This tutorial paper is intended for the reader who is unfamiliar with computer networks, to prepare him for reading the more detailed technical literature on the subject. The approach here is to start with an ordered list of the functions that any network must provide in tying two end users together, and then to indicate how this leads naturally to layered peer protocols out of which the architecture of a computer network is constructed. After a discussion of a few block diagrams of private (commercially provided) and public (common carrier) networks, the layer and header structures of SNA and DNA architectures and the X.25 interface are briefly described.

An introduction to network architectures and protocols

by P. E. Green

Ever since computer users began accessing the machine resources from remote terminals over twenty-five years ago, computer networks have become more versatile, more powerful, and, inevitably, more complex. Today's computer networks[1-5] range all the way from a single small processor that supports one or two terminals to complicated interconnections in which tens of processing units of various sizes are interconnected to one another and to thousands of terminals, often with various forms of special multiplexors and controllers in between.

As this evolution has proceeded, so have attempts to replace *ad hoc* methods of network design with systematic ways of organizing, understanding, and teaching about computer network details. Today there is a way of looking at networks in terms of *layered architectures* that all the experts use, but which is replete with its own jargon, and unclear and seemingly conflicting definitions, which often make it difficult to follow what is going on.

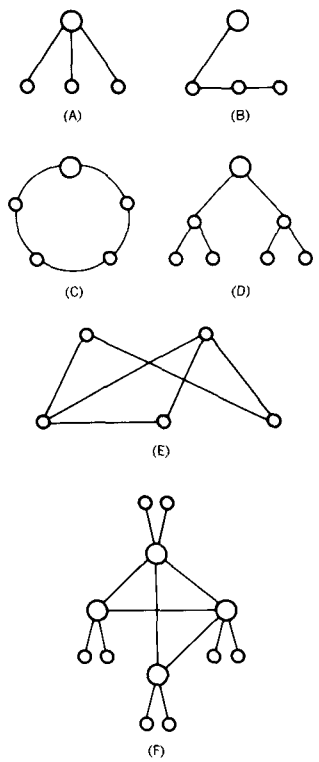
This paper aims at providing an introduction to how computer networks work from two perspectives: by briefly tracing the historical evolution of network implementations, and by summarizing some of today's layered architectures. These architectures are the rules upon which the implementations are based. In the following section, we analyze a list of the basic functions that the network provides in putting the parties that the network serves into communication with one another. This sets the stage for later discussion of layered architectures. Following these basic functions is a discussion of the evolution of private network implementations. Introduced next are the closely related *interface standards* of the common carriers. Finally, matters previously presented are re-presented in terms of the underlying layered architectures, which are expressed explicitly in terms of *protocols*.

A framework for discussing networks: the total access path between end users

The basic function to be performed by any computer network is the provision of *access paths* by which an *end user* at one geographical location can access some other end user at another geographical location. Depending on the particular circumstances, the pair of end users might be a terminal user and a remote application program he or she is invoking, two application programs interacting with one another, one application program querying or updating a remote file, and so forth. By *access path* we mean the sequence of functions that makes it possible for one end user not only to be physically connected to the other, but to actually *communicate* with the other in spite of errors of various types and large differences in the choices of speed, format, patterns of intermittency, etc. that are natural to each end user individually.

There are many ways of characterizing networks, as for example the following: (1) according to the particular application (banking, timesharing, etc.), (2) according to geography (in-plant, outplant), (3) according to ownership (public, private), and so forth. Another way of characterizing different network types is to examine the topological character of the web of transmission lines that connect together the nodes at which the different end users are located. Here, a *node* is a physical box such as a computer, controller, multiplexor, or terminal. Thus we have the various network types shown in Figure 1.

Figure 1 Six network topologies: (A) Star; (B) Multidrop; (C) Loop; (D) Tree; (E) Mesh; (F) Mesh of trees



None of these approaches really reveals what the network is actually doing. A much better scheme is to examine the total repertoire of functions that the network must provide in making up an effective access path between two end users. By doing this in an ordered way, one is in a good position to characterize the important features of both common carrier networks (of the leased,

Table 1 Access path requirements

To give a user access to processor-based resources, someone must:		
Make sure a transmission path exists.	Using	Common carrier provided links.
See that it talks in bits.	Using	Modems.
Move individual messages.	Using	DLCs.
Provide economies for intermittent use.	Using	Dial-up; multidropping, multiplexing, packet and fast circuit switching.
Send messages to correct node and correct subaddress within node. Bypass failed line or station.	Using	Addressing, routing.
Accommodate buffer size; avoid need to resend long messages.	Using	Packetizing-depacketizing.
Resolve mismatches between actual and accommodatable flow rates.	Using	Buffering, flow control.
Accommodate end-user intermittency patterns.	Using	Datagram, transaction, or session dialogue management.
Accommodate end-user format, code, language requirements.	Using	Protocol conversions.

dial, fast circuit switched and packet switched types that we shall define later in this paper) and the network designs of computer manufacturers. Two typical examples of the latter are the Systems Network Architecture (SNA) of IBM and DECNET of the Digital Equipment Corporation. Table 1 summarizes this discussion.

access path requirements

First, someone must make sure that a set of physical transmission resources (lines) exist that run from the origin node to the destination node, possibly by way of intermediate nodes. In out-plant situations (beyond one contiguous set of customer premises), this is done by *common carrier* provided links, either terrestrial or satellite.

Then one must see that the two ends of each line talk in bits using waveforms whose energy lies in a frequency range accommodated by the lines. *Modems* (modulator-demodulator units) provide this function.[6]

data link control

A capability must also be provided for making sure that the bit stream received is an error-free replica of the bit stream transmitted. This is one of the functions of *data link control* protocols[7-9] that see to it that successive groups of bits (frames) all

arrive successfully at the receiving node. This is done by checking at the receiver after each frame to see whether there has been a violation of an error check of information bits against redundant bits that were added to each frame at the transmitter. If a frame is found to be in error a retransmission is requested.

