ington, D. C.: Spartan, 1970, pp. 569–579.
[34] D. Kleitman, "Methods of investigating connectivity of large graphs," *IEEE Trans. Circuit Theory* (Corresp.), vol. CT-16, pp. 232–233, May 1969.
[35] P. Krolak, W. Felts, and G. Marble, "A man-machine approach toward solving the traveling salesman problem," *Commun. Ass. Comput. Mach.*, vol. 14, pp. 327–335, May 1971.
[36] S. Lin, "Computer solutions of the traveling salesman problem," *Bell Syst. Tech. J.*, vol. 44, pp. 2245–2269, 1965.
[37] T. Marill and L. G. Roberts, "Toward a cooperative network of time-shared computers," in *Fall Joint Computer Conf.*, *AFIPS Conf. Proc.* Washington, D. C.: Spartan, 1966.
[38] B. Meister, H. R. Muller, and H. R. Rudin, "New optimization criteria for message-switching networks," *IEEE Trans. Commun. Technol.*, vol. COM-19, pp. 256–260, June 1971.
[39] E. F. Moore and C. E. Shannon, "Reliable circuits using less reliable relays," *J. Franklin Inst.*, vol. 262, pp. 191–208, 1956.
[40] NAC 3rd Semiannual Tech. Rep. Project "Analysis and optimization of store-and-forward computer networks," Def. Doc. Cen., Alexandria, Va., June 1971.
[41] NAC 4th Semiannual Tech. Rep. Project "Analysis and optimization of store-and-forward computer networks," Def. Doc. Cen., Alexandria, Va., Dec. 1971.
[42] G. S. Pan, "Communications information system," *Telecommunications*, pp. 19–23, June 1970.
[43] J. Pierce, "A network for block switching of data," *Bell Syst. Tech. J.*, to be published.
[44] L. G. Roberts, "Multiple computer networks and inter-computer communications," presented at the Ass. Comput. Mach. Symp. Operating Systems, Gatlinburg, Tenn., 1967.
[45] ——, "A forward look," *Signal*, vol. 25, pp. 77–81, Aug. 1971.
[46] ——, "Resource sharing networks," presented at the IEEE Int. Conv., New York, N. Y., Mar. 1969.
[47] L. G. Roberts and B. Wessler, "The ARPA computer network development to achieve resource sharing," in *Spring Joint Computer Conf.*, *AFIPS Conf. Proc.*, 1970, pp. 543–549.
[48] ——, "The ARPA computer network," in *Computer Communication Networks*, F. Kuo and N. Abramson, Eds. New York: Prentice-Hall, 1973, to be published.
[49] B. Rothfarb and M. Goldstein, "The one terminal Telpak problem," *Oper. Res.*, vol. 19, pp. 156–169, Jan.–Feb. 1971.
[50] R. L. Sharma and M. T. El Bardai, "Suboptimal communications network synthesis," in *Proc. Int. Conf. Communications*, vol. 7, pp. 19-11–19-16, 1970.
[51] K. Steiglitz, P. Weiner, and D. J. Kleitman, "The design of minimum cost survivable networks," *IEEE Trans. Circuit Theory*, vol. CT-16, pp. 455–460, Nov. 1969.
[52] R. Van Slyke and H. Frank, "Reliability of computer-communication networks," in *Proc. 5th Conf. Applications of Simulation* (New York, N. Y.), Dec. 1971.
[53] ——, "Network reliability analysis—I," *Networks*, vol. 1, no. 3, 1972.
[54] ——, "Network reliability analysis—II," to be published in *Networks*, 1972.
[55] L. P. West, "Loop-transmission control structures," *IEEE Trans. Commun.*, vol. COM-20, pp. 531–539, June 1972.
[56] V. K. M. Whitney, "A study of optimal file assignment and communication network configuration in remote-access computer message processing and communication systems," *Dep. Elec. Eng.*, Univ. of Michigan, Ann Arbor, Rep. 02641-2-T, SEL 48, Sept. 1970.
[57] R. S. Wilkov, "Analysis and design of reliable computer networks," *IEEE Trans. Commun.*, vol. COM-20, pp. 660–678, June 1972.
[58] B. Yaged, "Minimum cost routing for static network problems," *Networks*, vol. 1, pp. 139–172, 1971.
[59] M. L. T. Yuen *et al.*, "Traffic flow in a distributed loop switching system," presented at the Symp. Computer Communications, Apr. 4–6, 1972.

Resource-Sharing Computer Communications Networks

ROBERT E. KAHN, MEMBER, IEEE

Invited Paper

Abstract—**The development of resource-sharing networks can facilitate the provision of a wide range of economic and reliable computer services. Computer-communication networks allow the sharing of specialized computer resources such as data bases, programs, and hardware. Such a network consists of both the computer resources and a communications system interconnecting them and allowing their full utilization to be achieved. In addition, a resource-sharing network provides the means whereby increased cooperation and interaction can be achieved between individuals. An introduction to computer-to-computer networks and resource sharing is provided and some aspects of distributed computation are discussed.**

I. Introduction

THE INTERACTION between computers and communications has steadily developed over the last two decades. While many universities, government agenies, and business firms prefer to make use of thier own computers, an increasing number of people are using communication facilities to access commercial computer services [1]. Time-sharing and batch processing services are offered in most major United States cities or are accessible via telephone circuits, and communication charges for local telephone access to these services are, in general, substantially lower than the computer charges. As the use of computer services increases, the demand for reliable and low-cost means of communicating over wide geographic areas also increases.

For many years, networks of interconnected computers have been planned or under study, and more recently several have been under development [16], [18], [24]. A common objective underlying the interest in these networks has been to demonstrate that economic savings or increased capabilities are possible by sharing computer or communication resources. Program access to specialized data bases is an important example of resource sharing in a computer-to-computer network.

The growing usage of these data-processing services and the objective of sharing resources raise communication issues far more extensive than those of achieving increased capability and lower costs in the telephone network, or developing improved communication services [2]. They involve a num-

Manuscript received June 7, 1972; revised August 1, 1972. This work was supported by the Advanced Research Projects Agency of the Department of Defense under Contract DAHC-71-C-0088.

The author is with Bolt Beranek and Newman Inc., Cambridge, Mass. 02138.

ber of complex regulatory issues, the need for common methods of access to and interchange between data-processing systems, the pooling of computer resources for increased utilization and reliability, the provision of specialized services, data conferencing, and so forth. A set of associated regulatory issues involving telecommunications policy has been raised and is under intensive study. Are separate common-carrier data networks desireable or not? What is the most effective way to plan for interconnection of data networks and how should their usage be tariffed? (Some of these questions are discussed in the paper by S. L. Mathison and P. M. Walker, this issue.)

It is too early to accurately predict in what way this interaction of computers and communications is likely to evolve. The technology is changing rapidly, and regulatory policies are in flux. If communication costs are not to dominate the overall cost of using remote data-processing services, technological advances must allow communications at substantially lower per bit costs than are possible with the current switched telephone plant.

In this paper, we present one view of computer communications network development and explore a number of the important issues in distributed computation which have arisen. This paper is neither a completely general treatment of computer networks, nor a full case study, but rather it contains selected aspects of the two. The reader will no doubt be able to identify where general considerations give way to specific ones derived primarily from the author's experience with the development of the Advanced Research Projects Agency Computer Network (ARPANET) [7], [8]. It is impossible for this to be an exhaustive treatise, or even a comprehensive one, and no such attempt is made.

