The Terminal IMP for the ARPA computer network*

by S. M. ORNSTEIN, F. E. HEART, W. R. CROWTHER, H. K. RISING, S. B. RUSSELL and A. MICHEL

Bolt Beranek and Newman Inc.
Cambridge, Massachusetts

INTRODUCTION

A little over three years ago the Advanced Research Projects Agency of the Department of Defense (ARPA) began implementation of an entirely new venture in computer communications: a network that would allow for the interconnection, via common-carrier circuits, of dissimilar computers at widely separated, ARPA-sponsored research centers. This network, which has come to be known as the ARPA Network, presently includes approximately 20 nodes and is steadily growing. Major goals of the network are (1) to permit resource sharing, whereby persons and programs at one research center may access data and interactively use programs that exist and run in other computers of the network, (2) to develop highly reliable and economic digital communications, and (3) to permit broad access to unique and powerful facilities which may be economically feasible only when widely shared.

The ARPA Network is a new kind of digital communication system employing wideband leased lines and message switching, wherein a path is not established in advance and instead each message carries an address. Messages normally traverse several nodes in going from source to destination, and the network is a store-and-forward system wherein, at each node, a copy of the message is stored until it is safely received at the following node. At each node a small processor (an *Interface Message Processor*, or *IMP*) acts as a nodal switching unit and also interconnects the research computer centers, or *Hosts*, with the high bandwidth leased lines.

A set of papers presented at the 1970 SJCC[1–5] described early work on the ARPA Network in some detail, and acquaintance with this background material (especially Reference 2) is important in understanding the current work. The present paper first discusses major developments that have taken place in the network over the last two years. We then describe the *Terminal IMP*, or *TIP*, a development which permits direct terminal access to the network. Finally we mention some general issues and discuss plans for the next stages in development of the network.

THE DEVELOPING NETWORK

The initial installation of the ARPA Network, in 1969, consisted of four nodes in the western part of the United States. A geographic map of the present ARPA Network is shown in Figure 1. Clearly, the most obvious development has been a substantial *growth*, which has transformed the initial limited experiment into a national assemblage of computer resources and user communities. The network has engendered considerable enthusiasm on the part of the participants, and it is increasingly apparent that the network represents a major new direction in both computer and communications technology.

Figure 2 is a logical map of the network, where the Host computer facilities are shown in ovals, all circuits are 50 kilobits, and dotted circuits/nodes represent planned installations. On this figure certain nodes are listed as a "316 IMP"; this machine is logically nearly identical to the original IMP, but can handle approximately two-thirds of the communication traffic bandwidth at a cost savings of approximately one-half. The original IMP includes a Honeywell 516 computer, and more recently Honeywell began to market the 316 computer as a cheaper, downward-compatible machine. As the network has grown, sites were identified which did not require the full bandwidth of the original IMP, and a decision was made to provide an IMP version built around the 316 computer. Also shown in Figure 2 are certain nodes listed as "TIP"; this new machine

* This work was sponsored by the Advanced Research Projects Agency under Contract No. DAHC15-69-C-0179.

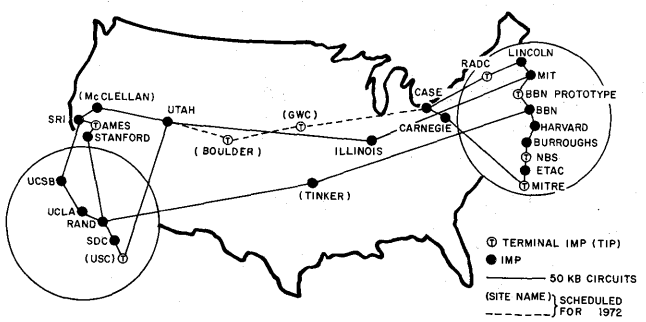

Figure 1—ARPA Network, geographic map, December 1971

is discussed in detail later in this paper. Site abbreviations shown on Figures 1 and 2 are explained in Table I.

As the network has grown, a great deal of work has been concentrated on the development of Host-to-Host protocol procedures. In order for programs within one Host computer system to communicate with programs in other Hosts, agreed-upon procedures and formats must be established throughout the network. This problem has, as predicted, turned out to be a difficult one. Nonetheless protocol procedures have evolved and are being accepted and implemented throughout the net. At the present writing, many of the Hosts have working "network control programs" which implement this protocol. Protocol development is more fully reported in a companion paper,[6] but we wish to make a general observation on this subject: the growth of the network has dynamically catalyzed an area of computer science which has to do with the quite general problem of *how programs should intercommunicate*, whether in a single computer or between computers. Thus the evolution of the Host-to-Host protocol represents a side benefit of the network that reaches well beyond its utility to the network alone.

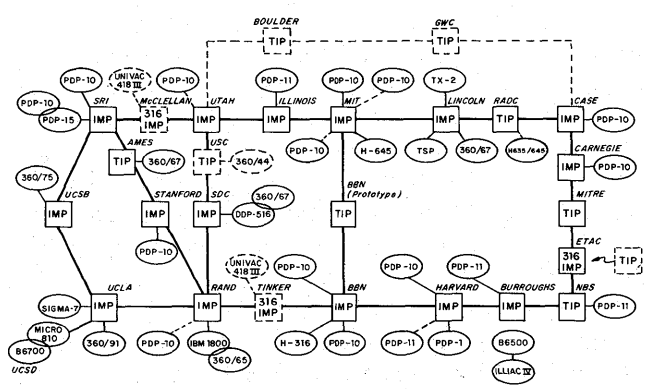

Figure 2—ARPA Network, logical map, December 1971

Since both hardware and software network connections must be implemented by each Host, it is important that the external characteristics of the IMP be relatively stable. This stability has been carefully maintained, while at the same time internal operation of the IMP program has undergone extensive revision and improvement. For example, trouble reporting, statistics gathering, and test procedures have been substantially improved. In addition to improvements that have already been incorporated into the program, there have also been extensive studies of performance and message flow control.[7] These studies have pointed up areas of vulnerability to perverse