The art of Data Link Controls (DLCs) has advanced considerably from the simple but inefficient and inflexible asynchronous (start-stop) DLCs, in which precious line capacity was wasted in adding to each character fixed bit patterns for synchronization. *Synchronous* character-oriented DLCs (such as BISYNC) alleviate many of the problems with start-stop, but have proved to retain several disadvantages, notably that the same alphabet set (for example ASCII or EBCDC) and the same positions in a frame are used for line control characters, text characters, and device control characters. Thus, a character of text could be spuriously converted by noise into a character that signals the end of a frame, for example. Another disadvantage of having line control characters drawn from the same alphabet as device control and text characters is that every time a new choice of alphabet is made for the peculiar needs of some particular end user, a new and different variant of the line control results. These difficulties as well as bit efficiency problems and other problems were alleviated in the new "bit-oriented" DLCs, such as the High Level Data Link Control (HDLC), Advanced Data Communication Control Protocol (ADCCP), and Synchronous Data Link Control (SDLC), which is a subset of HDLC and ADCCP. In these protocols, line control information always occurs at its own same place in a frame. Thus the time origin of the entire frame must be knocked out of line in order for link control and data to become confused, a much less likely circumstance than to have a character in error. The line control commands are specified as bit patterns that have nothing to do with any alphabet set. High Level Data Link Control is the standard being developed by the International Standards Organization, Advanced Data Communication Control Protocol is the standard of the American National Standards Institute, and Synchronous Data Link Control Protocol is the IBM version.

addressing and routing

The next problem to be faced is to exploit the intermittent ("bursty") nature of most end user traffic by sharing the capacity of one line across many such users. If each end user were to send bit streams at a constant rate, networks made entirely of simple point-to-point lines of the right capacity would be appropriate solutions. With multiple nodes per leased line comes the need to add to the DLC certain link address fields and control characters that are used by the DLC elements at each node to avoid conflicting attempts to use the line. Multistation DLCs thus perform *time division multiplexing* or interleaving of traffic from various stations on the same line. For rarely used connections, dial-up links

offer a solution. Even more attractive economically are the new *fast circuit switched* services with minimum billing times down to a fraction of a second and fractional-second time to connect. In circuit switching, the common carrier commits a path through his system until the users finish and break the connection. In *packet switching*, which aims at dynamically sharing intermittently used transmission resources, the user sends properly addressed frames or packets to the common carrier who delivers them individually.

The action taken in response to the addressing information is of course the *routing* operation. We have just encountered this addressing/routing requirement on a single link connecting several stations. When the nodes at which the end users are located are separated by not just one line but by one or more intervening nodes and links, addressing and routing become quite elaborate, particularly if there is a multiplicity of possible routes between the two end user nodes.[10,11] In such a topologically complex network, upon failure of a node or link, *alternate path routing* provides a powerful tool for recovery.

Before leaving the subject of addressing and routing, it should be noted that a line connected to a node often carries traffic to or from more than one location within the node. To resolve the ambiguity, an intranode addressing and routing function is required in such cases.

buffering

The next function that must be provided is the buffering of incoming messages until they can be serviced, and the buffering of outgoing messages until they can be carried away by the transmission line. Limitations on available buffer size and the desire for fast response time, together with the aforementioned need to do error checking on a frame-by-frame basis (while avoiding the need to retransmit long messages), lead to the need to segment (packetize) outgoing bit streams into elements of reasonable size and similarly to reassemble (depacketize) incoming bit streams.

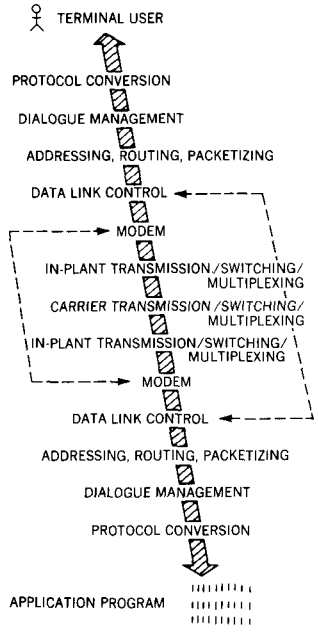
Next, the rate of flow of outgoing packets has to be regulated so as neither to overflow the buffers at the receiving station nor to leave the receiving end user waiting for more traffic. This can be accomplished by feeding back along part or all of the access path from receiving node to transmitting node special *pacing* or *flow control* signals. There are many options here. For example, the flow control signals sent from receiver to sender may simply turn off and on the emission of packets, they may tell the latter how many more packets can for the moment be safely sent, or there may be other strategies.[12]

end-user dialogue

The next function needed is a way for the end user to use all the functions just listed to set up a dialogue with the end user at the other end of the access path. The access path must be managed

so that the dialogue between end users has the *pattern of intermittency* that the end users require. For example, the pair of users might be such that a single packet should flow in just one direction. This simplest case has been termed the *datagram* type of dialogue (actually a monologue). Or there might be a tightly structured *transaction* form of dialogue in which, for example, a single packet in one direction elicits a fixed number of reply packets in the other direction. A third possibility is a *session* between end users in which the flow of packets back and forth is part of a related series of transactions. In analogy with a telephone conversation, it would be as though an access path were set up for each word, each sentence and its response, or for an entire telephone call, respectively. In managing the dialogue, there is the need not only to set up and take down the dialogue, but while it is in progress to associate related packets with one another, and to decide when an end user should listen and when it should talk.

Figure 2 Access path elements with dashed lines showing two examples of peer interaction



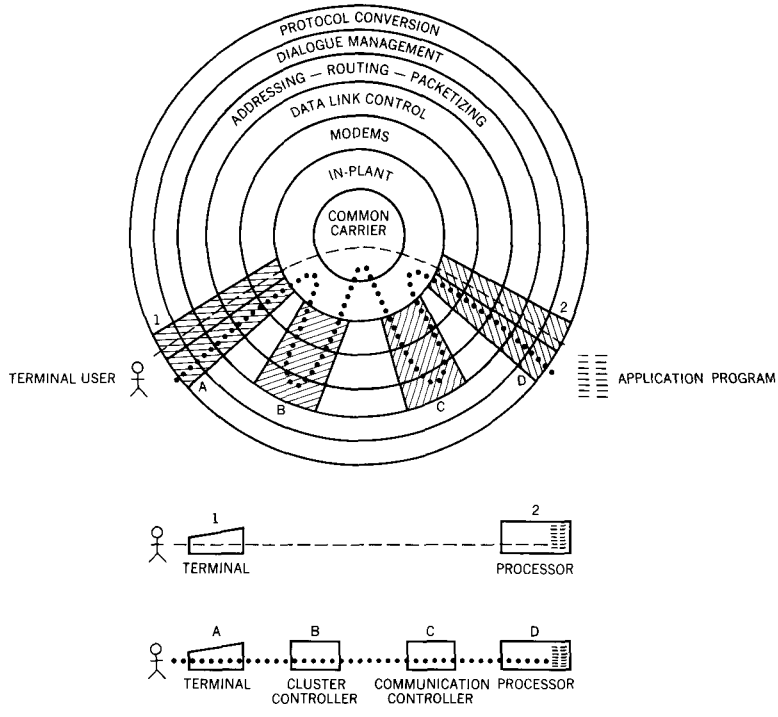
Once all the elements just listed are provided, the access path can be considered complete. This is shown in Figure 2, where the actions just discussed are listed in order. Two interesting things are immediately obvious: The elements occur in pairs and the two members of each pair talk essentially only to each other. For example, one modem talks to the other, ignoring both details of the transmission link and the meaning of bits it is handling. As another example, a DLC element ignores what its modem is doing about modulation and demodulation and also what the information field within a frame contains. A DLC interacts only with the DLC at the other end to convey the frame successfully from the sending node to the receiving node on the same line, and so forth.