II. Distributed or Concentrated Resources

Many economic factors support the conclusion that geographic "clustering" of computers is a desirable strategy for computer service organizations [9]. One possible advantage is better equipment utilization due to the pooling of resources. Clustering implies that a single maintenance staff (which is often underutilized) and scarce system personnel can support more equipment, more reliably, and that space, auxiliary equipment, and overhead can be consolidated. In fact, several commercial time-sharing firms have already chosen to concentrate their computer resources in a small number of geographic areas. In contrast, however, many individual research or development machines under private or government ownership are distributed throughout the United States. The valuable resources on many of these machines provide a strong incentive for them to be made available to users and computers at many other locations [10], [11], [33].

The location of computers at a few geographic locations requires that both local and remote users be provided with an economic and reliable way to access the service. The switched telephone network currently appears to be a poor candidate to provide the long-distance communications service. In addition to being considerably more costly than local service, the error performance of long-distance circuits is degraded from shorter circuits and is insufficient for many computer applications. In addition, frequent disconnections, busy signals, etc., during peak traffic hours often make its usage inconvenient. These factors, coupled with user desires for increased bandwidth, lower setup times, and more suitable tariffs, have encouraged several vendors to competitively enter the common-carrier market [28].

The tariffs and the technical characteristics of the circuit-switched telephone network reflect the nature of voice communication requirements that are quite different from those of computer communications. Due to the "bursty" nature of computer traffic and the extremely low utilization of a typical voice-grade circuit by a terminal, a substantial portion of the data-communications capacity in a circuit-switched system is simply not used. This results in inefficient utilization of telephone company resources from the users' point of view. Frequency- or time-division multiplexing techniques have been usefully applied for deriving individual channels, but the statistical nature of computer traffic makes fixed allocation strategies such as these inefficient or unacceptable.

On the other hand, statistical multiplexing techniques allow these circuit resources to be more widely shared, at the possible expense of occasional delays in transmission. Message switching employs a generalized form of multiplexing for a network environment that allows all circuits to be shared among all users in a statistical fashion without being allocated in advance. (Multiplexing is the subject of a separate paper in this issue by D. R. Doll.)

This has been the motivation for the development of new communication systems as well as combined computer communication networks. The construction of both common-carrier data-communication systems and "private" networks (using leased common-carrier facilities) is a natural outcome of the need for economic and reliable communication between users and geographically distributed computers. In addition to potential cost savings, many of these networks provide error control, as well as asynchronous operation, local echoing, speed, and code conversion, which are better suited to data communication with computers than use of the telephone network alone. A reevaluation of the tariff structures for data communication has recently been undertaken by the FCC, and efforts are being made to provide the public with data-communications service having lower error rates, smaller service charging intervals, and faster setup times than the switched voice network currently provides.

III. Computer-to-Computer Communications

A computer network is a complex collection of many types of resources, including data bases, programs, operating systems, and special-purpose hardware, all of which are capable of being accessed from any other resource in the net. Computer-to-computer communication is necessary to achieve effective resource sharing, but the ability to transfer information between machines does not automatically result in useful machine-to-machine interactions. Aroused by the exciting possibilities in using multiple machines, system designers have recently begun to provide the major technical effort required to achieve effective computer-to-computer communication. The existence of the ARPANET is having precisely this effect, and as a result the extent of computer-to-computer interactions is certain to grow substantially in the next few years [6].

The ARPANET is one of the most advanced examples of a computer communication network [8], [16], [18]. It consists of a geographically distributed set of different computers, interconnected by a communication system based upon very fast response (interactive) message switching. This network was developed to ultimately allow economic and reliable sharing of specialized computer resources. The ARPANET has demonstrated the feasibility of message-switching technology, illustrated its advantages, and fostered the development of techniques for computer-to-computer communication. It is interesting to note that the ARPANET was originally designed with the notion of computer-to-computer communication in mind. It has subsequently been extended in capability to allow users with terminal equipment but no computer to connect to the net and communicate with computers and other users. In this sense, the ARPANET has taken the opposite approach from every other network designed with user access originally in mind.

For many years, the National Physical Laboratory (NPL) in England has experimented with the use of "single packet" messages for switching in the "local area of a data communication network" [31]. A number of terminal devices were successfully interconnected into a local network at NPL, and recently they have been concerned with extending the local network into a distributed network [30]. A computer-to-computer network is also under development in France to allow data sharing without costly duplication of files and its attendant problems of control, updating, security, etc. Central files, each accessible via a local computer, will be made accessible to other computers and hence to an extended user community. This network is expected to use a message-switching technique similar to that used by the ARPANET in the United States. In addition, networks are under design or development in other countries (e.g., Canada and Japan). Some of the European networks are described in other papers in this issue.

In general, the properties and structure of a computer communications network must reflect the overall requirements for which it was designed. This may consist of high-speed (megabit/second) circuits for rapid computer-to-computer communication, or low-speed (voice and telegraph grade) circuits for terminal access or slow-speed communications; it may be circuit switched or message switched, etc. Whatever its detailed structure, the network contains a communication system (private or common carrier) and a set of computer-system resources and users that interact via the communications system. This system is also called a communication subnet or simply a subnet for short. This organization not only characterizes the organization of geographically distributed networks, but can also serve as a model for the local structure of a single computer complex [3]. Its structure is therefore quite fundamental.

In operating a computer communications network as a "marketplace" for computer-related services, a number of important issues arise [34]. We allude to a few of them here. What criteria are appropriate to determine whether a service may be removed from the system? When and where should additional services be incorporated and what procedures are needed to maintain effective competition? What subnet changes are appropriate for changes in the distribution of resources? The total operational procedure should also include a strategy for utilization of the resources consistent with its intended functions (e.g., load sharing, data sharing, etc.).

An overriding concern of the network design is the overall reliability of the communications and computer resources. For a user to entrust his computing to a network, he must develop confidence in its availability when he needs it. It must be convenient to use and it must provide a believable guarantee to maintain standard and expected grades of service. An investment in time and energy to use a network resource can be negated by the failure to maintain a consistent service offering. Insuring that proper concern exists for the remote user of a computer resource is an important administrative problem that affects almost every phase of computer network development.

IV. Message-Switched Communications

Since the message-switching technology is not as well established as the circuit-switched technology, the fundamentals of its operation are reviewed in this section. Considerable discussion on the nature of these two switching doctrines is taking place. Are they merely different ends of a common spectrum (with a key variable such as packet size), or are they fundamentally different communication techniques? An argument in favor of their similarity is that both types rely on store and forwarding of data, whether a single bit is transiently stored, a byte-sized envelope, or a larger sized packet. The most significant external characteristics that "appear" to distinguish the two systems are that 1) circuit-switching systems are better equipped to maintain a time frame for users that require continuity in transmission, as in speech, while 2) message-switching systems allow speed and code conversion, thus permitting direct connection of and communication between devices of widely varying type. But it is possible to mask even these "seemingly" essential differences by the provision of a small amount of buffering and "byte manipulation" capability at the periphery of either system. It is actually the manner in which internal system resources are managed and utilized that provides a useful measure of comparison between them.

