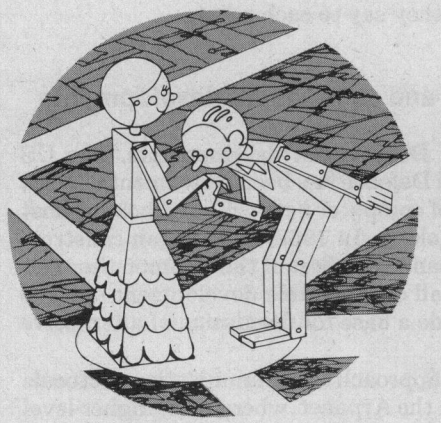
The growth of networking—more and more computers of different makes communicating with each other—will motivate the greater standardization of host-to-host protocols.



The Evolution of Host-to-Host Protocol Technology

David C. Walden
Alexander A. McKenzie
Bolt Beranek and Newman Inc.

The past decade has been one of great activity in the development of computer communications networks. No aspect of this activity has been more difficult than the development of the procedures and conventions used by "host" computers attempting to communicate with each other across a network. (We apply the term "host" to distinguish them from the machines often used to make up the communications subnetwork.)

These host machines are almost infinitely various. They can be made by different manufacturers and use different operating systems. They can be relatively accommodating, or extremely unaccommodating, to the addition of the computer communication function. They can be attached to communication subnetworks providing either a very sophisticated or a very primitive communications capability. They can be made by a vendor who believes in supporting communication with other vendors' equipment or who hopes to discourage it. Here, we discuss host communication procedures and conventions, tracing their evolution and relating them to other network technologies.

When computer communication development first entered the present era (with the beginning of construction of the Arpanet[1] in 1968), researchers unaccountably used the word "protocol" to mean a set of communication procedures and conventions.* Thus, we use the term "host-to-host protocol" for the subject of this paper.

All communication protocols of interest appear to have at least two common elements: (1) the establishment of path conventions and (2) the definition of path control procedures. Path conventions deal with timing considerations (e.g., communication with a 300-baud terminal), encoding or representational considerations (e.g., using the ASCII character set), and methods for multiplexing control information and data on the same communication path (e.g., provision of certain distinguishable control characters used to signal that the direction of transmission should be reversed). Path control procedures establish a virtual communications medium between the communicating entities; this medium has certain desirable characteristics which may not be possessed by the physical medium (e.g., provision for addressing, synchronization, error control, and flow control).

*Some writers[2] have quibbled with this use of the word, calling it a misuse of English language. However, dictionary justification can be found for use of the word, at least in retrospect:

"The word 'protocol' has been borrowed from the parlance of conventional social behavior to describe the orderly exchange of information between separate pieces of equipment."[3]

"In a human context, 'protocol' is defined as a code of diplomatic etiquette and precedence. In communications technology a protocol is a set of rules of etiquette and precedence governing communications between two computing elements, which may be terminals or computers. A protocol is a logical abstraction of the physical process of communication.

"An additional connotation of the word 'protocol' in the data communications field is that protocols define interactions among 'similar' or 'equal' entities, while interface refers to interactions among dissimilar entities. For example, it might be said that there is an interface between a Host computer and a network switching node, while there are node-to-node and Host-to-Host protocols. On the other hand, many carrier interfaces are structured in such a way that it is (almost or entirely) possible to interconnect two users either by interfacing them both to the carrier or directly to each other, without any difference in the user systems in the two cases. When this is true, the distinction between an interface and a protocol is virtually meaningless."[4]

0018-9162/79/0900-0029$00.75

Pre-Arpanet*

The protocols which evolved in accordance with the early point-to-point or star networks treated the "network" as a simple data pipeline. Further, these protocols were typically implemented in a single software module which handled all aspects of the protocol. As communication circuits began to be shared among many terminals, protocols were added to the application software to resolve conflicts between the terminals. The addition of front-end processors led to the centralization of protocol support in these processors, but required the definition of a new interface protocol between the mainframe and the front end. The development of terminal cluster controllers, data concentrators, and remote front ends each resulted in a change to the architectural model and the number of protocols needed.

The introduction of "common user" communications networks—single systems capable of simultaneously serving many applications—required a new level of protocol to be developed. With such networks it is no longer practical to view the communication facility as a simple data pipeline which can be tapped at two or more points and used for a specific application. Rather, the user must provide addressing information to the network, specifying where each unit of information is to be delivered. The advantage of such a system is that it makes possible the interconnection of any set of terminals and computers. However, this new possibility requires that terminals and hosts follow standard protocols in order to make sense of what they say to each other.

*As one reads the following sections, it may be useful to glance occasionally at Figure 1, which presents the authors' view of the evolution of host-to-host protocols.

The Arpanet and subsequent developments

US Defense Department developments. The US Department of Defense has been instrumental in the development of computer networks and host-to-host protocol technology. In 1968, DoD began construction of the Arpanet, the first of the common user networks.[5] We shall describe this development in some detail to provide a base for discussion of alternative approaches.