TABLE I—Site Abbreviations

NCAR	National Center for Atmospheric Research
GWC	Global Weather Central
SRI	Stanford Research Institute
MC CLELLAN	McClellan Air Force Base
UTAH	University of Utah
ILLINOIS	University of Illinois
MIT	Massachusetts Institute of Technology
LINCOLN	M.I.T. Lincoln Laboratory
RADC	Rome Air Development Center
CASE	Case Western Reserve
AMES	N.A.S.A. Ames Research Center
USC	University of Southern California
UCSB	University of California at Santa Barbara
STANFORD	Stanford University
SDC	Systems Development Corporation
BBN	Bolt Beranek and Newman
CARNEGIE	Carnegie University
MITRE	MITRE Corporation
ETAC	Environmental Technical Applications Center
UCLA	University of California at Los Angeles
RAND	Rand Corporation
TINKER	Tinker Air Force Base
HARVARD	Harvard University
BURROUGHS	Burroughs Corporation
NBS	National Bureau of Standards

heavy traffic patterns and have suggested still other possible improvements in the routing and flow control mechanisms. Potential changes are presently being being studied in some detail and will be incorporated into one or more forthcoming major revisions of the program. They will hopefully anticipate problems which might be expected to arise as traffic flow in the network becomes heavier.

Somewhat belatedly in the network design, the need to connect a single IMP to several Hosts was recognized. This required multiple Host interfaces at the IMP as well as more complex IMP software. Further, the various Host computers at a site are often physically

distant from one another, thus requiring an increase in the maximum allowable physical separation between a Host and its IMP. To connect to an arbitrarily remote Host would have meant a communications interface capable of attachment to common-carrier circuits via modems. It would furthermore have required cooperative error control from the Host end. At the time, we chose not to modify the logical way in which the IMP dealt with the Host and instead we provided more sophisticated line drivers which would handle distances of up to two thousand feet. Several such "Distant Host" installations are now working in the network. Unfortunately, as the network has grown, new sites have appeared where still greater Host/IMP distances are involved. The present scheme does not include error control, and use of this scheme over greater distances is not appropriate. At the present time we are therefore considering how best to arrange IMP/Host connections over large distances and additional options will be required in this domain.

Another facility which has been tested is the ability of the IMPs to communicate over 230.4 kilobit phone lines instead of 50 kilobit lines. A short, fast link was incorporated into the network for a brief period and no problems were encountered. To date, network loading has not justified upgrading any operational network circuits to 230.4 kilobits, but this will be considered as loading rises.

Substantial effort has gone into traffic and trouble reporting. A Network Control Center (NCC) has been built at Bolt Beranek and Newman Inc. in Cambridge, where a small dedicated Host computer receives reports each minute from every IMP on the network. Traffic summaries and status and trouble reports are then generated from this material. Specifically, a logger records any changes that the IMPs report to the NCC; it records line errors and IMP storage counts when they exceed certain limits, as well as unusual events, such as reloading from the net, checksum errors, etc. Figure 3 shows an example of this log. (The comments, in parentheses to the right, are not part of the log but have been added to explain the meaning of the entries.) In addition to this detailed log of interesting events, one may at any time obtain a quick summary of the status of the network. Finally, detailed and summary logs of Host and line traffic are produced. The NCC is a focal point not only for monitoring the network but also for testing and diagnosing remote troubles. Lines throughout the network can be looped from here in order to isolate difficulties. Personnel of the center coordinate all debugging, maintenance, repair and modification of equipment.

```
13ØØ     JUNE 16 197-     ARPA NETWORK LOG                PAGE 11

13ØØ  IMP     6:  HOST 1 UP                (Host 1 at MIT came up)
      IMP     1:  SS2 ON                   (Sense switch 2 was thrown at UCLA)
13Ø1  IMP     1:  1Ø SEC STAT ON           (UCLA is using IMP statistics)
13Ø5  IMP     1:  1Ø SEC STAT OFF          (UCLA has finished)
      IMP     1:  SS2 OFF                  (and turned the switch off)

13Ø7  IMP     4:  UP *****                 (Utah IMP was down, and has come up

      IMP     4:  RELOADED FROM NET        (A neighbor IMP sent Utah a copy of
      IMP     4:  VERSION 2614             (the IMP program over a phone line)
      IMP     4:  HOST 1 UP                (Host 1 at Utah is now up)

131Ø  LINE    4:  UP *****                 (one of Utah's lines is up)

      LINE    1Ø: UP *****                 (another is up)

      LINE    15: DOWN *****               (but the third is making errors)

1317  LINE    15: ERRORS MINUS 13/81       (Utah sees 20% error rate)
      LINE    15: ERRORS PLUS 7/81         (the other end sees a 10% rate)
      IMP     6:  HOST 1 DN                (Host 1 at MIT went down)
132Ø  LINE    15: ERRORS MINUS Ø/81        (the line is error-free)
      LINE    15: ERRORS PLUS Ø/81         (in both directions)

1321  LINE    15: UP *****                 (the line is declared usable)

                  .
                  .
                  .
                  .
```

Figure 3—Typical segment of NCC log

DIRECT TERMINAL ACCESS

During the early phases of network development a typical node has consisted of one or more large time-shared computer systems connected to an IMP. The IMPs at the various sites are connected together into a subnet by 50 kilobit phone lines and the large Host computers communicate with one another through this subnet. This arrangement provides a means for sharing resources between such interconnected centers, each site potentially acting both as a user and as a provider of resources. *This total complex of facilities constitutes a nationwide resource which could be made available to users who have no special facilities of their own to contribute to the resource pool.* Such a user might be at a site either with no Host computer or where the existing computer might not be a terminal-oriented time-sharing system.

A great deal of thought went into considering how best to provide for direct terminal access to the network. One possibility, which would have essentially been a non-solution, was to require a user to dial direct to the appropriate Host. Once connected he could, of course, take advantage of the fact that that Host was tied to other Hosts in the net; however, the network lines would not have been used to facilitate his initial connection, and such an arrangement limits the terminal bandwidth to what may be available on the switched common-carrier networks.

