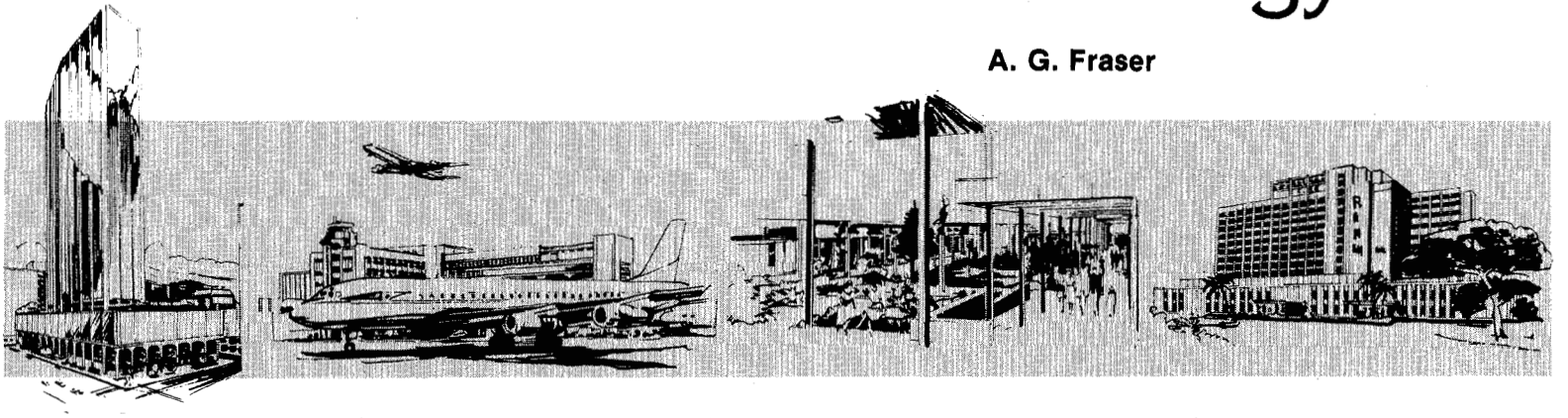
The Present Status and Future Trends in Computer/ Communication Technology

A. G. Fraser



Abstract—This paper[1] reviews some aspects of computer communication network design in the light of recent research. The paper first concentrates on local distribution systems that use demand shared transmission lines. Then there is a discussion of communication protocol and the impact that it has on network performance. Finally there is some discussion of issues that face the designer of a general-purpose data network.

I. INTRODUCTION

It is not difficult to observe the rapidly growing number of private and mutually incompatible data networks [1]. There are networks for credit-checking, for banking, for hotel reservations, for time-sharing services, for local government, and for monitoring continuously running equipment of many types. We may come to regret this diversity. In the modern world no industry is independent of all others and, as more processes become automated, the need for communication between subsystems will become more pressing. Already there are connections between the various airline reservation systems for such purposes as multi-carrier flight reservations and baggage claim. Some airlines have links to hotel-booking and car-rental systems. An increasing number of large corporations and banks have networks of their own and it seems very likely that these will become interconnected at least for the electronic transfer of payments. The trend can only continue and it poses a serious question. Is it practical to build a single network that can handle the great variety of terminal types, computers, and transmission systems that now characterize computer communications? And, if practical, what can current research tell us about the form that it can take?

Some initial steps have already been taken. While private companies have been building their own networks, the common carriers have been installing long-haul digital transmission facilities [2], [3]. These synchronous systems, by using regenerative repeaters, give much lower error rates than the circuits presently used for voice communication. Of course digital transmission is already used in the voice network but it carries data much less efficiently than the new facilities. When data are sent over the voice network, they are transmitted by modulating an analog signal in a modem and, if digital transmission is subsequently used, the analog signal is then digitized. The result is that a 9.6-kbits/s data stream gets converted into an analog signal with nominal 4-kHz bandwidth and that, when digitized, becomes a 64-kbits/s bit stream. Obviously, direct transmission of binary data over a digital circuit is more efficient than this.

However, there is some indication that making best use of available bandwidth is not the major problem in making a computer network. The cost of connecting a terminal to its nearest switching machine is a much more pressing issue. The equipment used to make that connection, called the local distribution system, involves costs that are multiplied by the number of subscribing terminals. Furthermore, an increasing part of that cost is for manpower that is difficult to replace by automation. Equally important and difficult are the problems of control that arise in computer communication systems. The principle problem is not that some required functions are necessarily expensive to perform but that we understand so little of the task that it is difficult to be

[1]The text of a talk given at a meeting of the Japanese Electronic Industries Development Association, Tokyo, Japan in October 1975.

The author is with Bell Laboratories, Murray Hill, NJ 07974.

certain that we are doing the right thing. By providing the wrong facilities we may invest in a network that eventually turns out to be as much a liability as it is an asset.

For these reasons this review of computer communications focuses first on research that has possible application in a local distribution system, and then on the role of communication protocols in packet-switched systems. The paper ends with some speculation on the prospects for new technological advances towards achieving a shared data network.

II. LOCAL DISTRIBUTION SYSTEMS

One way in which one can reduce local distribution costs is to minimize the number of transmitters and receivers required to connect N terminals to a switching center. Polling is one example of such a technique. Consider, if all terminals had a separate transmission line to the switching center there would be one transmitter and one receiver at each end of each line as shown in Fig. 1(a). A total of $2N$ transmitters and $2N$ receivers would be required. In a polling system the terminals each have one transmitter and one receiver but they use a common line and the switch has only one transmitter and one receiver for that line [Fig. 1(b)]. The total is therefore $N+1$ transmitters and $N+1$ receivers.

There are, of course, many variations of the polling technique, but they can all be typified as follows. Each terminal contains a buffer in which to queue data for transmission to the central switch. At regular intervals the switch polls each terminal by transmitting the terminal's address. The terminal is expected to respond by transmitting any data that it has ready for transmission. In this way each terminal gets a chance to use the transmission line, and the frequency of use is determined by the switch. A problem with the scheme is that much transmission line time is taken up by polling terminals that turn out not to have any data ready for transmission.

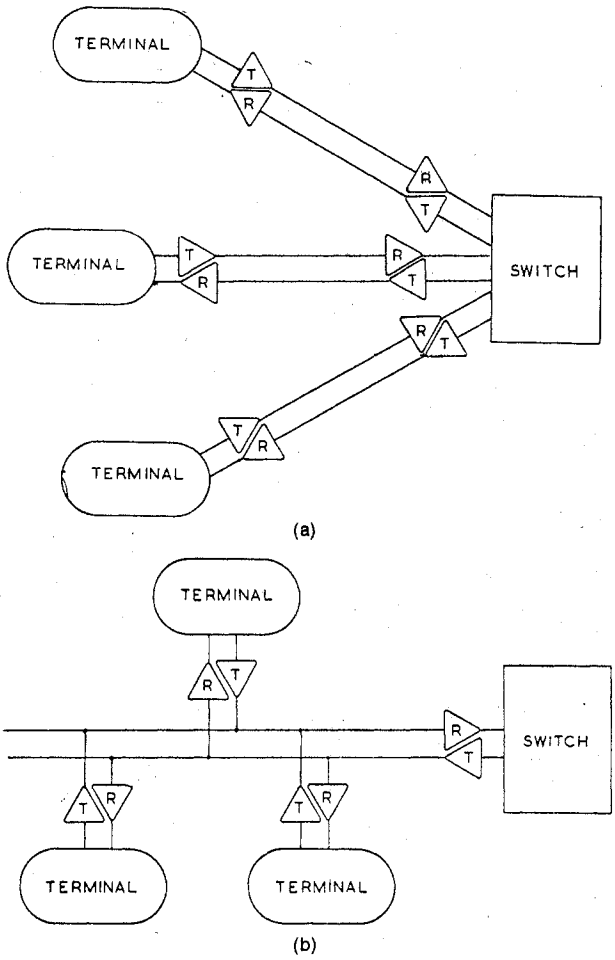


Fig. 1.