A "layered" approach to communication protocols was adopted in the Arpanet, wherein the higher-level protocols use the services of the lower-level protocols.[6] As shown in Figure 2, the lowest host layer is the IMP-to-host protocol, which specifies the electrical interface, link control, and message format for communication between an IMP (i.e., interface message processor—the switching node in the Arpanet)

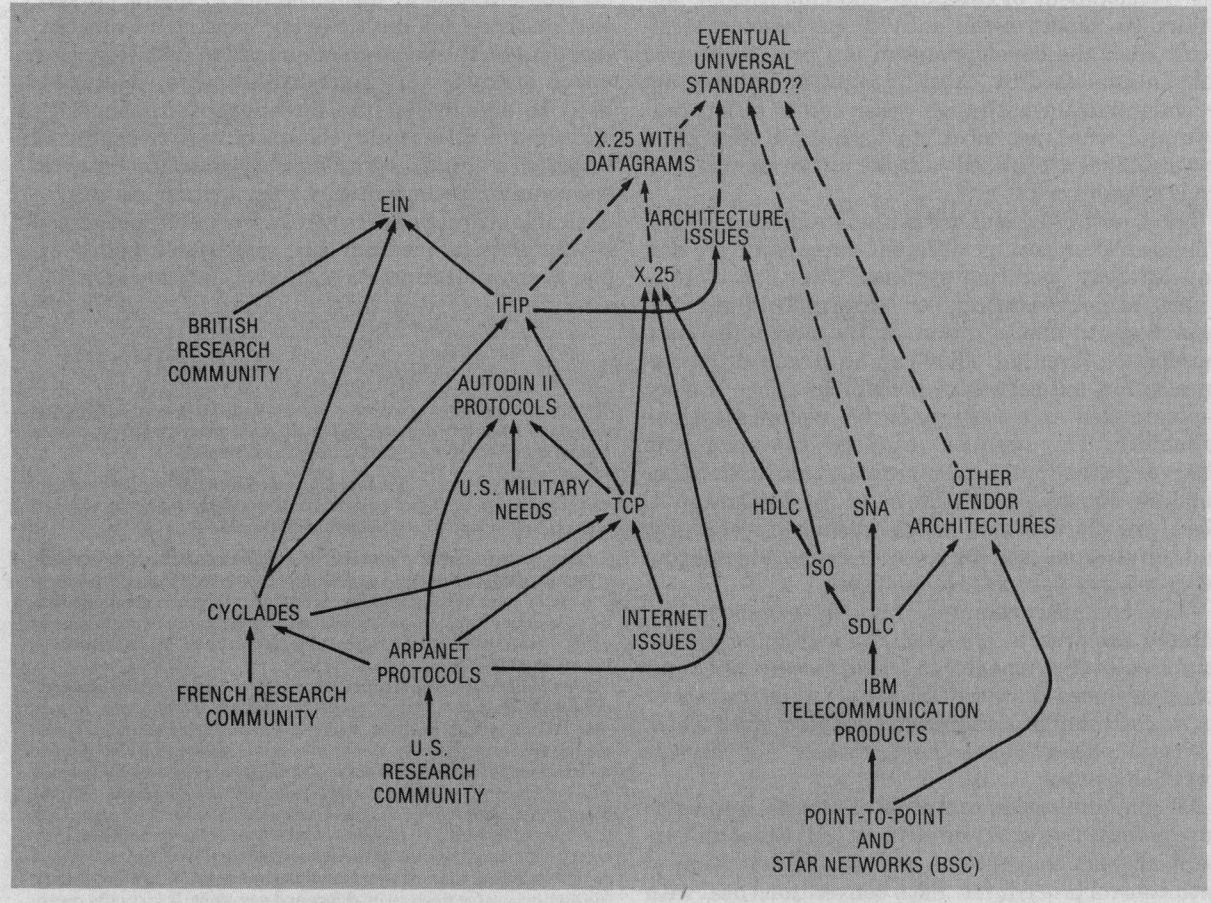


Figure 1. Approximate evolution of host-to-host protocols.

and a host. The next layer (called the host/host layer in the figure) specifies methods for establishing communication paths between hosts, managing buffer space at each end of a communication path, etc. Next, the initial connection protocol, or ICP, specifies a standard way for a remote user (or process) to attract the attention of a network host, preparatory to using that host.* The ICP provides a function analogous to the user pressing the attention button at a terminal connected directly to a host. The next layer is the telecommunications network or Telnet protocol, which was designed to support terminal access to remote hosts. Telnet is a specification for both a network standard terminal and a protocol for communicating between such a standard terminal and a host. The next logical protocol layer consists of function-oriented protocols, two of which—the file transfer protocol, or FTP, and remote job entry protocol, or RJE—are shown in the figure. Other function-oriented protocols defined for the Arpanet include a graphics protocol and a digital speech protocol. Finally, at any point in the layering process, it is possible to superimpose ad hoc protocols.

Figure 3 illustrates the Arpanet host/host protocol.[7] Before two processes can communicate, a *data connection* must be set up between a socket in one process's monitor (a *socket* is an element in a network-wide name space into which each monitor maps its own internal name space) and a socket in the other process's monitor. For two processes to make a data connection, each process makes a connection request to its own monitor. The two monitors then exchange these requests via messages over a *control connection,* a special logical connection between each pair of hosts that is always reserved for control messages. If both monitors are in agreement, the data connection is established. This new data connection exists until it is explicitly terminated, again using inter-monitor control messages over the control connection. During the "life" of a data connection, many messages may be sent from the socket at one end to the socket at the other. However, since the data connection exists over a series of many messages, a mechanism is provided to stop the flow of messages when a receiving process's host is overloaded. The mechanism consists of a buffer allocation system along with a requirement for the sending process to stop transmitting when it has exhausted the allocated capacity. A control messge is used to replenish buffer allocations. Multiple data connections may be in effect simultaneously. The Arpanet host/host protocol provides what have come to be called "virtual circuits" between communicating hosts. Specifically, a virtual circuit provides for the transmission of separate information units, at a rate acceptable to the receiver, with extremely low probability of loss, duplication, or improper ordering of any information unit.