peer interaction

This pairwise interaction, or *peer interaction*, of the functions we have enumerated is summarized in Figure 3, which is derived directly from Figure 2. Another way of thinking of Figure 3 is that it is in a sense the inverse of one end user's view of a network. Thus, instead of showing one end user at the center of his network, we show the transmission facilities at the center and the two end users at the periphery. The access path across the network is depicted at the bottom for illustrative cases of zero and two intermediate nodes. Note that when the access path goes through intermediate nodes, in each intermediate node it goes no higher in the layered structure than the routing operation.

Several caveats are in order about this seemingly tidy picture. For example, some generic functions can occur in more than one layer. Consider, for example, *multiplexing*, the interleaving of several traffic streams as they flow through the same path. We have already met this function in data link control. It also occurs within the common carrier transmission system. Moreover, several end users can be multiplexed on one transmission path, and, as one proceeds from a set of end users at a sending node inward

Figure 3 Peer pairs of access path elements (The modem may be absent in local in-plant connections.)



in the concentric circles of Figure 3, there is a choice of options as to the layer at which this merging might take place.

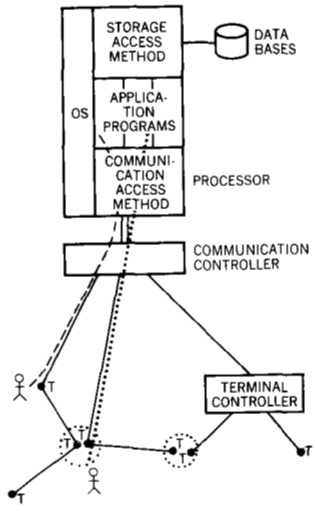
Also, there is some interlayer communication of control information within the same node. This weakens the prior statements to the effect that the two peer-related members of a given layer at the two ends of the access path ignore the contents of the bit stream handed down by the next higher layer and are also not involved in the service provided to them by the next lower layer. For example, in an intermediate node, the routing function must supply to the DLC function an address it can use in forwarding a message to the proper choice of several stations on the same link.

network control

Not shown in Figures 2 and 3 is *network control*,[13] the set of functions that do the activation and deactivation of the various portions of the access path shown, provide some of the control parameters required in their operation, and manage recovery. Network control can to various degrees be centralized (in one node) or decentralized (no single node dominant). The many network control functions that are required in forming the access path can be classified into several phases. One such rough classification is the following:

- Establishing the electrical transmission path between nodes. This may involve dial-up, which requires that appropriate telephone numbers be supplied to a participating node.
- Assigning data link addresses of stations, designating who is primary or secondary, and activating the DLC-level function.
- Establishing and updating routing tables that tell each node where to forward a message. If the message must proceed on to another node, the table must say which outgoing link to use.
- Establishing and updating directories of all end users in the network, and providing name-to-address conversion.
- Establishing and later disestablishing the datagram, transaction, or session connection out to the end users. Parameters must be supplied at each end to set up the specific dialogue convention required by the end user at that end. Queues of requests and responses within a session must be managed.
- Providing an interface to the human network manager. This includes problem determination functions, such as error reporting, testing, sending traces, and making measurements.

Figure 4 Typical teleprocessing system of the 1960s, such as System/360; *dotted* and dashed lines are access paths



In this section we have introduced the notion of layers of function as they occur in peer-related pairs to form an access path through the network. We have also mentioned the control of these functions. Before discussing how these ideas are manifested in specific network protocols of the computer manufacturers and the public common carriers, let us return to a topological view of things and examine in a little more detail what computer networks look like from that standpoint.

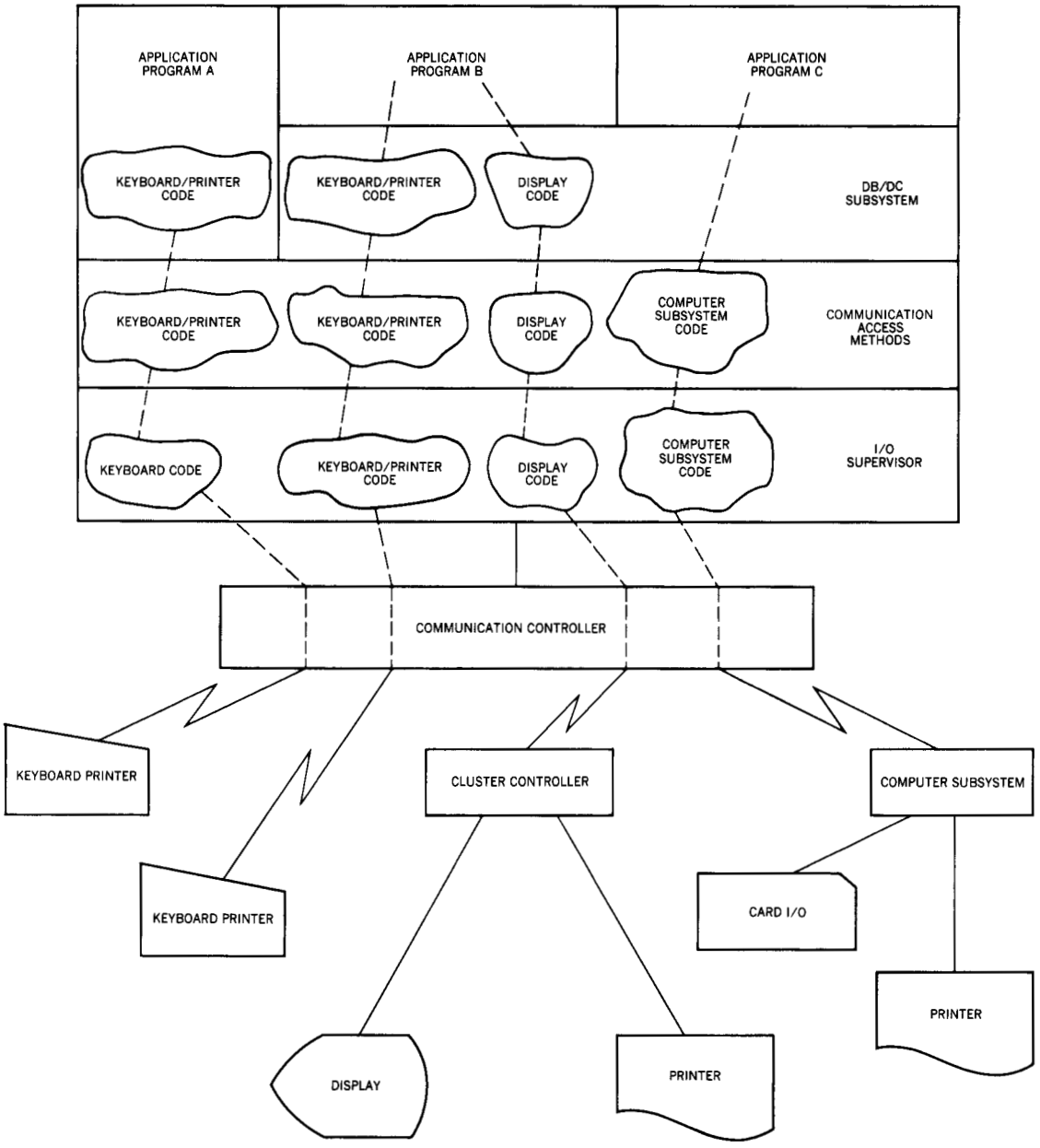
Networks of commercially provided access paths