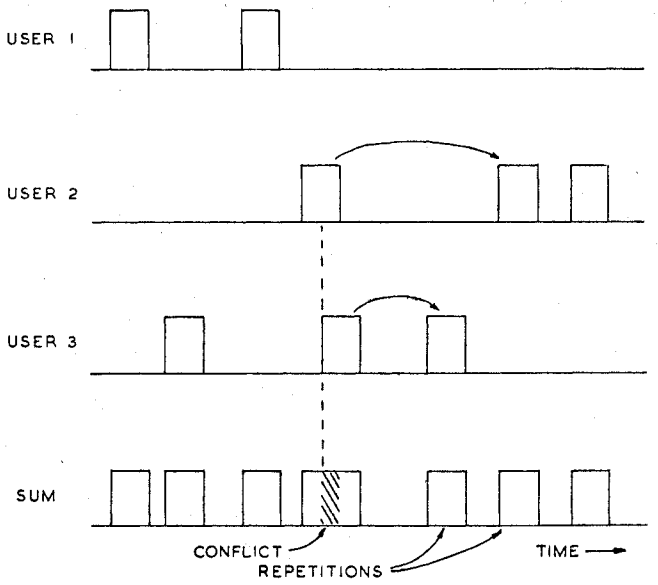


Fig. 2. Saturation load = 18 percent.

The ALOHA Technique

In 1969 Abramson [4] described an alternative to polling that still uses a single shared transmission path. (Abramson was in fact concerned with the use of a radio channel rather than a transmission line but the logic of what he did could be applied to either medium.) The technique is used in the ALOHA system at the University of Hawaii. Like a polling system, every terminal keeps a buffer with data queued for transmission. But unlike a polling system, the central switch does not attempt to coordinate the terminals (see Fig. 2). As a result several terminals may transmit at once. Of course, if there is not much traffic, the chance of two terminals transmitting at once will not be great, but as traffic levels increase, so does the chance of a collision.

Abramson gave a simple analysis of the technique and showed that the channel becomes saturated when the offered load is about 18 percent of that which the channel could carry if there were no sharing. When the load is very light the channel is idle most of the time and collisions are infrequent. As load increases the channel becomes more busy and collisions occur more frequently. Collisions cause retransmissions and the retransmitted packets cause a further increase in channel traffic with a consequent increase in the collision rate. The situation can be improved somewhat by arranging that all terminals synchronize to a common clock [5] (Fig. 3). Time is divided into time slots equal in length to the time to transmit one packet. Packets are transmitted in these time slots. The result is to raise the channel saturation point from an 18 percent load to a 36 percent load.

Each packet carries a checksum. When the switch receives a packet, the checksum is verified and, if good, an acknowledgment is transmitted back to the terminal that sent the packet. Packets with bad checksums are assumed to result from collisions when two terminals transmit simultaneously. They are ignored. When a terminal has a packet to send it transmits it immediately without regard to the data that are already being broadcast by other terminals. If an acknowledgment is not received, the terminal waits a randomly chosen length of time and then transmits again. The system assumes that if two terminals should transmit at the same time both transmissions will be received erroneously and so both terminals will have to retransmit. By requiring a random delay before retransmitting, the terminals reduce the chance of further conflict.

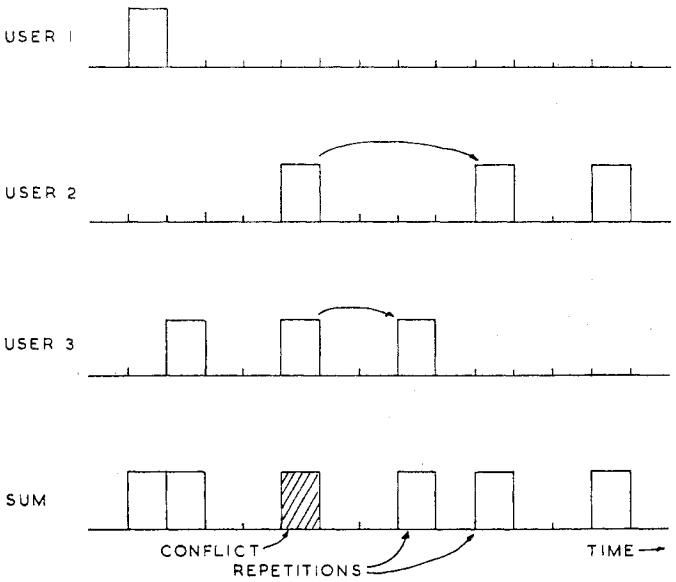


Fig. 3. Saturation load = 36 percent.

The expected delay for a message passing through an ALOHA channel depends upon how long it takes before a packet involved in a collision gets retransmitted. That delay has two components. A constant part of the delay is the time taken before the transmitting terminal knows that its transmission failed. The variable part is the random delay introduced by a terminal in order to minimize the chance of subsequent collision. To minimize the constant part of the delay one need not wait, as Abramson suggested, for a returning acknowledgment but the sending terminal could look at the transmitted signal and determine directly when conflict has occurred. To minimize the variable part of the delay is to increase the chance of a subsequent collision and so reduce the load at which the channel becomes saturated.

Suppose that the retransmission delay is chosen randomly to be between 1 and K packet transmission times. The relationship between delay, offered load, and K has been estimated by Kleinrock and Lam [6], and Fig. 4 shows their results for a 50 kbits/s satellite channel carrying 1125-bit packets. (It takes a radio signal about 0.27 seconds to travel from earth to a satellite and back, so that a transmitter cannot detect that a clash has occurred until that period has elapsed.) One can see that the optimum K, in this case at least, lies between 8 and 15 packet times. But we can also see that there are two values for delay at a given level of offered

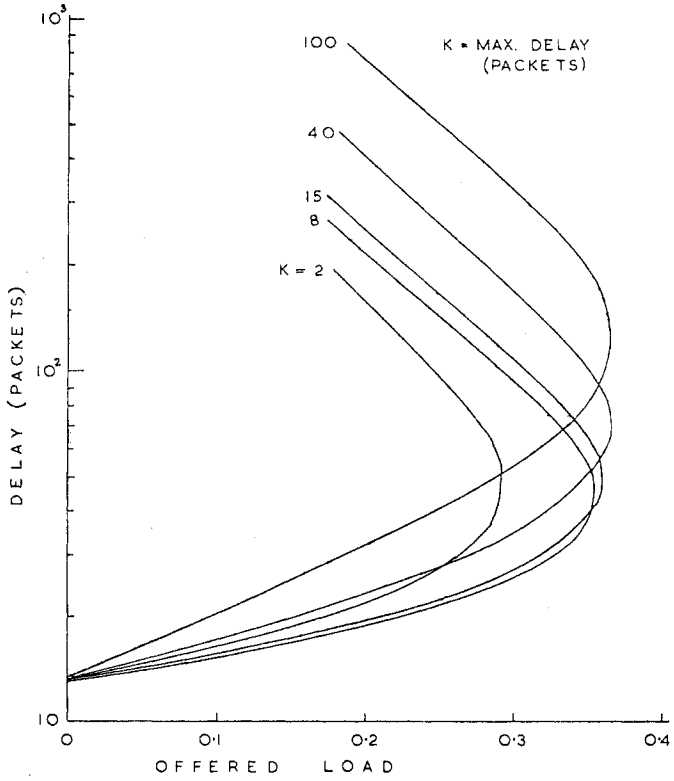


Fig. 4.

load. The system is apparently bistable.

