domain Name System)

The section entitled Communication and network administration functions contains a brief description of these functions.

2.1.2 Machines in the TCP/IP network

With TCP/IP, a basic distinction is made between two types of machine: local machines and remote machines.

The local machine is always the one at which you are currently working. This means that the local machine is the one whose name is output when you execute the command uname -n.

As far as you are concerned, all other machines are remote.

2.1.2.1 Machine addresses

In order to be able to address a certain machine in the network without cases of mistaken identity occurring,

two addresses are assigned to each machine:

- the Internet address (IP address) of the machine
- the MAC address of the relevant network interface

Internet address (IP address)

The Internet address is an address, unique in the network, that is assigned to a computer or another device connected to the network (such as an X terminal or a printer). Unique in the network means that this address may only occur once within networks that may be connected to each other. It is assigned when the machine is connected to the network, and is stored together with a symbolic name in the */etc/inet/hosts* file on the local machine.

You will find additional information on specifying Internet addresses in the section entitled Internet addresses in chapter 3.

MAC address

The MAC address serves to uniquely identify a LAN interface. It is 6 bytes long and is assigned to each controller by the maker of the controller when it is manufactured.

This means that every controller in the world can always be identified no matter which machine it is installed on.

The first 3 bytes of the MAC address are identical for all the controllers manufactured by a specific company.

The next three bytes are unique and are assigned by the individual manufacturer.

2.1.2.2 Machine names

A symbolic name is assigned to each Internet address when a machine is connected to the network. This name is freely selectable as long as it remains unique within the network. It may consist of up to 24 characters. The characters A-Z, a-z, 0-9, hyphen (-) and period (.) are permitted. The first character must be a letter, the last character must not be a hyphen (-) or period (.). A period is only permitted if it is used to separate the components of a domain name (RFC 921). No blanks are permitted and no distinction is made between uppercase and lowercase. Although Reliant UNIX actually supports machine names that are up to 63 characters in length (without domain components), a restriction to 24 characters ensures maximum interoperability with the Internet. The same applies to names for networks, gateways and to domain name components separated by a period.

The machine can be addressed by this name. It is a synonym for the Internet address; a name is usually easier to remember than an address.

When working with the communication functions you only need to know the machine names of the machines with which you wish to communicate.

Each machine name may be assigned alternative names, which are known as aliases. These aliases may also be used to address a machine in the network. Like machine names, aliases must also be unique within the network.

2.2 Transmission media

This chapter provides an overview of the transmission media you can use when running Reliant UNIX 5.45 with TCP/IP transport protocols. It outlines the essential features of the various media and the differences between them. Technical details and configuration and maintenance information can be found in the chapter entitled Transmission media.

The Reliant UNIX 5.45 network software supports the following transmission media:

- Ethernet
- FDDI
- Token Ring
- Serial connections via SLIP (Serial Line Interface)
- Serial connections via PPP (Point-to-Point Protocol)

The product CMX is required to load and administer FDDI and Token Ring controllers on RM600 machines.

2.2.1 Ethernet

Ethernet is a physical network with a bus topology. The data transfer rate in the Ethernet is 10 Mbit/s (million bits per second), 100 Mbit/s (for Fast Ethernet) or 1 Gbit/s (for Gigabit Ethernet).

There are currently six main types of Ethernet in use:

10BASE5

(Thick Wire or "Yellow Cable")

This is a thick coaxial cable. It can have a length of up to 500 m and can be set up with a maximum of 100 connections.

10BASE2

(Thin Coax or "Cheapernet")

This coaxial cable is thinner that the 10BASE5 cable and can have a maximum length of 185 m. Up to 30 stations may be connected.

10BASE-T or UTP
(Unshielded Twisted Pair)

This cable can be up to 100 m long and connects two devices. UTP cables are the same cables used to install telephones.

10BASE-FL
(Fiber optic)

This cable is a fiber optic cable that can have a maximum length of two kilometers.

100Base-T
(Fast Ethernet)

The UTP CAT5 cables (category 5 unshielded twisted pair) that are used connect two devices together; they may be up to 100 m long.

1000Base-SX
(Gigabit Ethernet)

The name SX on the physical layer specifies transfer using a short-wave laser via a multi-mode fiber optic cable. 62.5 micrometer fiber optic cable with a length of between two and 260 meters or 50 micrometer fiber optic cable with a length of between two and 550 meters can be used. The cable connects precisely two devices (see also [
Gigabit Ethernet Rich Seifert, Addison Wesley]).

Unlike other networks, Ethernet does not have a central management system controlling access to the network. Network access is regulated via an access procedure known as Carrier Sense Multiple Access with Collision Detect (CSMA/CD). Each link in the network is equally authorized to send data within the network provided the latter is free.

2.2.2 Token Ring

In a Token Ring network (IEEE 802.5 standard), one or more tokens (with a size of 1 byte) make a continuous circuit of the ring and are forwarded by every station connected to it. When a token arrives at a station, that station can stop the token and send a data packet around the ring. When the data packet arrives back at the station, the station regenerates the token on the network, thus giving the next station the chance of sending a packet.

Transmission rates of 4 Mbit/s and 16 Mbit/s are supported. At 4 Mbit/s one token traverses the ring, and at 16 Mbit/s there are four tokens.

The most important differences to Ethernet (IEEE 802.3) are:

- the use of a collision-free protocol
- the topology (ring instead of bus)
- the ring management options

A minimal Token Ring consists of one MAU (Medium Access Unit), to which eight stations are connected via data cables.

In order to use the LCT Multibus II Token Ring controller on RM600 systems, the software products CCP-Token Ring and CMX (Communications Method in SINIX) must be installed on your system. For more information, please consult the section entitled Loading the Token Ring and FDDI controllers (RM600 LCF, LCT only) in chapter 10.

CMX is not required in order to use the PCI Token Ring controller *madge*.

2.2.3 FDDI - Fiber Distributed Data Interface

The FDDI standard (ISO 9314-x) is a fiber optic-based ring network with a transmission speed of 100 Mbits/sec. A duplicate ring is used to enhance fail-safe performance; this second ring is currently not used in error-free operation.

FDDI also uses a token-based procedure similar to the one described in the previous section for Token Ring.

The ring can be up to 100 kilometers in length, and up to 500 stations (nodes) can be connected to it. No node can be more than 2 kilometers from the next one.

In accordance with the FDDI standard, several types of node exist, and these can be combined to form different network topologies.

In order to use an LCF Multibus II FDDI controller on RM600 systems, the software products CCP-Token Ring and CMX (Communications Method in SINIX) must be installed on your system. For more information, please consult the section entitled Loading the Token Ring and FDDI controllers (RM600 LCF, LCT only) in Chapter 10.

CMX is not required in order to use the PCI-FDDI controller *dfe*.

2.2.4 TCP/IP via serial lines

Two protocols are available for TCP/IP connections via serial lines. They enable TCP/IP connections to be operated on serial lines without the need for a LAN controller.

- SLIP (Serial Line Interface Protocol)

  SLIP supports all network communication functions. Machines that are interconnected via SLIP can set up connections with *rlogin*, *rsh*, *rcp*, *ftp* and *telnet*.

  SLIP connections are implemented via serial interfaces on the machine. Machines that want to communicate with each other directly via SLIP can be connected via modem or null modem cable.

- PPP (Point-to-Point Protocol)

  The Point-to-Point Protocol is responsible for setting up and controlling point-to-point connections between two machines.

  The machines can be connected to each other via a direct line (null modem) or a modem line.

For more information, please consult the section entitled SLIP - TCP/IP connections via serial lines and the section entitled PPP - point-to-point protocol in the "Transmission media" chapter.

2.3 Communication and network administration functions

The UNIX system Reliant UNIX 5.45 also offers the functions required to establish communication relations between systems in a network and to perform management tasks in that network. Other chapters in this manual describe the network management tasks to be carried out by the system administrator and the functions available to simplify these duties. The simple communications functions available to all network users for the purpose of setting up links to remote machines are described in the manual entitled "User Guide".

The following is a brief summary of the most important functions:

Opening sessions on remote machines

- *telnet*

  The Telnet protocol enables communication between terminals via terminal-oriented processes in a TCP/IP network. The *telnet* command is called on the local machine and the *telnetd* daemon on the remote machine. Telnet provides a user interface that enables the user to open a login session on the remote system. It does not matter which operating systems are running on the two machines.

- *rlogin*

  rlogin also lets you open a login session on a remote machine. The *rlogin* command is called on the local system and the *rlogind* daemon on the remote system. The *rlogind* daemon starts a normal login session on the remote machine. In contrast to *telnet*, however, this is only possible between two UNIX systems.

File transfer between two machines

- *ftp*

  The File Transfer Protocol (FTP) controls the transfer of files between two machines. The *ftp* command runs on the local machine, and the *ftpd* daemon on the remote machine. The operating system used by the two machines is of no significance. *ftp* activates a command interpreter that provides a range of commands. *ftp* together with the interpreter enables an interactive session to be executed on the remote machine.

  Along with the commands for file transfer, others exist for creating directories, navigating through the file hierarchy and deleting files.

- *tftp*

  The Trivial File Transfer Protocol (TFTP) also enables users to transfer files between two machines. Like *ftp*, *tftp* is implemented as a command on the local machine and as a daemon (*tftpd*) on the remote machine. However, *tftp* does not support interactive sessions.

- *rcp*

  rcp can be used to transfer files between UNIX systems.

Executing commands on remote machines

- *rsh*

  rsh can be used to execute commands on remote UNIX machines.

Distributed file system (Network File System)

- NFS

  The NFS decentralized file system enables local file hierarchies or parts of them to be added to file systems installed on remote machines. The whole process remains transparent for the user. The user can access the files added to the remote system just as easily as those on his or her own local system. Administration tasks related to NFS are described in the chapter entitled The Network File System (NFS).

Network administration and information functions

- Domain Name System

  The Domain Name System (DNS) is a protocol that maps the names of addressable objects, such as machines or mailboxes, to addresses on a network-wide basis.

  A detailed description of the Domain Name System can be found in the chapter entitled Using the Domain Name Service.

- Network Information Service (NIS)

  The Reliant UNIX operating system comes with an extensive network administration system, the Network Information Service (NIS). This service makes it possible to manage the machines in a local area network from a central point.

  NIS is an information system that contains data on how a group of machines is organized within a network. Commands are available for accessing this information.

  A detailed description of the NIS can be found in the chapter entitled The Network Information Service (NIS).

- SNMP

  The SNMP agent (Simple Network Management Protocol agent) is a tool for monitoring networks with TCP/IP protocols. In a network there are one or more network management stations from which the network is monitored and administered by means of the non-proprietary protocol SNMP. The SNMP agent runs as a daemon process.

  You will find more information in the chapter entitled Network monitoring with SNMP agents.

2.4 User interfaces for network administrators

As a network or system administrator, a number of options are available to you for integrating your system in a network, for working in the network, and for keeping your system's communication functions in good working order.

- You can perform all your network and system administration tasks at command level.
- The *sysadm* system administration user interface provides the SIfmllan menu system for dealing with the most important tasks.
- If you have a graphics-capable monitor and have the SINIX/windows graphical user interface installed on your system, you can perform the main network administration functions by clicking on the items displayed in the graphical interface.
- Web-based system configuration and administration is possible using *WebSysAdm* (WSA).

This manual mainly deals with network and system administration tasks at command level. One main focus is on the interrelationships between and the backgrounds of the various network functions. This information will also be relevant to users who use the menu system or the graphical user interface in the course of their daily work, in particular when problems and errors occur which cannot be diagnosed using the conventional means offered by the user interfaces.

The following chapters deal first with the network functions at command level before proceeding to the corresponding functions in the graphical user interface and how to use SIfmllan menus. A detailed description is not provided in this manual. The SIfmllan menu system, the graphical user interface and *WebSysAdm* have help systems.

A description of the SIfmllan user interface can be found in the manual "System Administration and Hardware Configuration using the SYSADM User Interface". Information on operating the SINIX/windows graphical user interface can be found in the SINIX/windows manuals. The manual "Reliant UNIX Operation" contains an introduction to *WebSysAdm*.

3 Configuring and administering TCP/IP networks

This chapter explains how to set up and administer a network based on the TCP/IP Internet protocol family implemented in Reliant UNIX 5.45. To carry out the necessary administrative work, you require an understanding of the concepts underlying them. Many of the administration tasks can be performed using the NIS network information system. This is described in the chapter entitled The Network Information Service (NIS).

This chapter concentrates on providing general information that will allow you to install the network most suited to your needs. The most important configuration files are described, and important information on error analysis and diagnostics is provided. Information on the BOOTP server and on the daemon for time synchronization is also provided.

The chapter is directed at system administrators already experienced in the administration of distributed systems. In addition, it is assumed the reader has a basic understanding of computer networking and data communications.

3.1 Overview of TCP/IP network administration

This section provides an overview of the network administration tasks and tells you where to find more detailed descriptions on the various topics.

Managing a TCP/IP network involves the following tasks:

- Planning the network

  Before you set up a network, you should first consider its logical structure. This includes defining addresses and names for the various machines and subnets, and deciding whether your network is to be part of the Internet domain. The next section contains more information on this topic.

- Defining the network administration structure

  Before a network is set up or extended, its administrative structure must be defined. For instance, do you want to set up a single network or several networks? Do you want networks to be subdivided into subnets? Consult the chapter entitled Expanding your network for more information on this topic.

- Installing the software and configuring the network interfaces

  The TCP/IP software is shipped preinstalled together with the operating system. Certain network configuration settings are made at installation (e.g. definition of the machine name and Internet address). The configuration files are also created. Once installation has been carried out, the network interfaces need to be configured and the network software started. Consult the section entitled Basic configuration for more information on this topic.

- Installing the NIS network information system

  This service supports network administration. If you want to make use of this service, you should first install an NIS master server, define any slave servers, then integrate the individual systems in the network. Consult chapter 7, The Network Information Service (NIS), for more information on this topic.

- Administering configuration files for machines and network services

  Any network has a series of files containing information on the connected machines and installed services. These files need to be kept up-to-date at all times. Consult the section entitled TCP/IP configuration files for more information on this topic.

- Implementing security measures

  In order to attain a degree of protection against unauthorized accesses via the network, you have to define and possibly modify authorization rights. Consult the section entitled Access options and access permissions for more information on this topic.

- Diagnosing errors when there are network problems

  TCP/IP offers a range of commands to support the system administrator when locating the cause of a network problem. Consult the section entitled Error analysis and diagnostics for more information on this topic.

3.2 Preparations

This section explains which fundamental issues need to be sorted out before you can start installing a TCP/IP network.

- Which Internet network addresses do you want to define for your network(s)?
- Which Internet addresses do you want to define for your systems?
- Which DNS domain names do you want to define?

If you are planning to connect several networks to one another, you will need to configure certain machines as routers. Other preparations are necessary in this case, and you will need to consult the chapter entitled Expanding your network.

3.2.1 Internet addresses

Sometimes packets travel only from one machine to another on the same local network. At other times, they travel from one local network to another network, going through a router. In the extreme, a packet may traverse many networks, crossing miles of routers, gateways, cabling, and telephone lines to reach its destination.

A TCP/IP network routes a packet according to the destination Internet address - an address provided by the IP protocol on the sending machine. To run TCP/IP properly, every machine on the network must have an Internet address.

The Internet address, which is also known as IP address, is a number four bytes in length. It is represented by four fields separated from each other by a period in each case. Each field represents a byte in decimal notation.

The Internet address is subdivided into the following fields:

- a network number
- a machine number
- a subnet number (optional)

The meaning of these fields is described below.

The network number is the first part of the Internet address. It can be 1, 2 or 3 bytes long. The remaining bits of the Internet address are for the subnet number (optional) and the machine number. The subnet number enables the network to be subdivided into subnets. The machine number uniquely identifies a specific machine in a network with common network and subnet numbers, a logical network. This logical network is referred to here as a local area network. Machines in a local area network can communicate with each other directly. When a number of local area networks are linked to each other, the machines in the different networks have different network and/or subnet numbers. Here, too, the machines can communicate with each other provided certain precautions are taken (see chapter 4, Expanding your network).

Network classes

The network number is divided into the following five classes. In this explanation we assume that no subnet numbers are used.

- Class A networks

  Class A networks contain a large number of machines, and there are therefore not very many of them. Typically, Class A networks belong to large organizations or universities. With Class A Internet networks, the network number occupies the first of four bytes. Class A Internet networks are assigned numbers from 1 to 127. The network number 127 is not assigned to networks. The address 127.0.0.1 is used as the loopback address of each machine.

- Class B networks

  Although still relatively large, Class B networks are much smaller than those of Class A. For example, the NIC assigns Class B addresses to medium-sized companies with a large number of machines spread across several networks. With Class B Internet addresses, the network number occupies two of four bytes. Class B networks are assigned numbers from 128 to 191.

- Class C networks

  Class C networks have a maximum of 254 machines. With Class C Internet addresses, the network number occupies three of the four bytes. Class C networks are assigned numbers from 192 to 223 in the first byte.

- Class D networks

  Class D is reserved for multicast addresses. Multicast messages are send to a defined group of machines. Network numbers from 224 to 239 in the first byte belong to this class.

- Class E networks

  Class E (networks from 240 to 247 in the first byte) is not currently used.

3.2.1.1 Selecting/assigning the network number

If you do not plan to connect your network to the official Internet, you can assign the network number yourself. However, it is advisable to request an official Internet address. Should you ever require the services offered on the Internet, such as the mail server, DNS name server or WWW server, your network needs to have an official Internet network address. The Network Information Center (NIC) maintained by SRI International is responsible for handling network registrations and assigning network addresses.

To obtain an Internet network address, request the registration from the NIC. You will find information on how to complete this form in the appendix of this manual.

In order to obtain an Internet network address, you must specify the desired network class.

When you choose a network class, choose the smallest that will accommodate network growth over the next few years. (The chances are slim that your network will need, or that NIC will grant, Class A classification.) For a fairly large organization that routes together local area networks in several buildings, consider requesting a Class B address. This is a good idea, particularly if you plan to subnet part of your local networks in the future. If your network will not conceivably support more than 254 machines in the future, ask for a Class C address.

As soon as you have received an Internet network address from the NIC, you can assign your machines Internet addresses. It is advisable to define all the addresses in writing or in an ASCII file first before entering them in the corresponding files.

The following table indicates the Internet address structure when there is no subnet number:

	Address range	**Network number**	**Machine number**
Class A	1-126	xxx	xxx.xxx.xxx
Class B	128-191	xxx.xxx	xxx.xxx
Class C	192-223	xxx.xxx.xxx	xxx

You assign the machine-specific part of the Internet address in the remaining octets after the network number. In the unlikely event that your machines are on a Class A network, you have three octets for defining the machine number. On a Class B network, you have two octets; on a Class C network, one.

You may assign values from 0-255 to each octet, provided all the octets in the machine number do not have values of 0 or 255. (The addresses 0 and 255 are reserved for broadcasting.) If, for example, you have a Class A network where the net address is 10 (without division into subnets), a machine could have the address 10.30.0.107 or 10.1.1.255, but it could not have the address 10.0.0.0 or 10.255.255.255; in the same way, if your network is Class C, with the network number 192.9.90, you should not assign the addresses 192.9.90.0 or 192.9.90.255 to any of its machines.

Suppose you want to install a machine named dancer on your Class C network. If this network has the network number 192.9.200, you might assign dancer the following Internet address

192.9.200.1

Then you might install other machines on the same network with Internet addresses such as 192.9.200.2, 192.9.200.3, and so on.

3.2.1.2 Selecting/assigning subnet numbers and using network masks

Subnets allow you more flexibility when assigning network numbers. A subnet number is not defined until the subnet is set up. It is assigned by the system administrator rather than the NIC. Thus, the Internet address of a system only contains a subnet number when the machine is connected to a subnet.

Typically, you create subnets by using a subnetting scheme called the address mask. When setting up your network, you should select a network-wide network mask. A network mask determines which bits in the Internet address will represent the subnet number. The remaining bits represent the machine within the subnet. For example, you could configure an organization's internetwork as a Class B network. Then you could assign each local subnet its own subnet number within that network. The 16 bits of the machine number could be allocated as eight for the subnet and eight for the machine, or nine for the subnet and seven for the machine, and so on. Your decision would be transparent to everyone outside that organization. The following

figure illustrates the effect of the network mask:

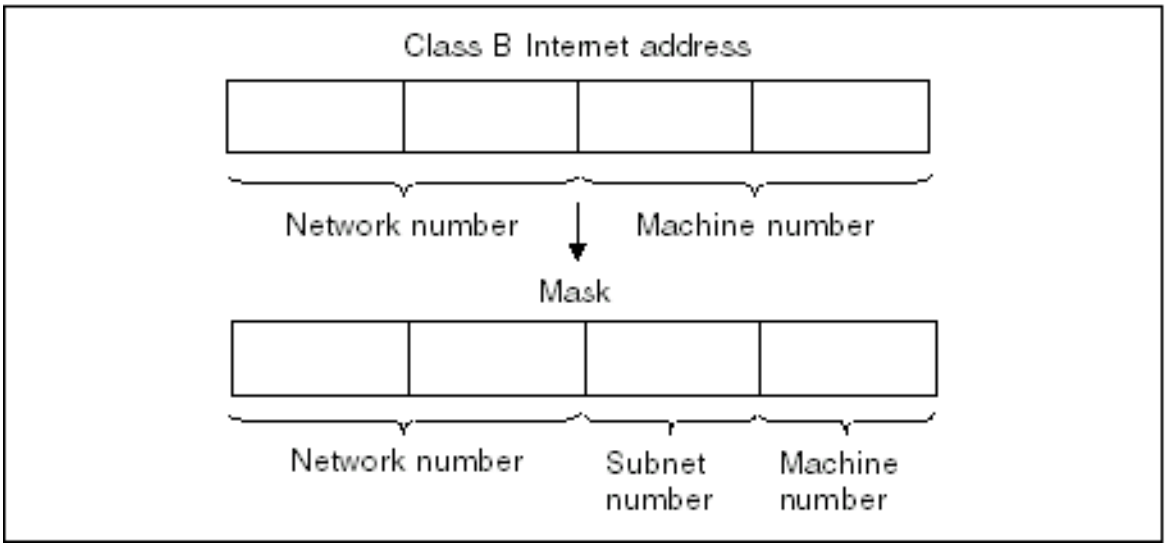


Figure 2: Network mask

You can express network masks as a single hexadecimal number or as four octets of decimal numbers. The default is a mask of 0xFF000000 (255.0.0.0) for Class A networks, 0xFFFF0000 (255.255.0.0) for Class B networks, and 0xFFFFFF00 (255.255.255.0) for Class C networks. You only have to specify a network mask explicitly when it is wider, i.e. consists of more one-bits, than the default mask. It should be noted that the subnet addresses "all 0" and "all 1" cannot be used

One common case is a Class C mask on a Class B network. A Class B network provides you with 254 possible subnets, each one of which can accommodate 254 possible machines (remember, 0 and 255 are reserved for other purposes). But you may know that none of your subnets will ever have more than, say, 126 machines, while you may need more than 256 subnets. In that case, you could decide to use nine bits for the subnet number instead of eight, and seven for the machine number. The appropriate mask for this would be 0xFFFFFF80, or 255.255.255.128.

Given the above scheme, and a network number of, for instance, 131.60, the address for the first machine in the first subnet would be 131.60.0.129.

3.2.2 Establishing a DNS domain

Once the Internet addresses have been selected and the software has been installed, you can establish DNS domains. A DNS domain is a set of machines that are administered and maintained as a single entity and served by a DNS server. Generally, all the machines on a local network comprise a domain on the larger network. However, you may choose to subdivide the local network into several administrative entities, called subdomains. For example, you may want all the machines in the Accounting department to comprise one subdomain, and all the machines in Marketing to comprise another.

To join the Internet, your domain must be named according to Internet naming conventions, and you must register your domain name with the NIC. Even if you do not intend to join the Internet at this time, it is recommended that you follow the Internet naming conventions and register with the NIC to avoid problems in the future, should you decide to join the network.

The following sections explain the Internet naming conventions. The appendix contains instructions for obtaining and completing the Internet domain registration form. If you intend to join a public network other than the Internet, such as BITNET or CSNET, contact the organization that administers the network for information about naming conventions and for a registration form.

3.2.2.1 The organization of the Internet

The Internet consists of four levels of domain: the root level, the top level, the second level, and the local level. Each level branches from the root level. Below the local level are machines and, optionally, subdomains.

The following illustration shows the different domain levels of the Internet and how they relate to each other.

The univers subtree is fictitious and serves as an example.

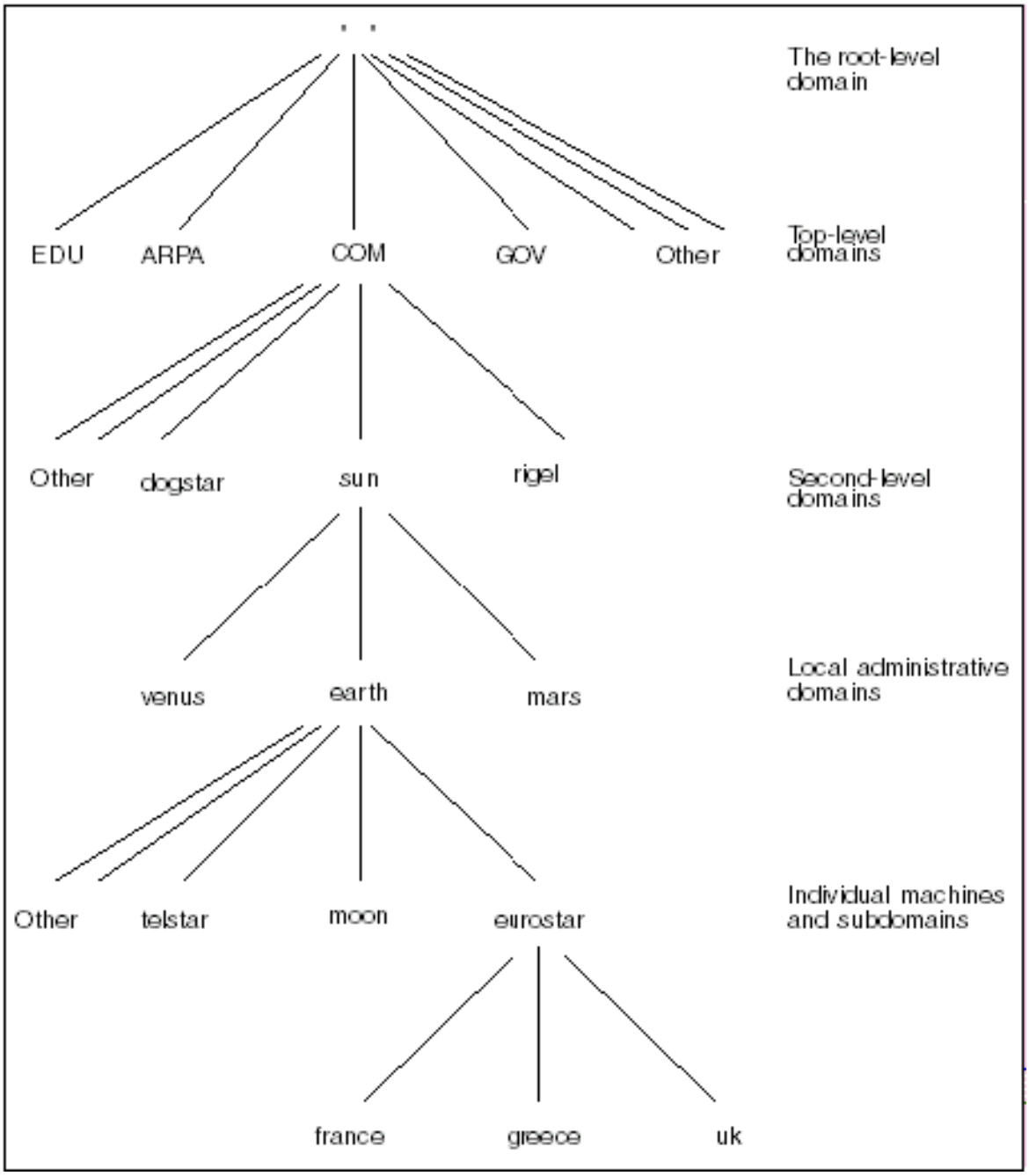


Figure 3: Example of an Internet domain hierarchy

At each level of the Internet there are name servers. A name server is a system that maintains information about servers at the next lower level and facilitates name-to-address mapping.

The different levels of the Internet are described below:

- The root-level domain

  The root level is the top of the entire Internet; it is maintained by the NIC. (Organizations in charge of other public networks, such as BITNET and CSNET, also administer the roots of their networks.)

  At the root level, the NIC administers root domain name servers, which maintain information about name servers at the next lower level.

- Top-level domains

  The root level branches into top-level domains.

  ISO country codes, for example DE for Germany, FR for France, NL for the Netherlands, etc. are used for top-level domains outside the USA.

  When you register with the NIC, it assigns your network to the appropriate domain.

- Second-level domains

  Each top-level domain branches into second-level domains. Member organizations at the second level appoint domain administrators to manage their name servers, and the NIC assigns a technical contact to coordinate administration across domains. Depending upon your site's size and requirements, as the system administrator you may have domain administrator responsibilities.

- Local administrative domains

  Within each second-level domain are local administrative domains. These are the domains you administer for your organization. A local domain can be as small as one machine, or large enough to include many machines and additional name servers. It may also have other administrative domains (called subdomains) nested within it.

For more information on name servers and domain management, see Using the Domain Name Service.

3.2.2.2 Selecting a domain name

Two concepts must be distinguished when assigning as domain name: the domain label and the domain name. You can freely select the domain label (taking certain rules into account), bit the domain name comprises the labels of the superordinate domains and the label of the current domain. All domain labels taken together form a tree structure (the name space). The name of the domain is the concatenation of all the labels of the domain from the root to the current domain, listed from right to left and separated by dots. The address of a machine named monet at Podunk University in Germany might look like this:

monet.podunk.DE

The top-level domain for Germany is DE, podunk is a subdomain of DE, and monet is the name of the machine.

Each domain label must be unique within its domain. For example, you could use podunk as a label for a domain consisting of all machines at Podunk University, provided that another entity in the domain DE has not used this name first. The label you assign a machine within your domain must be unique as well, but only within domain podunk.

Once you choose a label, you must concatenate it with the labels of all the higher level domains to create your domain name.

Domains are not limited to any fixed number of levels. For example, the Amalgamated Widgit Company might be registered with the NIC as belonging to the domain Widgit.COM; it might also have a subdomain called eng.Japan.Widgit.COM for the engineering group within their Japan subsidiary, and another called mktg.Japan.Widgit.COM for marketing in Japan. However, machines at company headquarters might belong to the subdomain called HQ.Widgit.COM.

Upper- and lower-case letters are not significant in domain names. The traditional UNIX convention is to use all lower-case; many other systems use uppercase.

At many sites, users name their individual machines while the administrators name the servers. The administrator should ensure that there are no duplicate names within a domain by using the *nslookup* program. You can use this (among other things) either to produce a listing of all the machines in the domain, or to obtain information about a machine. The program provides a command interface. If you want to know which DNS name server is responsible for your machine, enter the following (the name of the machine is balla):

```
# nslookup balla
Server:lux01.mchp.siemens.de
Address: 194.25.17.2
Name: balla
Address:138.21.4.14
```

If you want a list of all the machines in the domain, call the command without parameters. A prompt appears. Enter the *ls* command and the name of the desired domain.

```
nslookup
> ls mchp.siemens.de
```

You receive a list of all the machines in this domain.

You will find a detailed description of the *nslookup* command in the description of *nslookup(1M)* in the "Networking Reference Manual".

3.2.2.3 Registering your domain

After choosing the name of your upper domain (the equivalent of Widgit.COM in the example in the preceding section), you need to register it with a higher-level domain. To do this, you should get in touch with the administrator of the higher domain. The top-level domains are administered by the NIC at SRI International.

To register a domain with NIC, request the Domain Registration Form from NIC and complete it according to the instructions in the appendix.

Once your domain is registered, you can divide it into subdomains as the need arises.

3.3 Basic configuration

The TCP/IP software is installed at the same time as Reliant UNIX 5.45. You normally receive your computer with the software preinstalled. The files required for configuration are created and assigned default values, which you have to adapt to suit your requirements.

Before you can start using your machine in the network, you must configure and load the network interface(s). The *sysadm* system is available for this. The configuration values are stored in the */etc/default/inet* file.

3.3.1 Configuring LAN controllers

Before you connect a machine to the network for the first time after installing Reliant UNIX, you must configure the LAN controllers. To do this, enter *sysadm* and select *Configuration* from the menu. Then select the following menu items:

- *Load*
- *Boards*
- and the controller type, depending on the configuration of your system, e.g.:

lce	=	LAN Controller Ethernet (10 Mbit/s, Multibus II, RM 600)
lct	=	LAN Controller Token Ring (4/16 Mbit/s, Multibus II, RM 600)
lcf	=	LAN Controller FDDI (100 Mbit/s, Multibus II, RM 600)
mac569	=	LAN Controller Ethernet (10 Mbit/s, Systembus RM 600)
zx	=	LAN Controller Ethernet (10/100 Mbit, PCI RM 400, Systembus RM 600)
CSI	=	Central Service Interface with Ethernet interface (10 Mbit/s, Multibus II, RM600)
Motherboard	=	Ethernet interface (10 Mbit/s, RM400)
dfe	=	FDDI RM400, PCI RM600
madge	=	Token Ring RM400, PCI RM600

alt*n*	=	LAN Controller Gigabit Ethernet (1 Gbit/s, PCI)
zx	=	LAN Controller Ethernet (10/100 Mbit, PCI RM400/RM600, Systembus RM600

An input form is displayed in which you enter the Internet address for the controller and, if desired, a network mask (see Section "Selecting/assigning subnet numbers and using network masks") and a broadcast address. If you specify illegal values for the network mask and broadcast address, the machine cannot be connected to the network. The configuration becomes valid when you select *quit* and *set*.

You can also configure the LAN controller without using the menu by making all of the necessary entries directly in the configuration files, e.g. */etc/inet/hosts* and */etc/default/inet* file (see the Section "Defining configuration values for starting the network").

Please note, however, that these values are overwritten if *configure* is called at a later date.

After configuration, the machine is connected to the LAN using Slfmllan (by choosing the following menu items: *sysadm*
→*network_services*
→ *lan*
→ *connect*).

3.3.2 Defining configuration values for starting the network

The /etc/default/inet configuration file

The */etc/default/inet* file contains information which is evaluated or defined at system startup or when working with the *sysadm* menu system (Slfmllan and CONFIG).

Before you enter changes in the */etc/default/inet* file, you will need to terminate the network software with the following call:

sh /etc/rc2.d/s69inet stop

Only call this command from the system console, and ensure that there are currently no active network connections.

After making changes in the */etc/default/inet* file, you have to restart the network software with the start script provided:

sh /etc/rc2.d/S69inet start

For example, with

sh /etc/rc2.d/S69inet start | stop madge0

you can enable/disable the madge0 interface.

You can modify the */etc/default/inet* file either using a text editor, or by calling *sysadm* and selecting *Configuration*.

Using the *sysadm* operator system is a reliable way of entering the configuration parameters.

Under the menu item *Configuration* you select those interfaces that you wish to change.

There are four types of variable in this file:

- general variables
- route-specific variables
- network interface variables
- variables affecting RPC, NFS and NIS

When changes are made to the */etc/default/inet* file, certain changes do not become effective until the system is restarted. It is not enough to execute the sh /etc/rc2.d/S69inet start command. This applies to changes to the values for DEFAULTGATEWAY, TIMESYNC, GATED, GATEWAY and RWHOD.

Changes to RPC,NFS and NIS variables do not take effect until the associated scripts in */etc/rc2.d* are restarted.

3.3.2.1 General variables

NTP

The *in.xntpd* daemon for time synchronization is started automatically during system startup if the variable value is yes. The possible values for NTP are yes and no.

RWHOD

The *in.rwhod* system status server is started automatically whenever the system is booted, provided the RWHOD variable has been set to yes. The *in.rwhod* daemon provides information at regular intervals on the current state of the machines connected to the network. For example, information is displayed on whether a certain machine is active and, if so, for how long, how many users are currently working at it and how great the load on the system is. In large networks, this daemon can place a considerable load on the system. For this reason, the RWHOD variable is set to no by default. If the daemon is activated at all, it makes sense to activate it on all the machines, since only then will status information be delivered by all the machines.

STATE

Possible values of STATE are active and inactive.

If the value of STATE is active, the machine is connected to the network at system startup.

If no value is specified or if the variable does not appear in */etc/default/inet*, then active is assumed at system startup.

TIMESYNC

Possible values of TIMESYNC are master, yes and no.

If the value of this variable is yes or master, the time synchronization daemon *in.timed* is started at system startup.

If its value is master, the *in.timed* daemon is started so that the machine can act as a master for global time synchronization.

If no value is specified or if the variable does not appear in */etc/default/inet*, no is assumed at system startup. More information on time synchronization can be found in the Section "Time synchronization".

3.3.2.2 Route-specific variables

Information on routing, the *routed* and *gated* daemons and the *route* command can be found in the chapter entitled Expanding your network.

GATED

Possible values of GATED are yes and no.

If GATED is assigned the value yes and the */usr/sbin/gated* file exists, the routing daemon */usr/sbin/gated* is started and the GATEWAY variable (see below) is ignored. If GATED is assigned the value no, or the */usr/sbin/gated* file does not exist, further routing action depends on the value of the GATEWAY variable.

GATEWAY

Possible values of GATEWAY are yes, no, yes_enhanced, passive_enhanced and passive.

If GATEWAY is assigned the value yes, the *in.routed* daemon is started automatically without options when the system is started. If GATEWAY is assigned the value yes_enhanced on an "enhanced" system, the *in.erouted* daemon is started automatically without options when the system is started.

If GATEWAY is assigned the value passive, the *in.routed* daemon is started with the *-q* option, i. e. no route information is sent. If GATEWAY is assigned the value passive_enhanced on an "enhanced" system, the *in.erouted* daemon is started with the *-q* option, i. e. no route information is sent.

Depending on the application scenario, it is advisable on an "enhanced" system to specify yes_enhanced instead of yes or passive_enhanced instead of passive.

DEFAULTGATEWAY

If the DEFAULTGATEWAY variable is assigned a value, this value is set as the standard route at system startup, i.e. route add default $DEFAULTGATEWAY 1is executed.

Example:

Input:

DEFAULTGATEWAY="131.25.67.89"

At system startup, the following command is executed:

route add default 131.25.67.89 1

3.3.2.3 Network interface variables

The following variables relate to the network interfaces:

ETHERFLAGS

OLDBROADCAST

INTERFACES

The INTERFACES variable contains an entry for each interface to be configured. When the system is started up, an attempt is made to configure the interface specified in each entry. Configuration is performed using the *ifconfig* command or via the *sysadm* → *config* menu. Detailed information on this command can be found in the "Networking Reference Manual".

If an interface description in the INTERFACES variable contains no broadcast address specification, the value of the OLDBROADCAST variable is evaluated for that interface.

In the ETHERFLAGS variable, default values may be specified for the network interface parameters. These values are only used for an interface if they have not been specified in the INTERFACES variable.

If the INTERFACES variable has not been assigned a value, the interfaces to be configured are determined automatically (see INTERFACES).

ETHERFLAGS

This variable can be assigned values with the following format:
[[-]arp] [metric n] [netmask *mask*]

In this variable, default values may be specified for the network interface parameters.

If no value has been specified, or if the variable does not appear in */etc/default/inet*, arp is taken as the default value. In other words, the ARP protocol is used for Internet-to-MAC address mapping.

The value arp can, if required, be redefined by means of the interface entry in INTERFACES. The remaining parameters from *ifconfig* should be defined here only when they are not overwritten by any interface description in INTERFACES. For more information, see the description of *ifconfig* in the "Networking Reference Manual".

OLDBROADCAST

Possible values for OLDBROADCAST are yes and no.

The OLDBROADCAST variable defines the default format for the broadcast address. This value can be overridden by
broadcast:[*address*]

in the INTERFACES variable for each interface.

If no value has been specified or if the variable does not appear in */etc/default/inet*, no is used as the default value.

The way in which the broadcast address is transmitted via the interface is determined by *ifconfig* at the system startup. This is done in principle as follows:

yes: ifconfig ... oldbroadcast ...

no: ifconfig ...

The broadcast address has the following format, depending on the OLDBROADCAST variable and the network class:

no	yes	Network class
x.255.255.255	x.0.0.0	Class A
x.y.255.255	x.y.0.0	Class B
x.y.z.255	x.y.z.0	Class C

x, *y* and *z* correspond to the parts of the network number.

INTERFACES

A list of interface specifications can be specified as the value. The individual entries in the list must be separated by blanks and/or tabs.

If no network interface is specified with the STANDARDIF variable, the interface to be configured is determined automatically by means of *etherstat -e* and *uname -n*. *etherstat -e* determines the interface parameter *interface*, and *uname -n* determines the interface parameter *address* for the *ifconfig* call.

Each interface specification is in turn another parameter list. The parameter sets for each interface must be separated by colons.

If the parameter list starts with slip, the parameters that follow define a slip configuration. Any parameters listed after slip must observe the conventions specified in the section entitled SLIP - TCP/IP connections via serial lines.

A slip specification must have the following format:
slip:*tty*:[*baudrate*]:[*mtu*]:[*ip-source*]:[*ip-destin*] [:*options*]

E.g. EXTIF=slip:/dev/tty00:38400:578:orion:139.22.228.100

Default values:

baudrate 9600
mtu 1006
ip-source *machinename*

You define *machinename* using *uname -n*. If the parameter list does not start with slip, the parameters must be specified as for *ifconfig*.

The parameters that can be used are the parameters from *ifconfig*.

If there is only one entry in the parameter list, it must specify the interface (*interface* parameter). The address (*address* parameter) is determined automatically using *uname -n*.

If the parameter list contains more than one entry, the first two entries are assumed to be *interface* and *address*.

There is an entry in the */etc/default/inet* file for each interface which is one line in length and looks something like this:
EXTIF=lce1:43.22.4.2:netmask:0xff000000:broadcast \ :43.255.255.255

parameter entries (e.g. arp) in the parameter list overwrite corresponding entries in ETHERFLAGS.

If more than one interface is to be configured, the interface parameters must always include values for the interface parameters *interface* and *address*.

Network interface example

```
OLDBROADCAST=yes
STANDARDIF=lce0
EXTIF=lce1:193.51.47.11:netmask:0xffffff00:broadcast \ :193.51.47.255
INTERFACES="$STANDARDIF $EXTIF"
```

The interfaces lce0 and lce1 are configured as follows:

lce0:

"address":	Machine name □ (determined automatically)
"parameter":	
oldbroadcast	Because OLDBROADCAST=yes

lce1:

"address":	193.51.47.11
"parameter":	
broadcast 193.51.47.255	Overwrites the default specified by OLDBROADCAST=yes

In the case of a machine with the name danzig the following commands are executed at system startup:

```
ifconfig lce0 danzig oldbroadcast up
ifconfig lce1 193.51.47.11 netmask 0xffffff00 \
broadcast 193.51.47.255 up
```

3.3.2.4 Variables affecting RPC

PORTMAP_ONLY

The value of the PORTMAP_ONLY variable specifies whether RPC client programs use *rpcbind* or the older "Portmapper" protocol for determining the server addresses. With PORTMAP_ONLY, you can resolve problems arising from the fact that some RPC servers do not answer calls via the *rpcbind* protocol (e.g. SunOS 4.1.4).

The following values are possible for PORTMAP_ONLY:

yes RPC clients call the "Portmapper" service (RPC program 100000, Version 2) of the server instead of *rpcbind* (RPC program 100000, Version 3). Problems with RPC servers which do not answer Version 3 calls (e.g. SunOS 4.1.4) therefore no longer arise. The value yes has no effect on the local *rpcbind*. It supports both Version 2 and Version 3.

no RPC clients call *rpcbind* first. If this call is not successful, the "Portmapper" service is called.

3.3.2.5 Variables affecting NIS

Information on NIS and on the global user names discussed in the following can be found in the chapter entitled The Network Information Service (NIS).

DOMAIN

The value specifies the name of the NIS domain which is to be valid at a system startup.

GLOBALPW

This variable determines whether and, if so, how the entries in the NIS maps for global user names are transferred to the local files */etc/passwd*, */etc/shadow* and */etc/group*. The variable GLOBALPW may take the values new, auto, yes and no.

Periodic updating of the local files */etc/passwd*, */etc/shadow* and */etc/group* through the corresponding NIS files occurs only when the variable is set to new, yes or auto. Backup files with the names */etc/opassd*, */etc/oshadow* and */etc/ogroup* are created during the update.

If the environment variable GLOBALPW is set to new, the client will take */etc/passwd.local* and */etc/group.local* into account when it updates its local files. If the environment variable GLOBALPW is set to auto, the client will take */var/yp/pwpattern* into account when it updates its local files. If the GLOBALPW variable is set to yes, the client can only carry out a limited update of its local files (see Section "The environment variable GLOBALPW").

YPMIXER_CHECKDIR

YPMIXER_CHECKDIR determines how an NIS client handles the entry for the home directory of a global user. Possible values of YPMIXER_CHECKDIR are nocheck, create and default.

YPMIXER_CHECKDIR is effective only if the GLOBALPW variable has the value new or auto.

nocheck — The path name for the home directory of a global user is copied to the local */etc/passwd* entry without checking whether the directory exists on the client.

create — If the home directory of a global user does not exist on the client, it is generated and the contents of */etc/skel* are copied there.

default — If the home directory of a global user does not exist on the client, the local entry in the */etc/passwd* file contains */tmp* as the home directory.

YP_MODE

The variable YP_MODE may take the values client, client_auto, server, master and inactive.

This variable shows the NIS status of the machine.

AUTO_BINDING

Possible values of AUTO_BINDING are yes and no.

If the value of this variable is yes, the file */var/yp/binding/domainname/ypservers* of an NIS client machine is automatically updated to match the list of current NIS servers. In addition, a "+" entry is added to the file.

3.3.2.6 Variables affecting NFS

The value of MASTERMAP determines whether or not the NFS automounter is automatically started, and with which master assignment file. Possible values are path names or nomap. If the file specified by the path name does not exist, or the specified file is not a regular file, or MASTERMAP is not defined, the value nomap is used.

file — Path name of an NFS automounter master assignment file. If file exists, the automounter is started by the NFS startup script. A path name often used is */etc/auto.master*.

nomap — The NFS automounter is not started automatically.

3.3.2.7 Configuring several interfaces in the same network

It is possible under Reliant UNIX 5.45 and using the "enhanced" protocol stack to configure more than one network interface in the same IP network. This gives rise to certain peculiarities that affect the IP routing. These are described in the Chapter "Expanding your network" under the keyword Selective Net Routes.

3.3.3 Use of IP-Multicasting

IP-Multicasting is used to combine any number of machines in the network to form a logical group. These machines are then accessed under the IP-Multicast address assigned to their group. The members of a group do not need to know which other machines are in their group.

IP-Multicasting therefore takes up a position somewhere between unicasting (which involves controlled access to one specific partner) and broadcasting (access to all machines in the network).

IP-Multicasting is a software property, and should not be confused with multicasting on the MAC level, which is a network interface property.

IP-Multicast addresses are class D addresses; see the Section "Internet addresses".

You will find a detailed description of IP-Multicasting in RFC 1112 "Host Extensions for IP-Multicasting", which also explains the IGMP (Internet Group Management Protocol).

Reliant UNIX 5.43B or later supports IP-Multicasting and the IGMP protocol.

A sample application is the gated routing daemon in Version 3.5, which is delivered as of Reliant UNIX 5.43B. Specific IP-Multicast groups are used for communication with other routing daemons in the network (depending on the configured routing protocol): for example, address 224.0.0.2 is defined as the "all-routers-group", and is therefore used to access all routers in the system.

An application can use IP-Multicasting by joining certain predefined or user-selected multicast groups. It does this by making its membership of the groups known to the system. The application can also explicitly leave a group; it implicitly leaves all multicast groups of which it was a member when it is terminated.

In order to implement IP-Multicasting efficiently, the hardware characteristics of the network interfaces are utilized. When an application joins a multicast group, an additional MAC multicast address is implicitly registered on the interface. This address remains active until the last application on the local system has left this IP multicast group. The number of additional MAC addresses possible depends on the hardware.

IP-Multicasting can be used only via UDP, ICMP and RAW IP; it makes no sense to use IP-Multicasting via TCP because of the point-to-point character of TCP connections.

At present, no multicast routing functionality is (yet) available.

3.3.3.1 Configuring and deconfiguring network interfaces

IP-Multicasting is not supported for all network interfaces. For example, it is supported for Ethernet and FDDI, but not for Token Ring.

In contrast with system versions previous to Reliant UNIX 5.45, IP-Multicasting need no longer be explicitly activated with *ifconfig(1M)*. During system startup, all interfaces that support IP-Multicasting are automatically configured and implicitly join the "all-hosts group" with the address 224.0.0.1.

ifconfig(1M) can be used to check whether a particular interface supports IP-Multicasting.

3.3.3.2 Default route for multicast packets

A packet cannot be sent to a multicast group unless you have specified the interface via which the packet is to be sent. You can either do this explicitly in the program (see *ip(7)* in the reference manual) or install a default route entry for multicast packets. This is described in the Section "Routing with subnets".

To send packets to a multicast group your application does not need to join the multicast group on the local system. Membership of the multicast group affects only the receipt of packets intended for the group and the delivery of the packets to the application.

3.3.4 Selecting between Ethernet based on RFC 894 and RFC 1042

TCP/IP supports different frame formats depending on the underlying network architecture (Ethernet, Token Ring, FDDI, etc.).

In the case of Ethernet local area networks there are two frame formats: the so-called Xerox Ethernet format (RFC 894) and the format based on the IEEE 802.3 standard (RFC 1042).

Reliant UNIX systems are configured by default for the widely used Xerox Ethernet format, but the frame format based on the IEEE 802.3 standard can be used optionally. It is not possible to use both frame formats over a single Ethernet interface.

To run TCP/IP based on the IEEE 802.3 standard, the Ethernet interfaces must be reconfigured using the *ifconfig* command:

ifconfig *Xerox-interface IP-address* down #disable Ethernet
ifconfig *802.3-interface IP-address* up #enable 802.3

The entry of the interface for the Xerox Ethernet consists of the interface type and the interface number (e.g. lce0). For the IEEE 802.3 standard, an S is inserted after the interface type:

ifconfig lce0 *IP-address* down
ifconfig lceS0 *IP-address* up

It is not possible to effect an automatic change to the 802.3 format using the menu system or by making an entry in the */etc/default/inet* file.

The 802.3 protocol may only be used for the Gigabit Ethernet for operation with standard frames of 1500 bytes. See also the section Gigabit Ethernet in the Chapter "Transmission media".

3.3.5 Transport system selection

The function for selecting the transport system provides a simple and consistent interface that allows applications to select one or more suitable transport systems from those that are available. This makes the applications protocol- and media-independent.

3.3.5.1 Overview of transport system selection

The function for selecting the transport system is built around the */etc/netconfig* configuration file, which contains an entry for each transport system available to the system. The NETPATH environment variable is also available as an option. This contains an ordered list of transport system identifiers. These transport system IDs correspond to the *network_id* field in */etc/netconfig* and are used as links to the records in the */etc/netconfig* file.

The application programming interface for transport system selection consists of a set of access routines for the */etc/netconfig* file. One group of these library routines accesses only the entries in the */etc/netconfig* file that are identified by the NETPATH environment variable. Another group of routines enable direct access to the */etc/netconfig* file. The routines are described in the "Network Programming Interfaces" manual.

Applications should use the routines that access the NETPATH environment variable. They allow users to influence the selection of the transport system. If an application is supposed to prevent the user from influencing its decision, then the routines that access the */etc/netconfig* file directly should be used.

The */etc/netconfig* file is maintained by the system administrator. The NETPATH environment variable is typically set or modified by application programmers and users to suit the requirements of the relevant application program, but it may also be set by the system administrator in response to the needs of particular application programs.

3.3.5.2 The /etc/netconfig file

The */etc/netconfig* file contains an entry for each transport system available on the machine. This entry comprises the following fields:

network_id semantics flags pro_family pro_name \ device nametoaddr_libs

The individual entries are separated by end-of-line characters, and the fields are separated by spaces or tabs. A backslash is represented by \\. This section gives a brief description of each field. You will find a more detailed description of the */etc/netconfig(4)* file is in the "Networking Reference Manual".

network_id
: This field contains a character string that identifies the transport system. A maximum length is not defined. Each *network_id* must be unique within a system.

semantics
: This field contains a character string that describes the transport system. It can contain the following values:

tpi_clts
: Connectionless transport system

tpi_cots
: Connection-oriented transport system

tpi_cots_ord
: Connection-oriented transport system; supports orderly disconnection.

flag
: This field contains a string that specifies transport system attributes. It consists of a combination of individual characters. Each character stands for the value of the attribute that it represents. If the character is present, the associated attribute is set. If the character is not present, the attribute is not set. A dash means that no attributes are set. At present, there are two known attributes:

v Stands for "visible" and is used when the NETPATH environment variable is not set (see Section "The NETPATH environment variable").

b Indicates that broadcasts can be sent via the transport system.

pro_family
: The string contained in the *pro_family* field identifies a protocol family (e.g. Internet or ISO). A maximum length is not defined.

A dash means that none of the available identifiers applies for the protocol family. An application that requires certain characteristics of protocol families can check the entries for selecting a network (e.g. an application can search for an OSI family). In this case, the application is dependent on the protocol, since it only searched for OSI entries. The description of the */etc/netconfig* file provides examples of protocol family identifiers.

pro_name
: The *pro_name* field is reserved for protocol-specific applications. It contains a character string that identifies a protocol. At present, this field can only be used for the Internet protocol family. The field contains a dash for the protocol name of all the other families. A maximum length is not defined. The *pro_name* field can contain the following:

icmp Internet Control Message Protocol (ICMP)

tcp Transmission Control Protocol (TCP)

udp User Datagram Protocol (UDP)

device The *device* field contains the full pathname of the device file that is used to set up the connection to the transport system. This device file is generally located in the */dev* directory. The field must contain an entry.

nametoaddr_libs
: The dynamic libraries specified here contain the transport-specific routines for name-to-address

conversion. A dash in this field indicates that no libraries are available and that therefore no name-to-address conversion takes place. The field comprises a list of full pathnames separated by commas. Within a pathname, commas are represented by \, and a backslash is represented by \\.

The system administrator defines the order of the entries in the */etc/netconfig* file. The library routines for the transport system selection function, which access the file directly, return the entries in a specific order, starting at the beginning of the file. This means that the order in which the system administrator enters these transport systems will automatically be used as the default search path for user programs which select transport systems in order to set up a connection.

Example for the /etc/netconfig file

In the example below, the last two columns of the table are located under columns 1 - 5 for reasons of space.

```
# The Network Configuration File.
#
# Each entry is of the form:
#
# net_id    semantics     flags   pro_family   pro_name  ...
#
ticlts      tpi_clts      v       loopback     -
ticots      tpi_cots      v       loopback     -
ticotsord   tpi_cots_ord  v       loopback     -
#
udp□□□□□□□□□□□□tpi_clts□□□□□□□vb□□inet□□□□□□□□□udp
tcp□□□□□□□□□□□□tpi_cots_ord□□v□□□inet□□□□□□□□□tcp
icmp□□□□□□□□□□□tpi_raw□□□□□□□□-□□□inet□□□□□□□□□icmp
rawip□□□□□□□□□□tpi_raw□□□□□□□□-□□□inet□□□□□□□□□-
... device□□□□□□□□□□□□□□nametoaddr_libs
□□□□/dev/ticlts□□□□□□□□/usr/lib/straddr.so
□□□□/dev/ticots□□□□□□□□/usr/lib/straddr.so
□□□□/dev/ticotsord□□□□/usr/lib/straddr.so
□□□□/dev/udp□□□□□□□□□□□□/usr/lib/tcpip.so,/usr/1
□□□□/dev/tcp□□□□□□□□□□□□/usr/lib/tcpip.so,/usr/1
□□□□/dev/icmp□□□□□□□□□□/usr/lib/tcpip.so
□□□□/dev/rawip□□□□□□□□□/usr/lib/tcpip.so
```

3.3.5.3 The NETPATH environment variable

In most cases the user isn't interested in the transport system that handles a network operation. The default search path set up by the system administrator (the */etc/netconfig* file) is therefore used to determine a transport system for the establishment of a network connection. However, if the user or the system administrator wants to influence the choices made by applications, the search path can be modified using the NETPATH environment variable.

NETPATH consists of a colon-separated list of transport system names. Each transport system name corresponds to an entry in the *network_id* field in the */etc/netconfig* file. A colon within the field is represented by \:, and a backslash by \\. An empty component in NETPATH, signified by either a beginning colon, an ending colon, or two successive colons, is not a valid entry, since the empty string is not a valid transport name.

The NETPATH environment variable is not set in */etc/profile*. It can, however, be set in a user's *$HOME/.profile*. NETPATH is described under *environ(5)* in the "Programming Reference Manual".

Users and system administrators alike should be aware that the set of "default" transport systems is different for the routines that access */etc/netconfig* directly and the routines that access */etc/netconfig* via the NETPATH environment variable. For the routines that access */etc/netconfig* directly, the set of default networks is the entire */etc/netconfig* file; the set of "default" networks for the routines that access */etc/netconfig* via NETPATH is the "visible" transport systems in the */etc/netconfig* file. A transport system is "visible" if the system administrator has set the value v (visible) in the *flags* field. If NETPATH is not set, these "visible" transport systems will be the default search path for this second group of access routines.

3.3.6 Selecting the "enhanced" or the "fallback" protocol stack

In Reliant UNIX Versions up to Version 5.45 the TCP/IP protocol stack was available in two variants; one based on 4.3BSD and the other based on Streams. The Streams implementation was originally a component of UNIX SVR4 (System V Release 4). For this reason there were many TCP/IP modules in both variants. The BSD stack mainly offered, in comparison with the Streams stack, better performance for the socket interface and in some particular cases offered better adaptation to the original semantics of the sockets interface, which

was actually designed for a non-Streams environment. The Streams stack supported, from the user's viewpoint, not only the sockets interface, but also TLI and XTI interfaces as well as TPI interfaces (Transport Provider Interface) in the kernel.

In Reliant UNIX V5.45 the TCP/IP protocol stack has been redesigned and standardized. TCP/IP modules are based on 4.4BSD and these modules occur only once in the system. The performance of the socket interface is at least as good as the BSD stack in the previous versions. The Streams interfaces for TPI, TLI and XTI are made available at the beginning of the TCP/IP layer by means of adaptation modules.

The standardized stack in Reliant UNIX V5.45 is referred to here as the "enhanced" protocol stack. The "enhanced" protocol stack was carefully designed in order to ensure compatibility with existing applications.

In special cases where the precise behavior before Reliant UNIX Version 5.45 is required, there is an option for switching the system back to the old two-way implementation. This is referred to here as the "fallback" stack solution.

The "fallback" stack solution can easily be restored on any system with an "enhanced" protocol stack by

- modifying the value of the NETWORK_LINK_DEFAULT variable in the file */etc/conf/cf.d/net.default*,
- regenerating the kernel and
- restarting the system.

The same mechanism can be used to switch back to the "enhanced" stack (modification of the NETWORK_LINK_DEFAULT variable, calling the *idbuild* and restarting the system).

With this procedure, the files in the directories */etc/conf/sdevice.d* and */etc/conf/sfsys.d* remain unchanged in the event of the regeneration of a certain type of system kernel. Instead, the *idnetconf* command called by *idbuild* when generating the files *sdevice* and *sfsys* in the */etc/conf/cf.d* directory acts as a filter for specifying the modules required for a new operating system kernel which is being generated. When *idbuild* is running you receive a message, similar to the one below, indicating the type of kernel being generated:

Network type for this system build is enhanced
□□^^^^^^^^

When the network is started, various actions are performed. These actions depend on the type of system kernel that was started. The selection is made automatically using the */etc/rc2.d/S69inet* file. The new functions of the *slink* command are used here, as they permit flexible use of the */etc/inet/strcf* file in the network start script. This means that specific execution instructions can be skipped when *slink* is processing them, depending on the type of system kernel that is currently active.

For more information on *slink*, refer to the "Networking Reference Manual" or the man slink call.

3.4 TCP/IP configuration files

The files described in this section contain the configuration data required for a TCP/IP network. These files are created by Reliant UNIX 5.45 during installation and must be modified to suit your network.

If you want to modify your configuration, you will need to make changes to these files. If you are using the NIS network information system, you will only need to change these files on the NIS master server. The relevant information is then forwarded to the other machines in the network. If you are working with the *sysadm* system, you can use it to modify your configuration.

/etc/inet/hosts
: The *hosts* file maps machine names to Internet addresses. Every machine in the network must have a *hosts* file.

/etc/inet/networks
: The *networks* file maps network names to network numbers. Every machine in the network must have a *networks* file.

/etc/inet/protocols
: The *protocols* file lists the Internet protocols installed on your system and their numbers; it is created automatically when TCP/IP is installed and requires no modification on the part of the system administrator.

/etc/inet/services
: The *services* file lists the names of existing Internet services and their port numbers; it is read by programs that call network services. *services* is created automatically when TCP/IP is installed and only requires modification on the part of the system administrator if and when new services are installed on the machine.

/etc/inet/timed
: The *timed* file lists the names of the machines that can be considered as a master or slave for network-wide time synchronization by means of the *in.timed* daemon. Slaves are machines whose timestamp is used in addition to that of the master to calculate the time valid throughout the network.

/etc/inet/rc.sysctl.local
: Tuning parameters for the *sysctl* interface are set in the *rc.sysctl.local* file (see the Section "Die sysctl interface").

3.4.1 The hosts file

A machine name in */etc/hosts* may consist of up to 24 characters and the characters A-Z, a-z, 0-9, hyphen (-) and period (.) are permitted. The first character must be a letter, the last character must not be a hyphen (-) or period (.). A period is only permitted if it is used to separate the components of a domain name (RFC 921). No blanks are permitted and no distinction is made between uppercase and lowercase. Although Reliant UNIX actually supports machine names that are up to 63 characters in length (without domain components), a restriction to 24 characters ensures maximum interoperability with the Internet. The same applies to names for networks, gateways and to domain name components separated by a period.

The file contains the assignment of Internet addresses to the corresponding machine names, in the following format:

Internet address machine_name alias...#comment

where *Internet address* is the combination of network number and machine number assigned to a machine; *machine_name* is the official name of the machine; *alias* ... is a list of one or more alternative names that can be used to transfer messages or files to the machine over the network; and #*comment* is any kind of note you want to append to an entry.

The following line must always be in */etc/init/hosts:*

127.0.0.1 localhost

The file contains this entry when the system is shipped. The local address is 127.0.0.1 on every machine; the address is used by programs to reach services on the same machine from which they are invoked.

The entries in */etc/inet/hosts* ensure that applications that use the machine name to establish a connection are supplied the associated Internet addresses. The applications obtain this information either directly from the local */etc/inet/hosts* file or via the Domain Name System (DNS). You only have to make entries in the */etc/inet/hosts* file if you do not use either the NIS or DNS.

If this is the case, each remote machine that is to be accessible to the local machine must be entered in the */etc/inet/hosts* file. If you use the NIS network information service, these entries need only be made on the NIS master server. The information is then distributed to the other machines.

Example:

Suppose your system is the NIS master server, its name is dancer and its Internet address is 192.9.200.1. Next, assume you need to add three new systems to your network, namely: samba, ballet and raks. To update the hosts file, you would add the following (boldface) lines to dancer's database

192.9.200.1□□□□□□dancer
192.9.200.2□□□□□ballet # This is a comment
192.9.200.3□□□□□samba
192.9.200.4□□□□□raks
127.0.0.1□□□□□□□□□localhost

Finally, you must make sure that the changes are made known to all the machines in the network (see Distributing the maps).

3.4.2 The networks file

The*/etc/inet/networks* file contains the assignment of network names to the corresponding network numbers of the TCP/IP networks accessible to your system. This file should contain the names and numbers of all networks accessed by network name. Networks having a different network number to that of your machine need to be linked to your network via a router. You need only make entries in this file for each machine if you are not working with the NIS network information system. If NIS is used, the file on the NIS master server is updated and distributed to the other machines.

Entries in the file have the following format:

network_name network_number alias ... #comments

where *network_name* is the official name by which the network is known; *network_number* is the number assigned by the NIC; *alias* ... is a list of one or more alternative names by which the network is known; and #*comment* is any kind of note you want to append to an entry in the file.

Once the *networks* file has been created, it must be updated when the network is expanded:

It is particularly important that you maintain the *networks* file, since the *netstat* program dealt with in the section entitled Error analysis and diagnostics uses the information in */etc/networks* to produce status tables.

Example

Here is a sample */etc/inet/networks* file:

```
#
# Internet networks
#
arpanet         10              arpa
ucb-ether       46              ucbether
#
# local networks
loopback        127
eng             193.9.0         #engineering
acc             193.9.1         #accounting
prog            193.9.2         #programming
```


Networks that are to be accessible to your machine by means of their network names must not only be

entered in the *networks* file; they must also be given an entry in the routing table.

This also applies to the network to which the local machine is connected. For more information here, please consult the chapter entitled Expanding your network.

3.4.3 Configuration files for protocols and services

In addition to the files you need to create and maintain, TCP/IP installs two files, *protocols* and *services*, that are used by various programs. The contents of *protocols* are not modified, the contents of *services* only when new network services are installed on the machine. You may, however, want to display them for information purposes.

3.4.3.1 The protocols file

/etc/inet/protocols contains the names and protocol numbers of the TCP/IP protocols installed on your system. Below is an example of this file:

```
#
# Internet (IP) protocols
#
ip     0     IP    # internet protocol, pseudo protocol number
icmp   1     ICMP  # internet control message protocol
tcp    6     TCP   # transmission control protocol
udp    17    UDP   # user datagram protocol
```

3.4.3.2 The services file

/etc/inet/services contains entries for reserved TCP/IP network services and the port numbers reserved for them. Here is an excerpt from a typical */etc/inet/services* file:

```
#
# Services
#
echo        7/udp
echo        7/tcp
discard     9/udp       sink null
discard     9/tcp       sink null
systat      11/tcp
daytime     13/udp
daytime     13/tcp
netstat     15/tcp
ftp-data    20/tcp
ftp         21/tcp
telnet      23/tcp
time        37/tcp      timserver
time        37/udp      timserver
name        42/udp      nameserver
whois       43/tcp      nicname# usually to sri-nic
```

Notice that various network services like *ftp* and *telnet* are listed in the first column. The second column contains the service's port number and the transport layer protocol (either TCP or UDP) that handles it. Some of the reserved services may not be available on your machines; however, their port numbers must be reserved nevertheless.

3.4.3.3 The inetd.conf file

To avoid having an excessive number of active processes running on a permanent basis, the *inetd* daemon is started at system startup. This process is responsible for starting the network services. It finds out which network services to start from the */etc/inetd.conf* file. If *inetd* receives from a client a connection request for one of the services for which it is responsible, it calls daemons required to establish the connection. For example, the *rlogind* daemon is run by *inetd* only when there is a request from another machine, and it is terminated at

the end of the session.

You will find more information on *inetd* and *inetd.conf* in the "Networking Reference Manual".

3.4.3.4 The socklib.conf file

The introduction of the "enhanced" protocol stack has improved the configuration options of the socket interface. This has necessitated changes to the *libsocket* and *libxnet* libraries in order to adapt them to the improved configuration options. These libraries support access to this interface.

The *socklib.conf* configuration file is used to determine the type of socket (BSD or STREAMS) that will be generated for a call to the library function *socket()*. This file is ignored in a "fallback" system.

The */etc/inet* directory contains three preconfigured files. The configuration file actually used - */etc/inet/socklib.conf* - should be a copy of one of these files:

socklib.conf.default
: This file contains a standard configuration for *libsocket/libxnet*. It results in the use of BSD sockets for the AF_INET family and the types SOCK_STREAMS and SOCK_DGRAM.

socklib.conf.bsd
: This file contains a sample configuration for *libsocket/libxnet* in order to ensure that BSD sockets are always used for the AF_INET family.

socklib.conf.stream
: This file contains a sample configuration for *libsocket/libxnet* in order to ensure that STREAMS sockets are always used for all families.

When the system is delivered, the */etc/inet/socklib.conf* file has the same contents as the */etc/inet/socklib.conf.default* file.

For further information on the "enhanced" or "fallback" protocol stack, see the Section "Selecting the "enhanced" or the "fallback" protocol stack".

More detailed information on BSD and STREAMS sockets can be found in the manual "Network Programming Interfaces". The *socklib.conf* file is described in the "Networking Reference Manual".

3.4.3.5 The /etc/conf/cf.d/net.conf file

This file specifies which modules are to be used to generate an "enhanced" or a "fallback" system kernel.


This file must not be modified.

For further information on the "enhanced" or "fallback" solution, see the Section "Selecting the "enhanced" or the "fallback" protocol stack".

3.4.3.6 The /etc/conf/cf.d/net.default file

This file provides a default value for the environment variable NETWORK_TYPE. This value determines whether a system kernel is generated for the "enhanced" or the "fallback" solution. Please note that the default value is overwritten if the NETWORK_TYPE environment variable is set for a call to *idbuild*.

For further information on the "enhanced" or "fallback" solution, see the Section "Selecting the "enhanced" or the "fallback" protocol stack".

3.4.4 The /etc/inet/timed file

The *timed* file lists the names of the machines considered to be permissible masters or reliable slaves for time synchronization in the network. The *in.timed* process of a machine not defined as a permissible master cannot itself assume the master role in the network.

Timestamps of machines not registered as reliable slaves with the *in.timed* master are not included in the calculation of the network time. The network time is an average value calculated by the master on the basis of its own timestamp and those of the slaves. An entry in */etc/inet/timed* has the following format:

machine_name keyword #comment

The possible keywords are slave and master:

slave
: The specified machine is a reliable slave.

master
: The specified machine is a permissible master and a reliable slave.

If a machine is entered in */etc/inet/timed* without a keyword, the keyword master is assumed for this machine. Entries with incorrect keywords are ignored; in this case, the specified machine is neither a permissible master nor a reliable slave. Comments are optional and preceded by a number sign (#).

If you enter the wildcard character * as the name of the machine, the entry will apply to all machines that are not listed explicitly. If there is no wildcard entry, any machines that are not explicitly listed are neither permissible masters nor reliable slaves.

If there are multiple entries for the same machine, only the first entry applies.

Special case:

If the */etc/inet/timed* file does not exist or contains no entries, *in.timed* behaves as follows: all the other machines in the network are considered to be permissible masters/reliable slaves.

Example

```
saturn  master  # First master server
orion   master  # Second master server
*       slave   # All others are reliable slaves
```

3.5 Access options and access permissions

Working in a communication network such as a LAN offers the user many new opportunities and benefits, such as

- sharing data resources
- sharing programs and devices
- global mail delivery

On the other hand, sharing a network among a large number of machines and an even larger number of users can also cause problems. It is important to ensure that each user can only access the data and programs that he or she is authorized to access.

The Reliant UNIX operating system provides a number of options for this purpose. If you take advantage of these options, you can achieve optimum protection for your data.

Access permissions for access via network commands are defined by entries in files.

There are three kinds of access in the network:

- accessing a remote machine
- accessing a user name at the remote machine
- accessing files on the remote machine.

Access permissions are defined in two files:

1. The file *$HOME/.rhosts* located in the home directory of the user granting access permission.

   In this file, you may enter the user names on remote machines that you wish to allow access to your user name.

2. The file */etc/host.equiv* on the machines granting access permission.

   In this file, each system administrator may enter the names of remote machines whose users may access the user name on the local machine that is identical to his/her own.

You will find a description of how access permissions are set below. The files *.rhosts* and */etc/hosts.equiv* are described in detail in the "Networking Reference Manual".

3.5.1 Accessing remote machines in the network

When TCP/IP is installed, you must assign a machine name that is unique throughout the network and an Internet address to each machine connected to the network. The machine names and the Internet addresses of the machines in the network should be entered in the */etc/inet/hosts* file on the respective local machine and in the *hosts* NIS map on the NIS master server.

Accessing a user name on a remote machine

There are four ways of accessing a user name on a remote machine:

- opening a session on the remote machine with *rlogin*
- issuing commands under a user name on the remote machine with *rsh*
- copying files or directories from or to a remote machine with *rcp*
- opening a session on the remote machine using the FTP (or TFTP) file transfer program
- opening a session on the remote machine via the *telnet* program

Opening a session on the remote machine with rlogin

To start a session on a remote machine you can use the *rlogin* command or the corresponding function in the menu system (*Log on to remote computer*). You will then be working at the remote machine under a user name, just as you would at your own local machine. You have the same privileges. To do this, you need a user name known to the remote machine.

The system administrator of the remote machine can provide you with a user name. If there is no entry in the file */etc/host.equiv* or *$HOME/.rhosts* authorizing you to work at the remote machine, you will also need to know the corresponding password.

The following are four different cases to be considered when you log on to remote machine using *rlogin*:

- Login under your user name

  If your machine is entered in the */etc/hosts.equiv* file on the remote machine, you do not need to specify a password with the *rlogin* command.

  If your machine is not entered in the */etc/hosts.equiv* file on the remote machine but is listed together with your user name in the *.rhosts* file of your home directory on the remote machine, you do not need to specify a password.
- Login using a different user name

  You enter the other user name explicitly with the *-l* option of the *rlogin* command.

  If your machine and user name are entered in the *.rhosts* file of the home directory of the user on the remote machine, you do not need to specify a password with *rlogin*.

Issuing commands under a user name on the remote machine

You can issue the following commands under a user name on the remote machine without opening a session there:

rcp copy a file or a directory from or to a remote machine

rsh issue a shell command on the remote machine

Again there are two cases to consider:

1. If you are working on the local machine under the same user name as at the remote machine, your local machine must be entered in the */etc/hosts.equiv* file on the remote machine or in the remote user's *$HOME/.rhosts* file. If the user name is the same, it does not have to be entered in *$HOME/.rhosts*.
2. If you want to issue a command on the remote machine under a different user name to your own, your machine name and user name must be entered in the *.rhosts* file of the user name on the remote machine.

   The same applies if you want to use *rcp* to access the files or directories of another user on the remote machine.

Starting a session using ftp in heterogeneous networks

ftp is a file transfer protocol. You can start an *ftp* session and transfer files to or from a remote machine. This machine does not have to be a Reliant UNIX system but it must support the FTP protocol.

If you wish to access data using *ftp*, you always need the user name and password of the user whose data you wish to access. Attempts to access user names that have no password will be rejected.

If you wish to prevent remote user names from accessing your machine via *ftp*, you can enter these user names in the file */etc/ftpusers*.

The default entry in this file is:

root

Starting a session using the telnet program

Using the *telnet* program you can start a session at a machine that uses a different operating system to Reliant UNIX but which supports the Telnet protocol. Again you need the user name and the password of the remote user whose data you wish to access. Connections to user names that have no password will be rejected.

Accessing data of a user on the remote machine

For data access protection, you have the normal options offered by the Reliant UNIX operating system. In other words, you can protect your files against reading, writing and execution by means of the Reliant UNIX command *chmod* and the shell command *umask*.

To ensure the greatest possible degree of security, access permissions should, for the most part, be defined in the file *$HOME/.rhosts*. Because it is impossible for a system administrator of a machine to know which access permissions have been defined on a different machine, he/she also does not know which access permissions are being granted when he/she enters a remote machine in his/her */etc/hosts.equiv* file. Therefore, you should only make entries in this file in special cases.

3.5.2 The hosts.equiv and .rhosts files

The *hosts.equiv* and *.rhosts* files are dealt with in detail below.

The /etc/hosts.equiv file

In the simplest case, the */etc/hosts.equive* file is a list of machines, or hosts, from which users are permitted to log in (using *rlogin*) to your system using the same user name and without supplying a password. *hosts.equiv* is a local file, pertaining only to the system in which it is found. The machine's system administrator can modify *hosts.equiv* by logging in under root and using a text editor to make changes.

A typical *hosts.equiv* file has the following structure:

machine1
machine2

If a user attempting to log in from a remote machine listed in */etc/hosts.equiv* has an entry in the password database, he or she will be granted access without having to enter a password. If the same user attempts to log in from a machine that is not listed in *hosts.equiv* or in *$HOME/.rhosts*, he or she will be granted access only after entering a correct password.

A single + on a line in a machine's *hosts.equiv* file means that all known machines have access.

Example:

Suppose the *hosts.equiv* file on your machine looks like this (note that the file is just a list of machine names, one per line).

raks
dancers
jazz

Now you want to allow anyone on the machine called ballet to have access to your machine. Edit */etc/hosts.equiv* and add ballet to the list, as follows:

raks
dancers
jazz
ballet

Now add all the users on ballet to your machine using the *adduser* command. After you do this, all users who can log in to ballet can also freely *rlogin* to your machine, copy files and run commands on your machine without having to supply a password.

The .rhosts file

The *.rhosts* file located in the home directory enables a user to grant or deny access to users of remote machines. If you have set up user names on your system for remote users, the *.rhosts* file must be created in all the home directories. The following may be entered in the file:

machine1
machine2 user

The above entries in a user's *.rhosts* file mean that the user with an identical user name on machine1 and the user with the user name user on machine2 have access. This means that a user can allow others to log in using his or her own user ID.

If your site does not require stringent security measures, the easiest way to administer machine access at the user level is to give each user an account in the password database and a home directory on your machine(s). Then, the entrusted users can grant the further access permissions in their own *.rhosts* files.

If the security policy on your computer will not permit the risk associated with the *.rhosts* files, you can restrict their use or entirely disable them. This is done by means of the variables DISABLE_RHOSTS and STRICT_RHOSTS_MODES in the file*/etc/default/login*.

The following values are used:

STRICT_RHOSTS_MODES=yes

.rhosts can only have read, write and execute permissions for the owner. It must, however have at least read permission for the owner.

DISABLE_RHOSTS=yes

.rhosts is not evaluated. In this case, it is only possible to access the remote host without entering a password if the name of the local computer is entered as a "trusted" computer in */etc/hosts.equiv* on the remote host.

Example 1:

The user with the user name chris wants to let other users log into the machine using his user name. The *.rhosts* file in the HOME directory of the user name chris might look like this:

```
ballet
samba
raks
jazz     bob
dancer   jenny
```

Thus, the users of the following user names have access to the user name chris on this system: chris from ballet, samba, and raks; bob from jazz, and jenny from dancer.

Example 2:

Suppose you want only the user of the user name chris on the machine samba to have access to the user name chris on the local system. In this case, you would proceed as follows:

1. Make sure samba is not in your */etc/hosts.equiv* file.
2. Create the file *.rhosts* in the home directory of the user name chris home directory with the following entry:

   ```
   samba
   ```

3.5.3 Security issues

The only way to achieve anything resembling security in a sensitive environment is to exclude users from your password database; once someone knows a password, he or she can access your machine.

Once entered in the password database, a user can log into your system at any time. Access is only possible from a remote machine using commands such as *rsh* and *rcp* when there is an appropriate entry in the */etc/hosts.equiv* file or the *.rhosts* file in the home directory of the user name being accessed. As well as defining which users may start certain processes on your system via the network, the contents of these files also list the users who may log into your system without entering a password.

Some TCP/IP user services are especially problematic in an environment that requires strict security. For example, the *finger* command allows a user to display users on a remote machine.

The command starts the daemon *fingerd* on the remote machine, which then reports to the user the user name and full name of everyone who is logged in to the machine. There is no authentication of the requesting user and no auditing of the requests. This also applies to the *rusers* command and the associated *rusersd* daemon. Similarly, the *rwhod* daemon (which provides information for status reports with the commands *rhosts*, *ruptime* and *rwho*) regularly passes around information about who is logged in to a machine.

By default, these daemons are not started. The system administrator can specify whether or not these daemons are to run by entering them in or deleting them from (or commenting them out of) the */etc/inetd.conf* file. You can also specify in the */etc/default/inet* file whether or not the *rwhod* daemon is to be started (see Section "Defining configuration values for starting the network").

3.6 Using the BOOTP server

The BOOTP utility is provided for workstations without a hard disk (e.g. X terminals or diskless workstations). The BOOTP server enables these workstations to load and start an operating system via the network.

3.6.1 The in.bootp or in.bootpgw boot server

The server *in.bootp* is an Internet bootstrap protocol server (BOOTP) as specified in RFC951, RFC1532 and RFC1533. *in.bootpgw* implements a simple BOOTP gateway which can convey requests and responses between client machines on a subnet and a BOOTP server (*in.bootp*) on a different subnet. Either *bootp* or *bootpgw* will forward BOOTREPLY packets, but only *in.bootpgw* will forward BOOTREQUEST packets.

Each machine on each network segment is normally configured to run either *in.bootp* or *in.bootpgw* from *inetd*. This is effected by including one of the following lines in the file */etc/inet/inetd.conf*:

bootp dgram udp wait root /usr/sbin/in.bootp in.bootp
bootp dgram udp wait root /usr/sbin/in.bootpgw \
in.bootpgw *server*

This mode of operation is referred to as *inetd* mode and causes *in.bootp* and *in.bootpgw* to be started only when a boot request arrives from a machine. If it does not receive another packet within fifteen minutes of the last one it received, it will exit to conserve system resources. The *–t* option controls this timeout.

It is also possible to run *in.bootp* or *in.bootpgw* independently of the *inetd* daemon by simply invoking it from a shell like any other regular command. Standalone mode is particularly useful when *in.bootp* is used with a large configuration database, where the startup delay might otherwise prevent timely response to client requests. (You can also start standalone mode automatically by including the command *in.bootp* in the */etc/rc2.d/S69inet* file.) Standalone mode is less useful for *in.bootpgw* which has a very short startup delay because it does not read a configuration file.

Either program automatically detects whether it was invoked by the *inetd* daemon or from a shell and automatically selects the appropriate mode. The *-s* or *–i* option may be used to force "standalone mode" or "inetd mode" respectively. Detailed information on these options can be found under *in.bootp* in the "Networking Reference Manual".

Both *in.bootp* and *in.bootpgw* operate similarly in that both wait for any packets sent to the *bootp* port, and both simply forward any BOOTREPLY packets. They differ in their handling of BOOTREQUEST packets.

When *in.bootpgw* is started, it determines the address of a BOOTP server whose name is provided as a command line parameter. When *in.bootpgw* receives a BOOTREQUEST packet, it sets the fields *gateway address* and *hop count* in the packet and forwards it to the BOOTP server at the address determined earlier. Requests are forwarded only if they indicate that the client has been waiting for at least three seconds.

When *in.bootp* is started it reads a configuration file, (normally */etc/inet/bootptab*) which initializes the internal database of known clients and client options. This internal database is reloaded from the configuration file when *in.bootp* receives a hangup signal (SIGHUP) or when it discovers that the configuration file has changed.

When *in.bootp* receives a BOOTREQUEST packet, it looks for a database entry matching the client request. If the client machine is known, *in.bootp* composes a BOOTREPLY packet using this database entry and sends the reply to the client machine (possibly using a gateway). If the client is unknown, the request is discarded (with a notice, if debug > 0).

The receipt of a SIGUSR1 signal creates a dump of the internal database in the */etc/inet/bootpddump* file or in a file specified in the command line.

3.6.2 BOOTP files

The */usr/sbin/in.bootp* file contains the BOOTP server. The server is normally activated via the *inetd* daemon. When the program is run, */usr/sbin/in.bootpd* reads the file */etc/inet/bootptab*, which contains the configuration table. A typical example can be found in the */etc/inet/bootptab* file. For permitted values and their meaning refer to RFC1532.

The */etc/inet/bootptab* file is read once when the BOOTP server is activated, and then each time the date and time of the last write access to the file are changed. This makes it possible to update the data it contains without the need to send a special signal to the BOOTP server.


Make sure that the text editor you use to edit */etc/inet/bootptab* writes the modified file under the same

name as before.

If your text editor saves the old file with a backup name extension, for example, and writes the modified text into a new file with the original name, the BOOTP server continues to access the old file, because although the name of the file has changed, the date of the last modification has not.

The BOOTP server writes error messages via *syslog* to the */var/adm/log/messages* file.

3.7 Dynamic Host Configuration Protocol

In a large network, the assignment and administration of IP addresses may cost the system administrator a considerable amount of work, particularly if some of the machines being administered are in the network only briefly (e.g. for hours or days) as may be the case for test machines or portable computers (Notebooks).

The Dynamic Host Configuration Protocol (DHCP) was developed to reduce this administrative workload. DHCP makes it possible for a machine in a network to obtain its configuration parameters (e.g. IP address, name server, domain name, ...) from one (or more) DHCP servers. Such servers, unlike BOOTP servers, need not possess any information about the DHCP clients prior to contact being made.

DHCP was developed by the Dynamic Host Configuration Working Group of the Internet Engineering Task Force (IETF) and is essentially based on the BOOTP protocol.

Unlike BOOTP, which was developed to use IP addresses that are configured or allocated manually, DHCP makes it possible to assign IP addresses dynamically, even to machines which are in the network for the first time: when the server is configured, the system administrator provides a pool of IP addresses, and IP addresses are subsequently assigned to clients from this pool.

It is possible to stipulate the interval for which an IP address is to be available to a client by specifying a "lease time". At the end of this time, the client must either request the server to extend the lease time for the client's IP address, or relinquish the address. The flexibility of DHCP arises from the fact that clients which shutdown or reboot can release their IP address, and the DHCP server can then assign such a released IP address to some other client.

The DHCP server for Reliant UNIX 5.45 has been implemented in accordance with the following RFCs

RFC 1533 "DHCP Options and BOOTP Vendor Extensions"

RFC 1534 "Interoperation Between DHCP and BOOTP"

RFC 1541 "Dynamic Host Configuration Protocol"

RFC 1542 "Clarifications and Extensions for the Bootstrap Protocol"

3.7.1 How DHCP works

The DHCP server supports the following DHCP message types:

Type	Description
DHCPDISCOVER	Message broadcast by the client in order to log on to servers.
DHCPOFFER	Message from a server in response to a DHCPDISCOVER message, making the client an offer of configuration parameters.
DHCPREQUEST	Message from a client to the server requesting either, the proffered configuration parameters, or, an extension of the lease time, or, that an existing configuration should be checked.
DHCPACK	Message from the server to a client, transferring the requested parameters.
DHCPNAK	Message from the server to inform a client that its

	configuration parameters are not valid.
DHCPDECLINE	Message from a client to inform the server that the IP address offered by the server has already been assigned to some other client.
DHCPRELEASE	Message from a client to inform the server that the client will no longer be using its assigned IP address.

The client sends a DHCPDISCOVER message to the broadcast address 255.255.255.255 of its local network. Any server that receives the DHCPDISCOVER message can respond with a DHCPOFFER, which offers the client an IP address and a set of configuration parameters.

The client will receive one or more DHCPOFFER messages and can select a suitable configuration from those offered. It then sends a DHCPREQUEST with the server ID of the selected DHCP server. The server so addressed responds by sending a DHCPACK containing the IP address and the configuration parameters.

The client can now continue configuring its network and does not need to contact the DHCP server again until the lease time of the IP address expires or the client wishes to relinquish its IP address.

3.7.2 Server configuration

3.7.2.1 DHCP and BOOTP

It is not possible to use the DHCP server and BOOTP server simultaneously, because they both use the same port (UDP Port 67). If a BOOTP server is configured, you will first have to deactivate it before you can use the DHCP server. In this case you should carry out the following procedure:

- Deactivate the *in.bootp* daemon in the file */etc/inet/inetd.conf* by commenting out the line

  bootp dgram udp wait root /usr/sbin/in.bootp in.bootp
- Start the *inetd* daemon afresh:

  Send the *inetd* daemon a SIGHUP signal (command: kill -1)

3.7.2.2 Starting the DHCP server automatically

The DHCP server is supplied in Reliant UNIX 5.45 C as the package *SIdhcp* and is included in the default installation. The server is controlled by means of two variables in the file */etc/default/inet*.

- If you specify

  DHCP=server

  the DHCP server will be started automatically when the system is booted.
- You can use a second variable, DHCP_INTERFACES, to list the names of those network interfaces that the DHCP server daemon *dhcpd* is to monitor. The elements of the list must be separated by spaces.

  The default setting is

  DHCP_INTERFACES=

  which sets the variable empty. In this case, the DHCP server monitors all the network interfaces.

  The following entry is equivalent to this:

  DHCP_INETRFACES=all

  The effect of this setting is to monitor all the network interfaces (apart from point-to-point interfaces).