In order to discuss the rationale of access path implementations that have been of most interest, it is instructive to sketch the historical evolution of private networks since the 1960s. Let us look first at what has happened with large computers, then minicomputers, then common carrier computer network services.

early systems

The earliest systems were single-processor batch systems that later evolved to support a few local terminals. True teleprocessing (remote access of a terminal end user to an application program in a processor) came with systems such as that shown in Figure 4, of which a typical example was the IBM System/370 running the Basic Telecommunications Access Method (BTAM). Essentially all the processing was concentrated in the central host processor, as befitted the technology available at that time. Of the various access path functions we have enumerated in Table 1, only elementary DLC-level functions were performed outboard of the host, specifically in a *transmission control unit*, which was often hard-wired and not programmable. The other functions were

Figure 5 Distribution of terminal-specific code in an early teleprocessing system



never cleanly layered, as in Figure 3, but were so spread out among the different software systems (as shown in Figure 5) that a change in the configuration of a line or its attached terminal required reprogramming in all these software systems. Terminal cluster controllers performed the device control functions, but essentially none of the communication access path functions.

What proved to be a particularly inconvenient restriction was the lack of *line sharing* or *terminal sharing*. By this is meant that, since a given line and all the terminals on it were part of the access path to only one and the same application program, if a user wanted to access two different applications (e.g., savings accounts and credit checking) he required two terminals and two lines.

Systems Network Architecture

The next step came around 1974, with systems such as that of Figure 6, of which a typical example was the System/370 with software and hardware releases referred to as Systems Network Architecture (SNA) generations 1 and 2.[14-17] The transmission control unit gave way to a programmable *communication controller* that handled all data link control and a great deal more. In the communication controller code, the host communication access method code, and the cluster controller code, a significant attempt was made to delineate function into layers, as in Figure 3. Thanks to the availability of microcomputers and lowered cost of main and secondary storage, it began to be possible to execute limited application code, including that involving significant data bases, in the cluster controllers, and (for some non-IBM realizations) in the communication controller. Most significantly, this design allowed terminals to share a line to separate applications located in the same host and to do the same thing with applications in the cluster controller. Moreover, it allowed access paths between host application programs and cluster controller application programs.

computer networking

It was soon clear that this did not go far enough. Many users had multiple processors individually serving *tree networks* such as that of Figure 6. These networks could not intercommunicate. A given terminal user frequently wanted an access path to an application *in a different host* from the one that normally served him, and it was either uneconomical or infeasible to run a second copy of that application in his own host just to provide this service. Moreover, it became desirable for one application to talk to a remote other application. These capabilities were needed for sharing processor resources among locations and for improving system availability through remote backup. These requirements led to the *computer networking* solution shown in Figure 7C, realized in Systems Network Architecture with Advanced Communication Function (SNA/ACF), which is also known as SNA-3.[18] In this arrangement, any terminal can gain an access path to any of the applications in any of the hosts. Application-to-application access paths are also supported. Figure 7C shows several of the tree structures of Figure 6 (schematized in Figure 7B, just as Figure 7A abbreviates Figure 4) connected together into a mesh of trees (as in Figure 1F) by physical paths between communication controllers. Thus, an SNA tree network can be characterized as a *hierarchical* network with network control centralized in the

Figure 6 Typical teleprocessing system of the 1970s, such as System/370 with SNA

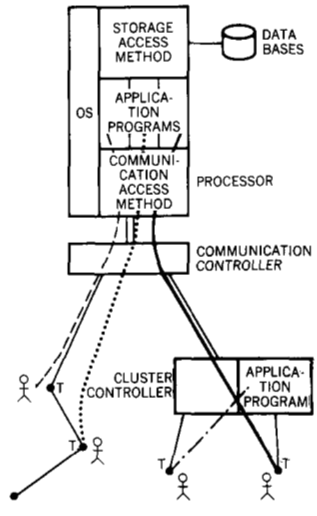
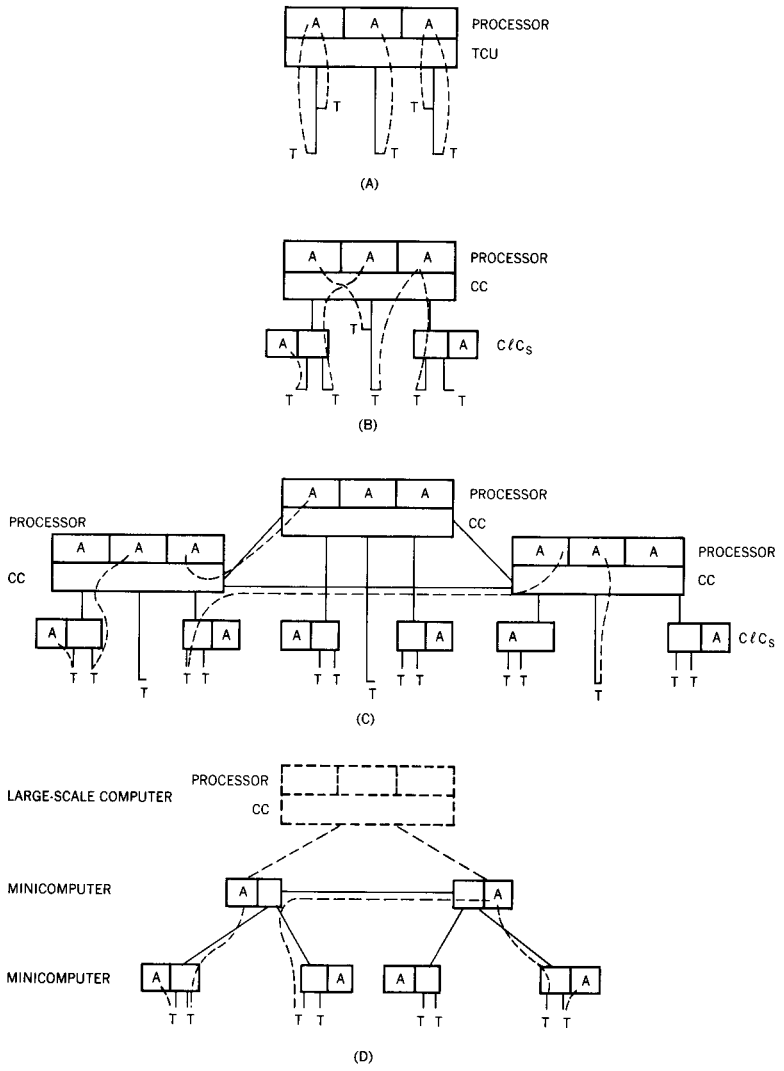