The problem is that, although the average load may be less than 36 percent of the line's capacity there will be fluctuations when the instantaneous load exceeds that value. Temporarily, there will be heavy traffic on the channel. Clashes will be more frequent and so the traffic on the channel will become heavier. And so the vicious cycle proceeds, bringing the system to a standstill. For a given statistical distribution of message arrivals there is a relationship between the number of users and the value of K which will cause the system to become unstable. For example, if the system whose behavior is shown in Fig. 4 is operated with $K = 10$, and if its users each contribute a load of about 0.11 percent, the system will be stable providing that there are no more than 110 users. Actually, if the number of users is increased slightly above this number, the channel will operate satisfactorily for a predictable length of time before a temporary fluctuation in the load causes it to seize up. For example, with 220 users the channel will fail approximately once in every two days.

An ALOHA system uses essentially the same transmission line layout as a polling system. In the ALOHA system bandwidth is wasted as the result of clashes between competing terminals. In a polling system bandwidth is wasted by polling terminals that have nothing to send. A compromise is possible. Suppose that we add one more transmission path, a "reservation channel," to a polling system. It is provided to allow the terminals to tell the central machine when they have something to transmit, which they do by transmitting their address using an ALOHA technique. The central machine now only polls those terminals that have something to transmit. Polling with a reservation channel is more efficient than a pure ALOHA system because it is only the reservation channel

that must operate with less than 36 percent efficiency and that channel, having only addresses to carry, need not have a large bandwidth. It is more efficient than a pure polling scheme if fewer than 1/6 of the terminals have anything to transmit when polled. A more sophisticated transmission system based upon the reservation-channel technique has been described by Roberts [7] in connection with satellite communications.

Data Loops

The ALOHA and polling techniques lose performance because of the need to allow for signal distortion introduced by a transmission line. In general, when a transmitter sends a signal to a receiver the receiver must be adjusted, or must automatically adapt, to the distortions introduced by the line. When the receiver must listen to a series of different transmitters time must be allowed between each transmission for the receiver to adapt to the new transmitter's signals. Thus there is always a necessary delay at the start of packet transmission in ALOHA and polling systems.

In 1969 Newhall and Farmer [8] described a technique in which these delays do not occur and yet their system uses the same number of receivers and transmitters as a polling system. Newhall and Farmer connected their terminals into a loop so that the output of one terminal fed the input of the next and the output of the last fed the input of the first (Fig. 5). (Actually they were not so much concerned with local distribution from a switching center as with providing a symmetric means of communication between all pairs of terminals on the loop.) Under this arrangement data can be passed in either direction between any two terminals provided that intervening terminals pass the information around the loop without alteration. For this purpose it is convenient to install special hardware, a "node," at the point where a terminal connects to the transmission lines. The node serves to regulate a terminal's direct access to the loop and to bypass the terminal when other conversations are in progress.

In the scheme developed by Newhall and Farmer, terminals transmitted messages to one another. Transmission on the loop itself was by means of a bipolar signal in which a single binary digit was represented by a pair of pulses with opposite polarities. Pulse strings that violated this format were used to denote start of message (SOM) and end of message (EOM). The message format is shown in Fig. 6. The messages are variable length and have the source and destination addresses in the first 12 bits. A terminal expecting to receive data scans the line for a passing message having the terminal's address in the destination address field. A terminal wishing to transmit must await its turn. Following the EOM symbol is a single bit, the "token." When a terminal completes sending a message it affixes a "one" token and passes it on to the next node on the loop. If that node has no data to transmit the token is passed on unaltered, but if there are data to transmit, the node changes the token to a "zero" and then proceeds to send its message.

Several schemes have since been used for controlling the sharing of loops [9]-[11], [13]. One that we have used in my laboratory treats the loop like a conveyor belt with fixed size slots for packets of data (see Fig. 7). The network in which the loop is used is called "Spider" [10]. It connects together a number of small computers. Data are transmitted and received in packets each bearing a seven-bit address. Each packet occupies one time slot on the conveyor belt and empty slots have zero in the address position. When a computer has data to transmit, its node inserts its own address in the packet address field and places the packet in the next empty slot on the transmission line. The packet is then carried to the switch. The packet address then tells the switch where the packet came from, and by referring to a route table, established by a previous call setup procedure, the switch decides to which computer the packet must be sent. The address of the destination computer is then written in the packet and it is sent out over the loop. The receiving computer's node recognizes its address in the packet, reads the packet from the line, and leaves the slot empty.

Loops do not have to be used in a demand-shared manner. The IBM 2790 [11] uses synchronous time-division multiplexing on a loop that connects several terminals to a central

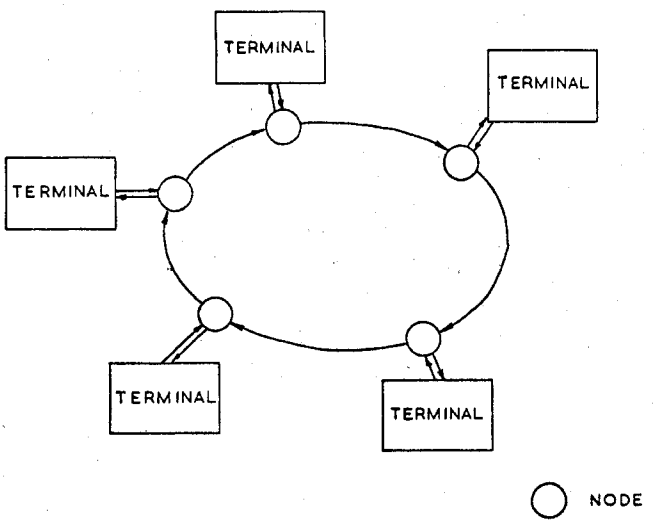


Fig. 5.

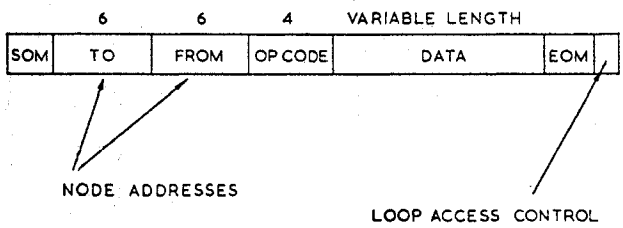


Fig. 6. Newhall/Farmer packet structure.