Example:

DHCP_INTERFACES="lce1 lce2 lcf1"

This setting instructs the daemon to monitor the network interfaces lce1, lce2 and lcf1.

The DHCP server can be permanently deactivated by placing the entry

DHCP=no

in the file */etc/default/inet* and then restarting the network (see the Section "Defining configuration values for starting the network").

3.7.2.3 Starting the DHCP server manually

If you call the DHCP server daemon */usr/sbin/dhcpd* with no arguments, you can then send and/or receive DHCP packets via all the available network interfaces (apart from point-to-point).

If you call the DHCP server daemon with an argument list, for example

dhcpd lce1 lce2 lcf1

then the DHCP daemon will monitor only the specified network interfaces.

3.7.2.4 Configuration

At present, the DHCP server supports the following network interfaces:

- Ethernet
- Token Ring
- FDDI
- Point-to-point (only together with a BOOTP relay)

Since the point-to-point protocol (PPP) uses its own procedure for assigning IP addresses (the IPCP protocol), there is no point in using PPP interfaces unless, for instance, the DHCP server is connected to a network over this PPP interface by means of a BOOTP relay. If you want the DHCP server to monitor a PPP interface you must explicitly include it in the variable DHCP_INTERFACES or in the argument list for the DHCP server.

You configure the DHCP server by editing the server configuration file */etc/inet/dhcpd.conf*. The syntax of this file and the available options are described in *dhcpd.conf(5)*. You should always start the DHCP server afresh after modifying the configuration file.

There are four groups of configuration statements:

1. "Global" statements which apply to all interfaces.
2. "Subnet" statements which apply only to interfaces belonging to the specified parameters (IP address, DNS Name, network mask).
3. "Shared-network" statements, which can be used to set up a number of logical subnets over one physical network.
4. "Host" statements, which contain parameters for those clients whose client ID or MAC address agrees with the specified values.

Each of these four statements must be terminated by a semicolon (;).

A "shared-network" statement is itself made up of "global" and "subnet" statements. "Global" and "subnet" statements which are located within a "shared-network" statement are not terminated by a semicolon.

There must be a "subnet" statement containing the appropriate network address for each subnet (network interface) that is to be monitored. A minimal configuration file will therefore consist of at least one "subnet" statement.

The parameters specified in the server configuration file are dealt with as follows:

- If there is a "host" statement for a client, client-specific parameters are extracted from it. The client is identified by means of its client ID, or its MAC address, if the client does not supply an ID.
- Additional parameters are taken from a "subnet" statement. The DHCP server identifies the corresponding "subnet" statement by means of the IP address of the network interface from which the DHCP packet was received. Parameters which have already been specified in a "host" statement will not be overwritten by information from "subnet" statements.
- If the located "subnet" statement is an element of a "shared-network" statement, further parameters will be taken from the global statements for the corresponding "shared-network". Parameters from "host" statements or "subnet" statements will not be overwritten.
- The remaining parameters are taken from the "global" statements which apply to all network interfaces.

Parameters which have already been specified in "host", "subnet" or "shared-network" statements will not be overwritten.

Sample server configuration:

```
# --- Beginning of global statements ---
# A server which serves more than one network interface
# must have a unique name.
server-identifier toccata.fugue.com;
# Global statements which apply to all subnets.
# Each statement must be terminated by a semicolon.
option domain-name "fugue.com";
option domain-name-servers toccata.fugue.com;
# --- End of global statements ---
# --- Beginning of a shared-network statement ---
# This is a group of "subnet" statements which share a
# physical network interface. DHCP does not use the name
# of the "shared-network" - it serves only to distinguish
# between a number of "shared-network" statements.
shared-network FUGUE
# Global statements which apply only to this group.
# These statements are not terminated by semicolons.
□□□□□option subnet-mask 255.255.255.224
□□□□□default-lease-time 6000 max-lease-time 7200
# One of the two "subnet" statements which share this
# physical network. This "subnet" statement is not
# terminated by a semicolon.
□□□□□subnet 204.254.239.0 netmask 255.255.255.224
□□□□□range 204.254.239.10 204.254.239.20
□□□□□option broadcast-address 204.254.239.31
□□□□□option routers prelude.fugue.com
# The second "subnet" statement. The semicolon at the end of
# this statement terminates the "shared-network" statement.
□□□□□subnet 204.254.239.32 netmask 255.255.255.224
□□□□□range dynamic-bootp 204.254.239.10 204.254.239.20
□□□□□option broadcast-address 204.254.239.31
□□□□□option routers snarg.fugue.com;
# --- End of the "shared-network statement ---
# --- Beginning of a "subnet" statement for a second
# physical network. This statement is not within a
# "shared-network" statement and must therefore be
# terminated by a semicolon.
subnet 192.5.5.0 netmask 255.255.255.224
□□□□□range 192.5.5.26 192.5.5.30
□□□□□option name-servers bb.home.vix.com, gw.home.vix.com
□□□□□option domain-name "vix.com"
□□□□□option routers 192.5.5.1
□□□□□option subnet-mask 255.255.255.224
□□□□□option broadcast-address 192.5.5.31
□□□□□default-lease-time 6000 max-lease-time 7200;
# --- End of the "subnet" statement ---
# --- Beginning of a "host" statement ---
# This is a client with a static IP address.
host fantasia hardware ethernet 08:00:07:26:c0:a5
□□□□□fixed-address fantasia.fugue.com;
```

```
# --- End of the "host" statement ---
```

3.7.3 Server characteristics

3.7.3.1 Assignment of IP addresses

There are three ways of assigning IP addresses:

1. Manual (static) assignment: the system administrator places entries containing the Ethernet address and IP address of the appropriate clients in the server configuration file. Such a client will therefore always receive the same IP address and an infinite lease time. DHCP behaves like BOOTP in such cases (example: the "host statement" at the end of the above configuration file).
2. Automatic assignment: the DHCP server assigns a permanent address with infinite lease time from a pool of IP addresses.
3. Dynamic assignment: the DHCP server assigns an address with a limited lease time from a pool of IP addresses. When the lease time has expired, the server may give this IP address to some other client.

Unlike BOOTP, which supports only manual (static) assignment, DHCP enables you to use all three types of assignment.

Examples:

A client hugo with the Ethernet address 05:03:00:34:e4:02 is to be assigned an IP address. We shall assume we are using the server configuration file given in the Section "Server configuration".

Manual assignment:

The following "host" statement is appended to the server configuration file:

```
server-identifier 120.120.120.18;
subnet 192.5.5.0 netmask 255.255.255.224
□□□option subnet-mask 255.255.255.224;
host hugo hardware ethernet 05:03:00:34:e4:02
□□□fixed-address 192.5.5.26;
```

The effect of this is that a client hugo with the corresponding Ethernet address will always be assigned the IP address 192.5.5.26. In this case, the DHCP server automatically sets an infinite lease time □ (lease-time = -1).

Automatic assignment:

The following "host" statement is appended to the server configuration file:

```
server-identifier 120.120.120.18;
subnet 192.5.5.0 netmask 255.255.255.224
□□□option subnet-mask 255.255.255.224
□□□range 192.5.5.26 192.5.5.30;
host hugo hardware ethernet 05:03:00:34:e4:02
□□□default-lease-time -1;
```

Since no IP address is specified here, the DHCP server will choose an IP address from the pool. The lease time has been set explicitly to -1 (infinite). The effect of this is to assign the client a permanent IP address.

Dynamic assignment:

This is the default. If there is no specified "host" statement for a client, it will be assigned an IP address dynamically. The lease time is generally specified in the global statements of the server configuration file. If no lease time is specified, the IP address will be assigned for a period of one day.

```
server-identifier 120.120.120.18;
subnet 192.5.5.0 netmask 255.255.255.224
□□□option subnet-mask 255.255.255.224
□□□range 192.5.5.26 192.5.5.30;
□□□default-lease-time 6000 max-lease-time 7200;
```

Wherever possible, the DHCP server tries to always give a client the same IP address. This will be possible so long as there are sufficient available IP addresses in the address pool. However if there are more clients than IP addresses in the address pool, or the number of clients exceeds 1000, the DHCP server will have to abandon this strategy.

3.7.3.2 Server facilities

In addition to the various types of assignment, the DHCP server also provides the following features.

- The server supports BOOTP clients.
- You can use the utility *bootp_to_dhcp(8)* to convert BOOTP configuration files to the format used by the DHCP server.
- The server evaluates the DHCP broadcast bit (a flag in the DHCP header). This enables a client to tell the server that a DHCP packet is to be sent as a broadcast rather than a unicast.
- The system administrator can set up the lease time. The lease time is specified in seconds.
- There are no restrictions on the lease time, in particular there is no minimum value for it.
- The system administrator can set up any of the DHCP options listed in RFC 1533.
- You can assign a number of IP address pools for different IP subnets via one physical network (see the "shared-network" statement in the Sample server configuration:).
- You can set up client groups based on the user/client IDs and class IDs supplied by clients.
- You can configure the client timers T1/T2 (rebinding time, renewal time).
- All assigned IP addresses are also recorded in the file */etc/inet/dhcpd.leases*. This protocol file is described in *dhcpd.leases(5)*.
- The server supports a number of network interfaces.
- When you make a manual assignment for different subnets, you can specify one fixed IP address for each of them. The IP address that is to be assigned is established from the IP address of the network interface from which the DHCP request was received. This means that you can operate one BOOTP client on different subnets with a fixed IP address.

  Here is an example of specifying several IP addresses for a manual assignment:

  You specify a list of IP addresses separated by commas.

  ```
  host aurora hardware ethernet 02:03:04:05:06:07
  □□□fixed-address aurora-1.fugue.com, aurora-2.fugue.com
  □□□filename "vmunix.aurora"
  □□□server-name "toccata.fugue.com";
  ```

- An optional way of setting up IP address pools. The address pool need not necessarily consist of consecutive addresses.

  Here is an example of a non-contiguous IP address pool:

  The IP address pool contains all the IP addresses in the ranges 204.254.239.10 - 204.254.239.20 and 204.254.239.40 - 204.254.239.50.

  ```
  subnet 204.254.239.0 netmask 255.255.255.224
  □□□range 204.254.239.10 204.254.239.20
  □□□range 204.254.239.40 204.254.239.50
  □□□option broadcast-address 204.254.239.31
  □□□option routers prelude.fugue.com;
  ```

  You will find details of these in the RFCs mentioned in Section "Dynamic Host Configuration Protocol".

3.7.4 Server restrictions

You cannot specify any value for the lease time of a manual assignment (see the Section "Server characteristics"). The DHCP server always selects an infinite lease time in this case.

The DHCP server will respond to all DHCP requests (dynamic assignment). It is not possible to assign IP addresses only to selected clients, for example, by specifying their Ethernet address. Exception: if there is no address pool, i.e. if dynamic assignment of IP addresses is not offered, then only those clients will be served for which there are "host" statements in the configuration file.

There is no way of interacting with a DNS server. The reason for this is that the DNS servers at present in use

make use only of static DNS entries. This means that a DHCP client (or the DHCP server) cannot carry out dynamic assignments of IP addresses and machine names.

Equally, it is not possible to interact with NIS.

The DHCP server cannot contact other DHCP servers and update the server database of another DHCP server.

The DHCP server does not check IP addresses (e.g. using the *ping* command) to establish whether an IP address has already been assigned (by a different server).

No check is made to determine whether a client relinquished its IP address at the end of its lease time. It can therefore happen under certain circumstances that one IP address is assigned more than once, but this can occur only if you operate with faulty clients.

3.7.5 Relay agents

If a DHCP server is also to manage clients which are not located in the same subnet, the DHCP packets will have to be forwarded from one subnet to the other. Since DHCP packets are generally sent as broadcasts, forwarding them requires either a router which supports DHCP, or a relay agent.

A relay agent is a daemon which forwards BOOTP requests and/or DHCP requests between the server and the clients. This makes it possible for a DHCP server to serve a BOOTP/DHCP client which is located on a different subnet without its own DHCP server. The procedure for this is as follows:

- The DHCP client sends a DHCP request as a broadcast.
- The relay agent receives the DHCP packet, enters its own IP address in the header of the DHCP packet and sends it as a unicast to the DHCP server.
- The DHCP server sends its response as a unicast to the relay agent.
- The relay agent sends the server's response as a broadcast over the network interface from which it received the original request.

The DHCP server implementation in Reliant UNIX 5.45 operates together with the BOOTP relay agent *bootpgw/in.bootpgw*. The BOOTP relay agent, and its configuration are described in *bootp(8)*. The relay agent is implemented as part of the system.

3.7.6 Client configuration

Each time the DHCP client is started, it once again reads in any existing earlier configuration parameters that are stored in the file */etc/inet/dhcpc.cache*.

The individual entries in the */etc/inet/dhcpc.cache* file are interface statements of the following form separated by semicolons:

interface
□□□starts 2 1996/10/15 03:59:19
□□□ends 2 1996/10/15 04:01:19
□□□name et0 hardware ethernet 0:0:e4:1:5:7d
□□□siaddr 139.22.228.9
□□□yiaddr 139.22.228.71;

A statement begins with the keyword interface.

The beginning and end of the period of validity is recorded in the first two lines. The next line contains the interface name as well as the hardware address. The last two lines contain the IP address as well as the last client IP address specified.

The client is configured by editing the entries in the file *etc/inet/dhcpc.conf* (the syntax and the possible options are described in *dhcpc.conf(5)*).

 An instruction must contain, at the very least, the name of interface.

In order to activate the select longest algorithm for the lce1 interface, for example, a minimal client configuration file could have the following format:

```
interface
□□□name lce1
□□□select longest;
```

The specification of configuration parameters for the DHCP client is optional. The client can also be started if the */etc/inet/dhcpc.conf* file is not available. In order to start the client successfully, only the variables in the */etc/default/inet* file (see the Section "Starting the client automatically") need to be set.

3.7.6.1 Starting the client automatically

The DHCP client is supplied as the package SIdhcp and is included in the default installation. The client is controlled by means of two variables in the */etc/default/inet* file.

If you specify

DHCP=client

the DHCP client will be started automatically when the system is booted. You can use a second variable, DHCP_INTERFACES, to list the names of those network interfaces that the DHCP client daemon *dhcpcd* is to configure.

The default setting is

DHCP_INTERFACES=

which sets the variable empty. A list of interface names must always be transferred. Wildcards are not permitted.

Example:

DHCP_INTERFACES="lce1 lce2 lcf1"

In this case, the interfaces lce1, lce2 and lcf1 are configured by the client daemon while the network is restarting (see the Section "Defining configuration values for starting the network"). This may take several minutes.

The DHCP server can be permanently deactivated by placing the entry

DHCP=no

in the file */etc/default/inet* and then restarting the network. Once the interfaces have been deactivated they can be reconfigured.

3.7.6.2 Starting the DHCP client manually

The client for the interfaces lce1, lce2 and lcf1 can be started manually using the following command:

dhcpcd lce1 lce2 lcf1

3.7.6.3 Starting DHCP clients via SYSADM

A DHCP client can be started via SYSADM (see the Section "Configuring LAN controllers").

After installing Reliant UNIX you must first configure the LAN controllers using SYSADM.

3.7.6.4 Client properties

The client can receive the following DHCP messages:

- DHCPOFFER
- DHCPACK
- DHCPNAK

The client sends a DHCPDISCOVER message to the broadcast address 255.255.255.255 in its local network. Each server that receives the DHCPDISCOVER message can respond with a DHCPOFFER message.

If the "LONGEST" option was set in the client configuration file, the client gathers a number of DHCPOFFER

message and then selects the one with the longest options field (i.e. the DHCPOFFER message with the longest configuration data). If the "LONGEST" option was not set, the "Select First" algorithm is used by default. The client this selects the first DHCPOFFER message received.

If the configuration parameters suggested in the options field of the DHCPOFFER are acceptable to the client, it select the appropriate DHCP server by sending a DHCPREQUEST message. If the configuration parameters suggested by the server are still valid, the selected DHCP server sends a DHCPACK. Thus the client received the configuration parameters (e.g. IP address) for an interface. These parameters are stored in the */etc/dhcp* directory in the file *Interface-Name*. The client can continues its network configuration once all of the required interfaces have been configured. The script */etc/dhcpc/dhcpc.netconfig* is started at this point to write the configuration data received from the DHCP server to the system configuration files (*/etc/default/inet, /etc/resolv.conf, ...*).

The original system files are backed up by the DHCP client in the */etc/dhcpc/backup/Client-Pid* directory, where *Client-Pid* is the client process ID from the */etc/inet/dhcpc.pid* file.

If the configuration of the network was concluded successfully, it is recorded in the */etc/inet* directory with the creation of an (empty) file with the name DHCP_CONFIGURED by the DHCP client.

This file can be removed once again from the DHCP client for a short period while it is running under certain circumstances, e.g. if the period of validity of an interface has expired and it can no longer be extended.

After the client is restarted, it can happen that the IP address assigned to the client is still valid. If this is the case, the client attempts to extend the period of validity of this IP address.

The client administers two timers, T1 and T2, that determine at which point the client will attempt to extend the period of validity of its IP address. These timers can be configured (on the server) and, if necessary, are transferred to the client via the DHCPOFFER message.

The default for these timers is:

T1 = 0.5x(period of validity of the IP address)
T2 = 0.875x(period of validity of the IP address)

Once the timer T1 expires, the client attempts to extend the period of validity by sending a DHCPREQUEST unicast to the selected server. If this does not work, the client attempts to extend the period of validity once more by sending DHCPREQUEST broadcast to all servers that can be accessed after the timer T2 expires. If this does not work either, the client stops its network once the period of validity expires.

When the client is *shutdown* or *rebooted*, the IP address is released once more by sending a DHCPRELEASE message.

3.7.7 Problems and their solution

All DHCP server messages are also recorded by *syslog* in the file */var/adm/log/debugmsgs* or */var/adm/log/messages*. Some of the most important messages are explained below:

- Can´t bind to dhcp address: Address already in use

  There may be a DHCP or BOOTP daemon already running (see the Section "DHCP and BOOTP").
- Packet from unknown subnet: <ip-addr>

  The DHCP server has received DHCP packet for which there is no subnet entry in the configuration file (see the Section "Server restrictions").

If a subnetwork contains a number of DHCP servers, it can happen that the DHCP client receives contradictory configuration parameters. If conflicts occur while the network is being configured, they are logged in the file */etc/dhcpc/dhcpc.warning*. Configuration of the network is not, however, canceled. You can set the debugging parameters using the *dhcp-if-config* command (see "Networking Reference Manual")

3.8 IP aliasing

IP aliasing is used to implement the transfer of an IP address from one system to another. This enables the other system to provide services without the user of these services being aware that they are being provided from a remote system. A number of IP addresses (aliases) are assigned to a physical network interface in this process. This enables a system to respond to requests that have not been directed to its original IP address, but instead to IP address which has been entered as an alias for this system.

The enhanced *ifconfig* command, available in Reliant UNIX versions 5.44A0 and later, is used for the configuration and administration of IP aliases. The *ifadmin* command can also be used, although it only supports alias IP addresses that are in the same subnet as the original IP address. A detailed description of the *ifadmin(1M)* command can be found in the Manual Pages for *ifadmin(1M)*.

With the following commands you can output information for an interface that has an IP alias entry:

netstat -p
: Outputs all network interfaces, including those that have an IP alias entry.

  Alias entries can be identified by the fact that they contain the characters ALI.

netstat -i
: Shows all alias interfaces under the name of the original interface along with their statistics.

  The columns *Network* and *Address* contain the values that were specified when configuring the alias interface.

For the "enhanced" stack there is no absolute maximum for the number of IP aliases. For the "fallback" stack the maximum number of IP aliases is limited by the value for the *mtune* variable IPPROVCNT (see the Section "Tuning parameters").

Configuring IP aliases

IP aliases are configured using the *ifconfig* command (*ifadmin*). *ifconfig* must be executed at system startup in order to define the network addresses of all interfaces available in the system. For configuring an alias IP address the command has the following format:
ifconfig *aliasname* alias *org_interface aliasadr* [*parameter*]

aliasname
: is an alias name for the interface *org_interface*.

org_interface
: is the original interface to which an alias is being assigned.

aliasadr
: is the IP address of the alias.

parameter
: denotes parameters such as *netmask* or *broadcast*.

Example

For the interface lce0 with the address 129.103.171.4, an alias with the interface name lce0:1 and the address 129.104.131.7 is to be configured:

ifconfig lce0:1 alias lce0 129.104.131.7

You can view the configured IP alias with:

ifconfig lce0:1

You receive an output something like this:

lce0:1:flags=23<UP,BROADCAST,NOTRAILERS>
inet 129.104.131.7 netmask ffffff00 broadcast 129.104.131.255

You can delete the alias lce0:1 with:

ifconfig lce0:1 delete

A detailed description of the command can be found in the Manual Pages for *ifconfig(1M)*.

3.9 Time synchronization

The *in.timed* daemon synchronizes the system clocks on the machines in a local network.

in.timed is well suited to synchronizing systems in a single subnet, but not to synchronizing several subnets. This is because *in.timed* uses broadcasts. That means that either the subnet routers must allow broadcasts to pass, or *in.timed* processes must be run as "submasters" on the routers themselves. However, it is not always possible or desirable to pass on broadcasts or to start the *in.timed* process on the router, since broadcasts are usually limited to subnets and routers are used as special systems.

in.xntpd is suitable for synchronizing systems in a single subnet and for synchronizing several subnets. However, the *in.xntpd* daemon must be configured.

3.9.1 in.timed

The *in.timed* daemon is activated at system startup if the TIMESYNC variable in the */etc/default/inet* file has the value yes or master. If the TIMESYNC variable has been assigned the value master, the *timed* daemon is started as a potential master.

One machine is defined as the master, the other machines are the slaves. The master is not necessarily the master for the NIS services. The master calculates the network time from the times on the slaves and on the master itself. This is then sent to all the other machines. If the master fails, the slaves automatically define a new master.

By making entries in the */etc/inet/timed* file, the system administrator can specify:

- which machines can become masters
- which slaves' timestamps are included in the calculation of the network

The timedc control program for the timed daemon

The *timedc* command enables you to monitor and control the functions of the *in.timed* daemon. The *timedc* command allows you to:

- measure the differences between the system clocks in your local network
- locate the machine on which the master time server is running
- activate or deactivate logging of messages received by the *timed* daemon on your machine
- analyze and correct synchronization faults between the system clocks of the machines in your local network

This command can only be called under the system administrator login name. You will find more information on *timed* and *timedc* in the "Networking Reference Manual".

3.9.2 in.xntpd

The *in.xntpd* daemon is started during booting of the system if the NTP variable in the */etc/default/inet* file has the value yes.

A configuration file (by default */etc/ntp.conf*) controls the behavior of *in.xntpd*, determining primarily whether or not and how the local *in.xntpd* is to be used as a client or server. The following configurations are possible:

- client of one or more server
- client and server at the same time (symmetrical synchronization)
- broadcast client
- broadcast server

If *in.xntpd* is configured as the client of several servers, *in.xntpd* automatically selects the server from which it accepts timestamps.

The selection criterion is the "stratum" of a server. This is a number indicating how many intermediate stations a server requires to query a timestamp from an external clock with as much precision as possible. Servers with stratum 1 have direct access to the external clock; servers with stratum 2 receive their timestamp from a server with stratum 1, etc. *in.xntpd* always selects the server with the lowest stratum.

From the point of view of *in.xntpd*, the local system clock is also a server. The default value for its stratum is 3. You can prefer the local system clock over other servers or set it to a lower priority by specifying a lower or higher-value stratum with *in.xntpd*'s *fudge* statement.

To call information on the status of the synchronization based on in.xntpd, use the *ntpq* and *ntptrace* commands.

You will find more information on *in.xntpd*, *ntptrace* and *ntpq* in the "Networking Reference Manual".

3.10 Error analysis and diagnostics

As well as describing certain errors that may occur, this section contains hints to help you with error diagnostics.

The various diagnostics commands are described first, followed by a list of common problems with their possible causes and solutions.

You will find information on error analysis and diagnostics for the various components of the network software at the end of the relevant chapters (e.g.the Chapter "Expanding your network", the Chapter "Using the Domain Name Service", the Chapter "The Network File System (NFS)" etc.).

3.10.1 Querying the network status of a machine

The ping command

The *ping* command offers the simplest way to find out if a machine on your network is up or down.

Calling *ping* sends an ICMP datagram to the machine you specify and expects an acknowledgment. The protocol ICMP (Internet Control Message Protocol) uses ICMP "Echo Request" and is responsible for transporting error information and diagnostic information within TCP/IP networks.

Suppose you typed:

$ ping ballet

If the machine is active and able to send an acknowledgment, you will receive the following message:

ballet is alive

However, if the machine is down or cannot receive the ICMP packets, *ping* outputs the following message:

no answer from ballet

If you suspect that a machine may be losing packets even though it is up, you can use the *-s* option of *ping* to try and detect the problem. For example, typing ping -s ballet causes *ping* to continually send packets to the machine ballet until you press the INTERRUPT key. The responses on your screen will resemble the following:

```
PING ballet: 56 data bytes
64 bytes from 129.144.50.21: icmp_seq=0. time=80. ms
64 bytes from 129.144.50.21: icmp_seq=1. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=2. time=0. ms
64 bytes from 129.144.50.21: icmp_seq=3. time=0. ms
...
----ballet PING Statistics----
4 packets transmitted, 4 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/20/80
```

The statistical messages appear after you type the interrupt. The packet loss statistic indicates whether packets have been lost.

Use the *-t lifetime* option to define the "time to live", i.e. the scope, of a query for multicast groups. A value of 0 causes the query to be sent only to the local computer and a value of 1 causes the query to be sent within the local network. Values 2 through 225 cause the query to be passed to other networks if a multicast router has been configured for the local network.

If *ping* outputs an error, use the *netstat* command described below to request network statistics.

The fping command

fping is a program similar to *ping* which uses the Internet Control Message Protocol to determine whether a machine is online. Unlike *ping*, *fping* allows you to enter any number of machines on command level, or you can specify a file containing a list of machines.

fping does not provide any detailed information on how long a machine takes to respond since its aim is merely to indicate whether the system is ready.

You can use the following command, for example, to list all accessible machines in the network:

```
fping -d -f /etc/inet/hosts
```

The traceroute command

The *traceroute* command can be used to determine which route a packet takes to different systems or which gateway it rejects. *traceroute* does this by analyzing the *ttl* (time-to-live) field in the IP protocol header and then attempts to get the ICMP message TIME_EXCEEDED from any gateway on the route to a specific system.

 This command is only to be used for diagnostics purposes.

The netstat command

The *netstat* and *enetstat* commands generate displays that show network status. You can run *netstat* (or *enetstat*) to display the status of network traffic in table format, including routing table information (available routes and their status), and interface information. You will find more information in the description of *netstat(1M)* (and *enetstat(1M)*) in the "Networking Reference Manual". According to the command line options chosen, *netstat* (or *enetstat*) provide a variety of outputs which are most useful for system administration.

Protocol statistics

You can use the option *-s* to have the statistics of the protocols IP, ICMP, TCP, UDP as well as information on routing changes displayed. When you run netstat -s, the result is a display resembling the following:

```
ip:
227201 total packets received
0 bad header checksums
0 with size smaller than minimum
0 with data size < data length
```

0 with header length < data size
0 with data length < header length
12 fragments received
0 fragments dropped (dup or out of space)
0 fragments dropped after timeout
3086 packets forwarded
0 packets not forwardable
0 redirects sent
icmp:
0 calls to icmp_error
0 errors not generated 'cuz old message was icmp
Output histogram:
address mask reply: 3
. . .
Input histogram:
address mask request: 3
3 message responses generated
tcp:
connections initiated: 7
connections accepted: 59
connections established: 66
connections dropped: 5
embryonic connections dropped: 9
. . .
packets rcvd after "close": 1
rcvd window probe packets: 0
rcvd duplicate acks: 90
rcvd acks for unsent data: 0
rcvd ack packets: 11230
bytes acked by rcvd acks: 901832
rcvd window update packets: 95
udp:
0 incomplete headers
0 bad data length fields
0 bad checksums

The statistical information can indicate areas where a protocol is having problems. For example, statistical information from ICMP can indicate where this protocol has found errors.

Status of the network interfaces

The -*i* option of *netstat* shows the status of the network interfaces of the local system. Here is a sample display produced by netstat -i.

Name Mtu Network Address Ipkts Ierrs Opkts Oerrs Collis
lce0 1500 sni_muc neptun 14093893 8492 10174659 1119 2314178
lce1 1500 sni_muc orion 9299762 54423 12451748 0 77512

Using this display, you can find out how many packets a machine thinks it is transmitting and receiving on each network. For example, the input packet count (*Ipkts*) displayed for a BOOTP server may increase each time a client tries to boot, while the output packet count (*Opkts*) remains steady. This suggests that the server is seeing the boot request packets from the client, but does not realize it is supposed to respond to them. This might be caused by an incorrect address in the *hosts* database.

On the other hand, if the input packet count is steady over time, it means that the machine does not see the packets at all. This problem is probably caused by a hardware error or faulty cable.

Route information

Calling *netstat* with the *-r* option activates the output of the routing information used in the system.

Subnet masks

The *-N* option outputs subnet masks and static routes:

```
$ netstat -rN
Routing tables
Destination Subnetmask  Gateway   Flags Refcnt Use Interface
127.0.0.0   0xff000000  127.0.0.1 UHNs  0      0   lo0
...
```

The enetstat command

On systems using the "enhanced" protocol stack, the transformation of the internal route organizational structure into a tree structure has also changed the contents of the routes managed in this structure. For example, along with the familiar system and network routes, you now also receive information that was managed in the so-called arp cache of a system using the "fallback" protocol stack (see "link-level" route entries).

The output of the *netstat -r* command displays the system and network routes in systems using both the "enhanced" and the "fallback" protocol stack. The additional information available for systems using the "enhanced" protocol stack can be requested with the *enetstat -r* command. Further information on the *enetstat -r* command can be found in the Section "IP routing for the "enhanced" protocol stack solution".

3.10.2 Querying interfaces

The information output for the following commands contains values that have been accumulated since the last system start or since the last input of *netstat* with the *-z* option.

The ifconfig command

The *ifconfig* command displays information on how a specific interface has been configured.

You simply enter the name of the interface for which you require information (e.g. the Ethernet interface). You can obtain the name of the interface by issuing the command *netstat -i*.

For instance, entering

$ ifconfig lce1

might result in the following output being generated:

lce1: flags=63<UP,BROADCAST,NOTRAILERS,RUNNING>
inet 129.144.50.28 netmask ffffff00 broadcast 129.144.50.0

This example shows the messages output by *ifconfig* when querying an Ethernet interface. (The output will, of course, be different if you query a different interface type.)

You can glean all sorts of information on the status of the Ethernet interface from the messages displayed. The flags line shows that:

- the interface is active (up)
- it is capable of sending broadcasts
- it does not use "Trailer Encapsulation". This is always the case for Reliant UNIX (you will find information about "Trailer Encapsulation" in "Richard Stevens, TCP/IP Illustrated, Volume 1: The protocols"; see the chapter on Related publications.)
- it is currently running without problems.

The second line contains the following information:

- the Internet address of the machine
- the network mask currently being used
- the broadcast address of the network.

If the status report indicates that the interface is down, an error might have occurred. In this case, *ifconfig* provides other functions and these are described further under *ifconfig(1M)* in the "Networking Reference Manual".

Displaying etherstat network interface statistics

This command supports Ethernet, Gigabit Ethernet, Token Ring and FDDI interfaces.

The *etherstat* command does not work for the loopback interface lo0, the slip interfaces sl0, slhios, slehios and PPP because no statistics are kept for these interfaces.

The *etherstat* command can be used to obtain the following statistical information about the network interfaces of a machine:

- the MAC address of the interface
- the hardware and firmware version of the interface, provided it can be read
- statistical information about packets sent or received via the interface
- information about the link layer interface

A machine can have more than one interface if it is connected to more than one network or network segment.

The *etherstat* command displays information on all configured interfaces. The interfaces are referred to by name. You can specify names in a variety of formats:

- The name of the interface type without any option

To request statistics for all interfaces of the specified type.

- The name of the interface type followed by a digit

  This requests the statistics for a particular interface of the specified interface type.

These names can differ depending on the hardware being used, and may change in later hardware versions. For example:

lcf0 = FDDI (RM600)

dfe0 = FDDI (RM600-PCI, RM400-PCI)

lce0 = Ethernet (RM600)

pnet0
= Ethernet (RM400-10)

lct0 = TokenRing (RM600

madge0
= TokenRing (RM600-PCI, RM400-PCI)

alt0 = Gigabit-Ethernet (RM600-PCI, RM400-PCI)

The names of the interfaces actually configured can be output with the *netstat -i* command.

If no options have been specified, the following is output for each configured interface (provided that the counter is supported or makes sense):

- the name of the interface
- the MAC address of the interface
- receiver overflows:

  how often the receive buffer overflowed
- receiver CRC errors:

  how many CRC (Cyclic Redundancy Check) errors occurred during reception
- receiver alignment errors:

  how many alignment errors were found during reception
- receiver short packet errors:

  how many short packets were received
- receiver queue full:

  how many packets were thrown away because of memory bottlenecks
- receiver packets for unknown sap:

  number of packets received for protocols which do not run on this machine (usually broadcast packets).
- receiver packets discarded for other software reasons

  how many packets received were thrown away because of other error conditions (e.g. format errors in the Streams message in *ullc*, packets too short in *ullc*, queue above *ullc* full, IP input buffer full)
- good packets received:

  how many packets were received without error
- transmitter collisions:

  (valid only for Ethernet in half-duplex mode) how many collisions occurred
- transmitter excessive (16) collisions:

  (valid only for Ethernet)
  how many packets could not be sent because they had at least 16 collisions
- transmitter heartbeat (SQE) check failures:

  (valid only for Ethernet) how many SQE check errors were found. These errors occur if the connection between the machine and the transceiver is faulty or if the SQE check is turned off on the transceiver.

- transmitter packets discarded for other software reasons

  how many packets sent were thrown away because of other error conditions (e.g. memory bottleneck when processing packets in *ullc*, *ullc* queue is full, the output buffer before the hardware driver is full, or packets to be sent using the direct path can no longer be accepted)
- good packets transmitted:

  how many packets were transmitted without error

The information output are values which have accumulated since system startup or since the last time the *etherstat* command was issued with the *-z* option.

The statistics available depend on the controller type. Certain statistics are not available on some interfaces. Statistics are only output once for each interface specified, even when entered more than once in the parameter list.

Example:

You would like all information concerning existing interfaces.

etherstat

Output:

```
lce0  (08-00-14-11-28-82):
0 receiver overflows
0 receiver CRC errors
0 receiver alignment error
0 receiver queue full
0 receiver packets for unknown sap
64543 good packets received
35 transmitter collisions
0 transmitter excessive (16) collisions
0 transmitter heartbeat (SQE) check failures
4823 good packets transmitted
vor good packets received
0 receiver packets discarded for other software reasons
vor transmitter collisions
0 transmitter underflow
vor good packets transmitted
0 transmitter packets discarded for other software reasons
```

No errors were received at the lce0 interface when data was received. 64543 packets were received without errors. No errors were detected during sending either. 4823 packets were sent without errors.

3.10.3 Querying data traffic in the network

The *tcpdump* command is a useful tool, which you can use to monitor and record the data traffic in a network. However, in order to use it and evaluate the output generated or the data recorded, you need a certain amount of knowledge of the Internet protocols. You should use *tcpdump* whenever you suspect there are communication problems between two or more machines in a network.

tcpdump can only be used for LCT Token Ring, SLIP and PPP interfaces in conjunction with the "enhanced" protocol stack. However, only IP traffic can be intercepted. This means therefore that it is not possible to check the LCP protocol, for example, when setting up a PPP connection.

tcpdump also allows data traffic transmitted via the *loopback* interface to be recorded and checked. For further details, please refer to the reference page for the *tcpdump* command.

If the packets contain data transmitted by means of an Internet protocol, *tcpdump* decodes the protocol information and presents it in a readable form. *tcpdump* can decode the protocol elements of most Internet protocols, e.g. IP, ARP, RARP, TCP, UDP, BOOTP, DNS etc.

Example:

You can use the following command to trace all the data traffic between machines A and B:

tcpdump machine *machine_A* and *machine_B*

The command can be called on any machine in the network that supports *tcpdump*, including machine A or B. However, it is advisable to use a third machine so as not to place an additional burden on machine A or B. If *tcpdump* is run on machine A or B, it cannot always be guaranteed that the displayed data appear on the network.

In addition to the command alone, as in the above example, *tcpdump* offers a range of options and filter functions with which you can isolate and display in detail the interesting parts of the data stream. Among other things, the *-m* option makes it possible to activate multicast promiscuous mode (in the basis of Ethernet). If you use the keyword *multicast*, you can force *tcpdump* to monitor the exchange of multicast packets. You will find a detailed description of the *tcpdump* command in the "Networking Reference Manual".

To execute the *tcpdump* command, you require system administration authorization.

3.10.4 Notes on system load

High system load can have many causes, and these cannot all be discussed here (see the "Tuning Guidelines").

At this point it is important to mention the particular load placed on the system by broadcast messages, which may be received unnecessarily by your system. Daemon processes such as the *routed* and *rwhod* daemons receive such messages.

The *rwhod* daemon greatly impairs system performance in large networks and should therefore only be activated in exceptional cases. By default, the daemon is not started (the */etc/default/inet* file contains the entry RWHOD=no).

The routing daemon *routed* should only be started on systems with adequate memory and, even then, only when several routers are available. To deactivate *routed*, edit the */etc/default/inet* file as follows:

GATEWAY=no

However, if your machine has limited memory and your network has only one router, then it is recommended that you run *routed* only on the router and disable it on all other machines.

For more information about configuring a router, see the chapter entitled Expanding your network.

3.10.5 Problems because of improperly changed machine names

The machine name should only be changed using the *sysadm* system, since the change must be made at several places and only the use of *sysadm* ensures that these changes are made consistently.

If you have used the command *uname -S* to change the machine name, it is not enough just to enter the new name in the */etc/inet/hosts* file. If a name change remains inconsistent, it can have serious effects on the behavior on certain network services. The *rpcbind* daemon cannot be started and thus no other RPC service, such as NFS or NIS. It is then impossible, for example, to mount an NFS file system from that machine.

You can solve the problem by entering the new machine name in place of the old one in the */etc/net/*/hosts* files. The system must then be restarted.

3.10.6 Problems when logging into a remote machine

Problem:

No connection from or to a machine is possible. The following message, for example, is displayed on the system console:

For RM600:

DRV: Warn: lce 62 (lce0) No carrier

For RM400:

CMN: Warning: cmn 14259 mac596 4719 logid0: NO_CARRIER

Cause:

The connection between the Ethernet interface and the transceiver is faulty. The LAN cable is probably not properly inserted.

It may be that your LAN cable is damaged or faulty. The carrier signal is then no longer received clearly.

Solution:

Check the cable and the connector. Replace your LAN cable.

Problem:

It is possible to establish a connection to a remote machine, but if you attempt to establish a connection from a remote machine to the local machine, the following message is issued:
Connection timed out.

or
Connection refused.

Cause:

The *inetd* on the local machine fails to start the appropriate daemon (*in.rlogind* or *in.telnetd*).

Solution:

Check the output from the command *netstat -a* on entries for *telnet* and *login*.

Most of the daemons are activated by the *inetd* daemon. Use the *ps* command to check whether this daemon is running, and check the file */etc/inet/inetd.conf* to see whether the corresponding daemon (*in.rlogind* or *in.telnetd*) has been entered.

Problem:

It is not possible to establish a connection to a remote machine. The *ping* command is not functioning.

Cause:

The machine's network interface is not configured or there is a fault with the hardware.

Input: netstat -i

1. If there is a zero address instead of a machine name in the address column for an interface, the Ethernet interface is not configured.
2. If errors are reported for the transmission of data packets (xxx packets transmitted with errors), there is probably a fault in the hardware.

Solution:

For 1:

Check */etc/default/inet* to see whether the Ethernet interface is activated (STATE=active). Call *sysadm* and configure the system using the *configuration* menu item. Then restart the system.

For 2:

Contact your customer service department.

3.10.7 Problems with the time of day

If the system clock on **one** machine is changed, the system clock on **all** the machines for which time synchronization has been enabled with *in.timed* is generally also changed.

At intervals (every four minutes), the *timed* master uses the timestamps of the *timed* slaves and its own time to calculate an average time, the network time.

Thus an incorrect time on a slave or the master affects the network time after four minutes at the latest and leads to an incorrect timestamp for all machines.

There are two ways to eliminate this problem:

1. All potential *in.timed* masters are started with the *-F* option. Calculation of an average time is then suppressed, and the master distributes its own timestamp as the network time.
2. The slaves whose times are included in the calculation of the average time are listed in the */etc/inet/timed* file. The timestamps of other slaves are then not taken into account.

4 Expanding your network

In time, you may need to attach more machines to your network than you have allowable addresses, or you might want two networks in different buildings to share resources. In this chapter we first discuss generally some of the hardware that connects networks. Then we will talk about two ways you can expand your existing network:

- by connecting two or more networks to create an internetwork (see the Section "Setting up an internetwork")
- by creating a subnet on your network (see the Section "Routing with subnets").

4.1 Hardware devices for expanding the local network

If your local network fails to meet your needs, you may want to expand it. This section presents some of the machines that allow you to connect two or more local networks.

- Repeater

  This is a device that connects two networks on the network access layer (see ...) and copies each bit of a packet from one to the other.

  There are inherent limitations in the use of repeaters, given that they copy **all** data from one network to the other. Implementations like Ethernet, which require that data traverse the network within a specific amount of time, impose a maximum permitted number of repeaters and maximum distance between them.

- Bridge

  This is a device used at the data link layer of the network protocol model. It selectively copies packets from one network to another.

  Bridges differ from repeaters in several ways. Because copying done by bridges is selective, this reduces traffic on the destination network. Since bridges copy whole packets, the geographical or timing constraints of the basic physical network can be extended.

- Router □
  (sometimes called a "gateway")

  This is a machine that forwards packets of a particular protocol family, in this case TCP/IP, from one logical network to another. A logical network makes sense only to a particular protocol. Usually there is a one-to-one mapping between physical network and logical network, but subnets (explained in a later section) and bridges are exceptions to this rule. A collection of logical networks (all using the same protocol) connected via routers is called an "internetwork".

  The router may forward packets between different physical types of networks, for example, from an Ethernet to a token ring network. It can also forward packets between two logical networks of the same type.

  During forwarding, the router reads the destination address specified in the IP header, and then looks in the routing table for a route. In addition to routers that only support one protocol type, it is possible to have a multiprotocol router - a single device that forwards packets for several protocol types, such as TCP/IP, ISO, or XNS.
- Application gateway □
  (sometimes called a "relay" or "forwarder"):

  This device and associated software enable networks using different protocols to communicate with each other. Because it translates protocols existing at all layers of the protocol model, you can use gateways to connect networks that differ on all layers from each other. You can also use an application gateway to connect parts of the same physical network that are using different protocols.

4.2 Setting up an internetwork

The TCP/IP protocol family provides intercommunication among host machines, terminal servers, and other equipment on one or more local-area or wide-area networks. A typical local network serves a limited area - within a building or between neighboring buildings. Stations attached to the local network may function as file servers, mail servers, print servers, terminals, and workstations.

Networks are often expanded by linking local networks together via a machine called an IP router. A network configuration consisting of several local networks linked together by routers is often called an internetwork.

Be careful not to confuse the term "internetwork" with the "Internet". Your company can set up an internetwork if it needs to expand communications services, and you or one of your co-workers may have the responsibility of managing it. By contrast, the Internet is the name of a particular internetwork.

4.2.1 IP routing for the "enhanced" protocol stack solution

One of the characteristics of the "enhanced" protocol stack solution is a complete reorganization of the IP routing schema.

With System V Release 4, two tables were used for IP routing: one table contained routing entries for machines and the other table contained routing entries for the networks. This organization has been replaced by a tree structure for the routing entries. The significance of an entry within the routing tree has also been extended.

A new API has been introduced for access to the routing tree. For reasons of downwards compatibility, the previous API (i.e. the socket Ioctl's SIOCADDRT and SIOCDELRT) can still be reimplemented for access to the routing information. This means that all of the commands (or daemons) which influenced both tables in the kernel of a system with the "fallback" protocol stack solution are still supported in the environment of the "enhanced" protocol stack.

4.2.1.1 Access to the routing information in the "enhanced" protocol stack

Since the organization of the routing information in a routing tree within the "enhanced" protocol stack provides more options than the tabular implementation of the "fallback" protocol stack, the "enhanced" solution offers some modified or enhanced commands for making the most of these options. These commands are: *eroute*, *in.erouted*, *rtquery* and *enetstat*.
eroute

The *route* command can be used in a system with the "enhanced" or with the "fallback" protocol stack. An enhancement of the *route* command is the *eroute* command, which is only supported in a system using the "enhanced" protocol stack and is based on the 4.4BSD *route* command.

in.erouted
: What applies to the commands *route* and *eroute*, also applies to the daemons *in.routed* and *in.erouted*. The most important new characteristic of the *in.erouted* daemon is the support of RIP Version 2.

rtquery
: The *rtquery* command can be implemented as a tracer for the *in.erouted* daemon or as an instrument for table queries from remote routing daemons. The command can only be used in a system with the "enhanced" protocol stack.

enetstat
: For the *enetstat* command, the same applies as is described for the *eroute* command. The *enetstat* command permits the output of all entries from the routing structure and of statistical information.

More detailed information on the "enhanced" or the "fallback" protocol stack can be found in the Section "Selecting the "enhanced" or the "fallback" protocol stack". The output from the *eroute* and *enetstat* commands is explained in the Section "Getting routing information".

For more information on the commands named above, refer to the "Networking Reference Manual".

4.2.2 Configuring a router

A TCP/IP network usually interconnects a number of machines. Your machine is connected to a TCP/IP network via a hardware network interface. Individual TCP/IP networks are in turn interconnected via IP routers. IP routers forward IP packets from one TCP/IP network to another, and can exchange routing information with each other to deliver packets across a number of networks. Other types of routers may forward traffic for protocol families other than TCP/IP.

If all of the machines at your site are connected to a single TCP/IP network that is **not** interconnected with any other TCP/IP networks, an IP router is unnecessary. If your site comprises many TCP/IP networks, or if you wish to interconnect your IP network with other TCP/IP networks, you must configure the interconnections with IP routers in order for all machines to communicate. A Reliant UNIX system can also function as an IP router.

When sending an IP data package, the originating machine must first find the local interface from which it wants to transmit the package. If the destination address is on a different segment of the local network than the transmitting machine, a router or gateway must be found to relay the data packet to this address. In a system with the "fallback" protocol stack, there are two routing tables in the system kernel, the entries of which are processed with the *route* command:

- a table with routes for computers, specified by their Internet addresses
- a table with the routes for networks, specified by the Internet addresses of the networks

In a system with the "enhanced" protocol stack, the routing tree can be processed with the route and eroute command.

A route points to a network interface on the local machine. If the destination address is not in the same network segment, the route also contains the Internet address of the computer in the local network segment which is acting as the gateway.

A default route entry can also be specified. This is then used for all IP packets for which no suitable route is available.

Many types of machines may serve as IP routers. A number of vendors offer IP routers, i.e. machines dedicated entirely to the function of IP routing. A system may act both as a machine (offering network services such as *rlogin*) and a router.

Depending on the size and requirements of your network, there are various ways of configuring your routers:

1. Static routing

   In static routing, the router transmits data packets between two networks using the information in its routing table. Changes to the routes must be entered directly in the table.

2. Dynamic routing

   In dynamic routing, the routers communicate with each other by means of routing daemons. The routing daemons exchange information on the routes possible in the network, establish when specific routes have been interrupted and maintain dynamic routing tables. A distinction is drawn between active and passive routers. Active routers send routing information themselves, whereas passive routers receive routing information and exchange route information with other routers but do not themselves make any changes to routing information.

Setting a machine up as a router first involves installing the relevant hardware. Once the router is connected to the networks it joins, you must configure the router's software. Before actually configuring the software, make the following preparations.

- Assign the router a machine name and a unique Internet address for each network it is on. The Internet Protocol architecture requires each interface to have a unique Internet address (see Internet addresses).
- If routing entries are to be resolved by name, make sure you have registered NIC network numbers for each network the router is to connect.
- Make sure that the TCP/IP software is running properly.

The following steps are only necessary when you are using neither the NIS nor the DNS.

1. Edit */etc/inet/hosts*.

   Enter the machine name and Internet address of each interface on the router.

2. Access */etc/inet/networks* and add the network names and Internet addresses of all the networks that the router can reach.

4.2.2.1 Setting up a static router (no routing daemon)

1. Since more than one network interface is to be used by TCP/IP, the default interface can be entered in the STANDARDIF variable of the */etc/default/inet* file. All additional interfaces are specified in the variables EXTIF, EXTIF1 and so on. You should use the *sysadm* menu system or *WebSysAdm* to make the entries.
2. Create the */etc/inet/routes* file, and enter the required routes for all networks to which the machine is connected:

   /usr/sbin/route add *network1 machine hops*
   /usr/sbin/route add *network2 machine hops*

   network1 and *network2* are the names or network numbers of the networks to which the router is connected. *machine* is the name or the Internet address of the router as entered in */etc/inet/hosts*. The */etc/inet/routes* file is read at system startup and the routing file created from it.

3. If it is not yet set, activate IPFORWARDING in the kernel. This ensures that packets which are not intended for this system are forwarded.
   - On a system with the "enhanced" or "fallback" protocol stack enter

     /etc/conf/bin/idtune IPFORWARDING 1

     and then regenerate the system kernel with the call
     /etc/conf/bin/idbuild

     Then restart the system with currently generated system kernel.
   - On a system with the "enhanced" protocol stack, IPFORWARDING can be activated for the running system with

     sysctl -w net.inet.ip.forwarding=1.

     This setting remains valid until the next change or the next system start. Permanent changes are made with an entry in the */etc/inet/rc.sysctl.local* file.

For more detailed information on the "enhanced" or the "fallback" protocol stack solution, refer to the Section "Selecting the "enhanced" or the "fallback" protocol stack".

More information on *sysctl* can be found in the "Networking Reference Manual". The possible *sysctl* parameters are described in the */etc/inet/rc.sysctl.readme* file.

Example of static routing information in configuration files

Consider a router called sun that connects two networks and, thus, has two network interfaces. For the network software to work properly, each interface in the */etc/inet/hosts* file must be assigned a machine name.

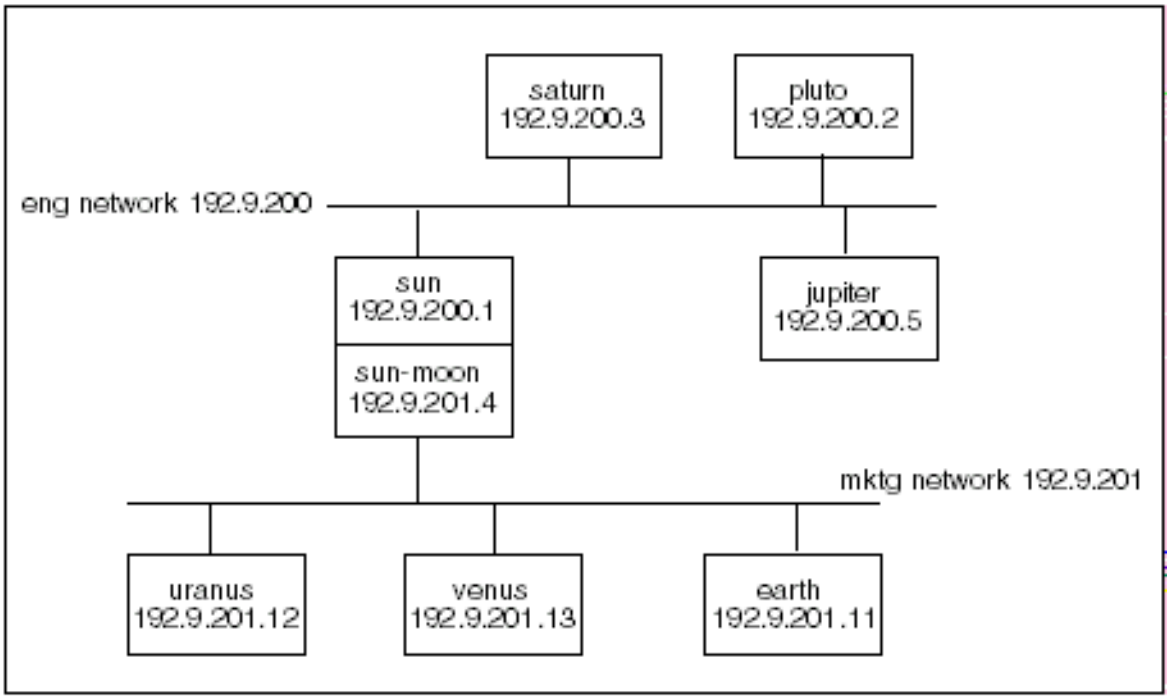


Figure 4: Sample configuration: static routing

Because the internetwork on which sun resides is a simple one - there is only one router, and no routing information is exchanged - a routing daemon is not required.

Notice in the sample file that, on the second network, sun has another name - sun-moon (sun is the router's primary name). For ease of administration, similar machine names are used for each network, i.e. users on both networks can address the router by the primary name sun. Only the system administrator is aware of the difference.

Note also that sun (alias sun-moon) has two Internet addresses in the file */etc/inet/hosts*.

```
#
# sample hosts file
#
# 192.9.200 -- eng -- Engineering Network
#
192.9.200.1       sun
192.9.200.2       pluto
192.9.200.3       saturn
192.9.200.5       jupiter
# 192.9.201 -- mktg -- Marketing Network
#
192.9.201.11      earth
192.9.201.12      uranus
192.9.201.13      venus
192.9.201.4       sun-moon
```

The following illustration is an excerpt from the */etc/inet/networks* file on the sun router. This file contains only the names and addresses of the networks to which sun is connected.

```
#
# sample networks file
#
eng      192.9.200   # Engineering Network
mktg     192.9.201   # Marketing Network
```

4.2.2.2 Setting up a dynamic router

1. Edit the defaults in the */etc/default/inet* file.

   The beginning of the file should contain the following lines, for example:

   ```
   #ident "$Header: inet 1.3 90/04/02 $"
   STATE=active
   STANDARDIF=lce1 ...
   EXTIF= ...
   INTERFACES="$STANDARDIF $EXTIF"
   OLDBROADCAST=yes
   RWHOD=no
   GATEWAY=no
   GLOBALPW=yes
   TIMESYNC=no ...
   GATED=yes
   ...
   ```

   Since TCP/IP uses more than one interface, the default interface can be entered in the STANDARDIF variable. The additional interfaces must be specified in the variables EXTIF, EXTIF1 and so on. You should use the *sysadm* menu system or *WebSysAdm* to make the entries.

   The *gated* routing daemon is suitable for processing several routing protocols (RIP, BGP, EGP, HELLO, SLSP, ICMP, OSPF and RDISC); activate this daemon by setting the GATED variable in the */etc/default/inet* file:

   GATED=yes

   You will find other options in the section entitled Installing a routing daemon.

2. If it is not already set, activate IPFORWARDING in the system kernel. This ensures that packets which are not intended for this system are forwarded.

   - On a system with the "enhanced" or "fallback" protocol stack enter

     /etc/conf/bin/idtune IPFORWARDING 1

and then regenerate the system kernel with the call
/etc/conf/bin/idbuild

Then restart the system with currently generated system kernel.

- On a system with the "enhanced" protocol stack, IPFORWARDING can be activated for the running system with

  sysctl -w net.inet.ip.forwarding=1.

  This setting remains valid until the next change or the next system start. Permanent changes are made with an entry in the */etc/inet/rc.sysctl.local* file.

For more detailed information on the "enhanced" or the "fallback" protocol stack solution, refer to the Section "Selecting the "enhanced" or the "fallback" protocol stack".

More information on *sysctl* can be found in the "Networking Reference Manual". The possible *sysctl* parameters are described in the */etc/inet/rc.sysctl.readme* file.

Once you have set up the router, enter the machine names and Internet addresses of the router interfaces in the */etc/inet/hosts* and */etc/inet/networks* files on every system in your local area network. If you are using the NIS network information system, you only need to make the changes on the NIS master server. You can then distribute the changed information to the other machines. (See the chapter entitled The Network Information Service (NIS).)

4.2.2.3 Routing with subnets

Many network operators decide to divide a network into several subnets. Subnets are logical subsections of a TCP/IP network. Small enterprises which assign class C numbers to each of their local networks may encounter difficulties in managing network numbers as their company increases in size. It is better to assign a class B network number to each large department: for example, one number to the development department, one to the production department, and so forth. Each of these networks can then, in turn, be subdivided into separate physical networks by creating subnets. Thus, if major changes are made to one of the company areas, they will not affect the whole network structure.

When creating a subnet, you can define the address structure in an address mask. If a mask of this kind applies to the whole network, it is referred to as a network mask. A network mask defines which bits in the Internet address are reserved for the network number and subnet number. You will find detailed information on the structure of the network mask in the Section "Selecting/assigning subnet numbers and using network masks".

The netmasks file

The */etc/netmasks* file contains the default network mask for your system, and can be edited when subnets are created. */etc/netmasks* could, for example, have the following contents:

```
#
# Netmasks
#
# only non-standard subnet masks need to be defined here
#
# Network       netmask
128.32.0.0      255.255.255.0
```

Every network which is subdivided into subnets is given its own entry, consisting of a network number and a network mask. Each entry must be on a separate line.

Only network masks with contiguous single bits (e.g. 0xfff00000) should be used. Classless network masks (e.g. 0xf0f00000) are not supported by most routing daemons.

You can modify the network masks manually with the *ifconfig* command. For more information on *ifconfig*, refer to the description of *ifconfig(1M)* in the "Networking Reference Manual".

The entry in the */etc/netmasks* file for a class B network with the network number 128.32 and an 8-bit wide

subnet field (and therefore also an 8-bit wide host field) would be as follows:

128.32.0.0 255.255.255.0

Up to SINIX V5.41, the network mask of the network interface was used to find an entry in the routing tables. However, this method did not allow genuine subnet routing. Under SINIX V5.42, the "best match" algorithm was used to determine the best route and to come closer to subnet routing. Under Reliant UNIX 5.43 or later, a route-specific network and subnet mask has been implemented for genuine subnet routing.

In Reliant UNIX 5.45 systems with the "enhanced" protocol stack and the routing tree structure, the options for subnet routing when adding routes to the system have been retained. Also retained is the "best match" algorithm for deriving a subnet mask when network routes are added without explicit subnet masks.

For more information, refer to the sections Structure of the routing information and Routes and corresponding network masks.

You will find further information on network masks in the description of *netmasks(4)* in the "Networking Reference Manual".

Creating a multicast default route

If you want to work with IP-Multicasting, you must create a default route for multicast packets. This is done with the *route* command. For example, to define interface lce1 as the default interface for transmitted multicast data packets, enter the following command:

/usr/sbin/route netmask 0xf0000000 add net 224.0.0 lce1 0

You can enter this line in the */etc/inet/routes* file.

Examples of subnets

The following examples show network installations with and without subnets:

128.32.0.0	Berkeley	Class B (with subnet)	Netmask 255.255.255.0
36.0.0.0	Stanford	Class A (with subnet)	Netmask 255.255.0.0
10.0.0.0	Arpanet	Class A (without subnet)	Netmask 255.0.0.0

All of the University of California at Berkeley is assigned the network number 128.32.0.0, so that any external router only needs to know one route to reach Berkeley. Within the campus, a class C subnet mask is used to give each local network a subnet number, with 256 machines on each of the 256 possible subnets. Stanford University uses a class A network number with a class B network mask, for 254 subnets of 65534 machines each. The Arpanet is a class A network without subnets; therefore, the default class A netmask is used.

Structure of the routing information

In Reliant UNIX versions up to and including V5.44, the system kernel maintained two tables with IP addresses: one table for machine routes, the other table for network routes. Depending on the type (machine or network), a given route was entered in one or the other table. The default route with the address 0.0.0.0 was entered in the network routing table.

When searching for a route there was a specific hierarchy to be followed: First a machine route was searched for. If the search result was negative a network route was then searched for. If this search also produced a negative result the default route was then searched for. Only when all three search results were negative was the destination deemed inaccessible.

The routing tree in a system with the "enhanced" protocol stack contains both machine and network routes and uses the addresses as sorting criteria.

The aim of a route search is to find the entry in the routing tree that best matches the specified destination. The term "best" implies that a machine route takes priority over a network route and a network route takes priority over a default route.

For every entry in the routing tree structure there is a network mask. For machine route entries a mask is implied, for which all bits are set.

This network mask also exists when a route has been installed using the *route* command without explicit specification of a (sub)network mask. It is then calculated using an adjustable derivation rule.

See also the section Routes and corresponding network masks.

An entry in the tree structure matches a search key when the search key is linked to the network mask by means of a logical AND operation and when the result is the same as the entry.

Example:

The route entry is to be determined for the search key 139.22.0.2:

		Search key = 139.22.0.1		Search key = 139.22.0.2	
		Machine route	Network route	Machine route	Network route
1.	Entry	8b160001	8b160000	8b160001	8b160000
2.	Search key	8b160001	8b160001	8b160002	8b160001
3.	Network mask	ffffffff	ffff0000	ffffffff	ffff0000
4.	Logical AND operation with 2. and 3.	8b160001	8b160000	8b160002	8b160000
5.	1. equals 4.	Yes	Yes	No	Yes

Table 1: Example of a routing search

The above example shows that

- the network route entry 139.22.02 is found for the destination address,
- although the address 139.22.01 matches both the route network entry as well as the machine route entry, the machine route entry takes precedence because it is a closer match for the specified destination address.

This decision is backed up by the organization of the routing tree.

Routes and corresponding network masks

In a system with the "enhanced" protocol stack the contents of the route entries and the corresponding network masks have changed. There are two distinct cases when a route is entered using the *route* command:

1. A specific network mask is specified in the *route* command.

   In this case the specified network mask is used on the route to be installed.

2. No network mask is specified in the *route* command.

   In this case the network mask is derived from the destination address, which is considered as a network address (in contrast to a fully-specified machine address).

There are three alternative algorithms for deriving the network mask from the destination address:

1. "Class based"

   The network mask derived from the destination address of the route is based on the class association of the destination address. This means

   0xff000000 for a destination address in class A,

   0xffff0000 for a destination address in class B

   0xffffff00 for a destination address in class C

   Example:

   For the network destination address 139.22.228.0, the "class based" rule derives a network mask 0xffff0000, because the destination network is a class B network.

2. "Widest"

   The network mask is derived from the lowest non-zero byte of the network address.

   Example:

   For the network destination address 139.22.228.0, the "class based" rule derives a network mask 0xffff0000, because the destination network is a class B network. The "widest" rule derives a network mask 0xffffff00, because the third byte in the destination network address is a non-zero byte.

   From the network destination address 139.0.0.0, the "class based" rule derives a network mask 0xffff0000. The "widest" rule would derive a network mask 0xff000000 in this case, because the second, third and fourth byte in the destination address of this network is null.

3. "Class based extended"

   The network mask is derived from the class of the network destination address and extended with the lowest byte outside of the class range.

   Example:

   From the network destination address 139.22.228.0, the "class based extended" rule derives a network mask 0xfffffc00, which the class B network extends with 0xfc00. This is the shortest mask which covers the lowest byte in the third byte of the network destination address (binary 1110 0100).

   From the network destination address 139.60.0.0, the "class based extended" rule derives a network mask 0xffff0000, because the address is a class B address and no bit is set in the third and fourth byte. From the destination address 139.0.0.0, the "class based extended" rule derives a network mask 0xffff0000, because the address is a class B address.

In the case of a system with the "enhanced" protocol stack, the derivation rule used can be checked and defined using the *sysctl* command and the *net.inet.ip.rtdefaultmask* variable. Possible values are:

0 for the "class based" algorithm

1 for the "class based extended" algorithm

2 for the "widest" algorithm

The default 1 ("class based extended") implements the "best match" algorithm from previous system releases and the "fallback" protocol stack.

"Best match" or "genuine" subnet routing

Even if there is no "best match" algorithm for systems with the "enhanced" protocol stack, and instead there are conclusion rules for deriving (sub)network masks from the entries in the routing tree structure, the following still applies: if an unexpected routing decision has been made in a system with the "fallback" protocol stack as a result of an error in the "best match" algorithm and explicit specification of (sub)net masks is required, the same is recommended with the "enhanced" protocol stack and the routing tree structure.

The following example illustrates the problems involved here:

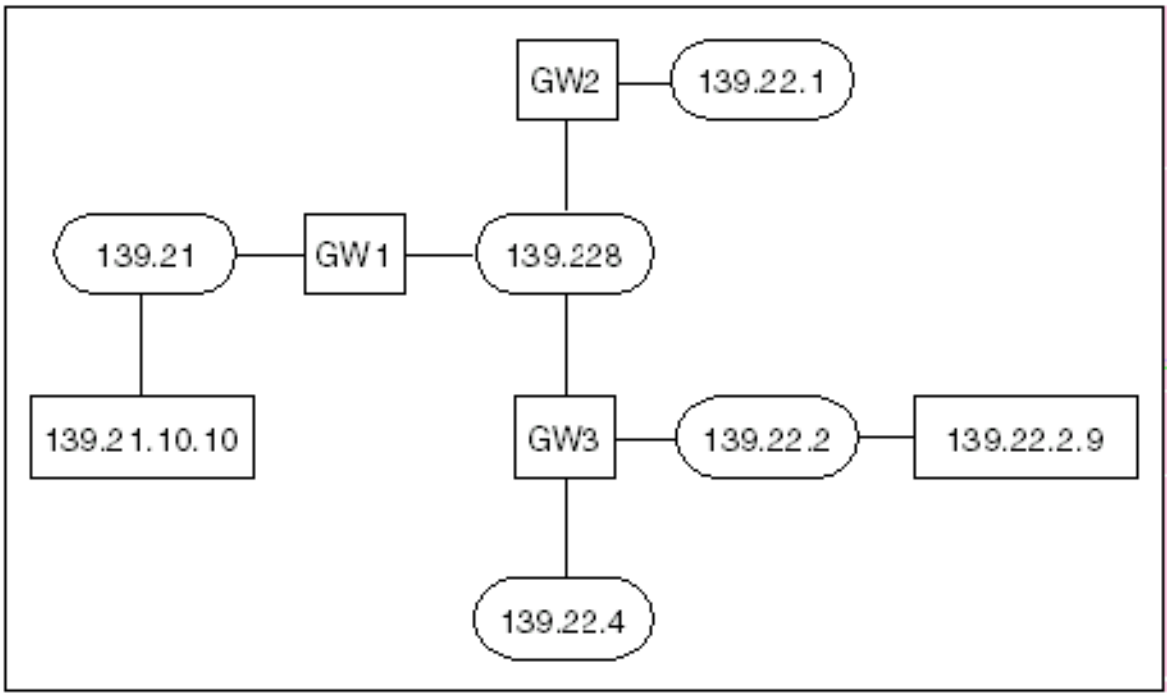


Figure 5: Best match algorithm

Routing table for gateway GW1

139.22.1 GW2
139.22.2 GW3

Let us assume that host 139.21.10.10 wants to communicate with host 139.22.2.9. Basing its search on the network IDs, router GW1 cannot find the correct path to network 139.22.2. It can, however, route the packet to GW2, since the route to {139.22.1, GW2} appears before {139.22.2, GW3} in the route table of the system kernel network. Without subnet routing, the class B network mask 0xffff0000 is the netmask used for searching the routing table. 139.22.2.9 is a class B address.

With the "best match" algorithm, router GW1 finds the correct path, since this algorithm attempts to link as many bits as possible from the destination host ID with the host part of the routing entry.

The disadvantage of this algorithm is that it requires a route entry for each subnet. Hosts in subnet 139.22.4 cannot be accessed with the usual routing table of GW1 (unless GW3 is the gateway of the default route to GW1).

For this reason, the following situation is also not handled correctly:

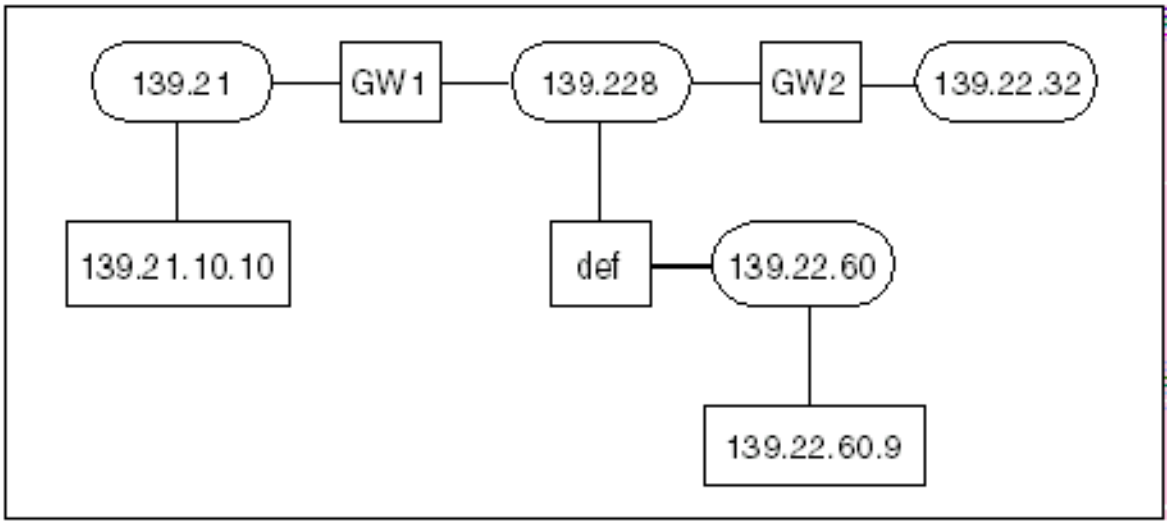


Figure 6: Best match algorithm - restrictions

Routing table for gateway GW1

default def
139.22.32 GW2

When an attempt is made to access host 139.22.60.9 from host 139.21.10.10, GW2 is incorrectly selected as the associated router for network 139.228. The default route would have been the correct choice.

The routing table at GW1, which uses subnet masks, would be as follows:

destination	mask	gateway
default	<unspec>	def
139.22.32	0xfffffc00	GW2

This table and subnet mask match in the system kernel, and are used to select the default route to GW1 when it connects hosts 139.21.10.10 and 139.22.60.9. In this case, subnet routing capability is required.

The routing table contains a field with the subnet mask for the relevant route entry. This subnet mask is referred to during determination of the suitable route.

By default, the "best match" algorithm is activated for the following reasons:

- in order to be compatible with existing installations which use routing entries without subnet masks.
- because *routed*, which is the most commonly used routing daemon, supports only the RIP Version 1 protocol, and this protocol does not use routes with subnet masks.

The "best match" algorithm is sufficient for most installations.

If you are using only static routing (i.e. without routing daemons) and you are experiencing problems with default route entries, you should assign subnet masks to all route entries and deactivate the "best match" algorithm for a system with the "fallback" protocol (see below).

The "best match" algorithm can be activated and deactivated with the BEST_MATCH_ROUTING tuning variable:

BEST_MATCH_ROUTING = 0 to deactivate the "best match" algorithm

BEST_MATCH_ROUTING = 1 to activate the "best match" algorithm

4.2.3 Installing a routing daemon

If a machine is to be used as a dynamic router, a daemon process must be active on it that supports the routing protocol used by the various routers to communicate with each other. In a Reliant UNIX system with the "fallback" protocol, two routing daemons, *routed* and *gated* are supported, and in systems with the

"enhanced" protocol stack, a third daemon, *erouted* is supported.

The *routed* program implements a standard routing protocol, RIP Version 1 (RFC 1058). If a system has only one network interface, the *routed* daemon will passively monitor the routing traffic (if that network is a broadcast network). If a system has two or more network interfaces, the *routed* program will actively participate in the exchange of routing information with other routers.

The *erouted* daemon is an enhanced version of *routed*. *erouted* supports RIP Version 1 (RFC 1058), RIP Version 2 (RFC 1723) and the Internet Router Discovery protocol (RFC 1256), for maintaining the routing table of the system kernel.

gated is a routing daemon that can handle several routing protocols. In Reliant UNIX, *gated* supports the following routing protocols HELLO (RFC 891), RIP Version 1 (RFC 1058), RIP Version 2 (RFC 1723), OSPF (RFC 1845), EGP (RFC 904), BGP (RFC 1265), SLSP and OSPF Version 2 (RFC 1503), RDISC (RFC 1256).

The main difference between *routed* and *gated* lies in the routing protocols that they support. The main differences between *routed* and *erouted* lie in the supported routing protocols and the access mechanism to the routing table.

Regardless of whether the "enhanced" or the "fallback" protocol stack is used on a system, if RIP Version 1 is sufficient for the network, it is recommended that *routed* is used because the protocol is simpler and *routed* is easier to configure (easier than *erouted* and considerably easier than *gated*). If RIP Version 2 is sufficient on a system with the "enhanced" protocol stack, it is recommended, for the same reasons given above, that *erouted* is used instead of *gated*.

Routers manage network traffic by maintaining routing tables - tables that contain information as to which networks and machines can be reached by which routes. A routing table can be either static or dynamic.

If a router is on a simple internetwork - one that consists of two or three local networks, for example - it can manage traffic with static routing tables. It knows how to get to every machine on the internetwork by virtue of being directly connected to the local networks, and no routing information has to be exchanged (e.g. no routing daemon, see the Section "Setting up a static router (no routing daemon)").

On a more complex network - one in which a router connects a local network to other routers and gateways - the router should be configured to use dynamic routing tables. Routers can be active or passive. An active router modifies routing tables in response to current network conditions (load, problems) and forwards them to other routers. Dynamic routing tables allow the router to route traffic to the most current gateway destinations.

Reliant UNIX comes with the routing daemons *gated*, *routed* and *erouted*. Only one of the three daemons can ever be used. As shown below, the values of the GATED and GATEWAY variables in the */etc/default/inet* file determine which of the three is to be used.

The value of the GATED variable takes priority over GATEWAY. This means that *gated* is always started when GATED has the value yes, regardless of the value assigned to GATEWAY.

GATED =	**GATEWAY=**	**Started daemon / system behavior**
yes	Any	*gated*
-	passive	*routed*; the system is passive
-	yes	*routed*; the system is active
-	passive_enhanced	*erouted* for the "enhanced" protocol stack, *routed* for the "fallback" protocol stack the system is passive
-	yes_enhanced	*erouted* for the "enhanced" protocol stack, *routed* for the "fallback" protocol stack the system is active

Table 2: Routing daemon in dependency with the value of GATED or GATEWAY

4.2.3.1 The routed daemon

The routing daemon *routed* creates and maintains dynamic routing tables. It exchanges routing information with gateways and other routers and uses this information to update its routing table.

routed can also run on a machine that is not configured as a router; in this case, however, the communication is one-side, i.e. the daemon listens for broadcasts from other routers and uses them to update its local routing table, but it does not send routing information itself. It runs as a passive *routed*. For information about running *routed* on a client, see the section Setting up router clients.

When *routed* is first initialized on a router, it builds its local routing table using the contents of the file */etc/gateways*. Once it starts to run, it immediately begins to update its local table based on broadcasts from other routers and gateways.

You need to create */etc/gateways*, using any supported text editor. The */etc/gateways* file consists of a series of lines, each in the format

[net|host] *host1* gateway *host2 value* [passive|active]

net|host
: net or host indicates that the route is to a network or to a specific machine, respectively.

host1
: This is the name or address of the destination network, or the name or Internet address of the destination machine, as specified in */etc/inet/networks* or */etc/inet/hosts*.

host2
: This is the name or address of the gateway to which messages intended for the specified network or computer should be forwarded.

value
: This is a number indicating the number of "hops" to the destination machine or network.

passive|active
: passive or active indicates that the gateway should be treated as either passive or active.

To run *routed* on a router automatically whenever the network is started, do the following:

1. Set up the */etc/gateways* file as described above.
2. Open */etc/default/inet* and change the following line:

   GATEWAY=yes

For more information on */etc/gateways* and *routed*, see the description of *routed* in the "Networking Reference Manual".

4.2.3.2 The erouted daemon

The *erouted* daemon is an enhanced version of the *routed* daemon. *erouted* sends and receives Multicast Router Discovery ICMP messages. If the machine is a router, *erouted* supplies copies of the routing tables to all directly connected machines and networks at regular intervals. It also announces or requests standard routes using Router Discovery ICMP messages.

The same applies to *erouted* as to *routed*. Like *routed*, *erouted* first generates its own local routing table from the contents of */etc/gateways* at startup. For example, an */etc/gateways* file set up for *routed* can be read and used by *erouted* without the need for any modification. The same is not necessarily true in reverse, because *erouted* recognizes more keywords in the */etc/gateways* file (which cannot be interpreted by *routed*); such as, for instance, lines that begin with neither host nor net:

if=*ifname*
: Indicates that the other parameters in the line are used for the interface with the name *ifname*.

trust_gateway=*rname*
: Causes RIP packets from this and from other routers that are named in other trust_gateway keywords to be accepted and RIP packets from other routers to be ignored.

redirect_ok
: Causes RIP *erouted* to permit ICMP Redirect messages if the system is functioning as a router and

forwarding packets. Otherwise ICMP Redirect messages are overwritten.

This list is not complete, it is intended only to give an overview of the extended syntax.

An */etc/gateways* file created for the *routed* daemon functions as expected when *erouted* is accessed. This is not the case the other way round, because *routed* ignores lines that it cannot interpret. If *routed* accesses an */etc/gateways* file created for *erouted*, some of the entries may have no effect.

If *erouted* is to be started automatically on a router with the "enhanced" protocol stack when the network is started, you must do the following:

1. Set up the */etc/gateways* file.
2. Open the */etc/default/inet* file and insert the following line:
   GATEWAY=yes_enhanced

If a "fallback" protocol stack is active on the router, *routed* is started, because *erouted* is not supported on a system of this type.

For more information on*/etc/gateways* and *erouted*, see the "Networking Reference Manual".

4.2.3.3 The gated daemon

gated is a routing daemon that handles multiple routing protocols. In Reliant UNIX, *gated* supports the following routing protocols HELLO, RIP Version 1, RIP Version 2, OSPF, OSPF Version 2, EGP, BGP, SLSP and RDISC.

gated can be configured to perform all the routing protocols or any combination of them. The configuration for *gated* is stored by default in the */etc/gated.conf* file.

You will find detailed information on the possible options in the description of *gated* in the "Networking Reference Manual".

Default configuration

gated normally reads configuration information from the */etc/gated.conf* configuration file.

gated uses the following default values:

```
rip on;
hello off;
egp off;
bgp off;
ospf off;
```

In addition, if the configuration file does not exist, there is only one network interface, and a default route is installed in the system kernel, *gated* will exit assuming that a simple default route is adequate.

Numerous options in the configuration file enable you to customize routing. There are options for:

- handling routing protocols
- handling routing information

A more detailed description of these options can be found in the description of *gated* in the "Networking Reference Manual".

Each time it is started, *gated* reads */etc/inet/gated.conf* to see how routing should be handled for the protocol in question.

Notes on configuration options

gated stores its process ID in the */etc/inet/gated.pid* file.

If EGP is being used to distribute the default route (via RIP gateway or HELLO gateway), and if all EGP neighbors have been lost, the default route will not be announced until at least one EGP neighbor is regained.

gated permits so-called routing restrictions to be formulated in its configuration file.

If routing restrictions are used, *gated* logs all invalid networks using *syslog* with log level LOG_WARNING and facility LOG_DAEMON.

Due to the complexity of the network topology and the many "back door" paths to networks, the use of routing restrictions is recommended.

gated internal metrics

gated stores all metrics internally as a time delay in milliseconds to preserve the granularity of HELLO time delays. The internal delay ranges from 0 to 30000 milliseconds with 30000 representing infinity. Metrics from other protocols are translated to and from a time delay as they are received and transmitted. EGP distances are not comparable to HELLO and RIP metrics but are stored internally as a time delay for comparison with other EGP metrics. The conversion factor between EGP distances and time delays is 100.

RIP and the interface metrics are translated to and from the internal time delays with the use of the following translation tables:

Delay			Metric □ RIP value	Metric □ HELLO value	Delay
0	-	0	0	0	0
1	-	100	1	1	100
101	-	148	2	2	148
149	-	219	3	3	219
220	-	325	4	4	325
326	-	481	5	5	481
482	-	713	6	6	713
714	-	1057	7	7	1057
1058	-	1567	8	8	1567
1568	-	2322	9	9	2322
2323	-	3440	10	10	3440
3441	-	5097	11	11	5097
5098	-	7552	12	12	7552
7553	-	11190	13	13	11190
11191	-	16579	14	14	16579
16580	-	24564	15	15	24564
24565	-	30000	16	16	30000

Notes on implementation specifics

All protocols have a two-minute hold-down. When a routing update indicates that the route in use is being deleted, *gated* will not delete the route for two minutes.

Changes can be made to the network interfaces and *gated* will notice them without having to restart the process. If the netmask, subnetmask, broadcast address or interface metric are changed, the interface should be marked down with *ifconfig*, then marked up thirty seconds later at the earliest. Flag changes do not require the interfaces to be brought down and then back up.

RIP is responsible for announcing and monitoring machine routes. The aim here is to enhance the consistency of point-to-point links. The current version also supports the RIP_TRACE commands.

Subnet interfaces are supported. Subnet information will only be propagated on interfaces to other subnets of the same network. For example, if there is a gateway between two class B networks, the subnet routes for

each class B net are not propagated into the other class B net. Only the class B network number is propagated.

gated listens to the routing REDIRECTs (for machines and network routes) and enters them in its own internal tables. This runs parallel to the system kernel's action. Unlike the system kernel, *gated* times out routes learned via a REDIRECT after six minutes. The route is then deleted from the system kernel routing tables. This helps keep the routing tables more consistent. Routes learned via a REDIRECT are not announced by any routing protocol.

The *gated* EGP code verifies that all nets sent and received are valid class A, B or C networks as per the EGP specification. Information about networks that do not meet these criteria is not propagated. If an EGP update packet contains information about a network that is not class A, B or C, the update is considered to be in error and is ignored.

Configuration examples

- RIP

  ```
  rip  on {
  broadcast;
  defaultmetric 1;
  interface lce version 2 multicast;
  query authentication none;
  };
  import proto rip
  {
  all;
  };
  export proto rip
  {
  proto direct
  {
  all;
  };
  };
  ```

  This example configures all lce interfaces for RIP Version 2. The RIP packets are sent as Multicast-IP packets (to the RIP multicast group 224.0.0.9). All routes determined by RIP are imported, and added to the system kernel routing table. All route entries for directly linked interfaces are exported.

- OSPF

  ```
  ospf  on   {      area 0.0.0.2 {
  authtype none;
  networks {
  139.22.228.0 mask 255.255.252.0;
  43.0.0.0 mask 255.0.0.0;
  };
  interface 139.22.228.9 {
  priority 9;
  };
  interface 43.22.4.99 {
  priority 7;
  };
  };
  };
  import proto ospfase
  {
  all;
  ```

```
};
export proto ospfase
{
proto direct
{
all;
};
};
```

One condition of using OSFP is that the relevant LAN interfaces must be configured for multicasting. This can be checked with *ifconfig(1M)*.

The two interfaces with the IP addresses 139.22.228.9 and 43.22.4.99 are configured for sending/receiving OSPF multicast packets. These packets are sent as Multicast-IP packets to the OSPF multicast groups (OSPF uses the three multicast groups 224.0.0.2, 224.0.0.5 and 224.0.0.6). The two supported networks 139.22.228.0 and 43.0.0.0 with the subnet masks are grouped together to form area 0.0.0.2 (a collection of networks and subnets). All routes determined by OSPF are imported and included in the system kernel routing table. All route entries for directly connected interfaces are exported.

Currently known restrictions

- When *gated* takes active part in an internal gateway protocol (RIP, HELLO, OSPF), ICMP rerouting is ignored, even if the rerouting is expressly permitted by the *redirect on* statement. Rerouting cannot be effectuated if it has already been deactivated automatically.
- If multiple interfaces are configured in the same subnet, RIP updates are sent only by the first interface on which the RIP output is configured.
- Only network masks with contiguous single bits (e.g. 0xffff0000) should be used on a system where *gated* has to run.

4.2.4 Setting up router clients

Once a router has been configured, its clients must be made aware of the networks or machines that have just become accessible. You can do this either by issuing the *route* or *eroute* command manually on each client machine, or have it performed automatically by the routing daemons *routed*, *erouted* or *gated* running on the clients.

If the client is linked to an active router, it can use the router's dynamic routing table via a passive local routing daemon. A passive routing daemon automatically collects information from its active neighbors, in doing so updating the routing table of its own system.

If no active router is accessible, or if you do not want to start a routing daemon on the client for some other reason, you will need to enter all the routing information using the *route* (or *eroute*) command or the */etc/inet/routes* file (for more information see the Section "Defining a single default router"). Two cases can be distinguished when entering fixed routes using *route* or *eroute*:

- Only one router exists in the local network.

  In this case, a single standard route entry is made. The client is no longer involved in routing decisions.
- Several routers exist in the local network.

  In this case, you will make a series of fixed route entries.

Entering routes or starting a routing daemon on a machine normally takes place when TCP/IP is initialized. The following sections describe how to configure a client for the various cases.

4.2.4.1 Defining a single default router

If a local network has only one router, all data traffic goes via this router by default.

There are two ways to set up a default router:

1. In the */etc/default/inet* file, enter

   DEFAULTGATEWAY=*name_of_gateway* | *ip_address_of_gateway*

   The system will then run the following command when TCP/IP is started:

   /usr/sbin/route add default *name_of_gateway* | *ip_address_of_gateway* 1
2. In the */etc/inet/routes* file, enter one of the following lines:

   /usr/sbin/route add default *name_of_gateway* | *ip_address_of_gateway* 1

   or

   /usr/sbin/eroute add default *name_of_gateway* | *ip_address_of_gateway* 1

   The system will execute this file when TCP/IP is started.

If you are defining a single default router for the system, you should choose the former procedure. Make an entry in */etc/inet/routes* only if you are including other routes besides the default one (see the Section "Defining several default routers"), or are intending to pass further parameters to the *route* command or are explicitly using *eroute* instead of *route*.

If you configure the default router by entering DEFAULTGATEWAY in */etc/default/inet*, you must not make a default entry in */etc/inet/routes*.

If you decide to use the *eroute* commando instead of the *route* command, remember that *eroute* will no longer function if you eventually decide to switch your system back to using the "fallback" protocol stack.

You can safeguard against this by adding the following lines to the */etc/inet/routes* file:

```
if [ "'test_enhanced'" = "enhanced" ]
then ROUTE=/usr/sbin/eroute
else ROUTE=/usr/sbin/route
fi
```

and then listing the commands:

$ROUTE add default *name_of_gateway* | *ip_address_of_gateway* 1

Example:

Let us assume that the eng and mktg networks are connected via a router. This router has two interfaces (one for each network), each with a different machine name. The machine name of the router for the eng network is sun, and the machine name of the router for the mktg network is sun-moon. This sample configuration is illustrated in the Section "Setting up a static router (no routing daemon)".

If your client machine is connected to the eng network, you will need to modify the */etc/default/inet* file there as follows:

DEFAULTGATEWAY=sun

Alternatively, if the client is linked to the mktg network, the entry in */etc/default/inet* will appear thus:

DEFAULTGATEWAY=sun-moon

In both cases, the client machine will use this router exclusively to communicate with machines on external networks, whatever the case.

4.2.4.2 Defining several default routers

If your machine is a client of more than one router that uses static routing tables, you may want to specify which routers should route traffic to which networks.

