BOLT BERANEK AND NEWMAN INC

CONSULTING • DEVELOPMENT • RESEARCH

Report No. 1763   January 1969

INITIAL DESIGN FOR INTERFACE MESSAGE
PROCESSORS FOR THE ARPA COMPUTER NETWORK

Submitted to:

Advanced Research Project Agency
Washington, D. C. 20301

Attn: Dr. L. G. Roberts

CAMBRIDGE   NEW YORK   CHICAGO   LOS ANGELES

INITIAL DESIGN FOR
INTERFACE MESSAGE PROCESSORS
FOR THE ARPA COMPUTER NETWORK

January 1969

Submitted to:

Advanced Research Projects Agency
Washington, D. C. 20301
Attn: Dr. L.G. Roberts

TABLE OF CONTENTS

page

LIST OF FIGURES

INITIAL IMP DESIGN

A. Introduction

In this report we present our proposed system design. We begin
by describing the most important features of the design, followed
by a description of the overall hardware configuration of the
IMP. The main part of the document is devoted to a detailed de-
scription of the process of message communication, including the
primary aspects of network message flow and the selected network
protocol. We discuss the function of the IMP/MODEM Interface
and the IMP/Host Interface. The logical organization of the
IMP buffer storage is then described in detail. The potential
causes of network congestion are summarized along with the pro-
visions we have included for handling this situation. Next we
discuss line quality determination and rerouting. Questions of
fault detection, status examination, and reporting procedures
are also discussed. The end of the document is devoted to the
main program structure and the support software.

Our experience convinced us that it was wrong to plan for an
initial network that permitted a sizable degree of external and
remote control of IMPs. Consequently, as one important feature
of our design, we have planned a network composed of highly

autonomous IMPs. Once the network is demonstrated to be success-
ful, then remote control can be added, slowly and carefully.
Messages are processed by an IMP using information which has been
received from other IMPs and Host computers in the network, but
special control messages or other external control signals are
initially avoided to the greatest possible extent. One specific
consequence of this policy is that the IMPs measure performance
of the network on a regular basis and report in special messages
to the network measurement center (presumably at UCLA).

A second important feature of our design is the provision of a
tracing capability which permits the operation of the net to be
studied in great detail. Any message may contain a "trace bit",
and each IMP which handles such a message generates a special
report describing its detailed handling of the message; the col-
lection of such special reports permits reconstruction of the
history of such messages as they traverse the system. This
technique permits highly flexible sampled study of the network.

We have also included an automatic trouble reporting capability
which detects a variety of network difficulties such as line
quality deterioration, and reports them to an interested Host
(perhaps, the network measurement center).

A principal feature of our system is a provision for letting
IMPs throw away packets which they have received but have not
yet acknowledged. Each IMP transmits packets to other IMPs at
its own discretion. Each time an IMP receives *and accepts* a
packet it returns a positive acknowledgment to the transmitting
IMP. The transmitting IMP retains its copy of the packet until
it receives the positive acknowledgment. The transmitting IMP

will retransmit the packet if an acknowledgment is not received
within a time-out period. It will continue to try transmissions,
via a different route if necessary, until such time as a positive
acknowledgment is returned. We have explicitly avoided the use
of negative acknowledgments which we feel are insufficient and
consequently redundant.

We have carefully provided for the preservation of natural word
boundaries in transmissions between computers with equal word
sizes (a thing which, despite intuition, does not tend to "hap-
pen naturally"). We introduce a technique of padding and mark-
ing which neatly and generally allows the beginning and end of
a message to be clearly indicated to a destination Host without
requiring the Host programs to count bits. [Although we have
made an effort to provide a network protocol that allows the
Hosts a great deal of flexibility, this is a difficult technical
area, and we would plan to examine further the problems associ-
ated with Host-Host word reformating.]

Another important feature of our design is a hardware modifica-
tion to the IMP computer that permits the program to set an
interrupt. This trick permits *three* levels of priority in the
operational program (interrupt routines, urgent task routines,
and background), which, in turn, has an important bearing on
the IMP Program's ability to handle occasional time-consuming
word-rate tasks (such as ASCII conversion, or other data trans-
formation).