The Arpanet host/host protocol has several flaws. Because of constraints initially imposed by the host/IMP protocol, the host/host protocol permits only one message to be traversing a connection at any one time; this requirement to "stop and wait" for an end-to-end acknowledgment severely limits the bandwidth of a single connection. The allocation mechanism in the host/host protocol is incremental, with no provision for resynchronization in the case of error. Because the communication subnetwork provides error control over the individual subnetwork circuits through use of checksums, the host/host protocol has no provision for end-to-end error control. The host/host protocol assumes that messages arrive in order,

*ICP is not a layer, strictly speaking, but it is adequate for this paper to think of it as one.

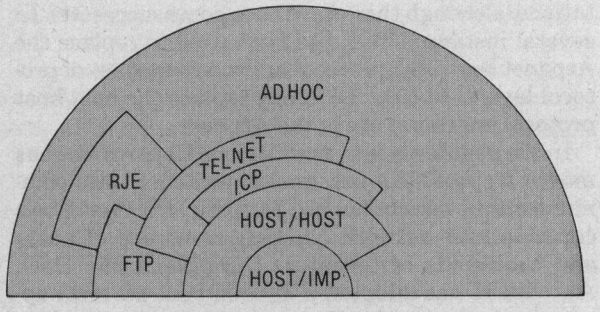


Figure 2. The Arpanet layers of protocol.

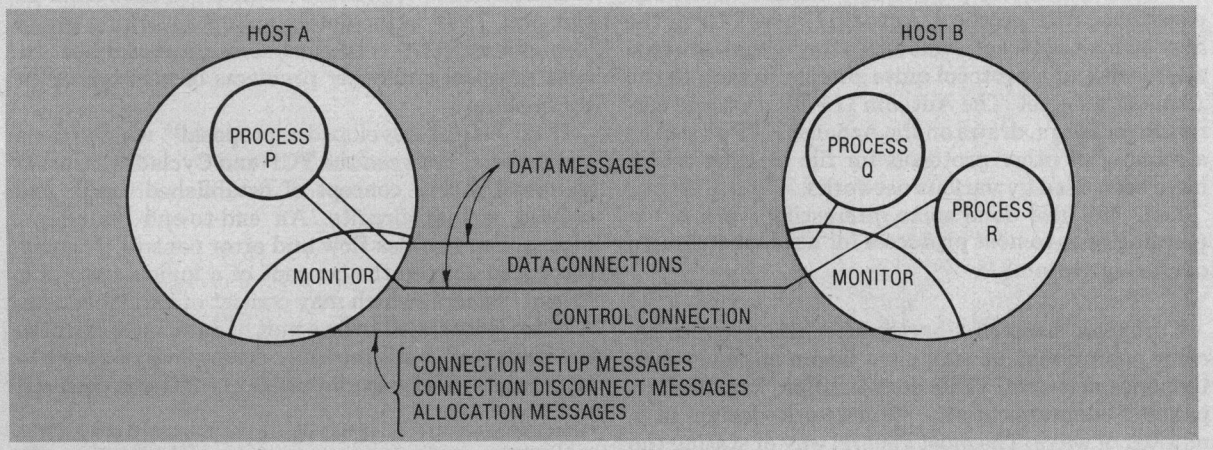


Figure 3. The Arpanet host/host protocol.

which can be overly constraining. The connections are simplex, when full duplex are usually preferred. It has been recognized that the Arpanet ICP is more complex than is usually necessary, since most hosts provide mechanisms which could be used instead. The provision for specifying host addresses is quite limited. There are other problems.[8]

In spite of the above, the Arpanet is reliable enough and the host/host protocol useful enough to permit widespread, adequately reliable use of the Arpanet. In fact, many of the above-mentioned problems could be fixed quite simply; in the Arpanet[9] and other similar networks, some have been fixed to a degree.

The desire to have a host-to-host protocol which is suitable for use in a multi-network environment, and which does not have the problems of the Arpanet host/host protocol, led to the specification of TCP.[10] Still assuming a layered environment and virtual circuits between communicating hosts, TCP accommodates communication between networks using different basic message sizes. It provides for end-to-end error control, retransmission, and data reassembly and reordering. It provides for more than one message to be in transit at a time. It provides resynchronization for the end-to-end flow control mechanism. It provides for more general host addressing conventions. It makes minimal assumptions about the network or networks with which it is used. TCP also features full duplex connections and no ICP. Of course, TCP's generality does not come without a cost: It puts a potentially large processing burden on the host; it does not lend itself to efficient implementations (although there have been some successes). In several instances TCP has been used to replace the Arpanet host/host protocol in the Arpanet set of protocol layers. In time TCP may replace the host/host protocol in general use in the Arpanet.

In the past few years, the Defense Department has moved to provide a new packet-switched data communications network called Autodin II.[11] It is to be a common user network handling hundreds of hosts and thousands of terminals throughout the DoD. Autodin II has adopted a layered host protocol approach similar to the Arpanet's. Naturally, the bottom level specifies the interface to the Autodin II switches and is not particularly similar to the Arpanet host/IMP protocol. Autodin II uses TCP as the host-to-host protocol. It also uses the virtual terminal approach and a protocol quite similar, in fact, to the Arpanet's Telnet. The Autodin II FTP protocol, currently in design, draws on the Arpanet FTP as well as a number of other protocols for file transfer which have been used by various networks.