A similar solution was to allow terminals to access the network through a Host at a nearby node. In such a case, for example, a worker in the New England area wishing to use facilities at a California site might connect into a local Boston Host and use that Host

as a tap into the network to get at the facilities in California. This approach would have required Hosts to provide hardware access facilities for many terminals which would use their systems only in passing. For many Hosts, the kinds of terminals which can be connected directly are limited in speed, character set, etc. In terms of reliability, the user would have been dependent on the local Host for access; when that Host was down, his port into the network would be unavailable. Furthermore, the Hosts would have been confronted with all of the problems of composing terminal characters into network messages and vice versa as well as directing these messages to the proper terminals and remote Hosts. Time-sharing systems are generally already overburdened with processing of characters to and from terminals and many are configured with front end processors employed explicitly to off-load this burden from the main processor. Increasing the amount of such work for the Hosts therefore seemed unreasonable and would have resulted in limiting terminal access. Instead, a completely separate means for accessing the network directly seemed called for: an IMP with a flexible terminal handling capability—a *Terminal IMP*, or *TIP*.

One of the fundamental questions that arises when considering this problem is: what is the proper distribution of computational power among the remote big facility, the terminal processor, and the terminal? Shall the terminal processor be clever, have sizable storage, be user programmable, etc., or shall it be a simpler device whose basic job is multiplexing in a flexible way? Serious work with interactive graphics seems to require the terminal to include, or be in propinquity to, a user-programmable processor and considerable storage. To date, such work has primarily been done with terminals attached directly to a Host. Some elaborate terminals, such as the Adage, include a powerful processor. Other kinds of terminals, including Teletype-like devices, alphanumeric displays, or simple graphic displays (excluding serious interactive graphics), do *not* require a user-programmable local processor or significant local storage.

We believe that the great majority of potential groups needing access to the ARPA Network will not need powerful interactive graphics facilities. Further, for the minority that will need powerful terminals, we believe that individual terminals with built-in or accompanying processors will become more common. For these reasons, we decided that the terminal processor should be simple and not programmable from the terminals. *The computational load and the storage should be in the Hosts or in the terminals and not in the terminal processor.* This simple multiplexing approach is amenable to some standardization and is philosophically close to the original IMP notion of a standard nodal device.

Another major question we faced was whether to build a separate terminal handler and then connect it to the IMP or to build an integrated unit that was housed in a common cabinet and used the IMP processor. One advantage of the two-machine approach is that it isolates the IMP functions from the terminal functions, thereby providing a barrier of safety for the net. This approach also provides the processing power of two machines and a potentially greater degree of user freedom in modifying or writing programs. Another interesting reason for considering separate machines was to reduce the large cost associated with I/O equipment (such as line controllers) by making use of the extra processing power. We discussed with several manufacturers the possibility of bringing terminal wires "into" their processors and decoding the basic line information directly in software. However, even with some of the new state-of-the-art machines, like the Meta-4, with fast 90 nsec read-only memories to handle character decoding, the I/O cost was still high, and in large part the necessary I/O equipment was yet to be designed. We therefore concluded that it was still somewhat early to proceed in this fashion, and two processors did not appear to save I/O equipment.

The principal disadvantages of the two-machine approach were the higher initial cost, the difficulties of maintaining two machines, and the software problems of dealing with two machines. In particular, the communication between two machines would require two hardware Host interfaces, and two software Host/IMP programs. This would result in a much poorer communication between the IMP program and the terminal handling program than would exist in a single machine. In either case, one machine or two, the