The Host computers have a few responsibilities for participation
in the network. Specifically, the Host must provide a network-
linking Program within its operating system to accept standard

format network messages and to generate network messages in accordance with this standard format. The Host message includes identification information that accompanies the message from the source to the final destination. The Host computer must not present a message of over 8080 bits to the IMP. Larger transmissions must therefore be broken up by a Host into a sequence of such messages.

The network is carefully designed to protect and deliver messages from the source Host to the destination Host. The operation is self contained, and does not in any way constrain the procedures a Host may use in communicating with other Hosts.

B. General Discussion of the IMP

The overall configuration of an IMP includes a Honeywell DDP-516 computer, which has a 0.96 µs cycle-time, a 16 bit word length and 12K of memory (expandable), 16 channels of priority interrupts (expandable), a relative-time clock, and a 16 channel data multiplexor as shown in Fig. 1. Also shown are several special interfaces, specifically one to the Host, and one to each modem. A paper tape reader has been included because we feel a very strong need for a device which does not depend upon the network or any Host computer for the loading of an IMP program. We believe that this is a simple, reliable and inexpensive way to read in new versions of a program during the initial phases of network operation. A teletype is required for maintenance of the IMP computer, but is not used by the main program and can be disconnected and removed during normal operation. A specially designed set of status-indicator lights are provided
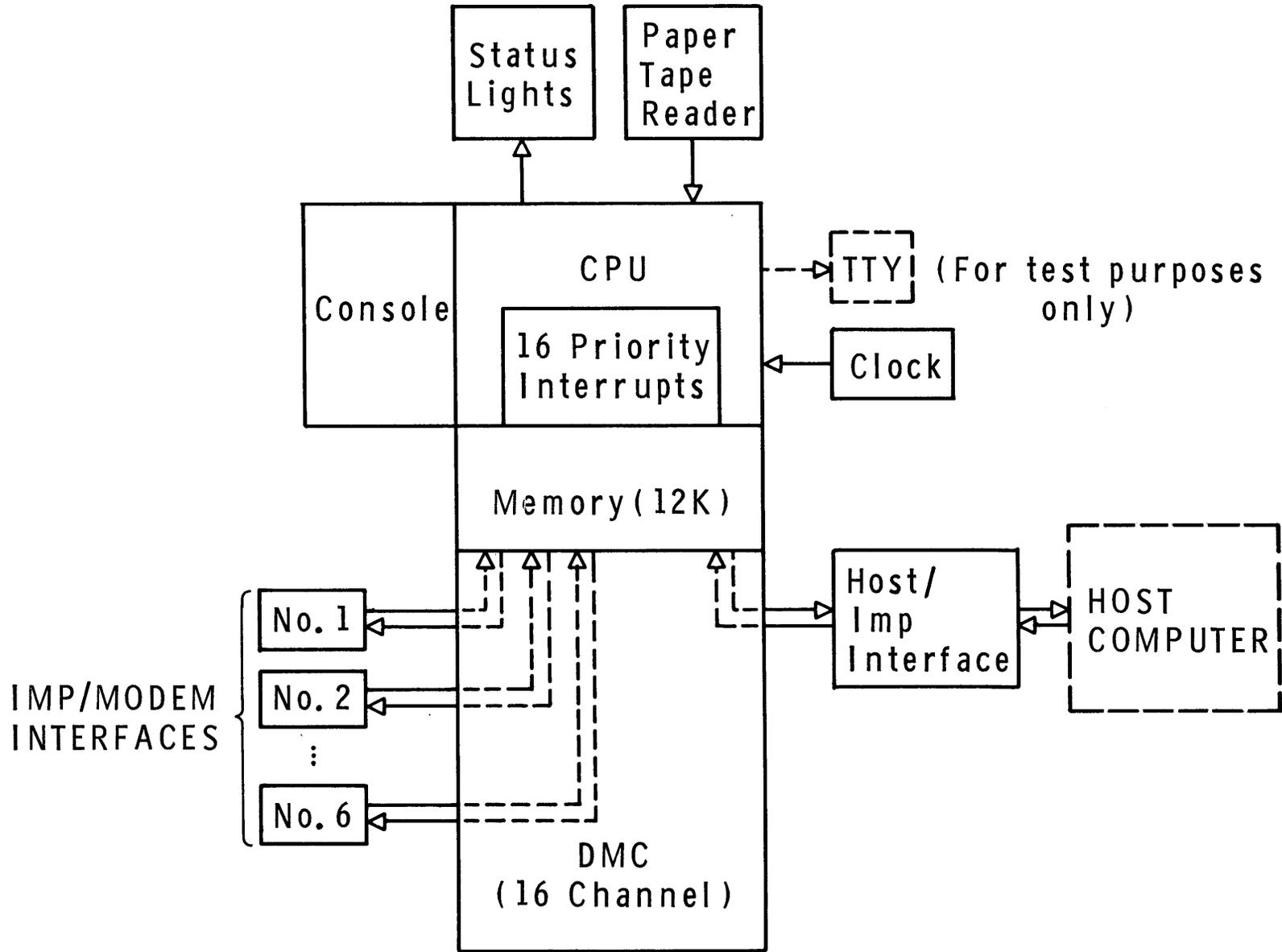
FIG. 1   IMP CONFIGURATION.