DoD has also done some interesting work in expanding host-to-host protocols for use in distributed operating systems.[12]

European research. Shortly after the Arpanet became operational, development began on the French Cyclades network.[13] This design differs from the Arpanet's communications subnetwork design in a number of ways. The most general way of stating the difference is to say that where the Arpanet tries to provide all the service it can for the user hosts (e.g., dynamic routing, complete reliability, perfect message ordering) and in a number of cases falls short of its goals, Cyclades leans wherever possible toward making things simpler for its communications subnetwork (called Cigale[14]) by putting the burden on the hosts. The designers reasoned that since the subnetwork could not do many of these things perfectly, the hosts would end up doing them anyway—so there was no need to bother to have the subnetwork do everything. Where the Arpanet subnetwork follows a virtual circuit approach, Cigale follows a "datagram" approach in which the communications subnetwork does not provide flow control, proper sequencing, and error control. The hosts have to provide these functions for themselves in the host-to-host protocol in order to obtain virtual circuit service.* The operation of a datagram packet network is more closely analogous to a high-speed electronic postal system than to a switched circuit system: Addressed packets are input at many points, mixed, and delivered; some are occasionally lost, and the delivery sequence is only loosely related to the input sequence. There is no call establishment concept in such a system. This philosophy has been carried over into larger European research networks.

Datagram service provides the advantage of supporting applications in which low delay (or variance in delay) is more important than the high level of reliability and message accountability provided by virtual circuits. Examples of such applications are digitized voice transmission, continuous sensor operation for process control, and other real-time operations. Datagrams may also be suitable for very short independent communications such as data base inquiries from credit terminals. For these applications the hosts in a datagram environment would not choose to implement a host-level virtual circuit.

IFIP recommendation. During the early 1970's, with a few packet networks implemented and many more in advanced stages of design, IFIP—the International Federation for Information Processing—recognized the desirability of interconnecting packet networks, both public and private, to permit communication between hosts on different networks. To this end, IFIP immediately set up a working group (designated WG6.1) to consider a universal host-to-host protocol and other problems of network interconnection.

IFIP WG6.1 developed a proposal[15] which was a compromise between the TCP and Cyclades ideas. It is based on the concept of established, used, and cleared virtual circuits. An end-to-end "window" mechanism handles flow and error control. The protocol also defines the concept of a logical data unit (called "letter") which may consist of many packets. A letter is intended to be a unit of data meaningful to the hosts, e.g., a file record or a complete response to an inquiary. The maximum size of a letter is declared

*Note that these functions are similar to the ones TCP provides; TCP and Cyclades were developed somewhat in parallel, in reaction to the flaws of the Arpanet host/host protocol.

during virtual circuit establishment and is an element of the flow and error control mechanism. The IFIP proposal assumes no more than a simple datagram service from any of the interconnected networks.

Even as the IFIP host-to-host protocol was being designed, events were bypassing it. DoD adopted TCP for future use and CCITT—Consultative Committee for International Telegraphy and Telephony, the major world-wide standards body—adopted a vastly different standard (as discussed in the next section). While the European Informatics Network[16] and some private organizations have adopted variants of the IFIP host-to-host protocol proposal, and both the International Standards Organization and CCITT are studying it, it is unlikely ever to have widespread direct applications. IFIP and WG6.1's main contribution is that they may provide a forum where researchers can compare views and thoroughly thrash out technical issues, as opposed to the atmosphere of compromise which prevails at the standards organizations.

CCITT's standards. Since public packet networks were being built in many countries (including the US, Canada, England, France, Spain, and Japan), CCITT defined a standard interface for packet networks, called Recommendation X.25,[17] in 1976. The X.25 protocol defines all three protocol levels needed at this interface: the physical link level (called X.21), the logical link control (provided by a subset of HDLC), and the "packet level interface" (called X.25 Level 3) which defines packet format and control procedures in terms of "virtual circuits." The X.25 interface's goal is to provide host-system designers with a circuit-oriented service almost identical to the physical circuits familiar to them, but with such low error rates and with enough flow control that higher-level and end-to-end protocols need not be concerned with error and flow control. Control packets between host and network provide for call establishment and call clearing; data packets are sent over established circuits. Flow and error control over each virtual circuit are handled by HDLC-type "window" mechanisms applied to the host-to-network portion of the virtual circuit. Once the communication subnetwork has acknowledged a packet on Level 3 at the input end, the network accepts responsibility for delivering a correct copy of the packet to the destination address, in the correct order with regard to other packets sent over the virtual circuit.

It is interesting that the thinking of DoD and European researchers has moved away from the Arpanet's virtual circuit approach toward the datagram, while at the same time CCITT has adopted an even stricter virtual circuit approach than Arpanet's. There are several reasons for this difference in direction. On the one hand, many researchers support the datagram approach because they feel it is purer. In addition, DoD and European researchers are in a position to implement the protocol they feel is best. CCITT, on the other hand, is dominated by workers with telephone circuit experience, and is faced with communications protocols originally designed in the era of point-to-point and star networks.