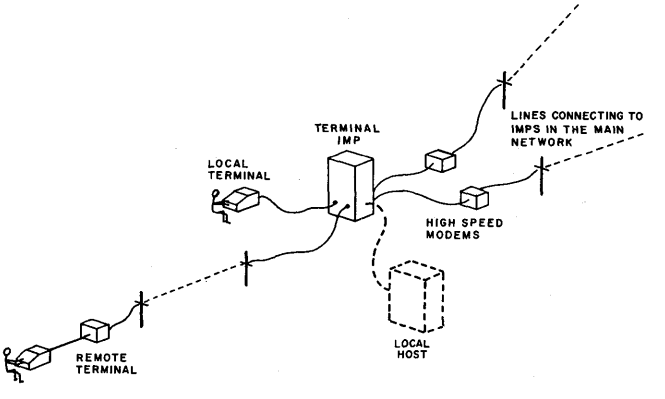

Figure 4—A TIP in the network

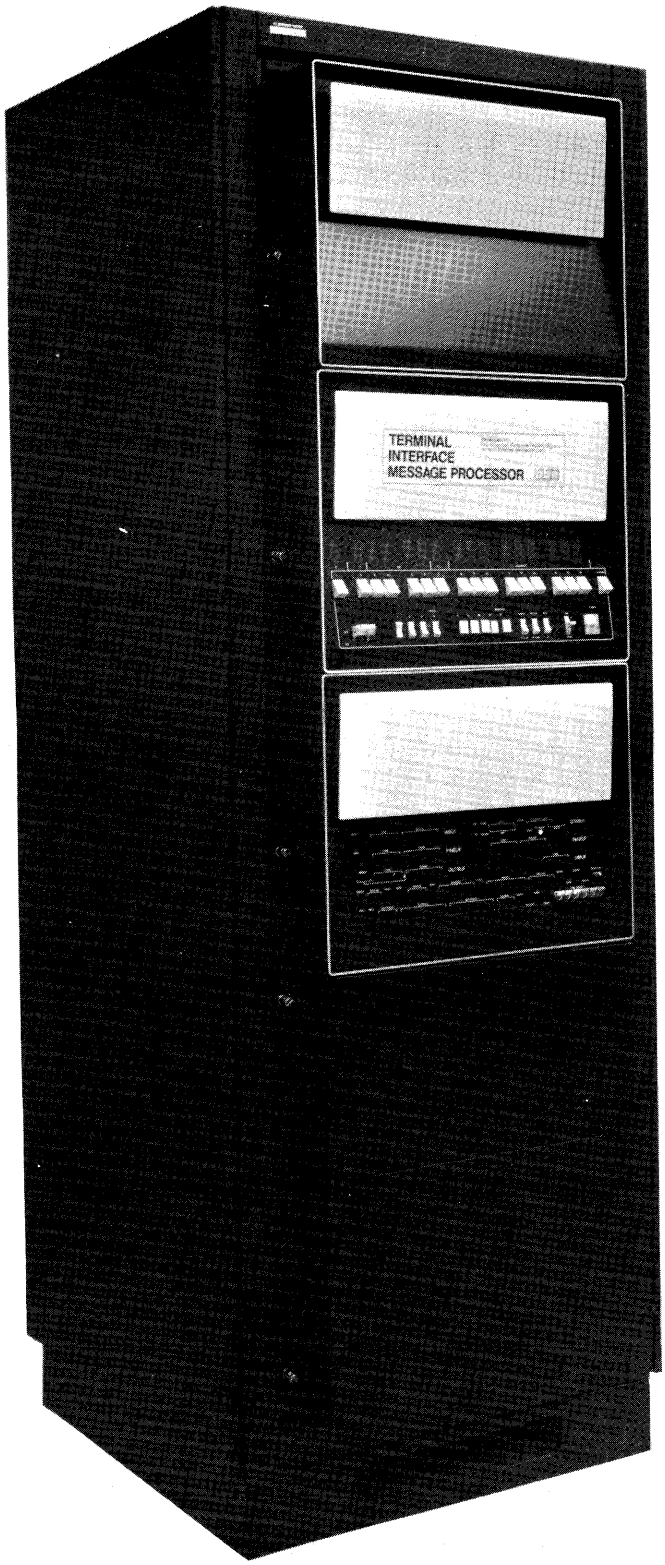

Figure 5—Photograph of TIP

terminal handling process required the implementation of a version of Host protocol.

We finally decided that the least expensive and most sensible technical alternative was to build the IMP and the terminal handler (with the necessary subset of Host protocol) together into a single machine. Then, since certain new machines appeared attractive in cost/performance relative to the Honeywell 16 series, we spent considerable time thinking about what machine to use. We decided that no alternate choice was sufficiently attractive to justify rewriting the main IMP program and redesigning the Host and modem interface hardware. This left us with a decision between the Honeywell 516 and 316 computers. We chose the 316 on the basis of size and cost, feeling that the somewhat higher bandwidth of the 516 was not essential to the job and did not justify its higher price and the second cabinet which would have been required.

TERMINAL IMP HARDWARE

Figure 4 shows how the TIP fits the user into the network. Up to 63* terminals, either remote or local, of widely diverse characteristics may be connected to a given TIP and thereby "talk" into the network. It is also possible to connect a Host to a TIP in the usual way.

The TIP is shown in Figure 5. It is built around a Honeywell H-316 computer with 20K (20,480 words) of core. It embodies a standard 16 port multiplexed memory channel with priority interrupts and includes a Teletype for debugging and program reloading. Other features of the standard IMP also present are a real-time clock, power-fail and auto-restart mechanisms, and a program-generated interrupt feature.[2] As in the standard IMP, interfaces are provided for connecting to high-speed (50 kilobit, 230.4 kilobit, etc.) modems as well as to Hosts. The single-cabinet version limits the configuration to a total of three modem and/or Host interfaces, but an expansion cabinet may be used to increase this limit. More basic limits are set by the machine's logical organization (specifically the number of available memory channels) and the program bandwidth capability as discussed below.

Aside from the additional 8K of core memory, the primary hardware feature which distinguishes the TIP from a standard IMP is a Multi-Line Controller (MLC) which allows for connection of terminals to the

* There are 64 hardware lines but line Ø is logically reserved by the program for special use.

IMP. Any of the MLC lines may go to local terminals or via modems to remote terminals. As shown in Figure 6 the MLC consists of two portions, one a piece of central logic which handles the assembly and disassembly of characters and transfers them to and from memory, and the other a set of individual Line Interface Units (all identical except for small number of option jumpers) which synchronize reporting to individual data bits between the central logic and the terminal devices and provide for control and status information to and from the modem or device. Line Interface Units may be physically incorporated one at a time as required.

The MLC connects to the high-speed multiplexed memory channel option of the H-316, and uses three of its channels as well as two priority interrupts and a small number of control instructions. The MLC is fabricated by BBN and is built from TTL integrated circuits. The MLC central controller, complete with power supply, is housed in an H-316 expansion chassis, and the entire MLC, as well as the computer itself, is mounted in a standard six-foot rack. Additional space is provided in the bottom of the rack for up to sixteen card-mounted modems of the Bell 103, 201, or 202 variety, together with their power supplies.

In order to accommodate a variety of devices, the controller handles a wide range of data rates and character sizes, in both synchronous and asynchronous modes. Data characters of length 5-, 6-, 7-, or 8-bits are allowed by the controller. Since no interpretation of characters is done by the hardware, any character set, such as Baudot, Transcode ASCII or EBCDIC may be used.

The following is a list of data rates accepted by the controller:

SYNCHRONOUS	*ASYNCHRONOUS* (Nominal Rates)		
Any rate up to and including 19.2 Kb/s	75	1200	
	110	1800	
	134.5	2400	
	150	4800	output only
	300	9600	
	600	19200	
	All above in bits/second		

The data format required of all devices is bit serial and each character indicates its own beginning by means of a start bit which precedes the data and includes one or more stop bits at the end of the character. This per-character framing is quite standard for asynchronous lines but synchronous lines, generally designed for higher bandwidths, frequently adopt some form of "binary synchronous communication" where the characters are packed tightly together into messages which are then framed by special characters. Framing is thereby amortized over the entire message, thus consuming a smaller fraction of the available bandwidth than the per-character framing which uses two or more bits for every character. The difficulties with this scheme, however, are that it is more complex, requiring more sophisticated hardware at each end, and that no real standards exist which are adhered to by all or even most types of synchronous devices. We therefore decided to adopt per-character framing with start and stop bits even on synchronous lines. At a cost of some twenty percent of the bandwidth for framing, a very simple and general scheme is thus arrived at. A number of high speed terminal manufacturers, faced with the same problems, have arrived at a similar conclusion.