for use by the IMP program to report trouble conditions to local
Host personnel or to maintenance personnel without necessitating
a halt in normal program operation.

The IMPs in the initial network will each have three built-in
full duplex modem interfaces, but the interface design is modular
and may be extended up to as many as six units, without a change
in packaging.

The IMP, including all interface hardware, will be packaged in a
single 69" × 24" × 28" rugged cabinet. (See Plate I.)

C. Host-Host Protocol and the Notion of Links

It is important to draw a sharp line between the responsibility
of the network facilities in transmitting information and the
responsibility of the Host organization for developing and
adopting procedures for utilizing this facility. However, in
considering the system design, it became clear that we would
have to pay some degree of attention to limitations that the
network protocol might place on the Host use of the network.
We reached the conclusion that a network protocol that satis-
factorily achieves the transmission requirement might nonethe-
less adversely affect the implementation by Host organizations
of certain very desirable protocol features.

We considered the problems introduced when a multiplicity of
user programs at a given Host installation are concurrently us-
ing the network and concluded that provisions for allowing such
usage were rather important. The Host computers view the net-
work as a means for passing messages back and forth between
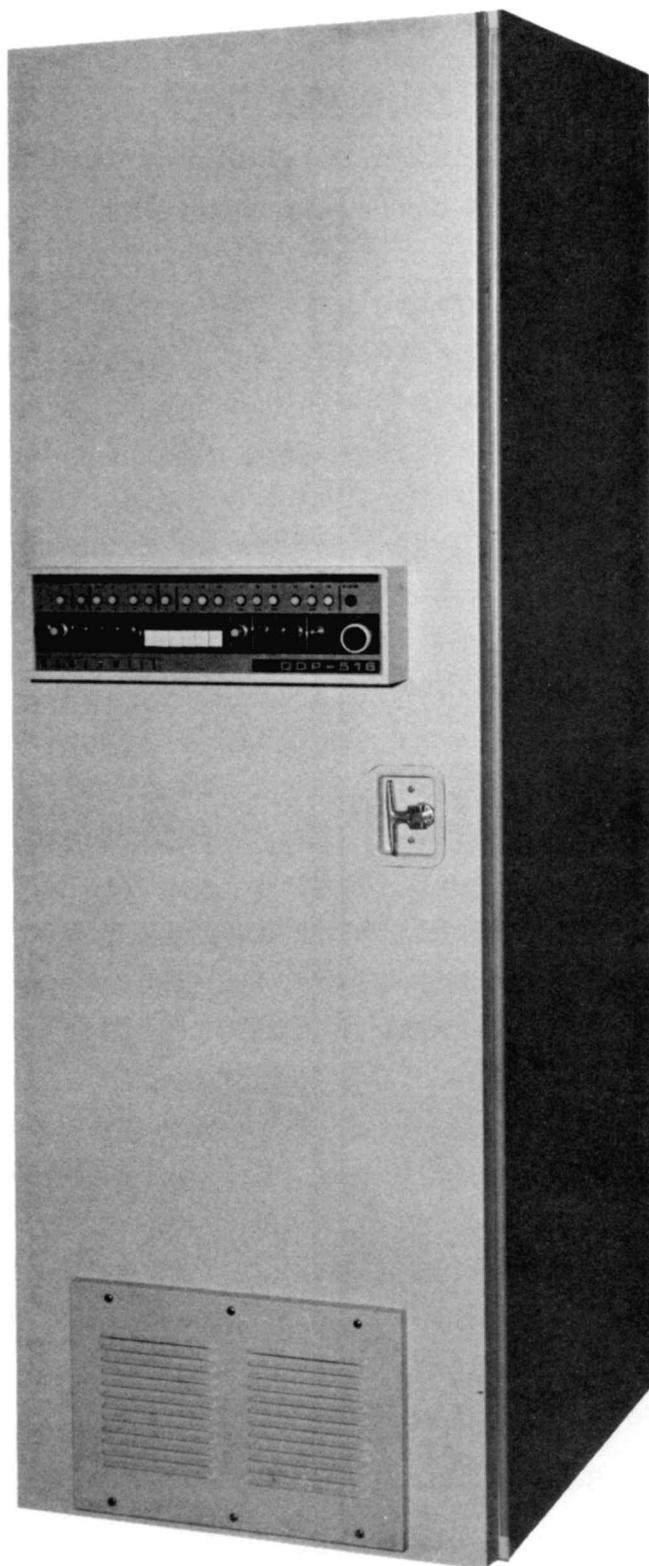
PLATE 1.   THE IMP.