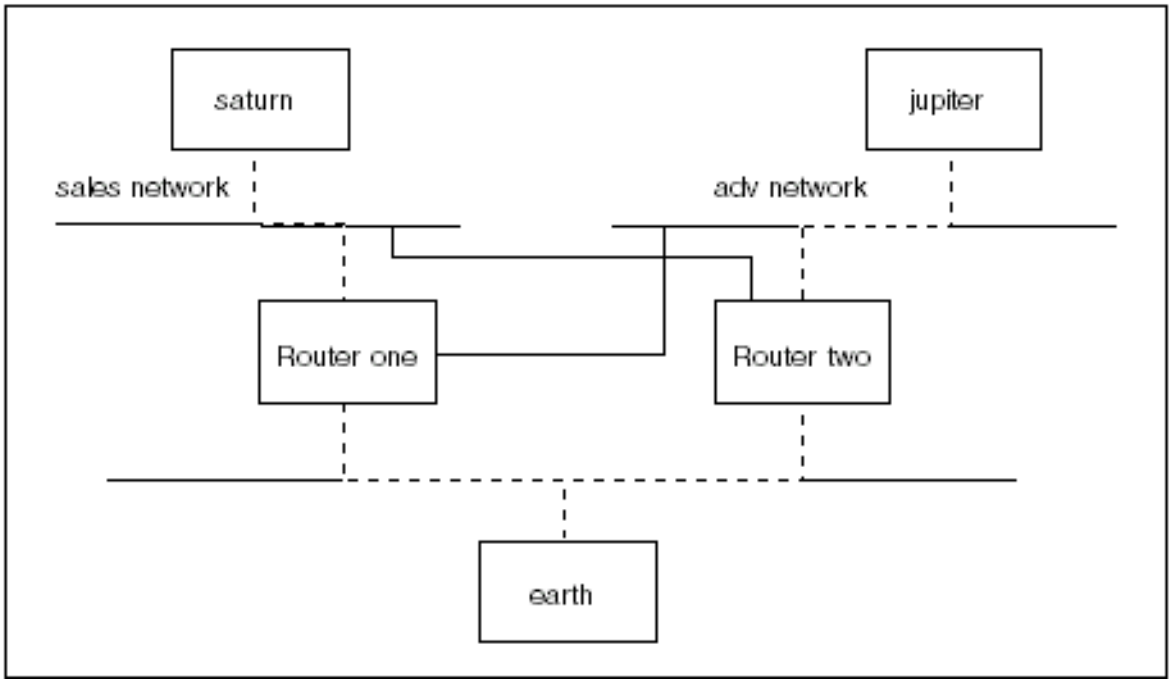


Figure 7: Example of a network configuration with several routers

For example, assume the earth client has two routers, one and two, available to it on the local network (in this example was have used the *route* command. However, you can carry out the same steps using the *eroute* command).

The client system communicates regularly with the networks named sales and adv. You know that router one has the route to the sales network in its routing table, and router two has the route to the adv network in its routing table. To specify that external data traffic of the client should follow these routes (hatched lines in the diagram), you would edit the *route* command in */etc/inet/routes* as follows:

/usr/sbin/route add net sales one 1

Below this statement in */etc/inet/routes*, you would add a second route command, as follows:

/usr/sbin/route add net adv two 1

These commands set up your system so that it sends messages intended for different networks through different routers, rather than sending all traffic through the same router.

A powerful way to set up routing for more complex networks is to provide several specific *route□add* commands for frequently used networks, as well as a *route□add□default* command to handle traffic to all other networks. This is particularly useful when routers, in turn, use other routers.

If the files */etc/default/inet* (entry DEFAULTGATEWAY) and/or */etc/inet/routes* are modified while the network is running, it should be noted that these changes will only take effect once the system has been rebooted. The system administrator can use the *route* command to modify individual routes while the system is running. The old route is deleted using *route□delete* and the new one installed using the command *route□add*.

4.2.4.3 Activating the routing daemon on a client

If your machine is the client of an active router that maintains a dynamic routing table, you can set up your client to update its own tables based on the router's broadcasts.

- *routed* or *erouted*

  If the router is running the *routed* daemon, *routed* or *erouted* listens for broadcasts from other routers and gateways and continually updates its routing table(s) based on the information in those broadcasts. It also broadcasts its own routing table(s) so that other machines can use them for updating their own routing table(s). Because *routed* or *erouted* on a router both broadcasts its own routing table and listens for broadcasts from other routers, it is known as an active *routed* or *erouted*.

  If *routed* or *erouted* is activated on a client (for example, a machine with only one network interface), it listens on the network for messages from the local router, i.e. the internal table is updated each time the local router broadcasts its routing table. Because *routed* or *erouted*, as a client daemon, does not broadcast its routing table, it is called a passive *routed* or *erouted*.

  Even if the local router is using dynamic routing tables, you can still elect not to run *routed* or *erouted* on the client. In this case, the client's tables remain static, even though the router's are dynamic. You might choose not to run *routed* on the client if the client is on a simple internetwork, and the routes to its usual destinations do not change. If the local router is using static routing tables (that is, it is not using *routed* or *erouted*), a daemon does not have to be running on the client.

  Proceed as follows to activate the *routed* routing daemon on a client:

  Change the following line in */etc/default/inet* on the client:

  GATEWAY=passive

  Proceed as follows to activate the *erouted* routing daemon on a client with an "enhanced" protocol stack:

  Change the following line in */etc/default/inet* on the client:

  GATEWAY=passive_enhanced

  If a "fallback" protocol is being used on the system, the *routed* daemon is activated instead of *erouted* (*erouted* is not supported for a system using the "fallback" protocol stack).

- *gated*

  gated can also be used on the client to keep the client's routing table up to date. The *gated* daemon should, in that case, be run passively.

  To do that, enter the following in the configuration files:

  in */etc/default/inet:*

  GATED=yes

  in */etc/gated.conf*:

Set quiet mode here for the relevant protocols, e.g.

```
RIP yes {
nobroadcast;
};
```

4.3 Error analysis and diagnostics

Several commands are available for analyzing routing problems. These enable you to determine, for example, whether routing tables have been set up properly on problematic machines, where errors occur, and via which routes data packets are transferred within your network.

4.3.1 Getting routing information

With the *-r* option of the *netstat* command you call up the contents of the IP routing table. In the example below the *-n* option is also used to suppress the conversion of IP address to network or machine names.

Example for netstat -rn

```
Routing Tables
Destination     Gateway         Flags Refcnt Use   Interface
default         139.22.7.94     UGs   2      40    lce2
10.22           10.22.4.102     UN    0      0     lce3
10.23           10.23.5.102     UN    0      0     lce4
127.0.0.1       127.0.0.1       UH    5      3999  lo0
139.22.5        139.22.7.41     UGs   0      0     lce2
139.22.7        139.22.7.102    UN    3      0     lce2
139.22.32       139.22.32.102   UN    1      0     lcf0
139.22.40       139.22.40.102   UN    0      0     lct1
172.22          172.22.20.102   UN    0      0     zx7
192.168.254.128 192.168.254.102 UN    1      0     zx4
192.168.255     192.168.255.102 UN    0      0     zx6
```

The first column contains the address of the destination network or system and the second column contains the address of the router or the local interface via which the data packets are forwarded to this network or this machine. The entries U, G, H, s and N in the *Flags* column have the following meaning:

U = Route active

G = Route leads to a gateway

H = Route leads to a machine

s = Route is static

N = Route has a subnet mask

The *Refcnt* column specifies how often a route is used, and the *Use* column contains the number of packets sent via this route. The Interface column indicates which interface the route uses.

The *netstat* command produces the same output here for both a system with the "fallback" protocol stack and a system with the "enhanced" protocol stack. The *enetstat -r* command can also be used to take the reorganization of the routing information in a system using the "enhanced" protocol stack into account.

Example for enetstat -rn

```
Routing Tables
Destination     Gateway         Flags  Refcnt Use   Interface
Internet:
1 default       139.22.7.94     UGSc   2      40    lce2
2 10.22         link#12         UC     0      0     lce3
3 10.23         link#14         UC     0      0     lce4
4 127.0.0.1     127.0.0.1       UH     5      3999  lo0
5 139.22.4.13   139.22.7.94     UGHW   1      4503  lce2
```

```
6  139.22.4.40     139.22.7.94      UGHW3   0      17863  lce2
7  139.22.5        139.22.7.41      UGSc    0      0      lce2
8  139.22.7        link#5           UC      3      0      lce2
9  139.22.7.16     8:0:6:b:20:25    UHLW    0      6      lce2
10 139.22.7.25     8:0:6:a:ed:3d    UHLW    0      2      lce2
11 139.22.7.30     0:a0:c9:d0:8e:30 UHLW    0      30     lce2
12 139.22.32       link#11          UC      1      0      lcf0
13 139.22.33.30    0:0:5a:42:65:e3  UHLW    1      1      lcf0
14 139.22.40       link#10          UC      0      0      lct1
15 172.22          link#3           UC      0      0      zx7
16 192.168.254.128 link#1           UC      1      0      zx4
17 192.168.254.158 0:a0:c9:d0:8d:31 UHLW    0      1      zx4
18 192.168.255     link#2           UC      0      0      zx6
```

The output here does **not** correspond to an actual output for the *enetstat -rn* command. To make the following interpretation easier to follow, the output has been assigned line numbers 1 through 18.

The first column contains the address of the destination network or machine. The meaning of the second column depends on the type of route entry involved. The values for *netstat -rn* which are not described in the *Flags* column in the example have the following meaning:

C = Route is used to derive "link-level" routes (cloning)

c = Route is used to derive protocol-specific routes (protocol cloning)

L = Route is an ARP entry (link-level route)

S = Route is static

W = Route was generated through derivation (was-cloned)

3 = Route no longer used and deleted

The lines 1, 2, 3, 4, 7, 8, 12, 14, 15, 16 and 18 correspond to the output for the *netstat -rn* command. It can be noticed that all of these entries have a c or C *Flag*: the system can derive new route entries from these entries. Lines 5, 6, 9, 10, 11, 13 and 17 are route entries that have been derived in this way. The different types of route entries and the derivation mechanisms are described in more detail in the next sections.

Interface route entries

Lines 2, 3, 8, 12, 14, 15, 16 and 18:

An interface route entry describes a sub(network) that can be accessed directly using a local interface. An interface route entry is identified by the link#<*nr*> entry in the *Gateway* column and the value UC in the *Flags* column. Example: The interface route entry in column 8 describes the local network which can be accessed via the lce2 interface; it has been configured here with the following command:
ifconfig lce2 139.22.7.20 netmask 0xffffff00 broadcast 139.22.7.255 up

The network masks linked with the route entries can be displayed using the *-N* option for the *netstat* and the *enetstat* command. For the interface lce2, the output for *enetstat -rnN* would look something like this:
Destination Subnetmask Gateway Flags Refcnt Use Interface

139.22.7 0xffffff00 link#5 UC 3 0 lce2

The value 3 in the *Refcnt* column indicates that 3 additional route entries were derived from this interface route entry; see the note in the section "link-level" route entries.

Static network route entries

Lines 1 and 7:

In the Example for enetstat -rn, lines 1 and 7 denote network route entries that have been integrated into the routing tree by explicitly executing the *route* command. Line 7 describes a route entry for network 139.22.5; this route entry is then always used when a machine in this network is being addressed.

Line 1 describes the default route entry; this is always used when a machine that cannot be accessed using any other route entry is addressed. Both route entries share the value S in the *Flags* column.

The meaning of the value c in the *Flags* column for these route entries is explained in the context of the "protocol cloning" mechanism in the section on "Cloning" route entries The cloning of route entries is an important mechanism in the internal system management of the routing tree. It can, for instance, establish the relationship between a "link-level" route entry and the network interface that it references. In the , the interface route entry for the lce2 interface (line 8) is a route entry from which the "link-level" route entries (lines 9, 10 and 11) are derived. Because a "link-level" entry must be created for each machine which can be accessed directly via one of the local system interfaces, each of the interface route entries is a "cloning" route entry and has the value C in the Flags column of the enetstat output (this does not apply for point-to-point interfaces). A second group of route entries is created using a similar mechanism: "protocol-cloning". A route entry from this group is displayed in the following format using the enetstat -rnN command (line 5, extended with the output for the subnet masks): Destination Subnetmask.

"link-level" route entries

Lines 9, 10, 11, 13 and 17:

The route entries in the routing tree of a system using the "enhanced" protocol stack contain, in addition to the machine and network routes which are familiar from previous system versions, "link-level" route entries. "link-level" route entries integrate the previously used arp cache into the routing tree and enable the conversion of IP addresses to "link-level" addresses. "link-level" route entries are created during execution of the ARP protocol (Address Resolution Protocol) when the "link-level" address for a specified remote system IP address is determined (the remote system is in this case a machine connected directly to the local system networks or a directly accessible IP router). A "link-level" route entry of this kind has the following format in the output for the *enetstat -rn* command:

Destination□□□Gateway□□□□□□□□Flags□□□Refcnt□□□Use□□□Interface

139.22.7.16□□□8:0:6:b:20:25□□UHLW□□□□0□□□□□□□□6□□□□□lce2

A "link-level" route entry is identified by the output of a "link-level" address in the *Gateway* column and the L in the *Flags* column ("link-level" route entry). The W in the *Flags* column also identifies this entry. This W stands for "was-cloned" and means that this route entry was derived from another, i.e. cloned. The value in the *Refcnt* column specifies how many connections currently exist to the specified destination system.

In the Example for enetstat -rn, the "link-level" route entries in lines 9, 10 and 11 have been derived from the interface route entry in line 8. This entry thus has a value of 3 in the *Refcnt* column. The same relationship applies to lines 13 and 12, as well as 17 and 16.

"Cloning" route entries

The cloning of route entries is an important mechanism in the internal system management of the routing tree. It can, for instance, establish the relationship between a "link-level" route entry and the network interface that it references.

In the Example for enetstat -rn, the interface route entry for the lce2 interface (line 8) is a route entry from which the "link-level" route entries (lines 9, 10 and 11) are derived. Because a "link-level" entry must be created for each machine which can be accessed directly via one of the local system interfaces, each of the interface route entries is a "cloning" route entry and has the value C in the *Flags* column of the *enetstat* output (this does not apply for point-to-point interfaces).

A second group of route entries is created using a similar mechanism: "protocol-cloning". A route entry from this group is displayed in the following format using the *enetstat -rnN* command (line 5, extended with the output for the subnet masks):

Destination Subnetmask Gateway Flags Refcnt Use Interface

139.22.4.13 - 139.22.7.94 UGHW 1 4503 lce2

This sample output contains a machine route entry (H in the *Flags* column). The output IP destination address is an address that cannot be accessed directly via one of the local interfaces. To access this address, the

router named in the output (*Gateway* 139.22.7.94) is required (Flag G). In this example, this router is determined using the default route entry, which has the following format in the output for the *enetstat -rnN* command (line 1):

Destination Subnetmask Gateway Flags Refcnt Use Interface

default - 139.22.7.94 UGSc 2 40 lce2

In this sample output you can see a c in the *Flags* column. This flag indicates that this is a "protocol cloning" route entry: if a new TCP or UDP connection that uses this route entry is established to a machine, a new route entry is derived for this machine (cloned) and installed in the local system routing tree. The machine route entry output in the second example is a derived entry of this type. The W flag that describes the derived nature of this entry is set: it has been cloned from the default route entry.

The route entries in lines 5 and 6 have both been derived from the default route entry by means of "protocol cloning". The value 2 in the *Refcnt* column for the default route entry (column 1) also refers to the existence of a route entry derived from the default entry. The value 3 in the *Flags* column in line 6 indicates that there is no TCP or UDP connection between the local system and the machine with the IP address 139.22.4.40 at present. This route entry will be deleted from the routing tree if a TCP or UDP connection to this machine is not re-established within a specific interval. If this is the case, the route entry that still exists for this machine can be re-used (see also the Section "Route metrics and "path MTU discovery"").

Route entries can also be derived from the network route entry in line 7. In the above example this is not the case, so there are therefore no connections to machines in the network 139.22.5.

The "protocol cloning" mechanism is only used in connection with the TCP and UDP transport protocols. Executing a *ping* or *traceroute* on a machine which is not directly accessible does not cause a new route entry to be derived because these programs use ICMP or RAWIP as the transport protocol.

Route metrics and "path MTU discovery"

The use of cloning enables the system to collect important data required for describing a connection to a remote system, i.e. route metrics. These metrics can be re-used when setting up a connection to a remote system to which there is already an existing connection. Derived route entries remain in existence for a longer period than a connection. This means that information previously gathered for a connection to a remote system can be used again for a new connection establishment, provided there is not already an active connection to the remote system.

One of the important parameters for a TCP connection to a remote system is to mss size (maximum segment size) to be used. This value defines the maximum data packet length for the sending of data via a TCP connection. This value has considerable influence on the data transfer rate that can be achieved with this TCP connection. It is assumed that the fragmenting of data packets on the IP layer should be avoided wherever possible.

If a remote system cannot be accessed via one of the local system interfaces, but instead only via additional IP networks, the value for the maximum data packet length (mss size) may be different from the mtu value (maximum transfer unit) specified for the local interface. The mtu value is the maximum packet size determined by the hardware (see also the Section "MTU (maximum transmission unit)"); it determines when the IP layer must split a data packet from a higher protocol layer (e.g. TCP) into several parts, in other words, fragment it. This is the case when IP networks used for accessing the remote system are governed by various physical constraints. To determine an optimum mss size from the point of view of the TCP in such a situation, the "path MTU discovery" procedure is implemented. If this procedure was executed successfully for the connection to a remote system, the result achieved is stored in the route entry derived for this connection. This result can then be made available for other connections being set up to this remote system.

The metrics determined for a route entry can be displayed using the *eroute get* command and specifying a destination address; they would look something like this for the "protocol cloned" machine route entry above:
eroute get 139.22.4.40

□□□□□route to:□□139.22.4.40

□□destination:□□139.22.4.40

□□□□□□gateway:□□139.22.7.94

□□□□interface:□□lce2

□□interf addr:□□139.22.7.20

□□□□□□□□flags:□□<UP,GATEWAY,HOST,DONE,WASCLONED>

□□recvpipe□sendpipe□ssthresh□rtt,msec□rttvar□hopcount□mtu□expire

□□32768□□□□32768□□□□0□□□□□□□□□0□□□□□□□□□0□□□□□□□0□□□□□□□□1500□3592

In this example, the mtu size for determining the mss size of a TCP connection is 1500 bytes. This size matches the mtu size of the lce2 interface (Ethernet) in this case. That means that all networks on the route to machine 139.22.4.40 can handle IP packets of this size without having to fragment them.

For further information on "path MTU discovery", see the Section "Routing metrics".

Displaying routing statistics

Combining the *-s* option with *netstat* or *enetstat* produces routing statistics. You can use the resulting display to determine whether problems occurred on your network during routing. Here is a sample of output from *netstat -s*

routing:
0 bad routing redirects
0 dynamically created routes
0 new gateways due to redirects
2 destinations found unreachable
2330 uses of a wildcard route

If the output indicates that bad routing redirects occurred or that a number of destinations were found unreachable, further measures (e.g. modification of the routing table) may be necessary.

Selective Net Routes

Using the enhanced protocol stack and its IP routing mechanism, it is possible to configure more than one network interface in the same IP network. Keywords in this context include "interface address group" and "selective net route".

Example

Let us look at a system with two Ethernet network interfaces, i.e. *zx0* and *zx1*. These are to be configured with the following commands:

(1) # ifconfig zx0 192.168.10.10 netmask 0xffffff00 up
(2) # ifconfig zx1 192.168.10.11 netmask 0xffffff00 up
(3) # ifconfig zx1-1 alias zx1 192.168.10.12 netmask 0xffffff00 up

An interface address group is a set of IP addresses (primary and aliases), which have been configured for a network interface and refer to the same IP network in terms of network masks. In the above example, lines (2) and (3) form an interface address group.

A selective net route is an entry in the routing tree with a reference not only to an individual IP address (and thus also a network interface) but also to a list of such IP addresses. The "selective" feature of such a route entry is then required if these IP addresses do not belong to a single interface address group. On every occasion that such a selective net route is cloned, i. e. when a new node route entry is derived from the selective net route, because the destination node is located in the IP network described by the selective net route, a decision must be made as to which IP address will be connected with the derived route and to which network interface the node route entry is to refer.

In the above example, the route entry for the network 192.168.10.0 is a selective net route: it references three IP addresses from two different interface address groups.

The IP address linked with a route entry is displayed using the *enetstat* command and the new *-v* option. For the above example, this produces:

```
# enetstat -rnv
Destination   Gateway       Flags   Refcnt   Use   Interface
192.168.10    link#1        UCs     0        0     zx0
---ifa        192.168.10.10 [PRI]   0        0     zx0
---ifa        192.168.10.11 [PRI]   0        0     zx1
---ifa        192.168.10.12 [SEC]   0        0     zx1
```


The "s" in the flags stands for "selective".

The standard method for selecting an IP address and thus a network interface is to use the first address from the list. In the above example, therefore, 192.168.10.10 and thus the network interface *zx0* would always be used.

Another possibility here is to use a "round robin" system; when a node route entry is derived from a selective net route, it is assigned the first IP address from the next interface address group relative to that from the last derive operation performed. In the above example, this means alternate use of the IP addresses 192.168.10.10 (*zx0*) and 192.168.10.11 (*zx1*). In this round robin system, it is possible to achieve a type of load balancing not in relation to individual connections but in relation to the derivation of new node route entries.


Interface address groups for which the UP flag is not set for the relevant network interface are not considered in the round robin system.

The *eroute* command has been extended by the *get2* sub-command in order to display the IP address list for a selective net route. For the above example, this produces:

```
# eroute get2 192.168.10.0
□ route to: 192.168.10.0
destination: 192.168.10.0
□ □ □ mask: 255.255.255.0
□interface: zx0
interf addr: 192.168.10.10
□ □ □ □ifa: 192.168.10.10 <zx0 cnt=0, factor=0, type=PRI>
□ □ □ □ifa: 192.168.10.11 <zx0 cnt=0, factor=0, type=PRI>
□ □ □ □ifa: 192.168.10.12 <zx0 cnt=0, factor=0, type=SEC>
□ □ □flags: <UP, DONE, CLONING, SELECT>
recvpipe sendpipe ssthresh rtt/msec rttvar hopcount mtu □ expire
32768 □ □32768 □ □0 □ □ □ □0 □ □ □ □0 □ □ □0 □ □ □ □1500 □-887
```

A further enhancement to IP routing is the use of an additional routing tree for internal (local) routes: an entry is created in this routing tree for each network interface configured.

To output the internal routing tree, you have to use the new*-L* option with the *enetstat* command:

```
# enetstat -rnL
Routing Tables:
Destination □ □ Gateway □ □ □ □ Flags □ Refcnt □ Use □ Interface
Internet (internal routes):
192.168.10.10 □ 192.168.10.10 □ UH □ □ □0 □ □ □ □0 □ □ lo0
192.168.10.11 □ 192.168.10.11 □ UH □ □ □0 □ □ □ □0 □ □ lo0
192.168.10.12 □ 192.168.10.12 □ UH □ □ □0 □ □ □ □0 □ □ lo0
```

Configuration options for use of selective net routes

From the point of view of IP, there are two different classes of network interface to which the selective net routes can be applied:

arp class
: e. g. Ethernet. These network interfaces do not have the NOARP bit set in the *ifconfig* output.

nonarp class
: e. g. ATM in classical IP mode. These network interfaces have the NOARP bit set in the *ifconfig* output.

The use of selective net routes can be configured using the *sysctl* command and the two variables *net.inet.ip.arp_select_ifgroup* for network interfaces from the *arp* class and *net.inet.ip.nonarp_select_ifgroup* for network interfaces from the *nonarp*-class.

Possible values:

-1
: Disable selective net routes. This is the standard behavior for 4.4BSD and prevents the configuration of more than one network interface in a single IP network.

0
: Enable selective net routes. The selection algorithm states that the first interface address group will always be used for a cloning operation. The behavior of a system with the fallback protocol stack is adjusted in this way.

1
: Enable selective net routes and use the round-robin system described above for selecting an IP address in a cloning operation.

The default for both variables is 0.

4.3.2 Logging routing problems

If you suspect a *routed* daemon malfunction, you may log its actions - and even all the packet transfers. To create a log file of routing daemon actions, just supply a file name when you start up the *routed* daemon, for example:

/usr/sbin/in.routed /var/routerlog

You get the same functionality when you start the *erouted* daemon by specifying the *-T* option, for example:

/usr/sbin/in.erouted -T /var/routerlog

 Logging is carried out almost all the time on a very busy network.

Whenever a route is added, deleted, or modified, a log of the action and a history of the previous packets sent and received will be printed in the log file. To specify full packet tracing, specify the *-t* option when calling the *routed* daemon.

When the *erouted* is used, the *-t* option increases the debug level. This results in more information being written to the trace log file specified with the *-T* option. The debug level can be increased or decreased using the SIGUSR1 or SIGUSR2 signals or the *rtquery* command.

For more information on *rtquery*, refer to the "Networking Reference Manual".

4.3.3 Querying the route packets take to their target machine

Tracking the route one's packets follow to different machines (or finding the troublesome gateway that is discarding your packets) can be difficult. The *traceroute* program evaluates the *ttl* (time-to-live) field in the IP protocol header and attempts to elicit an ICMP TIME_EXCEEDED response from each gateway along the path to a certain machine.

The only mandatory parameter is the name or Internet address of the destination machine. The default probe datagram length is 38 bytes, but this may be increased by specifying a packet size (in bytes) after the destination machine name.

You can specify a maximum lifespan (maximum number of hops) for outgoing probes. The default is 30 hops (the same as the default for TCP connections).

Example

A sample command line and output might be:

```
[yak 71]% traceroute nis.nsf.net.
traceroute to nis.nsf.net (35.1.1.48), 30 hops max, 56 byte packet
1 helios.ee.lbl.gov (128.3.112.1) 19 ms 19 ms 0 ms
2 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
3 lilac-dmc.Berkeley.EDU (128.32.216.1) 39 ms 39 ms 19 ms
4 ccngw-ner-cc.Berkeley.EDU (128.32.136.23) 39 ms 40 ms 39 ms
5 ccn-nerif22.Berkeley.EDU (128.32.168.22) 39 ms 39 ms 39 ms
6 128.32.197.4 (128.32.197.4) 40 ms 59 ms 59 ms
7 131.119.2.5 (131.119.2.5) 59 ms 59 ms 59 ms
8 129.140.70.13 (129.140.70.13) 99 ms 99 ms 80 ms
9 129.140.71.6 (129.140.71.6) 139 ms 239 ms 319 ms
10 129.140.81.7 (129.140.81.7) 220 ms 199 ms 199 ms
11 nic.merit.edu (35.1.1.48) 239 ms 239 ms 239 ms
12 * * *
13 128.121.54.72 (128.121.54.72) 259 ms 499 ms 279 ms
14 * * *
15 * * *
16 * * *
17 * * *
18 ALLSPICE.LCS.MIT.EDU (18.26.0.115) 339 ms 279 ms 279 ms
```

You will find more information on *traceroute* in the "Networking Reference Manual".

This program is intended for use in network testing, measurement and management. It should be used primarily for manual fault isolation.

It is unwise to use *traceroute* during normal operation due to the load it could place on the network.

5 Using the Domain Name Service

The Domain Name System (DNS) is an application layer protocol that is part of the standard TCP/IP protocol suite. Specifically, the DNS is a "naming service", i.e. its obtains information about the machines in a network and makes it available over the network.

The Domain Name System performs naming between machines within your local administrative domain and across domain boundaries. It is distributed among a set of servers, commonly known as name servers, each of which implements DNS by running a daemon called *named*.

The *named* daemon is also called the Berkeley Internet Name Domain service, or BIND, because it was developed at the University of California at Berkeley.

On the client's side, DNS is implemented through the resolver. The resolver is neither a daemon nor a particular program, rather it is linked into applications in the form of library routines. The resolver's function is to resolve users' queries; to do that, it queries a name server, which then returns either the requested information or a referral to another server.

DNS is dependent on the hierarchical structure of the Internet and adds to that structure the concept of domain zones. This chapter describes the hierarchical structure of the Internet and the function of the Domain Name System and then tells you how to set up and maintain DNS on your network machines.

5.1 The domain hierarchy and DNS administrative zones

Domain names used by the DNS reflect the hierarchical organization of the public networks. Before you set up the DNS for your organization, you should understand the organization of the network, your place in the overall structure of the network, and how domain names reflect that structure.

For information about the structure of the Internet and domain naming conventions, see the section entitled Establishing a DNS domain.

The Domain Name System introduces the concept of zones to the Internet network model. A zone is a hierarchical community of machines, administered by a single authority and served by a group of authorized name servers. The members of a zone include individual machines as well as name servers and their clients (subzones) nested beneath the authorized name servers for the zone. Zone boundaries are usually the same as the boundaries of administrative domains, i.e. a zone could, for example, correspond to a local administrative domain. However, a zone is not limited to just one administrative domain. A zone is a domain, plus all or some of the domains under it.

The next figure shows how some fictional zones might be delegated. The illustration shows four zones: the root zone, served by the Internet root domain name servers, and three zones served by name servers of the universe.DE domain. The dotted lines point to the borders making up each zone.

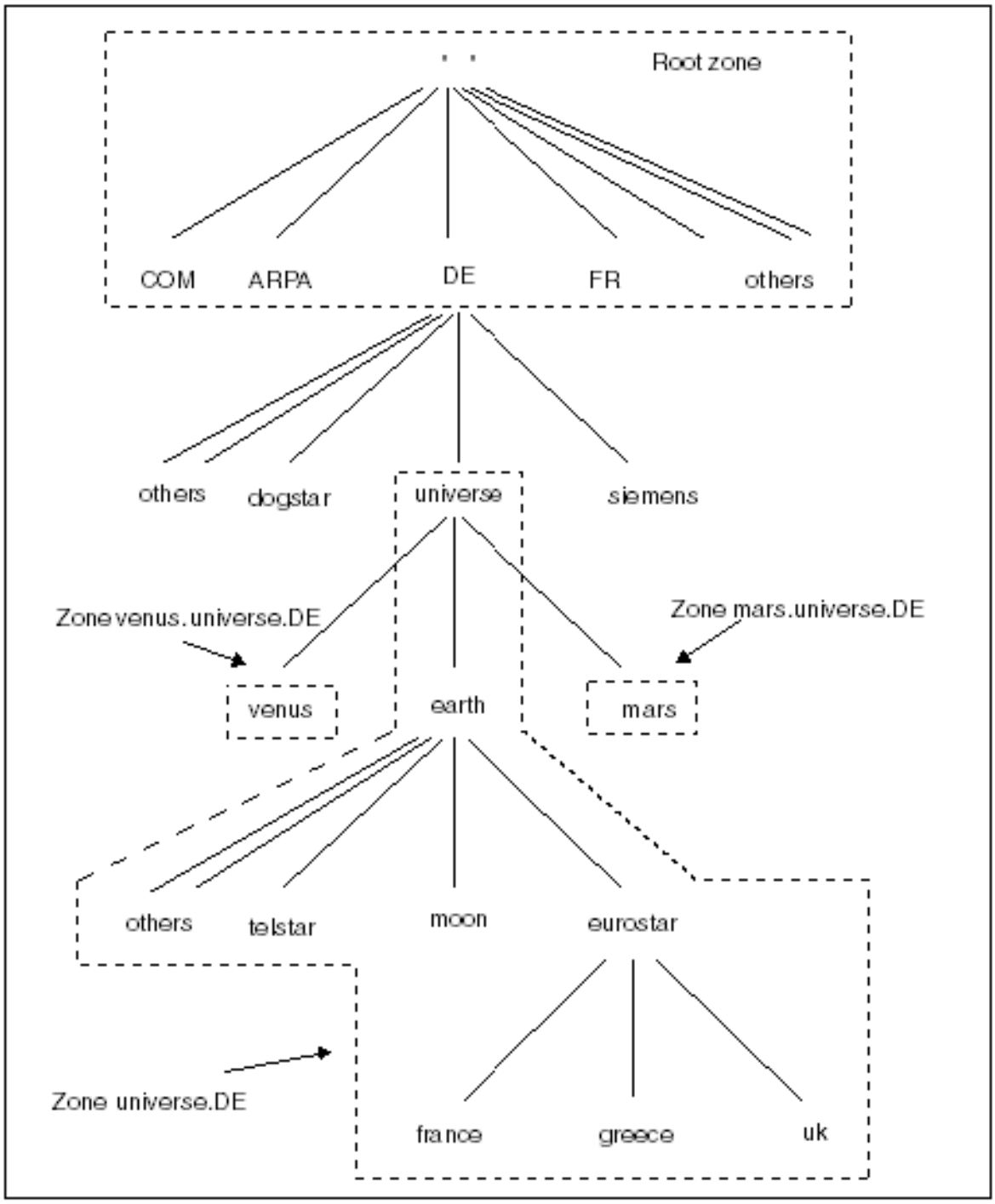


Figure 8: Administrative zones

Zones take their names from the label of the domain at the top of the zone hierarchy. In Figure "Administrative zones", the names of the four domains within dotted lines are:

universe.DE venus.universe.DE mars.universe.DE .(the root zone)

Zone universe.DE takes its name from the label of the second level domain universe.DE. However, zone universe.DE does not have the same administrative authority as the domain of the same name. Zone universe.DE consists only of

- domain universe

- local administrative domain earth
- subdomain eurostar.

The zone universe.DE does not include domains venus and mars. They have their own separately administered zones, venus.universe.DE and mars.universe.DE respectively, but these are part of the universe.DE domain hierarchy.

The domain hierarchy and name space described so far keeps track of information by machine name. This enables the *named* daemon to perform name-to-address mapping.

In addition, there is a special domain supported by the DNS called IN-ADDR.ARPA, which simplifies address-to-name mapping. IN-ADDR.ARPA contains essentially the same information as the machine name space, but it is expressed in terms of Internet addresses.

A name in the IN-ADDR.ARPA domain has four labels preceding it, corresponding to the four octets of an Internet address. This machine address is listed from right to left. For example, a machine whose Internet address is 128.32.0.4 has the following domain name IN-ADDR.ARPA.

4.0.32.128.IN-ADDR.ARPA.

Therefore, if *named* knows the Internet address of a machine, it can find out the machine's fully qualified domain name by consulting a file representing the IN-ADDR.ARPA domain. (This file, commonly called *hosts.rev*, is explained in a later section.)

5.1.1 The administrator's role

Administering the DNS involves not only running the appropriate programs on the servers and clients, but also determining domain names, answering complaints, and filling out registration and other forms, should you join a public network. This section contains overall procedures for setting up the DNS, based on the responsibilities you might have as a DNS administrator.

Your administrative responsibilities for the DNS depend on your domain's position within the overall network hierarchy. For example, managing one set of name servers in a small administrative domain entails less responsibility than managing the authoritative set for a large zone. Responsibilities depend on whether you are the chief authority for a domain or zone, or an administrator reporting to the chief authority.

The NIC divides administrators on the Internet into domain administrators, who have primary responsibility for a domain, and technical contacts, who work with domain administrators to maintain a zone. Descriptions of each position appear below.

5.1.1.1 The domain administrator

The domain administrator (DA) is a coordinator, manager, and technician for a second-level or lower domain. The DA's responsibilities include:

- Registering the domain.

  If your domain is going to be on a public network, you must register it (if you have not done so already). The domain must have a unique name within its level of the network hierarchy. To register your domain with the Internet, obtain the Domain Registration Form from NIC and complete it according to the instructions in Appendix A of this manual. To join other public networks, contact the organization in charge of the network and request the appropriate domain registration form.

  To link up to multiple networks such as the Internet, BITNET, CSNET, etc., you need to register with only one network, as domain names are independent of a network's physical topology.

- Naming machines and verifying that names within the domain are unique.

  At many sites, users name their individual machines while the administrators name the servers. The administrator should ensure that there are no duplicate names within a zone.

- Understanding the functioning of the name servers and making sure the data is current at all times.

  This includes either setting up the DNS-related files and programs, or delegating this authority to other technically competent people (such as the technical contact). This chapter should provide you with the basic information you need for understanding the DNS. However, as the domain administrator, you should gain more specific technical knowledge of your particular network. Contact the public network to which you belong and ask for technical papers regarding it.

5.1.1.2 The technical contact

The primary responsibility of a technical contact is to maintain DNS programs and files for a zone and to keep the zone's name servers running. Technical contacts interact with their Domain Administrators and with DA's of other domains to solve network problems.

5.2 DNS clients and servers

As mentioned earlier, there are two sides to DNS: name servers running the *named* daemon, and clients running the resolver.

5.2.1 Clients

A name server running *named* can also run the resolver; therefore, there can be two kinds of clients

- client-only
- client/server

A client-only machine does not run the *named* daemon. If need be, it consults the resolver, which provides a

list of possible name servers to which queries should be directed.

A client/server is a machine that uses the domain name service provided by the local daemon *named* in order to resolve a user's queries. However, if there is either no daemon or a hanging daemon, the resolver might be able to solve queries on a temporary basis by directing them to remote name servers.

5.2.2 Servers

You implement the DNS for a zone not on a single server, but on a set of servers. This set must include two master servers, and may or may not include other servers.

- Master servers

  The master name servers maintain all the data corresponding to the zone, making them the authority for that zone. These are commonly called "authoritative" name servers. The data corresponding to any given zone must be available on at least **two** authoritative servers. You should designate one name server as the primary master server and at least one as a secondary master server, to act as a backup if the primary is unavailable or overloaded.

  Changes to zone data always take place on the "primary" master server. This server loads the master copy of its data from disk when it starts up *named*. The primary server may also delegate authority to other servers in its zone, as well as to servers outside of it.

  The "secondary" master server is a name server that maintains a copy of the data for the zone. When the secondary server boots *named*, it requests all the data for the given zone from the primary. The secondary server then periodically checks with the primary to see if it needs to update its data.

  A server may function as a master for multiple zones: as a primary for some zones, and as a secondary for others.

- Caching and caching-only servers

  All name servers are caching servers. This means that the name server caches received information until the data expires. (The expiration process is regulated by the time to live field attached to the data when it is received from another server.)

  Additionally, you can set up a caching-only server. This server does not have any authority itself, i.e. it handles queries but does not administer any authoritative data, instead it requests the required information from other name servers who have the authority for the information needed.

- Forwarding servers

  You can optionally set up servers that handle requests by forwarding them to other servers. There may be one or more servers in the forwarding list, and they are tried in turn until the list is exhausted. A forwarding server can also be forced to refrain from resolving queries locally by designating it as a "slave" (a subordinate server).

  A typical scenario in which you may wish to use this scheme is if you have a large machine connected to the Internet or some other network and a series of smaller machines or workstations with no connection to the outside world. In order to give the workstations the appearance of access to the Internet or some other network, you would set up the small machines as forwarding slaves of the larger one, and their requests would be forwarded to it, which in turn would interact with other servers to resolve the query and return the answer.

  An added advantage is that the cache of the system to which the queries are forwarded would be able to build a very rich cache of data compared to the cache a typical workstation would build, thereby reducing the number of queries from the site to the rest of the network.

5.2.3 Configuring a DNS client

Setting up the resolver on a client involves two tasks: creating the file *resolv.conf*, and modifying the file *netconfig*.

5.2.3.1 Creating the resolv.conf file

The domain name server uses several files to load its database. At the resolver level, it needs a file (called */etc/resolv.conf*) listing the addresses of the servers where it can obtain the information needed. Whenever the resolver has to find the address of a machine (or the name corresponding to an address), it sends a query package to the name servers stored in */etc/resolv.conf*. The servers either answer the query locally or use the services of other servers and return the answer to the resolver.

The *resolv.conf* file is read by the resolver to find out the name of the local domain and the Internet addresses of the name servers. It sets the local domain name and instructs the resolver routines to query the listed name servers for information. For every DNS client on your network, you must create a *resolv.conf* file in the machine's */etc* directory.

The first line of the file lists the domain name in the form

domain *domain_name*

where *domain_name* is the name registered with the NIC. Succeeding lines list the Internet addresses of the servers that the resolver should consult to resolve queries. Internet address entries have the form:

nameserver *IP_address*

Below is a sample *resolv.conf* file:

```
; Sample resolv.conf file
domain univxy.Edu
nameserver 128.32.0.4
nameserver 128.32.0.10
```

For the remaining configuration parameters, see *resolv.conf(4)*.

5.2.3.2 Modifying the netconfig file

Once you have created the appropriate */etc/resolv.conf* file on each machine that is to run the resolver, you need to edit the */etc/netconfig* file. Look at the file and make sure that all transport protocols listed in the */etc/netconfig* file that use IP have the resolver library (usually with the name *resolv.so*) responsible for name-to-address mapping via the DNS listed in the *lookups* field. For example, if your */etc/netconfig* file lists the following entries:

```
#trans_id semantics    flag fam.  prot. device    lookups
#
udp      tpi_clts     vb   inet  udp   /dev/udp  /usr/lib/tcpip.so
tcp      tpi_cots_ord v    inet  tcp   /dev/tcp  /usr/lib/tcpip.so
```

you will have to modify the *lookups* field so it says:

```
#trans_id ......   lookups
#
udp       ......   /usr/lib/tcpip.so,/usr/lib/resolv.so
tcp       ......   /usr/lib/tcpip.so,/usr/lib/resolv.so
```

Do not modify the other entries in the file.

In single user mode, *named* is not available, so the */etc/netconfig* file **must** include a resolver library that uses local text files (this library, *tcpip.so* in the example above, is the default). You should therefore put the local system's interface addresses in */etc/inet/hosts*.

5.2.4 Configuring a DNS name server

To set up the DNS on a server, you need to create the configuration and data files that *named* will need.

5.2.4.1 Creating the configuration and data files

In addition to the daemon *named*, the DNS on a name server must also have a configuration file and local data files. The default location of the configuration file is */etc/named.boot*. Common names for the local data files are *named.ca*, *named.local*, *hosts*, and *hosts.rev*. In the descriptions of these files that follow, these names are used. However, you can name these files whatever you wish. In fact, it is suggested that you use names other than the ones used in this guide, to avoid confusion with files with similar names.

- *named.boot*

  The configuration file *named.boot* establishes the server as a primary, a secondary, a caching-only, or a forwarding name server. It also specifies the zones over which the server has authority, and which data files it should read to get its initial data.

  The configuration file is read and evaluated by *named* when the daemon is started. It directs *named* either to other servers or to local data files for a specified domain.

- *named.ca*

  named.ca contains information about the Internet's root servers. It establishes the names of the root servers and lists their addresses. *named* cycles through the list of servers until it contacts one of them. It then obtains from that server the current list of root servers, which it uses to update *named.ca*. To avoid confusion with the caching process, with which this has very little to do, you may want to call your file something like *named.root* or *boot.root*.

- *hosts*

  The *hosts* file contains all the data about the machines in the local zone. The name of this file is specified in the configuration file. To avoid confusion with */etc/inet/hosts*, name the file something other than *hosts*. You may want to give the file name the suffix *zone;* in the following illustration, the file is called *A.xyzone*.

- *hosts.rev*

  The *hosts.rev* file specifies a zone in the IN-ADDR.ARPA domain (the special domain that allows inverse mapping of machine addresses to names). The name of this file is specified in the configuration file. In the illustration, the file is called *xyhosts.rev*.

- *named.local*

  The *named.local* file specifies the address for the local loopback interface, or localhost, with the network address 127.0.0.1. The name of this file is specified in the configuration file. As with all other files, it can be called something other than the name used in this guide.

The relationship between the configuration file and the data files (or, alternatively, other servers) is illustrated by the following figure:

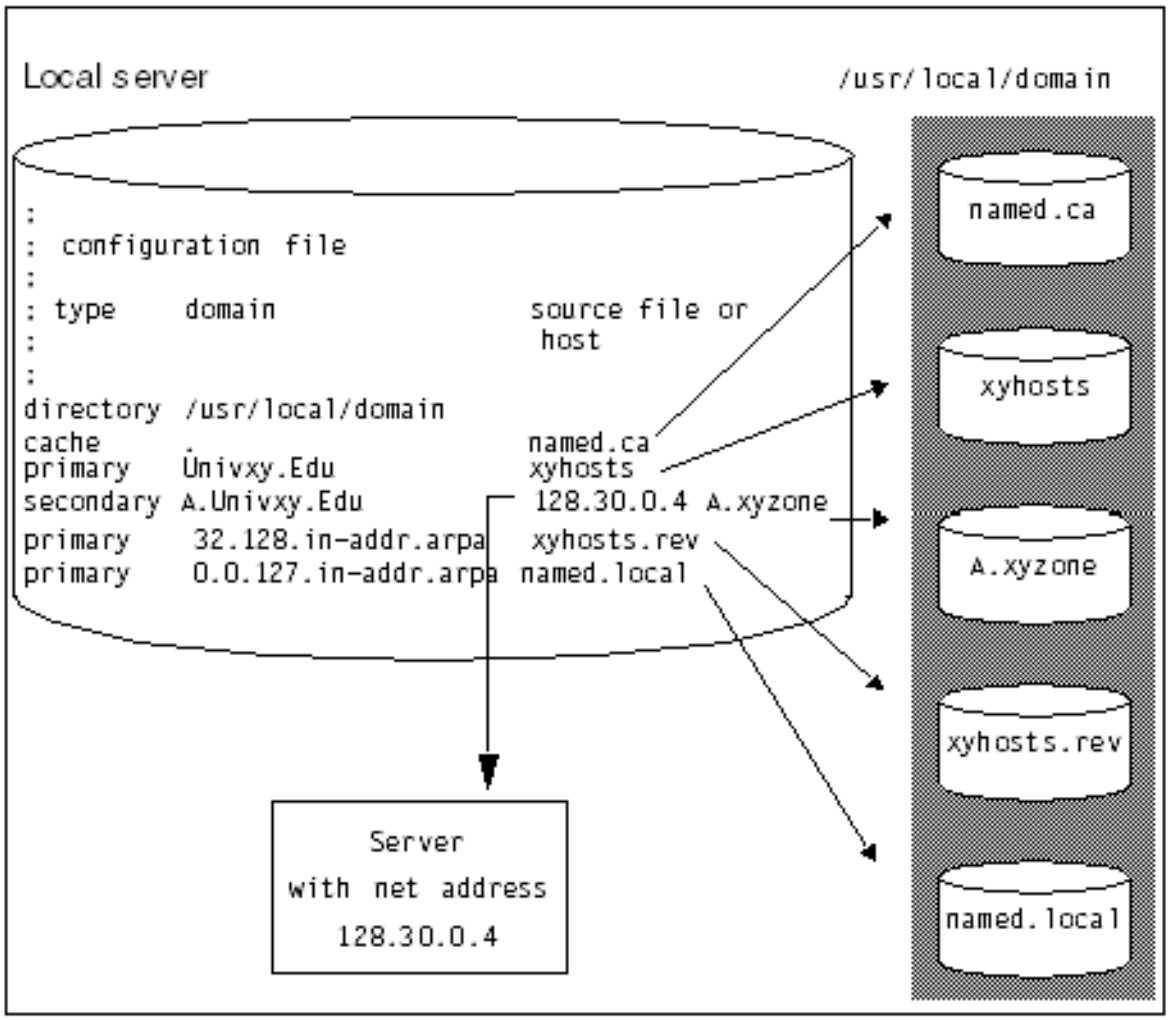


Figure 9: Configuration file and data files

Setting up the configuration file

The contents of the configuration file vary, depending on the type of server. This section describes configuration files for primary and secondary master servers, caching-only servers, and forwarding servers.

Configuration file for a primary master server

The following is a sample configuration file for a primary master server:

```
;
; Sample named.boot file for primary master server
```

```
;
; Type    Domain              Source file or machine
;
directory  /usr/local/domain
cache      .                   named.ca
primary    Univxy.Edu          xyhosts
primary    32.128.in-addr.arpa    xyhosts.rev
primary    0.0.127.in-addr.arpa   named.local
```

The entries in the file are explained below.

directory

The following line in the configuration file designates the directory containing the file names:
directory /usr/local/domain

This means that you can specify all the following files with relative path names or call them in a system file with the $INCLUDE directive. It is especially useful if you have many files to be maintained and you want to locate them all in one directory dedicated to that purpose.

If there is no directory line in the configuration file, all file names listed in it must be full pathnames.

cache

A name server needs to know which servers are the authoritative name servers for the root zone. In this line, you have to list the files which contain the addresses of these higher authorities.

All servers should have the following line in the configuration file to find the root name servers:
cache . named.ca

The second field indicates that the server will obtain root servers hints from the file indicated in the third field (in this case *named.ca* located in the directory */usr/local/domain*).

primary

The following line in the configuration file designates the server as a primary server for a zone:
primary Univxy.Edu xyhosts

The first field designates the server as primary for the zone stated in the second field. The third field is the name of the file from which the data is read. The lines
primary 32.128.in-addr.arpa xyhosts.rev

primary 0.0.127.in-addr.arpa named.local

indicate that the server is also a primary server for the domains 32.128.in-addr.arpa (that is, the reverse address domain for Univxy.Edu) and 0.0.127.in-addr.arpa (that is, the local machine loopback). The data for them is read from the files *xyhosts.rev* and *named.local* respectively.

Configuration file for a secondary master server

The following is a sample configuration file for a secondary server in the same domain as the above primary server:

```
;
; Sample named.boot file for a secondary master server
;
; Type     Domain          Source file or machine
;
directory   /usr/local/domain
cache       .                 named.ca
secondary   Univxy.Edu        128.32.0.4 128.32.0.10 \
128.32.136.22 xyhosts.zone
secondary   32.128.in-addr.arpa   128.32.0.4 128.32.0.10 \
128.32.136.22 xyrev.zone
primary     0.0.127.in-addr.arpa named.local
```

In appearance, this file is very similar to the configuration file for the primary server; the main difference is to

be found in the lines

```
secondary Univxy.Edu          128.32.0.4 128.32.0.10 \
128.32.136.22 xyhosts.zone
secondary 32.128.in-addr.arpa  128.32.0.4 128.32.0.10 \
128.32.136.22 xyrev.zone
```

The word secondary establishes that this is a secondary server for the zone listed in the second field, and that it is to get its data from the listed servers; attempts are made in the order in which the servers are listed. The listed servers are normally the primary server, plus one or more additional secondary servers. If there is a filename after the list of servers (as in the example above), data for the zone will be put into that file as a backup. When the server is started, data are loaded from the backup file, if it exists, and then one of the servers is consulted to check whether the data is still up to date.

This ability to specify multiple secondary addresses allows for great flexibility in backing up a zone.

The second secondary entry configures the server as a secondary server for the domain 32.128.in-addr.arpa.

The primary entry defines the server as the primary server for the domain 0.0.127.in-addr.arpa. This primary entry is necessary, even though the server is configured as a secondary server for the other domains.

A server may act as the primary server for one or more zones, and as the secondary server for one or more zones; it is the mixture of entries in the configuration file that determines it.

Configuration file for a caching-only server

The following is a sample configuration file for a caching-only server:

```
;
; Sample named.boot file for a caching-only server
;
; Type   Domain                Source file or machine
;
cache    .                     named.ca
primary  0.0.127.in-addr.arpa  named.local
```

You do not need a special line to designate a server as a caching only server. What denotes a caching-only server is the absence of authority lines, such as secondary or primary in the configuration file. As explained above, a caching-only server does not maintain any authoritative data; it simply handles queries and asks the machines listed in the *named.ca* file for the information needed.

Configuration file for a forwarding server

The following is a sample configuration file for a forwarding server:

```
;
; Sample named.boot file for Forwarding Server
;
forwarders 128.32.0.10 128.32.0.4
options forward only
```

forwarders

If you do not specify options forward only, any queries which the local name server is unable to answer will first be forwarded to the servers specified after the keyword forwarders and then, if necessary, directed to other servers.

options forward only

If you do specify options forward only, any queries which the local name server is unable to answer will be forwarded only to those machines in the forwarders list.

If you specify options forward only you must also include the line forwarders.

The parameter options forward only supersedes the parameter slave which was previously used, although slave is still supported for the sake of compatibility.

Additional options, such as "non-recursive queries" and "logging" are described in the "Networking Reference Manual" under *named(1M)*.

5.2.4.2 Setting up the data files

All the data files used by the DNS daemon *named* are written in Standard Resource Record Format. In Standard Resource Record Format, each line of a file is a record, called a Resource Record (RR). Each DNS data file must contain certain Resource Records. This section describes the DNS data files, and the Resource Records each file should contain. Following this section is a discussion of the Standard Resource Record Format, including an explanation of each Resource Record relevant to the DNS data files.

The hosts file

The *hosts* file contains all the data about all the machines in your zone, including server names, addresses, machine information (hardware and operating system information), canonical names and aliases, the services supported by a particular protocol at a specific address, and group and user information related to mail services. This information is represented in the records NS, A, HINFO, CNAME, WKS, MX, MB, MR, MG. The file also includes an SOA resource record, which indicates the start of a zone and includes the name of the machine on which the *hosts* data file resides.

Here is a sample *hosts* file:

```
; sample hosts file
@       IN    SOA    ourfox.Sample.Edu. kjd.monet.Sample.Edu. (
1.1       ; Serial
10800     ; Refresh
1800      ; Retry
3600000   ; Expire
86400 )   ; Minimum
IN    NS     ourarpa.Sample.Edu.
IN    NS     ourfox.Sample.Edu.
ourarpa  IN    A      128.32.0.4
IN    A      10.0.0.78
IN    HINFO  3B2 UNIX
arpa     IN    CNAME  ourarpa
ernie    IN    A      128.32.0.6
IN    HINFO  3B2 UNIX
ourernie IN    CNAME  ernie
monet    IN    A      128.32.7
IN    A      128.32.130.6
IN    HINFO  Sun-4/110 UNIX
ourmonet IN    CNAME  monet
ourfox   IN    A      10.2.0.78
IN    A      128.32.0.10
IN    HINFO  RM600 SINIX
IN    WKS    128.32.0.10 UDP syslog route timed domain
IN    WKS    128.32.0.10 TCP ( echo telnet
discard rpc sftp
uucp-path systat daytime
netstat qotd nntp
link chargen ftp
auth time whois mtp
pop rje finger smtp
supdup hostnames
domain
nameserver )
```

```
fox     IN   CNAME  ourfox
toybox  IN   A      128.32.131.119
IN   HINFO  3B2 UNIX
toybox  IN   MX     0  monet.Sample.Edu
miriam  IN   MB     vineyd.Neighbor.COM.
postmiss IN   MR     miriam
bind    IN   MINFO  bind-request kjd.Sample.Edu.
IN   MG     ralph.Sample.Edu.
IN   MG     zhou.Sample.Edu.
IN   MG     painter.Sample.Edu..
IN   MG     riggle.Sample.Edu.
IN   MG     terry.pa.Xerox.Com.
```

The named.local file

The *named.local* file contains the DNS entry for the local loopback interface of your name server. It needs to contain the name of the machine, plus a pointer to the machine name localhost, which represents the loopback mechanism. The server name is indicated in the NS resource record, and the pointer to localhost by the PTR record. The file also needs to include an SOA record, which indicates the start of a zone and includes the name of the machine on which the *named.local* data file reside.

Here is a sample *named.local* file:

```
; sample named.local file
@   IN  SOA    ourhost.Univxy.Edu. kjd.monet.Univxy.Edu. (
1        ; Serial
3600     ; Refresh
300      ; Retry
3600000  ; Expire
3600 )   ; Minimum
IN  NS     ourhost.Univxy.Edu.
1   IN  PTR    localhost.
```

The hosts.rev file

hosts.rev is the file that sets up inverse mapping. It needs to contain the names of the primary and secondary name servers in your local domain, plus pointers to these and other non-authoritative name servers. The names of the primary and secondary master servers are indicated by NS records, and the pointers by PTR records. The file also needs an SOA record to indicate the start of a zone and the name of the machine on which *hosts.rev* resides.

Here is a sample *hosts.rev* file:

```
; sample hosts.rev file
@    IN  SOA    ourhost.Univxy.Edu. kjd.monet.Univxy.Edu. (
1.1      ; Serial
3600     ; Refresh
300      ; Retry
3600000  ; Expire
3600 )   ; Minimum
IN  NS     ourarpa.Univxy.Edu.
IN  NS     ourhost.Univxy.Edu.
4.0  IN  PTR    ourarpa.Univxy.Edu.
6.0  IN  PTR    ernie.Univxy.Edu.
7.0  IN  PTR    monet.Univxy.Edu.
10.0 IN  PTR    ourhost.Univxy.Edu.
6.130 IN  PTR    monet.Univxy.Edu.
```

The named.ca file

The *named.ca* file contains the names and addresses of the root servers. Server names are indicated in the record NS and addresses in the record A. You need to add an NS record and an A record for each root server you want to include in the file.

The following is a sample *named.ca* file:

```
;
;Root domain servers
;
; list of servers ...
.               99999999   IN     NS   NS.NASA.GOV.
99999999   IN     NS   NIC.DDN.MIL.
99999999   IN     NS   A.ISI.EDU.
99999999   IN     NS   TERP.UMD.EDU.
99999999   IN     NS   C.NYSER.NET.
99999999   IN     NS   BRL-AOS.ARPA.
99999999   IN     NS   GUNTER-ADAM.ARPA.
; ...and their addresses
NIC.DDN.MIL.       99999999   IN    A    10.0.0.51
NIC.DDN.MIL.       99999999   IN    A    26.0.0.73
C.NYSER.NET.       99999999   IN    A    192.33.4.12
BRL-AOS.ARPA.      99999999   IN    A    128.20.1.2
BRL-AOS.ARPA.      99999999   IN    A    192.5.22.82
NS.NASA.GOV.       99999999   IN    A    128.102.16.10
TERP.UMD.EDU.      99999999   IN    A    10.1.0.17
A.ISI.EDU.         99999999   IN    A    26.3.0.103
GUNTER-ADAM.ARPA.  99999999   IN    A    26.1.0.13
```

You can obtain the names and addresses of the root servers from the NIC (Network Information Center); use, for instance, *telnet* to fetch the information from the machine RS.INTERNIC.NET. The system administrator should get the list updated at regular intervals. You may find it useful to get yourself put on the mailing list of BIND or NAMEDROPPER for this purpose.

5.2.4.3 The SSR format

In the Standard Resource Record Format, each line of a data file is a record called a Resource Record (RR), containing the following fields separated by white space:

[*name*] [*ttl*] *class record_type record_specific_data*

The order of the fields is always the same; however, the first two are optional (as indicated by the square

brackets), and the contents of the last vary according to the *Record Type* field.

name The first field is the name of the domain that applies to the record. If this field is left blank in a given RR, it defaults to the domain name given in the previous RR.

A domain name in a zone file can be either a fully qualified name, terminated with a dot, or a relative one, in which case the name of the current domain is appended to it.

ttl The second field specifies the time-to-live (*ttl* = time-to-live) of the data, i.e. the amount of time in seconds after which the data cached in the database will be declared invalid and new information requested from a server. By leaving this field blank, the *ttl* defaults to the minimum time specified in the Start Of Authority resource record discussed below.

If the *ttl* value is set too low, the server will incur in a lot of repeat requests for data refreshment; if, on the other hand, the *ttl* value is set too high, changes in the information will not be timely distributed.

Most *ttl*s should be initially set to between a day (86400) and a week (604800); then, depending on the frequency of actual change of the information, change the appropriate *ttl*s to reflect that frequency. Also, if you have some *ttl*s that have very high values because you know they relate to data that rarely changes, and you know that the data is now about to change, reset the *ttl* to a low value (3600 to 86400) until the change takes place, and then change it back to the original high value.

All RR's with the same name, class and type should have the same *ttl*.

class The third field is the record class. Only one class is currently in use: IN for the TCP/IP Internet protocol family.

record_type

The fourth field states the type of the resource record. There are many types of RR's; the most commonly used types are discussed in the section entitled Resource record types.

record_specific_data

The content of this field depends on the value entered for Record type.

Although case is preserved in names and data fields when loaded into the name server, all comparisons and lookups in the name server database are case insensitive. However, this situation may change in the future, and you are advised to be consistent in your use of lower and upper case.

Special characters

The following characters have special meanings:

. A free standing dot in the name field refers to the current domain.

@ A free standing @ in the name field denotes the default setting of the name field. The name of the domain as entered in configuration file named.boot is used as the default. This default can be changed using the $ORIGIN statement.

.. Two free standing dots in the name field represent the null domain name of the root.

X where *X* is an arbitrary character (other than one of the digits 0 to 9). This indicates that the special significance of the character does not apply in this case.

DDD

To ensure that a decimal number is interpreted as text and not checked for special meaning, you must prefix the digits (*DDD*) with a slash.

( ) Use parentheses to group data that exceeds a line. In this case, the end of the line is not considered to be the end of the record. Parentheses are not to be used within record-specific data (e.g. for HINFO), as some non-Siemens name servers configured as secondary servers cannot handle them.

; Semicolon starts a comment; the remainder of the line is ignored.

* An asterisk signifies wildcarding.

Most resource records have the current origin appended to names if they are not terminated by a ".". This is useful for appending the current domain name to the data, such as machine names, but may cause problems where you do not want this to happen. A good rule of thumb is to use a fully qualified name ending in a period if the name is not in the domain for which you are creating the data file.

Control entries

The only possible lines that do not conform to the standard RR format in a data file are control entry lines. There are two kinds of control entries:

$INCLUDE

An include line begins with $INCLUDE in column 1, and is followed by a file name. This feature is particularly useful for separating different types of data into multiple files and, where necessary, including it in a DNS system file, for example:
$INCLUDE etc/named/data/mailboxes

The line is interpreted as a request to load the file */etc/named/data/mailboxes* at that point. The $INCLUDE command does not cause data to be loaded into a different zone or tree. This is simply a way to allow data for a given zone to be organized in separate files. For example, mailbox data might be kept separately from host data using this mechanism.

$ORIGIN

The $ORIGIN command is a way of changing the origin in a data file. The line starts in column 1, and the word ORIGIN must be followed by a domain name. Hence, the specified domain becomes the default domain for this file. The default setting for the name field is used to create fully specified domain names from relative domain names. This is useful for specifying more than one domain in a data file.

Resource record types

The following are some of the most commonly used types of RRs:

Type	Description
SOA	Zone definition (Start of Authority)
NS	Name Server
A	Internet Address
CNAME	Alternative Name (alias)
HINFO	Host Information
WKS	Well Known Services
PTR	Complete Name
MX	Mail Exchanger

The following is an example of a *hosts* file. It is presented here for illustration purposes only. We will shortly analyze it in full.

```
;    sample hosts file
@       IN    SOA    ourfox.Sample.Edu. kjd.monet.Sample.Edu. (
1.1       ; Serial
10800     ; Refresh
1800      ; Retry
3600000   ; Expire
86400 )   ; Minimum
IN    NS     ourarpa.Sample.Edu.
IN    NS     ourfox.Sample.Edu.
ourarpa  IN    A      128.32.0.4
IN    A      10.0.0.78
IN    HINFO  3B2 UNIX
arpa     IN    CNAME  ourarpa
ernie    IN    A      128.32.0.6
IN    HINFO  3B2 UNIX
ourernie IN    CNAME  ernie
monet    IN    A      128.32.7
IN    A      128.32.130.6
IN    HINFO  Sun-4/110 UNIX
ourmonet IN    CNAME  monet
```

```
ourfox   IN    A      10.2.0.78
IN    A      128.32.0.10
IN    HINFO  RM600 SINIX
IN    WKS    128.32.0.10 UDP syslog route timed domain
IN    WKS    128.32.0.10 TCP ( echo telnet
discard rpc sftp
uucp-path systat daytime
netstat qotd nntp
link chargen ftp
auth time whois mtp
pop rje finger smtp
supdup hostnames
domain
nameserver )
fox      IN    CNAME  ourfox
toybox   IN    A      128.32.131.119
IN    HINFO  3B2 UNIX
toybox   IN    MX     0  monet.Sample.Edu
miriam   IN    MB     vineyd.Neighbor.COM.
postmiss IN    MR     miriam
bind     IN    MINFO  bind-request kjd.Sample.Edu.
IN    MG     ralph.Sample.Edu.
IN    MG     zhou.Sample.Edu.
IN    MG     painter.Sample.Edu.
IN    MG     riggle.Sample.Edu.
IN    MG     terry.pa.Xerox.Com.
```

SOA - Start Of Authority

The following is the format of a Start of Authority resource record:

name [ttl] class SOA origin person in charge (
serial
refresh
retry
expire
minimum)

The Start of Authority (SOA) record designates the start of a zone. The zone ends at the beginning of the next zone, i.e. if a new SOA record is specified.

name indicates the name of the zone. In the example below, @ indicates the current zone or origin.

class is the RR class. This is currently always IN for the Internet protocol class.

SOA is the type of this Resource Record.

origin is the name of the machine on which this data file resides.

person in charge
is the mailing address for the person responsible for the name server.

serial is the version number of this data file.You **must** increment this number whenever you make a change to the data: secondary servers use the *Serial* field to detect whether the data file has been changed since the last time they copied the file from the master server.

Note that the name server can handle no more than four positions after the decimal point.

refresh
indicates how often, in seconds, a secondary name server should check with the primary name server to see if an update is needed.

retry indicates how long, in seconds, a secondary server is to retry after a failure to check for a refresh.

expire is the upper limit, in seconds, that a secondary name server is to use the data before it expires for lack of getting a refresh.

minimum
is the default number of seconds to be used for the time to live field on resource records that don't have a *ttl* specified.

There should only be one SOA record per zone.

The following is a sample SOA resource record:

```
;name [ttl]  class  SOA     origin              person in charge
@           IN     SOA     ourfox.Sample.Edu. kjd.monet.Sample.Edu.(
1.1    ;Serial
10800  ;Refresh
3600   ;Retry
432000 ;Expire
86400) ;Minimum
```

NS - Name Server

The following is the format of an NS resource record:

[*name*] [*ttl*] *class NS Name-server name*

The Name Server record (NS) defines a name server responsible for a given domain. The name of the domain is entered in the name field. If no name field is listed, then it defaults to the last name listed. One NS record should exist for each primary master server for the domain. The following is a sample NS resource record:

```
;[name]  [ttl]  class   NS   server name
IN      NS   ourarpa.Sample.Edu.
```

A - Address

The following is the format of an A resource record

[*name*] [*ttl*] class A *adress*

The Address record (A) lists the address for a given machine. The *name* field is the machine name, and the *address* is the Internet address. One A record should exist for each address of the machine (in other words, gateways should be listed several times, one record for each interface address).

```
;[name]  [ttl]  class   A   address
ourarpa         IN      A   128.32.0.4
IN      A   10.0.0.78
```

HINFO - Host Information

The following is the format of a HINFO resource record:

[*name*] [*ttl*] *class* HINFO *Hardware OS*

It lists the hardware and operating system that are running at the listed machine. Note that if you want to include a space in the machine name, you must surround the name in quotes. The name field specifies the name of the machine. If no name is specified it defaults to the last named machine. One HINFO record should exist for each machine. The following is a sample HINFO resource record:

```
;[name]  [ttl]  class   HINFO  Hardware OS
IN      HINFO  RM600    SINIX
```

WKS - Well Known Services

The following is the format of a WKS resource record:

[*name*] [*ttl*] *class* WKS *address protocol list of services*

The Well Known Services record (WKS) describes the well known services supported by a particular protocol at a specified address. The list of services and port numbers come from the list of services specified in the *services* database. Only one WKS record should exist per protocol per address. The following is an example of a WKS resource record:

```
;[name] [ttl] class  WKS address     protocol  list of services
IN     WKS 128.32.0.10 UDP       (who route timed
domain)
IN     WKS 128.32.0.10 TCP       (echo telnet
discard rpc sftp
uucp-path systat
daytime netstat
qotd nntp link
chargen ftp auth
time whots mtp
pop rje finger
smtp supdup
hostnames domain
nameserver)
```

CNAME - Canonical Name

The format of a CNAME resource record is:

alias [ttl] class CNAME *Canonical name*

The Canonical Name resource record (CNAME) specifies a alias for a canonical name. A alias should be unique. All other resource records should be associated with the canonical name and not with the alias. Do not create a alias and then use it in other resource records. Aliases are particularly useful during a transition period, when a machine's name has changed but you want to permit people using the old name to reach the machine. The following is a sample CNAME resource record:

```
;alias          [ttl]  class   CNAME  canonical name
ourmonet                IN      CNAME  monet
```

PTR - Domain Name Pointer

The following is the format for a PTR resource record:

special name [ttl] class PTR *real name*

In pointer records (PTR), special names are defined as pointers to other names in the domain. PTR's are used mainly in the IN-ADDR.ARPA records for the translation of an address (the special name) to a machine name. PTR names should be unique to the zone. The PTR records below set up reverse pointers for the special IN-ADDR.ARPA domain.

```
;special name          [ttl]  class PTR   real name
7.0                    IN     PTR   monet.Univxy.Edu.
2.2.18.128.in-addr.arpa        IN     PTR   blah.junk.COM.
```

MX - Mail Exchanger

The following is the format for an MX resource record:

name [ttl] class MX *preference value mailer exchanger*

The Mail Exchanger (MX) resource records are used to specify a mailer exchanger, i.e. a system that can deliver mail to a domain or to a specific machine in a domain. There may be more than one MX resource record for a given name. In the example below, Seismo.CSS.GOV. (note the fully qualified domain name) is a mail gateway that knows how to deliver mail to Munnari.OZ.AU. Other machines on the network cannot deliver mail directly to Munnari. Seismo and Munnari may have a private connection or use a different transport medium. The *preference value* field indicates the order a mailer should follow when there is more than one way to deliver mail to a single machine. The value 0 (zero) indicates the highest preference. If there is more than one MX resource record for the same name, they may or may not have the same preference value.

You can use names with the wildcard asterisk (*) for mail routing with MX records. There are likely to be servers on the network that simply state that any mail to a domain is to be routed through a gateway. In the example below, all mail to machines in domain foo.COM is routed through RELAY.CS.NET. You do this by creating a wildcard resource record, which states that the mail exchanger for *.foo.COM is RELAY.CS.NET. Note that the route will apply to any machine or subdomain of foo.COM, but not to foo.COM itself.

```
;name          [ttl] class  MX pref. value  mailer exchanger
Munnari.OZ.AU.       IN     MX 0            Seismo.CSS.GOV.
foo.COM.             IN     MX 10           RELAY.CS.NET.
*.foo.COM.           IN     MX 20           RELAY.CS.NET.
```

5.2.5 Starting named

In SINIX, you can have *named* run automatically when the system is booted. This is possible if the */etc/named.boot* file exists and has been defined as a valid configuration file.

Do not attempt to run *named* from *inetd*. This will continuously restart the name server and defeat the purpose of the cache buffer.

5.2.6 Modifying the data files

When you add or delete a machine in one of the data files in the master DNS server, or modify otherwise the data files, you must also change the Serial number in the SOA resource record so the secondary servers modify their data accordingly; you should then inform *named* in the master server that it should re-read the data files and update its internal database, as explained below.

When *named* successfully starts up, it writes its process ID to the */etc/named.pid* file. To have *named* reread *named.boot* and reload the database, enter

```
# kill -HUP 'cat /etc/named.pid'
```

Note that all previously cached data is lost, and the caching process starts over again.

5.2.7 Debugging named

From time to time, you may need to debug *named*. You can do this by sending it signals through the *kill(1)* utility. Depending on the signal received *named* will change its behavior.

To establish whether *named* is working with the current database version, enter

kill -INT 'cat /etc/named.pid'

Upon receiving this signal, *named* dumps the current database and cache to */var/tmp/named_dump.db*. This should give you an indication of whether the database was loaded correctly.

When *named* is running incorrectly, you can also look in */var/adm/messages* and check for any messages logged by *syslog*. For instance, if there is a data file that lists a hostname as a alias and also has other data under that name instead of the machine's canonical name, you may see a message similar to:

May 4 02:35:26 hostname named[4804]: hazy.widget.junk.COM has CNAME and other data (illegal)

Or, if there is a problem with the database, you may see

May 1 11:02:33 hostname named[17808]: /etc/named/junk.zone: line 759: database format error ()

To turn on debugging, you can start *named* with the *-d* option, or you can enter, if *named* is already running,

kill -USR1 'cat /etc/named.pid'

(or use *named.debug(1M)*). Each subsequent USR1 increments the debug level. The output goes to */var/tmp/named.run*.

To turn off debugging completely, enter

kill -USR2 'cat /etc/named.pid'

Finally, *nslookup(1M)*, *dig(1M)*, *host(1M)* and *dnsquery(1M)* are provided as useful tools for investigating the function of name servers and those of the resolver. Please refer to the command description given in the "Networking Reference Manual".

5.3 A practical example

We can now start building up the files that an imaginary network would need. Let's assume that our network is composed of three Class C subnets:

name	**number**
junk	223.100.100
widget	223.100.101
zap	223.100.102

The names of the zones are also the names of the machines that we are designating as the master servers.

Our imaginary network can therefore be represented by the following figure:

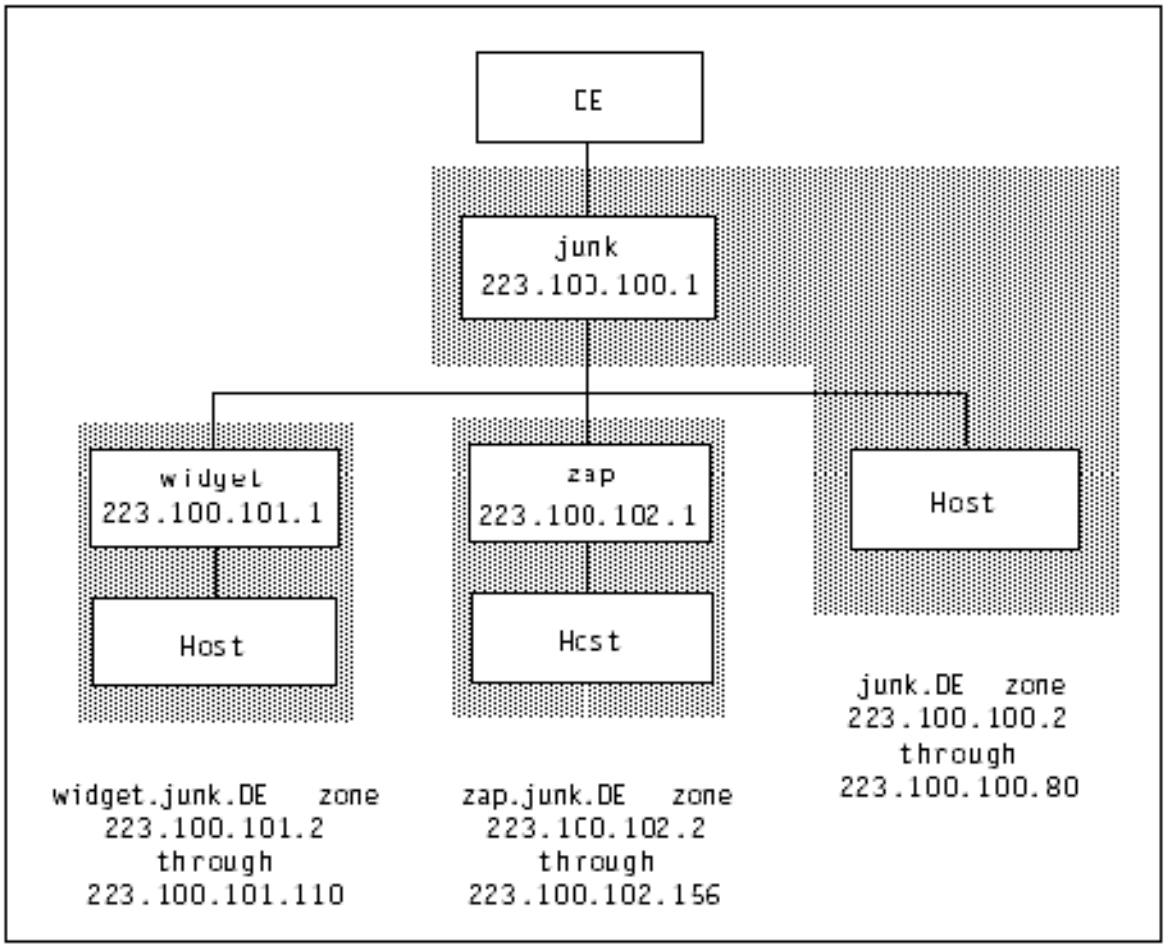


Figure 10: The sample network

Let's further assume that after careful consideration we decide that we want to set up the Domain Name System in our network so that each master server is the primary server for its zone and a secondary server for the other zones. All this can be summed up by the following tables:

junk zone		
Host name	Function	Address
junk	Primary	223.100.100.1
widget	Secondary	223.100.101.1
zap	Secondary	223.100.102.1
	Host	223.100.100.2-80

widget zone		
Host name	Function	Address
widget	Primary	223.100.101.1
junk	Secondary	223.100.100.1
zap	Secondary	223.100.102.1
	Host	223.100.101.2-110

zap zone		
Host name	Function	Address
zap	Primary	223.100.102.1
junk	Secondary	223.100.100.1
widget	Secondary	223.100.101.1
	Host	223.100.102.2-156

The following are the configuration files for the three servers in the network:

```
;       Configuration file for server junk
directory   /usr/local/domain
cache       .                       named.root
primary     junk.DE                 junk.zone
primary     100.100.223.in-addr.arpa junk.revzone
primary     0.0.127.in-addr.arpa    named.local
secondary   widget.junk.DE          223.100.101.1 223.100.102.1
                                    widget.zone
secondary   zap.junk.DE             223.100.101.1 223.100.102.1 zap.zone
secondary   101.100.223.in-addr.arpa 223.100.101.1 widget.revzone
secondary   102.100.223.in-addr.arpa 223.100.102.1 zap.revzone
```

```
; Configuration file for server widget
directory    /usr/local/domain
cache        .                          named.root
primary      widget.junk.DE             widget.zone
primary      101.100.223.in-addr.arpa   widget.revzone
primary      0.0.127.in-addr.arpa       named.local
secondary    junk.DE                    223.100.100.1 223.100.102.1 junk.zone
secondary    zap.junk.DE                223.100.100.1 223.100.102.1 zap.zone
secondary    100.100.223.in-addr.arpa   223.100.100.1 junk.revzone
secondary    102.100.223.in-addr.arpa   223.100.102.1 zap.revzone

;
; Configuration file for zap
directory    /usr/local/domain
cache        .                          named.root
primary      zap.junk.DE                zap.zone
primary      102.100.223.in-addr.arpa   zap.revzone
primary      0.0.127.in-addr.arpa       named.local
secondary    junk.DE                    223.100.100.1 223.100.102.1 junk.zone
secondary    widget.junk.DE             223.100.100.1 223.100.101.1
                                        widget.zone
secondary    100.100.223.in-addr.arpa   223.100.100.1 junk.revzone
secondary    101.100.223.in-addr.arpa   223.100.101.1 widget.revzone
```

The following are some sample *resolv.conf* files. If *named* is not running on a machine, the local machine address should not be specified.

```
;
; resolv.conf for server junk
;
domain     junk.DE
nameserver 127.0.0.1
nameserver 223.100.101.1
nameserver 223.100.102.1
;
; resolv.conf for a host in zone junk not running named
;
domain     junk.DE
nameserver 223.100.100.1
nameserver 223.100.101.1
nameserver 223.100.102.1
;
; resolv.conf for a host in zone widget.junk not running named
;
domain      widget.junk.DE
nameserver  223.100.101.1 ; mast. server for zone widget.junk.DE
nameserver  223.100.100.1
nameserver  223.100.102.1
;
; resolv.conf for a host in zone zap.junk not running named
;
domain      zap.junk.DE
nameserver  223.100.102.1 ; master server for zone zap.junk.DE
nameserver  223.100.100.1
nameserver  223.100.101.1
```

The data files for the local machines on the individual servers might appear as follows:

```
;
; named.local for server junk
;
@   IN  SOA     junk.DE.        ralph.sysad.zap.junk.DE. (
1.1     ;Serial
10800   ;Refresh
3600    ;Retry
432000  ;Expire
86400)  ;Minimum
IN  NS      junk.DE.
1   IN  PTR     localhost.
;
; named.local for server widget
;
```

```
@  IN  SOA     widget.junk.DE.    ralph.sysad.zap.junk.DE. (
1.1     ;Serial
10800   ;Refresh
3600    ;Retry
432000  ;Expire
86400)  ;Minimum
IN  NS      widget.junk.DE.
1   IN  PTR     localhost.
;
; named.local for server zap
;
@  IN  SOA     zap.junk.DE.       ralph.sysad.zap.junk.DE. (
1.1     ;Serial
10800   ;Refresh
3600    ;Retry
432000  ;Expire
86400)  ;Minimum
IN  NS      zap.junk.DE.
1   IN  PTR     localhost.
```

The following is the *hosts* file for server junk, followed by the relevant $INCLUDE file:

```
;
; junk zone hosts file for server junk
;
@               IN  SOA    junk.DE. ralph.sysad.zap.junk.DE. (
1.1    ;Serial
10800  ;Refresh
3600   ;Retry
432000 ;Expire
86400) ;Minimum
IN  NS     junk.DE.
IN  NS     widget.junk.DE.
IN  NS     zap.junk.DE.
widget.junk.DE.  IN  NS     widget.junk.DE.
IN  NS     junk.DE.
IN  NS     zap.junk.DE.
zap.junk.DE. IN  NS         zap.junk.DE.
IN  NS     junk.DE.
IN  NS     widget.junk.DE.
junk.DE.         IN  MX     10 junk.DE.
*.junk.DE.       IN  MX     10 junk.DE.
```

```
; hosts in the zone junk.DE
$INCLUDE /usr/local/domain/hosts/junk
; hosts in junk zone as listed in
; /usr/local/domain/hosts/junk
junk     A      223.100.100.1
A      10.1.0.56
MX     10 junk.DE.
widget   A      223.100.101.1
HINFO  "RM600"        SINIX
MX     10 junk.DE.
WKS    223.100.101.1  UDP syslog timed domain
WKS    223.100.101.1  TCP (echo telnet discard rpc sftp
uucp-path systat daytime netstat
qotd nntp link chargen ftp auth
time whots mtp pop rje finger
smtp supdup hostnames
domain nameserver)
zap      A      223.100.102.1
HINFO  "RM400"        SINIX
MX     10             junk.DE.
WKS    223.100.102.1  UDP syslog timed domain
WKS    223.100.102.1  TCP (echo telnet discard sftp
uucp-path systat daytime netstat
qotd nntp link chargen ftp auth
time whots mtp pop rje finger
smtp supdup hostnames
domain nameserver)
lazy     A      223.100.100.2
HINFO  "PCE5S"        SINIX
MX     10 junk.DE.
crazy    A      223.100.100.3
HINFO  RM400          SINIX
MX     10 junk.DE.
hazy     A      223.100.100.4
HINFO  RM400          SINIX
MX     10             junk.DE.
; all other hosts follow, up to 223.100.100.80
```

The following is the *hosts* file for server widget, followed by the relevant $INCLUDE file:

```
;
; widget zone hosts file for server widget
;
@ IN SOA widget.junk.DE. ralph.sysad.zap.junk.DE. (
1.1 ;Serial
10800 ;Refresh
3600 ;Retry
432000 ;Expire
86400) ;Minimum
IN NS widget.junk.DE.
IN NS junk.DE.
IN NS zap.junk.DE.
; junk.DE in hosts
$INCLUDE /usr/local/domain/hosts/widget
;
; hosts in widget zone as listed in /usr/local/hosts/widget
;
widget A 223.100.101.1
HINFO "RM600" SINIX
MX 10junk.DE.
whatsit CNAME widget.junk.DE
snelly A 223.100.101.2
HINFO PC SINIX
MX 10 junk.DE.
stinky A 223.100.101.3
HINFO 3B2 SINIX
MX 10 junk.DE.
dinky A 223.100.101.4
HINFO "RM200" SINIX
WKS 223.100.101.4 UDP who route
WKS 223.100.101.4 TCP (echo telnet discard sftp
uucp-path systat daytime
netstat ftp finger domain
nameserver)
MX 10 junk.DE.
; all other hosts follow, up to 223.100.101.110
```

The final example shows the file which contains the IN-ADDR.ARPA addresses for the machines in the zone junk. Since the complete classified domain name is specified, the machine names can be entered here without net addresses.

```
; IN-ADDR.ARPA addresses for the server junk in
; /usr/local/domain/junk.revzone
100.100.223.in-addr.arpa. IN SOA    junk.DE.   ralph.sysad.\
zap.junk.DE.(
1.1    ;Serial
10800  ;Refresh
3600   ;Retry
432000 ;Expire
86400) ;Minimum
IN NS     junk.DE.
1                 IN PTR    ljunk.DE.
2                 IN PTR    lazy.junk.DE
3                 IN PTR    crazy.junk.DE.
4                 IN PTR    hazy.junk.DE.
; Additional machines follow here, up to address 223.100.100.80
```

The files containing the IN-ADDR.ARPA addresses for the servers widget and zap are created analogously to the above file.

5.4 Interconnection between the DNS and NetBIOS naming systems

In addition to the DNS naming system there is the NetBIOS Name Service (NBNS), which is described in RFC1001/1002. This service is offered, for example, with the Advanced Server for UNIX (AS/X) WINS server from Siemens or with the WINS servers from Microsoft. It is used with the corresponding client software (resolver). Because both of these widely used NBNS servers are also known as WINS servers, we will refer to them as such below.

There are fundamental differences between the DNS and the NBNS:

1. While the DNS recognizes a hierarchically structured name space indicated by the names, which are separated by periods, with the NBNS there is only one hierarchy level; the names are not structured. A flat naming system of this type can only be used locally.

   Example:

   Because of the hierarchical structure of the DNS (which is described in Section "The domain hierarchy and DNS administrative zones", the IP address for a system saturn.xp.mchp.siemens.de. can be found from any DNS name server - and from any client that has access to a DNS name server. (This only applies if this machine name has been properly inserted into the hierarchical DNS domain system.)

   In comparison, the IP address for a system apollo can only be found in an NBNS naming system by querying the WINS server on which apollo is registered. There may be a large number of systems with the name apollo. Their different IP addresses are output by the various WINS servers on which the apollos are registered. This means that the response to the query for the address of apollo depends on which WINS servers the client queries.

   (A DNS name is only unique to the extent that the relevant domains are fixed in the official world DNS. It does not apply to private domains with separate Root servers. In DNS, a *gethostbyname()* call with an unstructured name, e.g. apollo, can also return a valid DNS response. This is because the resolver library has appended a suitable domain name from the search list in the */etc/resolv.conf* file to the unstructured name. The name server itself receives a query with a fully-specified DNS name, to which it responds.)
2. The characters permitted for forming machine/domain names differ for the DNS and NBNS. In general, the rules are stricter with the DNS. These rules can be found in RFC 1001/1002 (NetBios) and in RFC1035 (DNS).
3. In the NBNS naming system, address-to-name mapping is not implemented using the WINS server. Instead, the client interested in the name sends an *nbtstat* query directly to the address which is being queried. If a machine with this IP address exists and is a member of the NBNS naming system, it responds to the client with its name. Being a member of the NBNS naming system means, say for a Microsoft

Windows PC, that it is configured for WINS with a WINS name and with a WINS server address.

With the DNS, on the other hand, address-to-name mapping is carried out with the in-addr.arpa. domains, which are described in Section "The domain hierarchy and DNS administrative zones". On a server which, for example, is responsible for the addresses aaa.bbb.ccc.000 through aaa.bbb.ccc.255, a file is created for the zone ccc.bbb.aaa.in-addr.arpa.. The zone ccc.bbb.aaa.in-addr.arpa. is then made knowen to the higher-level server (the server for bbb.aaa.in-addr.arpa.). In this way, the name can be mapped to the address x.aaa.bbb.ccc using any DNS name server, provided there is an entry for x in the ccc.bbb.aaa.in-addr.arpa. database. (The client software (resolver) normally converts the address queries automatically into in-addr.arpa queries.)

5.4.1 Interconnection between the DNS name server and the NBNS server (WINS)

With the Reliant UNIX DNS name server it is possible to map names to addresses that are contained in a NBNS naming system rather than in the DNS. This means that an address for a name which is otherwise managed by a WINS server can be determined with an appropriately configured DNS name server.

The determination of names for addresses that are only known in the NBNS naming system is carried out by an appropriately configured Reliant UNIX DNS name server. A DNS name server of this type executes the *nbtstat* query referred to above for a DNS client and responds provided it receives a valid response from the system with the name in question.

This link between WINS and the DNS is especially advantageous when IP addresses on the WINS server or names and addresses on the WINS clients are automatically and dynamically modified (e.g. through the use of DHCP). Direct dynamic modification of the database of a DNS name server from another system is not currently possible for security reasons. However, by linking the DNS and NBNS (WINS), dynamic changes to name-and-address pairs in the WINS naming system can also be made visible in the DNS.

The DNS name server always checks its own database first. If it finds the required information there (a name or an address), it takes this and returns it to the client. Data that could have been obtained from a WINS server or from *nbtstat* queries is not accessible via DNS in this case.

5.4.2 Assigning unstructured NBNS names to hierarchical DNS domains

Unstructured NBNS names are organized into the DNS domain structure by assigning a WINS server to a zone for which the interconnected DNS name server is authorized.