Given these characteristics, then, the controller will connect to the great majority of normal terminal devices such as Teletypes, alphanumeric CRT units, and modems, and also (with suitable remote interface units) to many peripheral devices such as card readers, line printers, and graphics terminals. Either full or half duplex devices can be accommodated. The standard TIP program cannot deal with a magnetic tape unit through the MLC. However, as a special option, and with the use of additional core memory, standard Honeywell tape drives can be connected to the TIP as normal peripherals.

The individual terminal line levels are consistent with EIA RS-232C convention. Data rates and character length are individually set for each line *by the the program.* For incoming asynchronous lines, the program includes the capability for detecting character length and line data rate as discussed below.

Logically, the controller consists of 64 input ports and 64 output ports. Each input/output pair is brought out to a single connector which is normally connected to a single device. However, by using a special "Y"

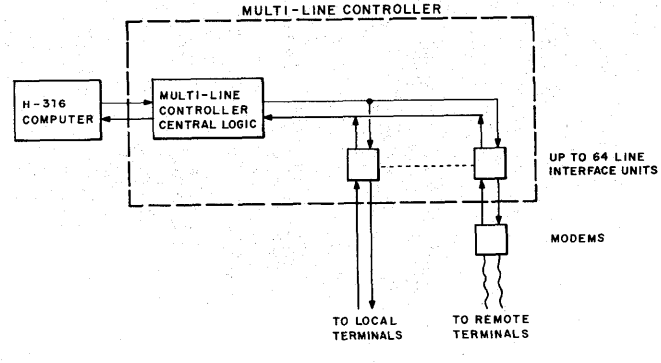

Figure 6—Block diagram of TIP hardware

cable, the ports may go to completely separate devices of entirely different properties. Thus, *input* port 16 may connect to a slow, asynchronous, 5-bit character keyboard while *output* port 16 connects to a high speed, synchronous display of some sort. In order to achieve this flexibility, the MLC stores information about each input and each output port and the program sets up this information for each half of each port in turn as it turns the ports "ON."

Several aspects of the MLC design are noteworthy. The central logic treats each of the 64 ports in succession, each port getting 800 ns of attention per cycle. The port then waits the remainder of the cycle (51.2 μs) for its next turn. For both input and output, two full characters are buffered in the central logic, the one currently being assembled or disassembled and one further character to allow for delays in memory accessing.

During input, characters from the various lines stream into a tumble table in memory on a first come, first served basis. Periodically a clock interrupt causes the program to switch tables and look for input.

Output characters are fed to all lines from a single output table. Ordering the characters in this table in such a way as to keep a set of lines of widely diverse speeds solidly occupied is a difficult task. To assist the program in this, a novel mechanism has been built into the MLC hardware whereby each line, as it uses up a character from the output table, enters a request consisting of its line number into a "request" table in memory. This table is periodically inspected by the program and the requests are used in building the next output table with the characters in proper line sequence.

The design of the terminal interface portion of the MLC is modular. Each Line Interface Unit (LIU) contains all the logic required for full duplex, bit serial communication and consists of a basic bi-directional data section and a control and status section. The data section contains transmit and receive portions each with clock and data lines. For asynchronous devices the clock line is ignored and timing is provided by the MLC itself. (For received asynchronous characters, timing is triggered by the leading edge of the start bit of each character.)

The control and status monitor functions are provided for modems as required by the RS-232C specification. Four outputs are available for control functions and six inputs are available to monitor status. The outputs are under program control and are available for non-standard functions if the data terminal is not a modem. For example, these lines could be used to operate a local line printer. RS-232C connectors are mounted directly on the LIU cards. To allow for variations in terminal and modem pin assignments, the signals are brought to connector pins via jumpers on the card.

The central MLC contains 256 ICs, many of which are MSI and some of which are LSI circuits, and it is thus about the same complexity as the basic H-316. In addition each LIU contains 31 ICs. A Terminal IMP including the MLC and with a typical interface configuration to high-speed circuits and Hosts is, order of magnitude, a $100,000 device.

THE TERMINAL USER'S VIEW

This section describes how a TIP user gains access to the network. The protocol described is of very recent origin and will undoubtedly change in response to usage which, as of this writing, is just commencing.

In general a user must have some foreknowledge of the resource which he expects to access via the network. The TIP program implements a set of commands[8] for connecting to and disconnecting from remote sites, but once a terminal is connected to a particular system, the TIP becomes transparent to the conversation unless specially signalled. This is equivalent to a time-sharing system where the executive program is essentially out of the picture during periods while the user is dealing directly with his own set of programs.

Because of the large number of different terminal types used in the network, the concept of the Network Virtual Terminal was developed. This is an imaginary but well-defined type of terminal. The TIP translates typed data to virtual terminal code before shipping it into the network, and conversely translates the remote system's response back into the local terminal's code. Thus, each Host system must deal only with this single terminal type.

When the user at a terminal needs to talk directly to his TIP instead of to the remote Host, he issues a command which is distinguished by the fact that it always starts with the symbol @. One or more words, perhaps followed by a single parameter, then identify the type of command.

Normally a user will go through four more or less distinct stages in typing into the net. First, he will be concerned with hardware, power, dialing in, etc. Then he will establish a dialogue with the TIP to get a comfortable set of parameters for his usage. Next, he will instruct the TIP to make a connection to a remote Host, and finally, he will mostly ignore the TIP as he talks to the remote Host.