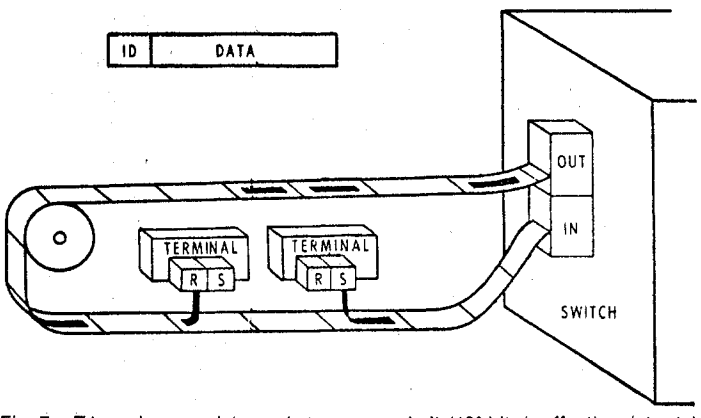


Fig. 7. T1 carrier as a data packet conveyor belt (10^6 bits/s effective dat rate).

machine. Each terminal is assigned a time slot and must hold its data until the proper time slot arrives. If the terminal has no data when its time to transmit comes round, the time slot goes unused. In general that scheme requires less complex control circuitry in each terminal, but uses the transmission line bandwidth less efficiently than the Farmer/Newhall and Spider schemes.

Hayes has published a queueing analysis [12] in which synchronous and asynchronous data loops are compared. He used the Spider system as a model and compared it with a synchronous system that uses the same packet size and line speed. He assumes that terminals generate messages at random and that these are queued up in the terminal until such time as they can be transmitted. Typical results are shown in Fig. 8. The delay, in packet times, is the time between a message being generated at a terminal and the last bit of the message being transmitted on the loop. The load is the total rate of information generated by all terminals expressed as a fraction of the capacity of the transmission line. Clearly, when the load is very small the line is usually available for asynchronous transmission and such transmission can proceed at the raw speed of the line. In the synchronous case transmission always proceeds at $1/N$ of the raw line speed where N is the number of terminals sharing the line. At high load levels both systems give long delays because the frequent bursts of message arrivals must be evened out over a long period and the queues are therefore long.

Loops are attractive because they give better line utilization than the ALOHA and polling systems. They have one major problem: they are vulnerable to mischief and malfunction in ways that the other systems are not. The problem is that one terminal node carries the traffic for many other terminals. There is a risk that a malfunction at one user site might deny service at many other sites. To overcome this last problem various schemes have been suggested to provide automatic error detection and loop reconfiguration. They all suffer from essentially the same problem. Since many users are inconvenienced when one part of a loop fails the loop must be reconfigured quickly to exclude the faulty component. That means quick and automatic diagnosis of the problem, and that is not easy.

III. PACKET SWITCHES

Pierce [13] has suggested that a network be made entirely out of interconnected loops (Fig. 9). Between one loop and another, he would install a switching machine much like a packet switch. However, his would be a much simpler device than the packet switches that are in use in some networks today. It would be concerned solely with switching, whereas current packet switches tend to include extra facilities such as automatic error control and flow control.

I shall use the ARPA network [14]-[17] as an example of the current trend in packet switching.

The ARPA network provides error control by automatically retransmitting packets that fail to be transmitted successfully. It provides flow control by limiting the size and number of messages that can be in the network at any one time. The mechanics of this rely upon a hierarchy of communication protocols [14], [15].

Level-0 Protocol: Suppose that computer A is sending messages to computer B (Fig. 10). A is connected to switch Sa and B to switch Sb. Between Sa and Sb are other switches S1, S2, etc. A message generated by A is passed to Sa where it is split into packets. The packets are routed through $S1$, $S2$, etc., until they reach Sb. (I shall discuss packet routing later.) At Sb the packets are reassembled into a message and the message is then delivered to B. The transfer of a packet from one switch to another proceeds as follows. Switch $S1$, say, has the packet in its memory. It gives the packet a sequence number and checksum and sends it to $S2$. $S2$ is expected to acknowledge receipt of the packet and $S1$ will then discard its copy. If $S1$ does not get an acknowledgment in a certain time, it retransmits the packet. There are several reasons why an acknowledgment may not reach $S1$. The packet may have been corrupted during transmission, or the acknowledgment itself may have been corrupted. Alternatively, $S2$ may not have enough storage to

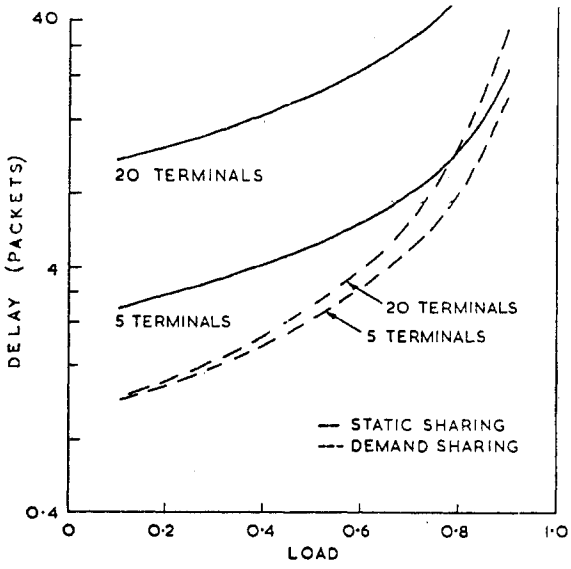


Fig. 8.

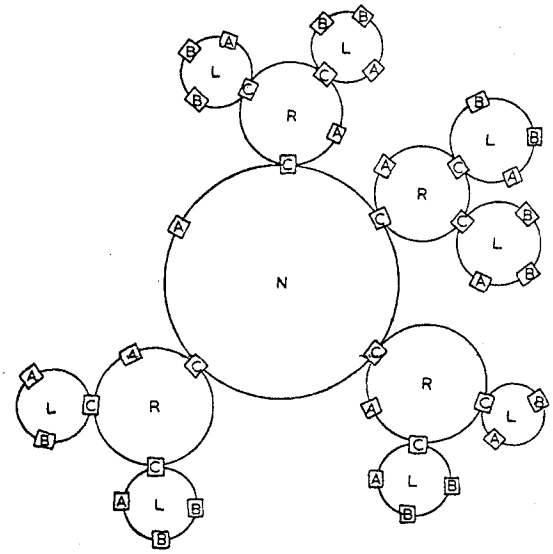


Fig. 9. Pierce loop system.

hold the packet or, because it is too congested, may have decided to ignore the packet temporarily. In any event, once S2 has acknowledged the packet, it is S2's responsibility to make sure that the packet does not get lost.