Several European government-sponsored carriers, each enjoying a monopoly position in its own country, have made a commitment to national data communications through packet switching and X.25. Thus, computer and terminal manufacturers wishing to do business in those countries will need to support X.25. Indeed, most major US manufacturers have already moved toward such support in the European market. This should in turn lead to wide availability of X.25 support in the US; American packet carriers are already committed to providing X.25 interfaces. This should eventually force non-X.25 networks to provide at least an X.25 network interconnection interface. Recognizing the inevitable, many researchers (e.g., in IFIP) are now concentrating on extending X.25 to support a datagram option. The fact is that both virtual circuit and datagram services are valuable for different applications—an international standard protocol would do well to support both.

Both virtual circuit and datagram services are valuable—an international standard protocol would do well to support both.

Vendor architectures. All of the above-mentioned research and development relating to host-to-host protocols has been aimed at facilitating the interconnection of products from many vendors. Of course, many of the major vendors have developed their own network architectures, both to maintain a level of support for older products (thus enabling customer network evolution) and to provide unique facilities or modes of operation further locking customers into using the vendor's own products. IBM, with its Systems Network architecture, or SNA,[18] is the dominant vendor in this category.

We will not attempt to describe SNA or the other vendor network architectures here, since they have not tended to be technically innovative. The literature provides comprehensive descriptions[19]; Falk[20] compares SNA with other architectures. The technical merits of SNA aside, IBM's market dominance gives almost every one of their products the status of a *de facto* standard. It seems clear that regardless of the progress made in international and national standardization of communication protocols, SNA protocols will be equally important in the United States, with many manufacturers designing their products for SNA compatability.

ISO's model. Computer communication's widespread availability has focused the need for greater standardization of communications protocols. In particular, a study committee of the ISO—International Standards Organization—has recently formulated the model[21] shown in Figure 4. This model identifies seven hierarchical protocol levels, divided into two

groups—the "providers of transport service," responsible for moving the data from one place to another, and the "users of transport service," the programs responsible for generating and processing the data.

Host-to-host protocols, as discussed here, do not fit neatly into the ISO model. This is understandable—we have been describing an evolutionary process, but ISO is trying to provide a tidy model. The protocols we have been examining are involved with the host portion of Level 4 and the operating system portion of Level 5. Fitting them to the ISO model, then, shows a blurring of these two levels, indicative of problems with either the protocols or the model. The primary purpose of dividing the architecture into a number of levels or layers is to provide a clear separation of implementation details. Thus, for example, it should be possible to make changes to the physical link without affecting Levels 2-7, or to change a packet-switched network to a circuit-switched network without affecting Levels 4-7. This is not possible when a single protocol is involved in multiple levels.

Technical issues

The remainder of our discussion addresses a number of technical issues in host-to-host protocol design and gives the authors' personal view of how well they are understood or solved. The reader wishing to study these issues in more detail should see the literature.[3,22,23,24]

There is now a good understanding of the possible network environments with which the protocol designer might have to deal.[3] In the best of all worlds the network would accept messages of arbitrary length and deliver them in order, with complete reliability. (Even under such an unrealistically ideal model, the protocol designer would still have to deal with questions of connection establishment and termination, flow control, multiplexing, and addressing.) If the network does not deliver messages in order, the host-to-host protocol designer must face issues of message reordering and reassembly. This complicates some of the above-mentioned questions and causes new problems, such as having enough buffer space available to reassemble messages (and avoiding deadlocks if one does not). The possibility of out-of-order messages also tends to make the separation of data and control information necessary, and this in turn can cause problems in synchronizing the control and data information. If a size limit is placed on data segments within the network, the host-to-host protocol designer must face the problem of breaking messages into segments and then recognizing the segments and reassembling them. It is also necessary to distinguish the last segment in a message.

Some of the most difficult problems result if the network is not perfectly reliable and the possibility exists for an occasional message to get lost or duplicated. Then there can be problems with end-to-end retransmission and duplicate detection, identification of messages and message segments, connection establishment and termination, deadlocks in the flow-control mechanism, and so forth.

Three principles most workers in the field now consider fundamental to protocol design are layering, interfacing, and symmetry.[3] The separation of protocols into layers, some of which are side by side and independent of each other and some of which support others, makes each layer simpler and allows different services to be offered at different layers. Layering depends on well-defined and stable interfaces between the layers. With appropriate layers and inter-