Example:

On a DNS name server which is authorized for the zone xp.mchp.siemens.de., a WINS server is specified in the relevant database (for the format see below). All queries relating to systems from this domain that cannot be found by the name server in its own database are then forwarded to the specified WINS server.

The DNS name server receives an address query for apollo.xp.mchp.siemens.de.. If it does not find apollo in its database for xp.mchp.siemens.de., it forwards a query for apollo to the WINS server. If the WINS server responds with the requested address information, this information is forwarded from the DNS name server to its client together with the name apollo.xp.mchp.siemens.de..

This procedure is transparent for the client. It means that the response to the client is authorized by the DNS name server. With this mechanism, the address apollo.xp.mchp.siemens.de. can be resolved globally, even though the name apollo is entered on a WINS server.

When addresses are mapped to names, the DNS name server only receives an unstructured name in response to a *nbtstat* query. The simple NBNS name is assigned to a DNS doman by appending a domain name to the NBNS name. This domain name which is being appended is specified in a data record in the relevant *in-addr.arpa.* domain (see below).

Example:

The machine apollo has the IP address 139.22.228.18. The DNS name server is configured for *nbtstat* queries (for the format see below) using a special entry in the database for 228.22.139.in-addr.arpa. It is given special authorization for this domain. However, this database does not contain an entry for 139.22.228.18. In this case the name server sends an *nbtstat* query to 139.22.228.18. It only receives the

simple response apollo in return. In order to be able to send a full domain name to its client, it must append another domain name to apollo. This domain name being appended is specified in the database for 228.22.139.in-addr.arpa (for the format see below), for example xp.mchp.siemens.de.. The DNS client receives the pair 18.228.22.139 - apollo.xp.mchp.siemens.de. as a response.

5.4.3 Name-to-address mapping

In order to make the information on a WINS server available via a DNS name server, a zone must first be selected, for which this DNS name server is authorized. All WINS server machines can then be resolved via this domain in the DNS.

A resource record is entered in the file for this authorized zone. This record specifies the address of a WINS server:

domain IN WINS *win_addr* [LOCAL]

domain is governed by the rules described above. Every request for the address of a machine from *domain*, whose name cannot be found in its own database, is forwarded to the machine specified with the IP address *win_addr*. This machine must be configured as a WINS server.

With multiple resource records up to three WINS server addresses can be specified. If one WINS server doesn't respond, the next is queried. If a WINS server answers with a negative response (NXDOMAIN, address for name not found), no other WINS servers are queried.

For information on the keyword LOCAL, see Section "The keyword LOCAL".

Example:

The file for the authorized zone xp.mchp.siemens.de. contains the following lines:
@□□□IN□□□WINS□□□139.22.28.19

There is no data record for apollo in this file. The machine with the IP address 139.22.28.19 is configured as a WINS server; its WINS database contains the IP address 139.22.228.18 for apollo.

The address query apollo.xp.mchp.siemens.de. from a client to the DNS name server results in a query for apollo being sent to the WINS server from the name server. The WINS server responds and the name server responds to its client with apollo.xp.mchp.siemens.de. has the address 139.22.228.18.

5.4.4 Address-to-name mapping

To activate *nbtstat* queries on a DNS name server, in other words to enable address-to-name mapping in an NBNS naming system, a in-addr.arpa. domain must first be selected, for which this name server is authorized. For example, if *nbtstat* queries are to be executed for addresses from the network 139.22.228, the name server must be authorized for 228.22.139.in-addr.arpa..

The following data record is added to the file for this zone:

domain IN WINS-R *append_domain* [LOCAL]

domain stands for the relevant in-addr.arpa domain. *append_domain* is the domain name appended to the machine name, which the DNS name server has received in response to its *nbtstat* query. The DNS client receives the full domain name as a response: *machine_name*.*append_domain*.

Example:

The following is specified in the database for 228.22.139.in-addr.arpa.:
@□□□IN□□□WINS-R□□□xp.mchp.siemens.de.

No machine is registered under□
18.139.228.22.in-addr.arpa. in this database. However, there is a machine called apollo with the address 139.22.228.18, which responds to *nbtstat* queries (e.g. a Windows PC). The PTR query to the DNS name server for 18.228.22.139.in-addr.arpa. results in a *nbtstat* query to 139.22.228.18. 139.22.228.18 responds to this query with apollo. The DNS name server responds to its client with 18.228.22.139.in-addr.arpa. has the machine_name apollo.xp.mchp.siemens.de..

In this way, the Reliant UNIX name server can resolve an address that is otherwise known only in the NBNS naming system. Furthermore, **"any"** client with a simple *gethostbyaddr(3N)* can resolve such addresses provided the DNS name server responsible for the in-addr.arpa domain has been configured with the WINS-R data record.

5.4.5 The keyword LOCAL

The keyword LOCAL means that the name server does not send the WINS or WINS-R data record for zone transfers. Secondary servers (secondaries) in particular, that do not know the name server extension, would reject the data of the entire zone if they received an unknown WINS or WINS-R data record. This is avoided by specifying the keyword LOCAL.

If LOCAL is not specified for a primary server (primary) and the secondary server (secondary) knows the WINS or WINS-R data record, the same interconnection between the DNS and a WINS server is set up for the secondary server as for the primary server or the secondary server is configured in the same way as the primary server for *nbtstat* queries.

The WINS and WINS-R data record is understood by both Reliant UNIX secondary servers configured with the name server software and by Microsoft DNS secondary servers.

Data received from a WINS server or as a result of *nbtstat* queries is never forwarded by the DNS name server in the event of zone transfers.

5.4.6 Zone transfers from Microsoft Primary servers

If, in the event of a zone transfer, WINS or WINS-R data records are to be transferred from a Microsoft primary name server to a secondary name server, the secondary must query the zone from this Microsoft name server in a specific proprietary format. This is achieved by appending the letters MS to the relevant "secondary" line in the *named.boot* file.

Example:

The server is to be the secondary server for the domain xp.mch.de and retrieve the data from the Microsoft name server 111.22.33.44. This server has WINS data records which need to be transferred. The line in *named.boot* therefore reads as follows:
secondary xp.mch.sni.de 111.22.33.44 MS

or, if the data is to be stored in the *xp.mch.de.CACHE* file:
secondary xp.mch.sni.de 111.22.33.44 xp.mch.de.CACHE MS

All of the data records for the zone are thus transferred to the secondary server. Any WINS or WINS-R data records will not be included without the appended MS.

5.4.7 Time behavior - cache

An address that the DNS name server has received from a WINS server is only stored for a brief period in the name server cache (a few minutes). The same applies to names received as a result of *nbtstat* queries as well as negative responses or names not received.

With this short lifetime in the cache, data modified dynamically (e.g. with DHCP) and updated automatically on the WINS server is visible after a short period via the interconnected DNS name server.

5.4.8 Time behavior - delay

If the name server has been configured with the described interconnection with the NBNS naming system, it executes additional queries for zones in which it is authorized. For each query where no answer is found in the database, at least one additional query is sent to the WINS server.

In the case of address-to-name mapping, these additional queries must be clearly traced if the address is not available in the NBNS naming system: no machine responds to the *nbtstat* query with its machine name. The name server must then send a query several times and wait for a certain period for the response, as the response may be lost due to network problems or be delayed. The response for a name server configured with WINS-R to addresses for which it is authorized and cannot be resolved is returned within a few seconds.

Example:

The name server configured for *nbtstat* queries is authorized for the domain 228.22.139.in-addr.arpa. There is no data record in the database for 17.228.22.139.in-addr.arpa.. There is no machine with this IP address.

If the name server now receives a PTR query for 17.228.22.139.in-addr.arpa., it sends a *nbtstat* query to the address 139.22.228.17. It receives no response, but does not know if this is perhaps due to network problems or the fact that this machine is not available. After a short interval, it therefore attempts further *nbtstat* queries to this machine. It finally gives up and returns the negative response NXDOMAIN to its client.

If, in the same example, the name server had not been configured for *nbtstat* queries, it would have returned the negative response immediately.

5.4.9 Restrictions

The names permitted in the NetBIOS naming system do not comply with the rules for DNS domain names. If a machine name is not permitted in the context of the DNS, it will not be found using the DNS server, even if it is contained in the interconnected NetBIOS database.

5.4.10 Dependencies with Advanced Server for UNIX (AS/X)

The name server uses port 137 for NetBIOS address-to-name mapping (*nbstat* queries). The NetBIOS component of AS/X also occupies port 137. More recent versions of AS/X have a procedure for sharing the port with the name server. If conflicts occur which prevent address-to-name mapping using the NetBIOS procedure from functioning with the name server, AS/X Version 4.0B00 or later needs to be installed. Conflicts are logged by the name server with *syslog()*.

Further evidence of a conflict may be in the fact that NetBIOS (and therefore AS/X) cannot be started with an old AS/X release even if the name server is already running.

6 The Network File System (NFS)

Reliant UNIX 5.45, through the use of the Network File System (NFS), allows you to share the resources of a machine across a network. Resources are files, file systems and parts of file systems.

NFS versions 2 and 3 are supported.

This chapter explains how to administer the Network File System. It explains how the NFS works, and provides the procedures for sharing resources on the network and for accessing resources shared by other machines.

Since this chapter is intended as an introduction to the NFS, the more sophisticated applications of the commands presented here are not discussed. For additional information on NFS commands, please see the corresponding descriptions in the "Networking Reference Manual".

Audience

This description is intended for use by staff experienced in the administration of distributed systems. You should be familiar with such administrative tasks as:

- managing local file systems
- assigning access rights
- maintaining system security.

Organization

This chapter assumes you are setting up your machine as an NFS client and/or an NFS server, i.e. both to share local resources on the network, and to mount remote resources shared by other systems. If you are administering a machine that will mount but not share resources, you can limit your reading to the NFS overview in this chapter, the chapters that explain how to mount remote resources, and the relevant information contained in the section entitled Error analysis and diagnostics.

This chapter is subdivided into the following sections:

- "Introduction", which describes fundamental aspects of the NFS.
- "Setting up an NFS server", which tells you how to start NFS operations on a server and provide resources for mounting on remote systems.
- "Mounting and unmounting remote resources on NFS clients", which tells you how to mount resources that have been shared by remote machines.
- "Information about resources", which tells you how to get information about resources that have been shared and mounted.
- "Using the automounter", which tells you how to use the NFS's automatic mounting program.
- "PC-NFS - NFS servers for PC clients", which describes how the *pcfnsd* daemon handles client queries from PCs.
- "Other services relevant to the NFS", which contains subsections on the NFS lock manager, the status monitor and asynchronous writing across the NFS.
- "Error analysis and diagnostics", which tells you how to deal with problems occurring with the NFS.

If you are using a system equipped with the graphical user interface of the SINIX/windows user environment, you can use the functions provided by the NFS Manager to share and mount resources. Detailed information on the NFS Manager is available from the interface's help system.

6.1 Introduction

The NFS is a service that enables machines of different architectures running different operating systems to share files and resources across a network. The NFS makes it possible for a machine to share local files and directories, and permits remote users to access those files and directories as though they were local to the user's machine.

The NFS provides file sharing in a heterogeneous environment, potentially containing many different operating systems; it has been implemented on operating systems ranging from MS-DOS to VMS.

The NFS can be implemented on different operating systems because it defines an abstract model of a file system. On each operating system, the NFS model is mapped into the local file system semantics. As a result, normal file system operations, such as read/write, operate in the same way that they operate on a local file system.

The benefits of the NFS are many. It allows multiple machines to use the same files, so the same data can be accessible to everyone on the network. Storage costs can be kept down by having machines share applications, and database consistency and reliability is enhanced by having all users read the same set of files.

6.1.1 NFS resources

The objects that can be shared through NFS servers include any whole or partial directory tree or local file hierarchy - including a single file. A machine cannot share a file hierarchy that overlaps one that is already shared.

Sharing of device files is not possible; peripheral devices such as modems and printers cannot be shared via the NFS.

For device files, such as the entries under */dev*, access with NFS is controlled by separate regulations (see the *mount* command, *-o devacc* option in the Man Pages). If a device file located on NFS is opened on an NFS client, the client device is addressed rather than the server device. This corresponds to the behavior whereby imported NFSs are considered as local file systems. This is useful with "diskless workstations", for example.

In most UNIX system environments, such a sharable file hierarchy will be a file system, or part of one. However, the function of the NFS does not depend on any particular operating system, and the concept of a "file system" may have no meaning in a non-UNIX environment. This guide will therefore always use the term "resource" when referring to a file or file hierarchy that can be shared and mounted over the NFS.

When you mount a local file system on a local mount point, you mount the entire file system, starting at its root. When you mount a remote resource through the NFS, you are not restricted to mounting the entire file system. You can mount any directory or file in the file system tree, and gain access only to that directory or file and anything beneath it. For example, in the illustration, Machine A is sharing its entire /usr file system. If Machine B wants access only to the files and subdirectories in /usr/man on Machine A, it can mount /usr/man rather than /usr; then, nothing above usr/man on Machine A appears in Machine B's directory tree.

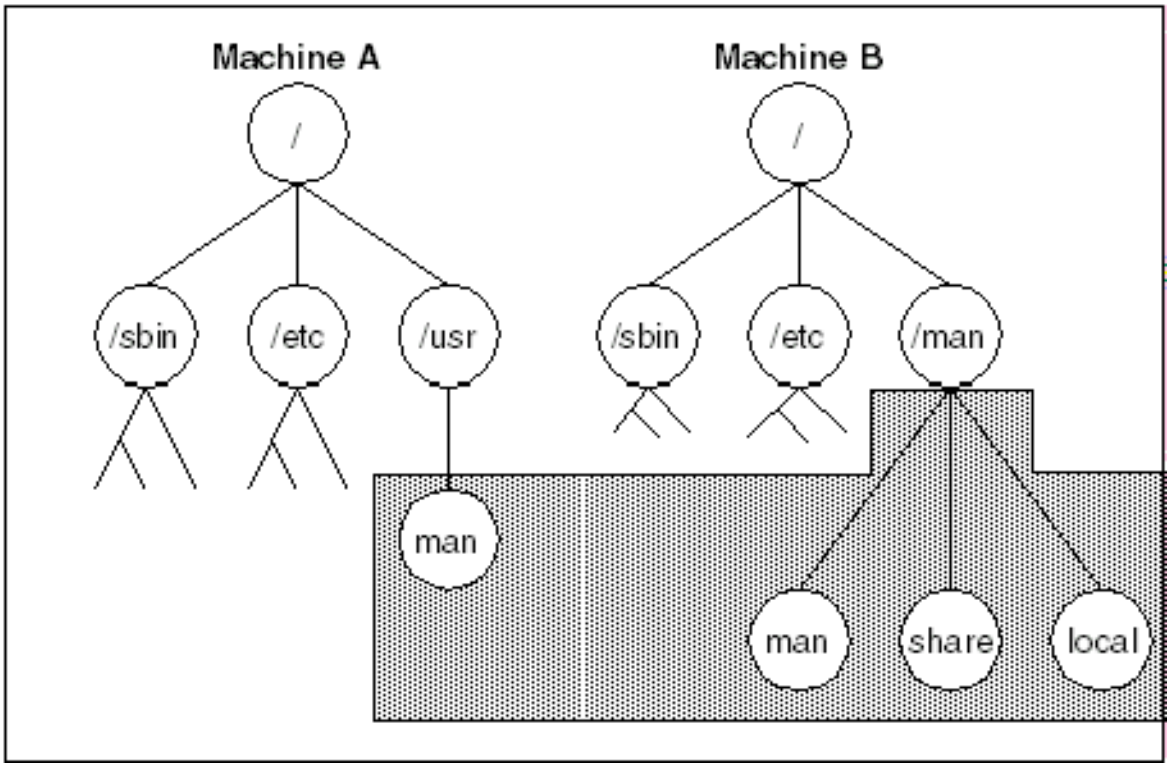


Figure 11: Mounting a remote resource

It would be illegal for Machine A to share both /usr and /usr/man if both resources reside on the same disk partition. In this case, you would have to share /usr and leave it to network machines to decide whether to mount /usr or /usr/man.

If you want to mount a single file, you must mount the file on a directory. Once it is mounted, you cannot remove it (using *rm*) or move it to another directory (using *mv*); you can only unmount it.

6.1.2 NFS servers and clients

A machine that makes a local resource available for mounting by remote machines is called an "NFS server" A machine that mounts a resource shared by a remote machine is an "NFS client" of that machine. Any machine with a disk can be a server, a client, or both at the same time.

A server can support a diskless client, a machine that has no local disk. Since a diskless client has no resources, it has to keep its files on the server. A diskless client can act only as a client - never as a server.

Clients access files on the server by mounting the server's shared resources. When a client mounts a remote resource, it does not make a copy of the resource; rather, the mounting process uses a series of remote procedure calls that enable the client to access the resource transparently on the server. The mount looks like a local mount, and users enter commands just as if the resources were local.

6.1.3 NFS in a 64-bit operating system

Reliant UNIX Versions 5.44 and later are 64-bit operating systems and, in particular, offer the option of working with files that are larger than 4 Gbytes.

NFS Version 3, which was already supplied with Reliant UNIX 5.43, supports files larger than 2 Gbytes (large files) and file systems larger than 4 Gbytes (large file systems). With Reliant UNIX 5.44 and later, it is possible to configure large file systems locally and make them available to NFS clients. The local file systems can contain one or more large files.

Support for large files extends to all types of access where data ranges are addressed, i.e. the reading, writing, querying, and modification of the file size as well as the setting of file locks.

Access to NFS files is always initiated by the NFS client and performed by the NFS server. If both the client

and the server support 64-bit operation, large files can be processed without restriction. Problems occur, however, if the client and server have different word sizes.

The following sections describe what happens when different operating system versions and NFS versions are used. The procedure described here may differ from that of other systems with different implementations (e.g. non-SNI systems).

6.1.3.1 32-bit clients and 64-bit servers

If you have a 64-bit server operating with 32-bit clients, the problem arises that the client may not be able to interpret certain file or file system parameters, such as the file size. As with 32-bit applications on local 64-bit file systems, client system calls may fail under these circumstances.

Exception:

All clients, for which a large file system is made available, may generally also mount this file system regardless of whether or not the file system parameters can be represented on the clients.

Behavior in relation to NFS V2 clients

If a large file system is made available for NFS clients under Reliant UNIX 5.44 and later, it can also be mounted by NFS V2 clients. The file system parameters (number of blocks etc.) are displayed correctly up to a maximum file system size of 1 Tbyte. However, the server refuses all access by the client to a large file because the client cannot represent the file size in 32-bit mode.

Behavior in relation to NFS V3 clients

If you are using NFS V3, the server cannot differentiate between a 32-bit and a 64-bit client. The 32-bit client therefore has to decide how to respond in the case of file or file system parameters that cannot be represented in 32-bit mode. The client will normally allow a system call to fail if the resulting parameters cannot be output in 32-bit mode.

6.1.3.2 64-bit clients and 32-bit servers

If Reliant UNIX 5.44 or later mounts a file system from a 32-bit NFS server, the server returns an error message when reading, writing and setting the file size if the associated parameters cannot be represented in 32-bit mode.

6.1.4 An overview of NFS administration

Your responsibilities as an NFS administrator depend on your site's requirements and the role of your machine on the network. You may be responsible for all the machines on your local network, in which case you may be responsible for installing the software on every machine, determining which machines, if any, should be dedicated servers, which should act as both servers and clients, and which should be clients only.

If your site has a network administrator, and you are the administrator of a client-only machine, you may be responsible only for mounting and unmounting remote resources on that machine.

Maintaining a machine once it has been set up involves the following tasks:

- Starting and stopping NFS operation
- Sharing and unsharing resources necessary during a server session
- Mounting and unmounting resources as necessary during a session
- Modifying administrative files to update the lists of resources your machine shares and/or mounts automatically
- Calling and analyzing network status information
- Localizing and fixing NFS-related problems as they arise
- Setting up maps to use the automounter (see Section "Using the automounter")

6.1.5 Starting and stopping the NFS

By default, the NFS becomes operational automatically whenever you enter run level 2 (multiuser mode).

The NFS allows you to share and mount resources explicitly by entering commands at the command line. You

can also set up the system to share and mount resources automatically by editing a number of administration files. If you set up automatic sharing and mounting, a pre-determined set of resources is shared and/or mounted whenever you start NFS operation.

When you exit run level 2, the resources you shared and mounted are automatically unshared and unmounted, and the NFS stopped.

To start and stop the NFS using the command line, type

sh /etc/init.d/nfs start

and

sh /etc/init.d/nfs stop

6.1.6 NFS-related processes

nfsd The NFS server processes. These must be running on every NFS server and execute the NFS requests of the clients. By default, four NFS server processes are started.

mountd
The mount server process. This must be running on every NFS server. It is used for mounting and unmounting operations.

lockd-clnt, *lockd-srv*
The processes of the Network Lock Manager (NLM). These are required to set locks on files or file areas. They must be running on both the client and the server.

statd The Network Status Monitor (NSM). This is required by the lock manager to restore lock tables after machines have failed and must therefore, like the lock manager processes, also be running on both the client and server.

biod Daemon processes for speeding up the read and write accesses of an NFS client. By default, four are started. The *biod* processes only need to run on the client.

bootparamd
Server process allowing diskless clients to query their boot parameters.

rpc.pcnfsd
Server processes for checking authorization and executing print jobs for PC-NFS clients.

commitd
commitd is a daemon on the client which periodically sends RPCs to a specific server to stabilize write blocks. *commitd* processes are generated with the *mount* command (when needed).

nfs_fsflush
The *nfs_fsflush* system process on the NFS client is used to stabilize write jobs on the server. An *nfs_fsflush* system process is started with the *mount* command for each mount point.

6.2 Setting up an NFS server

To make your machine available to NFS clients in the network as an NFS server, you must share local resources for mounting on remote machines. The following options are available here:

- You can make local resources available for use on a once-only basis. This is described in the next section.
- You can make local resources available for use on a permanent basis. This is described in the Section "Sharing resources on a permanent basis".

The procedure for revoking shared resources is described in the Section "Revoking shared resources".

6.2.1 Sharing resources for use on a once-only basis

This section shows you how to share NFS resources using the commands *share* and *shareall*.

6.2.1.1 The share command

To make a resource on your machine available to clients, you use the *share* command, which uses the following syntax:

share [–o *options*] [–d *description*] [*pathname*]

–o *options*
: is a list of file system-specific options that can follow the *-o* flag; if no option is specified, then the system assumes the *rw* option, and sharing is done read/write to all clients (see below for more information).

–d *description*
: is a description enclosed in quotation marks that you supply to describe resources to clients.

If you enter the *-d* option when you share an NFS resource, the description is output when you call the *share* command without options. However, if a client calls a list of the currently available resources with the *dfshares* command, this text is not displayed.

pathname
: indicates the pathname of the resource to be shared; pathnames should not exceed 255 characters.

When sharing a resource over the NFS, the following suboptions can follow the *-o* flag:

anon=*uid*
: sets the UID and GID of the system administration ID on the client system to *uid* for every NFS request on the server system. If this option is not specified, the UID of the client system administration ID on the server is set to UID_NOBODY (600001) and the GID is set to GID_NOBODY (600001). If *uid* equals -1, every NFS request under the client system administration ID is rejected.

anon=*uid*:
: the UID of the client system administration ID is set to *uid* for every NFS request on the server. The GID of the system administration ID keeps the default value GID_NOBODY. If *uid* equals -1, every NFS request under the client system administration ID is rejected.

anon=:*gid*
: the GID of the client system administration ID is set to *gid* for every NFS request on the server. The UID of the system administration ID keeps the default value UID_NOBODY.

anon=*uid*:*gid*
: the UID of the client system administration ID is set to *uid* and the GID is set to *gid* for every NFS request on the server. If *uid* equals -1, every NFS request under the client system administration ID is rejected.

mknod
: enables device files to be created in the shared file system. The device corresponding to the device file belongs to the client's local resources.

rw
: indicates that sharing is to be done read/write to all clients.

ro
: indicates that sharing is to be done read-only to all clients.

rw=*client*[:*client*]...
: indicates that sharing shall be done read/write to the listed clients; when sharing over the NFS, this suboption can be used in conjunction with the *ro* or *ro=* suboption to share the resource read/write to some clients and read-only to others.

ro=*client*[:*client*]...
: indicates that sharing shall be done read-only to the listed clients; when sharing over the NFS, this suboption can be used in conjunction with the *rw* or *rw=* suboption to share the resource read/write to some clients and read-only to others.

root=*host*[:*host*]...
: indicates that only the user name root (UID 0) of the listed machines has root access to the shared resource. If this suboption is not specified, no user name of an NFS client receives root access (except when *anon*=0).

The command fails if the same client is given ambiguous access rights, such as when the same client

name is included in both an *rw*= list and an *ro*= list, or when both the *rw* option and the *ro* option are given without arguments.

If no options are specified when the *share* command is entered, then the command displays a list of all resources on your system that are currently shared. The *share* command can be used in this capacity by users as well as administrators.

Example 1

You want to share the /home2 file system with NFS clients. The directory is to be shared with read-only access for all clients. In addition, the file system is to be output with the comment "Current projects" when the *share* command is called without options. Enter the following in the command line:

share –o ro –d "Current projects" /home2

Example 2

You want to share a partial file system with NFS clients as a resource. The root directory of the branch of the file system is graphics, which is a subdirectory to your /export directory. You want to share the directory read-only to all clients except an NFS client named art.dept, with which you want to share the directory read/write. At the command line, type

share –o ro,rw=art.dept /export/graphics

6.2.1.2 The shareall command

shareall enables you to execute several *share* commands with a single command. The resources are specified in a file that must already exist when you run the command. The format of the entries corresponds to the syntax of the *share* command and the entries in *dfstab*:

share [–o *options*] [–d *description*] [*pathname*]

Once you create the file, you specify it as the input file when you enter the *shareall* command.

If you do not specify an input file, the */etc/dfs/dfstab* file is used by default. If you enter a dash (–) in place of the name of an input file, the system expects standard input, i.e. you enter the appropriate *share* commands in the command line and then execute them pressing [CTRL] and [d]. This saves you from entering one *share* command, waiting for the system to execute the command and return your prompt, entering another command, and so on.

The syntax of the *shareall* command is as follows:

shareall [– | *file*]

– indicates that the command should accept standard input, and

file is the name of the file you created to be your input file.

Example 1

You create an input file called misc that contains commands to share three separate resources. The file looks like this:

```
#cat misc
share –o ro /usr/reports/mtg.notes
share –o ro,rw=art.dept /export/graphics
share /usr/man
```

To share all the resources listed in the file you created, type

```
shareall misc
```

Example 2

You want to share three separate resources. Because you will not be sharing the same resources as a set on a regular basis, you do not want to create an input file. Type

```
shareall –
```

Your cursor moves to a new line. Enter the following commands, pressing [Enter] after you enter each command.

```
share –o ro /usr/reports/mtg.notes
share –o ro,rw=art.dept /export/graphics
share /usr/man
```

To execute the commands once they are entered, press [Ctrl] + [d].

6.2.2 Sharing resources on a permanent basis

If you want to make certain resources permanently available to other computers, you can set up your system so that these resources are shared automatically whenever you start NFS operation. You should do this, for example, if you need to share a resource permanently and client access is never or very rarely withdrawn.

The file you need to edit to set up automatic sharing is *dfstab*, located in */etc/dfs*. *dfstab* is created automatically by the NFS at installation.

The *dfstab* file lists all the resources that your server shares with its clients and controls which clients may mount a resource. If you want to modify *dfstab* to add or delete a resource, or to modify the way sharing is done, simply edit the file with any supported text editor (such as *vi*). The changes take effect the next time the machine enters run level 2, a *shareall* command is executed or the NFS is stopped and restarted.

Each line in the *dfstab* file contains a *share* command - the same command you enter at the command line to share a resource explicitly. The *share* command was described in the section entitled The share command.

6.2.3 Superuser access to shared resources

A server uses NFS to make its resources available to other machines, which can mount these resources locally. However, a user who becomes the superuser at a client is denied access as the superuser to mounted remote resources. When a user logged in as root on one machine requests access to a remote file shared through the NFS, the user's ID is changed automatically to User ID 60001 (=UID_NOBODY, defined in *<sys/param.h>*) without further precautions. The group ID used is also 60001. The user with the ID 60001 has only the same access to a file as all the other users.

When you share a resource, you can permit the system administrator on a particular machine to have root access to that resource by editing */etc/dfs/dfstab* on the server, or by specifying the appropriate options on the command line. For example, suppose you want the machine samba (but no others) to have superuser access to the shared directory /usr/src. You enter the following command in */etc/dfs/dfstab*, or at the command line.

share -o root=samba /usr/src

If you want more than one client to have root access, you can specify a list, as follows:

share -o root=samba:raks:jazz /usr/src

If you want all client processes with *uid* 0 to have superuser access to /usr/src, you enter

share -o anon=0 /usr/src

anon is short for "anonymous". Anonymous requests, by default, get their user ID changed automatically from its previous value to the UID 60001. NFS servers label as anonymous any request from a root user (someone whose current effective UID is 0) who is not in the list following the root=*host* option in the *share* command. The above command assigns the UID 0 to anonymous requests from root users with system administrator privileges. In other words, they retain the UID 0 and the associated privileges.

Please note that this procedure can affect the system security of the NFS server.

6.2.4 Revoking shared resources

Shared resources can be revoked as follows:

- If you shared the resource by making an entry in */etc/dfs/dfstab* for permanent use, simply delete the entry there and use the *unshare* command to revoke the current resource sharing.
- Use the *unshare* command to revoke the sharing of a resource.
- Use the *unshareall* command to revoke the sharing of all resources or of resources of a particular type.

Before you "unshare" a resource using either the *unshare* or *unshareall* command, be sure you understand the effect of unsharing existing mounts. Unsharing an NFS resource will mean that the client will also no longer be able to access the resource if it was mounted before the command was executed.

6.2.4.1 The unshare command

The *unshare* command revokes client access to a resource; i.e. the resource can no longer be mounted on other systems, and access to already mounted resources is refused.

The *unshare* command can be used to unshare any resource - whether the resource was shared explicitly with the *share* command or permanently through the *dfstab* file. In the latter case, you should remember that *unshare* has no effect on the content of *dfstab* and that the resource will be shared again the next time you change to run level 2. The syntax for the command is:

unshare *pathname*

pathname

indicates the pathname of the resource to be unshared.

The *unshare* command supports file system-specific options so that file system types developed in the future can offer package-specific functionality. At this time, the NFS does not supply specific options to the *unshare* command.

Example

You want to revoke client access to the /store/invdata.88 directory. Enter the following:

unshare /store/invdata.88

You can use the *share* to make the directory available again immediately.

6.2.4.2 The unshareall command

unshareall enables you to revoke client access to all the resources currently being shared on your system. *unshareall* has the following syntax:

unshareall

6.3 Mounting and unmounting remote resources on NFS clients

Once a resource has been shared on a server via the NFS, it can be mounted by all authorized clients. Mounting can be done automatically when NFS operation begins on the client (when the client enters run level 2), or explicitly, by using the command line during a work session. If certain remote resources must be permanently available, it is recommended that you set up automatic mounting when you first set up NFS operation.

This section tells you how to mount and unmount remote NFS resources explicitly, using the generic *mount*, *umount*, *mountall*, and *umountall* commands. It also tells you how to mount remote resources automatically by making entries in */etc/vfstab* file. In the descriptions of the commands and files, only those aspects relevant to the administration of remote resources are included.

6.3.1 Mounting a remote resource on a once-only basis

You can mount a resource using the command *mount*. If you want to mount several resources, use *mountall*.

6.3.1.1 The mount command

The *mount* command allows you to mount both local and remote file systems. To mount a remote resource using the *mount* command, the server must be accessible and have shared the desired file system so that clients can access it.

The syntax of the *mount* command as it relates to mounting distributed file systems is:

Format 1:

mount [–F f*styp*] [–o *options*] {*resource* | *mountpoint*}

Format 2:

mount –F *fstyp* [–o *options*] *resource mountpoint*

You use format 1 if there is an entry in the */etc/vfstab* file for the desired resource that is to be executed with the *mount* command. You use format 2 when there is no entry in */etc/vfstab*.

–F *fstype*
: is the type of the resource you want to mount, (in this case, *fstype* has the value nfs). If the *–F* option is not specified, and *resource* and *mountpoint* are, then *mount* looks in */etc/vfstab* for a corresponding entry and mounts the resource according to the file system type specified there.

–o *options*
: is a list of file-system-type-specific options that can be specified after the *–o* option separated by commas (no spaces). The NFS-specific options are described later in this section.

resource
: is the resource to be mounted, specified as follows:

  server:pathname

  server is the machine name of the NFS server, *pathname* the full pathname of the resource of this server. If you do not specify a resource, the parameter is read from the *vfstab* file (provided one exists there).

mountpoint
: indicates where the resource should be mounted locally as follows: *pathname*

  If you do not specify a mount point, the parameter is read from the *vfstab* file (provided one exists there).

The options that can follow the *–o* option are:

devacc
: permits accessing of device files. Device files cannot be opened if this option has not been specified. This situation is sometimes desired since with the NFS, device files are opened on the client, not on the server.

rw | ro

rw indicates that the resource is to be mounted read/write, and *ro* indicates it is to be mounted read-only. If no option is specified, the resource is mounted read/write by default - as long as the resource was shared read/write. Alternatively, if the resource was read-only shared, it will be mounted in that state.

suid | nosuid
: *suid* indicates that set-uid bits are to be obeyed on execution, and *nosuid* indicates that they are to be ignored. (If no option is specified, set-uid bits are obeyed by default.) Every directory mounted *rw* is a good candidate for *nosuid*.

bg | fg
: *bg* indicates that a retry should be initiated in the background when the server does not respond, and *fg* indicates it should be initiated in the foreground. If no option is specified, the retry is initiated in the foreground by default. (This option does not apply to the automounter.)

soft | hard
: specifies whether the mount operation is "soft" or "hard". An NFS request for a file system mounted with the *hard* option is repeated until the server responds. A request for a file system mounted with the *soft* option is aborted when the server does not respond before the time limit has elapsed (the *timeo* option) or after a set number of retransmissions (the *retrans* option). The default setting is: hard.

locallocks
: Only sets locks on files in the mounted NFS file system locally on the client (not on the server). This is useful if the mounted client is the only client using the mounted NFS file system; for example a client without a hard disk.

intr
: indicates that signal interrupts should be allowed while NFS requests are being issued. If you do not specify this option, on a "hard mount" the client process is blocked until the request succeeds. This suboption affects all subsequent NFS requests for the mounted file system. Requests for a soft-mounted file system are interruptible.

retry=*n*
: indicates the number of times to retry the mount operation. The default for *n* is 10,000 times.

retrans=*n*
: maximum number of retransmissions in the case of NFS requests for a soft-mounted file system. The default for *n* is 5.

timeo=*n*
: specifies the maximum time (in tenths of a second) a client should wait for an NFS job to be carried out without a message being output. During lengthy communications via the NFS, the client will continually recalculate the timeout value to reflect the server's response times. The default is 11 (1.1 seconds).

uforce
: An attempt to unmount a resource which was mounted using the *uforce* option will succeed within a short time, even if the server cannot be reached. It is also possible to unmount a file system, which was mounted without the *uforce* option by invoking *umount -o uforce*.

v2
: mounts a resource of a server using the NFS protocol Version 2, even when the servers support Version 3.

Resources mounted explicitly with the *mount* command stay mounted during a work session unless you unmount them, using the *umount* command. If you exit run level 2 and then re-enter it, the resource is no longer mounted (unless you edited the *vfstab* file to include the mount automatically).

The *mount* command without options outputs a list of all the resources currently mounted on the local machine from both remote machines and the local machine.

Example 1

You want to mount an NFS resource of the server called tools.srv. The resource is a directory called /usr/graphics on the server. It is to be mounted in the graphics directory in /usr/tools on the client. You want to mount the resource read/write, and you want set-uid bits to be ignored. Type

mount –F nfs –o nosuid tools.srv:/usr/graphics \
/usr/tools/graphics

Because no access rights are specified, the directory is mounted read/write.

Example 2

You want to mount a directory containing online documentation (man pages) that resides on a server called docgroup. The request is to be repeated until the server responds (hard mount). You want the directory mounted read-only, with interrupt enabled. The directory is an NFS resource called /usr/man on the NFS server. You want to mount the directory to a mount point of the same name on your machine. Type

mount –F nfs –o ro,intr docgroup:/usr/man /usr/man

6.3.1.2 Mounting a set of resources - the mountall command

You can also mount a set of NFS resources locally at the same time by entering a single command - the *mountall* command. To use the command, you first create a file that lists the resources you want to mount. The syntax of the entries in your file is the same syntax as the entries in the */etc/vfstab* file, as follows:

resource fsckdev mountp fstype fsckpass automnt mntopts

For an explanation of these parameters, see the section entitled Mounting remote resources automatically.

Once you create a file, you specify it as the input file when you enter the *mountall* command.

If you do not specify an input file, the */etc/vfstab* file is used by default. If you enter a dash (–) in place of the name of an input file, the system accepts standard input, i.e. you enter the appropriate *mount* commands in the command line and then execute them pressing [CTRL] and [d]. This saves you from entering one *mount* command, waiting for the system to execute the command and return your prompt, entering another command, and so on.

The syntax of the *mountall* command is as follows:

mountall [–F *fstype*] [–l | –r] [– | *file*]

–F *fstype*
: indicates that the resources of the specified file system type with entries in the input file are to be mounted. If no *–F* option is given, then all resources with entries in the input file are mounted.

–l
: indicates that only local file systems listed in the input file are to be mounted. This option cannot be used together with the *-F nfs* option.

–r
: indicates that only remote resources listed in the input file are to be mounted.

–
: indicates that the command should accept standard input.

file
: is the name of the file you created to be your input file.

If the *mountall* command is entered without arguments, it mounts all local file systems and remote resources in the *vfstab* file that have the *automnt* field set to yes.

Example 1

You want to mount a set of remote resources. You create an input file called mntlist that lists three remote resources. The file looks like this:

#cat mntliste
docgroup:/usr/man – /usr/man nfs – yes ro
export:/export/data – /usr/old.blues nfs – yes rw
frontoffice:/usr/share – /usr/local/tmp nfs – yes rw

To mount all the resources listed in the file you created, type

mountall mntlist

Example 2

Your system mounts resources automatically when you enter run level 2, using the */etc/vfstab* file. While resources are still mounted, you add entries to the *vfstab* file. Now you want to mount the resources you added to the file, without exiting run level 2 and unmounting currently mounted resources. Enter:

mountall

The system uses the updated *vfstab* file as the input file, i.e. it mounts all of the new resources in the file and displays error messages for those resources that are mounted already. Entries deleted from */etc/vfstab* are not deactivated; they must be unmounted using the *umount* command.

6.3.2 Mounting remote resources automatically

The */etc/vfstab* file allows you to mount a local file system automatically when you boot the system. If you edit the file to include a remote file system or directory, the remote resource is mounted automatically when you change to run level 2.

A server can be a client of another server on a local network, in which case, the server's *vfstab* may need to include both local and remote mounts. For information about adding local mounts to the *vfstab* file, see the "System Administrator's Guide".

To mount a remote resource automatically, create a mount point for the resource using the *mkdir* command, then edit the *vfstab* file. Entries in the *vfstab* file require the following syntax:

resource fsckdev mountp fstype fsckpass automnt mntopts

resource
: is the resource to be mounted, specified as follows:
server:*pathname*

server is the name of the NFS server sharing the resource to be mounted, and *pathname* is the full pathname of the resource on this NFS server to be mounted.

fsckdev
: is the name of the raw device to *fsck*; for a remote mount, the parameter is not applicable, and the entry should be -.

mountp
: is the full pathname indicating where the resource is to be mounted locally, specified as follows:
pathname

fstype is the file system type of the resource you want to mount.

fsckpass
: is the pass number to use for multiple *fsck* passes. For a remote mount, the parameter is not applicable and must be replaced by a -.

automnt
: indicates whether the entry should be automounted when the client is booted or enters run level 2 (yes or no).

mntopts
: is a list of comma-separated suboptions identical to the options (*-o*) passed to the *mount* command, i.e. any of the NFS-specific *–o* options (with the exception of *bg* and *fg*) supported by the *mount* command can also be entered in the *vfstab* file. (These options were also described in the Section "The mount command".)

Once you create the mount point and edit the *vfstab* file, the file system or directory will be mounted automatically whenever you enter run level 2. If the updated */etc/vfstab* is to take effect immediately, you can specify this with the *mountall* command. It will continue to be mounted automatically until you modify *vfstab*, or until the server revokes the client's right to access the resource. The contents of */etc/vfstab* remain the same.

Example

You want to mount the NFS resource */usr/share* on a server named office. You want the resource to be

mounted automatically for read-only access every time you take your system to run level 2. You want to mount the directory on the mount point /usr/local/tmp. First create the mount point by typing

mkdir /usr/local/tmp

Make sure the mode and permissions of the new mount point match those of the resource you want to mount on it. Then use the *cd* command to move to your */etc* directory and edit the *vfstab* file, using any supported text editor. The file entry should look like this:

office:/usr/share – /usr/local/tmp nfs – yes ro,bg

The option *bg* is specified in order to avoid the NFS client hanging on entering run level 2 if the server office cannot be reached.

6.3.3 Unmounting a remote resource

Use *umount* to unmount a single resource, *umountall* to unmount several resources.

6.3.3.1 The umount command

The *umount* command allows you to unmount both local and remote file systems. You can unmount a resource with *umount* whether is was mounted explicitly using the *mount* command or automatically through the *vfstab* file.

umount in its modified form for unmounting remote resources is:

umount {*resource* | *mountpoint*}

resource

is the resource to be unmounted, specified as follows:
server:pathname

server is the name of the NFS server sharing the resource to be unmounted, *pathname* the complete pathname of the resource of this server to be unmounted.

mountpoint
is the pathname of the mount point on the client where the resource is mounted, specified as follows:
pathname

Example

You want to unmount a remote NFS resource by specifying the server docgroup sharing the resource and the pathname of the resource */usr/man* on the server. Type

umount docgroup:/usr/man

6.3.3.2 The umountall command

You can use the *umountall* command to unmount all the resources currently mounted on your system, except the root file system, or all the mounted resources of a specified file system type. The syntax of the *umountall* command is

umountall [–F *fstype*] [–k] [–l | –r]

–F *fstype*
indicates the file system type of the resources to be unmounted. When you specify a file system type, the command unmounts all local and remote resources of the file system type you specified. If no *–F* option is specified, then the command unmounts all local file systems, except the root file system, and all remote resources.

–k indicates that all processes with files open when the *umountall* command is executed are to be killed.

–l indicates that only local file systems are to be unmounted.

–r indicates that only remote resources are to be unmounted.

Example 1

You want to unmount all the remote NFS resources currently mounted on your system. Type

umountall –r

Example 2

You want to unmount all file systems currently mounted on your system, except the root file system. You want to kill all processes that have open files. Type

umountall –k

6.3.3.3 Unconditional unmounting

If the NFS server is not available at the time of unmounting, the *umount* command can be delayed and may even hang. This can then, for example, make the controlled shutdown of a client impossible.

Difficulties may also be experienced if the machines are configured as master and backup hosts in a high-availability configuration. If NFS file systems have been mounted on the backup machine by the master machine, and the master machine then fails, the backup machine has the problem of unmounting the mounted NFS file systems.

Problems of this kind can be avoided by mounting NFS file systems with the option *uforce*.

File systems mounted with the *uforce* option can be unmounted immediately, even if the server is not accessible.

It is also possible to unmount a file system, which was mounted without the *uforce* option by invoking *umount -o uforce*.

6.3.3.4 Canceling hanging processes

For various reasons it can be necessary to mount NFS file systems in such a way that they cannot be canceled, for example if programs are started whose binary images reside on the NFS server.

The file system cannot be unmounted if the server is not available but processes are still waiting for it to respond. This is also the case even if the *uforce* option was used during the mount process. In such a case, the hanging processes can be made cancelable using the following instructions:

```
# crash
> w nfsintr 1
> q
```

Note that this means that all NFS file systems are mounted in such a way that they can be canceled. Once the hanging processes are terminated by signals and the file system is unmounted, the cancel option should be reversed immediately using the following instructions:

```
# crash
> w nfsintr 0
> q
```

6.4 Information about resources

Using *share*, *mount*, *dfshares* and *dfmounts*, you can determine the server resources to which you currently have access, the resources currently mounted on your system that are available for sharing by clients, as well as the local resources currently mounted on the various client systems.

6.4.1 Displaying shared local resources

You use the *share* command to share resources on your system with network clients; however, you can also use the command to display information about shared resources on your system. The command's syntax when used in this capacity is:

share

This command displays all local resources that can be accessed from clients.

6.4.2 Displaying mounted resources

Generally, you use the *mount* command to mount local and remote resources on your system. However, if you enter the command without arguments, it displays a list of local and remote resources that are currently mounted on your system. *mount* accesses the */etc/mnttab* file, which contains all the system's mount procedures.

6.4.3 Displaying available remote resources

dfshares lets you see which remote resources have been made available on NFS servers for sharing by clients. This command uses the following syntax:

dfshares [–h] [*server*[*server*...]]

–h indicates that a header should not be output. If this option is not given, then a header is output by default.

server ...
: is a list of server names that must be separated by spaces. If you use this option, only information about the resources available on the specified servers is displayed.

The header output by *dfshares* by default contains the following headings:

RESOURCE SERVER ACCESS TRANSPORT

The column named RESOURCE contains the names of resources as specified in the *mount* command. SERVER names the servers on which the resources are available. ACCESS shows the access rights defined for the various resources (rw = read and write, ro = read only). The column named TRANSPORT shows you

which transport service is being used.

In the case of NFS resources, the ACCESS and TRANSPORT columns contain only hyphens (-), as the command is unable to generate this information for NFS resources.

If you enter *dfshares* without arguments, all the resources shared by the local system are displayed.

Example 1

You want to find out which NFS resources are available on server lib in the DNS domain sales. You also want this resource information to have a header. Enter:

dfshares lib.sales

The following is displayed:

```
RESOURCE□□□□□□□SERVER    ACCESS     TRANSPORT
lib:/sales     lib       -          -
lib:/usr/mem   lib       -          -
lib:/usr/mod   lib       -          -
lib:/letters   lib       -          -
```

Example 2

You want to find out which NFS resources are available on server main. You want this resource information to appear without a header. Enter:

dfshares –h main

The following is displayed:

```
main:/export     main   -   -
main:/usr/man    main   -   -
main:/usr/share  main   -   -
```

6.4.4 Displaying resources mounted by clients

The *dfmounts* command shows you which local resources or resources of another NFS server have been mounted on clients. The command accepts the following syntax:

dfmounts [–h] [*server*[*server*...]]

–h specifies that output of the header be suppressed. If this option is not given, then a header is output by default.

server
is replaced by the name of an NFS server. If you use this option, *dfmounts* displays information only about the server or resource you specified.

If you enter *dfmounts* without arguments, the system displays all local resources currently mounted by remote systems. The *dfmounts* command displays a header consisting of the following column headers:

RESOURCE SERVER PATHNAME CLIENTS ...

RESOURCE is the name of the mounted resource; SERVER is the name of the system providing the resource; PATHNAME is the pathname of the shared resource (as given in the *share* command); and CLIENTS is a list (separated by commas) of clients that have mounted the resource.

Example 1

Your local machine is called lib; it resides in the DNS domain sales. You want to find out which local NFS resources are currently mounted. You want the display to include a header. Enter the following:

dfmounts

The system displays:

```
RESOURCE   SERVER   PATHNAME          CLIENTS
-          lib      /usr/share/conv89  dept120.sales
-          lib      /usr/share/proj1   dept122.sales
-          lib      /usr/share/proj2   dept122.sales
-          lib      /usr/share/train
```

Example 2

You want to see which NFS resources are currently mounted on the system called main. Enter:

dfmounts –h main

The system displays:

```
-    main      /export     dancer, runner
-    main      /usr/man    jogger
-    main      /usr/share  dancer, runner
```

6.5 Using the automounter

Resources shared through the NFS can be mounted using a method called "automounting". The *automount* program mounts and unmounts remote directories as required. Whenever a user on a client running the automounter invokes a command that needs to access a remote file, the shared resource to which that file belongs is mounted automatically. When a certain amount of time has elapsed without the resource being accessed, it is automatically unmounted.

Once the automounter is invoked, an administrator does not have to set up automatic mounting with the *vfstab* file, or use the *mount* and *umount* commands at the command line. All mounting is done automatically and transparently.

Mounting some resources with *automount* does not exclude the possibility of mounting others with *mount*; in fact, in a diskless machine you **must** mount root (*/*) and */usr* with *mount*.

This section explains how the automounter works and provides instructions for setting it up.

6.5.1 How the automounter works

Unlike *mount*, *automount* does not consult the file */etc/vfstab* for a list of resources to mount automatically. Rather, it consults a series of maps.

The automounter always mounts its resources under the directory */tmp_mnt*, and generates a symbolic pointer from the mount point specified in the map to the actual mount point under */tmp_mnt*. For instance, if a user wants to mount a remote directory *src* under */usr/src*, the actual mount point will be */tmp_mnt/usr/src*, and generates a symbolic pointer from */usr/src* to this mount point.

When *automount* is called, it starts a server daemon to monitor all the mount points entered in the maps. The daemon sleeps until a request is made to access the corresponding resource. The daemon then mounts the remote resource and creates a symbolic pointer between the requested mount point and the actual mount point under */tmp_mnt*.

When a certain period of time has passed without the link being touched (generally five minutes), the daemon unmounts the resource and returns to the wait state.

A server neither knows nor cares whether the resources it shares are being accessed via *mount* or *automount*. Nothing, therefore, needs to be done to a server to run the automounter.

A client, however, needs special files for the automounter. As mentioned earlier, *automount* does not consult */etc/vfstab*; rather, it consults a map file (or files) specified at the command line.

6.5.2 The automounter's maps

There are three basic kinds of automounter map:

- a master map
- a direct map
- an indirect map

The master map is a general map that refers to other, direct or indirect maps that contain more specific information. If you specify a master map when you call the automounter, the master map calls direct and indirect maps to get the information it needs to give to the *automount* command.

You can specify which master map is to be used when *automount* is started automatically by defining the MASTERMAP variable in the */etc/default/inet* file. The values you can enter are either the absolute path name of the master map file, or *nomap* if you do not want the automounter to be started automatically. If the master map file you have specified exists, the automounter is started by the NFS initialization script. A common value for the master map file is */etc/auto.master*. If you do not specify a value for MASTERMAP, the automounter is not started automatically.

A direct map contains all the information *automount* needs to perform the mount. Direct maps contain the **absolute** pathnames of mount points. It can be called directly by the automounter, or through a master map.

An indirect map allows you to specify alternative servers for a set of resources. It also allows you to specify resources to be mounted as a hierarchy under the same mount point. Indirect maps contain the **relative** pathnames of mount points. An indirect map can be used either directly by the automounter or via a master file.

You should create all your automounter maps in your */etc* directory, using any text editor.

Conventions for creating maps

When creating any of the automounter maps, follow these conventions:

- Use a backslash (\) before characters identified by the automounter's parser as special characters. For example, suppose you want to mount a directory whose name includes a colon, such as rc0:dk1. This resource name might result in the map entry

  /junk -ro vmsserver:rc0:dk1

  The second colon in this entry may cause problems (the colon is normally used as a separator between the machine name and the path name). To avoid a problem, the entry should look like this:

  /junk -ro vmsserver:rc0\:dk1
- Use double quotes around a resource name to hide white space.
- If you enter a comment into a map, make sure each line of the comment is preceded by #.
- If you include a long entry in a map, use backslashes to split the line into two or more shorter lines. For example, the entry

  /usr/frame -ro,soft mahagony:/usr/frame.3 balsa:/export/frame

  could be written as

  /usr/frame -ro,soft mahagony:/usr/frame.3 \
  balsa:/export/frame

6.5.2.1 Creating a master map

Each line in a master map has the syntax:

mount-point map [*mount-options*]

mount-point is the full pathname of a directory on which resources should be mounted; *map* is the name of the map that lists the resources to be mounted and their locations; and *mount-options* is a comma-separated list of options that regulates the mounting of the entries mentioned in the *map* (unless the *map* entries list other options).

The options that can be specified for *mount-options* are the same options that can be specified with the *mount* command, except for *fg* and *bg*. See the description of the *mount(1M)* command in the "Networking Reference Manual".

The *mount-point* can be any mount point for a remote resource. Here is a sample entry in a master map:

/usr/man /etc/libmap -ro

This entry tells the automounter to look in the indirect map /etc/libmap and to mount all the resources listed there on */usr/man* on the local system, if need be, with read-only access (the mount points in indirect maps appear as relative path names). However, if libmap indicates that a resource should be mounted read-write, this setting has priority.

If the master map calls a direct map, the *mount-point* parameter should be replaced by /– (the mount points in direct maps appear as absolute path names). This tells the automounter to mount the resources listed in the direct map as required.

Below is a sample master map:

```
#mount-point   map             mount-options
/usr/reports  /etc/reportmap    -rw,intr,secure
/usr/man      /etc/libmap       -ro
/-            /etc/direct.map   -ro,intr
```

6.5.2.2 Creating a direct map

All entries in a direct map have the syntax:

mount-point [*mount-options*] *resource*

mount-point is the **full** pathname of the mount point; *mount-options* is a comma-separated list of options that regulates the mounting of the resource specified in the entry; and *resource* is the location of the resource, specified as *server:pathname*.

The *mount-options* can be any of the options that can be specified with the *mount* command, except for the options *fg* and *bg*. You will find more information in the section entitled The mount command or in the description of the *mount(1M)* command in the "Networking Reference Manual".

The following is a sample entry in a direct map:

/usr/fun -ro,soft peach:/usr/games

This entry means that the remote resource /usr/games on the server named peach should be mounted for read-only access on the local mount point /usr/fun. The option *soft* specifies that NFS requests for this file system should be aborted if the server does not respond within a period of time.

Whenever a user tries to access a file or directory that is part of the usr/fun directory tree, the automounter mounts the resource from server peach onto the mount point /tmp_mnt/usr/fun on the local system and creates a symbolic link between /tmp_mnt/usr/fun and /usr/fun. The user is unaware that the mount operation is taking place, and the resource appears to the user to be at /usr/fun.

The following is a typical direct map:

```
/usr/local \
/bin   -ro,soft   maple:/export/local/sni3 \
/share -ro,soft   maple:/export/local/share \
/src   -ro,soft   maple:/export/local/src
/usr/man            -ro,soft   oak:/usr/man \
rose:/usr/man \
willow:/usr/man
/usr/games          -ro,soft   fir:/usr/games
/var/spool/news     -ro,soft   spruce:/var/spool/news
/usr/frame          -ro,soft   mahagony:/usr/frame1.3 \
balsa:/export/frame
```

Note that, in the first entry, several mount points and resources are specified. This option is discussed later in this chapter.

6.5.2.3 Creating an indirect map

Entries in an indirect map have the syntax:

mount-point [*mount-options*] *resource*

mount-point is the **relative** pathname of the directory that will be used as the mount point. This pathname needs to be prefixed to produce an **absolute** pathname for the mount point (via an entry in the master map or as a call option).

mount-options is a comma-separated list of options that regulates the mount.

resource is the location of the resource, specified as *server*:*pathname*.

For example, suppose one of the entries in the master map reads:

/usr/reports /etc/reportmap -rw,intr

Here, /etc/reportmap is the name of the indirect map that lists the remote resources to be mounted under /usr/reports.

Here is a typical indirect map:

```
#mount-point     mount-options  resource
willow                          willow:/home/willow
cypress                         cypress:/home/cypress
poplar                          poplar:/home/poplar
pine                            pine:/export/pine
apple                           apple:/export/home
ivy                             ivy:/home/ivy
peach           -rw,nosuid      peach:/export/home
```

Assume this map resides on the machine called oak. /home/willow/laura is entered in the */etc/passwd* file as the home directory of the user with the user name laura. When laura logs into the machine oak, the automounter mounts the /home/willow directory of the machine willow on the machine oak as /tmp_mnt/home/willow. If one of the subdirectories is laura, laura is in her home directory (see also the section entitled Specifying multiple mounts).

Any option in the indirect map overrides all options specified in the master map or on the command line.

6.5.2.4 Specifying multiple mounts

An entry in a direct or indirect map can describe multiple mounts, where identical or different resources can be mounted using identical or different mount options. In the sample map described in the section entitled Creating a direct map, the first entry (which is actually one long entry split into three lines) mounts the resources /export/local/sni3, /export/local/share and /export/local/src from the server ivy onto local directories */usr/local/bin*, */usr/local/share* and */usr/local/src* with the options *ro* and *soft*. The entry could also read:

```
/usr/local \
/bin    -ro,soft   ivy:/export/local/sni3 \
/share  -rw        willow:/usr/local/share \
/src    -ro,intr   oak:/home/jones/src
```

In this case, resources are mounted by three servers using different options.

6.5.2.5 Specifying hierarchical mounts

Multiple mounts can be hierarchical. When resources are mounted hierarchically, each resource is mounted on a subdirectory within another resource. When the root of the hierarchy is mounted (*/* in the example), the automounter mounts the entire hierarchy including all subdirectories at the same time. The following illustration shows a true hierarchical mounting:

```
/usr/local \
/       -rw,intr     peach:/export/local \
/bin    -ro,soft     ivy:/export/local/sni3 \
/share  -rw          willow:/usr/local/share \
/src    -ro,intr     oak:/home/huber/src
```

If we take the directory */usr/local*, the whole hierarchy is mounted thus:

Under */usr/local*□□□□□□□□□□peach:/export/local

Under */usr/local/bin*□□□□□ivy:/export/local/sni3 etc.

There is, however, no requirement that the first mount of a hierarchy be at the mount root. The automounter will issue *mkdir* commands to build a path to the first mount point if it is not at the mount root.

A hierarchical mount can be a problem if the server for the root of the hierarchy goes down; any attempt to unmount the lower branches will fail, since the unmounting has to proceed through the mount root, which also cannot be unmounted while its server is down.

6.5.2.6 Specifying optional mount operations

A mount entry in a direct or indirect map can include more than one entry in its *resource* field. This means that the mounting can be done from any of the resource locations specified.

Specifying several resources for a mount point makes sense only when you are mounting a resource for read-only access; it is information duplicated on several servers that should not be modified inconsistently. An example of a read-only resource you might mount from a variety of locations is online documentation. In a large network, the current online documentation may be available on more than one server. It doesn't matter which server you mount this resource from, as long as the server is up and running and has shared the resource. If the online documentation resides in a directory called */usr/man* on three different machines called oak, rose, and willow, you can mount it from any of these locations by specifying the following in the direct map:

/usr/man -ro,soft oak:/usr/man rose:/usr/man \
willow:/usr/man

This could also be expressed as a comma-separated list of servers, provided the pathname is the same on all servers:

/usr/man -ro,soft oak,rose,willow:/usr/man

When the resource in question is accessed, the automounter first checks to see which servers from the list are available. The first server to respond is selected, and an attempt is made to mount from it.

If the server goes down while the mount is in effect, the resource becomes unavailable. An option here is to wait five minutes until the auto-unmount takes place and try again; next time around, the automounter automatically chooses one of the other available servers. Another option is to use the *umount* command, inform the automounter of the change in the mount table, and retry the mount. See Updating the mount table later in this chapter for more information.

6.5.2.7 Mounting subdirectories

Until now we have used the form *server*:*pathname* to indicate a location in a map entry; however, you can also specify a subdirectory in the *resource* field, using the syntax

server:*pathname*:*directory*

Assume you have a master map on the machine oak that contains the following entry:

/home /etc/auto.home -rw, intr

Here */etc/auto.home* is the name of an indirect map that contains the entries to be mounted under */home*. Below is the *auto.home* map:

```
#mount-point   mount-options   resource
cypress                        cypress:/home/cypress
poplar                         poplar:/home/poplar
pine                           pine:/export/pine
apple                          apple:/export/home
ivy                            ivy:/home/ivy
peach          -rw,nosuid      peach:/export/home
john                           willow:/home/willow:john
mary                           willow:/home/willow:mary
joe                            willow:/home/willow:joe
```

Look at the first entry in the map, and assume the user adam has his home directory on the machine cypress. /home/cypress/adam is entered in the */etc/passwd* file as the home directory of the user with the user name adam. If a user logs into the machine oak under the user name adam, the automounter mounts the /home/cypress directory residing on the server cypress (as /tmp_mnt/home/cypress). If one of the subdirectories is adam, adam will be in his home directory. Because of the options specified in oak's master map, adam's home directory is mounted for read and write access (rw) and is interruptible (intr).

Now look at the last three entries in the *resource* field of the indirect map. Note that these entries refer to the same resource (willow:/home/willow). Each resource is followed by the name of a subdirectory (john, mary, and joe).

If a user logs in to the machine oak and requests access to the /home/willow/john directory on the machine willow, the automounter mounts the resource willow:/home/willow. It then places a symbolic link between /tmp_mnt/home/willow/john and /home/john.

If the user mary then tries to access the home directory from the machine oak, the automounter sees that willow:/home/willow is already mounted, so all it has to do is create the link between /tmp_mnt/home/willow/mary and /home/mary.

In general, it's a good idea to specify a subdirectory entry in the *resource* field when several map entries refer to the same resource.

6.5.3 Using substitutions to simplify entries

If you have a map with a lot of subdirectories specified, as in the following indirect map

```
#mount-point  mount-options  resource
john                    willow:/home/willow:john
mary                    willow:/home/willow:mary
joe                     willow:/home/willow:joe
able                    pine:/export/home:able
baker                   peach:/export/home:baker
```

you should consider using string substitutions. The ampersand character (&) can be used to substitute the mount point wherever it appears. Using the ampersand, the above map would look like this:

```
#mount-point  mount-options   resource
john                          willow:/home/willow:&
mary                          willow:/home/willow:&
joe                           willow:/home/willow:&
able                          pine:/export/home:&
baker                         peach:/export/home:&
```

If the name of the server is the same as the mount point itself, as in the following case:

```
#mount-point  mount-options   resource
willow                        willow:/home/willow
peach                         peach:/home/peach
pine                          pine:/home/pine
oak                           oak:/home/oak
poplar                        poplar:/home/poplar
```

the use of the ampersand would result in:

```
#mount-point  mount-options   resource
willow                        &:/home/&
peach                         &:/home/&
pine                          &:/home/&
oak                           &:/home/&
poplar                        &:/home/&
```

If all entries in a map have the same format, you can use the catch-all substitute character, the asterisk (*), as in the following example:

```
#mount-point  mount-options   resource
oak                           &:/export/&
poplar                        &:/export/&
*                             &:/home/&
```

Once the automounter reads the catch-all key (*), it does not continue to read the map.

6.5.3.1 Using environment variables

You can use the value of an environmental variable by prefixing a dollar sign ($) to its name. Braces can also be used to delimit the name of the variable from appended letters or digits.

The environmental variables can be inherited from the system environment or can be defined explicitly with the *-D* option. For example, if you want each client to mount client-specific files from different servers in the network, you could create a specific map for each client according to its name, so that the relevant line for the machine oak would be:

/mystuff cypress,ivy,balsa:/export/hostfiles/oak

and for willow:

/mystuff cypress,ivy,balsa:/export/hostfiles/willow

This scheme is viable within a small network, but maintaining this kind of machine-specific map across a large network usually isn't feasible. The solution in this case would be to invoke the automounter with a command line similar to the following:

automount -D HOST='uname -n' .....

and have the entry in the direct map read:

/mystuff cypress,ivy,balsa:/export/hostfiles/$HOST

Now each machine would find its own files in the mystuff directory, and the task of centrally administering and distributing the maps becomes easier.

6.5.4 The -hosts map

Specifying *-hosts* as a map name in the master map or as an argument when calling the automounter has the following effect:

If, for example, the line

/net -hosts *mount_options*

is specified in the master map, the path which immediately follows */net* is taken to be the name of a machine. In this case, the automounter looks to see which resources are shared by that machine, mounts them at the mount point */tmp_mnt/net/hostname*, and generates as usual a symbolic link between */net/hostname* and */tmp_mnt/net/hostname*.

Using *-hosts* is, however, not always recommended if the server shares a large number of resources. In this case, the automounter would be forced to carry out numerous mounts which would, on the one hand, take a considerable amount of time and, on the other, may result in a good deal of possibly unnecessary data being provided for short periods of time.

6.5.5 Starting the automounter

Once the maps are written, you should make sure that there are no equivalent entries in */etc/vfstab*, and that all the entries in the maps refer to NFS shared resources.

The syntax to invoke the automounter is:

automount□[–mnTv] [–D *name*=*vvalue*] [–f *master-file*] \
□□□□□□□□□□□[–M *mount-point*] [–t *sub-options*] [-o hard| intr] \
□□□□□□□□□□□[*directory map* [–*mount-options*]] ...

You will find a detailed description of the options in the description of the *automount(1M)* command in the "Networking Reference Manual". You can use all the NFS options supported by *mount*, with the exception of *bg* (background) and *fg* (foreground), which do not apply.

The default mount point for all mounts is */tmp_mnt*. Like the other names, this is an arbitrary name. It can be changed when you enter the automount command with the *-M* option. For example,

automount -M /auto ...

causes all mounts to happen under the directory /auto, which the automounter creates if it doesn't already exist.

The automounter can be invoked in one of the following ways:

1. You can specify all the arguments in the command line without reference to the master map:

   automount /net -hosts /home /etc/*indirect_map* \
   □□□□□□□□□□□-rw,intr /- /etc/*direct_map* -ro,intr

2. You can specify all the arguments in the master map and instruct the automounter to look in it for instructions:

   automount -f /etc/*master_map*

3. You can specify more mount points and options in addition to those mentioned in the master map:
   automount -f /etc/*master_map* /src /etc/auto.src -ro,soft
4. By entering *-null*, you can get the automounter to ignore one of the entries in the master map (particularly useful if you are using a map that you cannot modify but does not meet the needs of your machine):
   automount -f /usr/lib/*master_map* /home -null
5. You can override an entry in the master map by specifying a different indirect map on the command line, as follows:
   automount -f /usr/lib/*master_map* /home \
   □□□□□□□□□□□/myown/*indirect_map* -rw,intr
   This command tells the automounter to mount the */home* resource according to instructions in /myown/*indirect_map*, and to ignore the reference to the regular indirect map in the master map for this resource.

6.5.6 Updating the mount table

If you use *umount* to unmount explicitly one of the automounted resources, have the automounter re-read the */etc/mnttab* file. This can be accomplished by sending signal 1 to the automount daemon, as shown below.

kill -1 PID

PID is replaced by the process ID of the automount daemon automount. It can be determined using the command *ps -ef*.

6.5.7 Modifying the maps

You can modify the maps at any time; however, the automounter looks at the master and indirect maps only when it is invoked. Changes to the master and indirect maps are only effective after the automounter has been restarted.Changes to a direct map take effect the next time the automounter mounts the resource named in the modified entry. The adding of new entries to the direct map is immediately effective, while deletion of entries is only effective if the relevant resource has been unmounted by the automounter (after a timeout).

Changes to entries in direct maps (new mount options or similar) are likewise only effective when the resource is mounted again, in other words possibly only if the resource is first unmounted with its existing options. If the resource is currently in use, the unmounting is delayed accordingly. This gives rise to an inconsistency between the automounter status and the direct map.

6.5.8 Restoring the mount table

If *automount* is terminated with signal number 9, the directories monitored by *automount* remain in an inconsistent state, which means that they can no longer be accessed. By calling *automount* again with the same parameters, access to these directories is restored: *automount* performs a recovery. To do this, the required recovery information is stored in the kernel and in a */tmp/automount.pid* file when *automount* is started. *pid* is the process number of *automount*.

6.5.9 Terminating automount

automount is terminated correctly with:
kill -15 *PID*

6.6 PC-NFS - NFS servers for PC clients

Reliant UNIX 5.44 systems and later operating as NFS servers can also serve clients located on different PCs. A daemon called *pcnfsd* is needed to do this. *pcnfsd* is an RPC server that supports ONC (Open Network Computing) clients on PCs (DOS, OS/2, Macintosh and other systems).

The daemon *rpc.pcnfsd* is started from */etc/init.d/nfs*. It reads the */etc/pcnfsd.conf* configuration file, if it exists.

To call the daemon, enter the following command:

/usr/sbin/rpc.pcnfsd

The inquiries handled by *pcnfsd* can be classified into three categories: authorization, printing and others. Only authorization and print services are of any significance for network administration.

6.6.1 Authorization checking

When *pcnfsd* receives a PCNFSD_AUTH or PCNFSD2_AUTH inquiry, it logs in the user by confirming the user and the password and then returns the relevant user name, group IDs (*gids*), home directory and user mask (*umask*).

The default setting is for *pcnfsd* only to allow authorization inquiries from users whose user ID is between 101 and 60002 (this is the range for non-administrative logins). To override this, you must enter the following line in the */etc/pcnfsd.conf* file:

uidrange range[,*range*] ..

The format for each *range* is

uid, or *uid-uid*

uid-uid indicates an inclusive range.

6.6.2 Printing

pcnfsd supports an NFS-based print model for transferring the print data from the client to the server. The client system sends a PCNFSD_PR_INIT or PCNFSD2_PR_INIT inquiry, and the server supplies the path of the spool directory which the client can use and which is made available by the NFS. *pcnfsd* creates a subdirectory for each of its clients. The parent directory is normally */var/spool/pcnfs*, and the subdirectory is the machine name of the client system. If you wish to use a different parent directory, you should insert the following line in the */etc/pcnfsd.conf* file:

spooldir *path*

When the client has accessed the spool directory with the NFS and has transferred print data to a file in this directory, it outputs a PCNFSD_PR_START or PCNFSD2_PR_START inquiry. *pcnfsd* processes this and most other print-related inquiries by constructing a command based on the print services of the server operating system and having it executed using the identity of the PC user. Because this entails privileges for setting user IDs, *pcnfsd* must be started in the root directory.

Each print inquiry by the client contains the name of the printer to be used. If you wish to process print data in a way which does not correspond to the default, you should define a new printer and instruct the client to use this printer. *pcnfsd* also contains a mechanism for defining virtual printers which are only known to *pcnfsd* clients. Each printer is defined by entering the following line in the */etc/pcnfsd.conf* file:

printer *name alias-for command*

name is the name of the printer you wish to define. *alias-for* is the name of the "real" printer corresponding to this printer, for example, the inquiry to display the queue for *name* is translated into the corresponding inquiry for the *alias-for* printer. If you have defined a printer so that it does not correspond to a "real" printer, you can use a - for this field (see the following sample definition for the test printer). *command* is a command which is executed when a file is printed on *name*. This command is always executed from the Bourne shell, in which */bin/sh* uses option *-c*. For complex operations, you should write an executable shell program which is executed in *command*.

The following expressions are used in *command*:

$*FILE*
: is the full pathname of the print data file. The connection to the file is cleared again when the command has been executed.

$*USER*
: is the name of the user logged into the client system.

$*HOST*
: is the machine name of the client system.

$*OPT* are the options specified by the client system.

Please note the following */etc/pcnfsd.conf* sample file:

printer rotated lw /usr/local/bin/enscript -2r $FILE

printer test - /usr/bin/cp $FILE /usr/tmp/$HOST-$USER

If a client system makes a request to the printer rotated, the *enscript* utility is called in order to prepare the *$FILE* file. In this case, the effect of option *-2r* is for the file to be output on the preset postscript printer in two-column rotation format. If the client requests a list of printer queues for the rotated printer, the *pcnfsd* daemon translates this into a request for a list by the lw printer.

The test printer is used for test purposes. All files sent to this printer are copied to */usr/tmp*. All requests asking the test printer to list the queue, check the status, etc. are rejected because alias-for was specified as -.

Reconfiguration

pcnfsd recognizes that printers have been added or removed and forms a new list of valid printers. To do this, the change time of */etc/lp/printers* is checked. The */etc/pcnfsd.conf* file is not monitored for updates; if you change this file, *pcnfsd* must be cancelled and restarted in order to record the changes.

6.7 Other services relevant to the NFS

This section deals with:

- the network lock manager
- the status monitor
- asynchronous write jobs using the NFS

6.7.1 The network lock manager

Processes which work with shared files synchronize access to such files by means of a locking function. Locks on files or records are set, released or queried using the system call *fcntl()*. The NFS protocol does not support the setting of locks. The reason for this is that the NFS protocol is created "without a status": if the NFS client initiates an action on the server, this action does not create a server status which would be lost if the server crashed. This means that either

- no status at all is generated on the server (e.g. when reading data) or
- a "permanent" status is generated on the server (e.g. when creating or deleting files). A permanent status will continue to exist after the server crashes.

From the point of view of an NFS client, a server which can start up again after a crash works extremely slowly. Similarly, a client which restarts after a crash is simply regarded by the NFS server as a client which has not issued any jobs for a long time. This convenient restart function makes the NFS very robust and is due to the fact that the NFS protocol has no status.

It is not possible, however, to set locks with a protocol which has no status. Locks must be registered on the NFS server and are lost after a server crashes (unless each lock is stored in a non-transient memory). For this reason, work which involves locks is carried out in a separate protocol called NLM (Network Lock Manager).

The Network Lock Manager supports so-called "advisory" locks, i.e. the locks only work if all user programs are working with locks for access to the relevant files or file sections (see also the Section "Synchronization of file accesses"). The *lockd-srv* program implements the NLM protocol on the server side. On the client side, the NLM is implemented in the kernel. Furthermore, the client can also use utilities which are provided by the *lockd-clnt* process which runs on the client.

When you are working with locks using the NFS, it is of vital importance to make sure that the client processes buffer all access to shared files. Setting NFS locks ensures that the NFS client does not buffer read and write data. However, when both buffered and unbuffered accesses are used (i.e. accesses which have not set a lock), the consistency of the shared files can no longer be guaranteed. There is an exception to this: when all client processes run on the same client machine (i.e. they also share the buffer).

If an NLM client or server goes down, clean-ups must be carried out on the relevant partner when the system is restarted. A server must delete all the locks of a client which is restarting. When a server is restarted, a client must set its locks again in order to restore the status which applied before the server crashed.

It can be necessary to restart systems for a whole range of network service states. For this reason, restarts are registered on the NSM (Network Status Monitor) in a service which is independent of NLM. The NSM then ensures that these are executed when a crashed partner is restarted. The NSM protocol is implemented in the

statd program. The NLM (*lockd-srv/lockd-clnt*) is so far the only service that uses the NSM (*statd*) (see also the descriptions of *lockd-srv* and *lock-clnt* in the "Networking Reference Manual").

6.7.2 The status monitor

The status monitor, implemented as *statd(1M)*, is very general and can also be used to support other kinds of state-oriented network service and application. Normally, recovering lost status information is one of the most difficult aspects of network application development. The status monitor makes it more or less routine.

The status monitor works by providing a general framework for collecting network status information. Implemented as a daemon that runs on all network machines, it provides a simple protocol that allows applications to monitor easily the status of other machines.

Its use improves overall robustness, and avoids situations in which applications running on different machines (or even on the same machine) come to disagree about the status of a site - a potentially dangerous situation that can lead to inconsistencies in many applications.

Applications using the status monitor do so by registering with it the machines that they are interested in. The monitor then tracks the status of those machines, and when one of them crashes (or, to be more precise, when one of them restarts after a crash), it notifies the interested applications of the restart. Then they take whatever actions are necessary to reestablish a consistent state.

This approach provides the following advantages:

- Only applications that use stateful services must pay the overhead - in time and in code - of dealing with the status monitor.
- The implementation of stateful network applications is eased, since the status monitor shields application developers from the complexity of the network.

For more information about the status monitor, see description of *statd(1M)* in the "Networking Reference Manual".

6.7.3 Asynchronous writes using the NFS

In the case of a synchronous write job, the server sends an acknowledgment the moment data is written to disk. With an asynchronous write, however, the server sends an acknowledgment when it receives the data, i.e. before data has been written to disk.

In SINIX V5.42 (that is, using NFS V2), data could be written asynchronously using the *share* option *async*. But systems which only supported NFS V2 were subject to data loss in the event of a server crash when this option was used. Asynchronous writing is not provided for in the NFS protocol of Version 2.

Reliant UNIX 5.43 or later now includes an implementation of NFS Version 3, which supports a safer and more reliable form of asynchronous writing.

NFS V3 uses a new procedure to prevent data loss in the event of a server crashing during an asynchronous write job. This might occur if the server's buffer still contained data at the time of the crash.

NFS V3 gets each client to issue a COMMIT-RPC for all asynchronous write jobs sent. The RPC ensures that any data on the server is first written to disk and, thus, stabilized. The client must take note of any jobs that are not yet flagged as stabilized, and repeat them in the event of a server crash.

The COMMIT-RPC uses a daemon called *commitd* on the client to stabilize data written asynchronously. This daemon is generated each time a file system is mounted. *commitd* daemons are retained in reserve for further mount operations even after the file system in question has been unmounted.

Stabilize runs are carried out at intervals by the client. These contain a file's handle, offset and length to identify each write job. A file handle is network-wide unique identifier for a file in a resource shared by a server. It is used internally in the NFS as a file descriptor. In the event of a server crash, all write jobs for this server that have not yet been stabilized need to be repeated in their entirety.

There is a system process available on the client called *nfs_fsflush* for stabilizing data written asynchronously with the COMMIT-RPC. This process is generated when the file system is mounted and disappears again when the system is unmounted.

The stabilization requests are issued periodically by the client after an excessive number of unstable write

requests have been collected. They contain an id for the associated write job based on the file handle, offset and length. A file handle is a unique network-wide identifier for a file within a resource made available by the server. The handle is used internally within NFS like a file descriptor. If a server crash is detected, write jobs for this server that have not yet been stabilized must be repeated completely.

6.8 Optimizing operation of NFS

This section describes how operation of NFS can be optimized in various areas. These areas are:

- RPCs
- block sizes
- buffered read and write operations
- asynchronous writing via NFS
- number of *nfsd* processes
- limit values for NFS server input
- NFS locks

6.8.1 NFS and RPC

File access via NFS is considerably slower than local file access, because the access must be sent to the server via the network as a RPC request (RPC = Remote Procedure Call), where it is then executed. The server returns the result of the access to the client as an RPC acknowledgment.

The topics described below are important factors influencing NFS performance.

6.8.1.1 Client caches

In order to reduce the number of RPCs, different data is buffered on the client:

- file attributes
  (NFS-specific attribute cache)
- mapping of filenames to V-nodes
  (Directory Name Lookup Cache, used by all file systems)
- user data
  (Page Cache, used by all file systems)

These cache areas can be influenced in that, for example, the size of the DNLC or Page Cache can be set with specific kernel parameters (NCSIZE or SM_SIZE, see the Section "Tuning parameters"), or the characteristics of the attribute cache can be defined with the NFS command *mount (1M-nfs)*.

The following rule applies on the server side: Measures implemented to improve the performance of a local file system also improve performance when this file system is exported for NFS clients.

6.8.1.2 Timeouts and retransmissions

When the client sends an RPC request, it waits for a specified period for the response. If the response is not received within this time, the request is repeated, whereby the receive timeout is doubled. This procedure continues until the maximum number of retransmissions is reached (*mount*, option *retrans*).

Timeouts and send retransmissions are an indicator of performance problems. You can display them with *nfsstat -cr*. Other statistics – *netstat*, *etherstat* – show the possible causes (buffer overflow, CRC errors, etc.).

6.8.2 Block sizes

The greatest network loads in NFS are normally created by read and write jobs. These jobs therefore considerably impact performance. The data is transferred in blocks of 8 Kbytes in the case of NFS V2 and 16 Kbytes (actual) or 32 Kbytes (nominal) with NFS V3.

With *mount*, options *rsize* and *wsize*, block sizes that are smaller than the above maximum values can be set. This is useful where the risk of packet loss is very high, e.g. because the line quality between the client and the server is very poor. On an Ethernet, 8 KB blocks would be fragmented into 6 Ethernet frames. If only one of these blocks were lost, the client would transfer the entire job again.

6.8.3 Buffered NFS read and write access

NFS read access is generally faster than NFS write access. The Page Cache on the file system is used by NFS read access on both the server and the client. NFS write access is normally buffered on the client by page size. If an entire block was written, the client sends a write job to the server. NFS V2 requires that the complete write data is available on the server disk before the server can send an RPC response to the client. This is **the** major performance bottleneck NFS V2!

NFS V3 allows the server to buffer the write data and send the response before all of the data is actually available on the disk. In terms of write access, NFS V3 therefore performs much better than V2.

6.8.4 Asynchronous write jobs

All NFS write jobs are asynchronous if the applications do not specifically require synchronous write jobs. "Asynchronous" means that a client process returns a *write()* system call after it has activated the write job and not when it has actually completed the job. An NFS client delegates the job to one of the *biod* processes.

6.8.5 Number of nfsd processes

It is difficult to make a general recommendation for a specific number, but 16-20 *nfsd* processes should be sufficient.

6.8.6 The NFS server receive buffer

All *nfsd* processes use the same socket (i.e. an internal UDP endpoint of the system kernel with the port number 2049). In other words, all NFS server communication is conducted using the same endpoint. The maximum number of requests in the receive buffer is defined in the */etc/conf/pack.d/nfs/space.c* file by the *nfsrpc_threshold* variable. You can check the number of lost NFS requests with the *netstat -sr* command (drops). If retransmitted NFS requests are lost, the value for *nfsrpc_threshold* should be increased.

6.8.7 NFS locks

NFS locks degrade performance, because the client can no longer buffer the locked areas. If there is only one client system whose processes are to be synchronized through the use of locks, *mount*, option *locallocks* should be specified. This will prevent performance degradation caused by NFS locks.

6.9 Error analysis and diagnostics

This section describes problems that may occur on machines using NFS services. Included are

- An overview of the NFS events
- Localizing NFS problems
- Steps to be taken in the event of server problems
- Steps to be taken in the event of client problems
- Steps to be taken in the event of a program failing
- Analyzing automounter problems
- NFS statistics using *nfsstat*

Before trying to clear NFS problems, you need some understanding of the issues involved. The information in this chapter contains enough technical details to give experienced network administrators a thorough picture

of what is happening with their machines. If you do not yet have this level of expertise, note that it is not important to understand fully all the daemons, system calls, and files.

However, you should be able to at least recognize their names and functions. For more information on the subjects dealt with in this chapter, please refer to the descriptions of the following commands and daemons in the "Networking Reference Manual":
mount(1M),

share(1M)

, mountd(1M),

nfsd(1M),

nfsstat(1M),

biod(1M).

6.9.1 An overview of the mount process

This section describes the remote mount process. It is not critical that you understand in depth how the daemons mentioned here work; however, you need to know that they exist, because you may need to restart them should they stop for any reason.

Here is the sequence of events when you first enter run level 2.

1. The appropriate script in */etc/init.d/nfs* starts the *mountd* daemon, and several *nfsd* server processes (the default is four).
2. The same script executes the *shareall* program, which reads the server's */etc/dfs/dfstab* file, then tells the kernel which resources the server can share and what access restrictions - if any - are on these files.
3. The script starts the *lockd-clnt* and *lockd-srv* lock manager processes, as well as the status monitor *statd*.
4. The script starts several *biod* client processes (the default is four).
5. The same script starts the *mountall* program, which reads the client's *vfstab* file and mounts all NFS-type files mentioned there in a manner similar to an explicit mount.

Here is the sequence of events when an administrator mounts a resource during a work session, using the command line:

1. The administrator enters a command, such as

   mount -F nfs -o intr,ro,soft dancer:/usr/src /usr/src/dancer.src
2. The *mount* command validates that the administrator has superuser permission, that the mount point is a full pathname, and that there is a file */usr/lib/nfs/mount*. If all three conditions are valid, */sbin/mount* then passes all the relevant arguments and options to */usr/lib/nfs/mount*, which takes control of the process (when we say *mount* from now on in this section, we will be referring to this last file.)
3. *mount* then opens */etc/mnttab* and checks that the mount was not done automatically at the start of the work session.
4. *mount* parses the argument *resource* into the machine called dancer and the remote directory */usr/src*.
5. *mount* calls dancer's *rpcbind* to get the port number of the *mountd* daemon.
6. *mount* calls the *mountd* daemon on dancer, sends the pathname */usr/src*, and requests a file handle for the directory. A handle is a file's identification within a resource shared by the server. The ID is unique to the file throughout the network. Within the NFS, the handle is used as a file descriptor.
7. The server's *mountd* daemon handles the client's mount requests. If the directory */usr/src* is available to the client, the *mountd* daemon passes the pathname */usr/src* with the system call NFS_GETFH, gets the file handle and passes this to the client's mount process.
8. *mount* checks if /usr/src/dancer.src is a directory.
9. *mount* passes the file handle and pathname /usr/src/dancer.src to the client's system kernel with the system call *mount(2)*.
10. The client system kernel enters the file handle at mount point /usr/src/dancer.src as the root of a mounted file system.
11. The client kernel sends a *statfs(2)* for the */usr/src* resource to the NFS daemon *nfsd* on the server.
12. The *mount(2)* system call sent by *mount* returns.
13. *mount* opens */etc/mnttab* and adds an appropriate entry to the end, reflecting the new addition to the list of mounted files.
14. When the client kernel does a file operation after the resource is mounted, it sends the NFS information directly to the server with an RPC request. Here, an *nfsd* daemon process handles the request.
15. The *nfsd* daemon knows how a resource is shared from the information sent to the server's kernel by *share* and allows the client to access the resource according to its permissions.

6.9.2 Localizing NFS problems

When tracking down an NFS problem, keep in mind that there are three main points of possible failure: the server, the client, or the network itself. The strategy outlined in this section tries to isolate each individual component to find the one that is not working.

The *mountd* daemon process must be active on the server for a remote mount to succeed.

If the daemon isn't running, log in as superuser and type /usr/lib/nfs/mountd. Remote mounts also require several *nfsd* server processes (the default is four) to be running on the NFS servers.

If they aren't, log in as superuser and type:

/usr/lib/nfs/nfsd 4

Although the client's *biod* NFS processes are not absolutely necessary for the NFS to work, they do improve performance.

If these daemons are not running, log in as superuser and type:

/usr/lib/nfs/biod 4

All the daemon processes needed to run the NFS are started by default by script */etc/init.d.nfs* the moment you switch to run level 2. The daemon *rpcbind* must always by running before any NFS services are invoked. By

default, this happens via the file */etc/init.d/rpc*.

6.9.3 Steps to be taken in the event of server problems

When the server has problems, programs that access hard mounted remote files will fail differently than those that access soft mounted remote files. Hard mounted remote resources cause the client's kernel to retry the requests until the server responds again. Soft mounted remote resources cause the client's system calls to return an error after trying for a while. *mount* is like any other program: if the server for a remote resource fails to respond, and the *hard* option has been used, the kernel retries the mount until it succeeds. When you use *mount* with the *bg* option, it retries the mount in the background if the first mount attempt fails.

When a resource is hard mounted, any program that tries to access it hangs if the server fails to respond. In this case, the NFS displays the message

NFS server *hostname* not responding, still trying

on the console. When the server finally responds, the message

NFS server *hostname* ok

appears on the console.

It is recommended to mount a resource interruptible (i.e., *-o intr*) so that if a program hangs due to the server not responding, the program can be interrupted.

A program accessing a soft mounted resource whose server is not responding may or may not check the return conditions. If it does, it prints an error message in the form

. . .*hostname* server not responding: RPC: Timed out

If a client is having NFS trouble, check first to make sure the server is up and running. From the client, type

/usr/sbin/rpcinfo *server_name*

to see if the server is up. If it is up and running, it prints a list of program, version, protocol, and port numbers, similar to the following:

```
program  version  netid     address        service  owner
100000     3      udp       0.0.0.0.0.111  rpcbind  superuser
100000     2      udp       0.0.0.0.0.111  rpcbind  superuser
100000     3      tcp       0.0.0.0.0.111  rpcbind  superuser
100000     2      tcp       0.0.0.0.0.111  rpcbind  superuser
100000     3      ticotsor  sfrjn.rpc      rpcbind  superuser
100000     3      ticots    sfrjn.rpc      rpcbind  superuser
100000     3      ticlts    sfrjn.rpc      rpcbind  superuser
```

If the server fails to print a list, try to log in at the server's console. If you can log in, check to make sure the *rpcbind* daemon is running on the server.

If the server is up but your machine cannot communicate with it, check the network connections between your machine and the server.

6.9.4 Steps to be taken in the event of client problems

This section deals with problems related to mounting. Any step in the remote mounting process can fail - some of them in more than one way. Below are the error messages you may see and detailed descriptions of the failures associated with each error message. *mount* can get its parameters explicitly from the command line or from */etc/vfstab*.