Briefly, in a circuit-switched network, the source and destination are connected by a dedicated communication path that is established at the beginning of the connection and broken at the end. This type of connection was specifically selected for use in switched telephony, where subscribers require a continuity in voice transmission and reception. Since the communication path remains fixed for the duration of a conversation, the output speech signal appears to be a time translate of the input speech signal as far as the ear can tell. In addition, for most voice conversations, the allocated analog voice channel is used in a fashion that seems reasonably efficient to the average user.

To establish a connection, the subscriber provides the local central office with an address which is used in setting up a path. Central office equipment detects off-hook, provides a dial tone, retains dialed digits, generates ringing, busy signals, etc. In the current telephone plant, long-haul circuits are primarily multiplexed analog channels. Routing selection is performed using a set of prespecified paths and usually based on the first few dialed digits. Call setup times generally take between 5 and 25 s, depending upon the number and type of central offices in the link and the amount of traffic. Recent experience has also indicated that reliability and overload problems are becoming increasingly prevalent in certain high-density population areas.

A message-switching system accepts, transmits, and delivers discrete entities called messages. In such a system, no physical path is set up between the source and the destination and no resources (e.g., capacity, buffer storage, etc.) are allocated to its transmission in advance. Rather, the source includes a destination address at the beginning of each message. The message-switching system then uses this address to guide the message through the network to its destination, provides error control, and notifies the sender of its receipt [17], [18].

A simple form of message-switching system employing a single central switching computer is commonly referred to as a "star" configuration and has all its lines connected to the central message switch. For many local applications this configuration can be quite practical. Three of its main disadvantages are 1) the central switch may be an unreliable link which will disrupt all communications if it fails, 2) the total circuit mileage for geographically distributed users to connect to the switch may be substantially larger than necessary, resulting in excessive communications cost, and 3) every circuit failure can result in some loss of user communications.

A distributed message-switching system is one in which many distributed switching computers are employed and the network control is decentralized in such a way that the failure of any switching computer disrupts communications only for its local customer. The distributed system is usually more economic and reliable than a star configuration for handling geographically distributed users.

The components of a message-switching system are dedicated point-to-point communication circuits and switching nodes which interconnect the circuits in such a way that a message arriving on one circuit may be transmitted out another. Communication over a message-switched system occurs via a sequence of transmitted messages, each consisting of its address followed by text. The address is inspected by each node in routing the message to the next node on the way to its destination. In the ARPANET, one or more computers may be directly connected to a node and are known as Host Computers, or Hosts for short. The nodes are called Interface Message Processors or IMP's for short.

A distributed message-switched network, such as the ARPANET, contains no mass storage, and as little buffering in the nodes as necessary to utilize the full capacity of the communication circuits.[1] The network design allows a message to remain in the net only as long as necessary to transport it from source to destination; no long-term storage is provided in the communication system. Messages that cannot be delivered to the destination are simply not accepted into the net and must be retransmitted at a later time. Clearly, one or more Hosts on the net with low cost per bit bulk storage could provide or even be dedicated to providing long-term storage of messages with subsequent automatic retransmission.

The combinatorial aspect of the interconnection of large numbers of computers is an important consideration in network design. Each computer in a message-switching system is connected to the net via a single full-duplex channel to its IMP over which messages are multiplexed. This single connection to the network makes the computing service accessible to all computers and all users on the net. Furthermore, all users and all computers on other digital networks can access this computer by the simple expedient of a single interconnection between nets. Thus not only is complete digital access possible, it is achieved in a strikingly economic way for each installation. This technique solves a massive combinatorial access problem with a single economic stroke.

In Fig. 1, we show the communications portion of the ARPANET as of April 1972 when it consisted of twenty-four nodes and 28 circuits. Since that time it has grown to over thirty nodes. Each node is a possible source and destination of messages. We assume (for the moment) that messages may be of variable length up to a maximum of 1000 bits,[2] and are known as packets while in the network. The path taken by a packet traversing the net from node 1 (UCLA) to node 6 (M.I.T.) is indicated by arrows in the figure. The circles indicate the nodal processors and the lines indicate synchronous point-to-point circuits. The message enters the net at node 1, which examines its address and decides to transmit it out its circuit to node 7 (RAND). Upon receipt, node 7 examines the address and decides to send it to node 21 (TINKER), which in turn sends it to node 5 (BBN) which sends it to node 6 (M.I.T.). Node 6, discovering the message is for itself, replaces the destination address by the source address (which is carried along by the message-switching system) and "delivers" the message. The text of the message thus appears at the destination exactly as it was transmitted and the address portion identifies the sender. After delivery, the sender is notified of its receipt by a small message that goes back across the network.

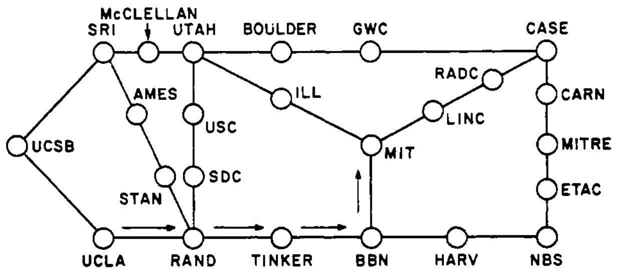

Fig. 1. Routing a message through the ARPANET.

An important part of a computer network design is the specification of the location and capacity of all circuits in the net. Fifty kilobit/second circuits are currently used in the ARPANET to achieve an average delay of 0.2 s or less. Programs have been developed that iteratively analyze various possible network configurations and select reliable high-throughput low-cost designs through the use of circuit-exchange heuristics [20], [22]. Analytical techniques have been developed for estimating the average transit-time delay under assumed traffic loads. These techniques show that the delay remains almost that of an unloaded net until the capacity of one or more "cutsets" begins to saturate [7]. (Network optimization is the topic of a paper by H. Frank and W. Chou in this issue.)

An important design consideration is the method for dynamically selecting routes. (We assume that routes are not allocated in advance.) A central controller could provide the routing information and distribute it to all the processors, or the processors could collaborate in computing the routing information directly. This is but one of many instances of a design choice between distributed and centralized control. In the initial ARPANET design, the route selection is performed independently by each IMP according to a distributed routing algorithm. Routing information is stored in a table and individually maintained by each IMP for rapid look-up. It identifies the output line to select for each destination and is updated according to a rule evaluated periodically (e.g., every half second). It could also be evaluated asynchronously (whenever status changes occur) or a combination of both. In the

[1] The Defense Departments Autodin network, however, employs mass storage in the communication network for deferred retrieval and delivery of messages.

[2] In the ARPANET, messages may actually vary up to 8095 bits in length.

simple algorithm used in Fig. 1, each IMP sends the message on its choice of a path with the fewest intermediate IMP's and, using the update procedure, each IMP adapts its routing to other IMP and circuit failures.

A simple method for implementing this algorithm is for each IMP to keep a table with the count of the number of IMP's on the shortest path to each destination, which it frequently transmits to its immediate neighbors. Each IMP also announces to its neighbors that it is 0 IMP's away from itself. Upon receipt of the routing information from its neighbors, the IMP increments the neighbors' counts and keeps the lowest value for each destination.