Level-1 Protocol: What I have just described, very briefly, is the lowest level communication protocol of the ARPA network. The next level of protocol is between *Sa* and *Sb*. It operates in terms of messages. Each message arriving at *Sa* from *A* is transmitted to *Sb* where it is delivered to *B* and at the same time an acknowledgment message is sent back to *Sa*. Flow control at this level is done in two ways. First, *Sa* cannot send more than four messages at a time to *Sb*. After sending the fourth such message *Sa* must wait for an acknowledgment from *Sb* before sending any more. Second, *Sa* must request message storage space in *Sb* before sending a long message to *Sb*. (A long message is one that occupies more than one packet.) *Sb* will respond to a request for space by returning an allocation message to *Sa* whereupon *Sa* can transmit its long message. (I shall discuss this again later.) When *Sa* receives a message from *A*, a software checksum is appended and is carried with the message for the rest of its journey to *Sb*. The problem is that the checksum used in the lowest level protocol between switches does not check that the switches themselves are working properly; it only checks the transmission lines.

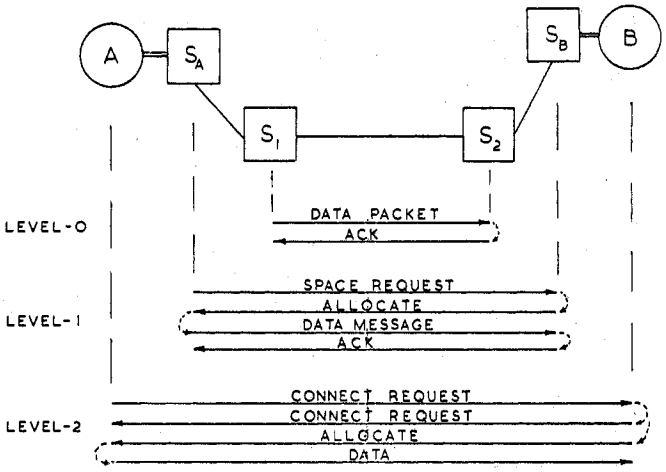


Fig. 10.

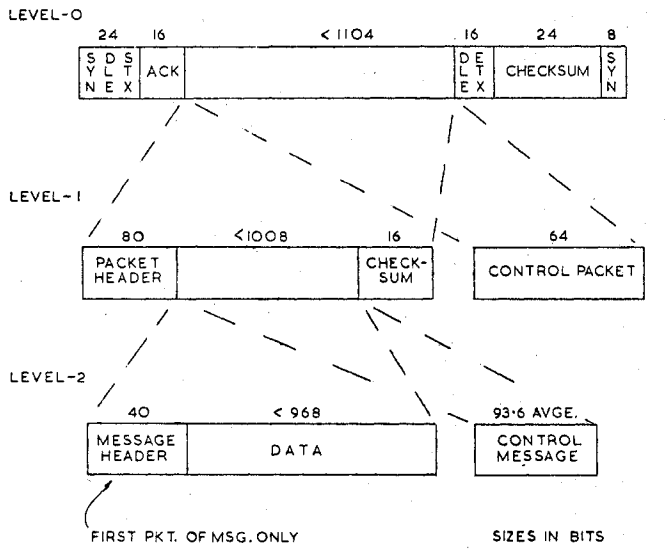


Fig. 11.

Level-2 Protocol: The next level of protocol is between *A* and *B* where virtual circuits are established. Messages simply become the vehicles for carrying data and control information on these circuits. Each computer can support 256 different virtual circuits at one time, and each is identified by an 8-bit "socket" number. When a connection is established between *A* and *B*, a socket belonging to *A* is connected to a socket belonging to *B*. To establish such a connection, *A* and *B* must exchange messages called "requests for connection." (Actually, connections are usually made in pairs because a single connection is only a simplex path.) Once the connection is established, data can be transmitted between the computers but only in accordance with a flow control mechanism that operates independently on each connection. If *A* is to send data to *B*, *B* must first send an allocation to *A* and that allocation is for a certain number of messages and a certain amount of data. *A* must not send more of either than it has been specifically allocated.

So far I have described the lower three levels of protocol. There are others [16], [17] but typically the hierarchy does not get more than about five deep.

Data formats used by the ARPA network are illustrated in Fig. 11. At each protocol level, we find that there are both data and control to be transmitted. The level-0 protocol uses 48 bits for frame alignment and 56 bits for control in every packet. That leaves 1088 bits for data and higher level protocols. The level-1 protocol takes a further 80 bits in each data packet, but also generates control packets that carry no data at all. The level-2 protocol uses 40 bits for control purposes in each data message, and it too generates messages that carry no data. And so it goes for each level of protocol. Measurements made during one week in 1974 [18] show that, on average, 6.7 percent of the line capacity was being used and, of that, only 0.6 percent was user data. Figures taken with so little traffic can be misleading unless used with care. If one assumes that the profile of user behavior remains constant, but that usage builds up to give a traffic intensity of 75 percent, one gets the following forecast of how the lines would be used in a fully loaded network (see Fig. 12). (Anything higher than 75 percent traffic intensity would result in undesirably long queues.) About 26 percent of line capacity would be used by the level-0 protocol, 18 percent by the level-1 protocol, and 9 percent by the level-2 protocol. There is a small amount (4 percent) of background traffic for network management, and the remaining 18 percent would be user data. If the users were to change their pattern of behavior to make the best possible use of the network line, they could use up to about 59 percent of the available line capacity. If they were all to operate terminals that generate, on the average, 12 characters per message (typical for users of time-sharing systems (19)), the figure would fall to 14 percent.

There is no doubt that some of this overhead is required, but it is plain that we have a lot to learn about protocol design and how to efficiently control computer communications.

The Importance of Delay

Each one of the protocols described above involves a

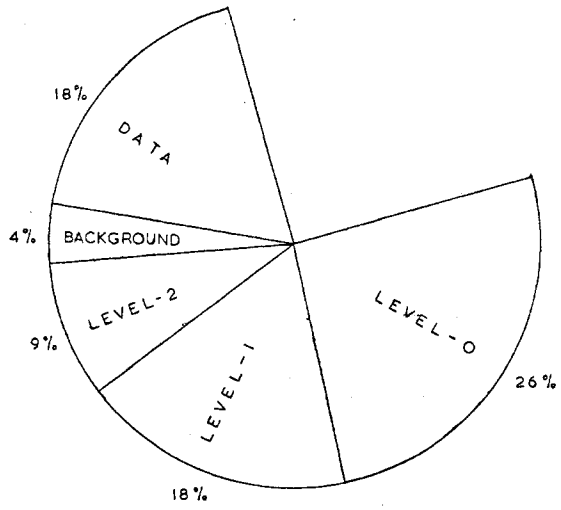


Fig. 12.

handshake procedure. The level-0 protocol requires that acknowledgments be transmitted for each packet received. At level-1 there are acknowledgments and space allocations. At level-2 space allocation messages must be sent from receiver to sender before data can flow. Handshakes are apparently typical of the control procedures used at all protocol levels. Consider, for example, a file transfer in the ARPA network. I estimate that it involves at least 28 end-to-end handshakes. At level-1 there is an end-to-end handshake for each message transmitted. At level-2 there is a protocol for creating and taking down a virtual circuit, and each of these actions requires an exchange of messages. At higher protocol levels there are handshakes when the user logs in, provides his password, establishes the parameters that are to govern the data transfer, and commands the transfer to take place. Therefore, although a user may think of a file transfer as a simple unidirectional data flow, it is in reality a quite extensive exchange of messages. The time required for a file transfer depends upon the speed with which the required handshakes can be completed.