The examples below assume command line arguments, but the same debugging techniques work if mounting is done automatically through */etc/vfstab*.

- mount: ... server not responding: RPC_PMAP_FAILURE - RPC_TIMED_OUT

  Either the server sharing the resource you are trying to mount is down or in the wrong run level, or its *rpcbind* process is dead or hung. You can check the server's current run level by entering the following command on the server:

  who -r

If the server is in run level 2, try switching to another run level and then back again, or try rebooting the server to restart *rpcbind*. Try to log in to the server from your machine, using the *rlogin* command. If you can't log in, but the server is up, try to log in to another remote machine to check your network connection. If that connection is working, check the server's network connection.

- mount: ... server not responding: RPC_PROG_NOT_REGISTERED

  mount got through to *rpcbind*, but the NFS daemon *mountd* is not registered there. Check the server's run level and make sure its daemon processes (*mountd*, *nfsd*, *statd*) are running.

- mount: ...: No such file or directory

  Either the remote directory or the local directory does not exist. Check the spelling of the directory names. Use *ls* on both directories.

- mount: not in share list for ...

  Your machine name is not in the list of clients allowed access to the resource you want to mount. From the client, you can display the server's share list by entering

  dfshares *server*

  If the resource you want is not in the list, log in to the server and run the *share* command without options.

- mount: ...: Permission denied

  This message indicates that the user does not have the appropriate permissions or that some authentication failed on the server. The reason for this is either the name of your system is not entered in the server share list or the server does not believe you are who you say you are. Check the server's */etc/dfs/sharetab* file.

- mount: ...: Not a directory

  Either the remote path or the local path is not a directory. Check the spelling in your pathname, and try to run *ls* to check whether the directories exist.

The *mount* command hangs indefinitely if there are no *nfsd* daemons running on the NFS server. This happens when there is no */etc/dfs/dfstab* file on the server when it enters run level 2. To clear the problem, restart the *nfsd* daemons by typing

/usr/lib/nfs/nfsd 4

The *statd* status monitor may report the following:

cannot talk to statd at *machine_name*

This can happen when a change to the machine name on the server is not carried out properly.

The *sysadm* menu system should always be used to change the machine name; this is the only way to guarantee that all the necessary entries are made. If inconsistencies occur, *rpcbind* cannot start and thus neither can any other RPC service. You can deal with this problem by entering the new machine name instead of the old one in the */etc/net/*/hosts* files.

To render the status monitor operational again, terminate the */usr/lib/nfs/statd* process and delete the files */etc/sm/old_machine_name* and/or */etc/sm.bak/old_machine_name*. Then restart the *usr/lib/nfs/statd* process.

6.9.5 Steps to be taken in the event of a program failing

If programs hang while doing file-related work, your NFS server may be dead. You may see the following message on your console:

NFS server *hostname* not responding, still trying

This message indicates that NFS server *hostname* is down, or that there is a problem with the server or with the network.

If your machine hangs completely, check the server(s) from which you mounted the resource. If one or more are down, do not be concerned. When the server comes back up, programs resume automatically. No files are destroyed.

If you mount a resource using the *soft* option and the server dies, other work should not be affected. Programs that time out trying to access soft mounted remote files will fail with errno ETIMEDOUT, but you should still be able to access other resources.

If all servers are running, ask someone else using these same servers if they are having trouble. If more than one machine is having problems getting service, there is a problem with the server. Log in to the server. Run *ps -ef* to see if *nfsd* is running. If it is not, it may be that the server has been taken to a run level that does not support file sharing. Use *who -r* to obtain the server's current run level.

If other systems seem to be up and running, check your network connection and the connection of the server.

If programs on the client are hung but the server is up, NFS requests to the server other than reads and writes are succeeding, and messages of the form

xdr_opaque: encode FAILED

are appearing on either the client or the server console, you may be requesting more data than the underlying transport provider can provide. Try remounting the file system using the *-o rsize=nnn,wsize=nnn* options to *mount* to restrict the request sizes the client will generate.

Fixing a machine that hangs part way through booting

If your machine boots normally, then hangs when it tries to mount resources automatically, most likely one or more servers are down. Use *init* to go to single user mode or to a run level that does not mount remote resources automatically. Then start the appropriate daemons in the background and use the *mount* command to mount each resource usually mounted automatically through the */etc/vfstab* file. By mounting resources one at a time, you can determine which server is down.

If you cannot mount any of your resources, most likely your network connection is bad.

Improving access time

If access to remote files seems unusually slow, type

ps -ef

on the server to be sure that it is not being affected adversely by a runaway daemon. If there is nothing unusual in the display, and other clients are getting good response, make sure your *biod* daemons are running. At the client, type

ps -ef | grep biod

Look for *biod* daemons in the display, then enter the command again. If the *biod*s do not accumulate excessive CPU time, they are probably hung. If they are dead or hung, follow these steps:

1. Kill the daemon processes by typing:

   kill -9 *pid1 pid2 pid3 pid4*
2. Restart the *biod* daemons by typing:

   /usr/lib/nfs/biod 4

If the *biod* processes are running, check your network connection. The command *netstat -i* can help you determine if your system is losing packets. You will find more information on *netstat(1M)* in the "Networking Reference Manual".

If you are experiencing problems with some of your links across the network, it may be helpful to reduce the size of your NFS data transfer jobs (using the mount options *rsize=nnn,wsize=nnn*). By reducing the number of retransmissions necessary, you may well find that data throughput improves.

6.9.6 Analyzing automounter problems

The following are error messages you may see if the automounter fails.

- *mapname*: Not found

  The required map cannot be located. This message is produced only when the *-v* (verbose) option is given. Check the spelling and pathname of the map name.
- dir *mountpoint* must start with '/'

  The automounter mount point must be given as full pathname. Check the spelling and pathname of the mount point.
- *mountpoint*: Not a directory

  The *mountpoint* exists but it is not a directory. Check the spelling and pathname of the mount point.
- hierarchical mountpoint: *mountpoint*

  The automounter will not allow itself to be mounted within an automounted directory. You will have to think of another strategy.
- WARNING: *mountpoint* not empty!

  The mount point is not an empty directory. This message is produced only when the *-v* (verbose) option is given, and it is only a warning. All it means is that the previous contents of *mountpoint* will not be accessible.
- Can't mount *mountpoint*: *reason*

  The automounter cannot mount itself at *mountpoint*. The *reason* should be self-explanatory.

- *hostname*:*filesystem* already mounted on *mountpoint*

  The automounter is attempting to mount a resource on a mount point, but the resource is already mounted on that mount point. This happens if an entry in */etc/vfstab* is duplicated in an automounter map (either by accident or because the output of *mount -p* was redirected to *vfstab*). Delete one of the redundant entries.
- WARNING: *hostname*:*filesystem* already mounted on *mountpoint*

  The resource *filesystem* of the NFS server *hostname* is already mounted in the directory *mountpoint*.

- couldn't create *directory*: *reason*

  The system could not create a directory. The *reason* should be self-explanatory.
- bad entry in map *mapname* "*map entry*"
- map *mapname*, key *mountpoint*: bad

  The map entry is malformed, and the automounter cannot interpret it. Recheck the entry; perhaps there are characters in it that need escaping.
- *hostname*: exports: *rpc_err*

  There is an error when the automounter tries to get a share list from *hostname*.This indicates a server or network problem.
- host *hostname* not responding
- *hostname*:*filesystem* server not responding
- Mount of *hostname*:*filesystem* on *mountpoint*: *reason*

  You will see these error messages after the automounter attempts to mount from *hostname* but gets no response or fails. This may indicate a server or network problem.
- *mountpoint* - *pathname* from *hostname*: absolute symbolic link

  When mounting a resource, the automounter has detected that *mountpoint* is an absolute symbolic link (beginning with /). The content of the link is *pathname*. This may have undesired consequences on the client; for example, the content of the link may be */usr*.
- Cannot create socket for broadcast rpc: *rpc_err*
- Many_cast select problem: *rpc_err*
- Cannot send broadcast packet: *rpc_err*
- Cannot receive reply to many_cast: *rpc_err*

  All these error messages indicate problems attempting to "ping" servers for a replicated file system. This may indicate a network problem.
- trymany: servers not responding: *reason*

  No server in a replicated list is responding.This may indicate a network problem.

- Remount *hostname*:*filesystem* on *mountpoint*: server not responding

  An attempted remount after an unmount failed. This indicates a server problem.

- NFS server (*pidn@mountpoint*) not responding still trying

  An NFS request made to the automount daemon with PID *n* serving *mountpoint* has timed out. The automounter may be overloaded temporarily, or dead. Wait a few minutes. If you then still get this error, terminate all processes which access automatically mounted resources. Abort the automount process and start the automounter afresh, specifying a lower value for the "ping timeout" (option *-tp*). If the problem persists you will have to restart the system.

6.9.7 NFS statistics with the nfsstat command

nfsstat outputs statistics on NFS communications between the server and the client. Some of the information provided by this command can relate to errors. A full description of the command can be found in the "Networking Reference Manual".

nfsstat outputs statistics on:

- the number of RPCs that have been sent and received, and on any errors that have occurred. A distinction is drawn here between RPCs for the client and RPCs for the server.
- the number and type of NFS calls, and any errors that may have occurred. Here too, a distinction is drawn between statistics for the server and those for the client.

This information is available for RPCs of NFS Versions 2 and 3.

The command *nfsstat* fetches the required information from the system kernel and outputs it to standard output.

RPC statistics

Client:

calls is the number of RPC calls made

badcalls is the number of unsuccessful RPC calls

retrans is the number of RPC packets within a call which had to be sent more than once because no acknowledgment or an incorrect acknowledgment was received.
If the *retrans* option lists less than 5% of the *calls*, that is normal.

badxid is the number of acknowledgments for RPC packets which arrived after termination of the RPC call

timeout is the number of RPC packets sent within a call to which there was no response within a specified time (see *mount* command, *timeo* option)

wait is the number of calls where it was necessary to wait for internal resources

newcred is the number of RPCs for which authentication parameters were requested anew from the server

badverf is the number of RPCs for which the authentication parameters returned by the server were invalid (should always be 0)

timer is the number of RPCs for which the repeat timer was set to a value greater than the minimum value

toobig is not used; should always be 0

nomem is the number of RPCs for which internal memory problems occurred

cantsend is the number of RPCs the client could not send

bufulock is the number of RPCs which, when certain resources were released, led to other processes being invoked which were waiting for these resources

Server:

calls is the number of RPCs received

badcalls is the number of bad RPCs received

nullrecv is the number of attempted receives that were unsuccessful owing to internal memory problems

badlen is the number of RPCs received that were too short

xdrcall is the number of RPCs received for which acknowledgments could not be issued

drops is the number of packets that were unsuccessful due to the NFS server receive buffer being full

NFS statistics

calls is the number of NFS jobs which were sent

badcalls is the number of unsuccessful jobs

nclget is the number of times internal data structures were requested

nclsleep is the number of times it was necessary to wait for *nclget*

The parameters shown below affect internal RPC calls sent to a server. An absolute number and a percentage are specified in each case.

NFS V2 and NFS V3:

null no action

getattr request attributes for a file

setattr set attributes for a file

root is not supported in this version

lookup localize a file

readlink read from a symbolic link

read read from a file

wrcache is no longer supported by this version

write write in a file

create create a file

remove delete a file

rename rename file

link set simple link (hard link)

symlink set symbolic link

mkdir create directory

rmdir delete directory

readdir read in a directory

fsstat fetch file system statistics

NFS V3 only

access check file access rights

commit stabilize (commit) a write job

fsinfo fetch file system statistics

fsstat fetch dynamic information on a file system

mknod generate a device file

pathconf fetch information on a file's path configuration (maximum path length, maximum number of links, etc.)

readdirplus read a directory as per NFS V3 (extended)

6.10 Semantic differences in accessing local and NFS file systems

Many of the commands and library functions described in the relevant manuals support access to files in various file systems. In the case of these interface descriptions, it is implicitly understood that these file systems are local file systems. NFS permits the mounting of file systems from other machines at any location in a local file system. From the point of view of the application, accessing NFS file systems is the same as

accessing a local file system. Based on the NFS specification, there are however semantic differences depending on whether an operation is being executed on a local file system or an NFS file system.

The following description of NFS particularities should contribute to a better understanding of the semantic differences in accessing a file system.

6.10.1 Accessing device files and "named pipes"

An NFS client cannot access device files on the server. The file attributes, for example, the major and minor number can be queried by the server, however any attempt to open such a file normally fails. If the *devacc* option is specified during the mount process and device files with a suitable major and minor number exist on the client, the corresponding device files on the client are accessed.

Access to a "named pipe" in an NFS file system is always local, i.e. goes to a corresponding pipe on the client. No data is copied to the NFS server with a "named pipe". Nor is it possible to exchange data between several clients using a "named pipe".

See also: the *devacc* option of the *mount* command (the Section "The mount command").

6.10.2 Access to read-only file systems

Write accesses to NFS file systems that were specified as read-only by the server are rejected with the error EROFS, regardless of whether the file system was mounted by the client with write permission.

See also the *rw* and *ro* options of the *mount* command (the Section "The mount command") and the *ro* and *ro=* options of the *share* command (the Section "The share command").

6.10.3 Mapping user IDs (UIDs) on the server

Depending on the manner in which a server provides a file system for a client (see the Section "The share command"), access to a file system entry can be denied even though access would have been allowed by the client.

The reason is that the server either

1. mapped a UID that is unknown to it to the UID of the "anonymous" user.

   See also the *anon=uid* option of the *share* command, or
2. mapped the UID of the client system administrator (UID 0) to the UID of the anonymous user, if the client was not assigned explicit system administrator rights when the file system was released.

   See also the *anon=uid* and *root=host* options of the *share* command.

If access is refused by the server for one of the reasons described above, EACCES (1) or EPERM (2) is returned as an error code.

6.10.4 Group association of newly generated files

The *mount* command with the *grpid* option generates a file whose group ID corresponds to the effective group ID of the calling process.

See also the *mount(1M-nfs)* description in the "Networking Reference Manual".

6.10.5 Execution of set user ID programs

Sets s-bits are taken into consideration by default during an execute process. This can be prevented using the *nosuid* option.

See also the *suid* | *nosuid* options of the *mount* command (the Section "The mount command").

6.10.6 Synchronization of file accesses

The administration of locks to synchronize access to file subranges is not implemented by the NFS daemon but rather by the lock daemon and the status monitor daemon. The process ID returned by the *fcntl()* system call by means of F_GETLK is not the process ID of the application that locked the relevant data record in a file as is the case in local file systems, but rather an internal identifier within the lock administration function for distributed file systems.

"Mandatory locking" is not supported via NFS. An attempt to lock a file is rejected with the error EAGAIN if the access rights of the relevant file qualify its locking as "mandatory".

See also the F_GETLK parameter of the *fcntl(2)*

system call and, for information on "mandatory locks", the description of the *chmod(2)* system call in the

"Programmer's Reference Manual"

6.10.7 Modifying the file access rights after opening the file

Access to an opened file can be refused by the NFS server if the access rights have changed or the file was deleted. Since the NFS server cannot carry any information on files opened by clients (status-less server), all client accesses must be validated explicitly. This means that an operation on an opened file can be canceled with EACCES (no access rights) or ESTALE (invalid "filehandle").

6.10.8 Overwriting programs being executed

An NFS client can prevent a program it executed on an NFS server being overwritten by the server or by another client. The modification of an object file can result in the abnormal termination of a program being executed on a client.

6.10.9 Temporary files for backing up deleted files

If a file in an NFS file system is opened by a number of applications on an NFS client and the file is deleted by one of these applications, the file is renamed to a temporary file in the NFS file system. It is assigned the name *.nfsXXXX*, where *XXXX* stands for a four digit hexadecimal number that varies with each rename operation. The temporary file is deleted once the last application on the NFS client closes the file. This guarantees that the local file system remains transparent for the applications on the client at least. However a side effect is that unwanted files can remain in the server file system should the client machine crash.

6.10.10 Influences of various caches used by NFS clients

Some semantic differences in operations on local and NFS file systems are the result of the various caches implemented for the NFS clients. They arise mainly from a client not seeing the current data on the server for a short period of time.

6.10.10.1 Attribute cache

NFS clients save metadata from NFS files in so-called attribute caches.

The fundamental data saved is the:

- file type
- access rights
- user and group identifier
- file size
- access times

Even though the attribute cache is regularly updated with the current data, the client can work with old data for a short period of time. The consequences could be, for example, that access to a file is not permitted even though the access rights are set correctly on the server or that a client does not really add data to the end of the file because the file was already continued by the server or another client.

See also the *noac*, *acregmin=*, *acregmax=* options of the *mount(1M-nfs)* command in the "Networking Reference Manual".

6.10.10.2 Directory cache

The directory cache saves metadata from directories. Similar problems as arise for normal files with the attribute cache arise here.

Semantic differences to local file systems can arise if a client still sees file directory entries that have already been deleted on the server. A *create* operation on the client could then terminate with an error even though the file no longer even exists on the server.

See also the *noac*, *acdirmin=*, *acdirmax=* options of the *mount(1M-nfs)* command in the "Networking Reference Manual".

6.10.10.3 Data cache

In the same way as for local file systems, the user data of NFS files is also maintained in the main memory of the client machine. However while there is a guarantee for local files that all applications see the current data version, this is not the case for NFS files. The NFS client reads the data from the server only if the file was modified since the last access.

The length of time before the client once again reads in the current data contents can vary depending on the configuration of the attribute cache (see the Section "Attribute cache"). In the case of read or write accesses, the client can perform the entire access or a part thereof on the cache. This can, for example, lead to the data not being visible on the server immediately after a write access by the client.

Furthermore, accesses that exceed NFS block boundaries, i.e. 8 Kbytes for NFS V2, max. 16 Kbytes (actual) or 32 Kbytes (nominal) for NFS V3 under Reliant UNIX, do not take place atomically. Parallel accesses by different clients can thus mingle on the server and lead to unexpected results. Such accesses that would not cause problems on a local file system, for example, writing in append mode, should always be synchronized for NFS files through the use of locks.

6.10.11 System time synchronization

An application cannot rely on the access times in an NFS file system being correct. The access times generally relate to the system clock of the server and not to that of the client that last modified the file. A comparison of the access times of an NFS file with the system clock of the client or with the access time of a local file can return incorrect results since

- the clocks of the client and server may not be synchronized
- different time zones may have been set for the client and server
- an incorrect timestamp may be maintained in the attribute cache of the client

6.10.12 Differences in runtime behavior, signal processing

The runtime of applications that access NFS file systems increase depending on the options with which an NFS file system was mounted. The reasons for this are generally the overloading or failure of the NFS server

or problems in the network. Since the NFS requirements are executed synchronously by the client, the process waits until the retry and retransmission counters have expired.

See also the *retry*, *timeo*, *retrans*, and *soft* | *hard* options of the *mount* command (the Section "The mount command").

If an application is in the state just described, the relevant process can only be canceled with a signal if the NFS file system was mounted with the *soft* or *intr* option. Otherwise the signal is only issued if the NFS request was answered by the server.

See also the *soft* and *intr* options of the *mount* command.

7 The Network Information Service (NIS)

This chapter describes the Network Information Service (NIS). It deals with the tasks of the NIS and explains how the network administrator can use the NIS to simplify the job of administering large networks. Specifically, the following aspects are described:

- The NIS environment
- Setting up the NIS
- Administering NIS maps
- Adding new NIS servers
- Handling global users and user groups
- Setting up global mailing lists
- NIS-related commands
- Fixing NIS problems

There are several ways in which you, as the system administrator, can use the NIS service:

- The complete range of NIS functions are provided at the command level. You can use the options here when you want to manage information which deviates from the standard type, or if you wish to set up unusual client/server configurations.
- The SIfmllan menu system in *sysadm* provides you with menus for all the important network administration functions. Please refer to the corresponding sections for information about working with the SIfmllan menu system.
- If you have access to the graphical user interface of SINIX/windows, you can carry out some functions using desktop objects. The items in question are the "Host Manager" and "User Manager" desktop elements. You will find references in the relevant sections.

7.1 Introduction

The NIS is a distributed information service designed to meet the administrative needs of large, diverse, and changing groups of communication participants in TCP/IP networks. It is a mechanism for centralizing information which is required by a defined group of machines for communication in the network. The NIS enables the information to be administered centrally and made available to the entire group. The NIS provides a uniform, network-wide storage and access method that is both protocol- and media-independent.

By running the NIS service, the superuser can distribute administrative databases (maps) among a variety of machines and can update those databases from a central point, ensuring that the shared database remains consistent for all NIS clients in the network.

Programs, daemons and directories which are used in conjunction with the NIS start with the letters yp because NIS functions were previously referred to as Yellow Pages.

7.1.1 The NIS components

The NIS service consists of the following components:

Domains
: Administrative units for all machines which access a common database

Maps
: Databases for storing information

Daemons
: Processes for exchanging information

Utilities
: Programs giving access to information

7.1.2 How the NIS works

The NIS service is based on information contained in NIS maps. Maps are non-ASCII administrative databases, which usually derive from ASCII files found in the */etc/inet* directory, for example. Each NIS map has a name which the programs use to access it. On a network running the NIS, at least one NIS server per domain administers a set of NIS maps; the other machines can request information from these maps.

The service is provided by the *ypserv* daemon.

7.1.2.1 The NIS domains

An NIS domain groups together machines which obtain their information from the same NIS server and make use of a common set of maps. A machine is assigned to a domain if it is assigned a domain name. The maps for each domain are stored in special */var/yp/domainname* directories on NIS servers. For example, maps for machines that belong to the accounting domain will be located in the */var/yp/accounting* directory on the NIS server.

No restrictions are placed on whether a machine can belong to a given domain. Assignment to a domain is done at the local level of each individual machine by the superuser. This can be done in two different ways: by an entry in */etc/default/inet* (see the Section "Defining configuration values for starting the network") or by entering the following command:

domainname *name*

where *name* is the name of the domain to which you want the machine to belong.

7.1.2.2 NIS machine types

There are three types of NIS machine:

- Master server
- Slave server
- Client

NIS servers

An NIS server is a machine on which the NIS maps for a domain are stored and kept up to date. The information is made available to the other network participants.

There are two types of NIS server: NIS master servers and NIS slave servers. Machines which are intended for use as NIS master servers receive the original versions of the maps. This is where the maps are updated as necessary. If you only have one NIS server in your network, this is the NIS master server.

You can designate additional NIS servers on your network as slave servers. A slave server has a complete copy of the NIS master server's set of NIS maps. Whenever the NIS master server's maps are updated, it copies the updated maps to the NIS slave servers. The use of NIS slave servers ensures that the workload which results from responding to NIS requests is distributed equally between all the NIS servers. In addition, it guarantees that NIS requests can be dealt with even if the NIS master server should fail for a brief period.

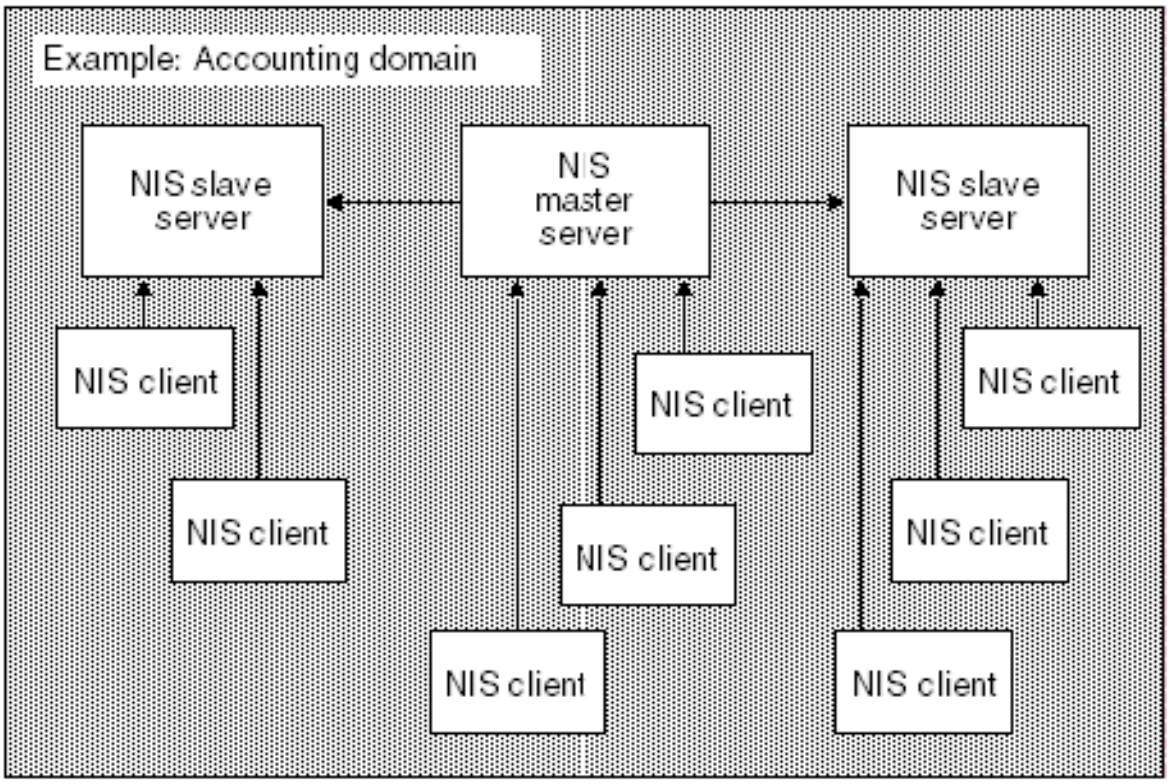


Figure 12: NIS servers and NIS clients

In Figure "NIS servers and NIS clients", you can see a typical NIS configuration. The NIS master server and two NIS slave servers respond to the requests of the NIS clients. Each NIS client is "bound" to one of the servers. Updates to the NIS maps are forwarded from the NIS master server to the NIS slave servers.

NIS clients

NIS clients run processes that request data from maps on the servers. Clients do not care which server is the NIS master server in a given domain, since all NIS servers have the same information. The distinction between NIS master and slave server only applies to where you make the updates.

Binding NIS servers and NIS clients

The NIS clients receive information from the NIS servers by means of a client process directing a request to the *ypserv* process on the server. This process involves the following steps:

1. A program running on the client which needs information from an NIS map queries the name of a server from the local *ypbind* daemon.
2. *ypbind* informs the client process which server it should use. The client then directs its request straight to

that server.

3. The *ypserv* daemon on the NIS server processes the request by looking in the corresponding map.
4. *ypserv* then sends the requested information back to the client.

This service attempts to balance the workload between the servers. As a result, the assignment of clients to servers can change if a server does not respond, for example as a result of it being overloaded. Consequently, a client may get its information from one server at one time and from another server at another time.

Use the *ypwhich* command to find out which NIS server is currently servicing a client:

$ ypwhich

7.1.2.3 NIS maps

NIS maps are databases which contain information about the network configuration in a *dbm* format. They are created from ASCII files using the *makedbm* command. These ASCII files are usually in the */etc/inet* directory. NIS maps contain information in the form of key and value pairs. In this system, the key is used for accessing the corresponding value by means of programs.

For example, the NIS map *hosts.byname* contains information about the machines known in the network. The machine name is the key; the values supplied are the corresponding Internet address and any other available information. The *hosts.byaddr* database contains the same information, although the Internet address is used as the key in this case.

The complete input format for the *makedbm* command is described under *ypfiles(4)* in the Networking Reference Manual. The entry can be made from a file or from standard input. After processing by *makedbm*, the information is located in the two non-ASCII files *mapname.dir* and *mapname.pag*, both of which are in the */var/yp/domainname* directory.

After installation, your system contains a series of standard maps which are required for network operation, such as a list of the machines known in the network or the networks which can be accessed. If necessary, you can also add your own maps (see Creating maps).

The following information is provided by the standard NIS maps:

- Which machines are known in the network
- Which networks can be accessed from the machines in the domain
- Which machine is the NIS master server and which NIS slave servers exist
- Which global users and user groups are in the domain
- Which global mailing lists have been set up
- Which services, protocols and RPC services are available
- Which masters and slaves are accepted for time synchronization.

The table below contains a list of the standard maps and the corresponding ASCII files. Some of the ASCII files are the Internet files which are used for configuring the network even when the NIS is not used (see Basic configuration). Some of these files already contain standard entries which you do not need to modify.

Information	ASCII file	NIS maps
Machines known in the network	*/etc/inet/hosts*	*hosts.byname* *hosts.byaddr*
Accessible networks	*/etc/inet/networks*	*networks.byname* *networks.byaddr*
NIS master servers and NIS slave servers	*/var/yp/ypservers* and */var/yp/binding/* \ *<domain>/ypservers*	*ypservers*
Global users	*/etc/yppasswd*	*passwd.byname*

		passwd.byuid
Global user groups	*/etc/ypgroup*	*group.byname* *group.bygid*
Global mailing lists	*/var/yp/mail.aliases*	*mail.aliases*
Network groups	*/var/yp/netgroup*	*netgroup.byhost* *netgroup.byuser*
Available services	*/etc/inet/services*	*services.byname* *services.byport*
Available protocols	*/etc/inet/protocols*	*protocols.byname* *protocols.bynumber*
RPC services	*/etc/rpc*	*rpc.byname* *rpc.bynumber*
Master and slaves for time synchronization	*/etc/inet/timed*	*timed*

7.2 Setting up the NIS

To set up the NIS, proceed as follows:

1. Set up the domains for your machines.
2. Set up the NIS master server.
3. Start the NIS services on the NIS master server.
4. Set up the NIS slave servers.
5. Start the NIS services on the NIS slave servers.
6. Initialize the clients.

These steps are described in more detail below.

7.2.1 Setting up a domain

Please note the following points before you start to set up a domain:

- Choose a name for the domain.

  The name for a domain can be up to 256 characters in length, but should not be longer than 14 characters if problems with older systems are to be avoided. The name for the directory on the servers in which the database is created (*/var/yp/domainname*) is generated from this name.
- Make a list of all the machines connected to the network which are to provide or use the NIS service within this domain.
- Decide which machines in this NIS domain are to be servers and which are to be clients.
- Decide which machine in this NIS domain is to be the NIS master server (you can always change this at a later date).
- Chose which machines in the network are to be NIS slave servers, if you want to have any.
- Finally, list all the machines which are to be NIS clients.

Once you have decided on a clear structure for your domain, you can assign the machines to the domain. Do this by logging on to all servers (NIS master and NIS slave servers) and clients in turn as the superuser.

Enter the following command:

domainname *name*

where *name* is the name of your domain.

This is only a temporary measure, however. You have to initialize the NIS using *ypinit* in order permanently to assign a machine to a domain. This causes the domain name to be entered in the */etc/default/inet* file.

7.2.2 Setting up the NIS clients and servers

ypinit is the central utility for setting up machines which use the NIS. *ypinit* makes it possible to initialize clients as well as NIS master and slave servers.

If any NIS services are currently active, *ypinit* will terminate them before carrying out the initialization, and all the necessary NIS services will be started after the initialization in any case. Normally, the NIS services are started automatically when the machine is booted in multiuser mode (run level 2), and terminated automatically when the system is closed down.

You can also start the NIS services explicitly by calling the start script as follows

/etc/init.d/yp start

and terminate them explicitly by calling

/etc/init.d/yp stop

7.2.2.1 NIS services (daemons)

The NIS includes a whole range of services, some of which run on both clients and servers. The NIS services are listed below.

ypserv
: This is the most important NIS service. It provides the clients with administrative information and the contents of NIS maps. This service only runs on NIS servers.

ypxfr_serv
: This service runs only on those clients which automatically receive the new data whenever a map is updated on the network master server (auto-update clients).

ypbind
: The assignment of clients to servers is not static. As a result, a client needs to know which server will handle a request. The process of determining which server is responsible is also referred to as "binding" the client to the server. *ypbind* is a service on the client which is used for this binding. NIS master servers and NIS slave servers normally also act as clients, so *ypbind* generally runs on clients and servers.

yppasswdd
: A service for password updates by global users (i.e. users who are entered in the *passwd* maps). This service only runs on the NIS master server.

yptransd
: A service for the selective distribution of global user data from a server to the clients. Consequently, the *yptransd* service runs on clients and is used by the servers by means of the *ryptrans* program.

ypupdated
: Allows NIS maps to be directly updated by clients. However, this is not normally wanted, and so Reliant UNIX 5.45 does not contain an application which uses this service.

7.2.2.2 Setting up the NIS master server

Here's how to set up an NIS master server:

1. Check that the initial files for the NIS map (see Section "NIS maps") are present and complete.
2. Use the command *domainname* to determine the name of the domain in which you want to set up the NIS master server.
3. Call *ypinit*.

ypinit -m

4. *ypinit* will ask whether you wish to activate a global user administration and which type you want to use. You can choose between the "old-style" user administration (as used in SINIX V5.43B and earlier releases) and a "new-style" administration (modeled on other NIS implementations).

   Old-style user administration gives you a choice between "limited access" and "full". Full details of global user administration are explained in the Section "Administering global users".

5. *ypinit* will then prompt you to enter a list of NIS servers. The name of the NIS master server will be included in the list automatically, you will only need to enter the slave servers.

   The list will be placed in the file */var/yp/ypservers*.

6. Before it creates the NIS maps, *ypinit* asks whether or not the creation process should be continued if errors are encountered (default setting: continue). If there are existing maps for the specified domains it will also ask whether these old maps may be deleted, and will abort unless you respond to this query with yes.

7. *ypinit* creates the NIS maps according to the instructions given in the file */var/yp/Makefile* (see Section "Administering the NIS maps").

8. If you want the NIS master server to inform not only the slave servers but also the auto-update clients whenever maps are updated, you should enter the names of the auto-update clients in the file */var/yp/auto_updaters*. When you have done this, recreate the map *auto_updaters* using

   # cd /var/yp
   # make auto_updaters

   The map *auto_updaters* will also have to be recreated whenever modifications are made to the file */var/yp/auto_updaters*.

    This requires the daemon *ypxfr_serv* to be running on the auto-update clients!

When the NIS master server has been successfully set up, the variable YP_MODE in */etc/default/inet* will have been set to master, the variable DOMAIN will have been set to the correct domain name, and the cron table will contain entries for *ypxfr* scripts.

7.2.2.3 Setting up the NIS slave servers

You cannot set up any slave servers until you have initialized the NIS master server as described above:

1. Use the command *domainname* to determine the name of the domain in which you want to set up the slave server.

2. Call *ypinit*.

   ypinit -s *master_server*

   where *master_server* is the name of the NIS master server

3. *ypinit* will ask whether you wish to activate a global user administration and which type you want to use. You can choose between the "old-style" user administration (as used in SINIX V5.43B and earlier releases) and a "new-style" administration (modeled on other NIS implementations).

   The old-style user administration gives you a choice between "limited access" and "full". Full details of global user administration are explained in the Section "Administering global users".

4. *ypinit* checks whether the name of the slave server is included in the list of permitted NIS servers on the NIS master server. If it is not, *ypinit* will abort with an error message.

5. Before transferring the NIS maps, *ypinit* asks whether or not the transfer should be continued if errors are encountered (default setting: continue). If there are existing maps for the specified domains it will also ask whether these old maps may be deleted, and will abort unless you respond to this query with yes.

6. *ypinit* copies the NIS maps from the NIS master server to the slave server.

When the slave server has been successfully set up, the variable YP_MODE in */etc/default/inet* will have been set to server, the variable DOMAIN will have been set to the correct domain name, and the cron table will

contain entries for *ypxfr* scripts.

7.2.2.4 Setting up the NIS clients

Here's how to set up an NIS client:

1. Use the command *domainname* to determine the name of the domain in which your machine is to operate as an NIS client.
2. Call *ypinit*:

   ypinit -c
3. *ypinit* will ask whether you wish to activate a global user administration and which type you want to use. You can choose between the "old-style" user administration (as used in SINIX V5.43B and earlier releases) and a "new-style" administration (oriented towards other NIS implementations).

   The old-style user administration gives you a choice between "limited access" and "full". Full details of global user administration are explained in the Section "Administering global users".
4. *ypinit* will ask whether you wish to set up the machine as an auto-update client. "Ordinary" clients get their data updated periodically from their server, but auto-update clients are supplied with the latest data directly from the NIS master server. This is achieved by placing the machine name of the auto-update client in the file */var/yp/auto_updaters* on the NIS master server (see the Section "Setting up the NIS master server").
5. *ypinit* will then interactively request a list of NIS servers. The server list is always entered in the file */var/yp/binding/domainname/ypservers*.
6. *ypinit* fetches the contents of the NIS maps from the server and stores them in local files. If the server list contains more than one name, then whichever machine in the list is the first to respond to a broadcast will become the server.

When the NIS client has been successfully set up, the variable YP_MODE in */etc/default/inet* will be set to client or client_auto, the variable DOMAIN will be set to the correct domain name, and the cron table will contain entries for *ypxfr* scripts.

Periodic updating of the NIS maps

Local processes of NIS clients in Reliant UNIX never work directly on NIS maps. Instead, they work on local files which are available in ASCII format under familiar names (e.g. */etc/services*) but which were created from the NIS maps. This offers the advantage that the processes do not "hang up" if the NIS servers fail. However, you must take care that the contents of the local files are sufficiently up to date. The updating process is carried out by the aforementioned crontab entries for *ypxfr_1hour*, *ypxfr_2day* and *ypxfr_1day*; this corresponds to hourly, twice daily or once daily intervals. For example, */etc/services* is updated once a day, whereas */etc/passwd* is updated hourly because it is more likely to have changed. *ypxfr_1hour*, *ypxfr_2day* and *ypxfr_1day* refer to a single script, */var/yp/ypxfr.all*, which selects the files to be updated on the basis of the program name. They are defined in advance in */var/yp/ypxfr.all* under the names *maps_1hour*, *maps_2day* and *maps_1day*. These definitions have to be extended to include new map names when new maps are added (see the section entitled Creating and updating new maps).

The */var/yp/map2system* script performs the actual conversion of the contents of the maps into local files. This script also has to be extended when new maps are added to include new instructions for obtaining the contents of the map (e.g. using *ypcat*) and copying them to the local file (see the section entitled Creating and updating new maps).

Auto-update clients

If a client was set up as an auto-update client when it was initialized (see *ypinit -c* in the Section "Setting up the NIS clients"), then it will receive the new data straightaway, whenever the files on the NIS master server are modified. This requires the daemon *ypxfr_serv* to be running on the client.

The name of the client must also have been entered in the file */var/yp/auto_updaters* on the NIS master server.

The NIS master server also administers the contents of the file */var/yp/auto_updaters* in the form of an NIS map which it distributes to the slave servers. This ensures that, in the event of a failure of the NIS master server, a slave server would be in a position to continue with the automatic client updates.

As well as the automatic updates initiated by the NIS master server, auto-update clients periodically update the data themselves, so the data will not be permanently out-of-date even if the automatic update should fail.

In a large domain, we do not recommend you to nominate many of the clients as auto-update clients, let alone all of them, because updating the data from the NIS master server is extremely time-consuming in such cases. The periodic updates of the ASCII files on the client will normally be perfectly adequate.

Local exclusion from update

Each individual client can exclude specific local ASCII files from being updated by the NIS. You do this by placing the names of those files which are to be excluded in the file */var/yp/suppress_rebuild* on the client.

This file contains one file name per line; lines of comment begin with the character "**#**".

Since the ASCII files on slave servers are updated in a similar way to those on clients (see the Section "Distributing the maps"), you can also use such an exclusion file here to suppress updating of individual ASCII files.

On the NIS master server, such an exclusion mechanism might at most be useful for the files */etc/passwd* and */etc/group* (see the Section "Administering global users"). For all the other files, the corresponding ASCII file on the NIS master server is in any case the starting-point from which the map is created.

Binding clients to a server

The binding process takes place before each NIS request. The sequence of events is as follows:

1. The client process sends a binding request to the local *ypbind* daemon.
2. *ypbind* checks if a server is already bound to the specified domain and contacts the server.
3. The existing bond can continue to be used if the server responds.

   A new bond is established if the server does not respond (or if this is the very first binding):

   - If the local */var/yp/binding/domainname/ypservers* file contains a + entry, *ypbind* sends out a broadcast across the network. The first server in the list following the + entry which responds is accepted and bound.
   - If the */var/yp/binding/domainname/ypservers* file contains an * entry, *ypbind* sends out a broadcast across the network. The first server which responds is accepted and bound.
   - If */var/yp/binding/domainname/ypservers* contains neither a + entry nor an * entry, the servers are contacted in the sequence given in the list. The first server which responds is bound.

The contents of the */var/yp/binding/domainname/ypservers* file are dependent on the value of the AUTO_BINDING environment variable. AUTO_BINDING is set in */etc/default/inet*. Its default setting is yes.

An NIS client for which AUTO_BINDING=yes is set adapts itself to changing definitions of NIS servers without requiring intervention by the superuser. The list of possible servers was initialized with *ypinit* and stored in the */var/yp/binding/domainname/ypservers* file. However, *ypservers* is an NIS map at the same time. AUTO_BINDING=yes means the client periodically updates its list of local servers using the contents of the map. In this process, a + entry is always added at the beginning of the list.

The NIS client's list of servers is not updated after setup when AUTO_BINDING=no. Local updates made by the system administrator are of course also possible. However, the list is not overwritten by the NIS server by means of periodic updates.

7.2.3 Working with SIfmllan

The *sysadm* user interface for system administrators provides the *SIfmllan* menu system for administering your network.

The functions of the SIfmllan menu system enable you to:

- Define and administer those networks which are to be accessible to the machines in the domain
- Administer the accessible machines in the network
- Administer the NIS slave servers in the domain
- Send updated NIS data to the NIS slave servers
- Create and administer global user groups (see Section "Entering global users using SIfmllan")
- Administer the global mailing list (see Section "Administering the global mailing list with SIfmllan")

Refer to Section "User interfaces for network administrators" for a description of how to call up the *sysadm* menu system.

7.2.3.1 Working on the NIS master server

Select *network_services* to display the menu for setting up and administering the network services. The functions for network administration are located in the *Network Services Management* menu.

```
Network Services Management
basic_networking    - Basic Networking Utilities Management
>lan               - LAN Administration (NIS)
mail-list          - Administration of Mailing List
name_to_address     - Machine and Service Address Management
remote_files        - Distributed File System Management
selection           - Network Selection Management
```

Only the menu items *lan* and *mail-list* are relevant to this section.

The menu items relevant to network administration have the following meaning:

lan For administering networks by means of the menu system. This is described in the next section.

mail-list
For administering mailing lists. This is described in Administering the global mailing list with SIfmllan.

If you use the *SIfmllan* menu system to administer the local domain, it takes over the job of updating the maps in *dbm* format and the ASCII files. In order to do this, it accesses the maps directly in *dbm* format. After the menu function has been terminated normally, the changes are also made in the ASCII files.

LAN administration

To administer networks, select *lan* from the *Network Services Management* menu. This calls up the *LAN Administration on Master* menu.

The following preconditions must, however, be fulfilled:

- The Internet services must be available (in other words, the *inetd* daemon process must have been started).
- The NIS services must be active (in other words, the *ypbind*, *ypserv* and *ypupdated* daemon processes must have been started).
- The machine must be set up as the NIS master server of a domain.

If these preconditions are not fulfilled, the *LAN Administration on Client* menu is displayed on the screen, indicating that your machine has been defined as an NIS slave server or client (see the section entitled Working on an NIS slave server or client machine").

```
LAN Administration on Master
>status      - Display Status Information of this Domain
connect      - Connect this Computer to LAN
```

disconnect - Disconnect this Computer from LAN
stop - Stop Administration through NIS services
administer - Administer Network Configuration
update - Send Updated Data to Slaves
slave - Make this Computer a Slave

status Select this menu item to view an output form containing information on the status of the local machine

connect
Select this menu item to obtain an input/output form that enables you to connect your machine to a domain. Do not execute this function from a remote workstation.

disconnect
Select this menu item to disconnect your machine from the current domain.

stop Select this menu item to interrupt all the network-related functions of your system.

administer
Select this menu item to view and modify the network configuration data.

update
Select this menu item to send modified network configuration data to the NIS slave servers

slave Select this menu item to reconfigure the NIS master server as a slave. You cannot implement this change until you have reconfigured a system previously designated as an NIS slave server as the NIS master server.

Connecting a machine to the LAN

You may have connected your machine to the LAN when you installed the operating system. In certain cases, however, you will have to connect your machine to the network yourself. This is the case:

- If names and addresses change in your network
- If you can only integrate your machine into a network subsequently
- If your machine is the NIS master server

The menu item *connect* on the *LAN Administration on Master* menu enables you to connect your machine to the network. The following input/output form is displayed:

Connect the Local Computer in the LAN
Name of Local Computer: neptune
Name of NIS Domain:
Do you want the global user file activated?
Do you want time synchronization enabled?

Name of Local Computer
This is an output field containing the name of the local machine, as entered in */etc/inet/hosts*. This name cannot be modified.

Name of NIS Domain
Enter the name of the domain in this field. Press CHOICES to obtain a single-choice list of the names of defined domains. The default is the name of the current domain. The only valid entry is a character string consisting of letters or digits. If you enter a new domain name, your machine becomes the NIS master server of this new domain.

Do you want the global user file activated?
Determines whether you want the global user file copied to the NIS master server and the entries written to the local user file. The possible entries are no, yes, auto and new (see the Section "Administering global users").

Do you want time synchronization enabled?
Determines whether you want the system time automatically synchronized. Valid entries are yes and no.

If no NIS master server is found when this machine is being connected to the network, you are asked if you wish to make your machine the NIS master server. You receive further instructions if you confirm that you

wish to do this.

If problems occur when connecting the machine, read the section entitled Error analysis and diagnostics.

Disconnecting a machine from the LAN

The menu item *disconnect* on the *LAN Administration on Master* menu enables you to disconnect your machine from the LAN. Disconnection means that:

- The NIS services are stopped.
- The slaves can no longer receive "updates" of the NIS data, because the NIS master server is no longer accessible via the network.
- No further changes can be made to the NIS data.
- The Internet services are stopped.
- Even after a reboot, the machine remains disconnected from the network.

After you have disconnected your machine from the LAN, you can no longer select *disconnect*, *administer*, *update* or *slave* from the *LAN Administration on Master* menu. You have to reconnect the machine to the LAN first.

The menu items originally displayed, namely *administer* and *slave*, are not shown any more because this is no longer an NIS master server.

Terminating administration by NIS services

The *stop* menu item on the *LAN Administration on Master* menu enables you to terminate administration of your machine by the NIS services. Unlike *disconnect*, this command does not stop the Internet services provided by the *inetd* daemon.

Administering the network configuration

Select *administer* from the *LAN Administration on Master* menu. The following menu is displayed:

```
Administration of Network Configuration (Master)
>networks     - Administer Network Addresses
hosts         - Administer Computers defined in the Network
ypser         - Administer Slaves defined in the Network
```

The NIS slave servers receive new administration data immediately, whereas client machines are only supplied with the new data at a later stage.

networks

The *networks* menu item enables you to administer the *networks* map.

hosts You can administer the machines that belong to the current domain and the accessible machines in remote networks on the NIS master server.

Before you add a machine to the list of accessible machines, you should make sure that:

- The network address of the network to which the machine in question belongs is entered in the list of defined networks.
- The machine number is unique in the network.
- The machine name is unique in the network.
- The alias or aliases are unique in the network.

The menu system incorporates all the functions you need to administer the machines in the network.

You can access these functions by selecting *hosts* from the *Administration of Network Configuration (Master)* menu. These functions act directly on the *hosts* NIS map.

ypserv

The *ypserv* menu item in the *Administration of Network Configuration (Master)* menu enables you to administer the NIS file that contains the list of NIS slave servers defined in the network.

Sending NIS data to the slaves

As a rule, the user interface automatically sends the updated NIS data after every modification. However, you can use the *update* menu item in the *LAN Administration on Master* menu to send the updated NIS data to the NIS slave servers if an NIS slave server was inaccessible, for example.

Reconfiguring the NIS master server as an NIS slave server

Before you can reconfigure the current NIS master server as an NIS slave server, you must set up a new NIS master server (see the Section "Working on an NIS slave server or client machine").

After defining the new NIS master server, you can reconfigure the original master as an NIS slave server.

Select the *slave* menu item from the *LAN Administration on Master* menu in order to do this.

7.2.3.2 Working on an NIS slave server or client machine

On an NIS slave server or client machine, you can:

- Connect your own machine to and disconnect it from the network
- Diagnose and rectify errors
- Modify the NIS mode of the machines
- Send NIS data from the NIS master server to the NIS slave servers
- Global users are created on the NIS master server; the global user must be active on the local system (see Section "Administering global users").

It is recommended that you perform these administrative tasks using the *SIfmllan* menu system.

Select *network_services* to view the menu that enables you to create and administer the network services.

```
Network Services Management
basic_networking    - Basic Networking Utilities Management
>lan                - Communication Management Program
mail-list           - Administration of Mailing List
name_to_address     - Machine and Service Address Management
remote_files        - Distributed File System Management
selection           - Network Selection Management
```

Only the *lan* menu item is relevant to the network administration of an NIS slave server or a client machine. The other items on the menu are described in the "System Administrator's Guide".

The *lan* menu item has the following meaning:

lan For administering an NIS slave server or client machine in the LAN

The following menu is displayed when you select *lan* and press ENTER:

```
LAN Administration on Client
>status      - Display Status Information of this Domain
connect     - Connect this Computer in the LAN and a NIS Domain
disconnect  - Connect this Computer from LAN and NIS Domain
stop        - Stop Administration through NIS services
list        - Display Network Configuration
update      - Get Most Recent Data from Master
assign      - Assign Servers of the LAN
```

status Select this item to view an output form containing information on the status of the local machine. The list may contain an "undefined" value if a particular variable has yet to be defined.

The *status* menu item on the NIS slave server/client machine has the same function as on the NIS master server.

connect

Select this item to obtain an input/output form that enables you to connect your machine to a domain.

The *connect* menu item on the NIS slave server/client machine has the same function as on the NIS master server (see the Section "Working on the NIS master server").

disconnect

Select this item to disconnect your machine from the current domain. Do not execute this function from a remote workstation.

The *disconnect* menu item on the NIS slave server/client machine has the same function as on the NIS master server.

stop

Select this item to interrupt all the network-related functions of your system. The *stop* menu item on the NIS slave server/client machine has the same function as on the NIS master server.

list Select this item to display the current network configuration.

update

Select this item to fetch the current network configuration from the NIS master server.

assign Select this item to reconfigure an NIS slave server as the NIS master server or a client machine as an NIS slave server.

Some of the items in this menu remain inactive until the machine has been connected to the network. In other words, you must perform a *connect* before you can use them.

Displaying the network configuration

If you want to display the network configuration data, you must select *list* from the *LAN Administration (NIS) on Client* menu. The following menu is displayed:

Display Network configuration (Client)
>networks - Display Network Addresses
hosts - Display Computers defined in the Network
slaves - Display Masters and Slaves

networks

Select this item to display a list of accessible networks

hosts Select this item to display a list of the machines defined in the network

slaves Select this item to display a list of the NIS master servers and NIS slave servers of the domain

Fetching network configuration data from the master

You are responsible for ensuring that the most recent version of the NIS data is available at all times on the NIS slave server. You must also ensure that the most recent versions of the files for global users and global user groups are transferred to the client machines.

In the SIfmllan menu system, the most recent NIS data is automatically copied to your machine after every modification. In addition, every time a system boots, the NIS data is copied from the NIS master server to your machine.

You will have to fetch the network configuration data from the NIS master server yourself if:

- You discover that a more recent version of certain NIS data exists on the NIS master server, and you prefer not to wait for the update time
- The NIS slave server, NIS master server or the network did not permit an update at a previous point in time

Changes in the NIS data for global users and user groups are not automatically forwarded to all other machines. This information is fetched from the NIS master server by all the other machines in the domain only at system start and as defined by the entry in the *crontab* file.

Select the *update* menu item from the *LAN Administration (NIS) on Client* menu if you want to update your NIS data.

Reconfiguring an NIS slave server or client machine

Select the *assign* menu item from the *LAN Administration (NIS) on Client* menu if you want to reconfigure an NIS slave server as the NIS master server or as a client machine. You can also use this function to reconfigure a client machine as an NIS slave server.

Assign Servers of the LAN (Client)

>slave - Make this Computer a Slave
master - Make this Computer the Master
remove - Delete this Computer as a Slave

slave This item enables you to reconfigure a client machine as an NIS slave server. You can do so only if the client machine has already been entered on the NIS master server as an NIS slave server (see the section entitled Section "Administering the network configuration").

An output form with a message is displayed if you select *slave*. The client machine is reconfigured as an NIS slave server if you quit the output form by pressing CONT.

master

This item enables you to configure an NIS slave server as the NIS master server by selecting *master* in the *Assign Servers of the LAN (Client)* menu. The current NIS master server must be active.

Proceed as follows after you have successfully reconfigured your NIS slave server:

- Log onto each NIS slave server and use the *ypinit-s* [*NIS_master*] command to inform the NIS slave server of the new NIS master server's identity
- As soon as possible, reconfigure the former NIS master server as an NIS slave server. This avoids inconsistencies in the NIS data or the loss of NIS data

remove

Select the *remove* menu item from the *Assign Servers of the LAN (Slave)* menu if you want to reconfigure an NIS slave server as a client machine. In the dialog box that appears, you can specify whether or not the NIS data is to be deleted.

If you enter yes and confirm your entry by pressing SAVE, your machine will be reconfigured as a client machine. You must then remove your machine from the list of NIS slave servers on the NIS master server, as otherwise your modification is not valid.

The machine is also configured as a client if you enter no. Nonetheless, the NIS data should be deleted because:

- It takes up memory unnecessarily.
- It is probably out-of-date if the machine is connected in the same domain again as a server.
- It is distributed unintentionally if the machine is connected to a different domain as a server.

7.3 Administering the NIS maps

This section describes how the maps of an existing domain are administered. It deals with the following topics:

- How to create maps
- How to update the contents of a map
- How to create a new map on the basis of a new source file
- How maps are distributed to NIS slave servers

7.3.1 Creating maps

NIS maps are created on the NIS master server. This is performed automatically by *ypinit -m* when the NIS master server is being set up. In order to create NIS maps, the information initially has to be present in the ASCII source files. The ASCII files are then converted to dbm format because the *ypserv* network service only works with this format. *ypinit -m* uses the *make* command together with the makefile specified in the */var/yp* directory for this purpose.

By way of example, the actions of the makefile are described here for the */etc/inet/hosts* file:

```
NOPUSH = ""
YPDBDIR=/var/yp
YPDIR=/usr/sbin
YPPUSH=$(YPDIR)/yppush
STDHOSTS=$(YPDBDIR)/stdhosts
```

```
HOSTS=/etc/inet/hosts
DOM= 'domainname'
ALIAS=/var/yp/ypalias
MAKEDBM=/usr/sbin/makedbm
hosts: hosts.push
hosts.push: hosts.time
-@if [ ! "$(NOPUSH)" -a -f $(HOSTS) ]; then \
$(YPPUSH) hosts.byname; \
$(YPPUSH) hosts.byaddr; \
/var/yp/map2system hosts;
touch $@; echo "pushed $(@:.push=)"; \
fi
hosts.time: $(HOSTS)
-@if [ -f $(HOSTS) ]; then \
domain=$(DOM); \
$(STDHOSTS) $(HOSTS) \
| awk ´{for (i = 2; i <= NF; i++) { \
if ($$i ~ /^#/) \
break; \
print $$i, $$0 \
} \
}´ \
| $(MAKEDBM) - $(YPDBDIR)/'$(ALIAS) -d $$domain'/ \
'$(ALIAS) hosts.byname'; \
$(STDHOSTS) $(HOSTS) \
| awk ´BEGIN { OFS="\t" } \
{ print $$1, $$0 }´ \
| $(MAKEDBM) - $(YPDBDIR)/'$(ALIAS) -d $$domain'/ \
'$(ALIAS) hosts.byaddr'; \
touch $@; echo "updated $(@:.time=)"; \
else \
echo "source of map $(@:.time=) not found; map not done";\
fi
```

The *hosts.time* target is created initially. The */etc/inet/hosts* file is converted to a standardized format (e.g. without comments and leading blanks) by the *stdhosts* utility within the NIS. The result is filtered by *awk* and supplied to *makedbm*. The effect of the *awk* filter is to call *makedbm* with the correct key fields. In this way, the *hosts.byname* and *hosts.byaddr* NIS maps are created in the form of the *mapname.dir* and *mapname.pag* files in the */var/yp/domainname* directory. The moment when the map is created is recorded as the modification stamp of the (empty) */var/yp/hosts.time* file. The maps *hosts.byname* and *hosts.byaddr,* which have just been created, are distributed to the slave servers and auto-update clients via the target *hosts.push* using the command *yppush*.

7.3.2 Updating the maps

You have to change the source file and then create the map again in order to update a map.

If, for example, a new machine is added to your network and you wish to use the NIS to inform all the machines in the network about the new machine, enter the name and Internet address of the machine in the */etc/inet/hosts* file on the NIS master server.

Then change into the */var/yp* directory and call *make*:

```
# cd /var/yp
# make
```

make ensures that all maps are brought up to date. In addition, they are distributed to the NIS slave servers.

7.3.3 Creating and updating new maps

As standard, the maps listed in the Section "NIS maps" are created and distributed by *ypinit -m*. If you also want other information to be accessible via the NIS, you have to create new maps for this. In order to do this, you can add to the */var/yp/makefile* file on the NIS master server or explicitly call the necessary commands there.

7.3.3.1 Adding to /var/yp/makefile

We will use an example to explain the expansion of */var/yp/makefile*. We will assume you wish to use the NIS to make a direct map globally available to the automounter. The map is an ASCII file called */etc/auto.direct*. (You will find a description of the automounter in the section entitled Using the automounter.)

1. Define an NIS alias for the new map. Do this by adding the following line to the */var/yp/aliases* file on the NIS master server:

   auto.direct ad

   This assigns the alias name ad to the new map. This can be used by NIS clients when accessing the map.

2. Define the new map as the make target in */var/yp/Makefile*.

   Do this by adding the map name auto.direct to the line starting with *rest:*

   ```
   rest: publickey passwd group hosts networks netgroup \
   protocols services rpc mail auto.direct
   ```

3. Enter the dependencies to the corresponding "time stamp" files in the makefile:

   ```
   auto.direct:            auto.direct.push
   auto.direct.push:       auto.direct.time
   □□□if [ ! "$(NOPUSH)" -a -f /etc/auto.direct ]; then \
   □□□□□□$(YPPUSH) auto.direct; \
   □□□□□□touch $@; echo "pushed $(@:.push=)"; \
   □□□fi
   ```

4. Enter the make rules for the auto.direct.time target in the makefile. These contain the actual creation of the map by *makedbm*. The key field for the new map is the first field (mount point), so the /etc/auto.direct file does not need to be changed for *makedbm*. However, the comment lines have to be deleted using a *sed* command.

   ```
   auto.direct.time: /etc/auto.direct
   □□□-if [ -f /etc/auto.direct ]; then \
   sed-e "/^#/d" /etc/auto.direct | \
   ```

```
$(MAKEDBM) - $(YPDBDIR)/$(DOM)/'$(ALIAS) \
auto.direct'; \
touch $@; echo "updated $(@:.time=)"; \
else \
echo "source of map $(@:.time=) not found; \
map not done"; \
fi
```

5. Create the new map by calling make auto.direct in the */var/yp* directory. The map is also distributed to the NIS slave servers using the *auto.direct.push* target.
6. Use ypcat - k auto.direct to check whether the map has been successfully created.

7.3.3.2 Creating a map explicitly

You can, of course, also create a map explicitly on the NIS master server by calling *makedbm*:

```
# cd /var/yp/domainnname
# /usr/sbin/makedbm /etc/auto.direct auto.direct
```

However, you should note that an explicitly created map is not automatically distributed to the NIS slave servers. You can use the *-u* option of *makedbm* in order to view the contents of a map as an ASCII file:

```
# /usr/sbin/makedbm -u auto.direct
```

This shows the contents of the map in standard output.

7.3.4 Distributing the maps

The periodic updating of the local ASCII files by the NIS clients has already been described in the Section "Periodic updating of the NIS maps". However, the NIS slave servers also have to harmonize their information with the NIS master server's information. In this case, the NIS slave servers have to receive the complete set of NIS maps. This can take place as follows:

- On the initiative of the NIS master server (*yppush* command)
- Through the NIS slave server itself (*ypxfr* command)

Both alternatives are used. When a map on the NIS master server is created afresh - for example, following modification of the original file - *yppush* then copies the map to all the slave servers listed in the file */var/yp/ypservers* on the NIS master server. For the auto-update clients, *yppush* transfers only the ASCII information to the client, not the actual map. For standard maps this will happen automatically by means of the actions specified in */var/yp/Makefile*.

In addition, each NIS slave server updates its maps periodically using the *ypxfr* entries in the cron table, based on the same criteria that a client uses for updating its local files. This is necessary because, for example, an NIS slave server which was temporarily unavailable may have "missed" the copy operation performed by the NIS master server.

Changing the frequency of updates

Which local files on NIS clients and which NIS maps on NIS slave servers are updated at what frequency is set in *ypinit* and *ypxfr.all*. The defaults can be changed as follows, for example:

- By regrouping the map lists in */var/yp/ypxfr.all*

  Example

  The *hosts* (=*hosts.byaddr*) map is to be updated hourly rather than twice daily. Remove the "hosts" entry from the *maps_2day* list in *var/yp/ypxfr.all* and add it to the *maps_1hour* list.

- By adding new map lists and *cron* table entries

  Example

You wish to update the *hosts* (=*hosts.byname*) map every quarter of an hour, for example. Remove the "hosts" entry from the *maps_2day* list in */var/yp/ypxfr.all* and define a new map list:

maps_4hour="hosts"

Add the new name to the *case* switch which evaluates the program name:

```
*4hour*)
maps='echo "$maps" | egrep "$maps_4hour"'
;;
```

Set up a reference to *ypxfr.all* with the new program name:

```
# ln /var/yp/ypxfr.all /var/yp/ypxfr_4hour
```

Set up a *cron* table entry for the new program name:

```
# crontab -e
```

Add the line:

```
0,15,30,45 * * * * /var/yp/ypxfr_4hour
```

7.4 Adding an NIS server

Even when the NIS machines have already been set up and are working, it may be necessary to configure an additional NIS server which is not yet contained in the NIS master server's list (*/var/yp/ypservers*). Because *ypservers* is an NIS map, proceed as described in the section entitled Updating the maps in order to inform all the machines in the network about the new server.

1. Log into the NIS master server as the superuser.
2. Edit the */var/yp/ypservers* file. Add the name of the new server to this file.
3. Create and distribute the new *ypservers* map.

   ```
   # cd /var/yp
   # make ypservers
   ```
4. Set up the slave as described in the Section "Setting up the NIS slave servers".

7.5 Administering global users

NIS permits you to set up global user IDs and global user groups. A global user ID is set up only on the NIS master server and applies within all the NIS domains that are administered from there. The procedure for setting them up is described in the Section "Setting up global users and groups on the NIS master server".

As an alternative to the previously used procedure for global user administration (until SINIX V5.43B) you can now make use of a newer method which is more closely oriented towards other implementations of NIS clients.

You select the desired method in the *ypinit* dialog. If you select the new procedure (which is the default), the environment variable GLOBALPW in */etc/default/inet* will be given the value new.

The new procedure is described in the Section "The new procedure for administering global users", and the previous method in the Section "Old-style administration of global users".

System commands and library functions in Reliant Unix do not normally address the NIS server, even when they need to access information which is held in the NIS maps. The local files are therefore periodically updated from the NIS maps, as described in the Section "Setting up the NIS clients".

Such an update consists of transferring the NIS map to the corresponding local file. However, the user administration usually permits not only local users and groups to access an NIS client but also "global" ones, that is, those listed in the NIS maps. In this case, updating the local files */etc/passwd*, */etc/shadow* and */etc/group* no longer consists merely of copying the contents of the maps to the corresponding files, the local files are constructed by merging the local and global entries, whereby this procedure is governed by selection criteria which specify the extent to which global entries are to be entered in the local files.

It is the specification of these selection criteria which constitutes the principal difference between the old

method and the new one:

- for the new method, the criteria are defined in the files */etc/passwd.local* and */etc/group.local*
- for the old procedure they are defined by the file */var/yp/pwpattern*.

The new procedure also enables you to specify password aging information for global user entries (see the Section "Password aging information for global users").

The transfer of a global user ID to an NIS client can be initiated either by the client or by the NIS master server.

You can decide separately for each client whether and by what means it should be able to access global user IDs. You do this by setting the environment variable GLOBALPW.

7.5.1 The environment variable GLOBALPW

The environment variable GLOBALPW is defined in the file /etc/default/inet. If it is set to no, the NIS client will have no access to global user IDs or user groups. The system administrator will have to enter all the possible users and groups in the appropriate files on this client: */etc/passwd*, */etc/shadow* and */etc/group*.

If GLOBALPW is set to new, yes or auto, global user IDs and groups in the local files will be taken over onto the NIS client. If GLOBALPW=new, this will be carried out using the new procedure which is described in the Section "The new procedure for administering global users".

If, when you called *ypinit* you selected the old procedure and the "limited access" variant then GLOBALPW will be set to yes. The variant "full" has the effect of setting GLOBALPW to auto. Both variants are explained in more detail under the description of the old method of administration in the Section "Old-style administration of global users".

7.5.2 The environment variable YPMIXER_CHECKDIR

The entry for each global user specifies a home directory. If such a global user entry is to be taken over by the NIS client and the home directory is not present on the client, then the value of YPMIXER_CHECKDIR governs how the client will determine the home directory for a global user.

The variable YPMIXER_CHECKDIR is defined in the file */etc/default/inet* and may take the values nocheck, create and default. Omitted values will be interpreted as though default had been set. Invalid values can also lead to other interpretations.

nocheck
: the name of the home directory is taken over from the global entry in the local */etc/passwd* without any checking.

create
: if the specified home directory is not present on the local machine it will automatically be created and the contents of */etc/skel* will be copied into it. The name of the directory will be copied to the local */etc/passwd*.

default
: if the specified home directory is not present, the directory */tmp* will be entered in the local */etc/passwd* for this global user.

7.5.3 The new procedure for administering global users

If the variable GLOBALPW in */etc/default/inet* is set to new this means that the new procedure was selected for administering global users. This procedure makes the definition of global users more flexible and is more closely modeled on other NIS implementations. It also provides facilities for modifying the passwords of global users.

In the new procedure, the question of which entries for global users and groups and which fields from these entries should be taken over onto the client is determined by consulting the files */etc/passwd.local* and */etc/group.local*.

The file /etc/passwd.local

Each client can use the file */etc/passwd.local* to specify which global user entries and which fields from these

entries should be taken over into the local */etc/passwd*. The new procedure offers the following features that are not provided by the old method, which made this selection using the file */var/yp/pwpattern*:

- the client can exclude global users (specify "all except")
- netgroups can be specified instead of users
- defaults can be set up on the client for specific fields of global user entries

The command *newpasswd* is used to merge global user entries into the local file */etc/passwd*, with reference to */etc/passwd.local*.

The file */etc/passwd.local* is laid out in the same way as */etc/passwd*, except that each entry is preceded by a "+" or "-".

A "+" signifies that the user should be taken over into the local */etc/passwd* and */etc/shadow*, provided there is an entry for that user in the map of the NIS master server. A "-" in front of the user name signifies that this user is not to be taken over. If the "+" is followed only by a ":" then all the global users will be taken over apart from those that have been excluded by previously encountered "-" entries. Subsequent "-" entries will then be disregarded.

The fields of the file */etc/passwd.local* which are separated by ":" correspond to the fields of */etc/passwd*. All those fields of */etc/passwd.local* that are not empty will be taken over into the local */etc/passwd*, the remaining fields will be taken from the global entry.

Exceptions: the fields "User number" and "Group number" are always fetched from the global entry.

These values are always first checked to see if there is a user of the same name or with the same user number on the local machine. If there is, the global user will not be taken over onto the local machine.

The names of all the global users that have been taken over are registered in the local file */var/yp/yppwusers*. This file must be present at all times, even if it is empty.

Example:

```
+thomas:
+fred::::::/sbin/ksh
-anna:
+::::teleservice
```

The global user fred will be taken over, but */sbin/ksh* will be specified as login shell in his */etc/passwd* entry.

The user with ID anna will on no account be taken over as a global user. All other global users will be taken over into the local */etc/passwd*, but their entries will include *teleservice* in the comment field.

If a name beginning with '@' is specified in */etc/passwd.local* this will be interpreted as the name of a netgroup. Elements of netgroups are tuples of the form

(*Machine name*, *User name*, *Domain name*)

Netgroup entries are evaluated only for *User name*, the entries *Machine name* and *Domain name* are irrelevant.

The netgroups themselves are global objects and are defined on the NIS master server. The corresponding ASCII file on the NIS master server is */var/yp/netgroup* (corresponds to the *netgroup* NIS map). For each netgroup the entry consists of the group name and a list of its members, for example

reliant (,ruth,) (,judith,) (,dave,)

The netgroup reliant consists of the IDs ruth, judith and dave.

A number of netgroups can be combined to a further netgroup:

Example:

```
unixgroup reliant driver networking
```

The netgroup unixgroup consists of the members of the netgroups reliant, driver and networking.

A netgroup entry in the file */etc/passwd.local* might have the following structure, for example:

```
+@unixgroup:
```

This has the effect that all the IDs from the netgroup unixgroup will be entered in the local */etc/passwd* on the client, provided they are recorded in the map of the NIS master server.

The file /etc/group.local

Global user groups are listed in the file */etc/ypgroup* on the NIS master server, in the same way as for the old procedure. NIS clients take over the contents of the map *group.byname* created from */etc/ypgroup* into their local */etc/group*.

According to the old procedure, all the entries were taken over from the server. The new procedure (GLOBALPW=new) permits the client to specify more precisely, in the file */etc/group.local*, which of the global groups it wishes to take over from the NIS server.

As for global users, the global groups are first checked to see whether there is already a group on the client machine with the same name or the same group number. If this proves to be the case, the global group will not be taken over onto the client.

In the same way as for */etc/passwd.local*, the entries in */etc/group.local* can also:

- mark global groups to be taken over: the entry begins with

  +groupname:
- exclude global groups from being taken over: the entry begins with

  -groupname:
- locally redefine the attributes of global groups that are taken over (e.g. group members).

Example:

```
+unix_team:::ellen,frank,peter
+:
```

The global group unix_team will be entered in */etc/group* with the global group number, but with members ellen,frank,peter. All other global groups will be taken over into */etc/group* unchanged.

The names of all the global groups that have been taken over are registered in the local file */var/yp/ypgrps*. This file must be present at all times, even if it is empty.

7.5.4 Old-style administration of global users

7.5.4.1 The /var/yp/pwpattern file

The entries in the */var/yp/pwpattern* file control which fields in the local */etc/passwd* and */etc/shadow* files are to be updated, i.e. which fields in the local files should be overwritten with the corresponding fields of the global users.

The entries in */var/yp/pwpattern* are structured as follows:

Name:*Password*:*UID*:*GID*:*Comment*:*Homedirectory*:*Shell*

Name can be the name of a user or *. The latter controls updating for all users who are not explicitly listed with *Name*. Entries subsequent to the * are not significant.

Password is YES if the password in */etc/shadow* is required to be updated. Otherwise, it is NO.

The UID and GID fields cannot be changed.

Comment, *Homedirectory* and *Shell* are YES if the corresponding fields in */etc/passwd* are to be updated. Otherwise, they are NO.

If a global user on the client is entered for the first time rather than updated, default values are selected for those fields which according to */var/yp/pwpattern*, are not to be updated (i.e. those containing NO). The default values are:

Password:

YP□

YP is also entered as the password if the *pwpattern* field is YES and the global password is blank.

Comment:

NIS

Homedirectory:

/tmp

Shell:

/sbin/sh□

/sbin/sh is also entered as the shell if the *pwpattern* field is YES and the specified shell does not exist or cannot be executed.

The entry for the global user is not accepted at all if the *pwpattern* fields *Password*, *Comment*, *Homedirectory* and *Shell* for that user are all set to NO. If the user already exists in the local files, he or she is removed.

Example

```
fred:YES:UID:GID:YES:NO:YES
anna:NO:UID:GID:YES:YES:YES
*:YES:UID:GID:YES:YES:YES
```

The user fred has a global password, i.e. he can log into the client and the server using the same password. However, his home directory on the client is different from the global entry.

The user anna has different passwords on the client and on the server. Her home directory and shell are identical on both machines.

All other global user entries are accepted in full in the local files */etc/passwd* and */etc/shadow*.

7.5.4.2 Updating the local passwd, shadow and group files

In order for a global user to be accepted in a local file on the client for the first time, there must be no other local entry with the same name or the same UID already on the client. Global users can be recognized as

such on the client because of the *YP* password or the *NIS* string in the comment field. If neither of these criteria apply, the user is purely local.This means global users on the client could be changed into local users by changing the password or the comment field.

However, there is more to changing the password than just using *passwd* (see the Section "Changing passwords for global users").

A similar procedure applies to user groups. A global group entry will be taken over only if there is not already a local group entry with the same name or the same GID. Global group entries are identified in the local file */etc/group* by means of the password field *YP*. Entries will be taken over into the list of users belonging to the group only if there is also a corresponding entry in */etc/passwd* for them on the client.

Updating of local files by the NIS client

Normally, the client updates the local files itself. The client performs the update every hour by means of the *ypxfr_1hour* entry in *crontab*. The value of GLOBALPW and the contents of */var/yp/pwpattern* specify which fields in the local files are updated.

GLOBALPW=yes

If the environment variable GLOBALPW in */etc/default/inet* is set to yes for this client, then the client has only limited access to global user IDs. In this case, the entries in the file */var/yp/pwpattern* are of no significance. "Limited access" means: only the *Username*, *UID*, *GID* and *Comment* fields are transferred every hour from the global user to the local */etc/passwd*. *Password*, *Homedirectory* and *Shell* are set to the default values *YP*, /tmp, /sbin/sh.

Users with this type of user name cannot log into the NIS client because they do not know the password. They can, however, use NFS services on this machine, for example, those which require user and group IDs that are unique throughout the network.

GLOBALPW=auto

If the environment variable GLOBALPW is set to auto, the client will update its local files according to the contents of */var/yp/pwpattern.*

Example

The */etc/yppasswd* file on the NIS master server and the associated maps contain the following entries:

```
fred:ab%d&hgx:2998:2500:this_is_fred:/home/fred:/sbin/ksh
anna:98796%*gGj:2999:2500:this_is_anna:/home/anna:/sbin/sh
tweety:Yh&#dF:3000:2500:this_is_tweety:/usr/tweety:/sbin/ksh
```

The client has set GLOBALPW=auto and is using the */var/yp/pwpattern* file (see Section "The /var/yp/pwpattern file"). It receives the following local entries after the global entries are transferred for the first time:

```
/etc/passwd
fred:x:2998:2500:this_is_fred*NIS*:/tmp:/sbin/ksh
anna:x:2999:2500:this_is_anna*NIS*:/home/anna:/sbin/sh
tweety:x:3000:2500:this_is_tweety*NIS*:/usr/tweety:/sbin/ksh
/etc/shadow
fred:ab%d&hgx:9280:2147483647:0::::
anna:*YP*:9280:2147483647:0::::
tweety:Yh&#dF:9280:2147483647:0::::
```

The home directory for fred and the password for anna will be set to the default values. They can subsequently be changed - by the system administrator or with his cooperation - to more appropriate values, which, as ordained by */var/yp/pwpattern,* cannot then be overwritten by the next update. The character string *NIS* in the comment field of */etc/passwd* marks these entries as global.

If the client has set GLOBALPW=yes, the first transfer of global entries results in the following:

```
/etc/passwd
fred:x:2998:2500:this_is_fred:/tmp:/sbin/sh
```

```
anna:x:2999:2500:this_is_anna:/tmp:/sbin/sh
tweety:x:3000:2500:this_is_tweety:/tmp:/sbin/sh
/etc/shadow
fred:*YP*:9280:2147483647:0::::
anna:*YP*:9280:2147483647:0::::
tweety:*YP*:9280:2147483647:0::::
```

Now only the names, UIDs and GIDs of the global users are known on the client. Note that the entries are now indicated as global by means of the *YP* string in the */etc/shadow* password field. This means an entry of this sort is no longer recognized as global if you issue it with a password on the client. A modification to the UID on the server would then no longer be transferred to the client, for example.

Updating of local users through the NIS master server

The transfer or updating of a single global user is explicitly initiated on the NIS master server by calling the *ryptrans* command. So that the local files on the client can be updated at all, GLOBALPW must, of course, not be set to no on the client. In addition, the *yptransd* daemon must be running on the client. *yptransd* is started by the */etc/init.d/yp* script when the system starts up.

The */var/yp/pwpattern* file on the client is always taken into account when global users are updated from the NIS master server. Therefore, it makes no difference in this case whether GLOBALPW is set to yes or auto on the client.

Example

```
ryptrans oslo fred
```

This updates the name of the user fred on the client oslo.

7.5.5 Setting up global users and groups on the NIS master server

The source files on the NIS master server for global users and groups are */etc/yppasswd* and */etc/ypgroup*. The */etc/yppasswd* file contains entries in the following format:

Username:Password:UID:GID:Comment:Homedirectory:Shell

These entries have to be made by the system administrator of the NIS master server. The format of the */etc/ypgroup* file is identical to that of */etc/group*. The NIS maps *passwd.byname*, *passwd.byuid*, *group.byname* and *group.bygid* are created from the source files */etc/yppasswd* or*/etc/ypgroup* using the following commands:

```
# cd /var/yp
# make passwd
# make group
```

The NIS maps are then available to the NIS clients in this format.

The NIS maps have to be created every time the source files are modified. Otherwise, the changes are not visible to the clients.

The system administrator can either use an editor to set up and modify the files */etc/yppasswd* and */etc/ypgroup* or do it from the *sysadm* user interface under the menu item *Global users and group administration*. Lines containing blanks or comments are not permitted in the */etc/yppasswd* file.

7.5.6 Password aging information for global users

The new procedure for global user administration makes it possible for the NIS master server to maintain aging information regarding the password of a global user and to distribute this to the clients. This information reveals when the password was most recently changed, expressed as the number of days since January 1 1970, and the permitted maximum and minimum number of weeks between two changes of the password.

This information must be encrypted in l64a(3C) format. It is recorded in the file */etc/yppasswd* on the NIS master server, after the encrypted password and separated from it by a comma.

You can use the command */usr/sbin/agestring* to convert the specified time to l64a format.

This command expects the following arguments

- date of the most recent change (0 = current date)
- maximum number of weeks before the password must be changed
- minimum number of weeks before the password may be changed

separated by spaces, in each case. It returns the encrypted password aging information which is then in the correct form to be entered in the file */etc/yppasswd*. NIS maps must now be regenerated.

The date of the last change for a global user is updated automatically when the command *yppasswd* is used to change the password.

Since there may be circumstances in which not all the clients will be able to evaluate aging information, the master server maintains two sets of NIS maps for the file */etc/yppasswd*

- The maps *passwd_pyr.** may contain password aging information,
- and the maps *passwd.** do not.

Password aging information is not visible to clients that were set up according to the old procedure. Such clients will access the *passwd.** maps.

7.5.7 Changing passwords for global users

It is not recommended to use the command *passwd* on the NIS client to change the password for a global ID, otherwise a locally changed password might be overwritten again during the next periodic update (see the section Updating of local files by the NIS client). Even on the NIS master server there would be problems with using *passwd* to change the password, because *passwd* modifies the file */etc/shadow* and not */etc/yppasswd*. For this reason, the entry "Days before password may be changed" in */etc/shadow* is automatically set to a very high value (2147483647 days).

Passwords for global users can therefore be changed from any NIS client using a special command, *yppasswdd*. This requires that the *in.yppasswdd* daemon must be running on the NIS master server. It is started there by the script */etc/init.d/yp* when the system starts up.

7.5.8 Entering global users using SIfmIlan

You can administer global users and user groups simply by editing the corresponding system files or with the *User* menu item in the *sysadm* menu *UNIX System V Administration*.

If you select this menu item, the selection form for administering local and global users will appear:

```
User Login and Group Administration
local  - Local User Login and Group Administration
>global - Global User Login and Group Administration
```

If you select *global*, the menu for administering global users and groups is displayed:

```
Global User Login and Group Administration
>add          - Add Users or Groups
list         - Lists Users or Groups
modify       - Modify Attributes of Users or Groups
remove       - Remove Users or Groups
start_dfs_id - Set Global User or Group Start-Id
```

add Add new global user groups or global users.

list List all global user groups or users.

modify
Modify the group number and group attributes of a global group. Modify the user attributes of a global user.

remove
Remove global users groups or users.

start_dfs_id
Set the group start ID or user start ID.

7.5.8.1 Adding global users or user groups

If you want to add global users or user groups, select the *add* menu item from the *Global User Login and Group Administration* menu.

Adding a new global user group

Select *group* in the *Add Global Users or Groups* input form in order to enter a new user group. Press SAVE and the following input form is displayed:

```
Add Global Group
Group name:□□□□□□□□□□□
Group ID:□□□□□□□□□□□□□2001
Members of Group:□□□□□□□□□□□□□□□□□□□ >
```

Group name
: Enter the name of the global user group you want to create. The name should not begin with an uppercase character.

Group ID
: The system suggests the next unassigned number as the group ID (2001 in our example). You can either accept this ID for the user group or enter a number of your choice.

Members of Group
: You can define several members of the group, but normally this field is left blank. The users are assigned to a group ID on the basis of their ID entries in the *passwd* user file. Any entries in this field are merely temporary assignments of users to an additional user group.

Adding a new global user

If you select *user* instead of *group* in the *Add Global Users or Groups* form, you will be prompted to enter a global user in the next form:

```
Add Global User
Comment:
Login Name:
User-ID:       2001
Group Name:
Home Directory: /tmp
Shell:         /usr/bin/sh
```

Comment
: You can enter an explanatory comment, for example the user's complete name.

Login Name
: Enter the user's login name. It should not begin with an uppercase character.

User ID
: The system suggests the next unassigned number as the user ID (2001 in this example). The menu system counts the user IDs internally, beginning with the start user ID, and suggests the next unassigned number as the ID. You can accept the suggestion or enter a number of your choice.

Group name
: Enter the name of the group in which the global user will work. You can list the possible group names by pressing CHOICES.

Home directory
: Enter the user's login directory. The login directory should already exist. The default is */tmp*.

Shell Enter the name of the program that will be loaded automatically every time the user logs on. There is a selection list.

7.5.8.2 Listing global users or user groups

On the master, you can check

- Which global user groups and global users exist

- Which group numbers are assigned to the global user groups
- If the global user group or user still exists
- If the global user group or user has already been entered

To do so, select the menu item *list* from the *Global User Login and Group Administration* menu. You can then select whether you want to inspect the list of user groups or the list of users.

7.5.8.3 Modifying attributes of global users or user groups

The menu item *modify* on the *Global User Login and Group Administration* menu enables you to modify the attributes of existing user groups or users.

7.5.8.4 Removing global users or user groups

The menu item *remove* from the *Global User Login and Group Administration* menu enables you to remove existing user groups or users.

7.6 Administering the global mailing list with SIfmllan

Just as you can create local mail groups, you can create global mail groups on the NIS master server. All the machines in the domain can access these mail groups.

Mail administration involves administering the global mailing list. The following can be entered in the mailing list:

- Mail groups
- The members of the mail groups
- Individual mail users

The members of a mail group can be other mail groups or individual mail users.

Example

There are three mailing lists:

- Mail_group_A
- Mail_group_B
- Mail_group_C

Mail users are entered in these mail groups, and, in the case of Mail_group_A, other mail groups are also entered.

Mail_group_A	Mail_group_B	Mail_group_C
Mail_user_1	Mail_user_11	Mail_user_21
Mail_user_2	Mail_user_12	Mail_user_22
Mail_user_3	Mail_user_13	
Mail_group_B		
Mail_group_C		

Mail sent to Mail_group_A, for example, is received by Mail_user_1, Mail_user_2, Mail_user_3, Mail_user_11, Mail_user_12, Mail_user_13, Mail_user_21 and Mail_user_22.

You can use any of the following formats for mail user entries in the mailing list (example):

maria		Format 1
kevin1	kevin	Format 2
paula	paula@boeing	Format 3

Description of formats

Format 1

maria is the mail userid of the user with the same name on the local machine.

Format 2

kevin1 is the mail userid and is associated with the user name kevin. This format assigns the mail userid kevin1 to the user kevin on the mail master. On the mail master, mail is delivered in accordance with the entries in the mailing list.

Format 3

paula is the mail userid and is associated with the user name paula and the name of a machine boeing. This format assigns the mail userid paula to the user paula on the machine boeing.

You can administer the mailing list by means of the menu item *mail-list* in the *Network Services Management* menu. The following menu is displayed:

```
Administration of Mailing List
add        - Add Mail Groups or Users
list       - List Mail Groups and Users
modify     - Modify Mail Group Membership
print      - Print Mailing List
remove     - Remove Mail Groups or Users
```

add Enables you to define mail groups and mail users.

list Enables you to view the current mailing list.

modify

Enables you to extend or shorten an existing mailing list.

print Enables you to print a hardcopy of the current mailing list.

remove

Enables you to remove mail groups and mail users from the mailing list.

Please make a note of the following if you want to create a new mailing list on the master:

- Before you can create mail groups, you must first define mail users. Only a user having a global userid can be assigned a mail userid.
- Once you have created the list of mail users, you can group the users in the list into mail groups.
- Once you have created mail groups, you can go on to create other mail groups which group together mail userids and mail groups.

7.7 Summary of NIS commands

domainname

Checks or sets the domain name.

ypinit This is the central utility for setting up machines which use the NIS. *ypinit* can be used for initializing clients, NIS master servers and NIS slave servers.

ypwhich

Indicates which NIS server is currently being used by a client. If called with the *-m mapname* option, it indicates which NIS server is the NIS master server of the specified maps.

make On the NIS master server, it updates and distributes the NIS maps by processing the *makefile* in the */var/yp* directory. You can use *make* for updating the standard maps or also for updating your own maps.

makedbm

Takes a file and converts it into the *dbm* files **.dir* and **.pag*, which are valid *dbm* format files. These are used as maps by NIS. You can also use *makedbm -u* in the other direction to make the data in a map (key/value pairs) visible in ASCII format.

ypxfr Copies an NIS map from a server to another machine. *ypxfr* is called on the machine that wants the map.

yppush

Copies a new version of an NIS map from the NIS master server to its NIS slave servers. *yppush* is run on the NIS master server.

ypset Tells a *ypbind* process to bind to a named NIS server. *ypset* is not for casual use.

yppoll tells which version of an NIS map is running on a server that you specify. It also lists the master server for the map.

ypcat Displays the contents of an NIS map.

ypmatch
Prints the value for one or more specified keys in an NIS map. You cannot specify which NIS server's version of the map you are seeing.

yppasswd
Changes the password of a global user.

7.8 Error analysis and diagnostics

In this section, various problems are described that may occur when you are working with the NIS. You are told what the cause of the problem may be and how to deal with it.

Machine cannot be connected

Problem:
No NIS server is found.

Cause:
The most frequent causes are:

1. No NIS server exists (the only NIS server is, for example, switched off).
2. There is a gateway (SIfmllan) between the client and the NIS server.
3. The broadcast address is incorrect (SIfmllan).
4. The */var/yp/binding/[domainname]/ypservers* file does not contain the correct entries.
5. The NIS server exists but is inaccessible (i.e. *ping* on the server does not work).

Solution:

For 1:

Check the following:

- Is the NIS master server accessible?
- Is the *ypserv* daemon active on this machine?
- Is it registered with the port mapper *rpcbind*?

For 2:

If the client attempts to determine its NIS server via a broadcast call, there must be no gateway between the client and the NIS server. As a rule, gateways do not forward broadcast calls.

Broadcast calls are always issued if the file */var/yp/binding/[domainname]/ypservers*

- does not exist
- the first line contains the character + or *
- if the machine has just been connected via the Slfmllan menu system.

If a */var/yp/binding/[domainname]/ypservers* file already exists containing + or *, delete the first line containing + or *. If such a file does not exist, create it and enter the name of the NIS master server or slave from the required domain in the first line.

The *ypbind* daemon does not issue a broadcast call in either case, but tries to reach the named machines directly. Check whether the servers entered are accessible with the *ping* command.