One of the more interesting features of the TIP is that it permits great flexibility in the types of terminals which may be attached to any port. This

TABLE II—Tip Commands

CLOSE
close all outstanding connections

HOST #
focus attention on this host 0<#<256

LOGIN
start the initial connection procedure to get Telnet connections

SEND BREAK
send a Telnet break character

SEND SYNC
send a Telnet sync character and an INTERRUPT SENDER message

ECHO ALL
local TIP-generated echo—TIP echoes everything

ECHO NONE
remote Host-generated echo—TIP will echo commands

ECHO HALFDUPLEX
terminal-generated echo—TIP echoes nothing

FLUSH
delete all characters in input buffer

TRANSMIT EVERY #
send off input buffer at least every #th character 0<#<256

TRANSMIT ON EVERY CHARACTER
like TRANSMIT EVERY 1

TRANSMIT ON LINEFEED
send input buffer every time a linefeed encountered

TRANSMIT ON MESSAGE-END
send input buffer every time an EOM encountered

TRANSMIT ON NO CHARACTER
do not transmit on linefeed or EOM

TRANSMIT NOW
send off input buffer now

DEVICE RATE #
is a 13 bit code specifying hardware rate and character size settings

DEVICE CODE ASCII
DEVICE CODE 2741
DEVICE CODE EXECUPORT
DEVICE CODE ODEC
} establish code conversion

SEND TO HOST #
RECEIVE FROM HOST #
SEND TO SOCKET #
RECEIVE FROM SOCKET #
} establish parameters for manual initiation of connections

PROTOCOL TO TRANSMIT
PROTOCOL TO RECEIVE
PROTOCOL TO CLOSE TRANSMIT
PROTOCOL TO COSE RECEIVE
} initiate connection protocol manually

PROTOCOL BOTH
PROTOCOL TO CLOSE BOTH
} abbreviations for simultaneous transmit and receive protocols

GIVE BACK
release control of captured device

DIVERT OUTPUT
capture device # and divert this terminal's output to it

ABORT LOGIN
abort the outstanding initial connection procedure

flexibility presents a problem to the program which must determine what kind of terminal (speed, character size, etc.) is attached to a given port before a sensible exchange of information is possible. To solve this problem, each terminal is assigned a special identifying character which the user types repeatedly when starting to use a terminal. When information starts to appear from a previously idle port, the TIP enters a "hunt" mode in which it interprets the arriving characters trying various combinations of terminal characteristics. All except the proper combination will cause the character to be garbled, producing something other than one of the legal terminal identifying characters. When the TIP thus identifies the correct set (which generally requires the user to repeat the identifier less than half a dozen times), it types out 'HELLO in the terminal's own language.

In the next stage, the user initializes certain conversation parameters relating to message size and when to echo. He then establishes connection to a remote site using two commands which identify the desired remote site and the fact that the user wishes to be connected to the logger at that site. These commands are:

@ HOST 15

@ LOGIN

The LOGIN command actually sets in motion an elaborate exchange of messages between the TIP and the remote Host which normally result in a connection being made to that remote system. This command will be answered by an appropriate comment to the user indicating either that a connection has been made, that the remote site is not up, that it is up but actively refusing to converse, or that it is up but not responding enough even to refuse the connection.

Once a connection is established, the user types directly to the logger. The TIP does not execute the actual login since this procedure varies from site to site.

Throughout the user-to-Host dialogue, commands remain available at any time. Prior to closing the connection, the user must log out as required by the system he is using. He then gives the command

@ CLOSE

which causes the TIP to close the connection, informing the user when the process is finished.

In addition to the above more or less standard procedures, there are a number of less usual commands which set such things as device rate, character size, code types, etc. Such commands are used by a terminal on one port in setting parameters for a non-interactive terminal, such as a printer or card reader, on some

other port. Other special commands permit conversations directly between terminals on the same or different TIPs, allow for binary mode, etc. Table II is a list of the commands with a brief explanation. For details refer to Reference 8. Figure 7 gives an example of typical dialogue.

THE SOFTWARE

Because the terminals connected to a TIP communicate with Hosts at remote sites, the TIP, in addition to performing the IMP function, also acts as intermediary between the terminal and the distant Host. This means that network standards for format and protocol must be implemented in the TIP. One can thus think of the TIP software as containing both a very simpleminded mini-Host and a regular IMP program.

Figure 8 gives a simplified diagrammatic view of the program. The lower block marked "IMP" represents the usual IMP program. The two lines into and out of that block are logically equivalent to input and output from a Host. The code conversion blocks are in fact surprisingly complex and include all of the material for dealing with diverse (and perverse) types of terminals.

As the user types on the keyboard, characters go, via input code conversion, to the input block. Information for remote sites is formed into regular network messages and passes through the OR switch to the IMP program for transmittal. Command characters are fed off to the side to the command block where commands are decoded. The commands are then "performed" in that they either set some appropriate parameter or a flag which calls for later action. An example of this is the LOGIN command. Such a command in fact triggers a complex network protocol procedure, the various steps of which are performed by the PROTOCOL block working in conjunction with the remote Host through the IMPs. As part of this process an appropriate special message will be sent to the terminal via the Special Messages block indicating the status (success, failure, etc.) of the procedure.

```
1                Character typed just after terminal turned on.  Used to
                 identify the terminal type for the Terminal IMP program.

HELLO            TIP indicates that terminal is ready for use.

@ HOST 1         User tells Terminal IMP which Host to connect to.

                 (Terminal IMP echoes extra carriage return line feed to
                 indicate it has accepted a command.)

@ LOGIN          User tells Terminal IMP to form a connection.
                 (TIP echoes extra carriage return line feed.)

T R OPEN         Terminal IMP has formed a connection.

LOG ON"          This is UCLA E7's greeting message.                    }
XXXX             User identification.                                   }
!                        .                                              }
S" .ABACUS:C             .                                              }
.                        .                                              } Dialogue between
.                        .                                              } terminal user
.                        .                                              } and UCLA E7.
X                        .                                              }
STOP                     .                                              }
NORMAL EXIT              .                                              }
!                        .                                              }
X                User exits from UCLA E7.                               }
LOG ON"          E7 closing message indicates user has logged off.      }

@ CLOSE          User detaches the Terminal IMP from computer being used
                 (i.e. closes connection).
                 (TIP echoes extra carriage return line feed.)

T R CLOSED       Terminal IMP has closed the connection.
```