In my experience, a network user estimates the performance of the system by dividing the time that it takes to transmit a file by the size of the file. A network with very high-speed lines will be regarded as slow if queueing delays within the network are long and handshaking is therefore slow. For this reason I believe that, as users become more sophisticated in their use of computer networks, the demand will grow for a network that has low queueing delays.

Static and Dynamic Routing

Many of the design features of a packet-switched network stem from one decision: whether the packets of one conversation are allowed to get out of sequence. In the telephone system, the user dials up a connection before he starts to talk. Part of the procedure for setting up the connection is to choose a route passing from the person making the call, through several switching machines, to the person being called. All information transmitted as part of the conversation then follows that route. The same procedure could be used in a packet-switched system and I shall call it a "fixed route" strategy. Another possibility is to choose a route separately for each packet that enters the network. Suppose that terminal *A* transmits a message consisting of several packets. The first packet enters switching machine *Sa*. That machine examines tables telling how busy the other parts of the network are and which transmission lines are inoperative. From that information it chooses the next switch which the packet should visit. The next switch does the same until eventually the packet reaches switch *Sb*. The second packet of the message from *A* undergoes the same procedure and eventually reaches *Sb*, but there is no guarantee that the second packet will follow the same route as the first. I call this a "dynamic route" strategy.

A number of network designers seem to have followed the ARPA lead and have used a dynamic route strategy. Thus we can now examine how successful that strategy has been. It has problems. Most of the difficulties arise when the packets transmitted from *A* arrive in a different sequence at *B*. That is a distinct possibility with a dynamic route strategy since there is nothing to prevent the first packet from being assigned a slow route while the second goes by a faster route.

It is impractical, in the space available, to show you how complex life can get when packets get out of sequence, so we must be content with one simple example (see [20] for other examples). The ARPA network promises its users that the data bytes of a message will leave the network in the sequence that they entered the network. Suppose that packet 2 reaches *B*'s switch before packet 1. The switch must store 2 until 1 arrives. In general *B*'s switch must be prepared to collect all the packets of one message before delivering any to *B*. That requires storage in the switch. Suppose that the storage to be used for that purpose is shared among the many users connected to the switch. In that way we take advantage of the bursty nature of data flow and need not provide as much storage as would be required to hold all messages for all terminals. But now suppose that, on a particularly busy occasion, the 2 packets of many messages arrive at the switch. Perhaps they will use up so much space that there is no room in which to put any packet 1. Thus the switch must turn away the very packets that will allow it to complete the delivery of data which are now blocking up its store. No further traffic can now pass through this switch. To overcome this problem the ARPA network has introduced an extra handshake procedure into the level-1 protocol. Before a multipacket message can be sent from *A* to *B*, *A*'s switch must ask for, and receive, an allocation of the necessary storage space in *B*'s switch. To do that adds an extra round-trip delay to the transmission delay for a message. So, to reduce the average additional delay, another bit of control procedure is added. After *B*'s switch receives one long message from *A* it automatically allocates storage for another. Now *A* must tell *B* if it does not need the extra storage.

When part, or all, of a network gets into a state where no further traffic can flow it is said to be in a "lock up" condition. There is a distinct chance of such a condition arising in many packet-switched networks, even when they preserve the sequence of data for one message. It can arise if the network attempts to control transmission errors separately on each link of the path between source and destination terminals. For example, suppose that switch *S1* sends data to *S2* but keeps a copy of those data until *S2* acknowledges their receipt. Unless precautions are taken, there may come a time when *S1* has its memory full of

packets for switch S2, while S2 has its memory full of packets for S1. Neither switch has room for more data so each must discard all further inputs. No further progress is made. That, of course, is a simple condition to anticipate and prevent, but more complex situations, involving many switches, can arise and are very difficult to handle. So far as I know, it is not possible to say for sure that any packet-switched network that uses a dynamic route strategy or does switch by switch error control is free from the possibility of lock-up.

For these reasons, I anticipate that a dynamic route strategy will not be a popular choice for future large-scale data networks. For years the common carriers have faced the problems of making a reliable switching machine and have developed the technique of switching in a new circuit when another fails. Those techniques can be applied to data communications and the control problems associated with dynamic routing can be avoided.

IV. FUTURE PROSPECTS

Turning now to the future, let us consider what advances in technology might be looked for in the next few years. I shall consider those aspects of computer/communications technology that will improve the prospects for achieving a single network to serve a wide variety of terminal types and usage patterns. Solutions to the following problems are required.

1) How to minimize the cost of local distribution and yet accommodate, in one network, a wide variety of terminal types and speeds.

2) How to control communications so that there can be full connectivity between devices that talk at different speeds and with different protocols.

3) How to design efficient protocols and how to prove that they work.

4) How to simultaneously obtain low delay and reasonable line utilization.

5) How to control the flow of information in a network so that there is a low probability of data loss due to transmission error or queue overflow.

6) How to reconcile the need for heavy investment, characteristic of communication networks, with the continuing high rate of change in computing technology.

Protocol design problems may turn out to be the most difficult. Our present ability to handle communication protocols reminds me of our abilities with programming language in the 1950's. At that time there was no convenient formalism in which syntactic constructs could be described. Without such a means of expression we were in a poor position to make proofs about the languages we invented and there was little that could be done to automate the process of compiler construction. The breakthrough came in about 1958 with the official description of Algol 60 and its use of the Backus-Naur Form (BNF) to describe the syntax of the language. Today we anxiously await a similar breakthrough for the description and manipulation of communication protocols.

For the experimentalist seeking to make progress on the other problems mentioned above, there are at least two promising possibilities. One is clearly the application of LSI, particularly microprocessors, to switching and control. (I shall say more about that later.) The other is to reevaluate ideas originally conceived in connection with digitized voice communication. Until recently there has been little interchange of ideas between communications engineers and those who specialize in computing science. As a result techniques known in one field have been slow to be applied to the other and there is little exchange of information about problems which are essentially common to the two fields. For example, a great deal is known about the construction of switching machines for digitized voice, yet apparently very little of that knowledge has been applied to data networks. Existing machines switch digital signals in time and space; addressed packets just represent one more dimension in which these machines might work. The computing literature has, in recent years, contained much discussion on the problems of finding a good model for interprocess communication. Even now few operating systems provide good facilities of that type. But signaling systems are the basis of all communication networks and they fulfill essentially the same role. Actually neither industry seems to have the subject under control and each is approaching the problem in a different way.

Let us now speculate a little on how some of the problems and opportunities just mentioned might impact computer communications systems in the next few years.

In order to understand how delay might be minimized, we can start with analytical results obtained by Chu [21]. Fig. 13 shows the relationship between delay and traffic intensity in a single packet switch. There are several curves, one for packets of fixed size and others for variable-size packets with different statistical distributions. (k is the coefficient of variation of packet size.) The main thing to notice about these results is that delay measured in packet transmission times is a constant for a given traffic intensity. Delay increases linearly with packet size and therefore so does the amount of queue storage space that must be provided in a switch. In order to obtain minimum delay one should use quite small packets. But small packets could result in low line utilization. In order to estimate how packet size affects line utilization, one can refer to a paper by Hayes and me [22] where it is shown that a good compromise is obtained by choosing a packet in which the data occupy about eight times as much space as the header.