Figure 7 Schematization of access paths: (A) Abbreviates Figure 4; (B) Abbreviates Figure 6; (C) Top-down network of trees; (D) Bottom-up approach of DECNET



processor (actually in a module called the System Services Control Point (SSCP) located in the communications access method). SNA/ACF is a *hybrid* peer-hierarchical structure, that is, hierarchical within each tree or *domain* (with its own SSCP), but with peer interconnection between trees at the level of the host-attached communication controllers. Not shown in the diagram is the *multitail* capability of communication controllers, in which one such controller can support more than one processor.

minicomputer networks

In the world of minicomputers, networks have evolved somewhat differently. Originally, minicomputers were used individually for stand-alone, real-time or batch processing or for supporting a few simple terminals. When the need developed for connecting these together, it was found desirable to do this in a strictly *peer* style of interconnection rather than the peer-plus-hierarchical pattern just discussed. Peer connection had been used in the ARPA network,[19] and the flexibility of this mode of operation undoubtedly had a strong influence on minicomputer networking. In the peer mode of interconnection, no one computer does network control for the other; there is no master/slave distinction and no identifiable central control point. Network control steps are managed in each node more or less symmetrically. In principle this allows a wide range of topologies to be implemented, but requires special procedures for managing routing tables, flow control, directory functions, and recovery operations, especially when the network consists of a large number of nodes. Presumably these problems will be thoroughly understood as peer networks with decentralized network control evolve.

DECNET

One of the better known of the peer computer network designs is the DECNET offering of the Digital Equipment Corporation.[20] The DECNET design has been implemented not only for the minicomputers of the DEC product line (e.g., PDP-8 and PDP-11) but also for the high end (e.g., DECSYSTEM-10). The ultimate objective is to connect the machines together in a mesh (as in Figure 1E) or in a hierarchy (as in Figure 7D), or other arrangements. In fact, a natural user evolution for minicomputer users has been for independent users to start with stand-alone minicomputers of roughly equal power, later to connect them together, and still later to connect this set to a single large host. This *bottom-up* evolutionary pattern may be contrasted with the *top-down* pattern of network growth experienced by many users of large machines, as just described.

Networks of access paths provided by carriers

In commercially provided networks, such as the IBM and DEC offerings just described, the physical transmission-level function between nodes in the network is, of course, provided by the common carriers. The carriers have been investigating whether there is any technical reason why other functions of Figure 3 at a higher level than the transmission level might not also be provided by them—for example, the next level up, the management of error-free frame transmission using standard data link controls. The accompanying paper by Halsey, Hardy, and Powning[21] details the status of common carrier offerings and data network interfaces.

fast circuit switching

The common carriers are in fact taking steps not only to improve service at the transmission level, but to provide higher-level serv-

ices. At the transmission level, an urgent need of the data processing community has been to have dial-up service with much faster connect times and much shorter minimum billing increments than ordinary voice grade dial-up service provides. There has also been the need to improve the space-division modem-to-machine interface, such as V.24, by providing a combined space- and time-division interface of wider generality. These needs have been met by the X.21 Recommendation[22] of the international standards body, the International Consultative Committee for Telephony and Telegraphy (CCITT). The twenty-one (or fewer) wires of V.24, each performing one and only one function, are replaced in X.21 by up to eight wires of which one is used in each direction to send bit patterns for specific control functions. By this means, the repertoire of control functions is flexible and expandable. In particular, it is meant to be used for dialing and disconnecting at data processing bit stream speeds, thus serving as the basis of *fast circuit switching* common carrier networks.

packet switching

Packet switching[23] seems to have been inspired by the idea of sharing communication channel capacity across a number of users by implementing the same time-slicing philosophy that had earlier proved so successful in sharing the execution power of a single processor across many user processes. Every user node that interfaces a packet-switched common carrier makes a contract with the carrier to hand him bit streams already segmented (packetized) as we have described in the beginning of this paper, with each packet supplemented with a header saying, among other things, to which other user node he wishes the packet delivered. Widespread interest in packet switching on the part of the carriers has led them to standardize this contract in the form of the CCITT Recommendation X.25,[24] which is discussed in the next section.

The contract includes an agreement on the electrical interface, the data link control, how the remote user is to be addressed, packet size, and how the flow of packets toward and out of the carrier's network is to be regulated. The contract also includes some network control functions such as protocols for establishing and disestablishing the access path. Thus two user nodes (say A and B) each agree to exchange packets with the carrier network using the X.25 standard and the carrier agrees to deliver to B properly addressed packets from A and vice versa. The combined actions of (1) the X.25 interface of A to the network, (2) the X.25 interface of B to the network, and (3) the network, provide a full duplex path, termed a *virtual circuit*, between the higher-level function at the two nodes.

There is currently some debate over whether a special form of virtual circuit, called the *datagram* mode of operation and re-

ferred to earlier in this paper, ought to be supported under X.25. There, the duration of the contract is essentially only one packet long.

Fast circuit switching and packet switching both offer the user the economies of paying for the transmission service only to the extent that it is used. Fast circuit switching has the particular advantage over packet switching that once the transmission path has been set up, it is totally transparent. That is, except for uncontrollable random errors, the bit stream out is the same as the bit stream in for a period of time whose duration is up to the user. Packet switching, although highly nontransparent (since the user is required to adhere to what the contract says about packet length, rate of flow, header structure, etc.) does allow the carrier to offer the user more of the access path function discussed earlier in this paper than does fast circuit switching.