Figure 7—Typical terminal user dialogue

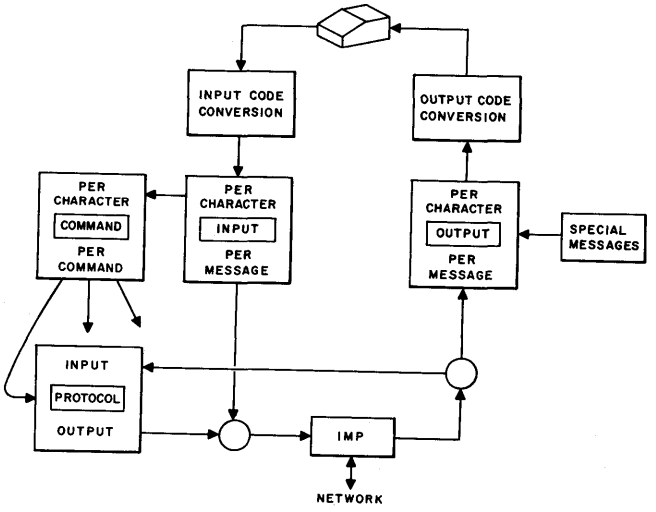

Figure 8—Block diagram of TIP program

Once connection to a remote Host is established, regular messages flow directly through the Input block and on through the IMP program. Returning responses come in through the IMP, into the OUTPUT program where they are fed through the OUTPUT Code Conversion block to the terminal itself.

Storage for the TIP program, including the standard IMP program, is just over 20K. This is roughly as follows:

Standard IMP Program with buffers & tables	12,000
Special TIP code	2,650
Tables of Parameters	1,800
Buffer Storage for messages to and from terminals	1,880
Miscellaneous—I/O buffers, constants, etc.	1,880

PERFORMANCE

The program can handle approximately 100 kilobits of one-way terminal traffic provided the message size

is sufficiently large that per-message processing is amortized over many characters. Overhead per message is such that if individual characters are sent one at a time there is a loss of somewhere between a factor of ten and twenty in bandwidth. A different way to look at program performance is to observe that the per-character processing time is about 75 μs.

These figures ignore the fact that the machine must devote some of its bandwidth to acting as an IMP, both for terminal traffic and for regular network traffic. About 5% of the machine is lost to acting as an IMP, even in the absence of traffic. If there is network traffic, more of the machine bandwidth is used up. Five hundred kilobits of two-way phone line and Host traffic saturates the machine without any terminal traffic*.

In addition to bandwidth which goes into the IMP part of the job, another 10 percent of the (total) machine is taken up simply in fielding clock interrupts from the Multi-Line Controller. This again is bandwidth used in idling even with no actual terminal traffic.

The following formula summarizes, approximately, the bandwidth capabilities:

$$P + H + 11T \leq 850$$

where:

- P = total phone line traffic (in kilobits/sec) wherein, for example, a full duplex 50 Kb phone line counts as 100;
- H = total Host traffic (in kilobits/sec) wherein the usual full duplex Host interface, with its usual 10 μs/bit setting, counts as 200; and
- T = total terminal traffic (in kilobits/sec) wherein an ASCII terminal such as a (110-baud) full-duplex Teletype (ASR-33) counts as twice its baud rate (i.e., 0.220 Kb).

This means that it takes eleven times as much program time to service every terminal character as it does to service a character's worth of phone line or Host message.

A further factor that influences terminal traffic handling capability has to do with the terminals themselves. Certain types of terminals require more attention from the program than others, independent of their speed but based rather on their complexity. In particular, for example, while an IBM 2741 nominally runs at 134.5 bits per second, the complexity is such that it uses nearly three times the program bandwidth that would be used in servicing a half-duplex ASCII terminal of equivalent speed. Allowances for such variations must be made in computing the machine's ability to service a particular configuration. It must be borne in mind that all of these performance figures are approximations and that the actual rules are extremely complex, and will change as the program matures.

* The number 500 kilobits is for full size (8000 bit) messages. Shorter messages use up more capability per bit and thus reduce the overall bandwidth capability.

DISCUSSION

As the ARPA Network grows, a number of areas of development seem likely to require attention. Certain of these are already pressing problems whereas others will begin to appear primarily as the network continues to grow and mature.

Perhaps the most difficult aspect of a network such as this is the problem of reliability. A great deal of thought and effort has gone into trying to make the system reliable and network topology has been designed with the need for redundancy in mind. Nonetheless the problem of keeping a large number of computer systems going at widely separated sites is non-trivial. An IMP's mean-time-between-failure (MTBF) has been on the order of one month; considerably lower than expected. The problems arise from a variety of sources and the system is sufficiently complex that frequently the cause has been masked by the time the failure has been noted. Often the same failure will recur several times until the problem is identified and eliminated and this fact tends to decrease the apparent MTBF. In general a preponderance of the troubles stem from hardware failures, and we are currently modifying several noisy and/or marginal portions of the I/O and interface hardware in hopes of obtaining significant improvement.

Our strategy has generally been to spend time identifying the source of trouble, keeping the machine off the air if necessary, so that eventually the MTBF will increase. This has often meant, however, that a "down" was much longer than it would have been, had the foremost goal been to get the machine running again immediately. With this strategy the average down time has been about 9 hours, giving rise to an average per-machine down rate of about 2 percent. We hope to improve this situation in the near future by providing improved facilities for obtaining "program dumps" when a failure occurs. This will make it possible to bring machines back on the air almost immediately in many cases, without jeopardizing valuable debugging data.