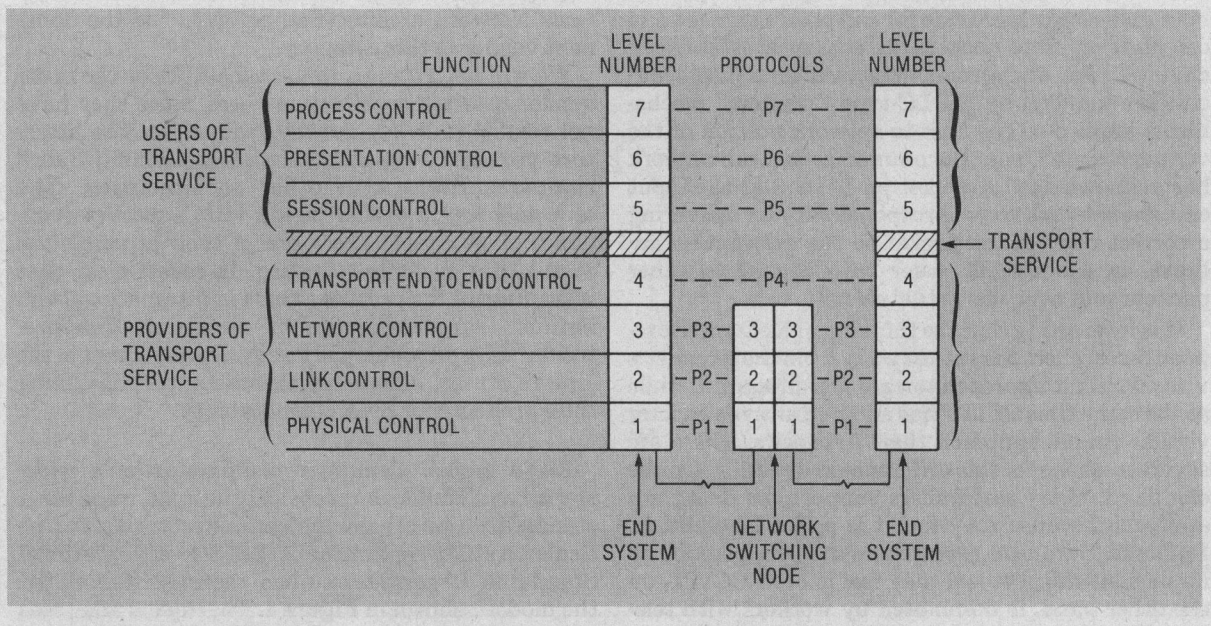


Figure 4. The ISO model.

faces, it is possible in many instances to change a layer without disruption of network services, thus accommodating network growth and evolution. Layering and interfaces also simplify program debugging and operational trouble shooting. While there are some exceptions, workers have found it useful to have symmetric protocols, that is, protocols with a participant at each end, with neither participant a master nor slave.

Another now well-accepted conclusion is that data and control information should be separated whenever possible, that is, they should flow over separate virtual channels. If they cannot, it is impossible to control the data flow in many cases.

The use of sequence numbers to order messages and to detect lost and duplicate messages has been widely implemented; it is also the subject of considerable theoretical study (probably because it is one of the few areas of computer communications that is analytically tractable without resort to unrealistic assumptions and simplifications). The establishment and termination of virtual circuits or connections have similarly been subjected to thorough practical experience and analysis. Synchronization of sequence numbers and connections is also well understood. In all of these cases, one can often implement a relatively simple version of the sequence number, connection, or resynchronization system without requiring implementation of everything the theory says is necessary in the extreme case.

Other well-understood techniques include the use of checksums to detect errors, and positive acknowledgments, time-outs, and retransmissions to provide error-free data transmission. Developments in these areas have represented a major improvement over the past (these are two of the few areas where there is a relevant past). Checksums, as formerly used, were commonly insufficient to catch many errors, and negative acknowledgment systems were used which simply did not work in the face of possible errors in control data.

Finally, the process of multiplexing several conversations on the same connection, or over the same interface, is well-understood. Multiplexing is efficient but tends to cause interference between conversations.

There are a number of other technical areas which are less well-understood or for which there is no agreement on the proper solution.

Over the decade since computer networks have come into use, there has been little commonality in the way various host operating systems provide the network inteface to user programs. Because they were mostly constructed at a time when intercomputer communication was a remote prospect, operating systems typically have no provision for conveniently adding a general communication function and providing a general interface to the user programs. At its most primitive, the user interface simply emulates an existing peripheral device, e.g., reading and writing to a certain hypothetical disk or tape drive or even to a hypothetical card-reader/card-punch combination.[25] The technique of reading and writing to a special file in the computer's file system is one step up from emulation of a physical device; in cases where handling files and devices is already done uniformly, providing network access in the same way is particularly convenient. Vendor network architectures tend to make the user interface look like an existing communication access method. In cases where it has been possible to make changes to the operating system, a quite powerful user interface can be provided through the addition of a set of new system calls such as "open a connection," "listen for a request to open a connection," "send a message," "close a connection," and "obtain the status of a connection." Another method of providing the user interface is to integrate it into the system's inter-process communication method,[26,27] which makes communication between processes across networks as natural as communication between processes within a single system.

How much of the host-to-host protocol is implemented within the operating system, and how much is implemented at user level, also varies greatly from system to system. In some instances the function is even implemented out-board in a communiations front end of one sort or another. There are a number of ways the host-to-host protocol function has been integrated into an operating system and the user interface provided.[25,28,29] Our hope is that convenient, efficient host-to-host protocol functions and user interfaces will be integral parts of new operating systems. There are signs that this will be the case,[30,31] but the issue is far from settled.

Our hope is that convenient, efficient host-to-host protocol functions and user interfaces will be integral parts of new operating systems.

The problem of naming the source and destination of a given bit of data has many possible solutions. Some believe that the naming should be hierarchical, as are the telephone national and regional area codes. Others favor variable-length names. Fixed vs. dynamic assignment of names is another issue. Should objects have multiple names? Which should have names? There are various answers to each of these questions, and known possible implementations for each answer. Which is best is still undecided.

Addressing is an area receiving increasing attention, but which still requires more work. Methods are being investigated, for instance, for providing broadcast addressing, or a host with two addresses on a network.