parties rather than between pairs of Host computers themselves.
We call a logical connection between two parties at remote Host
computers a *link*. Many different links may exist simultaneously
between a pair of Host computers. As illustrated in Fig.2,
our network protocol permits many concurrent links to time-
share the same physical network facilities. These links are
established, identified, and maintained by a network program in
each Host computer that effectively multiplexes outgoing mes-
sages from the parties into the network and distributes incoming
messages to the appropriate parties as illustrated in Fig. 3.
Writing and maintaining the Host's network program is, of course,
the responsibility of the individual Hosts.

An identification number is assigned by each Host computer to
each network party in his machine. The party that initiates a
link is known as the *caller*. The identification number of the
caller is used as an identification number for the link and, in
conjunction with the identity of the two Host computers, uniquely
identifies the link. Each message which the Host network pro-
gram presents to the network contains several pieces of informa-
tion used by the network. One of these is the link identifica-
tion number. The network uses this number to control the flow
of messages and passes it along to the receiving Host.

A message is designated by its link and its direction of travel.
(Source and destination are terms which identify the direction
of travel.) Thus, complete identification for a message con-
sists of the following four items:

1) Identity of Source Host;

2) Identity of Destination Host;
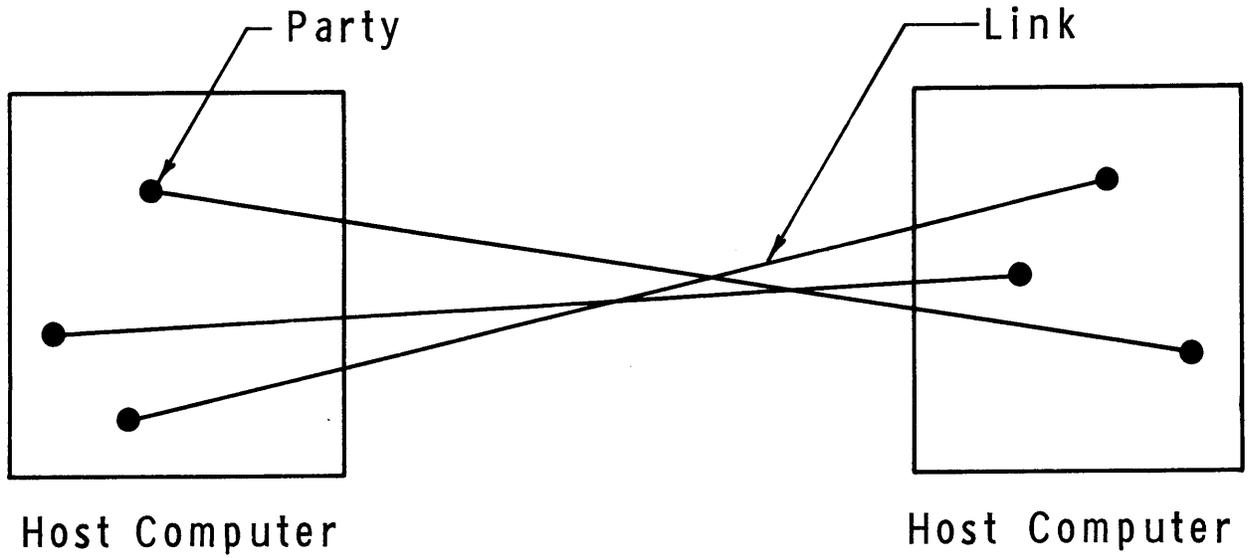
FIG. 2   MULTIPLE HOST-TO-HOST LINKS.
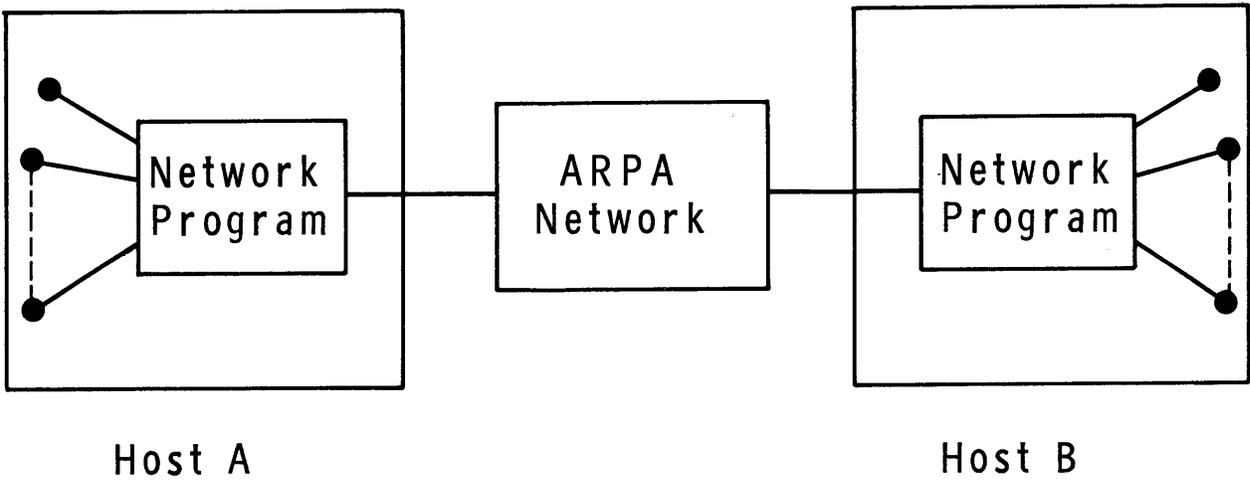
FIG. 3   MULTIPLEXED HOST-TO-HOST LINKS.

3) Link identification number; and

4) Caller location (at source or at destination).