Each IMP buffers a packet until receipt is acknowledged by the adjacent IMP. A cyclic checksum, generated in hardware by each IMP, is appended to the transmitted packet for error control. If an error is detected by the hardware at the adjacent IMP, or no buffer space exists, the packet is simply discarded and will shortly be retransmitted by the neighboring IMP when a condition (such as a time-out) occurs and no acknowledgement is received.

The design of an efficient network without mass storage requires that the number of buffers be kept to a minimum, and that they be used so that each IMP is able to use its circuits efficiently and to their maximum capacity. This means that the minimum number of buffers must be at least as large as the number of full-sized packets which must be stored from the time one full-size packet is transmitted until its acknowledgment returns. This number is determined by the circuit propagation delay, the packet size, and the circuit data rates, as well as the total number of circuits. To utilize these buffers efficiently, stored packets must be quickly released upon receipt of their acknowledgement or activated for retransmission, as appropriate, in a timely way.

Each IMP contains only a small amount of buffering for messages in transit and no mass storage, and a flow control strategy is needed to insure that the IMP's do not become "congested" thus preventing useful data from being communicated. This situation is particularly apparent if the network design allows the source or the destination to temporarily stop the transmission or reception of data and then continue without a loss of messages. This is appropriate to time-sharing computers, and is used in the ARPANET, because it allows occasional delays to occur, for example, while a word is stored in memory or a procedure is activated by the processor. In practice, a Host can neither guarantee to accept all messages at their instantaneous arrival rate, guarantee not to crash while receiving heavy traffic, or expect the transmitting Host to buffer messages should he prefer to discard them upon receipt. In particular, flow control is necessary to protect the network against the sudden dispatch of a larger number of messages to a single destination than it is prepared to accept [13].

An often overlooked but important consideration in the network design is whether or not to keep the circuits fully loaded even in the absence of maximum traffic. For instance, should "test messages" be continuously transmitted or only periodically transmitted to check circuits? Under light traffic loads, is it desireable to transmit duplicate packets and accept the first one with a valid checksum, in order to reduce occasional retransmission delays or to improve the response time on a very noisy circuit? For land-based circuits, the extra traffic during otherwise light loads appears to be acceptable and desireable to reduce delays on noisy circuits. The extra processor capacity is ordinarily available for heavy traffic in any event. For multiaccess satellite circuits, however, the extra traffic during light loads may interfere with other processors sharing the same channel.

In an unloaded net, the transit time is determined primarily by the number of IMP's encountered in the routing and the time for the packet to pass from one IMP to the next. This, in turn, is determined directly from the packet length, the circuit data rate, and the speed of light propagation delay. Under increasing traffic loads, the transit time also begins to increase due to occasional delays in the IMP's. However, if the network is designed to begin rejecting the further input of traffic as the capacity limitation of the network is approached, these delays can be kept to a few times that of an unloaded net. Traffic is thus queued outside the net (rather than allowed to enter and be queued inside the net) so the nominal transit time during peak traffic is not very different from that experienced in an unloaded net. In these cases, an attempt must be made to insure that the effective bandwidth is shared "fairly" among all the competing sites.

Network usage generally requires a combination of short transit times for interactive usage and high bandwidth for file transmission. These two objectives may be attained with single-packet messages. To achieve interactive transit times, no setup delay must be incurred. A simple way to achieve this is for the source IMP to retain a copy of each packet which is nominally discarded after the delivery is made, but retransmitted when, for lack of buffer space, the original is discarded at the destination IMP. To achieve high bandwidth, enough messages must be allowed to enter the net between source and destination so as to fill the "pipeline," but this flow must be able to be readily quenched at the source when the buffer space at the destination IMP begins to fill.

The current ARPANET design actually allows variable length messages with a maximum size just over 8000 bits. The message is partitioned by the source IMP into separate 1000-bit packets to speed its transmission through the network. Each packet makes its way to the destination independently where it may conceivably arrive out of order. These packets could be reassembled into the proper order by the destination Host (using sequence numbers), but when the assumption is made that the communications net should preserve sequencing at least at the level of a single process-to-process conversation, the IMP's are obliged to reassemble the packets. The destination Host thus receives the text of each message exactly as it was transmitted in a single-block transfer.

When these larger messages are used and the IMP's undertake the responsibility for reassembly, yet another type of congestion phenomenon called reassembly lockup is introduced [13]. The flow-control mechanism, which is used to prevent the backup of messages in the net, is also powerful enough to prevent the lockup problem. But in its application, it can subject long messages to setup delays and thus delay succeeding short messages from its Host.

If "sufficient" buffer space were available for reassembly at the destination IMP, there would be no *a priori* compelling reason for the subnet to preclude a Host from sending full 8000-bit messages (or even somewhat larger ones). However, the presence of 8000-bit messages may noticeably delay shorter messages from other Hosts 1) while it is being delivered to the destination, and 2) by tying up 8 buffers rather than 1 during reassembly. This provides one valid reason to restrict the Hosts to single-packet messages, if these delays become significant. However, as we indicate in Section V, there may be other factors which favor retention of the larger size.

If there is a fundamental distinction between circuit switching and message switching, it is undoubtedly in the way internal resources are managed. For example, circuit switching requires that network bandwidth as well as local control equipment and storage be allocated to a given transmission circuit in advance, whereas a message-switching system does not. Secondly, the presence of circuit-switched routes means that user messages are identified by their circuit and no user control signalling need accompany the transfer of information. In message switching, however, all record of activity (except accounting) associated with a message is contained in the message, which vanishes when the message leaves the system. This signaling information, in the form of an address, must accompany each message and the message must be examined and processed at each stage of the transmission process.

Two practical consequences of the difference are that the circuit-switched system usually requires a nontrivial setup time to allocate resources. Message-switched systems can avoid setup delays, but may introduce occasional variations in transit time. These delays can generally be maintained to within a few times the delay of an unloaded net, but wider variations may result from queueing delays outside the net, particularly under heavy traffic load. Under similar conditions, though, a circuit-switched user might fail to obtain a circuit and would incur this probabilistic situation on subsequent tries. Any allocated but idle channels are simply unavailable at this time to handle these overload conditions.

V. Network Use of Individual Computer Systems

The term network has been used and misused in a variety of ways. Some people have referred to the use of dial-up facilities to access a single computer as a network. Others have referred to any interaction between computers and a communication system as a network capability. Several distributed networks were developed to allow simple forms of communication between identical machines using standard dial-up or leased voice circuits, thus providing a convenient way to transfer jobs and files and to maintain and update the systems in the net. This latter application exemplifies a true networking activity, even though it only concentrates on selected aspects of computer resource sharing [19], [21].

In general, only a subset of the network sites possess computing power, and certain of them will offer regular service to users via the network. Other sites may choose to offer service only on a limited basis, or to cooperatively interact via the net with selected co-workers, but not offer general service. This latter situation is more likely to occur for many specialized research facilities. In addition, large private computer centers as well as commercial firms may welcome the opportunity to connect their systems, since it offers a large potential market for usage of unused capacity.