For 3:

A potential NIS server cannot recognize a broadcast call if the client issues it incorrectly.

This is often the case if the NIS server can only deal with an old broadcast message (machine address consists entirely of zeros), but the client is sending a new broadcast message (machine address consists entirely of the digit 1).

Problems can also occur when subnets are set up but the NIS server is not able to deal with them or the subnet mask is inconsistent.

Correct the OLDBROADCAST entry in the */etc/default/inet* file. You can change the subnet form using the *configuration* menu after calling *sysadm*.

The broadcast address and subnet mask must correspond on all participating machines.

For 4:

Check whether the */var/yp/ypservers* file on the NIS master server includes the machine names twice on each line (see also "For 2:" above).

For 5:

Use the *netstat -i* command to check whether network interfaces have been incorrectly configured and the *netstat -r* command to check whether the routing is correct.

Problems that occur when working with the Network Information Service

Problem:

A command is hung up and the following message is issued at the console:

yp: server not responding for domain [domain]. Still trying

Cause:

This message indicates that the *ypbind* daemon on the local machine cannot establish a connection to a *ypserv* daemon in the domain with the name [*domain*].

This happens if:

- None of the NIS servers on which the *ypserv* daemon runs is active.
- The NIS server(s) is (are) so overloaded that the *ypserv* daemon cannot respond within the specified time interval.

If the causes mentioned above are the reason for the problem, the other clients in the network will also be affected. This situation is usually temporary.

Solution:

- Restart the NIS server, or
- Wait until it is ready to function.
- Activate the *ypserv* daemon again.

Problem:

- Some commands appear to function correctly, while other commands result in error messages about the NIS Service not being available.
- Some commands or daemons abort with or without an error message.

For example, the following error messages may be issued:

Input at local machine:
ypcat file_name

Output at local machine:
ypcat: can't bind to yp server for domain [domain].

Reason: can't communicate with ypbind

Cause:

These symptoms normally indicate that the local *ypbind* daemon is not running.

Solution:

Use the *ps* command to check whether the *ypbind* daemon is running.

If the *ypbind* daemon is not running, it must be activated. After this has been done, you should be able to access the Network Information Service again.

Input at local machine:
ps -ef | fgrep yp

Output at local machine:
root 551 1 0 10:54:46 ? 0:01 /usr/lib/netsvc/yp/ypserv

root 591 1 0 10:54:47 ? 0:01 /usr/lib/netsvc/yp/ypbind

root 4430 1 0 14:54:56 tty000 0:00 fgrep yp

If *ypbind* appears as a process in the list output, *ypbind* must be removed from the table using the *kill* command before the system is restarted.

Input at local machine:
/usr/lib/netsvc/yp/ypbind

The ypbind daemon fails

Problem:

The *ypbind* daemon fails as soon as the system is started up.

Cause:

The cause is probably to be found in another part of the system.

1. Check whether the *rpcbind* daemon exists:

   Input at local machine:
   ps -ef | fgrep rpcbind

   Output at local machine:
   root 32 1 0 10:54:47 ? 0:01 /usr/sbin/rpcbind

   root 4461 1 0 14:54:56 tty000 0:00 fgrep rpcbind

2. If the *rpcbind* daemon is active there are fundamental problems.

Solution:

For 1.

The most frequent cause of error when the *rpcbind* daemon will not run is an incorrectly modified machine name. Check whether your machine name is entered twice in the */etc/net/*/hosts* files. Otherwise, set the machine name using the *sysadm* menu and restart the system.

For 2.

Check whether the local *rpcbind* can be addressed from a different machine:

Input at a different machine:
rpcinfo -p localhostname

Output at the other machine:
program vers proto port

. . . .

100007 3 tcp 2116 ypbind

100007 3 udp 2834 ypbind

. . . .

These two entries represent the *ypbind* daemon. If these entries are not output, the *ypbind* daemon on the local machine is not able to register its services. The system should be restarted.

If the situation has not changed after the system has been restarted, contact your customer service department.

The ypwhich command appears to be inconsistent

If at your local machine you issue the *ypwhich* command several times, a different machine name may be output each time. This is normal, since the NIS server providing the NIS service can change, depending on the load.

Problem:

ypwhich returns the name of its NIS server in the following format:

IP address.no1.no2

Cause:

The machine name of the NIS server was not known when the first *ypwhich* was executed (i.e. was not included in the file */etc/inet/hosts*).

The IP address is the IP address of the NIS server. The two numbers *no1.no2* indicate the port number (= *no1**256+*no2*) of *ypserv* on the NIS server.

Solution:

Enter the machine name in */etc/inet/hosts* and start *ypbind* again.

Different versions of the NIS files exist

Reliant UNIX slaves automatically update their NIS files when the system is booted. This cannot be guaranteed for computers from other manufacturers. In this case, different versions of an NIS file may exist if one of the slaves was not active when the NIS file was copied to the slaves.

If all the slaves in a domain are active, there should never be different versions of NIS files.

Problem:

The Network Information Service data is inconsistent.

Cause:

1. A slave is not able to receive the most recent versions of the NIS files.
2. The file directory */var/yp/[domain]* is present on a machine but it is not recorded in the network administration file *ypservers*.

Solution:

For 1.

The *ypxfr* command must be issued interactively at this slave:

Input at the slave:
ypxfr mname

If the *ypxfr* command is not successful, a message is issued telling why the command did not function, and the error should be dealt with.

If the *ypxfr* command is completed successfully, but you do not think this is always the case, you should generate a file into which the *ypxfr* command can write all messages.

Input at the slave:
cd /var/yp

touch ypxfr.log

Every action carried out by *ypxfr* is entered in this file along with the system time. When the problem has been located, you should delete the file because otherwise it could get very big.

If this does not help, check the file that activates *ypxfr* (*var/spool/cron/crontabs/root*).

You should also ensure that the slave has been entered in the NIS file *ypservers* on the master.

If you cannot localize the problem, you should deactivate the network software on the machine that is not functioning correctly using the menu selection and then reactivate it.

For 2:

Input at local machine:
ypwich

Output:
hostname

Input:
ypmatch *hostname* ypserver

Output:
Can't match *hostname* map ypserver.□
Reason: no such key in map.

The machine with the name *hostname* is obviously not a registered slave. It must be entered on the master or all the NIS functions on this machine must be deactivated.

Problem:

NIS data that was modified on the master does not appear on a particular slave.

Cause:

The version number of the NIS files on the master is not identical to the version number on the slave.

Input:
/usr/sbin/yppoll -h [*master*] hosts.byname

Output:
Domain [*domainname*] is supported.

Map hosts.byname has order number 573908022.

The master server is [*master*]

Input:
/usr/sbin/yppoll -h [*slave*] hosts.byname

Output:
Domain [*domainname*] is supported.

Map hosts.byname has order number 573908022.

The master server is [*master*]

In this case the version numbers are identical.

1. If the version number on the master is larger than the number on the slave, the slave does not have the most recent version of the NIS data.
2. If the version number on the master is smaller than the number on the slave, the NIS data will not be updated until the version number on the master is larger.

Solution:

For 1.

Check whether the slave has been entered in the NIS file *ypservers*. If not, the slave must be entered.

For 2.

Force the transfer of the NIS data.

Input:
ypxfr -f hosts.byname

The *-f* argument forces the transfer of NIS data even if the version number on the slave is larger than the version number on the master.

The ypserv daemon is terminated prematurely

Problem:

The *ypserv* daemon fails as soon as it is activated.

Cause:

The cause is probably to be found in another part of the system.

Solution:

Check whether the *rpcbind* daemon exists:

Input at local machine:
ps -ef | fgrep rpcbind

Output at local machine:
root 32 1 0 10:54:47 ? 0:01 /usr/sbin/rpcbind

root 4461 1 0 14:54:56 tty000 0:00 fgrep rpcbind

If the *rpcbind* daemon will not run, the most frequent cause of error is an incorrectly modified machine name. Check whether your machine name is entered twice in the */etc/net/*/hosts* files. Otherwise, set the machine name using the *sysadm* menu and restart the system. If the *rpcbind* is active, there are more serious problems. Check whether the local *rpcbind* daemon can be addressed from another machine.

Input at another machine:
rpcinfo -p lochost

Output at the other machine:
program vers proto port

. . . .

10004 2 udp 1031 ypserv

10004 2 tcp 1030 ypserv

. . . .

These two entries represent the *ypserv* daemon. If these entries do not exist, the *ypserv* daemon on the local machine is not able to register its services. The system should be rebooted.

If the situation has not changed after the system has been restarted, contact your customer service department.

Problem:

Separate NIS files have been generated and the following message is issued:
Can't get maplist: Reason: internal yp server or client error

Cause:

The name of the NIS file has not been included in the */var/yp/aliases* file

Problem:

The following error text is issued in conjunction with other message texts:
Reason: RPC: Name to address translation failed - n2a: hostname not found

Example

Input:
yppoll -h chicago ypservers

Output:
Can't create connection to chicago

Reason: RPC: Name to address translation failed - n2a: hostname not found

Cause:

The machine name specified has not been entered in the file */var/inet/hosts* or a typing error has been made.

Solution:

Enter the machine name in the NIS file *hosts*.

Problems with global user names

Problem:

The *yppasswd* command always terminates with the message:
couldn't change password

Cause:

The "old password" entered may have been incorrect.

Solution:

Edit the */etc/yppasswd* file on the NIS server. Delete the password fields of the relevant entries.

Configure the global users as described in the Section "Setting up global users and groups on the NIS master server".

Try once more to define a password using the *yppasswd* command.

Reply to the "Old NIS password:" prompt by pressing the [RETURN] key.

Problem:

NIS overwrites the HOME directory if it does not correspond to the global directory.

Cause:

The GLOBALPW variable is set to new or auto.

Solution:

There are a number of options available for solving this problem:

1. Set the GLOBALPW variable to =yes (restricted old SINIX format)

   This ensures that only user and group names and IDs are updated.

2. Set the GLOBALPW variable to =auto

   The */var/yp/pwpattern* file then contains the following entry

   *:YES:UID:GID:YES:NO:YES

3. Set a symbolic link from the global HOME directory to the local HOME directory.

   This solution is general and does not depend on the GLOBALPW variable.

Destroyed files

Problem:

The */etc/passwd* or */etc/shadow* file is empty or has been destroyed.

Cause:

Panic

Solution:

Find out if the file was saved in the */etc/opasswd* or */etc/oshadow* directory and copy it back to */etc/passwd* or */etc/shadow*.

If */etc/opasswd* and/or */etc/oshadow* are not available either, */etc/passwd* or */etc/shadow* will have to be reloaded from a backup.

8 Network monitoring with SNMP agents

The failure of systems, applications or components such as bridges, routers or gateways within a network may lead to major problems entailing high costs. It is therefore important to monitor networks, including their systems, applications and components, to detect potential problems and to take appropriate measures in good time.

Monitoring of this kind can be realized using the Simple Network Management Protocol (SNMP), a standardized protocol for managing TCP/IP networks.

8.1 Network managers, agents and communities

Every network has one or more systems equipped with a network manager that monitors and manages the network, i.e. its components, systems and applications. One example of such a manager is the TransView SNMP network management platform. Managers use the SNMP protocol to communicate with SNMP agents, which must be installed on the network elements that are to be monitored.

Since the monitoring of the network is controlled by the central manager, the burden on the components being monitored is kept to a minimum. An SNMP agent waits until it receives a message from its manager. Only then does the agent become active and provide the information requested by the manager. This makes it possible to place any UNIX system under SNMP-based network management.

The relationship between a group of SNMP agents and a manager assigned to them is defined as a "community". Under EMANATE (see section 8.2), a community always refers to exactly one manager. A community is defined through a character string (the community string) which is delivered with every message and thus expresses membership in the community. EMANATE master agents do not support a special role of the community string ANY, as other SNMP agents do.

A community string is also associated with access permissions such as "read-only" or "read-write". These access permissions can, however, only restrict and not extend the operations specified for individual objects in the MIB Management Information Base definition (e.g. the RFC 1213 MIB II standard). If the MIB definition specifies "read-only" for a given object, "read-write" cannot be set even if the community string is associated with the "read-write" access permission.

The following example is intended to illustrate the use of community strings and access permissions:

Example:

An SNMP agent belongs to a community named public, which has "read-only" access permission. The public community also includes a manager which can query information from this SNMP agent by sending an appropriate message accompanied by the community string public. The SNMP agent also belongs to a second community named net_5, which is associated with "read-write" access permission. The net_5 community has another manager. In this example, only the manager of the net_5 community is entitled to perform write operations via the SNMP agent.

If special events occur in a network component, the SNMP agent can send a trap message to one or more managers to inform them of this. Based on the trap message, the manager(s) can react to the events appropriately. The right of an SNMP agent to send a trap message to a manager is also expressed by the community string. When sending a trap message to a manager, an SNMP agent must use the appropriate trap community string in order for the manager to accept the message.

8.2 EMANATE agent architecture

One problem which has persistently plagued SNMP users has been to find a truly simple means of enhancing existing agents with new MIB objects. EMANATE offers a solution to this problem.

EMANATE stands for **E**nhanced **MAN**agement **A**gent **T**hrough **E**xtensions and is a concept and an architecture with a system-independent interface for making agent extensions. The most important features of EMANATE are:

- Principle of master agent/subagents

- Asynchronous handling of requests
- Dynamic loading of MIB extensions
- System-independent subagent API

- MIB tools for generating subagent skeleton code in C □
  (only with Subagent Development Kit)
- Trilingual agent concept (SNMPv1, SNMPv2c and SNMPv2*)

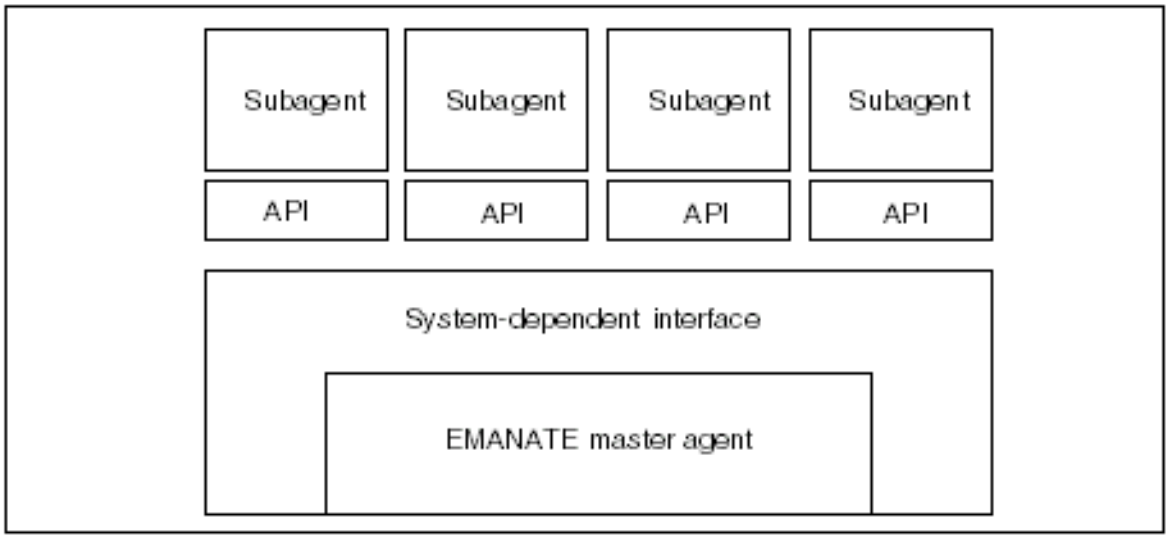


Figure 13: EMANATE agent architecture

SNMPv2c and SNMPv2* are variants of the SNMPv2 protocol (RFC1902 to RFC1908), which can be differentiated on the basis of their administration models. SNMPv2c is the community-based model in accordance with RFC1901, while SNMPv2* is the user-based model proposed by SNMP research.

Since none of the SNMPv2 models has been adopted de-facto to date, attempts are currently underway to merge the different models and to establish SNMPv3 as an internationally valid standard.

In EMANATE, the SNMP agent consists of an EMANATE master agent and one or more subagents. The functionality of this SNMP agent is determined to a large extent by the subagents.

All EMANATE-based SNMP agents are enhanced by adding further subagents. This does not require the manager to have any special information on the subagents. From the viewpoint of the manager, there is never more than a single SNMP agent, regardless of how many subagents there are. When a subagent is added, the only difference noticeable from the viewpoint of the manager is the enhanced functionality of the SNMP agent, as evidenced by, for example, new MIB objects. In the EMANATE architecture, the master agent performs the following tasks:

- Handling of the SNMP protocol
- Authentication, authorization, encryption
- Implementation of SNMPv1, SNMPv2c and SNMPv2*
- Asynchronous communication with subagents

Incoming requests are forwarded by the master agent to the relevant subagents.

Vis-à-vis the master agent, subagents should be designed as simply as possible. They have the following characteristics:

- Independence from the SNMP version
- Provision of methods
- Dynamic activation and deactivation of MIB information
- Realizable as DLLs (dynamic link libraries) or separate processes

Further options of enhancing the EMANATE master agent are offered by the TransView range of products.

8.3 SNMP agent

The Reliant UNIX operating system includes an SNMP agent consisting of the following components:

- EMANATE master agent (*SIsnmpdm* package)

- MIB II extension (*SImib2* package)
- Reliant UNIX SNMP agent adapter (*SIsnmpd* package)

Both the Reliant UNIX SNMP agent adapter (*SIsnmpd*) and the MIB II extension (*SImib2*) are implemented as EMANATE-based subagents. The Reliant UNIX SNMP agent adapter provides an execution environment for agent extensions which require an older version of the SNMP agent *SIsnmpd* (except for proxy extensions of the SNMP agent *SIsnmpd* V2.0).

The SNMP agent provides full support for the handling of MIB II objects (RFC1213) by a manager. The table below shows the categorization of MIB II objects into object groups and their respective assignments to the supporting components:

Level	Component	Information
System	Master agent	Node name, operating system version, name of person to contact, machine location, etc.
SNMP	Master agent	Statistics
IP	Subagent SImib2	Subnet mask, broadcast address, routing table and statistics
ICMP	Subagent SImib2	Statistics
TCP	Subagent SImib2	Number of active and passive connections, statistical data, TCP connection table, etc.
UDP	Subagent SImib2	UDP table

Table 3: Agent operations

For a detailed description of all MIB II objects (RFC1213), see the file */usr/lib/snmpd/doc/mib-II.doc*.

8.4 EMANATE master agent

The EMANATE master agent is the basic agent on which all other agent enhancements build.

8.4.1 Starting and stopping the master agent

The EMANATE master agent is started automatically during system startup as soon as run level 2 is reached, and stopped automatically during system shutdown when the system reaches run level 0 or switches to single-user mode (see "Reference Manual for System Administrators" [8], *init(1M)*).

If you have root privilege, you can also start or stop the EMANATE master agent by entering the command:

Syntax

/etc/init.d/snmpdm[start|stop]

Parameters

start Starts the EMANATE master agent running as a daemon process and the MIB II extension ema.mib2d. This means that, following startup, there is full MIB II support. In addition to an appropriate message, the system outputs the values of certain objects at startup time and returns the prompt. The agent then waits in readiness to receive messages from managers.

If the daemon process is already active when you call the startup script */etc/init.d/snmpdm*, the call has no effect.

stop Stops the EMANATE master agent and the MIB II extension ema.mib2d.

8.4.2 Entering settings for IP services

The EMANATE master agent offers services to the network manager. These services are entered in the */etc/services* file, which lists the names and port numbers of the available IP services. This information is read

by programs requesting network services. The */etc/services* file is created automatically when TCP/IP is installed and already contains the entries for the EMANATE master agent.

8.4.2.1 The services network services file

When the EMANATE master agent is installed, however, a check is once again carried out to determine whether these entries are present. RFC1157 stipulates that the */etc/services* file must contain the following entries for the EMANATE master agent:

```
snmp        161/udp
snmp-trap    162/udp
```

The names of the network services are shown in the first column. The second column contains the port number of the service and the name of the transport protocol, here UDP, under which the service operates.

If these entries are not yet present, they are automatically entered in the local */etc/services* file. However, if the network services file is administered centrally by your network administrator via NIS, you must ask the administrator to make an appropriate entry in the */etc/services* file. Otherwise, the entry required for the EMANATE master agent in the local file will be overwritten when the local */etc/services* file is updated from the central file. This is why the screen displays the following message:

If you are running NIS, add these services in your NIS-Master-Server!

8.4.2.2 Using nonstandard ports

In some cases, however, it may make sense to use ports other than those defined in RFC1157:

- when testing an agent,
- when using an agent in a customer-specific network environment or
- when an agent is in a proxy configuration.

You can define port numbers which differ from the standard by means of environment variables:

SR_SNMP_TEST_PORT
: The value of this variable defines the port number used for all SNMP communication except for SNMP traps.

SR_TRAP_TEST_PORT
: The value of this variable defines the port number used for all SNMP traps.

If only the variable SR_SNMP_TEST_PORT and not the variable SR_TRAP_TEST_PORT is defined, the port number of the SNMP traps is:

SR_SNMP_TEST_PORT + 1

If neither of these variables is defined, the settings in the */etc/services* file are used.

8.4.3 Configuring the master agent with sysadm

Following installation of the EMANATE master agent under the Reliant UNIX operating system, you have the option of using *sysadm* to start, stop or configure the agent via menu.

Note that your configuration data is initially written only to the temporary file */etc/snmp/agt/snmp.cnf.tmp*. Only when you open the *Config* menu and activate the Update function will your configuration actually be transferred from the master agent to the *snmpd.cnf* file, and the temporary file deleted.

If you do not activate the Update function, the *snmp.cnf.tmp* file will be overwritten, and your modified configuration data will be lost.

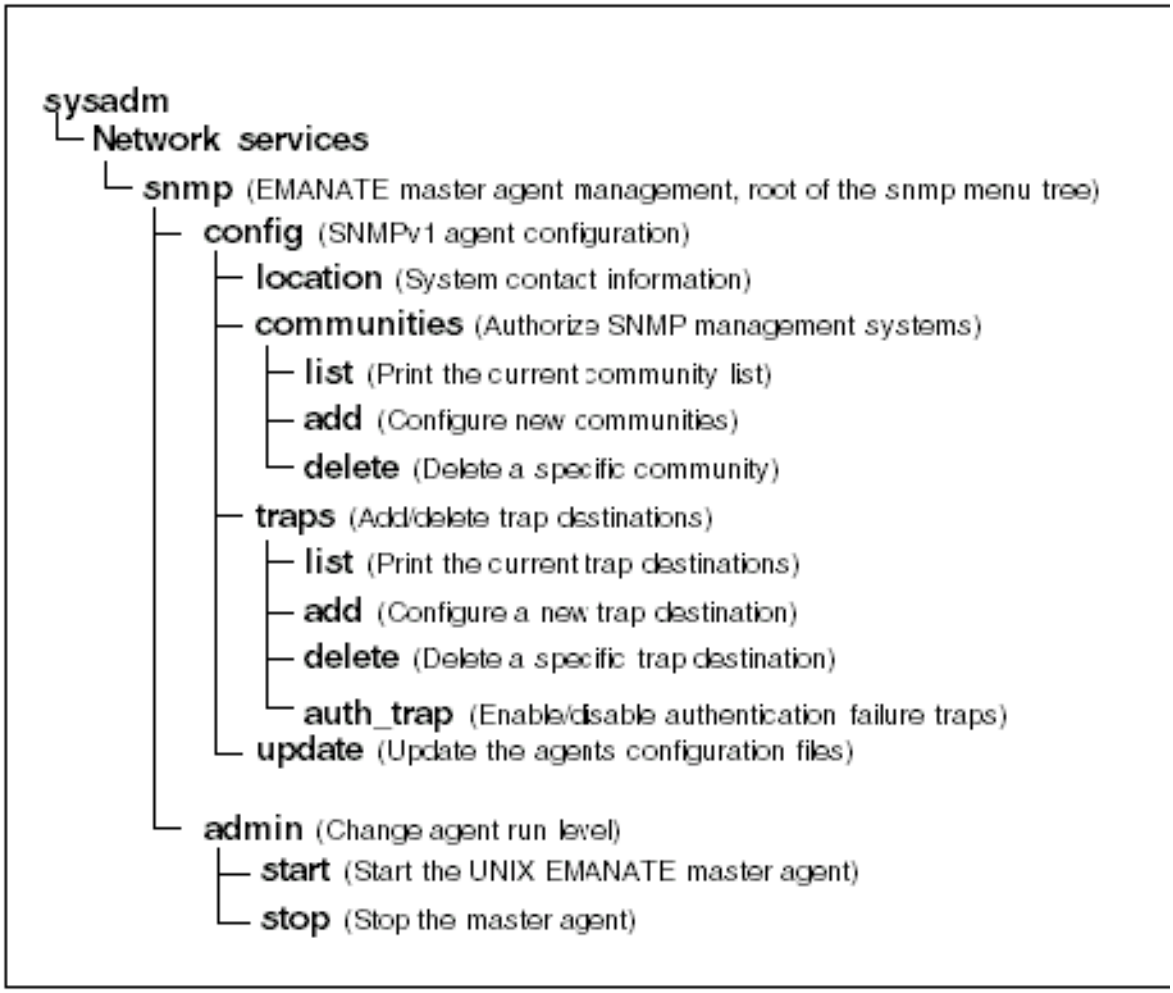


Figure 14: snmp menu tree in sysadm

8.4.4 Configuring the master agent without sysadm

The EMANATE master agent supports SNMPv1 but uses SNMPv2* internally. The master agent configuration is based on SNMPv2*. The earlier configuration which complies with RFCs 14xx is no longer supported.

In order to continue providing support for SNMPv1, the SNMPv1 configuration will be mapped to SNMPv2*. In principle, SNMPv1 and SNMPv2* can be differentiated on the basis of how they handle access rights using community strings and how read and write permissions are defined.

The full SNMPv2* configuration is stored in the */etc/snmp/agt/snmpd.cnf* file. The SNMPv1 parameters are configured in a separate */etc/snmp/agt/v1.cnf* file.

Configuring the EMANATE master agent without the aid of a sysadm interface involves three steps:

1. Editing the *snmpd.cnf* file.
2. Editing the *v1.cnf* file.
3. Stopping and restarting the master agent (see ...).

8.4.4.1 Editing the snmpd.cnf file

If your operating system does not provide a sysadm interface, you can configure the master agent by editing the ASCII file *snmpd.cnf* in the */etc/snmp/agt* directory or in the directory defined by the SR_AGT_CONF_DIR environment variable.

The full SNMPv2* configuration is stored in the */etc/snmp/agt/snmpd.cnf* file and is described by seven tables:

- userNameTable
- v2ContextTable
- viewTreeTable
- acTable
- communi
- notifyTable
- transportTable

These tables should not be modified.

You can make further entries in the *snmpd.cnf* file to define the presettings for MIB-II variables in the system group.

Lines containing configuration data are based on one of the following syntaxes and have the following structure:

tag value

The *tag* parameter assumes the name of one of the following MIB II variables: sysObjectID, sysLocation, sysContac, snmpEnableAuthenTraps.

The parameter *value* defines the value of the MIB II variables listed under *tag*. In each case, the syntax of the *value* parameter depends on the *tag* parameter specified.

Possible syntaxes

sysObjectID *OBJECT IDENTIFIER*

OBJECT IDENTIFIER
Object identifier of component.

sysLocation "*DisplayString*"

DisplayString
String of 0 to 255 characters in length. This string describes where the component is located.

sysContact "*DisplayString*"

DisplayString
String of 0 to 255 characters in length. This string contains the name of the contact person, along with the necessary data.

snmpEnableAuthenTraps 1|0
When a master agent receives a request, its first step is to run an authentication check. If the result of the check is negative, the request is ignored. If an authentication-failure trap was configured, an authentication-failure message will then be sent to the manager.

The variable *snmpEnableAuthenTraps* can assume the values 0 (disable) or 1 (enable).

Other parameters

The following parameters can be used to set internal limits for the master agent in the file (see the appropriate manual pages):

- MAX_THREADS
- MAX_PDU_TIME
- MAX_OUTPUT_WAITING
- MAX_SUBAGENTS

Sample configuration file

```
sysObjectID    1.3.6.1.4.1.231.1.2.5
sysLocation    "karpathos"
sysContact     "Dilbert"
snmpEnableAuthenTraps 1
```

8.4.4.2 The v1.cnf file

The SNMPv1 write and read rights, community strings etc. are mapped to SNMPv2* in order to continue providing support for the familiar and straightforward SNMPv1 configuration.

The SNMPv1 configuration is managed in a separate *v1.cnf* or *v1.cnf.host* file. Different tools can be used to generate this file from the *snmpd.cnf* file or to add it to the *snmpd.cnf* file.

v1dump
generates the *v1.cnf* file from the *snmpd.cnf* file

v1init adds entries from the *v1.cnf* file to the *snmpd.cnf* file

v1download
generates the local *v1.cnf.host* file from the *snmpd.cnf*file on a remote machine

v1upload
adds entries from the local *v1.cnf.host* file to the remote *snmpd.cnf*

For further information, please refer to the manual pages for *v1dump*, *v1init*, *v1.cnf*, *v1download*, *v1upload*.

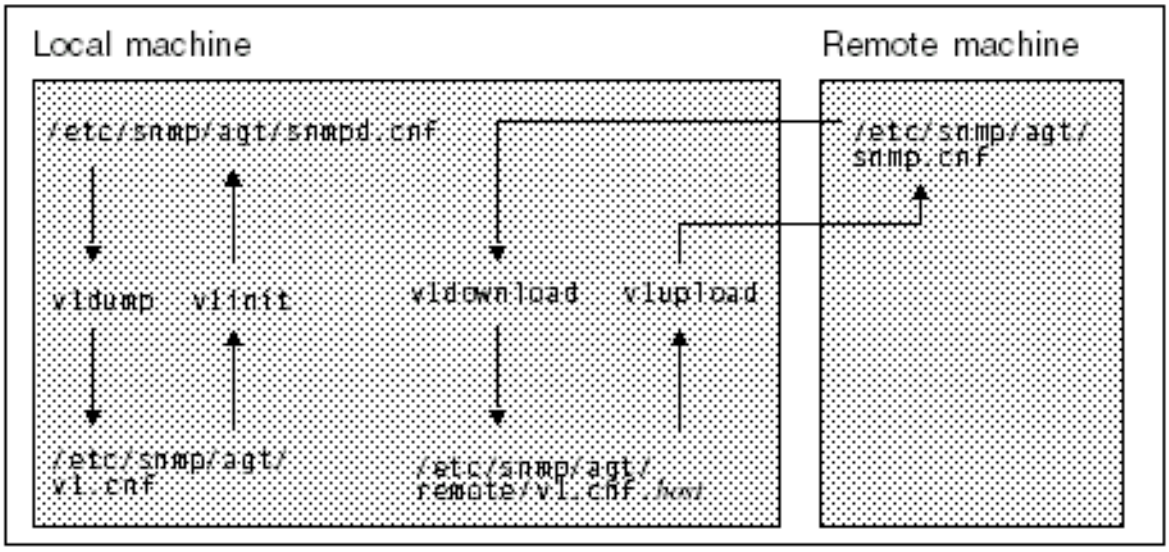


Figure 15: Interaction of tools for generating EMANATE configuration files

The *v1init* script has to reinitialize the local SNMPv2* configuration in accordance with the SNMPv1 entries in *v1.cnf*. A configuration over and above SNMPv1 is lost. However, because of the current status of SNMPv2*, it is not advised to use SNMPv2* intensively.

The *sysadm* menus can still be accessed locally using these four (*v1**) tools without modification (see Section "Configuring the master agent with sysadm"). The same points must be noted when using sysadm as for the *v1init* script.

Possible syntaxes in the v1.cnf file

community *community-string IP-address rights*

community-string
: Any character string. In the event that a community string occurs two or more times, the definition found first is used.

IP-address
: Defines the IP address for which the community string is valid. If the value 0.0.0.0 is specified for *IP-address*, any system under that community string can communicate with the agent.

rights Defines the access rights. *rights* can have the value read or write. The value read stands for read-only, while the value write stands for read-write.

trap *community-string IP-address*

community-string
: Any character string.

IP-address
: Defines an IP address for which the community string is valid. It is not permissible to specify the address 0.0.0.0.

Example:

```
community mystring    111.22.111.22 read
community yourstring  111.22.111.23 read
community superstring 111.22.111.24 write
trap      public      123.45.123.45
```

Local configuration

You have to work through the following steps in order to change the SNMPv1 rights:

1. Stop the master agent

2. Invoke *v1dump*

3. Edit the *v1.cnf* file
4. Invoke *v1init*
5. Start the master agent

The *v1init* procedure has to reinitialize the local SNMPv2* configuration in accordance with the SNMPv1 entries in *v1.cnf*. A previous configuration over and above than SNMPv1 is lost.

You have to work through the following steps if you simply want to modify the scalar objects in *snmpd.cnf*:

1. Stop the master agent
2. Edit the *snmp.cnf* file
3. Start the master agent

Remote configuration

As described above, an SNMPv1 configuration is also enabled on a remote system using the *v1download* or *v1upload* tool. The configuration information is fetched with SNMP-Get (*v1download*) and set with SNMP-Set (*v1upload*). A write community is required on the remote machine in both cases.

Example:

The MASTER_1 manager is entered on the servers SLAVE_1 to SLAVE_15 as a trap receiver and as a write enabler with the community secret1.

A second MASTER_2 (123.44.55.66) manager should be entered on all servers with the community secret2:

```
# uname -n
MASTER_1
# cd /etc/snmp/agt/remote
# for N in 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
> do
> v1download SLAVE_$N secret1
> echo "community 123.44.55.66 secret2 write" >> \ v1.cnf.SLAVE_$N
> echo "trap 123.44.55.66 secret2" >> v1.cnf.SLAVE_$N
> v1upload SLAVE_$N secret1
> done
```

8.5 Management tools

For simple tests, you can also use the following management tools from the */usr/sbin* directory to issue SNMP requests. You can call these tools on UNIX systems from the command level.

8.5.1 getone, getnext, getmany

For Get-Request calls, you can use the three management tools below. The syntax of their calls is to a large extent the same. The only differences between them are in the specification of the value for the parameter *variable_name*. Results are output to standard output.

getone
: Returns the value of exactly one instance.

getnext
: Returns the value of the object which lexicographically follows the specified object identifier.

getmany
: Returns the values of the objects of an entire group or a MIB.

Syntax

getone [-timeout *sec*] [-retries *num*] *agent_addr community variable_name ...*
getnext *agent_addr community variable_name ...*
getmany *agent_addr community variable_name ...*

Parameters

-timeout sec
Indicates the timeout value in seconds

-retries num
Specifies the number (*num*) of retries

agent_addr
IP address of the agent system

community
SNMPv1 community string

variable_name
Identifier of an object or instance. This identifier must be specified differently depending on which Get-Request command is used.

getone	Indexed identifier for an instance
getnext	Object identifier in dot notation, name in the form in which it appeared in the MIB
getmany	Group name in dot notation or as a MIB variable from the MIB document

Example:

```
/usr/sbin/getone yoursystem public sysName.0
/usr/sbin/getnext yoursystem public system
/usr/sbin/getmany yoursystem public iso
```

8.5.2 setany

Root privilege is required to execute the management tool *setany*. With *setany*, you can modify the values of instances on the agent system.

Syntax

setany [-timeout *sec*] [-retries *num*] *agent_addr community*
□□□□□□□*variable_name type value* [Ë*variable_name type value*]..

Parameters

-timeout sec
Indicates the timeout value in seconds

-retries num
Specifies the number (*num*) of retries

agent_addr
IP address of the agent system

community
SNMPv1 community string

variable_name
Identifier of an instance in dot notation. This instance identifier may also contain symbolic names, provided the relevant MIB is known on the management system.

type ASN.1 data type of the instance; optional field except for *DisplayStrings*. The *type* parameter may assume the following values:

-i INTEGER
-o OCTET STRING
-d OBJECT IDENTIFIER
-a Ip-Address
-c Counter
-g Gauge
-t TimeTicks
-D DisplayString
-N NULL

value New value of the instance

Example:

/usr/sbin/setany yoursystem superstring sysContact.0 -D "Mr. X"

The community string superstring must permit write accesses to the MIB objects of the yoursystem system.

8.5.3 traprcv

The trap-receive call *traprcv* provides you with information characterizing received traps. This call has the prerequisite, however, that the system where the command is being issued must be specified in the trap configuration of the sending system, and that the trap port (162) not be in use, e. g. by TransView SNMP, Trap Distributor, etc. Root privilege is required to execute the command *traprcv*.

The information provided consists primarily of information relating to the components of the trap-type macro (RFC 1215), the community string, the IP address of the sending system and a time stamp.

Syntax

traprcv [-d]

-d If the *-d* option is specified, the trap is also output as a hexadecimal dump.

9 The Basic Network Utilities (BNU)

The Basic Networking Utilities (BNU) allow UNIX machines to communicate with each other and machines of other vendors directly or via dial-up connections.

The BNU connection mechanism is relatively complex. It originates from a time when LAN networking was not as widespread as it is today. The product was developed by a vendor of telephone systems, so many of the functions are designed for this vendor's systems. Nevertheless, these functions have become a global UNIX standard and are therefore incorporated in Reliant UNIX 5.45.

The chapter begins with an overview of how the Basic Networking Utilities work and an introduction to the components involved in the process (commands, daemons, shell scripts, and files). Keep this overview in mind as you begin to work with BNU; it will help you keep track of where you are in the overall process. Following the overview, the basic procedures involved in BNU administration are described.

Sections detailing the structure and format of the various BNU database support, administrative, and *uucp* log files follow. As you gain more experience with BNU administration, these sections can be used as reference material for fine tuning your machine's configuration.

9.1 How the BNU work

The BNU components are installed with the operating system. After appropriate configuration, your machine will be ready to communicate with other machines. Here is a general view of how BNU accomplishes this communication process:

1. A user issues a command requesting file transfer or remote execution communication with a remote machine (Phase A). Several database support files are read to determine if the remote machine is accessible by the local machine and the priority of the user's request, as compared to other users' requests. This phase ends by queuing the user's request in a spool area on the local machine and triggering the next phase.
2. The *uucico* routine is triggered automatically (Phase B). It reads several BNU database support files to determine when the remote machine can be reached, how to establish the link to the remote machine, how to handle data flow between the local and remote machines, and if the maximum number of requests for communication to the remote machine has been reached.
3. The *uucico* routine on the remote machine is triggered automatically when a call for communication is received from the local machine (Phase C). In this phase, the remote *uucico* routine reads BNU database support files on its machine to determine if the calling machine is allowed access, and what action to take if the calling machine is not allowed access (Phase C).
4. Requests initiated on the local machine may contain commands to be executed on the remote machine (Phase D). When these commands arrive on the remote machine, they are stored. The *uuxqt* routine on the remote machine runs these commands on the remote machine during this phase.

The following diagram illustrates how the Basic Network Utilities work:

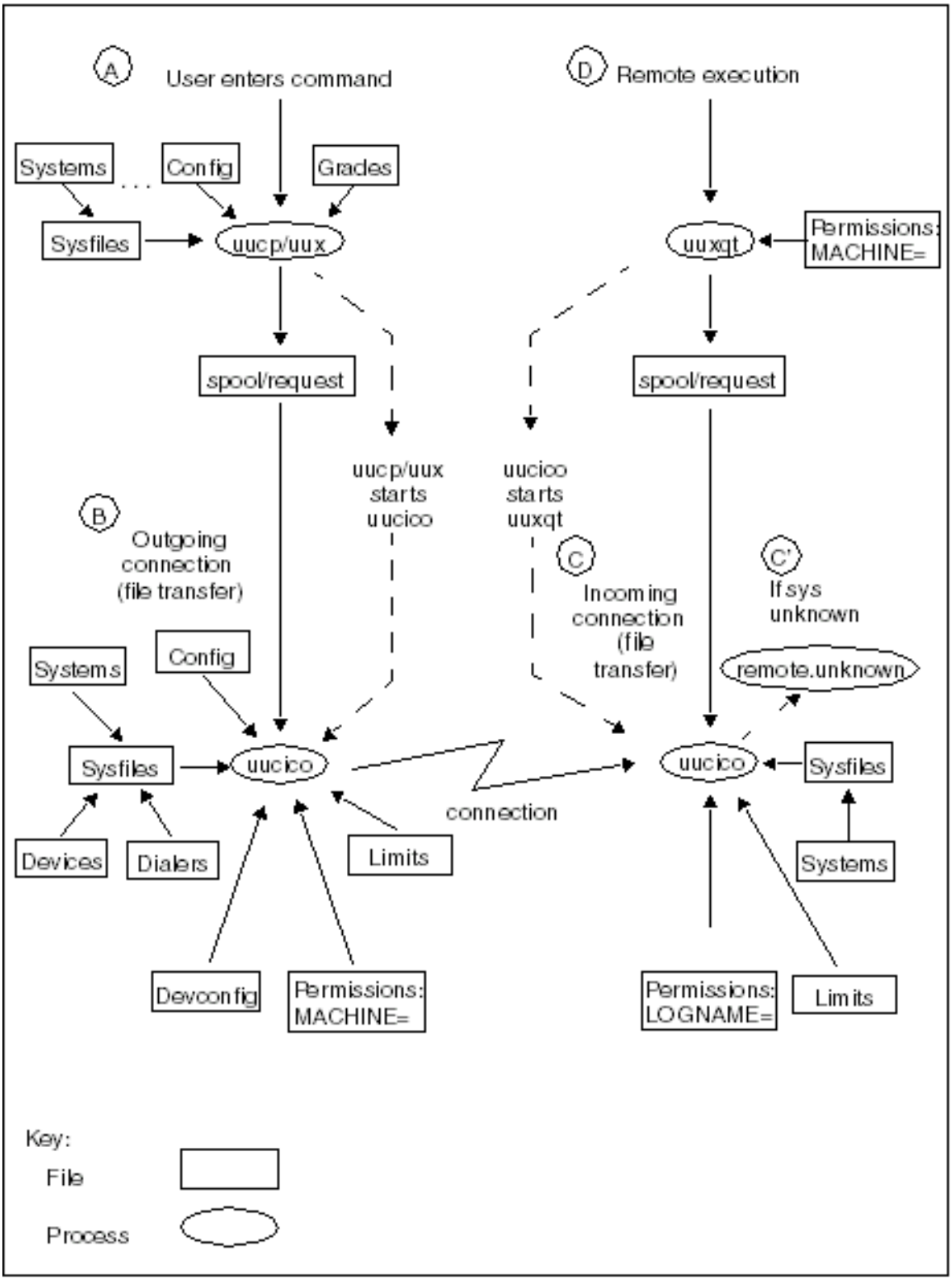


Figure 16: How the Basic Network Utilities work

9.2 BNU components

9.2.1 Commands

You will find the user commands for the Basic Networking Utilities in the */usr/bin* directory. No special permission is needed to use these programs. The following user commands are described in the manual's reference section:

ct	spawn login to a remote terminal
cu	call another UNIX system
uucp	UNIX-to-UNIX system copy
uudecode	decode an ASCII file after transmission
uuencode	encode a binary file before transmission
uuglist	print the list of service grades that are available on this system
uulog	query a log file of *uucp*
uuname	list the names of systems known to *uucp*
uupick	UNIX-to-UNIX system file copy
uustat	*uucp* status inquiry and job control
uuto	UNIX-to-UNIX system file copy
uux	UNIX-to-UNIX command execution

You will find the administrative commands in */usr/lib/uucp*:

uucheck(1M)	check the *uucp* directories and permission file
uucleanup(1M)	*uucp* spool directory clean-up
Uutry(1M)	try to contact remote system with debugging on

You should use the uucp login ID only when you administer BNU because it owns the Basic Networking and spooled data files. The home directory of the uucp login ID is */usr/lib/uucp*. The other Basic Networking login ID is nuucp, used by remote machines to access your machine.

9.2.2 Daemons

A daemon is a routine that runs as a background process and performs a system-wide public function. These daemons handle file transfers and command executions. The following three routines located under */usr/lib/uucp* and started as BNU daemons are described in the "Networking Reference Manual":

uucico(1M)
: file transport program for the *uucp* system

uusched(1M)
: the scheduler for the *uucp* file transport program

uuxqt(1M)
: execute remote command requests

9.2.3 Shell scripts

The BNU contain the following four shell scripts in the */usr/lib/uucp* directory:

uudemon.poll
uudemon.hour
uudemon.admin
uudemon.cleanup

These scripts will poll remote machines, reschedule transmissions, and clean up old log files and unsuccessful transmissions. They should be used regularly to keep your basic networking running smoothly.

Normally, they are run automatically with *cron(1M)* although they can also be run manually (see the section entitled BNU maintenance).

9.2.4 Files

There are three types of BNU support files:

Database files
: These files are responsible for much of the actual networking activity associated with your BNU package. They are located in */etc/uucp*. In general, they determine: what machines your machine will communicate with, the devices over which the communication will take place, and the protocols for communicating with remote machines. The most important files required for a connection description are systems, devices and dialers. See the section entitled Database files.

Administrative files
: Administrative files are created by network processes in spool directories to hold temporary data or store information about remote transfers or programs. See the section entitled Administrative files.

Log files
: Log files keep track of overall statistics of your computer network, particularly in the areas of security and accounting. See the section entitled Log files.

9.3 Basic procedures

The steps described below are carried out in BNU administration.

9.3.1 Setting up the BNU

The BNU are set up using a text editor to change and extend the basic files supplied with the system. The section entitled Database files describes the structure of the files and explains the possible entries.

When setting up the BNU, it is also possible to specify several uucp logins, so that incoming *uucico* requests from different remote systems can be handled differently.

As a rule, the */etc/passwd* file contains the following uucp logins:

```
uucp:x:5:5:0000-uucp(0000):/usr/lib/uucp
nuucp:x:10:10:0000-uucp(0000):/var/spool/uucppublic: \
/usr/lib/uucp/uucico
```

The nuucp entry indicates that a login request is answered via nuucp using */usr/lib/uucp/uucico*. The home directory is */var/spool/uucppublic*. The entry x indicates that an encrypted password is stored in */etc/shadow*.

9.3.2 BNU maintenance

BNU maintenance involves automatic and manual maintenance of your most important files for using the network, to ensure that the files are not too large and do not take up too much disk space. These maintenance procedures also ensure that BNU operation is error-free.

9.3.2.1 Automatic maintenance

The BNU is delivered with the appropriate entries for the shell scripts in the */var/spool/cron/crontabs/root* file. These entries enable some of the administration tasks for the BNU to be executed automatically with *cron(1M)* when using the following shell scripts:

- *uudemon.poll*

  The *uudemon.poll* shell script executes the following functions:

  - It reads the */etc/uucp/Poll* file.
  - If a machine in the */etc/uucp/Poll* file is reserved for cyclical polling, a request (*C.sysnxxxx*) is created in the */var/spool/uucp/node_name* directory, where *node_name* is the name of the machine that is polled.

  In the default setting, this script is executed twice every hour, immediately before *uudemon.hour*, so that the job files are there when *uudemon.hour* is called. The default entry for *uudemon.poll* is:

  ```
  1,30 * * * * /usr/lib/uucp/uudemon.poll > /dev/null
  ```

- *uudemon.hour*

  The *uudemon.hour* shell script executes the following functions:

- It calls the program *uusched* to search the spool directory for job files (*C.sysnxxxx*) that have not been processed. It then puts these files in order for transferring to a remote machine.
- It calls the daemon *uuxqt* to search the spool directory for executable files (*X.sysnxxx*) which were transferred to your machine but were not executed at the time of transfer.

In the default setting, this script is executed twice an hour. If you expect a high rate of errors, the script should be executed more often. The default entry for *uudemon.hour* is:

```
41,11 * * * * /usr/lib/uucp/uudemon.hour > /dev/null
```

- *uudemon.admin*

The *uudemon.admin* shell script performs the following functions:

- It starts the *uustat* command with the options *–p* and *–q*. The option *-q* reports on the status of the job files (*C.sysnxxxx*), the data files (*D.sysstmxxxxyyy*), and the executable files (*X.sysnxxx*) in the queue. The *-p* option provides process information for network processes that are executed in the lock files under */var/spool/locks*.
- It sends the status information via *mail(1)* to the *uucp* system administration login.

There is no default entry for *uudemon.admin*, but the following entry is recommended:

```
48 8,12,16 * * * /bin/su uucp –c \□
"/etc/uucp/uudemon.admin"> /dev/null
```

- *uudemon.cleanup*

The shell script *uudemon.cleanup* executes the following functions:

- It removes log files for individual machines from the directory */var/spool/uucp/.Log*, combines them and saves them, together with other old log information, in the directory */var/spool/uucp/.Old*. If the log files are too big, you may have to increase the value of the parameter *ulimit*.
- It deletes job files (*C.sysnxxx*) and data files (*D.systmxxxxyyy*) that are seven days old or more, and data files (*X.sysnxxx*) that are two days old or more, from the spool directory.
- It returns to the sender electronic mail that could not be sent.
- It sends a summary of the status information collected during the current day, to the *uucp* login for system administration.

There is no default entry for *uudemon.cleanup*, but the following entry is recommended:

```
45 23 * * * ulimit 5000; /usr/bin/su uucp -c \□
"/usr/lib/uucp/uudemon.cleanup" > /dev/null 2>&1
```

uudemon.cleanup is a shell script that should be called each day using *cron* in order to clean up the *uucp* spool directory and to free it from the log files that currently exist for *uucp*. *uudemon.cleanup* uses two techniques to delete log files:

1. The "multi-day" technique (multi)

   The logs for three days are saved to files called *Old-Log-1*, *Old-Log-2* and *Old-Log-3*. When *uudemon.cleanup* is implemented, *Old-Log-2* is renamed as *Old-Log-3*, *Old-Log-1* is renamed as *Old-Log-2*, and the current file is renamed as *Old-Log-1*. These files are saved to the directory */var/spool/uucp/.Old*.

2. The "single-day" technique (single)

   The current log is moved to */var/spool/uucp/.Old*. This means that the log is only kept for one day (provided *uudemon.cleanup* runs each day).

Although *uudemon.cleanup* is responsible for deleting old log entries, the system administrator must monitor the size of the log files. The method that *uudemon.cleanup* uses to remove old log entries differs, depending on the type of log file. The following section explains how the log files are cleaned.

File use	File name	Technique
Command	*/var/uucp/.Admin/command*	single

History	*/var/uucp/.Log/uucp/system*	multi
"	*/var/uucp/.Log/uucico/system*	multi
"	*/var/uucp/.Log/uux/system*	multi
"	*/var/uucp/.Log/uuxqt/system*	multi
Foreign	*/var/uucp/.Admin/Foreign*	single
Error	*/var/uucp/.Admin/errors*	single
Transfer	*/var/uucp/.Admin/xferstats*	single
Accounting	*/var/uucp/.Admin/account*	multi
Security	*/var/uucp/.Admin/security*	multi
Performance	*/var/uucp/.Admin/perflog*	single
Debugging	*/var/uucp/.Admin/audit*	single

Table 4: Overview of BNU log files

9.3.2.2 Manual maintenance

Some files increase in size indirectly as a result of *uucp* and other BNU activities. Both the following files need to be monitored and deleted if they get too big:

/usr/adm/sulog

This file keeps a record of all the commands that require root rights. Because the *uudemon* entries in the */usr/cron/root* file use the *su* command, the *sulog* file increases over time. If you have no further need for this file, you should delete it if it gets too big.

/usr/lib/cron/log

This file provides a record of the *cron* activities. Because the file is constantly being used and so increases in size, it is automatically shortened when the system switches to multi-user state.

9.3.3 BNU debugging

Debugging procedures help you to identify and remove the most frequently-occurring problems in the basic network functions.

9.3.3.1 Checking basic information

The following commands provide basic information on the network:

uuname

This command lists those machines your machine can contact.

uulog This command displays the contents of the log directories for particular machines.

uucheck –v

This command checks the presence of files and directories needed by *uucp*. It also checks the *Permissions* file and provides information on the permissions you have set up.

9.3.3.2 Checking for faulty ACUs/modems

You can check if the automatic call units or modems are not working properly in several ways.

- *uustat –q* gives you counts and reasons for contact failure.
- *cu –d –l line* gives you diagnostic information about connection establishment. If the communications line is connected to an autodialer, you must add a telephone number at the end of the command line you execute. Otherwise, *line* must be defined as direct in the *Devices* file.

9.3.3.3 Checking the Systems file

If you are having trouble contacting a particular machine, it is likely that the *Systems* file contains incorrect entries. Check whether the following entries match current conditions:

- Phone number
- Login
- Password

9.3.3.4 Errors during transmission

If you are unable to contact a machine, you can check out communications to that machine with *Uutry* and *uucp*.

1. Try to make contact, run

   $ /usr/lib/uucp/Uutry -r *machine*

   where *machine* is replaced with the node name of the machine you are having problems contacting. This command will

   - start the daemon *uucico* with debugging. You will get more debugging information if you are root.
   - direct the debugging output to */tmp/machine*.
   - print the debugging output to your terminal, using *tail –f*.

   You can copy the output from */tmp/machine* if you want to save it.

2. If *Uutry* doesn't isolate the problem, try to queue a job by running

   $ uucp -r *file machine*!*/directory/file*

   where *file* is replaced by the file you want to transfer, *machine* is replaced by the machine you want to copy to, and *directory/file* is where the file will be placed on the other machine. The *-r* option will queue a job but not start the transfer.

   Start *Uutry* again. If you still cannot solve the problem, you may need to call the SNI service. Save the debugging output; it will help diagnose the problem.

9.4 Database files

The BNU database files are in the */etc/uucp* directory. These files can be modified or extended using any text editor. In the main, of the database files described, only the *Systems*, *Devices* and *Dialers* files are used. However, highly complex connection definitions can be created by means of the other files. The following database files are available:

Config
: Contains a list of variable parameters within BNU. The administrator can set these parameters to configure the network.

Devconfig
: This file is used to configure your network connections.

Devices
: Contains information on interfaces to be used and the parameters for their initialization (e.g. the baud rate).

Dialcodes
: Contains dial-code abbreviations that may be used in the *phone* field of *Systems* file entries.

Dialers
: Contains information required for the initialization of automatic call units (modems) and direct connections.

Grades
: This file is used to define the job grades, and the permissions associated with each job grade, that users may specify to queue jobs to a remote machine.

Limits
: This file defines the maximum number of simultaneous *uucico*, *uuxqt*, and *uusched* calls permitted on your machine.

Permissions
: This file defines the access permissions granted to machines when they attempt to transfer files or execute remote commands on your machine.

Poll
: This file defines machines that are to be polled by your system and when they are polled.

Sysfiles
: This file is used to create groups of *Systems*, *Devices*, and *Dialers* files that can be used by *uucico* and *cu*.

Systems
: This file contains information on which remote machines are accessible and how a connection to them can be established.

remote.unknown
: This file is also a database file, but it does not contain any information on connection establishment. There is a brief description of it at the end of this section.

For some applications (e.g. *cu*, *PPP*), only the *Systems*, *Devices* and *Dialers* files are required. The following diagram shows an example of how these files are linked:

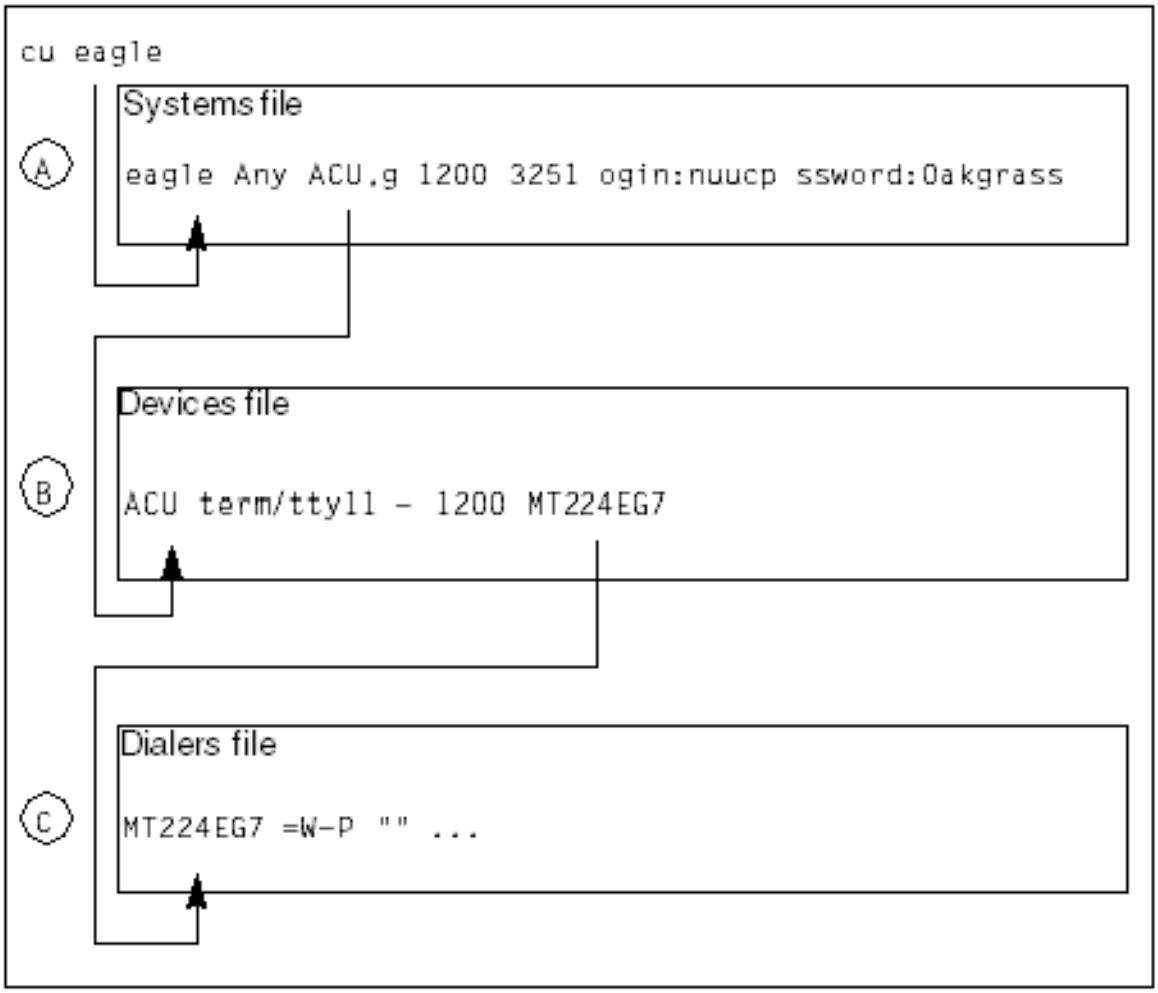


Figure 17: Links between the Systems, Devices and Dialers files

The machine name passed as an argument with the *cu* command is a keyword for the first field of an entry in the *Systems* file (A). The class specified in the third field must agree with the first field of an entry in the *Devices* file (B). The *dialer* field (fifth field of the entry in the *Devices* file) must correspond to the first field of an entry in the *Dialers* file (modem type).

The various files are described below. To facilitate understanding, the *Systems*, *Devices* and *Dialers* files are described first. The other files follow in alphabetical order.

9.4.1 The Systems file

The *Systems* file (*/etc/uucp/Systems*) contains the information needed by the *uucico* daemon to establish a communication link to a remote machine. Each entry in the file represents a machine that can be called by your machine. In addition, the Basic Networking Utilities are configured to prevent any machine that is not entered in this file from logging into your machine (refer to the section entitled The remote.unknown file for a description of the *remote.unknown* file). More than one entry may be present for a particular machine. The additional entries represent alternative communication paths that will be tried in sequential order.

Using the *Sysfiles* file, you can define several files to be used as *Systems* files. See the description of the *Sysfiles* file for details (the The Sysfiles file). Each entry in the *Systems* file has the following format:

System-Name Time Type[g,G,e] *Class Phone Login*

These fields are defined as:

System-Name
: This field contains the name of the remote machine, by means of which it can be accessed from your machine.

Time This field contains a string that specifies the day-of-week and time-of-day when the remote machine can be called. The format of the *Time* field is:
daytime[;retry]

The day portion may be a list containing some of the following:

Su Mo Tu We Th Fr Sa
: for individual days

Wk for any week-day (Mo Tu We Th Fr)

Any for any day

Never
: for a passive arrangement with the remote machine. If the *Time* field is Never, your machine will never initiate a call to the remote machine. The call must be initiated by the remote machine. In other words, your machine is in a passive mode with respect to the remote machine (see the section entitled The Permissions file).

The *time* portion should be a range of times specified in 24-hour notation, for example 0830-1230 for "8:30 a.m. to 12:30 p.m." If no *time* portion is specified, any time of day is assumed to be allowed for the call. A time range that spans 0000 is permitted. For example, 0800-0600 means all times are allowed other than times between 6 a.m. and 8 a.m. An optional subfield, *retry*, is available to specify the minimum time (in minutes) before a retry, following a failed attempt. The default wait is 60 minutes. The subfield separator is a semicolon (;). For example, Any;9 is interpreted as call any time, but wait at least 9 minutes before retrying after a failure occurs. Here is an example:

Wk1700-0800,Sa,Su

This example allows calls from 5:00 p.m. to 8:00 am, Monday through Friday, and calls any time Saturday and Sunday. The example would be an effective way to call only when telephone rates are low, if immediate transfer is not critical.

Type This field contains the device type that should be used to establish the communication link to the remote machine. The keyword used in this field is matched against the first field of *Devices* file entries as shown below:
Systems: eagle Any ACU,g D1200 3251 ogin: nuucp \

ssword: Oakgrass

Devices: ACU tty11 - D1200 MT224EG7

You can define the protocol used to contact the system by adding it on to the *Type* field. The example above shows how to attach the protocol g to the device type ACU. See the section entitled Protocols for details.

The TlisTcp entry is required if your machine is connected to a TCP/IP LAN that you want to access

using Basic Networking Utilities.

Class This field specifies the transfer speed of the device used in establishing the communication link. It may contain a letter before the speed specification (for example, C1200, D1200) to differentiate between classes of dialers (refer to the discussion of the *Class* field in the section entitled The Devices file). Some devices can be used at any speed, so the keyword Any may be used. This field must match the *Class* field in the associated *Devices* file entry as shown below:
Systems: eagle Any ACU D1200 NY3251 ogin: nuucp \

ssword: Oakgrass

Devices: ACU tty11 - D1200 MT224EG7

If no information is required for this field, use a "–" as a place holder for the field. This is the case when your machine is connected to a TCP/IP LAN that you want to access using Basic Networking Utilities, for instance.

Phone

This field allows you to specify the telephone number (token) of the remote machine for automatic dialers (LAN switches). The telephone number is made up of an optional alphabetic abbreviation (corresponding to the dialing code) and a numeric part. If an abbreviation is used, it must be one that is listed in the *Dialcodes* file. For example:
Systems: eagle Any ACU D1200 NY3251 ogin: nuucp \

ssword: Oakgrass

Dialcodes: NY 9=1212555

In this string, an equal sign (=) tells the ACU to wait for a secondary dial tone before dialing the remaining digits. A dash (–) in the string instructs the ACU to pause 4 seconds before dialing the next digit.

If your machine is connected to a TCP/IP LAN that you want to access using Basic Networking Utilities, this field must contain an entry in a specific format understood by the NLS listener serving the uucp connection via the network. The entry always begins with \x, followed by 32 hexadecimal characters, the first eight of which are always 0002021C. The next eight characters identify the Internet address of the remote system. The Internet address can be found in the */etc/inet/hosts* file but it must be converted to hexadecimal code. For instance, the Internet address 144.145.2.2 becomes the hexadecimal value 90910202. The remaining 16 positions in the entry contain zeroes. The associated entry in the *Devices* file should have a \D at the end of the entry to ensure that this field is not translated using the *Dialcodes* file.

Login This field contains login information given as a series of fields and subfields of the format:

expect send

where *expect* is the string that is received and *send* is the string that is sent when the *expect* string is received.

The *expect* field may be made up of subfields of the form:
expect[*-send-expect*]...

where the *send* is sent if the prior *expect* is not successfully read and the *expect* following the *send* is the next expected string. For example, with login––login, *uucp* will expect login. If *uucp* gets login, it will go on to the next field. If it does not get login, it will send nothing followed by a new line, then look for login again. If no characters are initially expected from the remote machine, the characters "" (null string) should be used in the first *expect* field. Note that all *send* fields will be sent followed by a new-line unless the *send* string is terminated with a \c.

Here is an example of a *Systems* file entry that uses an expect-send string:
owl Any ACU 1200 Chicago6013 "" \r ogin:-BREAK-ogin: \

uucpx word: xyzzy

This example says don't wait, just send a carriage return and wait for ogin: (for Login:). If you don't get

ogin:, send a BREAK. When you do get ogin:, send the login name uucpx, then when you get word: (for Password:), send the password xyzzy.

The following control characters can be helpful at login:

\N Send or expect a null character (ASCII NUL)

\b Send or expect a backspace character

\c If at the end of a string, suppress the new-line that is normally sent.

\d Delay two seconds before sending or reading more characters.

\p Pause for approximately 1/4 to 1/2 second.

\E Start echo checking. (From this point on, whenever a character is transmitted, it will wait for the echo to be received before doing anything else.)

\e Echo check off.

\M Turn on CLOCAL flag.

\m Turn off CLOCAL flag.

\n Send a new-line character.

\r Send or expect a carriage-return.

\s Send or expect a space character.

\t Send or expect a tab character.

\\ Send or expect a \ character.

EOT Send or expect EOT new-line twice.

BREAK
Send or expect a break character.

\K Same as BREAK.

ddd Collapse the octal digits (*ddd*) into a single character.

9.4.2 The Devices file

The *Devices* file (*/etc/uucp/Devices*) contains information for all the devices that may be used to establish a link to a remote machine. Provisions are made for several types of devices, such as automatic call units, direct links, and network connections.

This file works closely with the *Dialers*, *Systems*, and *Dialcodes* files. Before you make changes in any of these files, you should be familiar with them all. A change to an entry in one file may require a change to a related entry in another file.

Each entry in the *Devices* file has the following format:

Type Line[,*M*] *Line2 Class Dialer-Token-Pairs*

These fields are defined as:

Type This field may contain one of two keywords (Direct or ACU), the name of a local area network, switch, or system.

Direct
This keyword indicates a direct link (null modem cable) to another machine or a switch.

ACU This keyword specifies that the link to a remote machine is made through an automatic call unit (automatic-dial modem). This modem may be connected either directly to your machine or indirectly through a switch (TACSI, TCA, TACLAN).

LAN_Switch
This is the name of the LAN or switch. For instance, TCP could be the name for a TCP/IP network. Develcon could be the name for a Develcon switch connection.

Sys-Name

This value specifies a direct link to a particular machine. (*Sys-Name* is replaced by the name of the machine.) This naming scheme is used to convey the fact that the line associated with this *Devices* entry is for a particular machine in the *Systems* file.

The keyword used in the *Type* field is matched against the third field of *Systems* file entries as shown below:

Devices: ACU term/11 - 1200 penril

Systems: eagle Any ACU 1200 3251 ogin: nuucp \□
ssword: Oakgrass

You can designate a protocol to use for a device within this field. See the "Protocols" section at the end of the description of this file.

Line Identifies the device node (port) via which the connection is to be established. The optional parameter M, which can be appended to the device node, specifies that the device can be opened without waiting for the carrier (nonblocked open). The */dev* path before the device name can be omitted.

Line2 If the keyword ACU was used in the *Type* field and the ACU is an 801 type dialer, *Line2* would contain the device name of the 801 dialer. (801 type ACUs do not contain a modem. Therefore, a separate modem is required and is connected to a different line, defined in the *Line* field.) This means that one line would be allocated to the modem and another to the dialer. Since non-801 dialers will not normally use this configuration, they ignore the *Line2* field, but it must still contain a hyphen (–) as a placeholder.

Class If the keyword ACU or Direct is used in the *Type* field, *Class* specifies the speed of the device. However, it may contain a letter and a speed (for example, C1200, D1200) to differentiate between classes of dialers (Centrex or Dimension PBX). This is necessary because many larger offices may have more than one type of telephone network: one network may be dedicated to serving only internal office communications while another handles the external communications. In such a case, it becomes necessary to distinguish which line(s) should be used for internal communications and which should be used for external communications. The keyword used in the *Class* field of the *Devices* file is matched against the fourth field of *Systems* file entries as shown below:

Devices: ACU term/11 - 1200 penril

Systems: eagle Any ACU 1200 3251 ogin: nuucp \

ssword: Oakgrass

Some devices can be used at any speed, so the keyword Any may be used in the *Class* field. If Any is used, the line will match any speed requested in a *Systems* file entry. If this field is Any and the *Systems* file *Class* field is Any, the speed defaults to 1200 bps.

Dialer-Token-Pairs:

This field (which is also referred to as DTP) contains pairs of dialers and tokens. The *dialer* portion can contain the name of an automatic-dial modem or a LAN switch, or it may be direct or uudirect for a direct link device. You can have any number of Dialer-Token-Pairs. The *token* portion may be supplied immediately following the *dialer* portion, or, if not present, it will be taken from a related entry in the *Systems* file.

This field has the format:
dialer token [dialer token]

where the last pair may or may not be present, depending on the associated device (dialer). In most cases, the last pair contains only a *dialer* portion and the *token* portion is retrieved from the *Phone* field of the *Systems* file entry.

A valid entry in the *dialer* portion may be defined in the *Dialers* file or may be one of several special dialer types. The special dialer types are compiled into the software and are therefore available without having entries in the *Dialers* file.

801 Bell 801 auto dialer

TLI Transport Level Interface Network (without STREAMS)

TLIS Transport Level Interface Network (with STREAMS module tirdwr)

TCP TCP/IP Network (using sockets)

The *Dialer-Token-Pairs* (DTP) field may be structured differently, depending on the device associated with the entry:

- If an automatic-dial modem is connected directly to a port on your machine, the DTP field of the associated *Devices* file entry will only have one entry. This entry would normally be the name of the modem. This name is used to match the particular *Devices* file entry with an entry in the *Dialers* file. Therefore, the *dialer* field must match the first field of a *Dialers* file entry as shown below:

  Devices: ACU term/11 - 1200 att2212c
  Dialers: att2212c =+-, "" atzod,o12=y,o4=n\r\c \
  006 atT\T\r\c ed

  Notice that only the *dialer* portion (att2212c) is present in the DTP field of the *Devices* file entry. This means that the *token* to be passed on to the dialer is taken from the *Phone* field of a *Systems* file entry. (\T is implied; backslash sequences are described below.)
- If a direct link is established to a particular machine, the DTP field of the associated entry would contain the keyword direct or uudirect. This is true for both types of direct link entries, Direct and *System-Name* (refer to discussion on the *Type* field).
- If you wish to communicate with a machine that is on the same local network switch as your own, your machine must first access the switch and the switch can make the connection to the other machine. In this type of entry, there is only one pair. The *dialer* portion is used to match a Dialers file entry as shown below:

  Devices:develcon term/13 - 1200 develcon
  Dialers:develcon "" "" \pr\ps\c est:\007 \E\D\e \007

  As shown, the *token* portion is left blank. This indicates that it is retrieved from the *Systems* file. The entry in the *Systems* file for this particular machine will contain *token* in the *Phone* field, which is normally reserved for the telephone number of the machine (refer to *Systems* file, *Phone* field). This type of DTP contains an escape character (\D), which ensures that the contents of the *Phone* field will not be interpreted as a valid entry in the *Dialcodes* file.

- If an automatic dialing modem is connected to a switch, your machine must first access the switch and the switch will make the connection to the automatic dialing modem. This type of entry requires two *dialer-token-pairs*. The *dialer* portion of each pair (fifth and seventh fields of entry) will be used to match entries in the *Dialers* file as shown below:

  Devices: ACU term/14 - 1200 develcon dial att2212c
  Dialers: develcon "" "" \pr\ps\c est:\007 \E\D\e \007 □
  Dialers: att2212c =+-, "" atzod,o12=y,o4=n\r\c \006
  atT\T\r\c ed

  In the first pair, develcon is the dialer and dial is the token that is passed to the Develcon switch to tell it which device (auto dial modem) to connect to your machine. This token would be unique for each LAN switch since each switch may be set up differently. Once the modem has been connected, the second pair is accessed, where att2212c is the dialer and the token is retrieved from the *Systems* file.
- If the dialer type TLIS is chosen, ensure an entry exists in the *Devconfig* file for this device so that the appropriate modules will be pushed onto the stream.

Configuring a TCP/IP LAN switch, you must activate the □
TlisTcp tcp - - TLIS \D entry in the *Devices* file (remove comment).

There are two escape characters that may appear in a DTP field:

\T Specifies that the *Phone* (*token*) field should be translated using the *Dialcodes* file. This escape character is normally placed in the *Dialers* file for each caller script associated with an automatic dial modem. Therefore, the translation will not take place until the caller script is accessed.

\D Indicates that the *Phone* (*token*) field should not be translated using the *Dialcodes* file. If the dialer is an internal dialer, \T is the default. Otherwise, \D is the default. A \D is also used in the *Dialers* file with entries associated with network switches (develcon and dicom).

Protocols

You can choose the protocol to use with each device. Usually, it is not needed since you can use the default. If you do specify the protocol, you must do so in the form *Type,Protocol*[(*parameters*)]. Available protocols are:

g This is a generic packet protocol. It provides error detection and retransmission intended for use over potentially noisy lines. By its nature, it is relatively slow. Two parameters characterize the g protocol, *windows* and *packetsize*. *windows* indicates the number of packets which may be transmitted without waiting for an acknowledgment from the remote machine. *packetsize* indicates the number of data bytes in each packet. *windows* value is set at 7, and *packetsize* is set at 64 bytes.

G This protocol is identical to the g protocol in that it provides the same error detection and retransmission. However, in addition, the G protocol allows the number of windows and the packet size to be varied to match the characteristics of the transmission medium. When configured correctly, the transmission speed is higher with this protocol than with the g protocol. *windows* may range from 1 to 7, and *packetsize* may range from 32 to 4096 bytes, in powers of 2 (that is, 32, 64, 128, 256, 512, 1024, 2048, 4096).

e This protocol assumes error free transmission and performs no error checking or retransmission. Therefore it is the fastest of these protocols. It should be used for reliable local area networks. There are no parameters to be tuned within the e protocol.

Here is an example that uses the e protocol over a TCP Network and Transport Level Interface with streams. If the e protocol is not available, g will be used.

TlisTcp, eg tcp - - TLIS \D

Here is an example that uses the G protocol on a modem. The number of windows is set to 7, and the packetsize is 512 bytes. If the G protocol is unavailable, the standard g protocol will be used.

ACU,G(7,512)g term/11 - 9600 att2296a

Presumably, seven windows with a packet size of 512 bytes will provide optimum throughput for the specified device.

For incoming connections, the preferred protocol priority and parameters may be specified in the *Config* file using the *Protocol* parameter.

9.4.3 The Dialers file

The *Dialers* file (*/etc/uucp/Dialers*) specifies the initialization of the dialer (e.g. modem). It contains type-specific data such as *hayes* commands for modems.

As shown in the above examples, the fifth and subsequent odd numbered fields in the *Devices* file contain a reference to the *Dialers* file or an internal list of special dialer types (801, TLI, or TLIS). If the match succeeds, the *Dialers* entry is user directly or interpreted to initialization connection establishment via the dialer.

Each entry in the *Dialers* file has the following format:

dialer substitutions expect-send ...

The *dialer* field matches the fifth and additional odd numbered fields in the Devices file. The *substitutions* field is a translate string: the first of each pair of characters is mapped to the second character in the pair. This is usually used to translate "=" and "–" into whatever the dialer requires for "wait for dialtone" and "pause."

The *expect-send* field contains strings, for example *hayes* commands for modems. The *expect* strings are waited for, and, in the event of success, the *send* string is sent.

When you use device nodes which support modems, opening the device file neither waits for a carrier nor does it set the flag CLOCAL. The character strings *expect-send* must therefore begin with the escape character \M (set CLOCAL) and end with \m (clear CLOCAL).

The example below shows some strings in the *Dialers* file that are shipped with the BNU. The entries supplied in the */etc/uucp/Dialers* file contain exclusively *hayes* commands for some of the modem types used by Siemens. If other modem types are used, these entries must be derived from their descriptions.

modem-type: MT224EG7 v22bis =,-, "" \M\dAT&FX3\r\c OK\r
AT&E2&E5&E11&S0=1&W0\r\c OK\r AT$SB19200\r\c OK\r \EATDP\T\r\c CONNECT \r\m
modem-type: MT1432BG or MT932BG v32bis =,-, "" \M\dAT&FX3\r\c OK\r AT&E2&E5&E11&S0=1&W0\r\c
OK\r AT$SB19200\r\c OK\r \EATDP\T\r\c CONNECT \r\m
modem-type: MT1432BG or MT932BG (flow control + V42bis mode OFF)
#v32bis =,-, "" \M\dAT&FX3\r\c OK\r AT&E0&E3&E6&E8&E10&E12&S0=1&W0\r\c OK\r AT$SB19200\r\c
OK\r \EATDP\T\r\c C
modem-type: MT1932zdx 1932zdx =,-, "" \M\dAT&FX3\r\c OK\r AT&E2&E5&E11&S0=1&W0\r\c OK\r
AT$SB19200\r\c OK\r \EATDP\T\r\c CONNECT \r\m
!!!! USE this modem configuration for the "passive" side ONLY; modem-type: v22/v32bis
modem configuration;
modem-type: v22/v32bis mconf =,-, "" \M\dAT&FX3\r\c OK\r AT&E2&E5&E11&S0=1\r\c OK\r
AT$SB19200Q1&W0\r\c ""\r \K

The meaning of some of the escape characters (those beginning with \) used in the *Dialers* file are listed below:

\p Pause (approximately 1/4 to 1/2 second)

\d Delay (approximately 2 seconds)

\D Telephone number or token without *Dialcodes* translation

\T Telephone number or token with *Dialcodes* translation

\K Insert a BREAK

\E Enable echo checking (for slow devices)

\e Disable echo checking

\r Carriage return

\c No new-line or carriage return

\M Turn on CLOCAL interface parameter

\m Turn off CLOCAL parameter

\n Send new-line

nnn Send character represented by octal number *nnn*

Additional escape characters that may be used are listed in the section entitled The Systems file.

The entry v22bis in the file *Dialers* is carried out as follows:

First of all, the argument corresponding to the telephone number is converted, by replacing each "=" and each "-" by a "," (pause). The protocol that is specified by the remainder of the line operates as follows:

"" Don't wait. (carry on working up to the string expect-send.)

\M\dAT&FX3\r\c
Switch on the interface parameter CLOCAL (switches the modem into command mode), load the factory settings, set up the result code.

OK\r Wait for "OK".

AT&E2&E5&E11&S0=1&W0\r\c
Select the reliability mode and the flow-control mechanism. Specify the number of dial tones before the modem answers (S0=1) and store the current parameters.

OK\r Wait for "OK".

AT$SB19200\r\c
Set up the transfer rate of the serial interface (19200).

OK\r Wait for "OK".

\EATDP\T\r\c CONNECT
Activate the echo check and transfer the telephone number (\T) using pulse dialling (ATDP). Then wait for "CONNECT".

\r\m Switch off the interface parameter CLOCAL (exit from command mode)

9.4.4 The Config file

The */etc/uucp/Config* file allows the administrator to modify BNU parameters in the BNU manually. Each entry in the *Config* file has the following format:

parameter=value

Where *parameter* is one of the configurable parameters and *value* is the value to be assigned to that parameter. See the *Config* file provided with your system for a complete list of configurable parameter names.

The following *Config* file entry sets the default protocol ordering to Gge and changes the G protocol defaults to 7 windows and 512 byte packets.

Protocol=G(7,512)ge

9.4.5 The Devconfig file

The */etc/uucp/Devconfig* file is used if your machine is communicating over a network. *Devconfig* entries define the STREAMS module, which is used for a particular device.

An entry in the *Devconfig* file has the following format:

service=*command*[:*command*] device=*type* \ □
push=*streams_module*[:*streams_module* ...]

command
: stands for *cu* or *uucico*. It is also possible to specify both commands, separated by a colon.

type
: stands for the name of the network and it must be the same as the first entry in the file *Devices* and the third entry in the *Systems* file.

streams_module
: stands for the name of a STREAMS module that is engaged in the STREAM before the transfer starts. You can also specify several modules, separated by a colon. In this case, the STREAMS modules are engaged in the specified order.

Different modules and devices can be defined for the services *cu* and *uucp*.

The following example shows entries for a TCP/IP network:

```
service=cu      device=TlisTcp  push=ldterm:tirdwr
service=uucico  device=TlisTcp  push=tirdwr
```

9.4.6 The Dialcodes file

The *Dialcodes* file (*/etc/uucp/Dialcodes*) contains the dial-code abbreviations that can be used in the *Phone* field of the *Systems* file. Each entry has the format:

abb dial-seq

where *abb* is the abbreviation used in the *Systems* file *Phone* field and *dial-seq* is the dial sequence that is passed to the dialer when that particular *Systems* file entry is accessed.

The entry

jt 9=555-

would correspond to a *Phone* field in the *Systems* file such as jt7867. If jt7867 is entered there, the sequence 9=555-7867 is sent to the dialer if the escape character \T is set for the DTP (see *Dialer-Token-Pairs* in the Section "The Devices file").

9.4.7 The Grades file

The *Grades* file (*/etc/uucp/Grades*) contains the definitions for the job grades that may be used to queue jobs to a remote machine. It also contains the permissions for each job grade. Each entry in this file represents a definition of an administrator defined job grade that allows users to queue jobs.

Each entry in the *Grades* file has the following format:

User-job-grade System-job-grade Job-size Permit-type ID-list

Each entry in this file contains fields that are separated by white space. The last field in the entry is made up of sub-fields that are also white-space separated. If an entry takes up more than the physical line, a backslash (\) is used to indicate that the entry continues on the following line. Comment lines begin with a number sign (#) and occupy the entire line. Blank lines are always ignored. Here is a description of each field:

User-job-grade
: This field contains an administrative defined user job grade name of up to 64 characters.

System-job-grade
: This field contains a one-character job grade to which *User-job-grade* will be mapped. The valid list of characters is A–Z, a–z, with A having the highest priority and z the lowest.

Job-size
: This field specifies the maximum job size that can be entered in the queue. *Job-size* is measured in

bytes and may be a list of the following:

nnnn where *nnnn* is an integer that specifies the maximum job size for this job grade.

*n*K where *n* is a decimal number that represents the number of kilobytes and K is an abbreviation for kilobyte.

*n*M where *n* is a decimal number that represents the number of megabytes and M is an abbreviation for megabyte.

Any specifies that there is no maximum job size.

Here are some examples:

5000 represents 5000 bytes.

10K represents 10 kilobytes.

2M represents 2 megabytes.

Permit-type

This field contains a keyword that denotes how to interpret the ID list. The following list contains the keywords and their meanings:

User ID list contains the login names of users permitted to use this job grade.

Non-user

ID list contains the login names of users not permitted to use this job grade.

Group

ID list contains the group names whose members are permitted to use this group.

Non-group

ID list contains the group names whose members are not permitted to use this job grade.

Id-list

This contains a list of login names or group names that are to be permitted or denied queuing to this job grade. The list of names are separated by white space and terminated by a newline character. The keyword Any is used to denote that anyone is permitted to queue to this job grade.

The user job grade may be bound to more than one system job grade. It is important to note that the *Grades* file will be searched sequentially for occurrences of a user job grade. Therefore, any multiple occurrences of a system job grade should be listed according to the restriction on the maximum job size.

While there is no maximum number for the user job grades, the maximum number of system job grades allowed is 52. The reason is that more than one user job grade can be mapped to a system job grade, but each user job grade must be on a separate line in the *Grades* file, for example:

```
mail    N   Any   User   Any □
netnews N   Any   User   Any
```

Given this configuration in a Grades file, these two user job grades will share the same system job grade. Since the permissions for a job grade are associated with a user job grade and not a system job grade, it is even possible for two user job grades to share the same system job grades and have two different sets of permissions for each one.

The default grade

The binding of a default user job grade to a system job grade can be defined by the administrator. The administrator must use the keyword default as user job grade in the user job grade field of the *Grades* file and the system job grade that it is bound to. The Restrictions and id fields should be defined as Any so that any user and any size job can be queued to this grade. Here's an example:

```
default a   Any   User   Any
```

If the default user job grade is not defined by the administrator, then the built-in default grade, Z, will be used. Because it is assumed that the restriction field is Any, multiple occurrences of the default grade is not checked.

9.4.8 The Limits file

The */etc/uucp/Limits* file is used to define the maximum number of *uucico*, *uuxqt* and *uusched* processes that run simultaneously on your system.

An entry in the *Limits* file has the following format:

service=*daemon* max=*value*

daemon stands for *uucico*, *uuxqt* or *uusched*, and *value* is the permitted limit value for this service. The entries

can be in any order.

If the *Limits* file does not exist or cannot be read, no upper limits are defined for any service. If a service is not specified, or if *value* does not contain a positive number, no upper limits are defined for the service in question.

The following example illustrates meaningful entries:

```
service=uucico max=5
service=uuxqt max=2
service=uusched max=2
```

9.4.9 The Permissions file

The */etc/uucp/Permissions* file specifies the permissions that remote machines have with respect to login, file access, and command execution.

The following items should be considered when using the *Permissions* file to restrict the level of access granted to remote machines:

- All login IDs used by remote machines to login for *uucp* communications must appear in one and only one LOGNAME entry.
- Any site that is called whose name does not appear in a MACHINE entry, will have the following default permissions/restrictions:
  - Local send and receive requests will be executed.
  - The remote machine can send files to your machine's */var/spool/uucppublic* directory.
  - The commands sent by the remote machine for execution on your machine must be one of the default commands; usually *rmail*.
- When a remote machine calls you, unless you have a unique login and password for that machine, you don't know if the machine is who it says it is.

9.4.9.1 How entries are structured

Each entry is a logical line consisting of individual lines terminated by a backslash (\) indicating that the logical line continues. Comment lines begin with a number sign (#) and they occupy the entire line up to a newline character. Blank lines are ignored (even within multi-line entries).

There are two types of entry in the *Permissions* file, as follows:

LOGNAME=*login_name*[:*name* ...] [*option*=*value* ...]
MACHINE=*system_name*[:*name* ...] [*option*=*value* ...]

LOGNAME specifies the permissions that take effect when a remote machine logs into your machine.

MACHINE specifies permissions that take effect when your machine logs into a remote machine.

9.4.9.2 Options

This section describes each option, specifies how they are used, and lists their default values.

REQUEST

When a remote machine calls your machine and requests the transfer of a file, this request is granted or denied based on the value of the REQUEST option. The string

REQUEST=yes

specifies that the remote machine can request the transfer of files from your machine.

The string

REQUEST=no

specifies that the remote machine cannot request file transfers from your machine. This is the default value. It will be used if the REQUEST option is not specified. The REQUEST option can appear in either a LOGNAME entry (a remote machine calls you) or a MACHINE entry (you call a remote machine).

SENDFILES

When a remote machine calls your machine and completes its work, it can attempt to take on the requests queued for it on your machine. The SENDFILES option specifies whether your machine can send the requests queued for the remote machine.

The string
SENDFILES=yes

specifies that your machine may send the work that is queued for the remote machine as long as it logged in as one of the names in the LOGNAME option. This string is mandatory if your machine is in "passive" mode with respect to the remote machine.

The string
SENDFILES=call

specifies that files queued in your machine will be sent only when your machine calls the remote machine. The call value is the default for the SENDFILES option. This option is only significant in LOGNAME entries, since MACHINE entries apply when calls are made out to remote machines. If the option is used with a MACHINE entry, it will be ignored.

READ and WRITE

These options specify the various parts of the file system that *uucico* can read from or write to. The READ and WRITE options can be used with either MACHINE or LOGNAME entries.

The default for both the READ and WRITE options is the *uucppublic* directory as shown in the following strings:
READ=/var/spool/uucppublic □
WRITE=/var/spool/uucppublic

The strings
READ=/ WRITE=/

specify permission to access any file that can be accessed by a local user whose access permissions are set to "other".

The value of these entries is a colon separated list of pathnames. The READ option is for requesting files, and the WRITE option for depositing files. One of the values must be a component of any full pathname of a file coming in or going out. To grant permission to deposit files in /usr/news as well as the public directory, the following values would be used with the WRITE option:
WRITE=/var/spool/uucppublic:/usr/news

It should be pointed out that if the READ and WRITE options are used, all pathnames must be specified because the pathnames are not added to the default list. For instance, if the */usr/news* pathname was the only one specified in a WRITE option, permission to deposit files in the public directory would be denied.