The question of connecting networks has also received great study and some implementation, but so far there is no agreement on the proper method.[32] TCP[10] was originally designed with internetting in mind, and X.25 claims to support internetting. to some extent. The fact is that networks will be connected to each other. Hoping for a single internet

host-to-host protocol is probably unrealistic. We will have to face up to the ad hoc interconnection of somewhat dissimilar networks. On the other hand, the data communications standards organizations have been active in this area and it is likely that computer network interconnection will never require the linking of systems as dissimilar as was the case in the railway, electrical power, and telephone systems.

Flow control is a problem with a number of well-understood solutions.[33] However, there is normally a tradeoff against throughput—i.e., if one requires taut, complete flow-control, one must at times run below the maximum possible throughput for the channel. On the other hand, permitting maximum throughput occasionally allows the flow to get out of hand; the process of controlling it can actually result in a lower average throughput. This has proved to be a problem especially when the same sequence number mechanism is used for flow control and error control.

The most up-to-date methods of ensuring communication privacy work end-to-end. Thus, there is a natural interaction with the host-to-host protocol. For instance, how does one decide which encryption key to use for each virtual circuit? Substantial progress is being made here; several experimental and prototype systems have been developed and are being used.

While computer networks have now been in existence for a number of years, one of the original goals of such networks—resource sharing—is still only commonly achieved in very primitive forms such as remote use of a time-sharing system. More sophisticated resource sharing, such as automatic selection of the most available computer, is still very much in the development (if not research) stage. Even the low-level mechanisms for providing resource sharing are at a primitive stage of development, e.g., the mechanisms to update a distributed file system or to open a connection to any one of several qualified hosts.

Moving into the research domain, we find the question of proving or verifying the correctness of protocols. Host-to-host protocols are so complicated and program faults so common that there is a large potential payoff in being able to prove the protocols correct, in spite of the great inherent difficulty of the task. This is the subject of Carl Sunshine's article, appearing in this issue.

Another problem area, possibly not difficult to solve but receiving little or no attention, is performance and function monitoring of host-to-host protocols. This lack of investigation exists in spite of the faults and frequently poor performance these protocols are known to have.

One last area for study involves the journal-keeping capability of most commercial transaction processing systems. A journal is a log of all transactions passing through a data processing system; it can be used for recovery in the event of system failure. There is no current work toward providing a journal to track host-to-host transactions within a computer network. Until such a capability is available, some institutions which would profit most from networks will resist their use, having come to expect the benefits journal-keeping can provide.

Future directions

We see continued evolution of host-to-host protocols in the next few years, especially in research environments. But even as protocols evolve, international standards and vendor architectures will be settling fast and the majority of computer network users will offer greater and greater resistance to change and further evolution.

We have already mentioned the possible expansion of the X.25 standard to support virtual circuits and datagrams. This will bring the research community one step closer to the standards organizations. We see little evidence that the US Department of Defense will move toward any international standard, yet once Autodin II is in operation there will be great pressure to provide interfaces to various X.25 networks and to permit X.25 hosts to attach. Also, while DoD work has preceded the formation of standards, by the next generation (late 1980's) the standards will be firmly in place (if slightly changed) and there will then be no excuse for DoD to not use X.25.

The beginning of the era of distributed computation is the end of the era of one-vendor shops.

As for the vendor architectures, we think that they will have to move toward the standards, even though this has not been the case (especially with IBM) in the past. The beginning of the era of distributed computation is the end of the era of one-vendor shops. The average institution will have multiple computers—they will be from different makers and the owner will want them to communicate. This will be a force too powerful for any one group to successfully resist. ■

Acknowledgments

Individuals too numerous to mention have contributed to the development of host-to-host technology. We have drawn heavily on the work of others in the preparation of this paper, especially that of V. Cerf, J. McQuillan, and L. Pouzin. R. Brooks and D. Schwarzberg helped with the preparation of the manuscript.