Low delay therefore requires small packets which, in turn, require small packet addresses. To obtain a small packet address in a large network, one cannot afford to put the complete address of a destination in each packet. Instead, one should use an abbreviated addressing scheme (Fig. 14). When a conversation is first established, its route can be chosen, and at the same time the packet addresses to be used on each leg of the journey can be chosen. The address used on a given transmission line need only be sufficient to distinguish between the conversations being carried by that line. (Assigning a packet address to a conversation in a packet-switched system can be like assigning a time slot in a synchronously multiplexed system.) The packet address might be as short as one byte.

A small packet may give rise to minimum delay but it poses a problem in switch design. Most current computer networks use switches constructed out of general-purpose minicomputers. The more high-performance of these transfer packets into and out of memory by means of direct memory access channels. The processor is only called in to

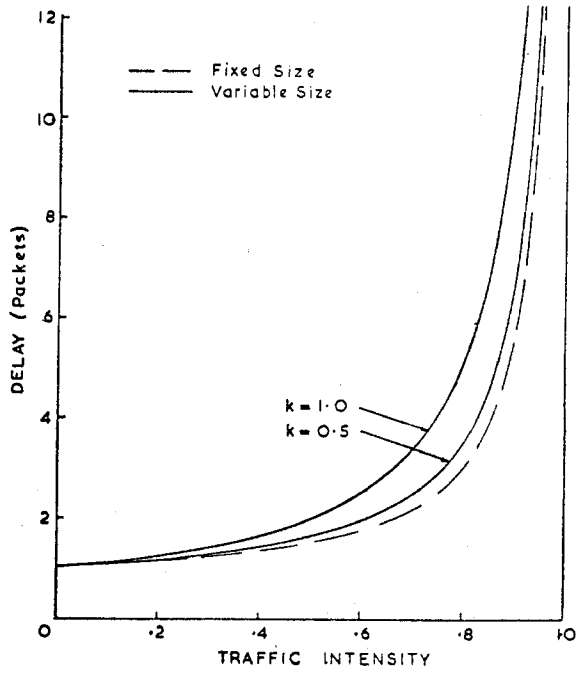


Fig. 13. Queueing delay.

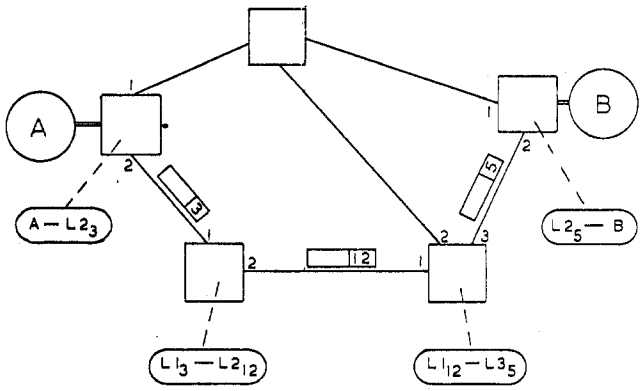


Fig. 14.

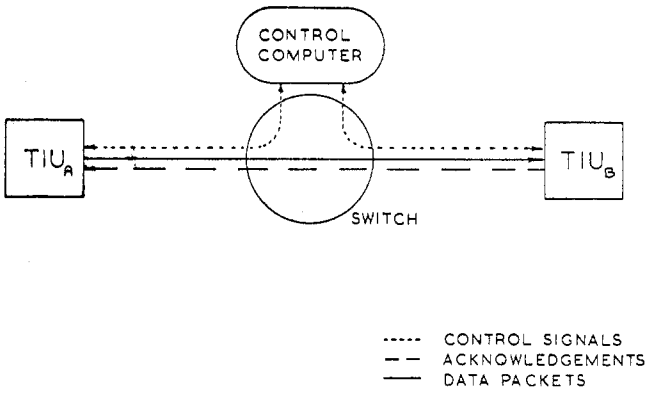


Fig. 15.

process the packets when they have been completely assembled in core. Thus the throughput of a switch is most accurately stated in terms of packets, not bytes, handled per second. If the packets are small, the machine will be able to handle fewer bytes per second.

Small packets will be one reason for exploring alternative mechanisms for packet switching. Other reasons will be to get more throughput for a given amount of switching hardware, to get better modularity so that small switches can be cheap and yet can grow gracefully as demand dictates, and to provide greater versatility in networks that employ a mixture of packet addressing and time-division techniques. Packet switches of the future may look less like general-purpose computing systems and a lot more like the synchronous machines used for switching digitized voice signals.

Incompatibilities between terminals, and variety in protocols will require the network to perform some processing on transmitted data. The ARPA terminal interface processor (TIP) is an example of what can be done. Terminals whose data are routed through that processor appear to other machines on the network as if they conformed to some standard definition of a terminal. In particular, the TIP does flow control so that a computer talking to a terminal does not have to acquire explicit knowledge of the terminal's speed. Protocol translations of this type are an appropriate application for microprocessors, and a suitable way of providing such services is to give the user the option of routing his conversation through such machines when he "dials" his call. The technique can be extended to other application areas. For example, there is emerging a standard format for credit cards. The data generated by a credit card reader could be interpreted by a service processor, and thereafter routed to the appropriate credit checking agency.

There is apparently a big role for microprocessors in data communications and it comes about because of the complexity that seems inevitably to accompany data communication protocols. By way of illustration, I shall conclude by describing how microprocessors are used in the Spider network [10].

Computers connect to Spider through an "intelligent" terminal interface unit (TIU) (Fig. 15). The transmission line is connected to one side of a TIU and the computer to the other. The computer interface with the TIU is asynchronous: one byte is transferred for each handshake on a pair of control leads. The TIU transmits packets into the network and it is the TIU's job to generate the necessary headers and checksums. Within the network the packets are routed without regard to their content; only the packet address is examined. The destination TIU has the task of checking packet sequence and checksums. If all is well, the destination TIU sends an acknowledgment packet to the transmitting TIU. The users see nothing of this and have, in effect, a network with automatic error and flow control.

The crucial element in a TIU is a small computer, called "Fly," which we made for the purpose. It is an 8-bit computer having 16-bit instructions. There are 256 words in its read-only program store and 16 words in its data store. In addition the TIU contains two data buffers each big enough to hold one packet. Since the transmission line is quite fast, 1.544 Mbits/s, Fly must also be fast. It executes one instruction in 200 ns.

It is of course not reasonable to attempt to put all the code required for a practical error control system in such a small machine. Fortunately, most of the code is required for events that occur only very infrequently. We have placed that code in the central switching machine and have arranged that a TIU can call upon the central machine for help when errors

occur. The ultimate mechanism provided for this purpose is a means by which the central machine can read any word of Fly's data store.

V. CONCLUSION

At the start of this paper we noted the wide variety of computer and terminal types which a network might have to support. It seems that there will also be variety in local distribution systems. No one distribution system is best for all circumstances.

With such variety, control and signaling become a central design issue for a data network. As yet there is no sign that these matters are even adequately understood. We must hope that a better understanding of protocol design comes soon.