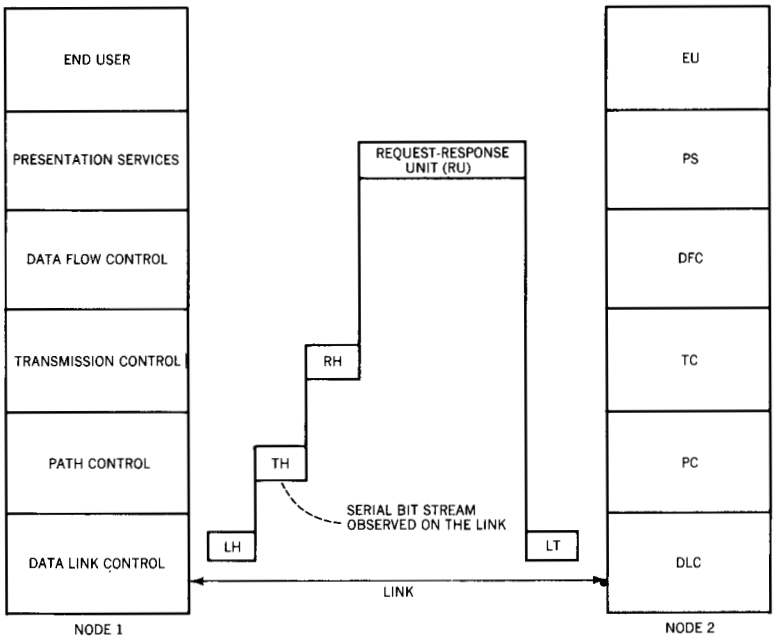
Network architectures and protocols

architecture vs implementation

The precise definition of the functions that a computer network and its components should perform is its *architecture*. Exactly by what software code or hardware these functions are actually performed is the *implementation*, which is supposed to adhere to the architecture. Both the data processing and carrier communities have expressed their network ideas in layered, peer architectures that in one way or another resemble Figure 3. Communication architecture is different from processor architecture or storage subsystem architecture in that it always involves a *pairwise* interaction of *two* parties. For example, as we have said earlier in this paper, a DLC element in one node interacts with a DLC element in another; the routing functions in two nodes interact specifically with each other, and so forth. The set of agreements for each of these pairwise interactions may be termed a protocol, and thus we find network architecture specified in terms of *protocols for communication between pairs of peer-level layers*. A network protocol consists of the following three elements: (1) *syntax*—the structure of commands and responses in either *field-formatted* (header bits) or *character-string* form; (2) *semantics*—the set of requests to be issued, actions to be performed, and responses returned by either party; and (3) *timing*—specification of ordering of events.

We shall now briefly discuss SNA, DECNET, and X.25 from this point of view, saying something about semantics and syntax and nothing about timing. All three of these structures make strict definitions of protocols between the two members of a pair of functions at the same level (although in different nodes), but leave details of interaction of adjacent layers in the same node to be decided by the implementer. They are all slightly different in the way they

Figure 8 SNA architectural layers; compare with Figure 3; Request-Response Unit (RU) is usually converted user information



assign functions to the different layers, in spite of the fact that these assignments may at first glance appear to be equivalent. The SNA and DECNET architectures are different in kind from X.25. The former two manage the access path from end to end. On the other hand, X.25 is not an end-to-end protocol, but a node-to-packet network protocol; it manages the access path from a user node to the immediately adjacent node internal to the packet network. End user to end user functions are transitory, occurring only during call establishment and disestablishment.

SNA

Figure 8 shows the layers in two SNA nodes. No intermediate nodes are shown, but in practice one or more of these could exist along the access path. Furthermore, the layers at one end could be in more than one physical box. For example, at the host end, all function could be in the host (as in the System/370, Models 115 and 125), or function roughly corresponding to Link Control and Path Control could be in the software of a separate communication controller and the rest in the host. Or it might be possible to move almost all the access path functions out to a *front-end communications processor*, leaving the host processor freer to concentrate its resources on application processing. At the terminal end, all the functions shown might be in the same box in the case of an "intelligent terminal," or almost all except the upper layer might be in the cluster controller that supports a number of "dumb" terminals.

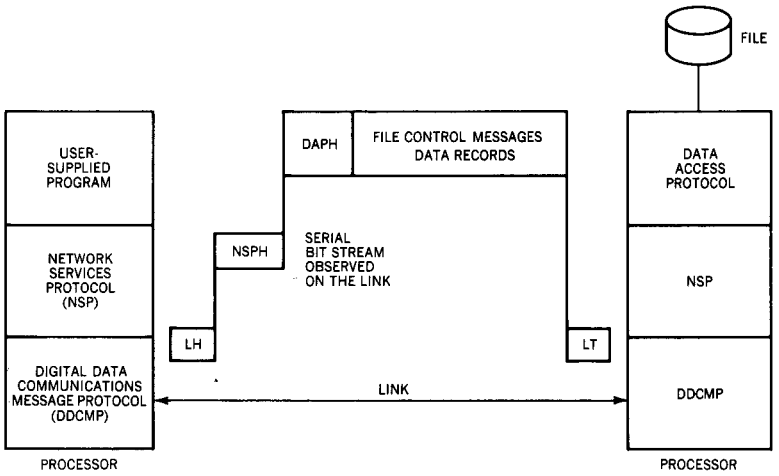
The functions of the SNA protocol layers are as follows:[1,25]

- *Data Link Control* (DLC) transfers packets intact across the noisy transmission facility. For every line attached, there is one instance of DLC or *DLC Element* (DLCE).
- *Path Control* (PC) routes incoming packets to the appropriate outgoing DLCE or to the correct point within its own node. It also does packetizing of outgoing and depacketizing of incoming messages. There is one instance of PC per node.
- *Transmission Control* (TC) manages pacing, helps manage session establishment/disestablishment, and performs a number of other functions on behalf of one of the end users. There is one instance of TC, namely, Transmission Control Element (TCE), per session per end user. Each TCE can be thought of as one end user session's "front office" to the communication network.
- *Data Flow Control* (DFC) has the function of accommodating the idiosyncrasies of message direction and intermittency demanded by the end user. Such idiosyncrasies include, for example, whether a user wants to communicate duplex or half-duplex or whether the separate messages are parts of larger units of work as seen by the end user. For example, different packets might represent different lines of text that make up a single display screen of text. There is one instance of DFC per end user session.
- *Presentation Services* (PS) define the end user's port into the network in terms of code, format, and other attributes. The pair of PS realizations in the pair of nodes have the job of accommodating, for example, the totally different interfaces seen by a terminal end user (and his supporting device control hardware or code) and the application that is being accessed. The PS layer (and other layers as well) are designed for flexibility as to the fraction of the complexity that lives at each member of the peer pair. It is thus possible to have a small or even null PS function in a simple terminal while doing most of it in the processor. As has been mentioned, it is possible optionally to have not just one but a number of concurrently operating "sub-end users" for each end user (as we have employed the term *end user*), so that a form of multiplexing takes place at the PS level that is roughly analogous to that at the DLC level.