Host service on the network ought to be as reliable as the communications, although this objective is often difficult to achieve. For example, in the ARPANET, total uptime of the IMP at any site is currently on the order of 98 or 99 percent, while Host availability is generally no higher than 90 percent. It is certainly possible to improve on this score; some commercial firms claim to provide over 99-percent availability of service, and certain private and government systems must obviously be operated with near-perfect reliability. The airline systems and the computers in the space program provide two key examples.

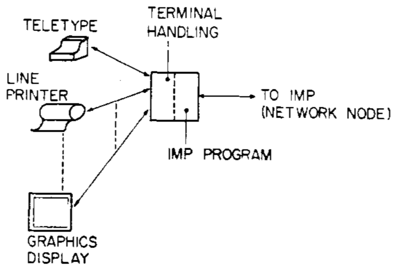

Fig. 2. Terminal IMP.

A. Network Access

In a network that supports computer-to-computer communication, user groups with a local computer can access another computer in the net by first logging into their home computer and then into the other computer, using the home computer in a transparent mode as a switch. However, this is an expensive way to access another computer since it incurs charges in two computers and ties up jobs in both. Furthermore, since Hosts may be unreliable, the connection is more vulnerable than a direct connection into the other machine.

Sites with expensive computer installations might deem it economic to depart from their machine if "equivalent" service can be economically obtained via the net. In general, such a site requires the ability to service local users with a full complement of terminals and peripheral devices, such as Teletypes, graphics displays, line printers, magnetic tapes, and possibly other mass storage. In addition, many sites with no computer will derive maximum benefit in connecting to a net if the full range of peripherals can be provided locally. Other users, however, may be satisfied with a simpler approach that provides terminal access to remote computers but does not handle peripheral devices other than line printers.

For this latter class of users, an addition may be made to the IMP which allows a set of terminals to be directly connected. This addition consists of multiplexing equipment that collects characters from the terminals and packages them in the form suitable for delivery to the IMP. Likewise, it receives messages from the IMP and sorts the characters out to the various terminals. This addition requires hardware and software designed to make the set of terminals interface to the network as a "mini-Host," and this IMP is referred to as a Terminal IMP. (See Fig. 2.)

A more elaborate approach is appropriate for user sites that wish to support many different types of peripheral equipment. Since their characteristics and operation can vary widely, device-dependent programming is mandatory, and substantial buffering may be required for the higher speed devices. Furthermore, sites with mass storage will generally wish it to be accessible from other computers in the net, which generally requires the implementation of a full set of standard and specialized network protocols. These considerations make it appropriate to provide a separate processor devoted to the handling of peripherals.

This latter approach is particularly desirable for a site which is both a user site and a server site. The architecture of such a site should be organized so that if the serving site central processing unit (CPU) is down, local users can access other network computers and local mass storage can be accessed by them over the net. Similarly, if the local storage should fail, others across the net can temporarily replace them. If the net fails, local users can still obtain full access to the local system. Only if both the local CPU and the network fail will the users be unable to obtain computation.

Modularity and logical reconfiguration are conveniently achievable in this way. Substantial progress in the design of modular communication-oriented architectures can be expected from innovative usage of interconnection ideas [3]–[5].

B. User Requirements

Let us now turn our attention to the use of these facilities by the user. We note three potential locations where user computation can be performed—in the terminal itself, in the peripheral processor (or Terminal IMP), or in the Host computer(s). Although the bulk of the "pure computation" will undoubtedly take place in the Host computers, some aspects of the processing must, in general, be distributed. For example, local echoing is required to obviate the otherwise noticeable effects of speed of light propagation delay, as on satellite links. This raises the important question of location of functions in a distributed network. In other words, what intelligence is needed to allow distributed system usage and where should it be placed?

Let us concentrate on the echoing problem for the moment. As a general rule, a remote user should see the same output and otherwise obtain service from a remote Host as if he were a local user. To achieve this objective, the user's local system (programmable terminal, "mini-Host," etc.) must have considerable information available to it about each subsystem in use at the remote Host. For example, a simple local echo/remote echo strategy is generally insufficient to handle echoing for users on half-duplex terminals, or users on full-duplex terminals that prefer to type ahead. A remote user editing the character string ABCDEFG can delete the last three characters by typing successive delete characters (echoed as\) and he sees the output ABCDEFG\G\F\E. Using the system from a remote site with local echoing and typeahead he would see ABCDEFG\\\GFE.

More striking, perhaps, is the remote use of a debugging program DDT. To examine successive registers 120–123, a local user would first type 120/ to print the contents of the first register and then strike successive line feed characters to examine the successive registers. The system would respond with (say the contents are all zeroes):

```
120/ 0
121/ 0
122/ 0
123/ 0
```

and leave the cursor following the last 0. A remote user with typeahead and local echo would see one of several possible responses (depending on the remote systems response to a received line feed). Assuming the remote system echoes only the formatted data and the local system echoes linefeed as linefeed (no carriage return), the output would look as follows:

```
120/



                         0121/ 0122/ 0123/ 0.
```

In general, the local echoing system should have full knowledge of the time-varying syntactical operation of the subsystem in use. This requires feedback of information about subsystem break and separator characters, control signalling, special conventions, etc.

Each terminal has characteristics peculiar to it and a convention is required for a computer system to initially recognize a terminal. Although the remote computer could then convert to the characteristics of the terminal, it is far more manageable if each terminal could appear to the network as a standard terminal employing an agreed upon set of characters and signalling conventions. One such standard (developed for the ARPANET) is 7-bit United States of America Standard Code for Information Interchange (USASCII) with the eighth bit set to 0. In that scheme, the other 128 possible characters are reserved for special control characters. In addition, most terminals need attention to details, such as carriage return, keyboard locking and unlocking, interrupt signalling, and other peripherals indicate out of paper, buffer full, and may require complete two-way channel control, etc.

Local computation is therefore needed for the user's terminal to interact properly with other remote systems and their subsystems. At a minimum, his local computation must allow the user to 1) identify his terminal to the network, 2) select a destination Host, 3) select a transmission mode, 4) perform echoing and code conversion, and 5) allow the remote Host to be interrupted.

It seems probable that, in the long run, many terminals will contain mini-processors and thus user programming in a separate "mini-Host" will be unnecessary. (This is amplified in the paper on terminals by L. C. Hobbs in this issue.) However, until this possibility is a widespread reality rather than an expectation, users may be hindered if they are unable to provide local user code in one place or the other.

C. Message Processing

Before considering various examples of usage of a computer network, let us briefly indicate how messages are processed within the Host computers [15], [32]. Messages travel through several layers of protocol in the Host system. The first layer of protocol handles the IMP, activating I/O buffers, fielding control messages, etc. The second layer interacts with the local processes and remote Hosts monitor, allocating buffer storage, providing process identification, formatting control information, etc. Subsequent layers correspond to specific user-oriented functions, such as the standard network terminal, file transmission, etc.

The ARPANET Host protocol utilizes the notion of connections over which messages are transmitted. A connection must first be established before communication over it may occur. The Hosts at either end of the connection must keep full information about the use of the connection (which is obtained during its establishment) to handle flow control. This strategy appears to be close in spirit to telephone circuit switching.

A few limitations to this strategy are apparent [14]. An important concern is that it requires each Host to maintain resources in the form of connection tables that can become filled, thus preventing any further communication with that Host. In particular, a single process can attempt to establish its maximum limit of connections, although it cannot, in general, make full use of them at one time. Entries in this connection table are permanently allocated, and thus only a fixed number of connections can be established at any time.