The computer, besides being the source of much of this complexity, will be its solution. We are fortunate that the large-scale integration of digital circuits has already advanced to the point where we can consider putting small processors in communications equipment and terminals. I expect that microprocessors will play a big part in future computer communications systems.

REFERENCES

[1] D. Bernard, "Intercomputer networks: An overview and a bibliography," rep. NITS AD-769-232, May 1973.
[2] Various papers, *Bell Syst. Tech. J.*, vol. 54, May-June 1975.
[3] D. J. Horton and P. G. Bowle, "An overview of DATAROUTE: System and performance," and other papers in *Proc. Int. Conf. Commun.*, IEEE catalogue 75 CHO 859-9-CSCB, June 1974.
[4] N. Abramson, "The ALOHA Sytem—Another alternative for computer communications," in *AFIPS Conf. Proc.*, vol. 37. Montvale, NJ: AFIPS Press, 1970, p. 281.
[5] L. Kleinrock and S. S. Lam, "Packet switching in a slotted ALOHA channel," in *AFIPS Conf. Proc.*, vol. 42. Montvale, NJ: AFIPS Press, 1973, p. 703.
[6] ——, "Packet switching in a multiaccess broadcast channel: Performance evaluation," *IEEE Trans. Commun.*, vol. COM-23, p. 410, Apr. 1975.
[7] L. G. Roberts, "Dynamic allocation of satellite capacity through packet reservation," in *AFIPS Conf. Proc.*, vol. 42. Montvale, NJ: AFIPS Press, 1973, p. 711.
[8] W. D. Farmer and E. E. Newhall, "An experimental distributed switching system to handle bursty computer traffic," in *Proc. ACM Symp. Problems in the Optimization of Data Communications Systems*, Pine Mountain, GA, Oct. 1969.
[9] D. J. Farber and K. Larson, "The structure of a distributed computer system—The communications system," in *Proc. Symp. Computer Networks and Teletraffic.* New York: Polytechnic Inst. of Brooklyn Press, 1972, p. 21.
[10] A. G. Fraser, "Spider—An experimental data communications system," in *Proc. Int. Conf. Commun.*, IEEE catalogue 74 CHO 859-9-CSCB, June 1974.
[11] E. H. Steward, "A loop transmission system," IBM Corp., Research Triangle Park, NC, 1970.
[12] J. F. Hayes, "Performance models of an experimental computer communications network," *Bell Syst. Tech. J.*, vol. 53, Feb. 1974.
[13] J. R. Pierce, "Network for block switching of data," *Bell Syst. Tech. J.*, vol. 51, July-Aug. 1972.
[14] F. E. Heart *et al.*, "The interface message processor for the ARPA computer network," in *AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970, p. 551.
[15] C. S. Carr, S. D. Crocker, and V. G. Cerf, "HOST-HOST communication protocol in the ARPA network," in *AFIPS Conf. Proc.*, vol. 36. Montvale, NJ: AFIPS Press, 1970, p. 589.

Cont. on page 27

What's New

A four-page product sheet describing the type 700F1 RF Repeater, which offers a low-cost alternative to both active baseband and passive "billboard" type repeaters, is available from GTE Lenkurt, Inc.

An entirely new type of radio-frequency repeater for use as an intermediate link between two microwave radio terminals, the self-contained and self-powered assembly's only active elements are redundant linear amplifiers. Signals processed through the 700F1 are simply filtered and amplified but not shifted in frequency or otherwise changed.

The 700F1 is type accepted for use with GTE Lenkurt type 70F1 microwave radio terminals in the 2.11-2.20-GHz frequency band. Capacity of the RF repeater is 36 voice or data channels, plus order wire channels. It has a minimum gain of 45 dB, and a noise figure of 3.5 dB per amplifier. Total power drain is less than 4 W, and it features solar power for applications in remote areas.

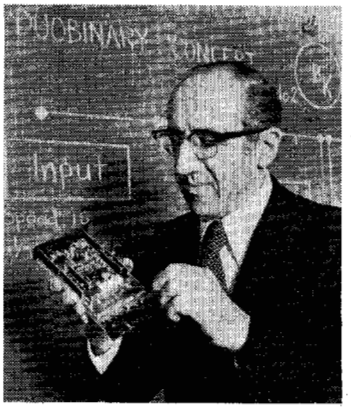
For a copy of the informative document describing the 700F1 RF Repeater, write GTE Lenkurt, Inc., Dep. C720, 1105 County Road, San Carlos, CA 94070.

Duobinary Repeater System Designed by Dr. Adam Lender of GTE Lenkurt

Development of a highly sophisticated electronic "repeater" system, which will enable telephone companies to double the number of conversations carried by digital cable links between telephone exchange offices without costly construction programs, was announced on June 17 by GTE Lenkurt Inc., a subsidiary of General Telephone & Electronics Corporation.

The new duobinary system increases to 48 from 24 the number of separate telephone conversations that can be transmitted over older cable carrier installations, normally ranging from 5 to 50 miles in length, which are widely used between telephone offices throughout the United States. These links use pulse code modulation (PCM), which converts voice signals into coded pulses for high-quality transmission.

The key elements of the system are a new type of electronic office repeater within the telephone facility which doubles the capacity of the cable by compressing the telephone signal bandwidth, and a series of line repeaters or regenerators which renew and amplify the signal.



The duobinary technique was invented in the early 1960's by Dr. Adam Lender of GTE Lenkurt's Advanced Development Laboratory (shown holding a duobinary repeater) to increase the volume of computer-type data signals that could be transmitted over an existing communications facility. Its use in the duobinary repeater system, which was designed by Dr. Lender, represents the first application of the technique to voice communications. □

Computer Communication Technology

Cont. from page 19

[16] J. Malman, "Terminal interface message processor, user's guide," NTIS AD-782 172, June 1974.
[17] A. Bjushan, "The file transfer protocol," Advanced Research Projects Agency, ARPA Network Working Group, rep. RFC 354 NIC 10596, July 1972.
[18] L. Kleinrock, W. E. Naylor, and H. Opderbeck, "A study of line overhead in the ARPANET," in *Proc. Nat. Telecommun. Conf.*, 1975.
[19] P. E. Jackson and C. D. Stubbs, "A study of multiaccess computer communications," in *AFIPS Conf. Proc.*, vol. 34. Montvale, NJ: AFIPS Press, p. 491, May 1969.
[20] H. Opderbeck and L. Kleinrock, "The influence of control procedures on the performance of packet-switched networks," in *Proc. Nat. Telecommun. Conf.*, San Diego, CA, 1974.
[21] W. W. Chu, "A study of asynchronous time division multiplexing for time-shared computer systems," in *AFIPS Conf. Proc.*, vol. 35. Montvale, NJ: AFIPS Press, 1969, p. 669.
[22] J. F. Hayes and A. G. Fraser, "Optimum packet size for data communications," *Proc. IEEE*, p. 1397, Oct. 1974.

A. G. Fraser has specialized in computer communications since joining Bell Laboratories in 1969. His work includes the Spider network of computers and a network-oriented file store. Prior to joining Bell Labs, he was at Cambridge University where he wrote the file system for the Atlas 2 computer. Dr. Fraser has a B.Sc. degree in aeronautical engineering from Bristol University and a Ph.D. degree in computing from Cambridge University. He is a member of IEEE, ACM, and the British Computer Society. □