You should be careful what directories you make accessible for reading and writing by remote systems. For example, you probably wouldn't want remote machines to be able to write over your */etc/passwd* file, so */etc* shouldn't be open to writes.

NOREAD and NOWRITE

The NOREAD and NOWRITE options specify exceptions to the READ and WRITE options or defaults. The strings
READ=/ NOREAD=/etc WRITE=/var/spool/uucppublic

would permit reading any file except those in the */etc* directory (and its subdirectories - remember, these are prefixes) and writing only to the default */var/spool/uucppublic* directory. NOWRITE works in the same manner as the NOREAD option. Both options can be used in LOGNAME and MACHINE entries.

CALLBACK

The CALLBACK option is used in LOGNAME entries to specify that no transaction will take place until the calling system is called back. There are two aspects to the possible use of CALLBACK:

- Security: The calling system must first identify itself and is only called back if it is known on the called system.
- Cost: If you intend to carry out extensive data transmissions over long distances, you can choose the machine to which each call is charged.

The string
CALLBACK=yes

specifies that your machine must call the remote machine back before any file transfers will take place.

The default for the CALLBACK option is
CALLBACK=no

Note that if two sites have this option set for each other, a conversation will never get started.

MYNAME
: This option allows you to specify the name that UUCP will use to identify itself. It defaults to the node name of the system (as found by using the command, *uname -n*).

PUBDIR
: This option is used to override the standard *uucppublic* directory (default is */var/spool/uucppublic*).

DIRECT
: If this option is set to yes, files will be copied directly into the target directory, rather than being first placed in a temporary directory and then moved to the target directory when the transfer is complete. The advantages are savings in time and space. The risk is possibly corrupted or incomplete files in the target directory. Default is no.

COMMANDS
:  The COMMANDS option can compromise the security of your system. Use it with extreme care.

The *uux* program will generate remote execution requests and queue them to be transferred to the remote machine. Files and a command are sent to the target machine for remote execution. The COMMANDS option can be used in MACHINE entries to specify the commands that a remote machine can execute on your machine. Note that COMMANDS is not relevant in a LOGNAME entry; COMMANDS in MACHINE entries defines the command permissions regardless of whether we call the remote system or it calls us.

The string
COMMANDS=rmail

specifies the default commands that a remote machine can execute on your machine. If a command string is used in a MACHINE entry, the default commands are overridden. For instance, the entry
MACHINE=owl:raven:hawk:dove \ □
COMMANDS=rmail:rnews:lp

overrides the COMMANDS default so that the machines owl, raven, hawk, and dove can now execute rmail, rnews, and lp on your machine.

See the section entitled The Permissions file for more information.

In addition to the names as specified above, there can be full path names of commands. For example,
COMMANDS=rmail:/usr/lbin/rnews:/usr/local/lp

specifies that the command *rmail* uses the default path. The default path for remote execution is */usr/bin*.When the remote machine specifies *rnews* or */usr/lbin/rnews* for the command to be executed, */usr/lbin/rnews* will be executed regardless of the default path. In the same way, when *lp* is called, the */usr/local/lp* command is executed.

Including the ALL value in the list means that any command from the remote machine(s) specified in the entry will be executed. If you use this value, you give the remote machine full access to your machine. Be careful. This allows far more access than normal users have.

The string
COMMANDS=/usr/lbin/rnews:ALL:/usr/bin/lp

illustrates two points:

- The ALL value can appear anywhere in the string.
- The pathnames specified for *rnews* and *lp* will be used (instead of the default) if the requested command does not contain the full pathnames for *rnews* or *lp*.

  The VALIDATE option (see below) should be used with the COMMANDS option whenever potentially dangerous commands like *cat* and *uucp* are specified with the COMMANDS option. Any command that reads or writes files is potentially dangerous to local security when executed by the *uucp* remote execution daemon (*uuxqt*).

VALIDATE

The VALIDATE option is used in conjunction with the COMMANDS option when specifying commands that are potentially dangerous to your machine's security. It is used to provide a certain degree of verification of the caller's identity. The use of the VALIDATE option requires that privileged machines have a unique login/password for *uucp* transactions. An important aspect of this validation is that the login/password associated with this entry be protected. If an outsider gets that information, the VALIDATE option can no longer be considered secure. VALIDATE is merely an added level of security on top of the COMMANDS option (though it is a more secure way to open command access than ALL).

Careful consideration should be given to providing a remote machine with a privileged login and password for *uucp* transactions. Giving a remote machine a special login and password with file access and remote execution capability is like giving anyone on that machine a normal login and password on your machine. Therefore, if you cannot trust users on the remote machine, do not provide that machine with a privileged login and password.

The LOGNAME entry
LOGNAME=uucpfriend VALIDATE=eagle:owl:hawk

specifies that if one of the remote machines that claims to be eagle, owl, or hawk logs in on your machine, it must have used the login uucpfriend. If an outsider gets the uucpfriend login/password, security is no longer guaranteed.

But what does this have to do with the COMMANDS option, which only appears in MACHINE entries? It links the MACHINE entry (and COMMANDS option) with a LOGNAME entry associated with a privileged login. This link is needed because the execution daemon is not running while the remote machine is logged in. In fact, it is an asynchronous process with no knowledge of what machine sent the execution request. Therefore, the real question is: How does your machine know where the execution files came from?

Each remote machine has its own "spool" directory on your machine. These spool directories have write permission given only to the UUCP family of programs. The execution files from the remote machine are put in its spool directory after being transferred to your machine. When the *uuxqt* daemon runs, it can use the spool directory name to find the MACHINE entry in the *Permissions* file and get the COMMANDS list, or if the machine name does not appear in the *Permissions* file, the default list will be used.

The following example shows the relationship between the MACHINE and LOGNAME entries:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
COMMANDS=rmail:/usr/lbin/rnews \
READ=/  WRITE=/
LOGNAME=uucpz VALIDATE=eagle:owl:hawk \
REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

The value in the COMMANDS option means that remote mail and */usr/lbin/rnews* can be executed by remote users.

In the first entry, you must make the assumption that when you want to call one of the machines listed, you are really calling either eagle, owl, or hawk. Therefore, any file put into one of the eagle, owl, or hawk spool directories is put there by one of those machines. If a remote machine logs in and says that it is one of these three machines, its execution files will also be put in the privileged spool directory. You therefore have to validate that the machine has the privileged login uucpz.

MACHINE entry for "other" systems

You may want to specify different option values for the machines your machine calls that are not mentioned in specific MACHINE entries. This may occur when there are many machines calling in and the command set changes from time to time. The name OTHER for the machine name is used for this entry as shown below:

```
MACHINE=OTHER \
COMMANDS=rmail:rnews:/usr/lbin/Photo:/usr/lbin/xp
```

All other options available for the MACHINE entry may also be set for the machines that are not mentioned in other MACHINE entries.

Combining MACHINE and LOGNAME entries

It is possible to combine MACHINE and LOGNAME entries into a single entry where the common options are the same. For example, the two entries

```
MACHINE=eagle:owl:hawk REQUEST=yes \
READ=/  WRITE=/
LOGNAME=uucpz REQUEST=yes SENDFILES=yes \
READ=/  WRITE=/
```

share the same REQUEST, READ, and WRITE options. These two entries can be merged as shown below:

```
MACHINE=eagle:owl:hawk REQUEST=yes \
LOGNAME=uucpz SENDFILES=yes \
READ=/  WRITE=/
```

9.4.10 The Poll file

The */etc/uucp/Poll* file contains information on how often a remote machine is to receive a send request. Each entry in this file contains the ID for the remote machine, followed by a tab character (a blank space is not suitable for this function), and the time at which the machine is to be called.

An entry in the *Poll* file has the following format:

systemname <TAB> *hour ...*

The *uudemon.poll* script does not execute the request; it merely creates a job file (*C.sysnxxxx*) for calls in the spool directory. *uudemon.hour* starts the scheduler, which then checks all the job files in the spool directory.

The following example shows an entry in the *Poll* file which ensures that the machine eagle receives a send request every four hours:

eagle 0 4 8 12 16 20

9.4.11 The Sysfiles file

The */etc/uucp/Sysfiles* file allows you to specify groups of *Systems*, *Devices*, and *Dialers* files for use with *uucp* and *cu*. This type of optional file is useful in any of the following cases:

- You require different *Systems* files so that requests for login services can be sent to addresses other than those of the *uucp* services. Furthermore, because the *Systems* file in particular can be very big, it is advisable to divide it into several smaller files.
- You require a number of *Devices* files for different device types.
- You require different *Dialers* files, so that different methods of establishing a connection can be used for *cu* and *uucp*.

An entry in the *Sysfiles* file has the following format:

service=*command*[:*command*] *filename=file*[:*file* ...] ...

command stands for uucico or cu. It is also possible to specify both commands, separated by a colon.

filename stands for one of the keywords systems, devices or dialers, which identify the file type.

file stands for a file which is used as a *Systems*, *Devices* or *Dialers* file, depending on the file type. If a full pathname is not specified, it is assumed that the particular file is located in the directory */etc/uucp*. A number of files can be specified, separated by a colon. In this case, the files are searched in the specified order, i.e. if the first file does not contain the appropriate entry or is errored, the second file is searched, and so on.

A backslash at the end of a line can be used to continue an entry onto the next line.

If different *Systems* files are defined for *uucico* and *cu* services, your system stores two different system lists. You can view the *uucico* list using the command *uuname* and the *cu* list using the command *uuname -c*.

The following example illustrates how to use local files, as well as general files:

```
service=uucico    systems=Systems.local_cico:Systems \
devices=Devices.local_cico:Devices \
dialers=Dialers.local_cico:Dialers
service=cu        systems=Systems.local_cu:Systems \
devices=Devices.local_cu:Devices \
dialers=Dialers.local_cu:Dialers
```

9.4.12 The remote.unknown file

The *remote.unknown* file is a binary program file that is activated when a remote machine that is not found in any of the *Systems* files tries to establish a connection. This attempt is logged, and the connection is dropped.

If you change the permissions of the *remote.unknown* file so it cannot execute, your system will accept

connections from any system.

9.5 Administrative files

The Basic Networking Utilities administrative files are described below. These files are created in the */var/spool/uucp/nodename* directory to lock devices, hold temporary data, or store information about remote transfers or executions. *nodename* indicates the name of the remote machine.

9.5.1 Temporary files

Temporary files are created by Basic Networking processes when a file is received from another machine. The names of these files have the following format:

TM.*pid*.*ddd*

pid is the process-ID of the Basic Networking Utilities process. *ddd* is a sequential three-digit number starting at 0.

When a complete file is received, it is moved to the pathname specified in the job file that caused the transmission. If processing is abnormally terminated, the temporary file may remain in the *nodename* directory. Temporary files should be automatically removed by *uucleanup*.

9.5.2 Lock files

There are two kinds of lock files:

- LCK lock files prevent duplicate conversations and transfers. The names of these files have the following format:

  LCK.*system*.*grade*

  system is the name of the remote system. *grade* is the system job grade being processed.
- LK lock files prevent duplicate use of calling devices. The names of these files have the following format:

  LK.*MAJ*.*maj*.*min*

  MAJ is the major number of the device containing the directory entry. *maj* and *min* are the major and minor numbers, respectively, of the device itself.

Both kinds of lock files contain the process ID of the process holding the lock. This process ID remains valid as long as the process is active.

9.5.3 Job files

Job files are created when jobs (file transfers or remote command executions) have been queued for a remote machine. The names of these files have the following format:

C.*sysnxxxx*

sys is the name of the remote machine. *n* is the ASCII character representing the grade (priority) of the job. *xxxx* is the four-digit job sequence number assigned by *uucp*. Job files contain the following information:

- Type of request, S (send) or R (receive)
- Pathname of the file to be sent or received
- Pathname of the destination or user file name
- User login name
- List of options
- Name of the associated data file in the spool directory. If the *uucp –c* or *uuto -p* option was specified, a dummy name is used
- Access permissions of the source file
- Remote user's login name to be notified upon completion of the transfer

9.5.4 Data files

Data files are created when it is specified in the command line to copy the source file to the spool directory.

The names of these files have the following format:

D.*systmxxxxyyy*

systm is the first five characters in the name of the remote machine. *xxxx* is a four-digit job sequence number assigned by *uucp*. The four digit job sequence number may be followed by a sub-sequence number, *yyy*, which is used when there are several data files created for a job file.

9.5.5 Control files

Control files are created when processing terminates abnormally. The names of these files have the following format:

P.*systmxxxyyy*

systm is the first five characters in the name of the remote machine. *xxxx* is a four-digit job sequence number assigned by *uucp*. The four digit job sequence number may be followed by a sub-sequence number, *yyy*, which is used when there are several control files created for a job file.

When the entire file is received, it is moved to the pathname specified in the job file that caused the transmission. Unlike the temporary file, a control file is not removed automatically. Therefore, when the transfer session is re-established, the length of the control file serves as an appropriate place to restart the transfer of the file, instead of restarting from the beginning. When check logging is terminated for a file transfer from another machine, the control file is used instead of a temporary file.

9.5.6 Execute files

Execute files are created in the spool directory prior to remote command executions. The names of these files have the following format:

X.*sysnxxxx*

sys is the name of the remote machine. *n* is the character representing the grade (priority) of the work. *xxxx* is a four digit number assigned by *uucp*. Execute files contain the following information:

- Requester's login and machine name
- Name of file(s) required for execution
- Input filename to be used as the standard input to the command string
- Machine and file name to receive standard output and standard error from the command execution
- Command string
- Option lines for return status requests

9.6 Log files

The BNU commands make log files available, some of which are optional. These files are described below with the help of examples.

9.6.1 The command log file

This file contains the commands issued by the user or system administrator. It can help the system administrator in trouble-shooting. The full pathname of the file is */var/spool/uucp/.Admin/command*. The format of each entry is as follows:

user1	(5/16-19:00:31)	uucp test.c mach1!~/user2
a	b	c

a: the user's login name

b: date and time the command was entered

c: command line

9.6.2 The system history log file

The system history log files can be created using *uucp*, *uucico*, *uux* or *uuxqt*, and stored in the subdirectories *uucp*, *uucico*, *uux* and *uuxqt* of the directory */var/spool/uucp/.Log*. These files contain records of all the activities that change the status of the system and the queue. The following is a sample entry recorded by *uucico* in /var/spool/uucp/.Log/uucico/mach1:.

```
uucp  |mach1|  |(2/12-8:18:42,  |2427,  |0)
a     b     c  d                e       f

CONN FAILED  |(NO DEVICES AVAILABLE)
g            h
```

a: user who submitted the job
b: name of the remote machine
c: job ID if a job is currently being processed; otherwise null
d: date and time that the entry was written to the file
e: process ID of *uucico* in this example
f: file transfer sequence number within the current invocation of *uucico*\
g: status message
h: status message elaboration

9.6.3 The error log file

This file contains the network's error messages. The full pathname of the file is */var/spool/uucp/.Admin/errors*.

Example:

```
ASSERT ERROR  |(uucico)  |pid: 1513  |(5/15-9:03:45)  |can't open
a             b          c           d                e

system  |3  |[FILE:pk1.c,  |LINE:356]
f       g   h              i
```

a: error type
b: program name
c: process ID
d: date and time that this entry was written to the file
e: error message part 1
f: error message part 2
g: error number
h: name of calling module
i: line of calling module where the error occurred

9.6.4 The transfer log file

This file contains information pertaining to file transfer. For example, it contains the number of bytes transferred and how long the transfer took. After both *uucicos* (master and slave) agree on the protocol, the transfer information is written into the */var/spool/uucp/.Admin/xferstats* file.

Example:

ihx!	user	M	(5/20-6:10:10)	(C,	6559,	1)	[DKH]
a	b	c	d	e	f	g	h

-> 1024/0.13 secs,	7877 bytes/sec	""
i	j	k

a: name of remote system
b: user login
c: M=master, S=slave
d: date and time that entry was written to log
e C=*uucico*, U=*uucp*, X=*uux*, Q=*uuxqt*
f: process ID of *uucico*, *uucp*, *uux*, or *uuxqt*
g: sequential number for each file transferred in a session
h: device name of media
i: direction and data bytes / clock time to transfer
j: transfer rate (bytes/sec)
k: "PARTIAL FILE" if the file was not completed because of a transmission error; "" if the file was transmitted completely (as in this example)

Statistical details on file transfer can be recorded in a file irrespective of the transfer protocol used. The *–sfile* option is specified with the *uucp* command for this purpose. This file contains three lines for each file transfer request.

Example:

First line:

uucp job:	mach1N1131	(6/12-13:34:58)	(0:0:19)
a	b	c	d

a: message header (always the same)
b: job ID assigned by *uucp*
c: date and time the file transfer completed
d: *hh:mm:ss* of elapsed time between time of creation of queue file and completion of the file transfer

Second line:

ihnp1!	/n15/user1/liblog	->	mach1!	~/userid	(user1)
a	b	c	d	e	f

a: sender node name
b: source file name
c: direction (always the same)
d: receiver node name
e: destination directory/file name
f: requester login ID

Third Line (status):

copy succeeded

a

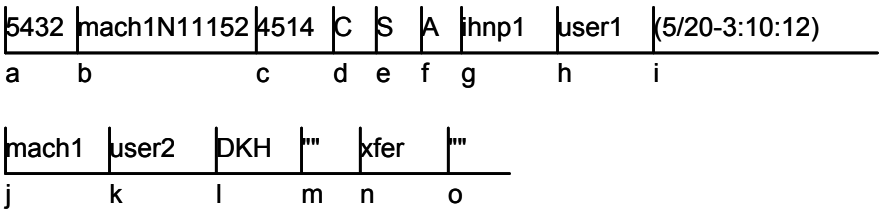
a: message

9.6.5 The accounting log file

This file contains information required for network accounting. When a job is completed successfully, accounting information is written to the */var/spool/uucp/.Admin/account* file of the person who submitted the job, if the transaction involved file transfer. If a remote program was executed, the accounting information is written to the */var/spool/uucp/.Admin/account* file of the system where the program was executed (target system).

Accounting information is only collected if an account file exists and is writable by *uucp*; the file is not created automatically. As an administrator, you can create or remove the accounting log on your machine.

Example:



a: user ID

b: job ID assigned by *uucp*

c: size of job in bytes if the job transaction is a file transfer;
time of job in seconds if the job transaction is a remote execution

d: C = job completed. P = partial job completed

e: service class of the job:
- premium
- standard
- economy.

At present, only "standard" service class is supported.

f: job grade identification

g: originating system's name

h: originator's login name

i: date and time the job originated

j: destination system's name

k: destination user's login name

l: device name of media

m: ID of physical network (always "")

n: type of transaction:
xfer = file transfer
rexe = remote execution.

o: The command if the job transaction is a remote execution.

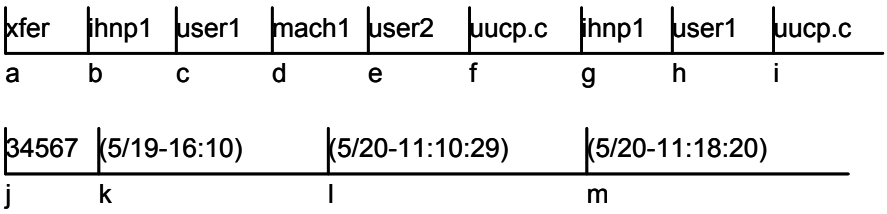
9.6.6 The security log file

This file contains the job transactions that attempt to violate system and user security measures. An attempted security violation is detected when the requester fails to pass the security checks specified in the */etc/uucp/Permissions* file, or when an attempt is made to access a protected source or destination file. The occurrence is logged for further analysis in the */var/spool/uucp/.Admin/security* file. Two different entries can appear in the security log.

xfer file transfer

rexe remote execution

Example:



a: record type (always *xfer*)

b: requester node name

c: requester user login

d: destination node name

e: destination user login

f: destination file name

g: source node name

h: source file owner login

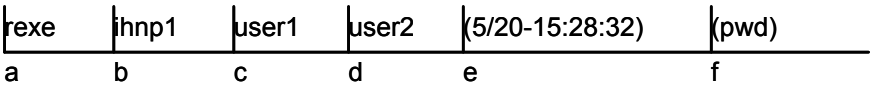
i: source file name

j: source file size in bytes

k: modification date and time of source file

l: date and time that transfer started

m: date and time that transfer completed



a: record type (always rexe)
b: client (requesting) node name
c: client (requesting) user login
d: server (destination) user login
e: date and time that command was executed by server
f: command name and options

9.6.7 The performance log file

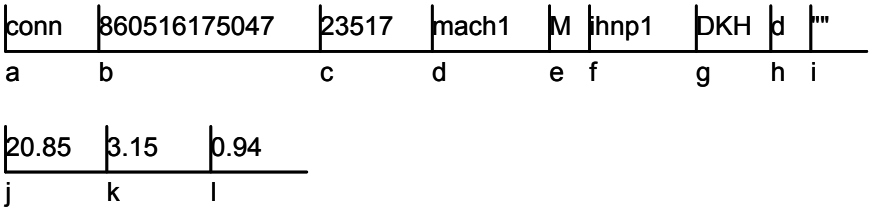
This file contains statistics about the operation of *uucico*. *uucico* writes the log entries to */var/spool/uucp/.Admin/perflog*. Statistics are only collected if *perflog* exists when *uucico* starts; the file is not created automatically. As an administrator, you can turn on or turn off performance logging at your machine. Two types of records will be written to the file:

Fields that are not applicable or unknown are marked by double quotes.

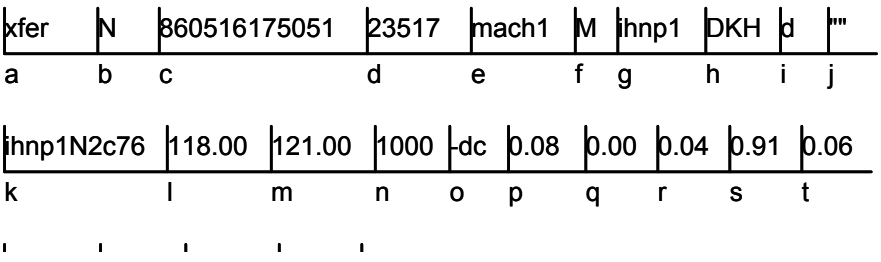
conn contains statistics about the successful establishment of a connection

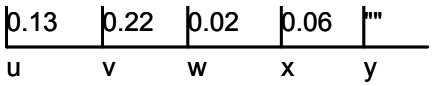
xfer contains statistics about a file transfer.

Example:



a: record type (always conn)
b: time stamp *YYMMDDhhmmss* for sorting
c: *uucico*'s process ID
d: the name of the machine where the record was written
e: M = master, S = slave
f: name of remote system
g: device name of media
h: the protocol that was used for communication
i: physical network ID (always "")
j: real time to connect
k: user time to connect
l: system (kernel) time to connect





a:
record type (always xfer)

b:
job grade ID

c:
time stamp (*YYMMDDhhmmss*) for sorting

d:
uucico's process ID

e:
the name of the machine where the record was written

f:
M = master, S = slave

g:
name of remote system

h:
device name of media

i:
the protocol that was used for communication

j:
physical network ID (always "")

k:
job ID if master, "" if slave

l:
time in seconds that job was in queue if master, "" if slave

m:
turn around time in seconds if master, "" if slave

n:
size of the file that was actually transferred successfully or partially because of transmission error

o:
command line options if master, "" if slave

p:
real time to start up transfer

q:
user time to start up transfer

r:
system (kernel) time to start up transfer

s:
real time to transfer file

t:
user time to transfer file

u:
system (kernel) time to transfer file

v:
real time to terminate the transfer

w:

user time to terminate the transfer

x:
system (kernel) time to terminate the transfer

y:
"PARTIAL FILE" if the file was not completed because of transmission error; "" if the file was transmitted completely (as in this record)

Start-up time includes the time for the master to search the queues for the next job, for the master and slave to exchange job vectors, and the time to open files.

Transfer time is the time it takes to transfer the data, close the file, and exchange confirmation messages.

Termination time is the time it takes to send mail notifications and write status files.

Turnaround time is the difference between the time that the job was queued and the time that the final notification was sent.

9.6.8 The foreign log file

This file contains a list of unknown systems that attempted to connect to the current machine. The full pathname of the file is */var/spool/uucp/.Admin/Foreign*. The format produced by *remote.unknown* is as follows:

Wed Jan 16 15:44:20 1987	udummy
a	b

a: date and time that the entry was written to the file

b: the message logged by *remote.unknown*

10 Transmission media

This chapter provides you with a description of the transmission media supported. These are:

- Ethernet
- FDDI
- Token Ring
- SLIP and PPP connections via serial lines

10.1 Ethernet

Ethernet is a standard for local area networks (LANs), developed in the early 70s by Xerox PARC. At present, seven main types of Ethernet are in use:

10BASE5 (Thick Wire or "Yellow Cable")
: This is a thick coaxial cable (with an internal diameter of approx. 1.3 cm). It can have a length of up to 500 m and can be set up with a maximum of 100 connections. It is therefore especially well suited to large-scale LANs. The cable is, however, quite difficult to install and is relatively expensive.

10BASE2 (Thin Coax or "Cheapernet")
: This coaxial cable is thinner than the 10BASE5 cable and can have a maximum length of 185 m. Up to 30 stations may be connected. The 10BASE2 cable is easier to handle and is therefore also easier to install than the 10BASE5 cable. It is also cheaper.

10BASE-T or UTP (Unshielded Twisted Pair)
: This cable can be up to 100 m long and connects two devices. UTP cables are the same cables used to install telephones. The amount of effort required to install it and the cost is therefore reduced to a minimum.

10BASE-F (Fiber optic)
: This cable is a fiber optic cable that can have a maximum length of two kilometers. It is very expensive but at the same time very reliable and powerful. A 10BASE-F cable can also be used for other purposes. Like a UTP cable, this cable is used to connect two devices.

100Base-FX (Fiber optic)
: This cable can be used in "full-duplex mode" up to a length of 2 km (in other modes the lengths are shorter). Although fiber optic cables are more expensive than copper cables, such as 100BASE-T for instance, they offer much better performance. Cables of this type are used to connect two devices.

100Base-T (Fast Ethernet)
: UTP CAT5 cables (unshielded twisted pair, category 5) may be up to 100 m long and are used to connect precisely two devices.

1000Base-SX (Gigabit Ethernet)
: The name SX on the physical layer specifies transfer using a short-wave laser via a multi-mode fiber optic cable. 62.5 micrometer fiber optic cable with a length of between two and 260 meters or 50 micrometer fiber optic cable with a length of between two and 550 meters can be used. The cable connects precisely two devices (see also [
Gigabit Ethernet Rich Seifert, Addison Wesley]).

The description below of the way in which Ethernet functions is based on a 10BASE5 cable. It is a coaxial cable up to 500 meters in length. To avoid the reflection of the electrical signals, a resistor is installed at each end between the cable core and the sheath.

Ethernets can be extended by means of repeaters. A maximum of two repeaters are permitted. This means that the maximum length of a network is 1500 meters.

Each connection between the Ethernet and the networked machines consists of the following components:

- Transceivers connected to the Ethernet cable
- Transceiver cables between the transceivers and the network interface
- The machine's interface

The transceiver converts the analog electrical signals received into digital signals and vice versa. It can also determine whether or not signals will be sent at a particular time.

The transceiver cable between the transceiver and the network interface transports not only the signals but also supplies the transceiver with power.

Unlike 10BASE5 or 10BASE2 cables, which are based on a bus concept, "twisted pair" and "fiber optic" cables are based on a structured cabling concept. All network components such as machines, switches, hubs, etc. form a star topology. Direct connections between two machines are established using "cross connect" cables.

10.1.1 Characteristics of Ethernet

Ethernet is a technology with distributed access control. When data is transferred from one station to another using bus technology and a network whose linkage points are hubs, all stations receive this data. If switches are used, the data only flows via the connection between the two stations.

The transfer speeds are 10 Mbit/s, 100 Mbit/s (Fast Ethernet) and 1 Gbit/s (Gigabit Ethernet).

Unlike other networks, Ethernet does not possess a central administration that controls access to the network. Access to the network is called "Carrier Sense Multiple Access with Collision Detect" (CSMA/CD).

Because Ethernet has a maximum packet size, each transmission is of a limited duration. A short pause occurs between the transmission of each data packet so that it is not possible for a single station to occupy the network exclusively, thus ensuring access to the network to all stations.

10.1.2 Detecting and dealing with collisions (half-duplex)

When a transceiver initiates data transmission, its signal does not reach all the parts of the network simultaneously. The signal is transmitted through the network at approximately 80% of the speed of light. This means it is possible for two transceivers to start their transmissions at the same time, resulting in collisions.

In order to ensure that such collisions do not have any effect, each transceiver monitors the cable during data transmission to determine whether a foreign signal has disrupted its own signal. This type of monitoring is called collision detect (CD). If a collision is detected, the network interface aborts transmission, waits until the Ethernet is free and attempts transmission again.

During a renewed attempt to transmit data, there is again the danger of both transceivers starting transmission at almost the same time. To avoid this happening, each sender delays its activities by a defined time unit (different for each sender), by two time units after the second attempt, four time units after the third failed attempt, and so on. Furthermore, another brief delay is built into each transmission to prevent two transceivers from sending simultaneously again.

10.1.3 Addressing

Each machine connected to the Ethernet has its own Ethernet address or MAC address. This ensures that the data packets transmitted reach only the machine for which they were intended. Only data packets with this target address (or an appropriate broadcast or multicast address) are forwarded from the network interface to the machine.

The Ethernet address consists of 48 bits and is assigned by the manufacturer of the Ethernet controller. This means that every controller in the world can be identified no matter which machine it is installed in. The Ethernet address is also referred to as the physical address or the hardware address. There are three types of Ethernet addresses:

- The physical address of a network interface

  The first three bytes of the Ethernet address are the same for all controllers of a specific vendor. The last three bytes are assigned uniquely by the vendor.
- The broadcast address

  This address is reserved for sending data to all the stations in the network simultaneously.
- The multicast address

This address is used to send data simultaneously to a group of stations in the network.

The network interfaces must be capable of storing all three address types. A network interface accepts at least two types of data:

- Data with a physical address
- Data addressed to the broadcast address

Some interfaces can also accept multicast addresses.

10.1.4 Fault tolerance for Gigabit Ethernet Controllers ("Dual Homing")

Reliant UNIX can be configured with a feature referred to as "Dual Homing", which allows the failure of a Gigabit Ethernet Controller to be tolerated transparently for TCP/IP.

Dual Homing (or DH for short) means that where there is a pair of PAGE controllers (CL18) within a system, one of these controllers works actively while the other stays in standby mode.

If the active controller fails, the standby controller automatically takes over the work of the defective controller. This is fully transparent for the local system and for the other systems in the network because both controllers of the DH pair work with the same MAC and IP address. TCP connections, in particular, are not interrupted by the switchover.

Since DH is based on what is called the Spanning Tree Protocol, this protocol must be enabled on the switches to which the PAGE controller is connected. It should be noted that DH will only work if both controllers are connected to one switch, which supports the Spanning Tree Protocol. This means that DH cannot be used for two systems linked in cross-connect mode.

The new */usr/sbin/altdh* command is used to configure the DH feature of the *alt* driver (PAGE, CL18) and to allow its status to be displayed. You will find further details on DH in the *altdh(8)* manual page.

10.2 Token Ring

In a Token Ring network (standard IEEE 802.5), a token (with a size of 1 byte) carries out a continuous circuit of the ring, passing every station connected to the ring. When the token arrives at a station, that station can stop the token and send a data packet round the ring. When the data packet arrives back at the station, the station regenerates the token on the network, thus giving the next station the chance of sending a packet.

Transmission rates of 4 Mbits and 16 Mbits are supported. The boards used in a Token Ring network must all have the same transmission rates, otherwise the network will come to a complete standstill.

The most important differences to Ethernet (IEEE 802.3) are:

- The use of a collision-free protocol
- The topology (ring instead of bus)
- The ring management functions

10.2.1 Structure of the Token Ring

A minimal Token Ring consists of one MAU (**M**edium **A**ccess **U**nit), to which eight stations are connected via data cables. The data cables have a 9-pin connector at one end, which is connected to the system's Token Ring board, and at the other end a connector for the MAU. These cables between the controller and the MAU are also referred to as "lobes". Patch cables can be used as extensions.

As well as the 8 terminal ports, a MAU also has a ring-in port and a ring-out port. Several MAUs can therefore be connected; patch cables are used to connect the ring-in port of one MAU to the ring-out port of the next.

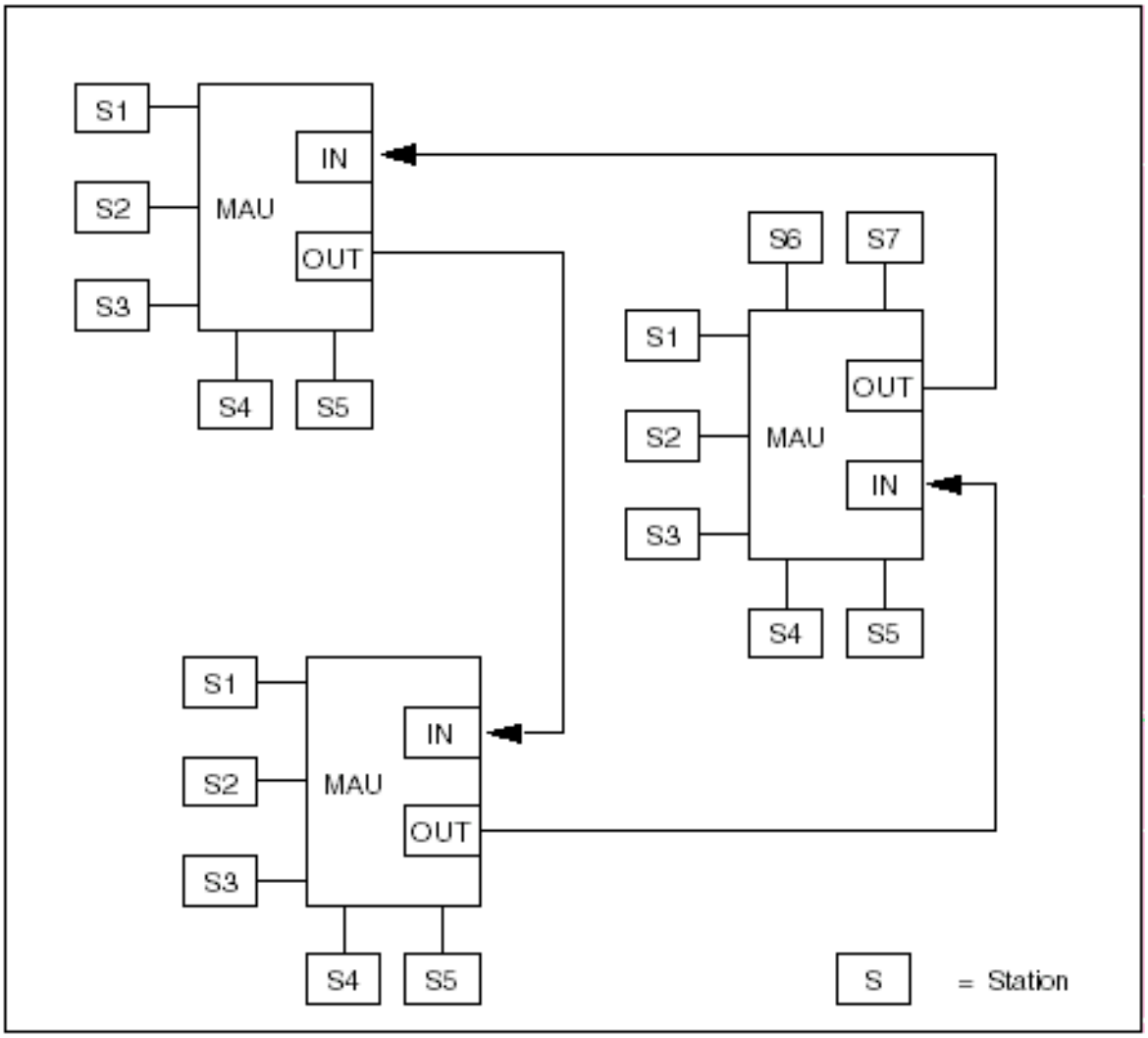


Figure 18: Token Ring networking

The first station connected to the Token Ring is the ring monitor; each additional station is a standby monitor. The ring monitor is, for example, responsible for the following:

- Detecting ring errors
- Monitoring the master clock in the ring
- Initiating the NAUN process (every 7 seconds) via which the stations receive information on their ring neighbors
- Detecting the loss of a token

In the event of an error, the ring monitor sends a RING-PURGE-MAC frame. If the standby monitors detect that the ring monitor has crashed, they initiate the token claiming process, which determines which station is to be the new ring monitor. This is usually the station with the smallest MAC address.

Like Ethernet, Token Ring networks can also be extended with bridges. Bridges work on layer 2 of the ISO reference model (the data link layer): they connect two Token Ring networks and pass packets from one network to the other only if the destination station of the packet is not on the same network.

With Ethernet, communication between the connected stations is completely transparent. With Token Rings, however, each end station must include, after the Token Ring header in its packet, the source routing information which indicates via which links the destination system is to be accessed.

Remove-MAC frame

The ring manager of a Token Ring network can send a Remove-MAC frame to a station in the ring at any time, if, for example, it detects that the station is sending faulty packets. When this happens, the controller disconnects itself from the ring, and communication is then no longer possible via this network interface. If the *madge* controller (RM400/600) is used, a message is output on the console when this happens. In the case of the RM600 LCT controller, this information is made available by the CMX error notification system.

If a station is removed from the ring by a Remove-MAC frame, the user should contact the system administrator of the ring manager to find out the cause. Before the station is reconnected to the ring, when using the LCT Multibus II controller, the controller must be reloaded (see Loading the Token Ring and FDDI controllers (RM600 LCF, LCT only)). The system does not carry out automatic reconnection; the system administrator has to initiate it (after removing the cause of the error) using the *ullcadmin* command.

10.2.2 Implementing source routing

The source routing implementation of Reliant UNIX V5.45 conforms to the RFC 1042 recommendations. A cache with routing information, similar to the ARP cache (which contains the mapping of Internet addresses to physical addresses), is used. This cache contains the mapping of physical addresses to routing information.

Configuration options

The following variables can be set in */etc/conf/pack.d/ullc/space.c* (a new system kernel must be built if any changes are made):

- *ullc_token_mtu*

  The default value is 0; in other words, use the MTU supplied by the driver.

  Determines the MTU used by the data-link-layer independently of the layer 3 protocol being used. If set to 0, the default values supplied by the driver for the MTU are used (4472 or 17800). If a value greater than 0 is entered, it is used as the MTU, as long as it does not exceed the maximum MTU, which is 4472 or 17800 depending on the speed (4 or 16 Mbit). If a bridge to a destination machine does not support the default MTU, *ullc_token_mtu* should be entered. Examples of reasonable values are: 516, 1470, 2052, 4472, 8144, 11454, 17800.

- *ullc_token_routing_on*

  The default value is 1 (i.e. on). Switches source routing on and off. It is a good idea to switch the source routing off if you are not using Token Ring bridges, as this reduces the overhead.

- *ullc_token_allroutes_br, ullc_token_singleroutes_br*

  Format of single-routes and all-routes-broadcast packets. If you do not want all-routes-broadcast packets to be sent, you can assign a single-routes-broadcast value to *ullc_token_allroutes_br*, or you can change the longest-frame field (some routers send only single-routes broadcasts). Should not be changed.

- *ullc_token_rcv_allinfo*

  The default value is 0 (i.e. off). Specifies whether the routing information of all received packets (including the data packets) is to be entered in the RIF cache, or only the routing information of LLC packets which are not unnumbered information and of ARP packets. Should be switched off to reduce the overhead.

- RIF cache size

  There is one RIF cache for each Token Ring interface (default: 171). This should not be changed.

The following variables can be set in */etc/conf/pack.d/arp/space.c* (a new system kernel must be built if any changes are made):

- *app_token_mtu*

  (Default: 8136)

  Specifies the MTU to be used for the Internet protocol family. It is reduced by 8 bytes relative to the *ullc_token_mtu*. These are the 8 bytes for the LLC/SNAP header which is in the data part of the packet. If the MTU used there is greater than *app_token_mtu*, the IP-MTU is reduced. *app_token_mtu* is currently "8136" to restrict the MTU at speeds of 16 Mbit. Reasonable values are: 17792 to make use of the full MTU at 16 Mbit, but at least 2002 to conform to RFC 1042.

10.2.3 Error diagnostics

If there is a machine on the ring which is operating normally, the *tcpdump* program should be used for debugging purposes. *tcpdump* switches the controller to promiscuous mode and displays either all or only selected packets with decoded TCP/ IP headers. If no filter expression is used, all LLC and MAC packets are displayed, whereby only the decoded headers of the IP packets are displayed and not the contents of the packets.

Options:

-e: Additional display of the link level header with routing information

-x: Print the entire packet contents in hexadecimal form

-A *no:*

Attachment unit with reference to the ULLC (default 0). This unit number is incremented for all controllers connected under ULLC.

It is identical to the device number of the device node

/dev/dlpi/madge.... (RM400)
/dev/dlpi/lct... (RM600)

The options are followed by a filter expression, e.g.:

host *host*:

Display all packets from/to this machine

host *host1* and *host2*:

Display data traffic between two machines

Other filters according to protocol, port etc. are also possible.

10.2.4 Token Ring drivers

You will find information on the configuration and diagnostic options of the Token Ring driver for the *madge* controller (RM400/RM600) in the description of *madge(7)* in the "Networking Reference Manual". Refer to the section entitled Loading the Token Ring and FDDI controllers (RM600 LCF, LCT only) for information on installing and running the RM600 Token Ring driver for the LCT controller.

10.3 FDDI - Fiber Distributed Data Interface

FDDI is the local area network standard specified in the ISO9314-x document. It is a glass fiber-based ring network with a transmission speed of 100 Mbit/s. A duplicate ring is used to reduce the risk of failure; this second ring has not been used (up to now) in error-free operation.

FDDI also uses a token-based system as described for Token Ring in the previous section.

Ring networks can be set up with a maximum of 500 stations and a ring length of no more than 100 km. The maximum distance between two stations is 2 km.

Layers 1 and 2a of the ISO/OSI reference model have been standardized. Layer 1 has been subdivided into a PMD (**P**hysical **M**edium **D**ependent) layer and a PHY (**Phy**sical) layer. In addition, **S**tation **M**anagemen**t** (SMT), which is not contained in the ISO/OSI reference model, has been standardized in

ISO 9314-6.

SMT can be regarded as the part of the FDDI station responsible for administration. It has the following tasks, amongst others:

- Connection management (connecting/disconnecting the stations)
- Ring management (e.g. recognizing duplicated addresses)
- Administering the Management Information Base (MIB), in other words the configuration database
- Exchanging sequential SMT information frames with the other stations in the ring

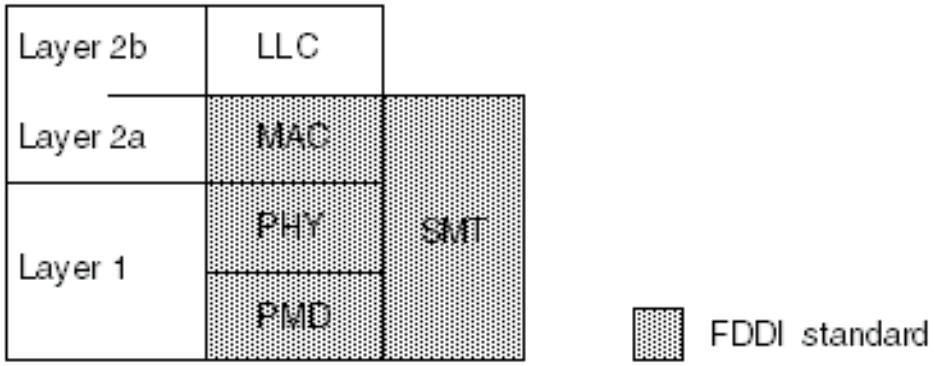


Figure 19: FDDI ISO layers

10.3.1 Ring components and topologies

An FDDI ring consist of the following components:

- Data stations

  These are machines that are connected to the FDDI LAN.

- Concentrators

  These are distribution boxes with ports for additional data stations or concentrators. In other words, concentrators can be cascaded.

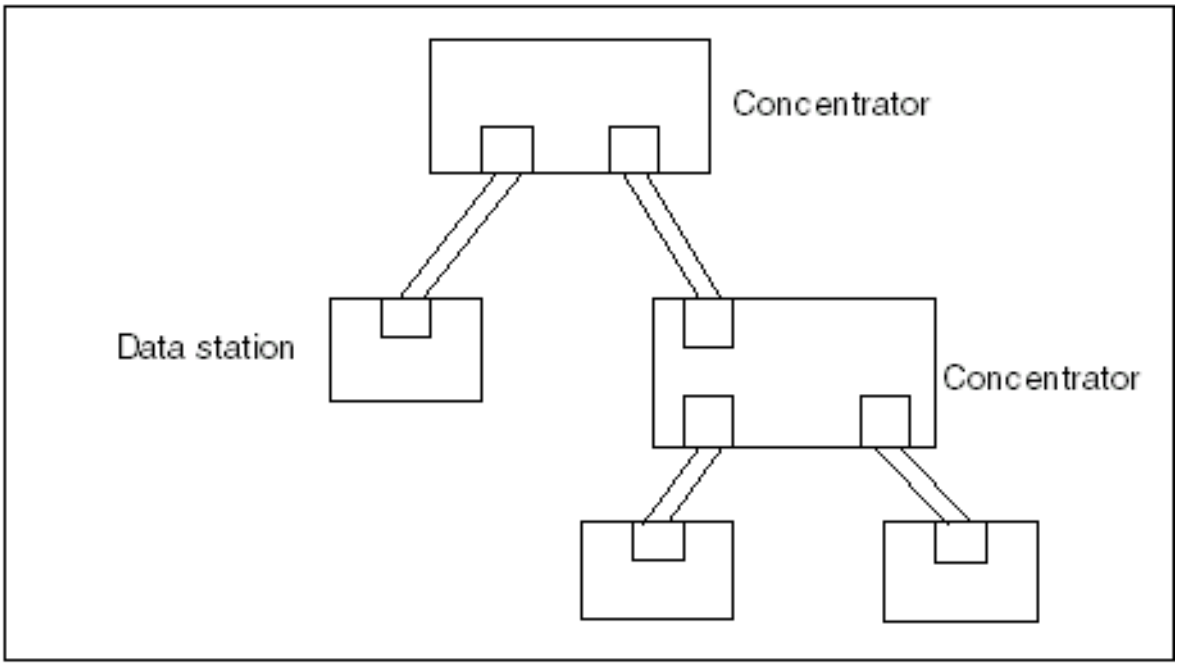


Figure 20: Components of an FDDI ring

Data stations and concentrators can be connected to the FDDI ring in two different ways. A distinction is drawn between a trunk ring, which is a dual ring consisting of a primary ring and a secondary ring running in opposite directions, and tree structures, which, as shown in the diagram, consist of at least one concentrator and one or more data stations. An FDDI ring that contains all these components is referred to as a dual ring of trees.

The components are linked by FDDI cable. An FDDI cable contains two glass fibers; in other words, a port for an FDDI cable contains an optical receiver and an optical transmitter.

The components in the trunk ring are connected to their neighbors by a single FDDI cable in each case. In problem-free operation, only one fiber of each cable is used (it appears as a thicker line in the diagram).

The ports of a component to the predecessor and successor in the dual ring are known as port A and port B.

The components in the trunk ring are called DAS (dual attached station) and DAC (dual attached concentrator).

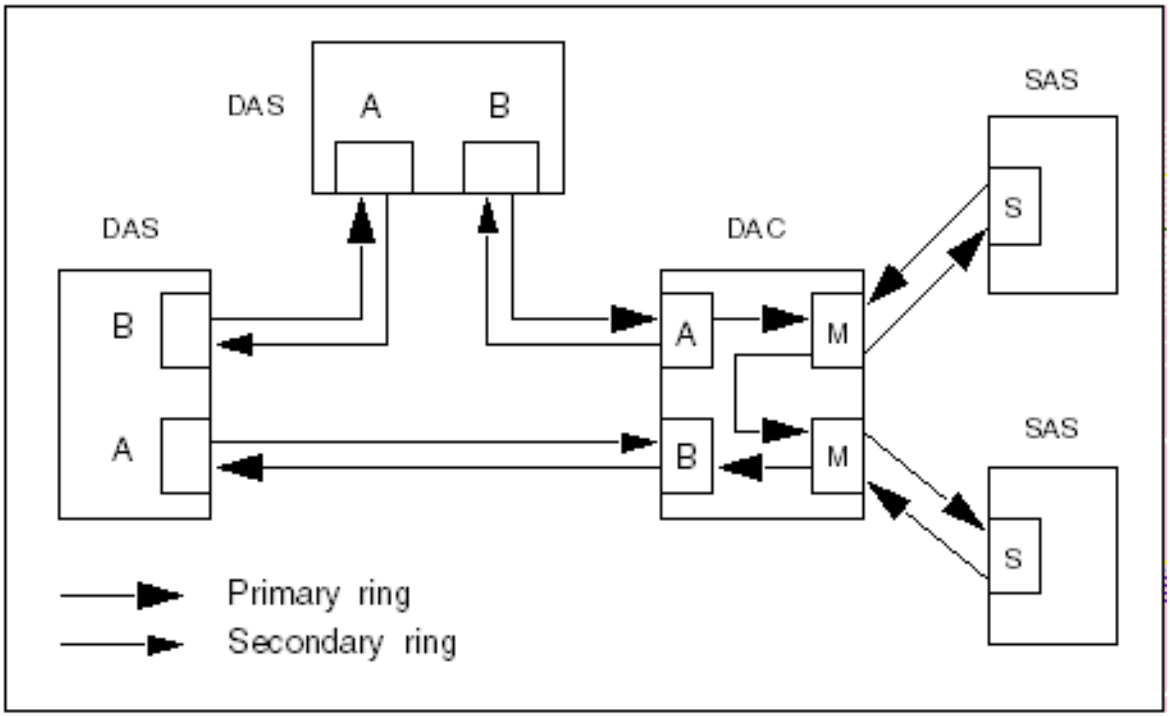


Figure 21: Dual attached stations and a dual attached concentrator

The components in the tree structures are connected to each other by the same cables that are used in the trunk ring. In this case, both fibers of the cable are used. The ports in the concentrators are M (master) ports, and those in the stations or cascaded concentrators are S (slave) ports. The components of the tree structure are known as SAC (single attached concentrator) and SAS (single attached station).

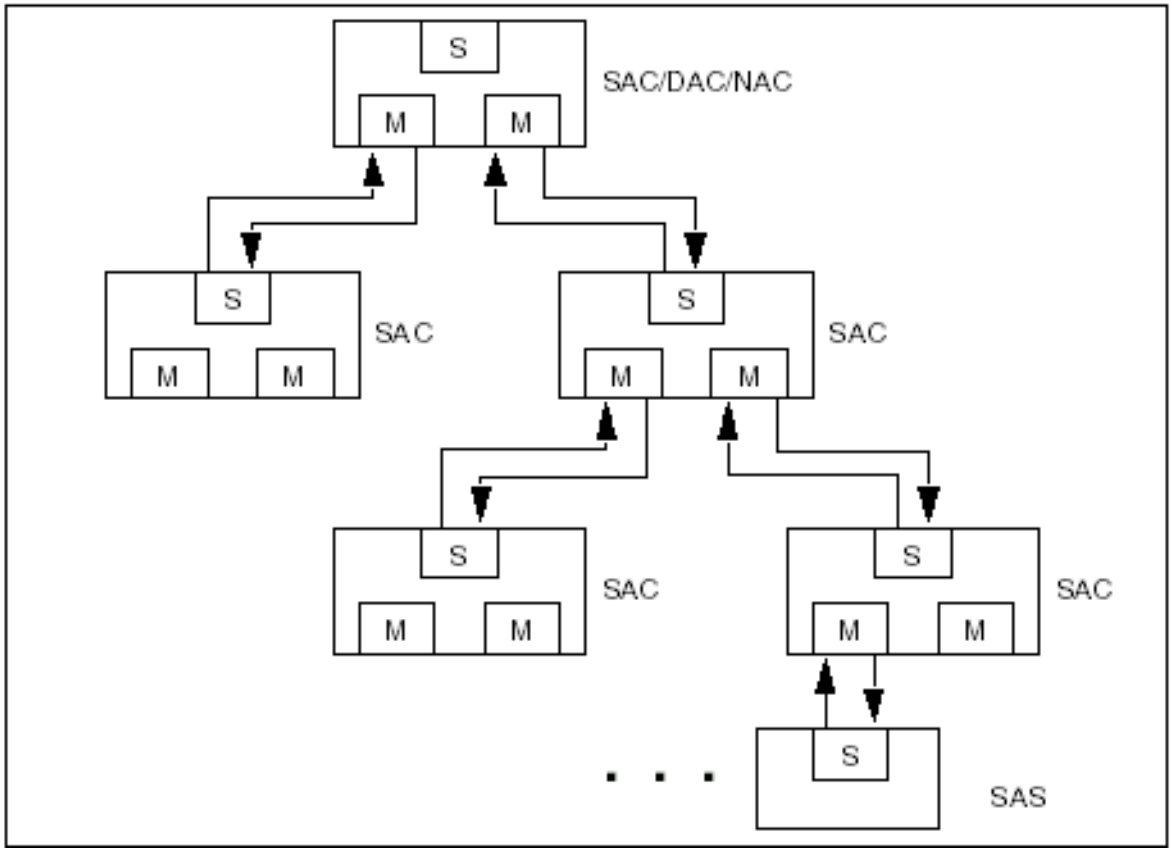


Figure 22: Tree structure

In smaller configurations, the FDDI ring must be a tree structure. The concentrator required for this is an NAC (null attachment concentrator). Its job could of course also be done by a DAC or an SAC whose A/B or S port is not used.

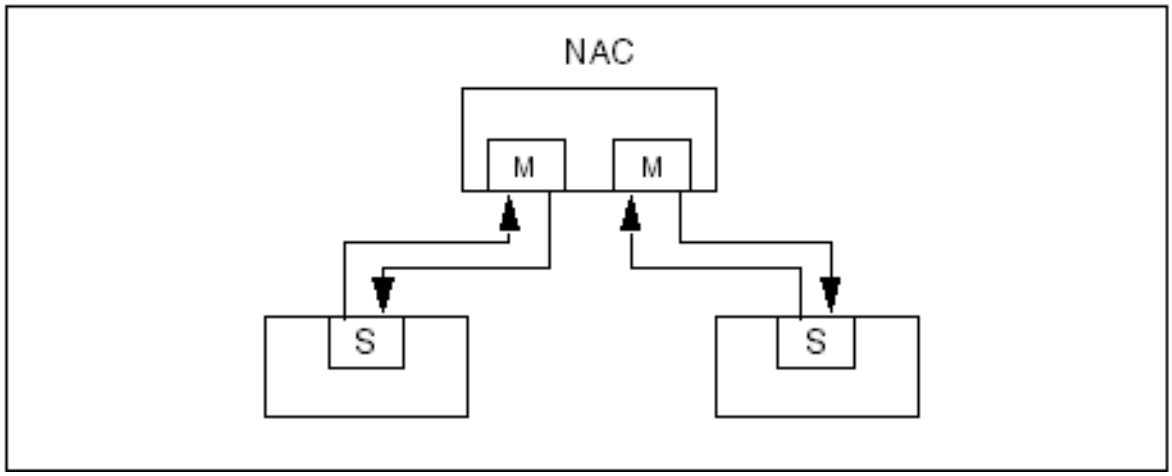


Figure 23: Null attachment concentrator

In a minimum configuration or for test purposes, two stations can also be connected to each other directly by

an FDDI cable.

10.3.2 Availability of the FDDI ring

The FDDI standard provides various mechanisms to ensure that the ring continues to function even when stations are switched off or fail.

10.3.2.1 Stations in the trunk ring

Generally, only stations that have a very high degree of availability are accepted in the dual ring.

If one of these stations nevertheless fails or there is a problem with a cable, the broken dual ring is closed again via the secondary ring by means of the two adjacent to the relevant component. This is known as wrapping.

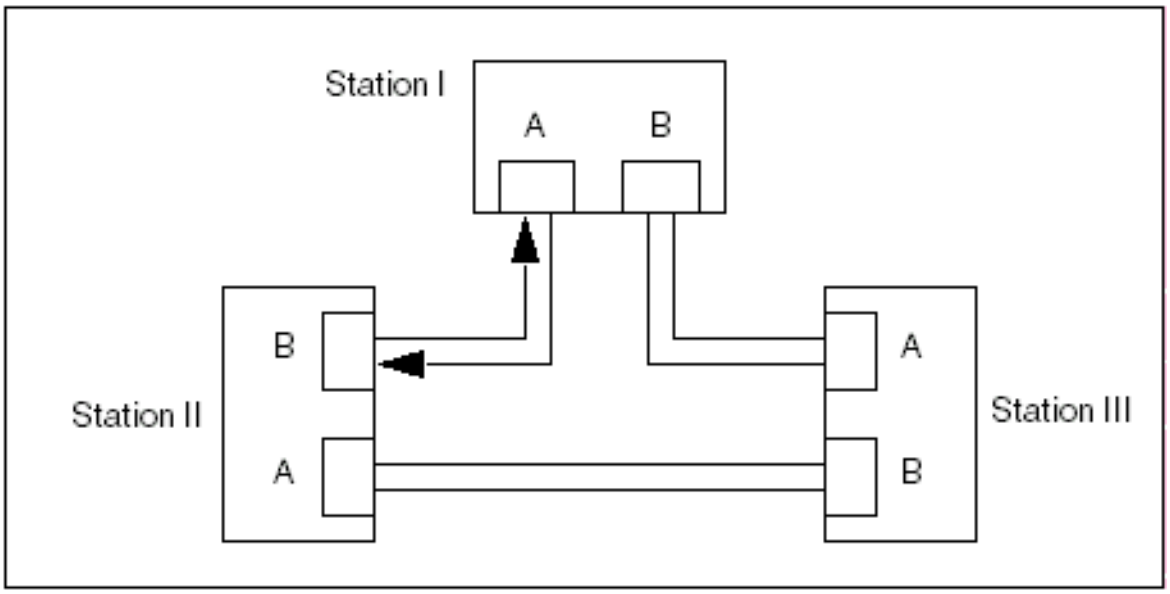


Figure 24: Wrapping in the event of a station failing (station III)

In the event of additional defects occurring in the trunk ring, renewed wrapping can result in a number of isolated subrings.

10.3.2.2 Optical bypass

To prevent the ring becoming segmented, it is possible to provide each endangered station with an optical bypass. In the event of the station failing, this bridges the ring optically at the affected point so that wrapping does not occur. However, attenuation means that the number of optical bypasses that can be used in a ring is limited and this solution is unsuitable for a configuration with a large number of stations that are not always available.

10.3.2.3 Stations at the concentrator

Stations that are frequently switched off or unavailable for other reasons should be connected to the ring as SASs by means of a concentrator. When the station fails, the connection is switched through the concentrator, thus preventing any impairment of the ring.

To prevent the ring being affected by the failure of concentrators, the FDDI standard employs the principle of redundancy in a system known as dual homing. A DAS station is connected to the M ports of two different concentrators at the same time. During problem-free operation, only one of the two connections is used. If the concentrator fails, an automatic switchover to the other concentrator is carried out.

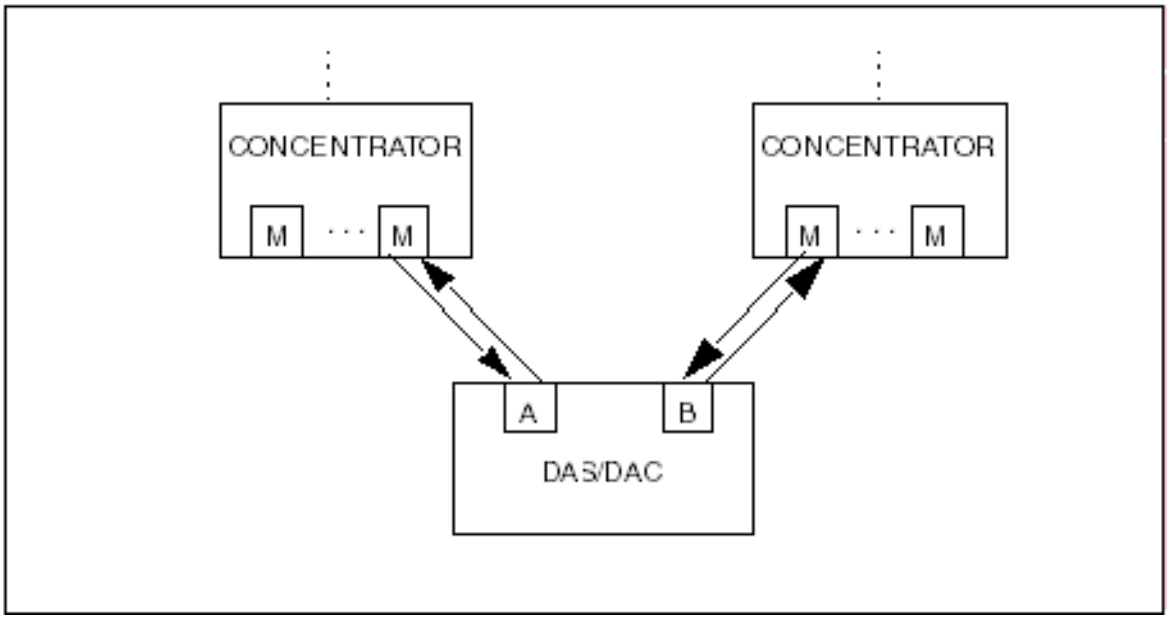


Figure 25: Dual homing

10.4 Loading the Token Ring and FDDI controllers (RM600 LCF, LCT only)

The Multibus II controllers for FDDI (LCF) and Token Ring (LCT) must be loaded before using RM600 systems. Loading the controllers (as well as dumping and administration) is undertaken by CMX. The FDDI or Token Ring drivers cannot communicate with their controllers until loading is completed.

The following packages must be installed in order to run FDDI and Token Ring on an RM600 using the LCF or LCT Multibus II controllers:

- CMX - includes the load drivers, among others
- FDDI - FDDI loadware or
- TR-LLC - Token Ring loadware

CMX is not included in the basic configuration of Reliant UNIX 5.45. It has to be purchased as a separate product.

After installation, these controllers (LCF and LCT) have to be configured. This is described in the section entitled Basic configuration.

When the system boots, the sequence in which the start scripts in *etc/rc2.d* are executed ensures that loading takes place in the correct order.

1. The controllers are loaded (S68cmx).
2. The network drivers access the controllers (S68lcf/S68lct).
3. The controllers are initialized, and the protocol stack is set up (S69inet).

In the online documentation for FDDI, under */opt/readme/fddi.D*, you will find information on controller maintenance and diagnostics in the event of problems occurring, as well as a configuration description.

If problems occur with the Token Ring controller, refer to the "CCP-TR" manual.

You will find information on CMX and the configuration menu in the "CMX Communications Manager in UNIX" manual.

10.5 SLIP - TCP/IP connections via serial lines

SLIP (Serial Line Interface Protocol) is a protocol that makes it possible to set up TCP/IP network connections via serial lines.

The SLIP protocol adds control characters to the packets that are to be transmitted to ensure that the receiver can recognize the beginning and end of the transferred blocks even when transmission is non-synchronous.

SLIP supports all network communication functions. Machines that are connected via SLIP can set up connections with *rlogin*, *rsh*, *rcp*, *ftp* and *telnet*.

SLIP connections between machines are implemented via null modem cables or modem.

The precondition for using SLIP is that the SLIP interface driver is incorporated in the UNIX kernel. (This driver is part of the base system.)

The maximum number of SLIP connections depends on the number of available serial interfaces, and is restricted by the number of configured IP providers. This is defined by the value of the IPPROVCNT variable. The value can either be changed by means of the *idtune* command or directly in the */etc/conf/pack.d/ip/space.c* file. After changes have been made, a new system kernel has to be generated.

10.5.1 Preparing to use SLIP

The preparations that must be made before SLIP can be used on a machine depend on:

- Whether the machine also has a LAN interface (e.g. Ethernet)
- Whether the machine only has SLIP interfaces

Preparing to use SLIP on a machine with a LAN interface

The following steps must be taken on a machine with a LAN interface before SLIP can be used:

- The SLIP interfaces must be configured (see the Section "Configuring and attaching a SLIP interface").
- Any required routes must be entered (see the Section "Deactivating a SLIP interface").

Preparing to use SLIP on a machine without a LAN interface

The following steps must be taken on a machine without a LAN interface before SLIP can be used:

- The name and Internet address of your machine and all the machines to be accessed via SLIP connections must be entered in the */etc/inet/hosts* file.

  The machine name must be made known to the system when the operating system is installed.
- The SLIP interfaces must be configured (see the Section "Configuring and attaching a SLIP interface").
- If the Network Information Service NIS (see the Chapter "The Network Information Service (NIS)") is to be used, the following must be done:
  - The domain name must be made known to the system by means of the *domainname* command. To ensure that this is done each time the system is started, the domain name should be entered in the */etc/default/inet* file.
  - The name of the NIS master server must be entered in the first line of the file */var/yp/binding/[domainname]/ypservers*. If the file does not exist, it must be created. The file */etc/hosts* must contain the entry for the NIS master server.
- Any routes that may be required must be entered (see the Section "Routing with SLIP").

10.5.2 Configuring and attaching a SLIP interface

To use SLIP interfaces on a machine, they must be configured and attached to a TTY interface using the *slattach* command.

The *slattach* command attaches an as yet unassigned SLIP interface to a device file */dev/term/ttyx*. There are a number of device files for serial connections available in your system. *x* stands for the number of the device file, for example 01. Use a number which has not been assigned.

The *slattach* command configures the TTY to be used for the SLIP protocol and sets the data transfer rate of the TTY interface and the packet size. The command must be called on both machines in order to activate the SLIP interface.

Example:

Entering the command

slattach -m 256 -s 19200 tty01 lochost remhost

configures the /dev/term/tty01 interface as a SLIP interface. The *-s* option is used to set the interface to 19200 baud (bit/sec) (the default is 38400). Other baud rates are also possible (see the *slattach* command in the "Networking Referencing Manual"), but make sure that you do not use a higher baud rate than is permitted for the interface. Two machines connected to one another must use the same baud rate.

The command configures the SLIP interfaces for a connection between a local machine (lochost) and a remote machine (remhost). Machine names are normally specified here, provided the machine names have been entered together with the associated Internet addresses in either the local file */etc/inet/hosts* or in the NIS map *hosts*. If this is the not the case, the Internet addresses must be specified. These addresses must always be unique throughout the network, whatever Internet interface you use - whether SLIP, Ethernet or some other type.

The option *-m 256* sets the maximum IP packet size to 256 bytes (default: 1006). IP packets are transported transparently in packets of a size set by means of *-m*. The same packet size must be set in both partner systems. Generally speaking, longer packets achieve better throughput, however with interactive applications (e.g. *ftp*, *rlogin*) the response times are increased.

Automatic configuration at system startup

slattach is automatically called at system startup (change to run level 2) if an entry is specified in the */etc/default/inet* file under STANDARDIF or EXTIF in the following format:

EXTIF=slip:*tty:[baud]:[mtu]:ip_source:ip_dest:[options]*

Meanings:

tty The device file of the interface used (e.g. /dev/term/tty00 or /dev/term/tty01).

baud The baud rate used (9600 / 19200 / 38400**)**.

mtu Maximum transmission unit (e.g. 256).

ip_source
The Internet address of the user's SLIP interface, either in full or as an alias from */etc/inet/hosts*. It is also possible to output the Internet address using dots.

ip_dest
The Internet address of the remote machine, either in full or as an alias from */etc/inet/hosts*.

options
Options as with other LAN interfaces (netmask, broadcast).

ip_source
Name of the local machine

options
No options

This type of configuration may be used only for "direct" connections. It may lead to problems in the case of modem connections due to the lack of carrier when starting up.

Example:

EXTIF=slip:/dev/tty01:38400::177.77.77.1:177.77.77.2

The TTY interface used in this case is /dev/term/tty01, the baud rate is 38400, and the MTU is 1006 bytes (default).

If more than one SLIP connection is required, you can create extra entries (e.g. EXTIF1, EXTIF2, ...). In this case, make sure that the names of these entries are added to the INTERFACES entry.

Checking a configured connection

The number of configured SLIP interfaces and their status can be checked by means of the *netstat -i* command.

You can check whether the connection is functioning with the *ping* command.

10.5.3 Deactivating a SLIP interface

You deactivate a SLIP connection with the *sldetach* command.

Example:

The command

sldetach term/tty05

causes the correct and controlled cleardown of the SLIP connection. The connection via TTY interface tty005 is then cleared down, and the TTY interface tty005 of this machine is again available for use.

10.5.4 Routing with SLIP

As it is possible to configure more than one SLIP interface on a machine, large-scale networks can be created using SLIP connections. This is done using the *route* command (see the *route* command in the "Networking Reference Manual").

Configuration examples

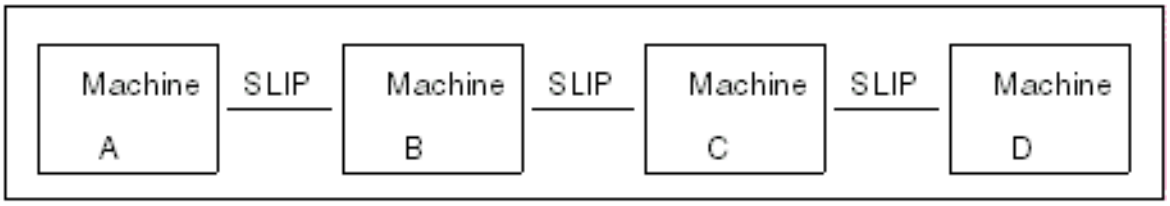


Figure 26: Bus configuration

Machines B and C each have two configured SLIP interfaces.

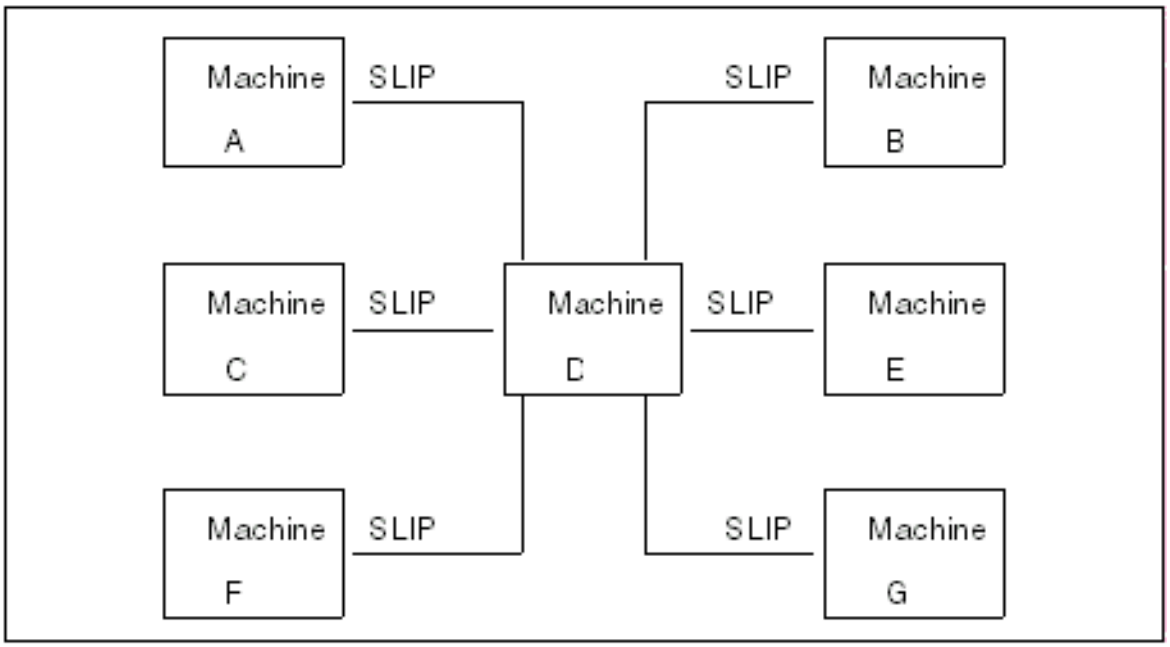


Figure 27: Star configuration

Machine D has six configured SLIP interfaces.

If a machine has more than one SLIP connection, it can function as an IP router. In other words, if packets arrive at this machine that are destined for somewhere else, this machine will try and forward these packets.

Coupling with LAN networks

If a LAN interface and one (or more) SLIP interfaces have been configured on a machine, this machine can act as a router between two networks.

This makes it possible, for example, to connect two LAN networks via the SLIP interfaces on two machines. Before this can be done, IPFORWARDING must be set as specified in the section entitled Configuring a router.

Example:

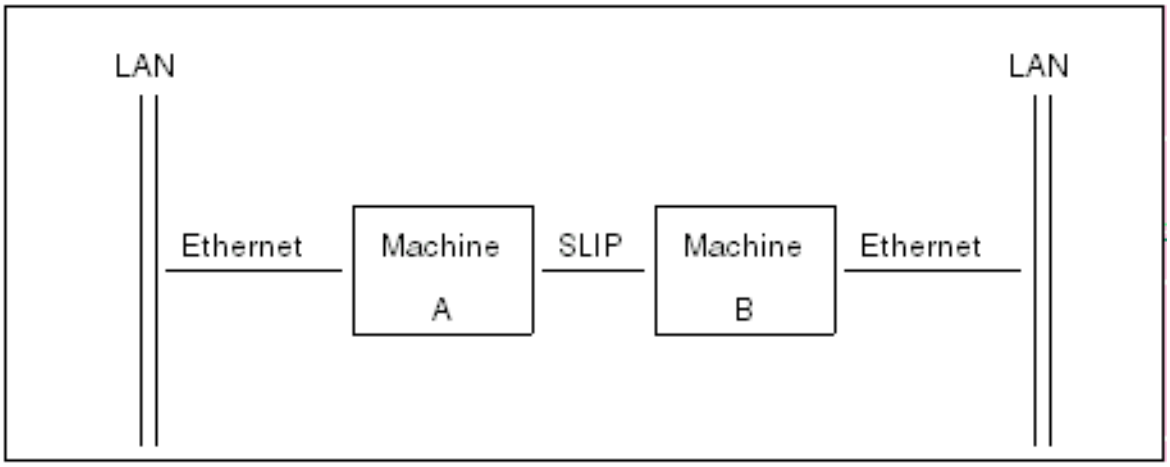


Figure 28: Coupling two Ethernet networks

Entries in the routing table

In order for a machine to be able to send packets to a remote machine to which it is not directly connected, an entry must be made in the routing table on the local machine. The routing table is maintained in the kernel and contains the following information:

- The name or Internet address of the machine to be accessed
- The name or Internet address of the machine that sets up the connection to the machine that you want to access
- The number of intermediate stages that must be passed through before you reach your destination

Example:

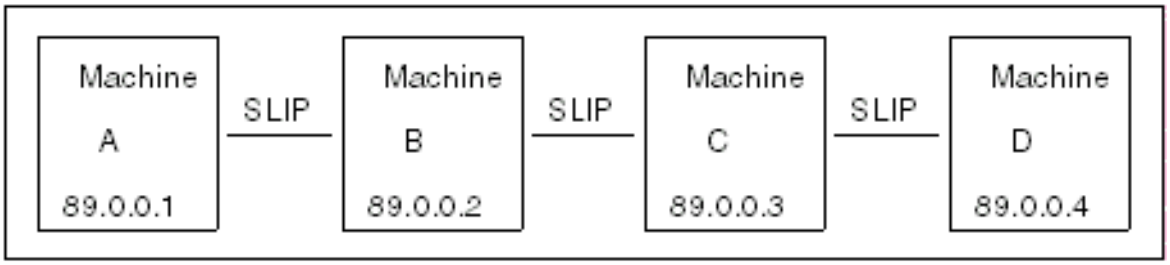


Figure 29: Routing with SLIP connections

The following entries must be made in the routing table if all the machines in this network are to be able to communicate with each other:

Machine	Entries in the routing table		
	Destination	Via	Stages
A A	89.0.0.3 89.0.0.4	89.0.0.2 89.0.0.2	1 2
B	89.0.0.4	89.0.0.3	1
C	89.0.0.1	89.0.0.2	1
D D	89.0.0.1 89.0.0.2	89.0.0.3 89.0.0.3	2 1

Appropriate entries must also made if the destination is a machine in a LAN network or if a whole LAN network is the destination.

Example:

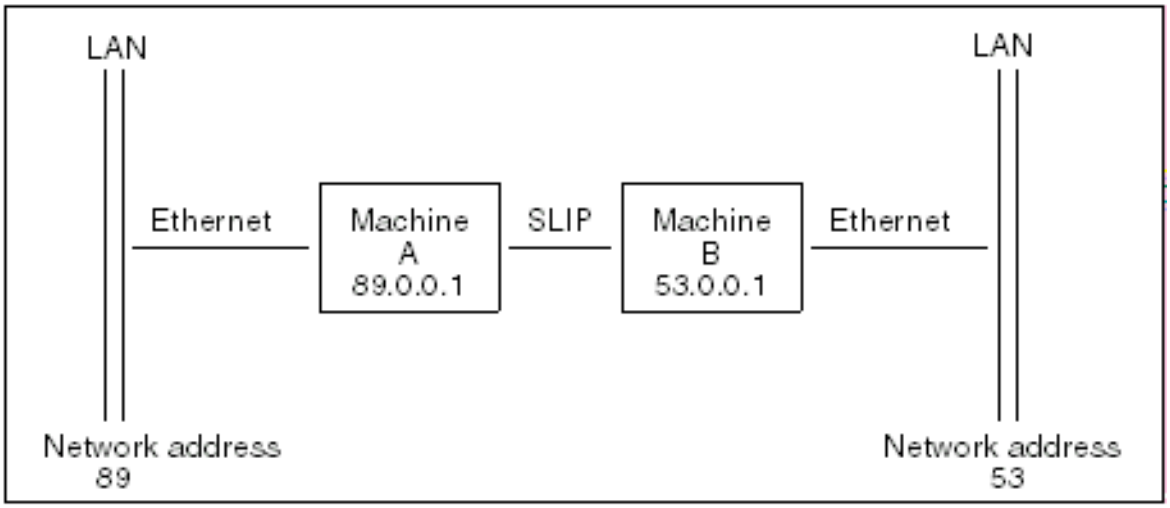


Figure 30: Routing with coupling of two Ethernet networks

The following entries in the routing tables are required:

Machine	Entries in the routing table		
	Destination	Via	Stages
A	53	53.0.0.1	1
B	89	89.0.0.1	1
In network 89	53	89.0.0.1	2
In network 53	89	53.0.0.1	2

Here, only the network address of the remote network is specified as the destination. This means that all the machines connected to this network can be accessed.

Entries are made in the routing table by means of the *route* command (see chapter 4 Expanding your network).

As the routing table is maintained in the system kernel, it must be created each time the system is started.

To avoid having to do this, entries can be made in the */etc/inet/routes* file. This file contains the commands that force the route entries.

10.5.5 SLIP and LAN on one machine

SLIP and LAN (e.g. Ethernet) may be used simultaneously. The same Internet address may be assigned to the LAN interface and to a SLIP interface on the same machine.

If this is the case, the connection to this machine will always be set up via SLIP even if a connection were possible via the faster LAN interface. If you want to be able to select whether a connection is set up via SLIP or LAN, the interfaces must be configured using different Internet addresses.

10.5.6 SLIP using modem connections

This connection mechanism should only be used when the connection cannot be established via PPP. (PPP is a substitute for SLIP and is described as of Section "PPP - point-to-point protocol".)

The following describes the setting up of a connection from a machine A called orion (client) to a machine B called neptun (server). The connection is established on the basis of the Basic Networking Utilities (BNU). In this case, the *cu* command is used as the dialer. For more information, refer to the chapter entitled The Basic Network Utilities (BNU).

10.5.6.1 The client (active end)

Setting up the first device node

If necessary, set up a (modem-capable) device node. Use the *sysadm* or *termadd* commands in order to do this. The *cu* will run via this node later.

1. *sysadm*

   Select the menu items *configuration*, *serial devices*, *ACTIONS* to reach the submenu for setting up a device node. Select a name (e.g. /dev/term/cu1).

   When the device node is set up, a port service is created for the relevant interface. This must be removed, because a connection setup using *cu* requires the interface to be available. Select the menu items *ports*, *port_services*, *remove* to reach the submenu for removing a port service. Select the corresponding service tag (in this case *cu*).

2. *termadd*

   You can also set up device nodes directly using the *termadd* command.

    Device nodes set up using the *termadd* command cannot be administered using *sysadm*.

   Syntax:

   termadd -T *terminal_type* -n *node_name* -b *board*

   These terms have the following meanings:

   terminal_type
   Type of terminal. The values can be: generic, serial

   node_name
   Name of the node

   board Board, interface to poll using *termadd -b*

   Examples:

   RM600/SINIX-Y (Terminal2 CSI)

   termadd -T generic -n cu1 -b csi0,3

   RM400/SINIX-N (Serial III)

   termadd -T generic -n cu1 -b motherboard0,6

   This only sets up the device node (no port service).

Setting up the second device node

Now set up a second device node with the same device classes and device numbers (major and minor device numbers). Use the *mknod(1M)* command to do this. The SLIP connection is subsequently activated via this node by means of the *slattach* command.

mknod *name* c *major minor*

These terms have the following meanings:

name Name of the node

major Device class number

minor Device number

Example:

The first device node to be set up has the major and minor device numbers 129, 258:

mknod /dev/term/slip1 c 129 258

BNU configuration for establishing the connection

The dialer integrated in the *cu* command uses the BNU configuration files under */etc/uucp* (see also the chapter entitled The Basic Network Utilities (BNU)).

Systems

The slave needs an entry for establishing the connection. The format of the entry is:

name time type class phone login

name Name of the server (e.g. neptun)

time A string determining the time when the server can be dialed (default: Any)

type The device type to be used for establishing the connection. This is a keyword which is the first field of an entry in the *devices* file (ACU in this case, for a dial-up connection).

class Transmission speed (default: 9600)

phone Telephone number

login Strings in the format *expect send*

The dialer searches at the input side for a pattern of the form *expect* and when it encounters one it sends the string *send*. (in this case, for a dial-up connection e.g.: "" \d\d\d'.)

In this example, the dialer at the input side waits until it receives an empty string (i.e. no input) and then waits another three seconds.

Example:

The sample configuration thus produces the following entry in the file *Systems*:

neptun Any ACU 9600 44193 "" \d\d\d

Devices

The *devices* file contains information about the device files used for establishing the connection. The entry has the following format:

type line line2 class dialer-token

type One of the keywords direct (for direct connections) or ACU (for modem connections). This keyword must correspond to the third field of an entry in the *systems* file.

line Device name of the port that is used (in this case, /dev/term/cu1)

line2 Not used; must contain a hyphen (-)

class Transmission speed (default: 9600)

dialer-token

This is a keyword which refers to an entry in the *dialers* file. *dialer-token* refers to the modem type used or direct for direct connections.

Example:

Modem type: v22bis

ACU /dev/term/cu1 - 9600 v22bis

Dialers

The *dialers* file contains initialization/dialing commands required by the modem used.

Example:

Modem type: v32bis

v32bis =,-, "" \M\dAT&F\r\c OK\r ATQ2X3&W\r\c
OK\rATZ\r\c OK\r ATM1\r\c OK\r AT&D2\r\c OK\r
AT&C1\r\c OK\r ATS0=1\r\c OK\r AT&E4\r\c OK\r
AT&E2\r\c OK\r \EATDP\T\r\c 00 \r\m\c

10.5.6.2The server (passive end)

Setting up the device nodes

If necessary, set up a separate (modem-capable) device node for SLIP. Use the *sysadm* or *termadd* commands in order to do this.

1. *sysadm*

   Select the menu items *configuration*, *serial devices*, *ACTIONS* to reach the submenu for setting up a device node. Select a name (e.g. /dev/term/cu2).

   After you have set up the node, make sure that a corresponding port service is active (in this case, */usr/bin/login*). In general, a sleep 999999 is active after the node has been set up.

   Select the menu items *ports*, *port_services*, *modify* to reach the submenu for modifying a port service. Select the corresponding service tag (in this case, *cu2*) and modify the following entries:

Create 'utmp'	Yes □ (Sets up an entry in the */var/adm/utmpx* file)
ttylabel	generic
Service command:	/usr/bin/login
Prompt:	Login:

2. *termadd*

   You can also set up device nodes directly using the *termadd* command.

    Device nodes set up using the *termadd* command cannot be administered using *sysadm*.

   Syntax:

   termadd -T *terminal_type* -n *node_name* -b *board* -1 *ttylabel* \
   -p *prompt* -t *tagname*

   These terms have the following meanings:

   terminal_type
   : Type of terminal. The values can be: generic, serial

   node_name
   : Name of the node

   board Board, interface to poll with *termadd -b*?

   ttylabel
   : Entry in */etc/ttydefs* (in this case, generic)

   prompt
   : Login prompt (in this case, login:)

   tagname
   : Name of the service tag

Example:

RM600/SINIX-Y (Terminal2 CSI)

termadd -T generic -n cu2 -b csi0,3 -l generic \
-p "login:" -t cu2

RM400/SINIX-N (Serial II)

termadd -T generic -n cu2 -b motherboard0,5 -l generic \
-p "login:" -t cu2

termadd implicitly creates the associated port service.

The BNU configuration files

The BNU configuration files are not used on the server.

10.5.6.3 Establishing the connection

The connection is established from the client.

Establish a connection with the server using the *cu* command and log yourself on as the superuser (root). If successful, this launches a login shell. For example, to set up a connection to a machine called neptun, enter the following command:

cu neptun

SLIP is activated from the login shell at the server end by means of the command *slattach*.

exec slattach [-s *baud*] [-m *mtu*] *device-node ip-source ip-dest*

These terms have the following significance:

baud The transmission speed used in the BNU configuration files (client) or in the tty label (server) (default: 38400)

-m *mtu*
Maximum transmission unit (maximum size of the data packets)

device-node
The name of the device node set up

ip-source
The Internet address of the server

ip-dest
The Internet address of the client

The *slattach* command overlays the login shell.

Example:

The Internet address of the server is 129.15.35.1 and the Internet address of the client is 129.15.35.2.

exec slattach -s 9600 /dev/term/cu2 129.15.35.1 129.15.35.2

The login shell is no longer active. You can no longer establish connections via this port using the *cu* command.

Enter the *cu* command ~! (in words: a tilde followed by an exclamation mark) in order to launch a subshell on the local machine (client). The *slattach* command is used to activate SLIP at the client end:

slattach [-s *baud*] [-m *mtu*] *device-node ip-source ip-dest*

These terms have the following meanings:

baud The transmission speed used in the UUCP config files (client) or in the tty label (server) (default: 38400)

device-node
The name of the device node set up

ip-source
The Internet address of the client

ip-dest
The Internet address of the server

Example:

exec slattach -s 9600 /dev/term/slip1 129.15.35.2 129.15.35.1

Please note the order in which the IP addresses are transferred. The Internet address (or the name) of the local machine must always be specified for *ip-source*, and that of the remote machine must be specified for *ip-dest*.

Once the SLIP has been activated, the *cu* command terminates with the following messages:

cu: Lost Carrier
cu: Disconnected

These messages only refer to *cu*, not to the SLIP connection.

The modem connection remains in existence until you execute the *sldetach* command on the client or the server.

10.5.6.4 Terminating the connection

To terminate the connection, start the *sldetach* command on both the client and the server:

sldetach *device_node*

device_node

The name of the device node used for the SLIP connection

10.6 PPP - point-to-point protocol

The point-to-point protocol can be used to establish TCP/IP connections between machines via serial lines. This makes it possible to run TCP/IP connections without LAN controllers. PPP supports all the communications functions (*rlogin*, *rsh*, *rcp* and *telnet*). PPP connections are implemented via serial ports on the machine. Machines that are to communicate with each other via PPP directly can be connected by modem or null modem cable.

The PPP mechanism is to be preferred to the SLIP mechanism for modem connections.