References

1. P. M. Karp, "Origin, Development, and Current Status of the ARPA Network," *Proc. COMPCON 73*, pp. 49-51.

2. P. J. Denning, "Operating Systems Principles for Data Flow Networks," *Computer,* Vol. 11, No. 7, July 1978, p. 90.
3. J. M. McQuillan and V. G. Cerf, *A Practical View of Computer Communications Protocols,* IEEE Computer Society, Long Beach, Calif., 1978 (IEEE-CS Catalog No. 201).
4. A. A. McKenzie, manuscript in preparation for publication, Jan. 1979.
5. F. E. Heart, R. E. Kahn, S. M. Ornstein, W. R. Crowther, and D. C. Walden, "The Interface Message Processor for the ARPA Computer Network," *AFIPS Conf. Proc.,* Vol. 36, 1970 SJCC, pp. 551-567.
6. S. D. Crocker, J. F. Heafner, R. M. Metcalfe, and J. B. Postel, "Function-Oriented Protocols for the ARPA Computer Network," *ibid.,* pp. 271-279.
7. A. A. McKenzie, "Host/Host Protocol for the ARPA Network," *ARPANET Protocol Handbook,* Defense Communications Agency, Dec. 1978.
8. D. C. Walden, "Host-to-Host Protocols," *International Computer State of the Art Report No. 24: Network Systems and Software,* Infotech, Maidenhead, England, pp. 287-316.
9. *Specifications for the Interconnection of a Host and an IMP* (May 1978 revision), BBN Report No. 1822.
10. V. G. Cerf and R. E. Kahn, "A Protocol for Packet Network Interconnection," *IEEE Trans. Communications,* Vol. COM-22, No. 5, May 1974, pp. 637-648.
11. P. Sevcik, "Autodin II Subscriber Access Protocols and Interfaces," *NTC '77 Conf. Record,* Vol. 3, pp. 37:6-1 to 37:6-6.
12. H. C. Forsdick, R. E. Schantz, and R. H. Thomas, "Operating Systems for Computer Networks," *Computer,* Vol. 11, No. 1, Jan. 1978, pp. 48-57.
13. L. Pouzin, "Presentation and Major Aspects of the Cyclades Computer Network," *Proc. 3rd ACM-IEEE Communications Symposium,* Nov. 1973, pp. 80-87.
14. L. Pouzin, "Cigale, the Packet Switching Machine of the Cyclades Computer Network," *Proc. IFIP Congress,* Aug. 1974, pp. 155-159.
15. V. Cerf, A. McKenzie, P. Scantlebury, and H. Zimmermann, "Proposal for an International End to End Protocol," *Computer Communication Review,* Vol. 6, No. 1, Jan. 1976, pp. 63-89.
16. D. L. A. Barker, "A European Informatics Network," *Computer Communication Review,* Vol. 5, No. 3, July 1975, pp. 12-15.
17. *Orange Book,* Vol. VIII-2, CCITT X.25, 1977, pp. 70-108.
18. J. P. Gray, "Network Services in Systems Network Architecture," *IEEE Trans. Communications,* Vol. COM-25, No. 1, Jan. 1977, pp. 104-116.
19. R. G. Berglund, "Comparing Network Architectures," *Datamation,* Vol. 24, No. 2, Feb. 1978, pp. 78-85
20. G. Falk, "A Comparison of Network Architectures: The ARPANET and SNA," *AFIPS Conf. Proc.,* Vol. 47, 1978 NCC, pp. 755-763.
21. ISO/TC 97/SC 16 N34, "Provisional Model of Open Systems Architecture," *Computer Communications Review,* Vol. 8, No. 3, July 1978, pp. 49-61.
22. L. Pouzin and H. Zimmermann, "A Tutorial on Protocols," *Proc. IEEE* (special issue on packet communication networks), Vol. 66, No. 11, Nov. 1978, pp. 1346-1376.

23. *Proc. Symposium on Computer Network Protocols,* Liege, Belgium, Feb. 1978.

24. L. Garlick, R. Rom, and J. B. Postel, "Reliable Host-to-Host Protocols: Problems and Techniques," *Proc. Fifth Data Communications Symposium,* Sept. 1977, pp. 4-19 to 4-25.

25. J. P. Johnson and J. M. Heap, "Designing an X.25 Network-to-Host Interface," *Computer Communications,* Vol. 1, No. 1, Feb. 1978, pp. 13-18.

26. D. C. Walden, "A System for Interprocess Communication in a Resource-Sharing Computer Network," *CACM,* Vol. 15, No. 4, Apr. 1972, pp. 221-230.

27. E. Akkoyunlu, A. Bernstein, and R. Schantz, "Interprocess Communication Facilities for Network Operating Systems," *Computer,* Vol. 7, No. 6, June 1974, pp. 46-54.

28. J. B. Postel, *Survey of Network Control Programs in the ARPA Computer Network,* MITRE Report MTR-6722, June 1974.

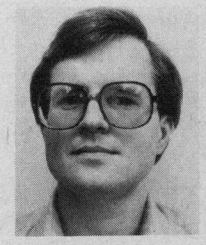
29. J. Davidson, W. Hathaway, N. Mimno, J. Postel, R. Thomas, and D. Walden, "The ARPANET TELNET Protocol: Its Purpose, Principles, Implementation, and Impact on Host Operating System Design," *Proc. Fifth Data Communications Symposium,* Sept. 1977, pp. 4-10 to 4-18.

30. D. Retz, "Operating System Design Consideration for a Packet Switching Environment," *AFIPS Conf. Proc.,* Vol. 44, 1975 NCC, pp. 155-160.

31. W. G. Probst and G. Bochman, "Operating System Design with Computer Network Communication Protocols," *Proc. Fifth Data Communications Symposium,* Sept. 1977, pp. 4-19 to 4-25.

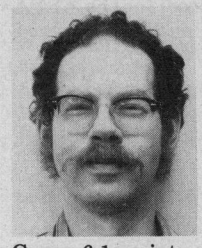
32. V. G. Cerf and P. T. Kirstein, "Issues in Packet-Network Interconnection," *Proc. IEEE* (special issue on packet communication networks), Vol. 66, No. 11, Nov. 1978, pp. 1386-1408.

33. *Proc. Conf. on Flow Control in Computer Networks,* Versailles, France, Feb. 1979.



David C. Walden is employed by Bolt Beranek and Newman Inc., of Cambridge, Massachusetts, where he has been active in the design, implementation, and operation of several computer networks. He has been editor and served on the editorial board of several computer communications journals.

Walden received a BA from San Francisco State College.



Alexander A. McKenzie is also employed by Bolt Beranek and Newman Inc., where he has been involved with computer networks for the past decade. He was for several years manager of Arpanet operations and maintenance. He has participated in the design of many computer network protocols, and been active in protocol standards work. Presently chairman of IFIP Working Group 6.1 on international packet switching and editor of the ACM *Computer Communication Review,* he is an associate guest editor of an upcoming special issue of the *IEEE Transactions on Communication* on network architecture and protocols.

McKenzie received a BS from Stevens Institute of Technology and an MS from Stanford University.