A second limitation with the use of a connection table is that it can be vulnerable to error conditions and Host status, particularly since both Hosts must generally agree on its contents for flow control. Finally, the strategy requires the connection information to be used for termination, which means that information which otherwise would be nominally discarded by Hosts with limited space, must be retained merely to close the connection. These limitations, as well as others, may be obviated with a message-switched Host protocol [14].

The desired size of Host messages may have an important impact on the operating system as well as on the communication system. The original ARPANET design specification allowed individual messages to be as large as 8192 bits, a design choice based largely on intuition.

As the design specification originally stated [29]:

> . . . a packet is defined as the inter-IMP unit and Message as the inter-Host unit. A packet will not exceed 1024 bits in length. The IMPs must break all longer messages into multiple packets. Messages will be limited to 8192 bits so as not to require excessive buffer space.

Undoubtedly, this latter reference is interpretted as referring to buffer space in the IMP's, but it could equally well apply to buffer space in the Hosts. In particular, the argument defends why the size is not larger, but does not entertain the possibility that it ought to be kept smaller for any specific reason.

If there is a convenient maximum Host message size, it is probably a maximum-sized page, which corresponds to 1 K of 36-bit (or possibly 48-bit) words. However, transmission of such large messages (say 50 000 bits) to the IMP and from the IMP to the Host produces excessive delays for short messages queued up behind them, and provides a prime reason for Hosts to prefer that these long messages be subdivided into shorter messages. Since no experience with network software was available during the initial design, it was intuitively concluded that a shorter 8192-bit message was short enough. Interestingly, we note that two hardware paths between each Host and its IMP, one for short messages and one for long messages, could remedy this problem at some extra cost in hardware and buffer storage.

Since efficient transmission is possible with 1024-bit packets, it appears in retrospect that the selection of the larger message size may be unnecessary. The Host overhead in network communication increases with the number of messages, so there is some incentive for making all Host messages sufficiently large that a typical short transaction can occur in a single message. No evidence yet obtained by us indicates that 1024-bit Host messages would impose a limitation that is significant, but an increased demand for page transfers or the presence of higher bandwidth circuits could tip the balance more strongly in favor of a larger size. The jury seems to be still out on this issue.

VI. Applications of Multicomputer Interaction

Network utilization involving the combined use of two or more computers in a productive way began during the initial experimentation with the ARPANET. It has provided experience in the development of techniques for performing distributed computation and allowed some simple application areas to be identified. Some applications involving multiple computers have been discussed for many years, partly as a result of their inherent interest and ease of conceptualization. One important example is the access to specialized data bases that are only available from a remote source. Several information banks have already been developed or are under development, and their expected usage is being projected upward. Another example is in the use of "future" computer communication networks for handling the distribution and delivery of mail and other transient information. However, these applications are only beginning to develop in any significant way. Much effort has already been devoted to the study of topics such as concurrency and parallel processing, which may result in faster program execution and otherwise make efficient usage of available resources. We expect that computer networking will enhance these efforts. For other applications, the sensation of dealing with one system rather than two (or more) is overwhelmingly evident to the user and this pleasant feeling often generalizes to other multicomputer interactions as well.

Three areas in which applications have already occurred are briefly identified below.

A. File Transfer

The first application for combining two computer systems in the ARPANET in a nontrivial way involved the use of an XDS-940 computer at Stanford Research Institute (SRI) and a PDP-10 computer at Utah. SRI, anticipating the delivery of a PDP-10, began to use the Utah machine in the development of PDP-10 software.

At first a higher level language was developed. Source code was generated on the 940, converted to object code, and executed on the 10. Patches were made on the Utah machine during debugging and, periodically, an updated source and binary version would be generated at SRI and sent over the net. Subsequently, other higher level languages were similarly developed.

A simple protocol to handle file transfers was developed for the TENEX operating system [10] and has proven useful for transferring new subsystems and system revisions between TENEX sites. In addition, it has been a useful initial step to allow cooperating processes in two TENEX systems to share a single file.

In this protocol, the network appears as a device to which a file may be output or from which a file may be input. The two ends of the transfer must coordinate by having one end execute the input and the other end execute the output. This simple file transfer protocol requires the intervention of the user to log into both ends, assign a file name for the destination, invoke the proper format, etc. Other experience in the transfer of files has been recorded by the University of California at Santa Barbara, as well as by IBM, by Control Data and others [19], [21].

B. Remote Job Service

A simple example of a computer-to-computer interaction is provided by users who write, debug, edit, and store programs on an interactive time-sharing facility and run them on a separate batch processing system. While time sharing has created an interactive environment for programming and the development of programming techniques, batch processing systems, and small dedicated computers have maintained a predominance for performing extensive computations. The availability of both kinds of service in a computer-communications network provides a single user with convenient access to the best features of both. (See Fig. 3.)

While a user can become accustomed to using both services independently, he need not be required to physically collect

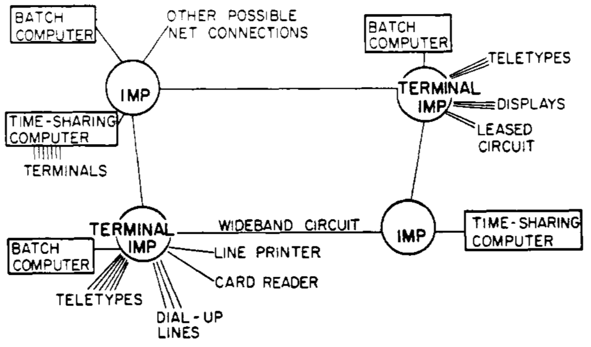

Fig. 3. A simple IMP network.

time-sharing output on tape or punched cards for submission to the batch system. The most convenient user option is for the interactive machine to submit his job to the batch processing machine under user initiated control. He can then specify the location for output to be stored or printed, revise the program in the time-sharing system, and resubmit it under fully interactive procedures from a single location, with no need to keep physical copies of files, program, etc.

For over a year the RAND Corporation had been using the ARPANET for remote job service from an IBM 360/65 at RAND to a 360/91 at UCLA [26], [27]. This facility was only accessible to internal RAND users until recently when it was replaced by a PDP-10, which allowed network users to create and submit jobs for remote service elsewhere.

Both the program and its relevant files must be transferred to the 360/91 before a job can be run. They are typically shipped together as successive "card images." The remote job service program will allow the users to start or stop the execution of his program, cause the system output to be stored on a designated file, or be output on a device such as a local printer. The user is also provided with options to check the status of the execution, receive confirmation and error messages that indicate its progress, and allow certain actions to be taken.

These facilities are used by RAND researchers in the generation and processing of simulated weather data. Weather modeling programs can be activated from a remote site, output from these programs can be temporarily stored, or shipped to a remote site for preparation and display. This separation of the computation into components is particularly appropriate when one part may be devoted almost exclusively to extensive numerical or symbolic computation and another part to user-related manipulation or preparation of the output data.

A "complex" weather simulation program requires many hours of computation on the 360/91, and thus is not well suited to rapid on-line activities such as the updating of a display. Rather, precomputed weather data (from the models) are retrieved from 360/91 disk packs (with operator assistance) and used by the PDP-10 for further processing and display. The availability of a high-speed parallel processing system such as the Illiac IV [33] may eventually allow real-time weather experimentation without operator intervention.