SINIX-PPP consists of several parts:

- The two STREAMS modules *ppp(5)* and *ppa(5)* at system kernel level
- The PPP daemon *ppd(1)* at user level
- The dynamically loadable protocol libraries *pap.so*, *chap.so* and *lqm.so*

The connection is established as follows:

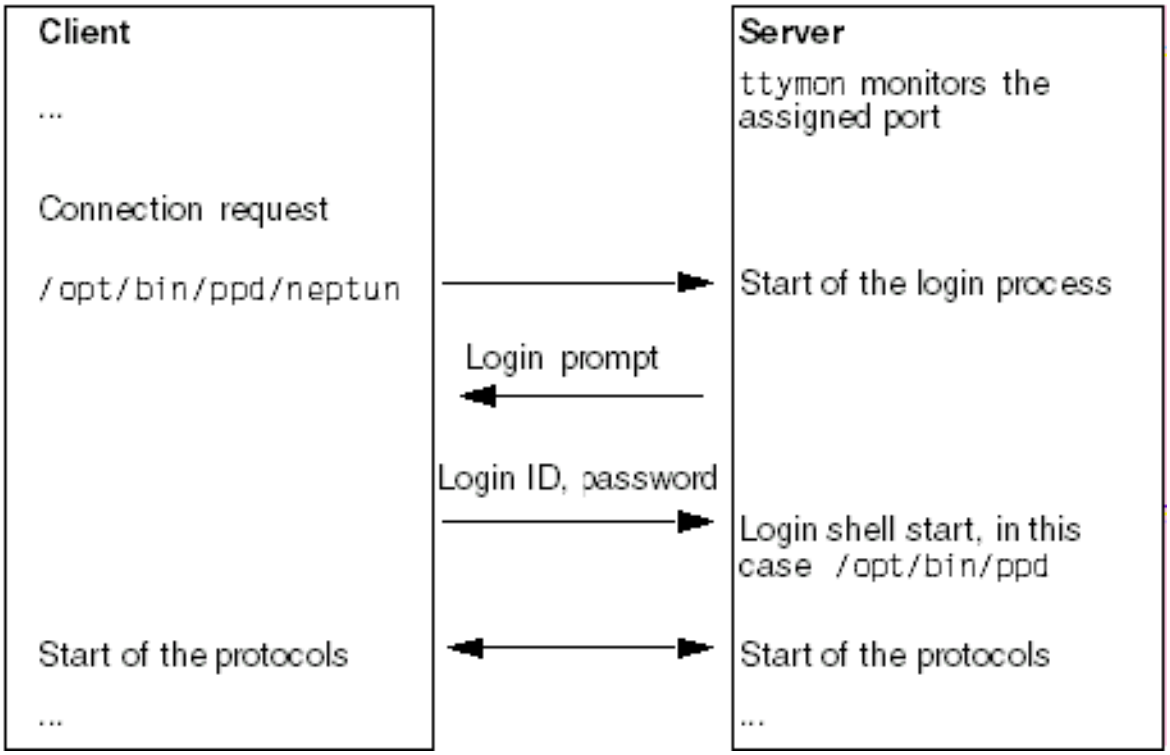


Figure 31: Establishing a connection between a PPP client and server

The maximum number of PPP connections possible on a machine is set in the MAX_PPDEVS variable (the default is 16). (See also the section entitled Adjustable variables in space.c.) Another restriction is the number of configured IP providers. This is set by the value of the IPPROVCNT variable. The values can be changed using the *idtune* command or directly in the */etc/conf/pack.d/ip/space.c* file. After changes are made, a new system kernel must be generated.

A description follows of how to establish a connection from a machine A called orion (client, active end) to a machine B called neptun (server, passive end). In this case, the machines can be connected via a direct line (null modem cable) or a modem link.

This results in an asymmetry in the configuration of the machines. After the connection is established (IP LAYER UP), both machines are completely symmetrical as far as the user is concerned.

10.6.1 The client (active end)

Setting up device nodes

Always set up an individual (modem-capable) device node for PPP. Use the *sysadm* or *termadd* commands in order to do this.

1. *sysadm*

   Select the menu items *configuration*, *serial devices*, *ACTIONS* to reach the submenu for setting up a device node. Select a name (e.g. /dev/term/ppp001).

   When the device node is set up, a port service is created for the corresponding interface. This must be removed, since establishing a connection via *cu* requires a free interface. To do this, select the menu items

ports, *port_services*, *remove*, and then select the corresponding tag (in this case, ppp001).

2. *termadd*

   You can also set up device nodes directly using the *termadd* command.

    Device nodes set up using the *termadd* command cannot be administered using *sysadm*.

   Syntax:

   termadd -T *terminal_type* -n *node_name* -b *board*

   These terms have the following meanings:

   terminal_type
   : Type of terminal. The values can be: generic, serial

   node_name
   : Name of the node

   board Board, interface to poll using *termadd -b?*

Examples:

RM600/SINIX/Y (Terminal2 CSI)

termadd -T generic -n ppp001 -b csi0,3

RM400/SINIX-N (Serial III)

termadd -T generic -n ppp001 -b motherboard0,6

This only sets up the device node (no port service).

BNU configuration for establishing the connection

The dialer integrated in the *ppd* PPP daemon uses the BNU configuration files under */etc/uucp* (see also the chapter entitled The Basic Network Utilities (BNU)).

Systems

Each server needs an entry for establishing the connection. The format of the entries is:

server_name time type class phone login

server_name
: Name of the server (e.g. neptune)

time A string determining the time when the server can be dialed (default: Any)

type The device type to be used for establishing the connection. This is a freely-selectable keyword which corresponds to the first field of an entry in the *devices* file. (direct in this case, for a direct connection.)

class Transmission speed (default: 9600)

phone
: Telephone number (direct for direct connections)

login Strings in the format *expect send*□
On the input side, the dialer looks for the *expect* format pattern and sends the *send* string if its search is successful. For a direct connection in this case, e.g.:
'"" \r ogin: ppp word: ppplogin'

This means:

- The dialer is waiting for an empty string on the input side (i.e. no input)
- The dialer then sends a carriage return. This starts the login process on the server.
- The dialer waits for the "ogin:" model (login prompt) and answers with the login ID (in this case, ppp)
- The dialer then waits for the "word:" model (Password:) and sends the password (in this case, ppplogin)

The client logs onto the server as user ppp.

Example:

In the configuration of the example, this results in the following entry in the *systems* file:

Direct connection:

neptun Any Direct 9600 direct "" \r ogin: ppp word: ppplogin

Dial-up connection:

neptun Any ACU 9600 44193 ogin: ppp word: ppplogin

Devices

The *devices* file contains information about the device files used for establishing the connection. The entries are in the following format:

type line line2 class dialer-token

type One of the keywords direct (for direct connections) or ACU (for dial-up connections). This keyword is the reference of a *type* field in the *Systems* file.

line Device name of the port that is used (in this case, /dev/term/ppp001)

line2 Not used; must contain a hyphen (-)

class Transmission speed (default: 9600)

dialer-token

This is a keyword which refers to an entry in the *Dialers* file. In the case of dial-up connections, *dialer-token* contains the modem type used or direct for direct connections.

Example:

Direct connection:

Direct/dev/term/ppp001 - 9600 direct

Dial-up connection: (modem type v32bis)

ACU /dev/term/ppp001 - 9600 v32bis

Dialers

The *Dialers* file must contain the initialization/dialing commands required for the modems used. There is a blank entry for direct connections.

Example:

Direct connection:

direct "" "" \M

Modem connection: (v32bis)

v32bis =,-, "" \M\dAT&F\r\c OK\r ATQ2&W\r\c
OK\r ATZ\r\c OK\r ATM1\r\c OK\r AT&D2\r\c
OK\r AT&C1\r\c OK\r ATS0=1\r\c OK\r AT&E4\r\c
OK\r AT&E2\r\c OK\r \EATDP\T\r\c 00 \r\m\c

The ppd.config file

Entries have to be made in the *ppd.config* file (*/usr/lib/ppp/ppd.config*) for each protocol used and each server. The syntax and the possible parameters are described in the Section "The structure of the ppd.config file". Make sure that dialout authorization is entered for the configured remote machine, in this case neptun (i.e. the client may dial the remote machine)

Minimum configuration example:

```
protocol ipcp
addr 129.22.4.59 # Local Internet address of the client
end
host   neptun         # The name of the server
protocol ipcp
dialout        # Dialout authorization
end
```

10.6.2 The server (passive end)

Modem presets

With modem connections, unlike direct connections (null modem cable), the modem must be configured so that it can receive incoming calls. The default port monitor *ttymon* does not have the call accept characteristic. The modem settings can be loaded using the *cu* command if there is no other utility available. The AT command sequence for the modem type contained in the *.../Dialers* file is transferred to the modem. Because ECHO mode must be deactivated due to the PPP, the *cu* command does not receive an OK; it then repeats the load procedure and then terminates. Even if *cu* outputs a large number of protocol records during loading, the modem is still loaded correctly and is not affected by power-down procedures.

Example for multi-modems (e.g. MT2834ZDXG)

Call:
cu -d modemconf

Mandatory entries in the *uucp* files */etc/uucp/...*:
../Systems:□modemconf□□□□Any□□modconf□□38400□□-□□□"" \r

../Devices:□modconf□□□□□□/dev/term/hios41,M□□□-□□□38400□□□mconf

../Dialers:□mconf =,-,"" \M\dAT&FX3\r\c OK\r AT&E2&E5&E11&SO=1

(continuation of line)□□\r\c OK\r AT$SB38400Q1&WO\r

You can find a description of the syntax in the header of the *.../Dialers* file or in the Section "The Dialers file").

Serial interface settings

In order to avoid loopbacks (i.e. the PPP receives the data it itself has sent via the *echo* function), it is necessary to set up a separate tty label for PPP (entry in */etc/ttydefs*). To do this, use the already existing generic entry, duplicate it, replace generic with ppp, for example, and add -echo to both parts of the entry (the separator is : in the middle). Give this label a new name. The new entry in the */etc/ttydefs* file is, for example:
ppp:9600 cs8 -parenb -ixany -clocal cread -istrip -echo :9600 cs8 -parenb -ixany -clocal cread -istrip opost ixon

-ixoff onlcr -echo echoe echok icanon isig hupcl ::ppp

Setting up the device nodes

Always set up a separate (modem-capable) device node for PPP. Use the *sysadm* or *termadd* commands in order to do this.

1. *sysadm*

   Select the menu items *configuration*, *serial devices*, *ACTIONS* to reach the submenu for setting up a device node. Select a name (e.g. /dev/term/ppp002).

   After you have set up the node, make sure that a corresponding port service is active (in this case,

/usr/bin/login). In general, only sleep 999999 is active after the node has been set up.

Select the menu items *ports*, *port_services*, *modify* to reach the submenu for modifying a port service. Select the corresponding service tag (in this case, ppp002) and modify the following entries:

Create 'utmp':	Yes (creates an entry in the */var/adm/utmpx* file)
ttylabel:	ppp
Service command:	/usr/bin/login
Hangup:	No
Connection-carrier:	No
Bidirectional:	Yes
Wait-read:	Yes
Wait-read-count:	0
Prompt:	Login:

2. *termadd*

   You can also set up device nodes directly using the *termadd* command.

   Device nodes set up using the *termadd* command cannot be administered using *sysadm*.

   Syntax:

   termadd -T *terminal_type* -n *node_name* -b *board*
   -l *ttylabel* -p *prompt* -t *tagname*

These terms have the following meanings:

terminal_type
Type of terminal. The values can be: generic, serial

node_name
Name of the node

board Board, interface to poll with *termadd -b?*

ttylabel
Entry in */etc/ttydefs* (in this case, ppp)

prompt
Login prompt (in this case, "login:")

tagname
Name of the service tag

Example:

```
RM600/SINIX-Y (Terminal2 CSI)
termadd -T generic -n ppp002 -b csi0,3 -l ppp \
-p "login:" -t ppp002
RM400/SINIX-N (Serial II)
termadd -T generic -n ppp002 -b motherboard0,5 -l ppp
-p "login:" -t ppp002
```

The command also activates the port service. However, you should check that the port service contains the values specified under point 1. (*sysadm*).

Setting up a new user

As has already been described, the client logs into the server as a normal user. This means it is necessary to set up a new user on the server. This user must have the login name and password used in the client's */etc/uucp/systems* file. You can set up a separate user for each PPP connection, or a global user for all PPP connections (i.e. each PPP connection uses the same login name and pairing password). To do this, use the *sysadm* or */usr/sbin/adduser* commands.

The user set up in this way does not need a home directory.

The ppd.config file

A machine entry must be set up for the client. Use the login ID of the client as the machine name in this case. The syntax and the possible parameters are described below and under *ppd(1)*in the "Networking Reference Manual". Make sure the configured clients have dialin authorization (i.e. they can dial the server).

Minimum configuration example:

```
host   ppp            # Login ID of the client
protocol  ipcp
dialin         # dialin authorization
addr 129.22.4.90 # Local Internet address of the server
end
```

10.6.3 The structure of the ppd.config file

Library pathname

The pathname of the directory is set by the following entry:

libdir *pathname*

This directory is where the PPP will find the dynamically loadable protocols (*pap.so*, *lqm.so*, *chap.so*). The default for *pathname* is */usr/lib/ppp*.

Machine entries

Machine entries have the following format:

host *hostname*
options
end

hostname is a keyword for the first field of an entry in the */etc/uucp/systems* file.

In the following, *flag* is a Boolean variable that can adopt the values on or true for "On" and off or false for "Off". The following options can be used to switch on a parameter:

option on
option true
option (without flag)

Options (*option*) can be used in the protocols specified under "Application" (see the list below) or in the machine entry. Specifications within a protocol entry are valid for all connections. Specifications in the machine entry only apply on a connection-specific basis. The following options are defined:

device	*host*	*host* is used instead of *hostname* as keyword in the *systems* file. (Application: machine entry only, if not already defined for host *hostname*)
dialin	*flag*	Depending on *flag* (on/off), an incoming call is accepted. (Application: machine entry only, as it is connection-specific)
dialout	*flag*	Depending on *flag* (on/off), dialing is possible. Application machine entry only, as it is connection-specific)
timeout	*secs*	The (physical) connection to the partner is broken if no PPP packets are sent or received for *secs* seconds. This parameter is needed for demand dialing (see Section "Demand dialing"). (Application: machine entry or LCP protocol)
protocol	*pname*	The specified monitoring protocol (LCP, IPCP, PAP, CHAP, LQM) should be used. The LCP and IPCP protocols should only be specified if other protocol-specific options are required. These protocols are defined implicitly. (Application: PAP, CHAP, LQM in the machine entry or LCP protocol)
debug		Writes debugging information to *stdout* (client) or to the */tmp/ppd.trace* file (server). (Application: machine entry)
trace		Writes trace information to *stdout* (client) or to the */tmp/ppd.trace* file (server). (Application: machine entry)

Please refer to the Section "Minimum configuration example:" for an example of a minimal *ppd.config* file.

The protocols

Currently, the following protocols are implemented:

LCP Link Control Protocol

IPCP Internet Protocol Control Protocol

LQM Link Quality Monitoring

PAP Password Authentication Protocol

CHAP Challenge Handshake Authentication Protocol

Protocol entries have the following format:

protocol *protocol_name*
options
end

protocol_name is the name of the protocol to be configured. In addition to the protocol-specific parameters (see Section "Protocol-specific parameters:"), *options* can adopt the following non-protocol-specific values:

active
: The protocol is operated in active mode. ACTIVE is the opposite to PASSIVE. If this flag is set, a client starts the protocol directly when an outgoing connection is established and begins sending packets to the partner for this protocol.

passive
: The protocol is operated in passive mode. PASSIVE is the opposite to ACTIVE. If this flag is set, a server (incoming connection) only starts the protocol when the partner sends a packet for this protocol.

initial
: The protocol is started immediately after LCP. (This is useful for authentication protocols.)

!initial
: The protocol is not started until a corresponding packet has been received. (This is useful, for example, if the server does not know whether the client also supports the corresponding protocol.)

disabled
: The configured protocol is not used.

library *lib*

Only for the PAP, CHAP and LQM protocols.

lib is the name of the corresponding protocol library which is dynamically loaded (*pap.so*, *chap.so*, *lqm.so*). *lib* can be an absolute pathname or a relative name (relative to */usr/lib/ppp*). (The LCP and IPCP protocols are permanently loaded.)

If no protocol-specific parameters are used, protocol entries can also take the following form:

protocol *protocol_name lib*

(This only makes sense for CHAP, PAP and LQM, because LCP and IPCP are loaded permanently.) *protocol_name* is the name of the protocol, and *lib* is the name of the protocol library (*pap.so, chap.so*, *lqm.so*).

Example:

protocol lqm lqm.so

Protocol-specific parameters:

LCP

The LCP protocol is always the first protocol to be started, and its purpose is to establish the logical connection. For example, the partner is informed which additional protocols should be used. The LCP protocol is always used, so it is permanently loaded. The parameters executed in this packet can also be executed on a connection-specific basis in the machine entry.

mtu *size* Maximum transmitted packet size (in bytes). *size* can adopt values between 100 and 1548 (default: 1500). If the set value is too high for the partner, PPP reduces the value step by step until an acceptable setting has been reached.

The following parameters *charmap, rvcmap, sndmap* are only significant if a partner has an old PPP implementation in accordance with RFC1331. The parameters can be ignored in current PPP implementations.

charmap *map* The characters specified in *map* and the corresponding characters with bit 7 set are displayed at both the sending and receiving ends.

The situation may arise that the serial driver below the PPP interprets some of the ASCII characters in the 0x00-0x1f range (e.g. DC1 = Control-Q or DC3 = Control-S). Modems can also interpret the corresponding characters with bit 7 set, e.g. 0x93 as Control-S with parity. In order to avoid this, all ASCII characters in this range can be mapped to other values.

This mapping involves conversion to an escape sequence. In each case, *map* can be a 32-bit long hexadecimal number (prefix 0x), an octal number (prefix 0) or a decimal number (no prefix). Bit 0 (the one with the lowest value) corresponds to ASCII 0, bit 1 corresponds to ASCII 1, ..., and bit 31 (the one with the highest value) corresponds to ASCII 31.

If a bit is set, this means the corresponding ASCII character is mapped. A deleted bit means the corresponding ASCII character is not mapped. HUHU*map* can also be a string, in which case the characters represented by the control key are given: @ for ASCII 0 = Control-@, S for ASCII 19 = Control-S etc. For example, you can specify the values

		0x000a0000, 00002400000, 655360 (bit 17 and bit 19 set) or the string QS for *map* in order to map the characters Control-Q (ASCII 17) and Control-S (ASCII 19).Default: charmap 0xffffffff, i.e. all characters are mapped.
rcvmap	*map*	The characters specified in *map* are mapped when they are received (see charmap). Default: 0xffffffff, i.e. all characters are mapped.
sndmap	*map*	The characters specified in *map* are mapped when they are sent (see charmap). Default: 0xffffffff, i.e. all characters are mapped.
pfc	*flag*	The protocol field is compressed (default off). It is advisable to switch this on at low transmission speeds, because it can allow the protocol field to be shortened from 2 bytes to 1 byte.
acfc	*flag*	The address and control fields are compressed (default off). It is advisable to switch this on at low transmission speeds.
rcvpfc, rcvacfc		These two boolean switches control whether PFC and ACFC options are accepted and whether corresponding compressions are to be performed (any compressed packets received are always handled correctly even if it has been negotiated that these should not be sent).

rcvpfc and rcvacfc are deactivated by default.

auth	*id*	The authentication protocol *id* (PAP or CHAP) is to be used. The parameters relevant to PAP or CHAP are defined in a separate protocol entry.
quality	*id time*	The specified link quality protocol *id* (LQM) should be used. *time* is the time interval (in 1/100th of a second) at which the LQM packets are exchanged.

Example:

```
protocol lcp
charmap "QS"
pfc
acfc
end
```

IPCP

IPCP is permanently loaded because only one network protocol (IP) is supported at present. The parameters executed in this packet can also be executed on a connection-specific basis in the machine entry.

vjc	*flag*	This boolean switch activates transmission of Van-Jacobson-compressed TCP frames. This is particularly advisable for slow connections because the 40-byte long standard TCP/IP header is shortened to 4 bytes through the compression (default: off).
recvvjc	*flag*	Van Jacobson-compressed TCP headers can be received (default: on)
addr	*ipaddr*	Informs the partner what the local Internet address

		is. PPP requests a value for *ipaddr* directly from the partner if no value is specified for it.
remoteaddr	*ipaddr*	The Internet address of the partner. This is required if the partner does not communicate its own Internet address or if demand dialing is used (see Demand dialing).

If IP addresses are defined twice (*addr* and *remoteaddr* parameters) they are rejected: NAK, not accepted. In this case, the IP address configuration must be corrected on one of the two sides of the PP connection being established.

Example:

```
protocol ipcp
vjc
addr 129.22.4.90
end
```

LQM

The LQM protocol is used within PPP in order to check the quality of the connection. Consequently, statistical information is transmitted every *time*/100 seconds. This information includes, among other things, the number of bytes or packets which have been transmitted. PPP breaks off the connection to the target machine if too many packets have got lost (e.g. due to disturbance on the line). If the value of *time* is 0, this means an LQM packet is only sent when an LQM packet has been received. The parameters executed in this packet can also be executed on a connection-specific basis in the machine entry or in the LCP.

The control-specific parameter *time*, which belongs to LQM, is specified in the following format as an entry in the LCP protocol configuration:

quality lqm *time*

Do not use the LQM protocol in conjunction with demand dialing. The PPP connection may never be broken if you select too short an interval for *time* (see Section "Demand dialing").

Example:

LQM packets are to be exchanged every 10 minutes (= 60000/100 seconds). The Internet addresses are not specified in the IPCP entries in this example; instead, they are specified in the machine entries.

```
Client                          Server
libdir /usr/lib/ppp             libdir /usr/lib/ppp
protocol lqm lqm.so             protocol lqm lqm.so
protocol lcp                    protocol lcp
charmap "QS"                    charmap "QS"
acfc                            acfc
pfc                             pfc
quality lqm 60000               quality lqm
end                             end
protocol ipcp                   protocol ipcp
vjc                             vjc
end                             end
host neptun                     host ppp
addr 129.22.4.59                addr 129.22.4.90
protocol ipcp                   protocol ipcp
dialout                         dialin
end                             end
```

PAP

PAP is a straightforward authentication protocol. The server requests a login ID/password pairing from the client. In contrast to CHAP, authentication only takes place when the connection is established. A user (login ID: *name*, password: *pass*) must be set up on the server in order for the PAP protocol to be used. The server sends a PAP request to the client, to which the client must respond with the corresponding *name*, *pass*. The server clears the connection if the authentication process is unsuccessful several times.

The PAP protocol is always started after the LCP protocol. The network checking protocol (IPCP) is only started after successful authentication. While the authentication of received IPCP packets is discarded, the connection is not canceled. The parameters executed in this packet can also be executed on a connection-specific basis in the machine entry.

PAP sends the login ID and password over the line in uncoded form. For this reason, the CHAP protocol should be used for more secure authentication.

paphost	*name*	This is only significant for the client. *name* must be a valid login ID on the server.
passwd	*pass*	This is only significant for the client. *pass* is the password associated with *name*.

Example:

A user has been set up on the server with a login ID hugo and password blablabla. The PAP protocol is configured on the server as initial (i.e. it is started immediately after the LCP protocol). The IPCP protocol is configured as !initial (i.e. IPCP is not started until the client sends an IPCP packet). The auth pap parameter was specified in the machine entries rather than in the LCP entries.

```
Client                  Server
libdir /usr/lib/ppp     libdir /usr/lib/ppp
protocol pap            protocol pap
library pap.so          library pap.so
paphost hugo            initial
passwd blablabla        end
```

```
end
protocol lcp                protocol lcp
charmap "QS"                    charmap "QS"
acfc                    acfc
pfc                     pfc
end                      end
protocol ipcp                   protocol ipcp
vjc                     vjc
end                             !initial
end
host neptun                 host ppp
addr 129.22.4.59            addr 129.22.4.90
protocol ipcp             protocol ipcp
auth pap                  auth pap
dialout                 dialin
end                       end
```

CHAP

CHAP is an authentication protocol. The server sends the parameters of a one-way hash function to the client together with the name of a "secret file". The client encodes the contents of the corresponding "secret file" using this data and sends the result to the server. The server in turn encodes the contents of its "secret file" using the data sent to the client. It breaks off the connection if the result does not comply with the result sent from the client. In contrast to PAP, authentication is performed at periodic intervals. A file called *secrets* must be in the */usr/lib/ppp/secrets* directory of both the client and the server in order for the CHAP protocol to be used. This file contains a string (identical on both server and client and of any length). This string is used for encoding.

The CHAP protocol is always started after the LCP protocol. The network checking protocol (IPCP) is only started after successful authentication. While the authentication of received IPCP packets is discarded, the connection is not canceled. The parameters executed in this packet can also be executed on a connection-specific basis in the machine entry or in the LCP.

The superuser ID root must be the owner of the *secrets* file, and the file must have file mode 0400 (exclusive read permission for the owner) (see the following example). The CHAP protocol is not started unless these conditions are fulfilled.

chaphost *secret-file*

Name of the file in the *chap directory*, which contains the secret key. If no value is specified for *secret-file*, PPP uses its own node name.

The value of the access rights must be set to 0400 for this file.

secrets *chap directory*

Name of the directory containing the key files.

The value of the access rights must be set to 0700 for this directory.

/usr/lib/ppp/secrets is the default directory.

secret_term_lf

If this parameter is specified, the last line feed character (LF) of the "secret" (contents of *secret-file*) is not interpreted; in other words it is as if this character does not belong to the key.

This parameter may be necessary in order to exchange a "secret" with remote systems.

Example:

A file called erwin has been created in the */usr/lib/ppp/secrets* directory on both the client and the server. Each erwin file contains the following string:

We don't have a chance but we'll grab any chance we get.

The access permissions are as follows:

-r-------- 1 root other 45 Nov 22 13:18 erwin

The CHAP protocol has been configured as initial on the sever because it should be started immediately after the LCP protocol. The IPCP protocol has been configured as !initial. In addition, LQM packets should be exchanged every 10 minutes. The *auth chap* parameter has been specified in the machine entries rather than the LCP entries.

```
Client                    Server
libdir /usr/lib/ppp       libdir /usr/lib/ppp
protocol lqm lqm.so       protocol lqm lqm.so
protocol chap             protocol chap
library chap.so           library chap.so
end                       initial
end
protocol lcp              protocol lcp
charmap "QS"              charmap "QS"
acfc                      acfc
pfc                       pfc
quality lqm 60000         quality lqm
end                       end
protocol ipcp             protocol ipcp
addr 129.22.4.59          addr 129.22.4.90
vjc                       vjc
end                       !initial
end
host neptun               host ppp
protocol ipcp             protocol ipcp
auth chap                 auth chap
dialout                   chaphost erwin
end                       dialin
end
```

10.6.4 Establishing the connection

The connection is established from the client using the *ppd* command (see also the Networking Reference Manual and the manual pages for PPD, which contain a short description of PPP).

ppd *hostname*

where *hostname* is a valid entry in the */usr/lib/ppp/ppd.config* file.

Example:

```
ppd neptun
```

10.6.5 Clearing the connection

The (physical) connection can be cleared from the client or from the server by calling the *ppp_off* command. You then receive a table of all PPP connections with the following format:

```
Login-ID/Server-Name  Remote IP-Address   PID
name                  addr                pid
Enter Process-ID (PID) of PPP connection to terminate:
```

In this table, the first column is the login ID of the client or the name of the server, the second column is the Internet address of the target machine, and the third column is the process ID of the associated *ppd* daemon. Enter the process ID of the corresponding PPP connection. This terminates the *ppd* processes associated with this connection on both the client and the server, thereby terminating the PPP connection normally.

If you only have one PPP connection, you can also clear it by sending the SIGTERM (kill -15) signal to the

corresponding *ppd* process on either the client or the server.

Example

The process ID of the *ppd* process on the server is 13451.

The following command causes the *ppd* processes on the server (and consequently also on the client) to be terminated:

kill -15 13451

The PPP connection is then terminated normally.

10.6.6 Demand dialing

In some applications, it makes sense not to establish a PPP connection until a TCP/IP connection is used (e.g. *rlogin*, *telnet*, ...).

This is done by calling the *ppd* command with the *-W* option. A routing entry is made, but no connection to the server is established. Following this, PPP waits for packets that are to be sent. The *-W* option can only be used on the client.

Syntax:

ppd -W *hostname*

where *hostname* is a valid entry in the */usr/lib/ppp/ppd.config* file.

If the *ppd* command is called using the *-W* option, both the local Internet address and the Internet address of the corresponding server must be specified in the *ppd.config* file (see the following example).

The following parameter can be entered in the *ppd.config* file in the corresponding machine entry:

timeout *time*

If this is done, the connection to the server is cleared if no PPP packets are transmitted over the line within *time* seconds.

This takes into account PPP packets (i.e. and LQM packets, for example), not just IP packets. In particular, use of the LQM protocol can mean that the connection to the server is never terminated (if, for example, the interval at which LQM packets are sent is less than the value of *time*), or the server is periodically selected despite the fact that the user does not make any entry.

If the CHAP protocol is used and the timeout value is greater than the retry rate (120 sec), the retry rate of the CHAP protocol is adapted on the client side to the timeout value in order to ensure that the PPP connection is cleared down correctly following a timeout.

However, the server normally requests CHAP authorization, which means that the timeout value of the client must be greater than 120 seconds. The client cannot influence the CHAP retry rate of the server.

Example:

The connection to the server is to be cleared if no PPP packets are sent or received for a period of 30 seconds.

Client
□□
Server

```
protocol lcp            protocol lcp
charmap "QS"            charmap "QS"
acfc                    acfc
pfc                     pfc
end                     end
protocol ipcp           protocol ipcp
```

```
vjc                          vjc
end                          end
host neptun                  host ppp
addr 129.22.4.59 1)          protocol ipcp
remoteaddr 129.22.4.90 2)    dialin
protocol ipcp                end
timeout 30
dialout
end
1) Local Internet Address    2) Server Internet Address
```

The following command is called on the client (client% is the shell prompt on the client):

client% ppd -W neptun

The routing table then has the following contents, for example:

```
client% netstat -nr
Routing tables
Destination  Gateway     Flags  Refcnt  Use  Interface
127.0.0.1    127.0.0.1   UH     0       0    lo0
129.22.4.90  129.22.4.59 UH     0       0    ppp0
```

No PPP daemon (*ppd*) is as yet running on the server, and there is also as yet no physical connection to the server.

When an application is started that must first establish a connection to the server (e.g. *ftp*, *rlogin*, *telnet*), the server may take a little time to react. Dialing and establishing the connection takes about 20-40 seconds.

Example:

```
client%ftp 129.22.4.90
```

Applications which are running when the connection to the server is terminated after *time* seconds are not affected by this.

Only the *ppd* daemon on the server is terminated. When there is a new input from the client, it takes a little time for the corresponding application to respond to the input (the time it takes to dial the server).

In order to avoid TCP/IP timeouts as a result of slow PPP connection setup times, it is advisable to increase the *keepinit* value as follows:

```
<sysctl -n net.inet.tcp.keepinit>        Read timeout
<sysctl -w net.inet.tcp.keepinit=waitx>  Set timeout
```

e.g.:

```
sysctl -w net.inet.tcp.keepinit=300
```

10.6.7 Routing with PPP

More than one PPP interface can be configured on one machine, so it is also possible to create large networks with PPP connections. Routing via PPP connections offers the same functionality as routing via SLIP connections, and the configuration procedure is the same (see the section entitled Routing with SLIP).

10.6.8 Problems and how to deal with them

If you make an error during complex configuration and the PPP connection is not established, call the *ppd* command again using the *-T* option and possibly the *-D* option (only possible on the client) or insert the keyword debug or trace in the host instruction (see the Section "The structure of the ppd.config file"). This gives you more detailed information on the activities of the PPP.

PPP-specific messages are written to the */tmp/ppd.trace.account* file on the server.

account is the name of the id used by the client to log on to the server. Server trace files are connection-specific.

Messages are written to *stderr* on the client.

Syntax:

ppd -T *hostname*(Trace Mode)
ppd -T -D *hostname*(Verbose Mode)
ppd -T -D -D *hostname*(Full Debug Mode)

These are the most frequently occurring messages and their causes:

Message:

Cannot dial *host*, line problem (open() failure)

Cause:

The server cannot be dialed.

- There is something wrong with the connection to the server.
- Either no port service or no service command has been configured (see Setting up the device nodes).
- The device nodes have been set up incorrectly (see Setting up the device nodes).
- Client and server use different baud rates (with direct connections).

Message:

Cannot dial host, system not in Systems file

Cause:

Error in the UUCP *config* files

- There is no entry in the */etc/uucp/Systems* file for the specified machine (see The client (active end)).

Message:

Caught signal poll

Cause:

The partner's *ppd* daemon terminated as a result of an error, or it was not even started by the *login* process.

- An incorrect tty label was used (see Serial interface settings).
- Access permissions for the */opt/bin/ppd* daemon were changed on the server.
- The client's login ID or password were not correct (see The client (active end)).
- On the server, the line protocol ipcp is missing from the */usr/lib/ppd.config* file in the configuration entry for the client.

Message:

Cannot link stream to multiplexor errno 22

Cause:

Error creating the device nodes

- No entry was made in the */var/adm/utmpx* file on the server (see Setting up the device nodes).

Message:

connection timeout (re-establishing a connection)

Cause:

The PPP connection is taking longer to establish than the 75 second TCP connection timeout:

- modem dialing is taking too long (pulse dialing, pauses),
- because of a bad line or incompatible protocols, it is taking too long to negotiate the speed with the partner modem,
- because of an incorrect configuration, it is taking too long to negotiate the PPP connection protocol.

If the time cannot be reduced to below 75 seconds, there is the option of increasing the TCP connection timeout with the tuning parameter TCPTV_KEEP_INIT or the *sysctl* variable *net.inet.tcp.keepinit*.

Another reason why it may be impossible to set up a connection is if the password of the ID which is calling the *ppd* has expired.

10.7 MTU (maximum transmission unit)

The MTU for a network type is defined as the maximum length of an IP datagram, including the IP header, which can be transported as one data block over the network. Depending on the LAN type, this limit is either derived from the physical limits of the transmission medium or is specified as the logical upper limit in a standard (RFC).

Unfortunately, (smaller) MTU sizes deviating from those of other standards have come into widespread use, in Token Ring, for example. In order to be able to communicate with or via such components (e.g. bridges), the local LAN interface must be adapted to the smaller MTU size.

For the LAN interfaces used by Reliant UNIX V5.45, the default value or quasi-default value is set as the MTU size. However, some of these values can be altered by means of global kernel variables in order to allow communication even in networks with different MTU sizes, when necessary.

Ethernet

The MTU size of Ethernet based on RFC894 (Xerox protocol) is 1500 bytes. This value is specified by the drivers. When Ethernet is used with the IEEE 802.3 protocol, from an IP viewpoint, the length of the LLC(3)/SNAP(5) header (according to RFC 1042) - i.e. a total of 8 bytes - has to be subtracted from those 1500 bytes.

This is not necessary when the Xerox protocol (the default) is used.

Gigabit Ethernet

The same generally applies for Gigabit Ethernet (JUMBO_N in *etherstat -v*) as for Ethernet. The Alteon controller available for RM600-E and RM400 also supports jumbo frames (JUMBO_Y). In this case, the MTU is 9000 bytes for operation with a Xerox protocol. It is not possible to operate an interface, which has been configured for Jumbo frames, with the IEEE 802.3 protocol. The configuration of jumbo frames is described in the Manual Pages for *alt(5)*.

A controller must be configured for jumbo frames in order to receive them. Special network switches are also required for operating a network with jumbo frames. The performance of the network is considerably improved when using jumbo frames instead of standard frames.

Token Ring/4Mb

The MTU size of Token Ring/4Mb based on RFC 1042 is 4464 bytes. This already takes into account that an IP packet of this size is preceded by an LLC/SNAP header of 8 bytes and a MAC header of up to 32 bytes (Access Control 1, Frame Control 1, Source Address 6, Destination Address 6, Routing Information Field 0-18).

Implementations of Token Ring/4Mb are known which only allow an MTU size of 2002 bytes. If an RM400/RM600 is used in a ring together with these machine types, the local MTU size can be adapted by setting app_token_mtu=2002 in */etc/conf/pack.d/arp/space.c* (see Section "").

Token Ring/16Mb

The MTU size of Token Ring/16Mb specified in RFC 1191 is 17914 bytes.

An MTU size of 8136 bytes is set, since the widely used CISCO router operates with this MTU size. Here too, it is possible to set a different MTU size using *app_token_mtu* in */etc/conf/pack.d/arp/space.c*. In the case of the RM600 LCT, note that the Token Ring HW does not support MTU sizes above 8227 bytes. The HW limit on the Madge controller is 17792.

FDDI

The MTU size of FDDI is set to 4352 bytes.

This takes into account that an IP packet of this size is preceded by an LLC/SNAP header of 8 bytes and a MAC header of 16 bytes (Preamble 2, Starting Delimiter 1, Frame Control 1, Destination Address 6, Source Address 6) and succeeded by a MAC trailer of 6 bytes (FCS 4, ED 1, FS 1).

The FDDI MTU size can be modified if necessary by means of the *app_fddi_mtu* variable in */etc/conf/pack.d/arp/space.c*.

The currently set MTU sizes for the LAN interfaces can be requested using the *netstat -i* command.

The MTU sizes set apply to all interfaces of the corresponding type of LAN. With Token Ring on the RM600 (LCT), the loadware for a controller can be reconfigured as an alternative to this, so that the changed MTU size applies only to this controller. Whichever value is the lower out of the controller MTU and the value from *app_token_mtu* is taken as the MTU size for an interface.

The loadware is reconfigured for LCF in the CMX menu for controller configuration. This is described in the CMX documentation.

SLIP and PPP

The MTU can be specified by the superuser. (Please refer to the corresponding description as in the sections dealing with SLIP and PPP.)

Summary:

Network type	MTU	Standard
Ethernet Xerox	1500	RFC894
Ethernet 802.3	1492	RFC1042
Token Ring/4Mb	4464	RFC1042
Token Ring/16Mb	8136	RFC1042 due to CISCO router
FDDI	4352	RFC1188
SLIP	Variable	
PPP	Variable	

11 Changing system limits and system parameters

This chapter describes the most important configuration parameters of the network software of the RM systems. These can be set by the system administrator. Some parameters allow you to change certain system limits, and others allow you to change the system's behavior.

There are four types of parameters:

1. Tuning parameters, which can be set with */etc/conf/bin/idtune* within predefined limits. On systems with the "enhanced" protocol stack the *sysctl* interface can also be used.
2. Variables, which can be changed in the *space.c* file of the relevant software packages
3. Global kernel variables, which can be changed with *crash* in *rc* scripts.
4. Route-specific parameters, which can be changed on systems with the "enhanced" protocol stack using the *eroute change* command.

In the case of parameters changed with *idtune* and in the case of 2, new kernel must then be created using the command */etc/conf/bin/idbuild*. The new kernel does not become effective until the next system startup. Changes made to systems with the "enhanced" protocol stack via the *sysctl* interface only affect the active system. To make such changes permanent in the sense that they are repeated for each system startup, the relevant *sysctl* commands can be entered in the */etc/inet/rc.sysctl.local* file.

In the case of 3, the system administrator can change kernel variables for the active system with *crash*. A change of this type only remains valid until the next system startup. Permanent changes to the kernel variables must be made in the *rc* scripts. Caution should be exercised when changing kernel variables.

In the case of 3, the change is made in the kernel's object file. If the change is made to the active system (patch), it takes effect immediately; if it is made to the */unix* file, it becomes effective at the next system startup. Caution should be exercised when changing kernel parameters.

In the case of 4, changes can be made to the active system, which will take effect the next time the modified route entry is used, e.g. when establishing a new connection.

A list of all the system parameters which can be set for a system with the "enhanced" protocol stack via the *sysctl* interface can be found in the */etc/inet/rc.sysctl.readme* file. This file also contains the *net.inet.** parameters, which are not referred to here.

11.1 Tuning parameters

MAXNETUSERS
Meaning:

Number of users for network utilities

A number of other system parameters are set that depend on MAXNETUSERS.

Default:

MAXUSERS

(i.e. the maximum number of users of this system)

NUMSOM
Meaning:

Specifies the number of *sockmod* control structures.

These are generally required by the socket interface. A *sockmod* is required for each open connection.

Default:

(MAXNETUSERS * 5) + 96

NUMTIM
Meaning:

Specifies the number of *timod* control structures.

These are generally used by the *tli/xti* interface. A *timod* is required for each connection.

Default:

Actual: (MAXNETUSERS * 4) + 64

Target: (MAXNETUSERS * 26) + 64

Relmail:

A0441017

NUMTRW
Meaning:
Specifies the number of *tirdwr* control structures.

These are generally required by the *tli*/*xti* interface if functions of the input/output interface (*read*/*write*) are to be used.

Default:
MAXNETUSERS + 16

NUDP
Meaning:
Specifies the number of possible minor device numbers on the UDP streams multiplexer.

This is the same as the number of *open()* calls that can be executed on */dev/udp*. This value therefore specifies the maximum number of UDP connections (via sockets, etc.).

This tuning parameter only affects streams socket implementation in the kernel.

This parameter is valid globally on systems with the "fallback" protocol stack. On systems with the "enhanced" protocol stack the parameter is only valid for TPI endpoints.

Default:
2*MAXNETUSERS + 32

ARPTAB_SIZE (only for systems with the "fallback" protocol stack)
Meaning:
The ARP table is a two-dimensional table of the size ARPTAB_BSIZ * ARPTAB_NB (see the next parameter).

This determines the number of possible ARP entries on a system at any one time.

ARPTAB_NB:
Number of lines in the table.

ARPTAB_BSIZ:
Number of entries in a line.

ARPTAB_SIZE
is the maximum number of entries to be taken into account when searching for an element in the table.

Default:
18

ARPTAB_NB (only for systems with the "fallback" protocol stack)
Meaning:
See ARPTAB_SIZE.

ARPTAB_NB should be a prime number so that the entries can be distributed evenly in the ARP table and individual rows of the table are not given an unnecessarily heavy load.

Default:
41

MAXDUPREQS
Meaning:
This parameter specifies the number of entries in the NFS duplicate cache. The NFS duplicate cache is used to recognize repeated sends for NFS jobs that the server has already carried out.

In the event of a repeated send, the NFS server must avoid repeating the execution of any jobs which are "non-idempotent" (such as deleting a file); instead it must send the client the result of the first execution. Non-idempotent jobs and the corresponding results are therefore stored in the NFS duplicate cache.

Default:
4000

NCSIZE
Meaning:
The value of this parameter specifies the size of the Directory Name Lookup Cache (DNLC) in Kbytes. The size of the DNLC is important within the file systems of a system for compiling the names into the internal kernel *vnode* structures. The value of NCSIZE depends on the value of NPROC, which determines the maximum number of processes configured for the system.

The minimum value for NCSIZE is 100, the maximum value is 4096.

Default:
(NPROC+100<2048)*(NPROC+100)+(NPROC+100>=2048)*2048

SM_SIZE
Meaning:
The value of this parameter specifies the percentage of physically available memory that may be used for buffering file input/output.

Minimum 5 percent, maximum 100 percent.

Default:
30

11.1.1 TCP parameters

SOMAXCONN
Meaning:
The value of this parameter determines the maximum number of "attached" TCP connections. This parameter has the same effect as the *sysctl* variable *kern.somaxconn*.

This value defines an upper limit specifying how many incoming connection requests for a specific service can be accepted before the relevant *accept(3N)* calls are terminated and the new connection is decoupled by the current server.

Minimum is 8, maximum is 1024.

Default:
Actual: 128

Target: 500

Relmail:
somaxconn

NTCP
Meaning:
The number of possible minor device numbers of the TCP streams multiplexer.

This is the same as the number of *open()* calls that can be executed on */dev/tcp*. This value is made up of the maximum number of TCP connections (via sockets, etc.) plus the number of TCP ports waiting for connections (e.g. with a *listen* call when using sockets). This tuning parameter only affects streams socket implementation in the kernel.

The parameter is valid globally on systems with the "fallback" protocol stack. On systems with the "enhanced" protocol stack, the parameter is only valid for TPI endpoints.

Default:
(MAXNETUSERS * 4) + 64

TCPNODELACK
Meaning:
This variable controls the acknowledgment behavior of TCP. It has the same effect as the *sysctl* variable *net.inet.tcp.nodelack*.

If this parameter has the value 0, then there will be delays of up to 200 ms on acknowledging (ACK) received data packets if you have no data of your own to send. If this parameter is set to a non-zero

value, then each packet will be acknowledged as soon as it is received. Please note that this circumvents the "Nagle" algorithm on the attached remote machines; this algorithm is urgently recommended for TCP.

For further details please refer to the titles listed under "Publications for more Information".

Default:

0

TCPACKONPUSH
Meaning:

Activates/deactivates ACK on PUSH behavior for immediate TCP acknowledgment.

If the PUSH bit is set in the TCP header of a data packet, an acknowledgment is sent as soon as it is received. This behavior partly circumvents the delay strategy of TCP as well as the Nagle algorithm on the receiving side and can lead to an increase in data traffic in the network.

0: locks immediate acknowledgment
1: releases immediate acknowledgment

TCPACKONPUSH has the same effect as the *sysctl* variable *net.inet.tcp.ackonpush*.

Default:

0

TCPTV_KEEP_INIT
Meaning:

Number of half-seconds permitted before a connection being established is aborted. This parameter has the same effect as the *sysctl* variable *net.inet.tcp.keepinit*.

Minimum is 150, maximum is 4000 half-seconds.

Default:

150

TCPTV_KEEP_IDLE_SECS
Meaning:

Interval in seconds after which TCP sends keep-alive packets to check whether the other partner is still active when no more data is transmitted via a TCP connection. The prerequisite for this is that the keepalive functionality is turned on for the application that set up the connection. This is done by means of the socket option, for example.
setsockopt(s,SOL_SOCKET,SO_KEEPALIVE,*opt_val*,sizeof(*int*))

Reliant UNIX systems send keep-alive packets by default (see also *tcp_compat_42*) without user data. This parameter has the same effect as the *sysctl* variable *net.inet.tcp.keepidle*.

See also TCPTV_KEEPINTVL_SECS.

Default:

(120*60)

TCPTV_KEEPINTVL_SECS

Meaning:

Interval between sent keep-alive packets. This parameter has the same effect as the *sysctl* variable *net.inet.tcp.keepintvl*.

Reliant UNIX systems send eight keepalive packets at intervals of TCPTV_KEEPINTVL_SEC seconds. If the other partner does not reply within this time, the connection is cleared, since it must be assumed that the partner machine has crashed, the physical connection has been interrupted or something similar. If a reply is made to one of the packets, the connection is regarded as "still active". This means that the KEEPALIVE is reset to TCPTV_KEEP_IDLE_SECS (see the previous parameter).

Default:

75 seconds

tcp_sendspace (only for systems with the "enhanced" protocol stack)

tcp_recvspace

Meaning:

These global parameters define the lower limit for the send and receive space size of a TCP connection in bytes. The values are system-wide defaults and can be checked and changed using *sysctl net.inet.tcp.sendspace* and *sysctl net.inet.tcp.recvspace*.

Default:

tcp_sendspace = 32768

tcp_recvspace = 32768

tcp_sendfactor (only for systems with the "enhanced" protocol stack)
tcp_recvfactor
Meaning:

The current send and receive space size (*sendpipe* and *recvpipe*) for an interface is calculated based on the corresponding maximum transmission unit size (*mtu*) and the value of *tcp_sendfactor* or *tcp_recvfactor* and is then increased by the value for the lower limit (*tcp_sendspace* or *tcp_recvspace*).
sendpipe=MIN(*tcp_sendfactor*mtu*, *tcp_sendspace*)

recvpipe=MIN(*tcp_recvfactor*mtu*, *tcp_recvspace*)

The values are system-wide defaults and can be checked and changed using *sysctl net.inet.tcp.sendfactor* and *sysctl net.inet.tcp.recvfactor*.

Default:

tcp_sendfactor = 12
tcp_recvfactor = 12

tcp_mssdflt (only for systems with the "enhanced" protocol stack)
Meaning:

The maximum segment size (*mss*) in bytes for connections that are not local (default is *TCP_MSS*). The value is a system-wide default and can be checked and changed using *sysctl net.inet.tcp.mssdflt*.

Default:

512

11.1.2 IP parameters

BEST_MATCH_ROUTING (only for systems with the "fallback" protocol stack)
Meaning:

Activation (=1) and deactivation (=0) of the "best match" routing algorithm (see also the Section "Routing with subnets").

The first value returned with the following command shows the current parameter setting:

grep BEST_MATCH_ROUTING /etc/conf/cf.d/?tune

Default:

BEST_MATCH_ROUTING=1 (i.e. the "best match" routing algorithm is activated)

rtq_reallyold (only for systems with the "enhanced" protocol stack)
Meaning:

This parameter specifies the number of seconds that elapse before a non-referenced route is removed from the routing tree structure. The value is a system-wide default and can be checked and changed using *sysctl net.inet.ip.rtexpire*.

Default:

3600

IPFORWARDING
Meaning:

This parameter specifies whether the machine is to behave as a router (1) or not (0). If it is, IP packets that are not meant for this machine are forwarded. This parameter has the same effect as the *sysctl* variable *net.inet.ip.forwarding*.

Default:

0

i.e. this system does not forward IP packets.

IPSENDREDIRECTS
Meaning:

This parameter determines whether ICMP messages of the type ICMP_REDIRECT are generated and returned to the source of the sent packet.

If a local system is configured as a router, an incoming packet can be forwarded to another router in this network segment via the same interface through which it was received. IPSENDREDIRECTS determines whether the source of the packet is to be told to send the packets directly to the other router in future. In this case the local system generates an ICMP message of the type ICMP_REDIRECT and sends it back to the source of the packet (see also IPFORWARDING).

0: no messages are generated□
1: messages are generated

This parameter has the same effect as the *sysctl* variable *net.inet.ip.redirect*.

The messages should only be generated and returned if the source of the packet sends directly to the local system without specifying any options. They should not be generated and returned if the forwarding mechanism uses a default route or, if the route has already been modified with an ICMP_REDIRECT, if proxies are used.

Default:

0

IPCNT

Meaning:

Number of minor device numbers supported by the IP multiplexer */dev/ip*. The multiplexers TCP, UDP, ICMP and RAWIP each occupy one minor device number. In addition, a minor device number is required for each direct *open()* access on */dev/ip*. (There are a number of functions that execute *ioctl()* calls on */dev/ip* directly.)

This tuning parameter only affects the streams socket implementation in the kernel.

Default:

24

IP_DOSOURCEROUTE (only for systems with the "enhanced" protocol stack)

Meaning:

IP_DOSOURCEROUTE influences behavior with source routing options. This parameter has the same effect as the *sysctl* variable *net.inet.ip.sourceroute*.

This parameter controls the behavior of the network on the IP layer if IP datagrams have been received with or without data record routing options. If the parameter is set to 1, these datagrams are processed (though there is a security gap here). If the parameter value is 0, these datagrams are rejected and returned with an ICMP message (ICMP_UNREACHED, ICMP_UNREACHED_SRCFAIL).

Default:

0

IPPORT_USER_START
IPPORT_USER_END
Meaning:

The default range of freely assignable port numbers for TCP and UDP comprises values 1024-5000.

The IPPORT_USER_START parameter defines the lower limit and the IPPORT_USER_END parameter defines the upper limit for a single range.

Default:

IPPORT_USER_START = 1024
IPPORT_USER_END = 5000

Value range:

If a parameter is not set the default applies.

Parameter	Minimum	Maximum
IPPORT_USER_START	1024	61559
IPPORT_USER_END	IPPORT_USER_START+1	IPPORT_AUX_LO-1 or 65535

IPPORT_AUX_LO
IPPORT_AUX_HI
Meaning:

By setting IPPORT_AUX_LO and IPPORT_AUX_HI an additional range can be specified if the default range (IPPORT_USER_START and IPPORT_USER_END) is not sufficient or, for instance, a range of ports is to be kept free for specific applications whose servers are linked to ports in this range (e.g. 3000-4000).

The additional range must comprise at least 100 port numbers.

The additional range is only valid for the streams stack on systems with the "fallback" protocol.

Default:

IPPORT_AUX_LO = 0
IPPORT_AUX_HI = 0

Value range:

Parameter	Minimum	Maximum
IPPORT_AUX_LO	IPPORT_USER_END	65435
IPPORT_AUX_HI	IPPORT_USER_START+1	65535

IPPROVCNT

Meaning:

Maximum number of network interfaces that can exist under the IP multiplexer. Each serial connection via SLIP or PPP counts as a network interface.

Default:

128

SUBNETSARELOCAL

Meaning:

The value of this parameter determines whether the maximum size of a transmission unit (*mtu*) should be set to 512 bytes if a subnet is to be accessed via a router. This parameter has the same effect as the *sysctl* variable *net.inet.ip.subnetsarelocal*.

Default:

1

Set the value to 0 if there are routers between the subnets which cannot handle packets of the interface-specific maximum transmission unit size (*mtu*).

11.1.3 Differences between systems with the "fallback" and the "enhanced" protocol stack

In systems with the "enhanced" protocol stack there are some *mtune* parameters, which in some cases have been retained in systems with the "enhanced" protocol stack, while they are no longer used in others. These parameters are described in the table below along with their relevance for systems with the "fallback" and the "enhanced" protocol stack:

Parameter	Description	contained in	
		fallback	enhanced
BEST_MATCH_ROUTING	Affects IP routing decisions	Yes	No See 1.
MAXNETUSERS	Number of users for network services	Yes	Yes
NUMSOM	Max. number of instances of the *sockmod* module	Yes	Yes For TPI endpoints only
NUMTIM	Max. number of instances of the *timod* module	Yes	Yes

			For TPI endpoints only
NUMTRW	Max. number of instances of the *tirdwr* module	Yes	Yes For TPI endpoints only
NUDP	Max. number of open UDP endpoints	Yes	Yes For TPI endpoints only
ARPTAB_SIZE	Length of the *arp* table	Yes	No
ARPTAB_NB	Width of the *arp* table	Yes	No
MAXDUPREQS	Number of entries in the NFS duplicate cache	Yes	Yes
NCSIZE	Size of the DNLC	Yes	Yes
SM_SIZE	Size of memory for buffering file input/output	Yes	Yes
SOMAXCONN	Max. number of attached TCP connections	Yes	Yes
NTCP	Max. number of open TCP endpoints	Yes	Yes For TPI endpoints only
TCPNODELACK	Global lock for TCP confirmation delay	Yes	Yes
TCPACKONPUSH	Deactivates TCP acknowledgment	Yes	Yes
TCPTV_KEEP_INIT	Affects connections in the process of being established	Yes	Yes
TCPTV_KEEP_IDLE_SECS	Number of seconds elapsed before TCP sends keep-alive packets	Yes	Yes
TCPTV_KEEPINTVL_SECS	Interval between sent packets	Yes	Yes
tcp_sendspace□ tcp_recvspace	Lower limit for send and receive space	No	Yes

tcp_sendfactor tcp_recvfactor	Factor for calculating the size of the send and receive space	No	Yes
tcp_mssdflt	Max. segment size	No	Yes
rtq_reallyold	Number of seconds after which a non-referenced route is removed from the tree structure	No	Yes
IPFORWARDING	Releases the IP forwarding mechanism	Yes	Yes
IPSENDREDIRECTS	Affects generation of ICMP messages	Yes	Yes
IPCNT	Number of IP users	Yes	Yes
IP_DOSOURCEROUTE	Affects source routing options	Yes	Yes
IPPORT_USER_START IPPORT_USER_END	Default range of freely assignable port numbers for TCP and UDP	Yes	Yes
IPPORT_AUX_LO IPPORT_AUX_HI	Additional range of freely assignable port numbers for TCP and UDP	Yes	Yes
IPPROVCNT	Number of IP providers	Yes	Yes
SUBNETSARELOCAL	Controls the setting of the MTU to 512 bytes	Yes	Yes

Table 5: mtune variables for the "fallback" or "enhanced" protocol stack

11.1.4 Die sysctl interface

The *sysctl* interface is a more convenient option than *idtune* for influencing the behavior of a system with the "enhanced" protocol stack. *sysctl* permits intervention in the active system. The */etc/conf/cf.d/stune* file is first modified with *idtune*. This file is overwritten with the values for *mtune* when the kernel is relinked. The system kernel is then relinked and the system is restarted so that the changes take effect.

Changes made to systems with the "enhanced" protocol stack via the *sysctl* interface take effect immediately in the active system. The changes are reset when the system is restarted. To make the changes permanent, the relevant commands can be entered in the */etc/inet/rc.sysctl.local* file. In systems with the "enhanced" protocol stack, this file is executed with each system startup. The specified parameters are therefore reset with every system startup.

The script */etc/rc2.d/S69PREinet* (i.e. */etc/init.d/PREinetinit*) is called directly before *S69inet* during startup and

- checks whether the system is starting with an "enhanced" protocol stack; □
  if not, nothing further happens - if yes
- a check is performed to determine whether the file */etc/inet/rc.sysctl.local* is available, if it is, it is executed.

Example:

To release the message ICMP_HOST_PROHIBITED, which is required in OBSERVE configurations, enter the following line in the */etc/inet/rc.sysctl.local* file:
sysctl -w net.inet.icmp.obsnet_enabled=1

There are a number of *sysctl* parameters which receive their defaults from the files */etc/conf/cf.d/mtune* or *stune*. If you modify such parameters which *sysctl*, the corresponding values in these files are no longer taken into account.

More information on *sysctl* can be found in the "Reliant UNIX V5.45 Networking Reference Manual". A list of the system parameters that can be set in systems with the "enhanced" protocol stack via the *sysctl* interface can be found in the */etc/inet/rc.sysctl.readme* file.

For more detailed information on the "fallback" or the "enhanced" protocol stack, see the Section "Selecting the "enhanced" or the "fallback" protocol stack".

Following an update installation of the system, the contents of an existing */etc/inet/rc.sysctl.local* file may have been expanded. It is therefore advisable following an update installation to check the contents of this file and the settings resulting from this at system startup.

11.2 Adjustable variables in space.c

These files are located in */etc/conf/pack.d/packetname*. *packetname* is specified for each variable in the following description.

tcp_compat_42 in tcp

Meaning:

This variable allows you to specify that the keep-alive packets of the system are structured in accordance with the old model of 4.2BSD systems (setting = 1) or in accordance with a modern model (setting = 0).

In the old model, a keep-alive packet contains one byte of user data (the contents are insignificant). In the new model, it does not contain any user data.

Note: There are TCP/IP implementations that work in accordance with the BSD 4.2 model and require a data byte in the keep-alive packets.

This tuning variable only affects the streams socket implementation in the kernel.

Modification of this variable in the *space.c* file is only valid for the "fallback" protocol stack. For the "enhanced" protocol stack (for both BSD and STREAMS sockets), the *sysctl* variable *net.inet.tcp.compat_42* must be used.

Default:

0

i.e. no user data is sent in the keep-alive packets.

tcp_keep4_3 in bsdnet (only for systems with the "fallback" protocol stack)

Meaning:

This variable allows you to specify that the keep-alive packets of the system are structured in accordance with the old model of 4.2BSD systems (setting = 0) or in accordance with a modern model (setting = 1).

In the old model, a keep-alive packet contains one byte of user data (the contents are insignificant). In the new model, it does not contain any user data.

Note: There are TCP/IP implementations that work in accordance with the BSD 4.2 model and require a data byte in the keep-alive packets.

This tuning variable only affects the BSD socket implementation in the kernel for systems with the "fallback" protocol stack.

Default:

0

i.e. as for the old model (one byte of user data is sent).

N_ARP in arp

Meaning:

Number of minor device numbers made available by the ARP driver.

Each network interface for Ethernet, Token Ring or FDDI occupies one ARP minor device number. In addition, there are functions that access */dev/arp* directly with *ioctl()* calls and thus also require a free minor device number.

Default:

64

MAX_PPDEVS in ppp

Meaning:

Maximum number of serial connections for PPP.

Default:

16

MAX_CDEVS in ppp

Meaning:

Maximum number of upper streams for PPP.

An upper stream is required for each protocol (LCP, IPCP, PAP, CHAP, LQM and IP). Thus, at least three upper streams are required for a normal PPP connection (for the protocols LCP, IPCP and IP).

Default:

64 (= 4*MAX_PPDEVS)

app_fddi_mtu in arp

Meaning:

This variable sets the MTU size (maximum transmission unit) for the FDDI network.

The system uses whichever is lower of the MTU values reported by the driver (RM400) or board (RM600) and app_fddi_mtu (see also the section MTU (maximum transmission unit)).

Default:

FDDIMTU = 4352 (based on RFC 1188)

app_token_mtu in arp

Meaning:

This variable sets the MTU size (maximum transmission unit) for the Token Ring network.

The system uses whichever is lower of the MTU values reported by the driver (RM400) or board (RM600) and app_token_mtu (see also the section MTU (maximum transmission unit)).

Default:

TOKENMTU = 8136 (on account of the CISCO router)

nfs_hiwat in ktli

Meaning:

This variable sets the buffer size for NFS streams at the receiving end.

The aim of this variable is to enhance NFS performance by limiting the probability of data loss in an NFS stream. You can enter any value between 5120 and 262144.

Default:

0

i.e. the default value of 5120 applies.

wan_routing in ip (only for systems with the "fallback" protocol stack)

Meaning:

If *wan_routing* is set to a non-zero value, then any desired remote machine which can be reached via a host entry in the routing table can be used as a gateway to other networks that are attached to the same remote machine.

Default:

1

ullc_token_routing_on in ullc

Meaning:

This variable is for turning source routing on and off (1 or 0) in the Token Ring-specific routing information cache.

It makes sense to turn source routing off when no Token Ring bridges are used. This reduces the load on the system.

Default:

1

i.e. source routing is active.

ullc_token_rcv_allinfo in ullc

Meaning:

This variable specifies which packets with source routing information are taken into account.

When this variable is set to a value other than 0, source routing information on all incoming packets (including data packets) is taken into account. If the value is 0, only ARP broadcasts and LLC packets (except LLC_UI) are taken into account. A value of 0 reduces the system load caused by the evaluation of the source routing information. It must be ensured that all the stations in the ring exchange source routing information by means of ARP broadcasts and LLC packets.

Default:

1

i.e. all incoming packets are taken into account.

ullc_token_mtu in ullc

Meaning:

This variable allows you to limit the MTU (maximum transmission unit) size reported by the controller or driver for Token Ring. The system uses whichever value is lower from the value reported by the driver or controller, the value entered in *app_token_mtu* and this value.

Default:

0, which means that the MTU decision is not influenced at *ullc* level.

ullc_token_singleroutes_br in ullc
ullc_token_allroutes_br in ullc

Meaning:

These variables describe the two-byte routing control fields in allroutes broadcast packets or singleroutes broadcast packets. The values are defined in the Token Ring standard and should therefore not be modified. If no allroutes broadcast packets are to be sent, the *ullc_token_allroutes_br* variable can be assigned the value of *ullc_token_singleroutes_br*.

Default:

ullc_token_singleroutes_br: 0xc270
ullc_token_allroutes_br: 0x8270

max_nlm_range in nlm

Meaning:

The maximum lock range for a lock applied by the NFS client if 0 was specified as the length of the lock. The lock then applies until the end of the file.

max_nlm_range can be configured as there are NFS servers that either do not execute the lock operation or execute it incorrectly with some fixed maximum ranges.

max_nlm_range is specified as a multiple of 1 Gbyte. The highest possible value on a

- 32-bit system is 4 (max. 4 Gbytes)
- 64-bit system is 524288 (max. 512 Tbytes)

Default:

The default and lowest value is 2, i.e. the maximum range of a lock is 2 Gbytes.

11.3 Global kernel variables

Global kernel variables can be changed in *rc* scripts with *crash*:

```
crash > /dev/null 2>&1 <<_EOF_
w variablename variablevalue
_EOF_
```

udpcksum

Meaning:

The *udpcksum* variable controls the formation and checking of the optional (for UDP) checksum of the data in a UDP datagram.

When UDP packets are received, if the packet has a checksum and udpcksum = 1, the checksum is checked. If udpcksum = 0, the checksum is not checked.

When packets are sent, the route entry for the UDP connection is evaluated. If the RTF_NO_CHECKSUM flag is set here, no checksum is entered in the UDP header. If the flag is not set, the value of *udpcksum* is evaluated. If udpcksum = 1, a checksum is formed; if udpcksum = 0, the checksum is omitted. You can use the *route* command to set the RTF_NO_CHECKSUM flag in a routing entry when you create a new route entry.

In a system with the "enhanced" protocol stack, this variable should be modified using the *sysctl* variable *net.inet.udp.checksum*.

Default:

1

See also:

route command

11.4 Routing metrics

For a system with the "enhanced" protocol stack, the information for an entry in the routing tree has been expanded to include routing metrics.

Routing metric are parameters that you receive in the last two lines of the output of the *eroute get* command for a specific endpoint of a TCP connection. The routing metrics show specific routing information for this connection and can be changed specifically for this connection using the *eroute change* command. This does not change the system-wide defaults for other connection parameters (see the Section "Tuning parameters"). Caution must be exercised when changing the parameters for specific connections, as such changes may have a considerable affect on the performance of the connection.

Information on the options for the *eroute* command can be found in the "Networking Reference Manual".

recvpipe
sendpipe
Meaning:

Indicate the send and receive space sizes for a TCP endpoint belonging to a specific routing entry.

Default:

is calculated by the system (see *tcp_sendfactor* and *tcp_recvfactor* in the Section "TCP parameters").

mtu
Meaning:

The maximum size of a transmission unit (*mtu*), in bytes, that can process a given interface. The *mtu* value has a significant effect on the maximum segment size (*mss*), which is negotiated between two endpoints when a TCP connection is being established.

If there already was/is another TCP connection between both machines, the route in the routing tree provides the local system with specific information on the connection to the other machine. Included in this information can be the *mtu* value, adapted to the remote connection through a successfully executed "path MTU discovery". This value is used during the establishment of a new TCP connection to negotiate another value for the maximum segment size (*mss*).

Default:

is adapted to this connection by means of the "path MTU discovery" (see below).

expire
Meaning:

Is the time in seconds after which a non-referenced route is removed from the routing tree structure.

Default:

3600 seconds (see *rtq_reallyold* in the Section "IP parameters").

Determining the maximum transmission unit size (path MTU discovery, PMTU)

When TCP connections are being established, a "path MTU discovery" is carried out to the remote system when the first data is being sent. This means that the smallest transmission unit (*mtu*) of all the transmission units on the route section is determined.

This procedure enables a TCP connection to select a segment size (*mss*) which does not require fragmenting on the IP layer. This procedure is useful for avoiding unnecessary retransmissions of an entire group of IP packets if only one packet in this group was faulty.

The maximum segment size (*mss*) determined using this method is based on the smallest *mtu*. If a "path MTU discovery" were not used, the *mss* size for a TCP connection to a remote system (accessed via at least one router rather than directly) would be set to the default for *mss* (*sysctl net.inet.tcp.mssdflt*). Reducing *mss* to the value set in *net.inet.tcp.mssdflt* can impair the data transmission rate. The result of a successfully executed "path MTU discovery" is retained by the system and used again when a new connection is established to the remote system: the *route* entry generated for the remote system contains this information as a routing metric. This *route* entry is retained for a certain amount of time after the TCP connection to the remote system is released and is reused when a new connection is established.

The "path MTU discovery" can be deactivated for a specific *route* entry with the *eroute* command:
eroute change -lock -mtu *value route*

If this happens, for example, for a network route entry that is used when any remote system in a network is to be addressed, no more "path MTU discovery" operations are carried out when setting up a connection to a remote system in this network.

If a network is divided into a number of subnets as a result of network administration measures, all partners in these subnets are considered as local if the system parameter *subnetsarelocal* is set to 1 (see the Section "IP parameters", SUBNETSARELOCAL parameter). This means that all machines which can be accessed in the subnets are considered as locally addressable, and if no "path MTU discovery" is carried out to these machines, the *mtu* of the relevant network interfaces is accepted as the definitive size for the *mss* calculation.

Glossary

alias

An alias is a short form of a name, e.g. the name of a
→ machine or of an
→ NIS map. It facilitates the input of names. The aliases for a machine are stored in the */etc/inet/hosts* file on the
→ NIS master server. Computers with a particular role have an alias.You can obtain a list of valid aliases for NIS maps with the *ypmatch* command.

ARP protocol

Address Resolution ProtocolThe ARP protocol ascertains the
→ MAC address when only the
→ Internet address is known. If an IP packet is transmitted, only the Internet address of the target machine is known at first. The target machine, however, knows its own Internet address and its MAC address. The sender finds out which MAC target address is mapped to the Internet target address by sending an Ethernet packet to everyone (
→ broadcast). Only the IP module to which the Internet address is assigned will respond. The Ethernet packet sent as the response contains the desired target address. This information about which MAC address is mapped to this Internet address is stored so that other IP packets can be transmitted quickly.

automounter

The automounter is an NFS process that automatically mounts on the client a
→ resource shared by an
→ NFS server when an application tries to access the resource or subareas of the resource.

baseband cable

yellow cableA baseband cable is a coaxial cable that permits single-channel transmission in local
→ networks. The coding of the data on a baseband cable is very similar to that on magnetic tapes. In contrast, broadband cables provide more than one channel.
→ Ethernet uses a baseband cable as its transmission medium. (Synonym: Ethernet cable)

BOOTP server

The BOOTP server is made available to diskless workstations. Using BOOTP, these workstations can find out their Internet addresses and boot file names.

bridge

A bridge is a device on the data link layer of the
→ ISO/OSI reference model that copies
→ data packets selectively between networks of the same type.

broadcast message

A broadcast message is a message sent to all stations connected to the
→ network. There is a defined broadcast address for this purpose. If the destination address of a data packet is the broadcast address, the packet is received by all the
→ machines in the local network.

cache-only server (DNS)

A domain name server that is not authorized for a
→ domain. This DNS server type requests data from authorized servers and stores it in a cache.

client

A client is a
→ machine that requests network services from another machine. A machine can, as a client, request network services for certain functions and at the same time, as a
→ server, provide network services.

client process

A client process is a process that runs on the
→ client. It uses the services of a
→ server process.

CSMA/CD

(Carrier Sense Multiple Access / Collision Detect)CSMA/CD is an access method used by the
→ Ethernet protocol.Reception: Everyone listens to all network traffic. The data part of the packet sent to your own system is passed on to the next higher protocol.Transmission: If there is no packet on the line, a machine wishing to transmit may use the line for its packet. Otherwise, this machine waits until the line is free. If two machines start to transmit simultaneously, the potential collision is recognized and a warning signal (jamming signal) is generated. Both machines terminate transmission and, after waiting a certain period of time (which is different for each machine), start again.

daemon

A daemon is a background process which, once started, generally performs its activities unbeknown to the user. It only terminates when the computer is shut down. The best example of a daemon in SINIX is the printer daemon, which ensures that a file is printed while the user goes on with his or her work. With TCP/IP, daemons are responsible for enabling communication between machines. They accept messages from other machines and ensure that the requests are fulfilled.

data packet

A data packet is a transmission unit.
→ Datagrams consist of one or more data packets.

datagram

In packet switching networks there are basically two kinds of data transfer: in the form of datagrams or via a virtual circuit. In the case of datagram transfer, all transferred data packets are mutually independent. In other words, unlike packets in virtual circuits, each data packet contains information regarding its target address in the network.

domain

A domain part of a
→ network that is administered by a network information system. In this manual, a distinction is drawn between two types of domain: DNS domains and NIS domains. DNS domains are logical administrative units in the Internet in which central DNS servers keep information on the addresses and names in the network. An NIS domain has an
→ NIS master server where all the administrative data regarding all the machines in this domain is stored. A network may consist of more than one domain.

Domain Name System

The name service of the TCP/IP protocol family. The DNS administers network-specific information, such as
→ Internet addresses and
→ machine names, and makes it available to applications and users.

Ethernet

Ethernet is a local area
→ network with bus topology and
→ CSMA/CD access. Ethernet was developed in 1979 and later adopted by the IEEE (Institute of Electrical and Electronic Engineers) as standard IEEE 802.3. For data transfer, it uses a
→ baseband cable with a transmission rate of 10 Mbit/s.

Ethernet protocol

The Ethernet protocol regulates physical data transfer via the network and defines the packet structure of the message packet.
→ CSMA/CD is used as the access method. An Ethernet packet consists of the target address, the source address, the type field, the data field and a frame check sum.

FDDI

Fiber Distributed Data Interface□
CSMA/CDFDDI is an ANSI and ISO standard for LANs (ISO9314-x). It is a fiber-based ring network with a transmission speed of 100 Mbit/s. A duplicate ring running in the opposite direction is used to reduce the risk of failure.

file handle

A file handle is a file identifier used by
→ NFS clients and servers for file operations that is unique throughout the network.

frame

A frame is a message block sent across the network as a unit. In addition to user data, it also contains all the protocol information needed to ensure that the data is transmitted safely.

FTP

File Transfer ProtocolFTP is a
→ protocol for file transfer from and to remote machines. It can be used directly by the user. The prerequisite is that the remote machine also has the FTP protocol at its disposal. The *ftp* program is available for the direct use of this protocol by users of TCP/IP.

hierarchy

A hierarchy is a complete file structure or a part thereof. The term hierarchy refers both to the directories and the files in the file structure.

Internet

A wide area network originally sponsored by the U.S. Department of Defense that uses the TCP/IP family of protocols for data exchange. The *Internet* has since come to be used as a generic term for the ARPANET, DARPANET, DDN Internet and DoD Internet.

Internet address

The Internet address is a network-wide unique address for a
→ machine. It is four bytes long and consists of the
→ network number, a
→ processor number and, optionally, a
→ subnet number. A machine may be accessed by its Internet address or its machine name.

internetwork

A group of networks connected to each other by
→ routers

Internet protocol (IP)

IP is a
→ protocol that carries out path selection (routing) in a computer network. The protocol works with a 4-byte long
→ Internet address for specifying the target and the source addresses. These addresses contain information regarding which network and which machine in the network is to be accessed.

IP MULTICAST

IP-Multicasting is used to combine any number of machines in the network to form a logical group. These machines are then accessed under the IP-Multicast address assigned to their group. The members of a group do not need to know which other machines are in their group.IP-Multicasting therefore takes up a position somewhere between unicasting (which involves controlled access to one specific partner) and broadcasting (access to all machines in the network).IP-Multicast is a software property, and should not be confused with multicasting on the MAC level, which is a network interface property.

ISO/OSI reference model

The OSI reference model (Open Systems Interconnection) represents a framework for the standardization of communication between open systems. ISO, the International Organization for Standardization, has described this model in its international standard publication ISO IS7498. The ISO reference model divides the functions required for system communication into seven logical layers. Each of these layers has clearly defined interfaces to the neighboring layers.

LAN

Local Area NetworkA LAN is a computer
→ network limited spatially to a particular area. In general, the extent of a LAN is restricted to the site of the user. The operator and user of a LAN are generally one and the same person. A LAN can be connected to other computer networks as a private subnetwork, thereby becoming part of a larger network such as a
→ WAN. (Synonyms: local computer network, local network)

local machine

For a particular user, the local machine is always the
→ machine on which he or she happens to be working. All other machines in the
→ network are thus
→ remote machines. If a user logs on at a remote machine, e.g. using the *rlogin* command, this machine then becomes his or her local machine. (Synonym: local host)

MAC address

The MAC address serves to uniquely identify a LAN controller. It is 6 bytes long. The MAC address is assigned by the manufacturer to each controller as it is produced. Every controller worldwide can be uniquely identified by means of its MAC address.

machine

Whenever we refer to a machine in this manual we always mean a computer that is connected to a
→ network. In computer networks, a distinction is made between
→ local machines and
→ remote machines. (Synonym: host)

machine name

A machine name is assigned to the
→ Internet address of a machine. The machine can be addressed by this name in the network. (Synonym: host name)

machine number

The machine number is part of the
→ Internet address. It uniquely identifies a
→ machine in a network. (Synonym: host number)

map

A map is a file in which
→ mount points and
→ resources of the automounter are entered. The
→ automounter reads the maps to establish where and with which parameters a file structure should be mounted.

mount point

The point in the directory tree via which a system accesses a mounted
→ resource. It is normally an empty directory.

mounting

The process by which a client accesses a shared server directory. Rather than copying the
→ resource, the client accesses it transparently. In other words, the same methods are used as when accessing local resources.

MTU (maximum transmission unit)

The MTU for a network type is defined as the maximum length of an IP datagram, including the IP header, that can be transported over the network at one time. Depending on the LAN type, this limit is derived either from the physical limit values of the transmission medium or specified as a logical upper limit in a standard (RFC).

network

A network is a combination of two or more machines via a physical connection with the aim of enabling data exchange on an equal basis between these machines. There are two kinds of networks: local (→ LAN) and non-local (→ WAN).

Network Information System

The Network Information System consists of the NIS → maps, commands that access the maps and → daemons that supply network services.The NIS is a tool that allows the machines of a → domain to be administered centrally.

network mask

The network mask is a four-digit number by means of which TCP/IP can recognize which part of an → Internet address corresponds to the local → subnet number.

network number

All
→ machines that have the same network number belong to a single network. The network number is the first part of the
→ Internet address. It can be one, two or three bytes long. The address is assigned by the
→ NIC (Network Information Center).

NFS

Network File SystemThe Network File System is a distributed file system. In other words, files located on a remote machine can be mounted at the local machine and be processed without having to be physically transferred to the hard disk of the local machine.

NFS client

A system that mounts
→ resources shared by a server

NFS server

A system that shares
→ resources for mounting on remote systems

NIC (Network Information Center)

The NIC is an information center maintained by the Stanford Research Institute that is responsible for administering the IP network numbers and domain names.

NIS

Network Information ServiceThe Network Information Service is an information system that contains all the network organization data for specific machines. This information can be accessed using commands.

NIS maps

NIS filesIn order to perform network administration tasks in a → domain, NIS maps are created on the → NIS master. These files contain information on the machines in the network, NIS servers, accessible networks, mail groups, global user names and groups, and the protocols used. The NIS maps are stored on the NIS master in two formats: the dbm (database manager) format and in ASCII format. Entries in the NIS maps can be made by means of the menu system or directly by editing the ASCII files. Information can be obtained from the NIS maps by means of the *ypcat* and *ypmatch* commands. The most recent version of the NIS maps is copied to the → NIS slaves.

NIS master server

The NIS master server administers all the machines in a → domain. The → NIS maps are located on the master and must be kept up to data by the network administrator. The network administrator may define one or more → NIS slaves that can carry out the same functions as the master should the master fail or if it is overloaded.

NIS slave server

The slaves relieve the → NIS master server of part of its workload. When a → machine requests NIS services, they are provided either by the NIS master server or one of its slave servers. Thus, the → NIS maps must also exist on the slaves. Every time the files are modified on the master, the most recent version of the data must be copied to the slaves so that the whole NIS is consistent.

packet switching

Packet switching is a method of data transmission used in networks whereby the connection is maintained only for as long as it takes to complete the transmission and is then cleared.

port number

A port number enables you to address a certain application within a computer. It corresponds to the address of an application in a computer. The combination of Internet address and port number (=socket) uniquely identifies the receiver or sender of a data packet within the network. Several applications work with predefined port numbers. Others are assigned a free port number each time they are accessed.

PPP

point-to-point protocolThe point-to-point protocol can be used to establish TCP/IP connections between machines via serial lines. This makes it possible to run TCP/IP connections without LAN controllers. SLIP supports all TCP/IP communications functions. Machines connected to each other via PPP can establish connections with *rlogin*, *rsh*, *rcp*, *rtar*, *ftp* and *telnet*.

primary master server (DNS)

The primary name server of a → zone, which administers all the data of the corresponding → domain

protocol

The term protocol is used to mean a convention regarding the setting up, administration and clearing down of a connection for purposes of communication in a → network. A data connection requires a number of different protocols. Protocols are broken down into seven layers in accordance with the → OSI reference model. The lower four layers are used for data transport across a network, the upper three for editing the data for the application. Each protocol contains rules governing the use of data

formats, the sequence of operations and any necessary error handling on the current level.

remote machine

In a local area network, a distinction is made between
→ remote machines and
→ local machines. All the
→ machines in the network that a certain user is not working at are, for this user, remote machines. The user can communicate with all remote machines in the network. If a user logs on at a remote machine, this machine then becomes his or her local machine. (Synonym: remote host)

remote procedure call (RPC)

A client/server implementation in which one process (the calling process) can request another process (the server) to carry out a procedure call. Thus, a remote procedure rather than a local procedure is called.

repeater

A device used on the physical layer that transfers data from one network segment to another (see → bridge).

resource

A file system, section of a file system or individual directory shared by an
→ NFS server for mounting on other machines

RFC (request for comment)

An RFC is a standard defined in the Internet family of protocols (e.g. TELNET or FTP). It is issued under an RFC number.

router

A router is a node machine that transfers packets between two
→ networks using the same protocol architecture (also referred to as a gateway).

routing

It is possible to link networks together by passing packages of the
→ ISO layer 2 from one network to another. This is handled by special hardware, known as the MAC Layer Bridge (MAC = Media Access Control). This "bridging" is transparent to the computers involved; in other words, computers have the same Internet number although they are connected to different physical networks. Another possibility is to pass the packages at ISO Layer 3. In Internet protocols this is implemented via the IP protocol. This procedure is called routing, and the hardware which connects different networks to one another is called an IP router, or simply a router. In this context, the Internet literature also uses the term gateway, although the term gateway is otherwise used only in connection with higher protocol layers.

secondary master server (DNS)

A secondary master server is a backup server that temporarily assumes the tasks of the
→ primary master server if the latter becomes overloaded or fails.

server

A server is a
→ machine that provides network services to other machines (
→ clients).

server process

A server process is a special
→ daemon that performs the network services of a
→ server. A server process receives the access request from another
→ machine and carries out the desired access operation.(Synonym: server)

sharing of resources (NFS)

This is the process by means of which an NFS server enables other machines in the network to access some or all of its file systems.

SLIP

Serial Line Interface ProtocolSLIP is a protocol that makes it possible to set up TCP/IP network connections via serial lines. The SLIP protocol adds control characters to the packets that are to be transmitted to ensure that the receiver can recognize the beginning and end of the transferred blocks even when transmission is non-synchronous.This makes it possible to use TCP/IP connections without Ethernet controllers and using serial lines. SLIP supports all TCP/IP communication functions. Computers that are connected via SLIP can set up connections with *rlogin*, *rsh*, *rcp*, *rtar*, *ftp* and *telnet*.

SNMP

Simple Network Management ProtocolSNMP is a vendor-neutral
→ protocol which provides a tool for managing and monitoring the components of a
→ LAN with
→ TCP/IP protocols.

socket

A socket is an interface for accessing a network. A socket is assigned a machine address (the Internet address) and a
→ port number. This allows a specific application in the network to be addressed uniquely.

subnet

An administrative subarea within a larger network

subnet number

The part of an
→ Internet address that refers to a specific
→ subnet

TCP

Transmission Control ProtocolTCP is a
→ protocol that handles data transport between two
→ machines. TCP roughly corresponds to layer 4 of the
→ ISO reference model. TCP protocols are very secure transmission control protocols. TCP works with port addresses. This enables multiplexing of the machine-to-machine addressing supported by the lower layers. Connections are set up via a so-called "three-way handshake": system A reports that it is ready to transmit and specifies the sequence number of the last byte of its transmit data area that was sent. System B responds with an acknowledgment and its own sequence number. System A now sends its sequence number and an acknowledgment. Transfer then begins. In addition to other information, the receiver may also specify in the TCP header how many more bytes he or she is prepared to receive.

TELNET

Like
→ FTP, TELNET is a
→ protocol for communication with
→ machines running under different operating systems. TELNET makes it possible to set up a session at a remote machine that also has the TELNET protocol at its disposal. TELNET can be called by the user directly using the *telnet* command.

Token Ring

Token ring is a local
→ network with ring topology. It has been accepted by the IEEE as standard IEEE 802.5. It supports transmission rates of 4 Mbit/s and 16 Mbit/s.

transceiver

The transceiver is the adapter unit on the
→ baseband cable. It transmits on the cable and monitors all the information on the cable.

UDP

User Datagram ProtocolUDP is a
→ protocol that handles data transport between two
→ machines. UDP roughly corresponds to layer 4 of the
→ ISO reference model and thus on the same level as
→ TCP. UDP is a datagram protocol that supports broadcasting. Unlike TCP (secured end-to-end protocol), UDP is not secured.

WAN

Wide Area NetworkA WAN is a
→ network that is not restricted spatially to a particular area. In contrast to a
→ LAN, the operator and user of a WAN are generally not the same person. A WAN may consist of LANs.

zone (DNS)

An administrative unit within a
→ domain. Zones often consist of one or more subdomains.

Related publications

[1] **Reliant UNIX 5.45**
Networking Reference Manual

Target Group
Users, system administrators, programmers

Contents
This manual contains the complete description of the commands, C functions, files, drivers, and protocols of the UNIX network software.

[2] **Reliant UNIX 5.45**
Network Programming Interfaces

Target Group
Programmers

Contents
This manual describes the TLI, sockets, and RPC programming interfaces that UNIX makes available for the development of communication software.

[3] **Reliant UNIX 5.45**
Programmer's Reference Manual

Target Group
Programmers

Contents
The Programmer's Reference Manual (Reliant UNIX 5.45) contains a description of the commands, system calls, library functions, file formats, and auxiliary tools that are used by C programmers on the RM400 (or RM200 and RM300) and RM600 system.

[4] **Reliant UNIX 5.45**
System Administrator's Guide

Target Group
System administrators

Contents
This manual provides an introduction to UNIX system administration. It describes, among other things, how to configure and maintain a UNIX system, how to calculate system use, and also gives information on system reliability.

[5] **Reliant UNIX 5.45**
Tuning-Guide

Target Group
System administrators

Contents
This manual contains a description of how a load measurement is carried out and how the measurement results are analyzed to detect existing bottlenecks in performance problems. Specific tuning measures are described, i.e. disk segmenting or the minimization of the process number.

[6] **TRANSVIEW-SNMP V4.0 (UNIX)**
Operation and Functions
User Guide

Target Group
- Network operators
- Network planners
- Programmers of network management applications

Contents
The manual describes how a TCP/IP network is managed using the TRANSVIEW-SNMP graphical user interface.

[7] **CMX V5.1 (SINIX, Reliant UNIX)**
Communication Manager
Operation and Administration
User Guide

Target Group
System administrators

Contents
The manual describes the functional scope of CMX as an agent between applications and the transport system. It contains basic information on the configuration and administration of networked UNIX systems.

[8] **CCP-TR V5.1 (SINIX, Reliant UNIX)**
Communication Control Program Token Ring
User Guide

Target Group
Network administrators

Contents
This manual describes how UNIX systems are connected to Token Ring LANs. Based on CCP-TR, TRANSIT(UNIX) products can be used to communicate with SNA systems in the Token Ring LAN.

Ordering manuals

The publications are ordered through your local Siemens office.

Publications for more Information

[9] **Internetworking with TCP/IP**
Volume 1: Principles, Protocols, and Architecture
Volume 2: Design, Implementation, and Internals
Douglas E. Comer, Prentice Hall

[10] **TCP/IP Illustrated**
Volume 1: The Protocols
Volume 2: The Implementation
Volume 3: T/TCP Illustrated
W. Richard Stevens, Addison Wesley Publishing Company

[11] **TCP/IP Network Administration**
Craig Hunt, O'Reilly

[12] **Managing NIS and NFS**
Hal Sten, O'Reilly

[13] **Managing UUCP and Usenet**
Tim O'Reilly and Grace Todino, O'Reilly

[14] **Using UUCP and Usenet**
Grace Todino and Dale Dougherty, O'Reilly

[15] **DNS and Bind**
Paul Albitz and Cricket Liu, O'Reilly

[16] **The Whole Internet**
Ed Krol, O'Reilly

[17] **Firewall and Internet Security**
William R. Cheswick and Steven M. Bellovin, Addison Wesley

[18] **Network Programming, Volume 1**
W. Richard Stevens, Prentice Hall

[19] **UNIX System V Network Programming**
Stephan A. Rago, Addison Wesley

[20] **Gigabit Ethernet**
Rich Seifert, Addison Wesley