There are a number of other SNA functions that have to do with network control,[13] but which are too detailed for a thorough discussion here. These network control functions involve a separate family of access paths that emanate from the System Services Control Point, which might be in some other node not shown, and terminate in modules (also not shown) that control the various functions shown in Figure 8.

The function of the various headers is shown in Figure 8. The zigzag strip shows the bit stream that would be observed on the

Figure 9 DNA architectural layers; compare with Figure 3



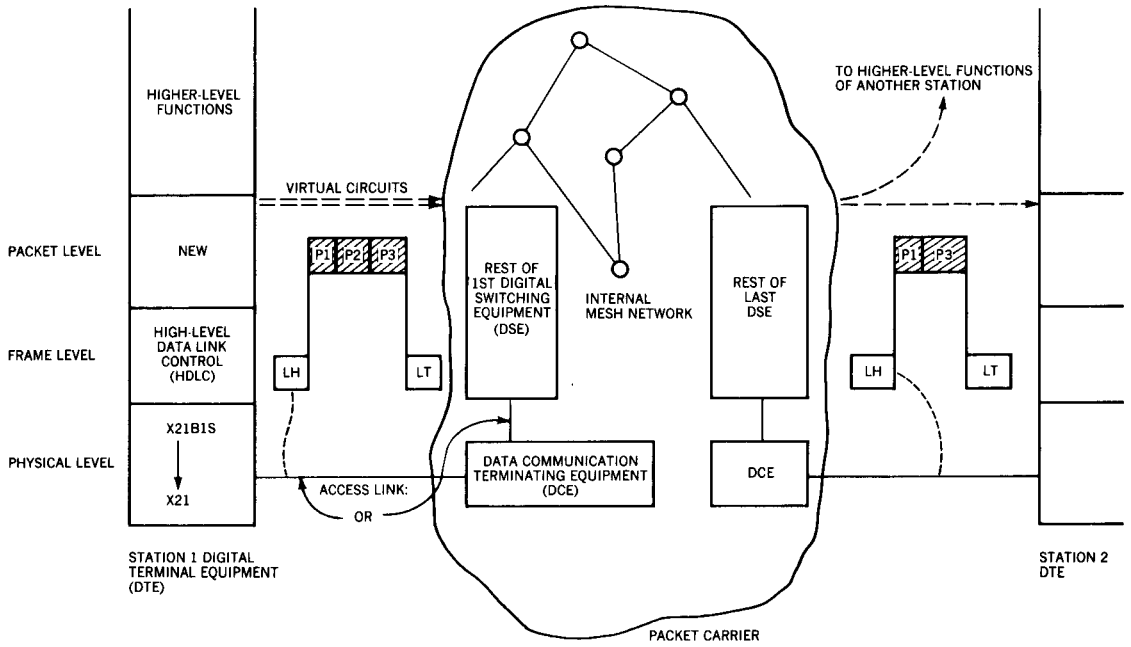
line. On an outbound message, TC adds to the RU a Request/Response Header (RH) on behalf of itself and DFC, PC adds a Transmission Header (TH), and DLC adds a Link Header (LH) and Link Trailer (LT). Inbound, each layer strips off its appropriate header (and trailer) and forwards what is left. If there is multiplexing within PS there is still another header, namely, the Function Management (FM) header, not shown. All of this illustrates the following important property of peer protocols: It is by means of the *header* that belongs to a given layer of the protocol that the interaction of the peer pair constituting that layer takes place.

DNA

The architecture on which the DECNET implementations are based is DNA (DEC Network Architecture.)[20] In the DNA set of protocols, illustrated in Figure 9, there are three basic layers, of which the bottom two are architected (i.e., the protocols are defined) and the top one is a user implementation, or (in the case of file management) vendor-supplied. The bottom two layers of DNA correspond roughly to the bottom three layers of SNA, as shown in Figure 8. The Physical Link Level is exactly the DLC level of Figures 3 and 8, the preferred realization being the DEC line control, Digital Data Communications Message Protocol (DDCMP). DDCMP is character-oriented (like BISYNC), but has many of the characteristics of bit-oriented DLCs. As in SDLC, for example, control and data characters are distinguished positionally.

The Logical Link Level layer, as defined by the Network Services Protocol, is roughly analogous to Path Control plus Transmission Control in SNA. Internal and external routing take place here, as does packetizing/depacketizing, network flow control, and the establishment and disestablishment of much of the access path. Provision is allowed for non-FIFO (first-in-first-out) arrival of packets.

Figure 10 Layers in X.25



The X.25 protocol is illustrated in Figure 10. The X.21 protocol, mentioned earlier in this paper, is specified for providing the electrical interface between the user node and the nearest Data Switching Equipment (DSE) node owned by the carrier. The X.25 specification allows for use of X.21 bis (in which the interface appears to each user as a V.24 interface) as an interim solution. As Figure 10 shows, this nearest node can be on the customer's premises (in which case the customer's access link is the multiwire X.21 set) or off-premises (in which case the access link is a telephone company provided line). In Figure 10, stations 1 and 2 are the Data Terminal Equipments (DTEs) or business machines. Packets P1 and P3 are intended for station (DTE) 2 and packet P2 is intended for some other station. The Frame Level protocol, which manages error-free transfers of strings of packets to and from the packet network, is equivalent to the DLC layer of SNA and the Physical Link layer of DNA. The Frame Level protocol uses one of two variants of HDLC. The preferred one at the moment appears to be "Link Access Protocol B," specified as the full-duplex Asynchronous Balanced mode of HDLC. Here each of the two DLC stations is neither solely a primary station nor a secondary station, but a "combined" station that is able to take responsibility unilaterally for transmission and recovery.

X.25

The Packet Level protocol produces the Virtual Circuits (VCs) referred to earlier. There may be one or many (as in Figure 10) VCs multiplexed onto one access line. These may be *fixed* (assigned upon initial subscription to the service and always in place) or *switched* (invoked *ab initio* as needed). These virtual circuits have end-to-end aspects during setup or takedown of the VC and end-to-network aspects otherwise. For example, flow control operates only to regulate traffic between the user node and the network. After a VC is initially set up, the addressing is between each end node and the network, not between end users. These are clearly end-to-network functions. But in initially establishing the VC, the end-user node must know how to address the other end-user node. This is clearly an end-to-end function.