A word about our experience with the phone lines appears appropriate here as well. We have apparently been an unusual, if not unique, customer for the common carriers in our degree of attention to line failures. Our ability to detect and report even brief outages has led through measured skepticism to eventual acceptance by the carriers. In general, tests have indicated that the phone line error rates are about as predicted, i.e., approximately one in 10^5. These occasional errors or error bursts do not appear to affect network performance adversely. The IMP-to-IMP error checking procedure has not, to our knowledge, admitted a single erroneous message. We have, however, had some difficulty with lines which were simply out of commission from time to time for extended periods (hours or even days). The reliability of the phone lines is roughly equivalent to the reliability of the IMPs, based on the number and duration of outages. Down times have decreased as the carriers have come to accept our complaints as generally legitimate. *Overall the performance of the telephone equipment does not appear to constitute a problem in network growth.*

From a strictly technical viewpoint we view the incorporation of higher bandwidth facilities as a natural and key part of network growth. While the present facilities are not saturated by the present loads, we view this situation as a temporary one and something of a period of grace. As the network continues to grow over the next few years traffic can be expected to increase in a very non-linear fashion. Host protocol procedures (which have presented a sizable stumbling block to usage) are settling down so that software commitments can be made, and the advent of the Terminal IMP will bring an influx of new users who do not have the alternative of local resources. As a result we expect that traffic will begin to saturate some parts of the network. Terminal IMPs may well be called upon to service a larger number of terminals of higher bandwidth than can be handled by the present version.

In anticipation of these requirements we are presently considering the design of a significantly faster and more modular version of the IMP. Its higher speed will permit it to take advantage of high speed (i.e., megabit and above) communication lines which the common carriers are currently developing. More generally it will be able to service high bandwidth requirements whenever and however they occur within the net. We are currently studying a modular design which will permit connection of a greater number of interfaces than the present IMP can handle. While this work is only in the preliminary design stage, we feel that the interest and enthusiasm which have greeted the initial network suggest that it is not too early to consider ways to cope with growing traffic requirements.

Another area of network development which has already received a great deal of attention and will require much more in the future is that of technical information interchange between the sites. To the user, the resources which are becoming available are staggering if not overwhelming. It is very difficult for an individual to discover what, if any, of the mass of programs that will be available through the network may be of interest or of use to him. To aid in this process a number of mechanisms are being implemented and we are cooperating closely with these efforts. In particular, at the Stanford Research Institute a Network Information Center (NIC) acts essentially as a library for documentation concerning the network. Here are maintained a number of pointer documents, specifically, (1) a Directory of Participants which lists critical personnel, phone numbers, etc., for each participating site, (2) a catalogue of the NIC Collection which lists the available documents indexed in a variety of ways, (3) a Resource Notebook which is a compendium that attempts to familiarize the reader with available program resources at each of the participating sites.

Finally, there is the broad and exciting issue of how to cope with success. As the ARPA Network grows, and as diverse resources and users join the net, it is clear that a technology transfer must occur; the network probably must shift from a government-supported research and development activity to an ongoing national *service* of some kind. However, the computer-communications environment in the U.S. is rather complex, and is populated with many legal, economic, and political constraints. Within this environment, it is not easy to perform the technology transfer, and many groups, including ARPA and BBN, have been considering possible alternative plans. We are optimistic that a way will be found to provide a suitable legal and political base for expansion of the present ARPA Network into a widely useful public communication system.

ACKNOWLEDGMENTS

In addition to the authors, a great many people have contributed to the work reported here. Dr. L. G. Roberts of the Advanced Research Projects Agency has continued to lend crucial support and encouragement to the project. Messrs. Blue, Dolan, and Crocker of the ARPA office have been understanding and helpful. Suggestions and helpful criticism have come from many present and prospective participants in the network community.

At BBN, W. B. Barker is responsible for much of the cleverness which appears in the MLC hardware; R. E. Kahn has contributed in many areas and has been deeply involved in studying traffic flow control; A. McKenzie has compiled the Network Resource Notebook and been concerned with Host relations and many Host protocol issues; M. Thrope has managed the Network Control Center and has kept the network on the air; J. Levin helped in implementation of the TIP software; B. P. Cosell has patiently labored with the IMP software, with help from J. McQuillan and M. Kraley; and R. Alter has offered much helpful advice and criticism.

REFERENCES

1 L G ROBERTS B D WESSLER
Computer network development to achieve resource sharing
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

2 F E HEART R E KAHN S M ORNSTEIN W R CROWTHER D C WALDEN
The interface message processor for the ARPA computer network
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

3 L KLEINROCK
Analytic and simulation methods in computer network design
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

4 H FRANK I T FRISCH W CHOU
Topological considerations in the design of the ARPA computer network
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

5 C S CARR S D CROCKER V G CERF
Host-host communication protocol in the ARPA network
AFIPS Conference Proceedings Spring Joint Computer Conference 1970

6 S CROCKER et al
Function-oriented protocols for the ARPA computer network
AFIPS Conference Proceedings Spring Joint Computer Conference 1972

7 R E KAHN W R CROWTHER
Flow control in a resource sharing computer network
Proc Second Symposium on Problems in the Optimization of Data Communications Systems October 1971

8 *User's guide to the terminal IMP*
Bolt Beranek and Newman Inc Report No 2183

9 *The BBN terminal interface message processor*
BBN Report 2184

10 L G ROBERTS B D WESSLER
The ARPA network
Computer Communication Networks Chapter 6 In preparation

11 L G ROBERTS
A forward look
Journal of Armed Forces Communications and Electronics Assoc Vol XXV No 12 August 1971 pp 77-81

12 *The MIT Lincoln Lab terminal support processor*
Graphics Semi-Annual Tech Summary Reports ESD-TR-70-151 p 355 May November 1970

13 *Progress report #1*
University of Michigan MERIT Computer Network Report #0571PR4 May 1971

14 A B COCANOWER W FISCHER W S GERSTENBERGER B S READ
The communications computer operating system—The initial design
University of Michigan MERIT Computer Network Manual #1070-TN-3 October 1970

15 R E KAHN
Terminal access to the ARPA computer network
Courant Computer Symposium 3 Computer Networks Courant Institute New York November 1970—Proceedings to be published by Prentice Hall Englewood Cliffs New Jersey In preparation