C. *Multiprocess Operation in Many Machines*

The combined use of two or more computers allows additional processing capability over the use of a single system. One such example is provided by the McRoss system [12] that coordinates the operation of two or more cooperating air-traffic-control simulation programs running in one or more TENEX systems. Each simulation program, called Route Oriented Simulation System (ROSS) [23], models the airspace of one air-traffic-control center in detail. To simulate the airspace of a Boston to New York flight, four simulation programs would be activated; one for the Boston terminal area, one for the Boston enroute area, one for the New York enroute area, and one for the New York terminal area. The four ROSS programs may be run simultaneously in as many as four TENEX systems in the ARPANET.

When a single machine is used to house all the components of a programming system, it has the disadvantage that computing will stop if that machine crashes. When one piece of a multicomputer programming system becomes unavailable, the other parts can learn to adjust to the change in configuration. A desirable objective is to provide enough backup information to enable the multicomputer programming system to be restarted in the event of a single Host failure and to proceed from a recent point in simulated time as if nothing had happened.

Other applications involving multiple computers are certain to arise for which simple examples are more difficult to construct. For example, as special areas of expertise develop, it is natural to expect that individual efforts by specialists also trained in the use of computers will produce new and useful resources on different machines. These resources may represent state of the art or proprietary developments that cannot be conveniently transferred to other machines, and must therefore be used at the site of their creation or where they currently exist. An important application for distributed computation is thus likely to involve the coordination of separate research projects into combined efforts that utilize these specialized or proprietary and hence nontransferable resources.

A second major application of distributed computation is likely to be the facilitation of interactive cooperation between people at different locations. Interactive cooperation may be regarded as an extension of normal voice communication to include the ability for several persons at different locations to "simultaneously" observe, communicate about, and manipulate both common data structures and programs. Since the people are assumed not to be colocated, the programs which support the interactive cooperation (such as display protocol routines) must also be distributed.

A third major application for distributed computation is in providing for conveniently feasible demonstrations of prototype systems to be performed from different locations. This technique can allow new capabilities to be readily conveyed without the inconvenience of moving the observer to a home site for the demonstration.

VII. A Distributed Operating System

A network in which basic differences exist between the computers at each installation is said to be inhomogeneous. It is possible to develop a standard network protocol for an inhomogeneous system that allows usage of various pieces of the system to be coordinated in a uniform manner. However, this task is one of substantial complexity that will probably require changes in system architecture and program design techniques before it can be fully realized. Even if it were a straightforward matter, it would not be generally useful to transfer portions of any one system to another, and standard operations that involve systems at a remote site must typically be performed at that remote site.

A collection of similar operating systems may also be organized into a virtual subnetwork of homogeneous computers that interact with each other in a uniform way. These systems are more easily organized into a single distributed operating system with common file systems, address space, naming conventions, etc. In general, every type of interaction between two systems in a homogeneous network must be evaluated to determine what is to be transmitted and what is to be remotely evaluated. No single answer will suffice for all applications. As we noted above, it is not generally possible to apply both alternatives in an inhomogeneous network. We consider some of the properties a system like this ought to possess.

The user accesses the distributed homogeneous network by logging into a distributed system rather than into a specific computer in the net. An appropriate machine is selected for him and he logs in with the standard log-in sequence for his home computer, including password, account number, and other information as required. Upon completion of the log-in sequence, the computer initiates a brief exchange with the users home computer to notify it of the impending job which it then proceeds to service. The home computer may then request that the job be transferred, alter credit or accounting information, or merely note the event.

Under conventional design constraints, the combined operation of several Host computers will require a separate job to be established in each machine. In a distributed system, though, it is important to allow access to each system without the user logging into each system individually. Furthermore, it is also desireable to permit certain transient activities to occur which do not tie up valuable resources or otherwise interfere with users on the system. The system merely performs the transient activity and logs the transaction into a suitable file for accounting purposes. Once logged into one of the Hosts, the user is able to access and utilize any programs, files, and most other facilities on other computers in the system as if they were all on one virtual machine.

The availability of many resources in this system makes it possible to achieve reliable operation when one or more resources are disabled. The user can be affected by failure in several ways. For example, his program or a piece of it may be aborted by machine failure or he may lose part or all of his files. The user may also find the local file storage to be unusable while running his job. If local storage is not initially available, he can specifically designate another system to store his files. Alternatively, he can allow the local system to store files in other Hosts and expect them to be returned without his knowing the identity of the temporary storage location. Obviously, a small amount of local storage is needed for this application. The distributed system thus not only makes resources more widely available; it can use them to provide increased reliability to a user.

A system designed to operate stand-alone may not be as efficient in serving its network users as in serving its local users. Certain performance improvements are obtainable by streamlining critical portions of the system code, attention to organizational details and to carefully engineered improvements to scheduling, the file system, etc. However, a major improvement in speed and efficiency may require structural overhaul of the system organization to allow for efficient process-to-process communication at high bandwidths and for efficient overall utilization of resources. In particular, the portion of the system devoted to protocol and message handling (byte manipulation) can consume a considerable amount of CPU time at high bandwidth. In a time-sharing system, particularly, these functions, which appear to be communication functions, can be usefully separated from the "computation" functions. This decomposition also allows more human engineered network interfacing, since a local CPU failure can usually be reported by the communications portion of the system if it remains up. The delegation of all protocol functions to a separate processor that can directly deposit into and retrieve from process buffers makes it possible for the operating system to communicate over the network at speeds at least an order of magnitude faster than before. This increase in capability is achieved by performing the protocol operations on the fly in a separate processor and by avoiding the unnecessary overhead in moving real-time data around in memory.

Techniques for computer-to-computer communication are still in their infancy and a great deal of exploration and experimentation is occurring in this area. How should programs be written to run in a network environment and what debugging and control techniques are suitable for distributed computation? What operating system architectures are appropriate to computer communication? The efficient utilization of a distributed operating system involves the sensible decomposition of a task into components. This requires timely access to status information and the ability to use this information wisely in the allocation of tasks to resources and in their scheduling. Just as the management of communication resources was central to the operation of a communication subnet, so will the management of computer resources be to the overall utilization of a computer network.

VIII. Conclusions

A principal motive underlying computer network development is to provide a convenient and economic method for a wide variety of resources to be made available and to be shared. Such a network provides more than an increased collection of hardware and software resources; it affords the capability for computers as well as individuals to interact in the exchange and processing of information.

It is not usually the case that a program written for one computer can be shipped to another computer and run there to completion, correctly. It may be possible in a number of cases where the machines are nominally identical, but it is usually the case that a program must be run on the machine for which it was written. It is thus desireable to strive for compatibility between at least a subset of the system resources, including the use of machine-independent higher level languages, the use of network wide standard protocols, or the use of nominally identical systems.

The development of communication subnets has been strongly influenced by the regulatory climate and the need for reliable and economic ways to achieve both remote terminal access and high bandwidth switched computer-to-computer communication. Message switching has emerged as a strong contender for computer-to-computer communications. It has been demonstrated to provide a highly reliable, error-free method of achieving interactive switched communications. Although its technical feasibility has been firmly established, its practical utility is under evaluation, and under close scrutiny it may prove to be a viable economic alternative to conventional circuit switching.