As Figure 10 shows, there are two X.25 interfaces between each of the two customer-owned end nodes and the network. The packet carrier appears in this diagram in roughly the position where a single intermediate node would appear in Figures 8 and 9. If an SNA or DECNET system operates across an X.25 packet carrier facility, there are some divided responsibilities. For example, the SNA and DECNET implementations have specific rules about packet size, addressing/routing, flow control, internal multiplexing of flows, and recovery from error and lost- or duplicated-message conditions. When X.25 services are used, these responsibilities may overlap with those that the carrier is willing to undertake. The accompanying paper by Corr and Neal[26] discusses how these overlaps may be resolved.

Before ending this brief review of network protocols, architectures, and implementations, it should be mentioned that there is considerable interest and activity[26] in the standards bodies that have defined HDLC, X.21, X.25, etc. in standardizing even higher-level functions than those represented by the Packet Level of X.25. This is being attempted by adding four more layers above the X.25 Packet Level, making seven in all. (The bottom four layers provide end-to-end access path function roughly equivalent to the bottom two levels of DNA shown in Figure 9 and the bottom three levels of SNA shown in Figure 8.) This rapidly becomes as much of a data processing end-user issue as a communications issue, and because of the bewildering variety of special end-user needs to be accommodated, one may expect this difficult task to succeed only very slowly.

Concluding remarks

Even though networks have been growing more complicated, they should be getting easier to dissect and understand as systematic formalization and layering become more pervasive in the implementations. One reason for persistence of complexity is

that, until now, the architects have carried a heavier burden than is commonly realized of maintaining compatibility with individual software and hardware product offerings that antedated the evolution of systematic, clearly layered sets of network protocols. These earlier offerings are gradually disappearing or in later releases are adhering more and more to the strict terms of the architecture. The modularization means that new ideas ought to be more easily incorporated without producing system-wide disruptions. Continuing research will provide such new ideas.

ACKNOWLEDGMENT

The author thanks R. J. Cypser, A. Endres, R. F. Steen, and an anonymous reviewer for their helpful comments.

CITED REFERENCES

1. R. J. Cypser, *Communications Architecture for Distributed Systems*, Addison-Wesley Publishing Company, Reading, MA (1978).
2. P. E. Green, Jr. and R. W. Lucky (Editors), *Computer Communications*, IEEE Press, New York (1975).
3. M. Schwartz, *Computer-Communication Network Design and Analysis*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1977).
4. D. W. Davies and D. L. A. Barber, *Communications Networks for Computers*, John Wiley & Sons, Inc., New York (1973).
5. L. Kleinrock, *Queuing Systems, Volume II*, John Wiley & Sons, Inc., New York (1976).
6. J. R. Davey, "Modems," *Proceedings of the IEEE* **60,** 1284-1292 (November 1972). See also J. R. Davey, "Modems," reprinted in P. E. Green, Jr. and R. W. Lucky (Editors), *Computer Communications*, IEEE Press, New York (1975), pp. 188-196.
7. R. J. Cypser, *Communications Architecture for Distributed Systems*, Addison-Wesley Publishing Company, Reading, MA (1978), Chapter 11.
8. J. P. Gray, "Line control procedures," *Proceedings of the IEEE* **60,** 1301-1312 (November 1972). See also J. P. Gray, "Line control procedures," reprinted in P. E. Green, Jr. and R. W. Lucky (Editors), *Computer Communications*, IEEE Press, New York (1975), pp. 212-223.
9. B. W. Stutzman, "Data communication control procedures," *Computing Surveys* **4,** No. 4, 197-220 (December 1972).
10. M. Schwartz, *Computer-Communications Network Design and Analysis*, Prentice-Hall, Inc., Englewood Cliffs, NJ (1977), Chapters 2 and 11.
11. L. Kleinrock, *Queuing Systems, Volume II*, John Wiley & Sons, Inc., New York (1976), Chapter 6.
12. L. Kleinrock, "On flow control in computer networks," *Proceedings of the International Communications Conference (Toronto)* (June 1978), pp. 27.2-27.5.
13. J. P. Gray, "Network services in Systems Network Architecture," *IEEE Transactions on Communications* **COM-25,** No. 1, 104-116 (January 1977).
14. J. H. McFadyen, "Systems Network Architecture: An overview," *IBM Systems Journal* **15,** No. 1, 4-23 (1976).
15. P. G. Cullum, "The transmission subsystem in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 24-38 (1976).
16. W. S. Hobgood, "The role of the Network Control Program in Systems Network Architecture," *IBM Systems Journal* **15,** No. 1, 39-52 (1976).
17. H. R. Albrecht and K. D. Ryder, "The Virtual Telecommunications Access Method: A Systems Network Architecture perspective," *IBM Systems Journal* **15,** No. 1, 53-80 (1976).

18. *Introduction to Advanced Communication Function*, Order Number GC30-3033, IBM Corporation, Data Processing Division, White Plains, New York 10504.
19. P. E. Green, Jr. and R. W. Lucky (Editors), *Computer Communications*, IEEE Press, New York (1975). See reprinted papers on the ARPA network.
20. G. E. Conant and S. Wecker, "DNA, an architecture for heterogeneous computer networks," *Proceedings of the Third International Conference on Computer Communications (Toronto)* (1976), pp. 618-625.
21. J. Halsey, L. Hardy, and L. Powning, "Public data networks: Their evolution, interface, and status," *IBM Systems Journal* **18,** No. 2, 223-243 (1979, this issue).
22. H. Folts, "X.21, the international interface for new synchronous data networks," *Proceedings of the International Communications Conference (San Francisco)* (June 1975), pp. 1.15-1.19.
23. L. Roberts, "Data by the packet," *IEEE Spectrum* **II,** No. 2, 46-51 (February 1974).
24. A. Rybczinski, B. Wessler, R. Despres, and J. Wedlake, "A new communication protocol for accessing data networks—the international packet mode interface," *AFIPS Conference Proceedings, National Telecommunications Conference* **45,** 477-482 (June 1971).
25. *SNA Format and Protocol Reference Manual*, Order Number SC30-3112, IBM Corporation, Data Processing Division, White Plains, NY 10504.
26. F. P. Corr and D. H. Neal, "SNA and emerging international standards," *IBM Systems Journal* **18,** No. 2, 244-262 (1979, this issue).

Reprint Order No. G321-5093.