It is important that a communication system not preclude the possibility that separate or private data networks may be accessed through it in a standard and convenient way. A digital message-switched network has this property, while an analog frequency-based system may not. Incompatible data networks are clearly undesirable if all resources are to be mutually accessible. If separate data networks are jointly planned before development, at least at the interconnection level, they may be connected at a later date and viewed together as a single network that evolved by way of separate networks.

The great diversity of resources in a computer network may initially hinder its growth. Users must familiarize themselves with many different systems often without the aid of substantive interaction with systems personnel or clear and complete documentation. But the potential benefits of computer networks are sufficiently great that, over time, this obstacle will surely be surmounted, and in the process may lead to superior standards for system operation and documentation.

Computer networks provide a unique mechanism for increased participation between individuals. Participation in research and development using the distributed resources of a computer network can lead to the close cooperation between individuals who might otherwise have little incentive to work together. This interaction can further cross-fertilize the network community and encourage even higher levels of achievement through technical cooperation.

References

[1] "Pinched budgets promote growth of university computer networks," *Commun. Ass. Comput. Mach.*, vol. 15, no. 3, pp. 206–207.

[2] P. M. Walker and S. L. Mathison, "Regulatory policy and future data transmission services," *Computer Communication Networks*. Abramson and Kuo Eds. Englewood Cliffs, N. J.: Prentice-Hall, 1973, ch. 9.

[3] D. Farber and K. Larson, "The architecture of a distributed computer system—An informal description," Dept. Informat. Comput. Sci., Univ. of California, Irvine, Tech. Rep. 11. 1970.

[4] H. Baskin, B. Borgerson, and R. Roberts, "PRIME—An architecture for terminal oriented systems," presented at the AFIPS Spring Joint Comput. Conf., pp. 431–437, 1972.

[5] R. Davis, S. Zucker, and C. Campbell, "A building block approach to multiprocessing," presented at the AFIPS Spring Joint Comput. Conf., pp. 685–703, 1972.

[6] S. Crocker *et al.*, "Function oriented protocols for the ARPA network," presented at the AFIPS, Spring Joint Comput. Conf. pp. 271–279, 1972.

[7] H. Frank, R. E. Kahn, and L. Kleinrock, "Computer communication network design—experience with theory and practice," presented at the AFIPS Spring Joint Comput. Conf., pp. 255–270, 1972.

[8] L. G. Roberts and B. Wessler, "The ARPA network," in *Computer Communication Networks*, Abramson and Kuo,. Eds. Englewood Cliffs, N. J.: Prentice-Hall, 1973.

[9] M. Beere and N. Sullivan, "Tymnet—A serendipitous evolution," presented at the 2nd Ass. Comput. Mach. IEEE Conf. Problems in the optimization of Data Communication Systems, Palo Alto, Calif., 1971.

[10] D. G. Bobrow, J. Burchfiel, D. Murphy, and R. Tomlinson, "TENEX—A paged time-sharing system for the PDP-10," *Commun. Ass. Comput. Mach.*, vol. 15, no. 3, pp. 135–143, Mar. 1972.

[11] F. J. Corbato *et al.*, "An introduction and overview of the multics system," presented at the AFIPS Fall Joint Comput. Conf., pp. 185–196, 1965.

[12] R. Thomas and D. Henderson, "McRoss—A multi-computer programming system," presented at the AFIPS Spring Joint Comput. Conf., pp. 281–293, May 1972.

[13] R. E. Kahn, and W. R. Crowther, "Flow control in a resource-sharing computer network," *IEEE Trans. Commun. Technol.* (*Special Issue on Computer Communications*, pt. II), vol. COM-20, pp. 539–546, June 1972.

[14] D. C. Walden, "A system for interprocess communication in a resource sharing computer network," *Commun. Ass. Comput. Mach.*, vol. 15, no. 4, pp. 221–230, Apr. 1972.

[15] *The Host/Host Protocol for the ARPA Network*, Network Informat. Center, Stanford Research Institute, Menlo Park, Calif., Document No. 7147.

[16] L. G. Roberts and B. Wessler, "Computer network development to achieve resource sharing," presented at the AFIPS Spring Joint Comput. Conf., pp. 543–549, 1970.

[17] P. Baran, S. Boehm, and P. Smith, "On distributed communications," RAND Corporation, Santa Monica, Calit., Series of RAND Reps., 1964.

[18] F. Heart, R. E. Kahn, S. Ornstein, W. Crowther, and D. Waldenk. "The interface message processor for the ARPA computer network," presented at the AFIPS Spring Joint Comput. Conf., pp. 551–567, 1970.

[19] A. H. Weis, "Distributed network activity at IBM," IBM Res. Rep. RC3392, June 1971.

[20] H. Frank, I. Frisch, and W. Chou, "Topological considerations in the design of the ARPA computer network," presented at the AFIPS Spring Joint Comput. Conf., pp. 581–587, 1970.

[21] W. J. Luther, "Conceptual bases of CYBERNET," Courant Symp. 3, 1970, in *Computer Networks*. Englewood Cliffs N. J.: Prentice Hall, pp. 111–116.

[22] R. Van Slyke and H. Frank, "Reliability of computer communication networks," in *Proc. 5th Conf. Applications of Simulation*, New York, Dec. 1971.

[23] W. Sutherland, T. Myer, D. Henderson, and E. Thomas, "Ross, A route oriented simulation system," in *Proc. 5th Conf. Applications of Simulation*, New York, Dec. 1971.

[24] T. Marill and L. G. Roberts, "Toward a cooperative network of time-shared computers," presented at the Fall Joint Computer Conf., 1966.

[25] Project MAC Conf. on Concurrent Systems and Parallel Computation, Woods Hole, Mass., June 1970.

[26] E. Harslem. RAND Corporation, Unpublished Memo, 1972.

[27] R. Braden, Univ. of California at Los Angeles, Unpublished Memo, 1972.

[28] C. R. Fisher, "Introduction to the DATRAN switched digital network," in *Proc. Int. Conf. Communication*, pp, 23-1–23-3, June 1971.

[29] Request for Quotation (Interface Message Processors for the ARPA Computer Network), Defense Supply Service, Washington, D. C., July 1968.

[30] R. A. Scantlebury and P. T. Wilkinson, "The design of a switching system to allow remote access to computer services by other computers," presented at the 2nd Ass. Comput. Mach. IEEE Symp. Problems in the Optimization of Data Communication Systems, Palo Alto, Calif., Oct. 1971.

[31] D. W. Davies, K. A. Bartlett, R. A. Scantlebury, and P. T. Wilkinson, "A digital communication network for computers giving rapid respcnse at remote terminals," presented at the Ass. Comput. Mach. Symp. Operating System Principles, Gatlinburg, 1967.

[32] "Specifications for the interconnection of a Host and an IMP," Bolt Beranek and Newman Inc., Cambridge, Mass. 02138, Rep. 1822, Oct. 1971.

[33] W. J. Bouknight *et al.*, "The Illiac IV system," *Proc. IEEE*, vol. 60, pp. 369–388, Apr. 1972.

[34] *Datamation*, pp. 36–46, Apr. 1972.