For example, if party n in Host A calls Host B, the message will
be identified as going from source A to destination B and the
caller for the link will be party n at the source. A return
message from Host B on this link is identified as going from
source B to destination A and the caller for the link will be
party n at the destination.

We introduce the notion of a *link* early in this design discus-
sion primarily because we wish to include the link identifica-
tion number as an integral part of the identification informa-
tion passed from Host to IMP, from IMP to IMP in the network,
and finally from the destination IMP to the destination Host.

D. Messages and Packets; HOST-IMP, IMP-IMP, and IMP-HOST Proto-col

Hosts communicate with each other via sequences of messages. A
message is taken into an IMP from its Host computer in segments.
These segments are formed into packets and separately shipped
out by the IMP into the network. They are reassembled at the
destination IMP and delivered in sequence to the receiving Host,
who obtains them as a single unit. Thus the segmentation of a
message during transmission is completely invisible to the Host
computers.

The transmitting Host attaches identifying information to the
beginning of each message which it passes to its IMP. The IMP
forms a *header* by adding further information for network use.
The header is then attached to each segment of the message.

The transmitting hardware computes parity check digits that are
shipped with each segment and that are used for error detection.
The destination IMP performs an error check, strips off the
header from each segment in the course of reassembly and attaches
identifying information at the beginning of the reassembled
message for use by the destination Host.

A message from a Host is legislatively limited to be less than
8080 bits, and is sent to its IMP via a single block transfer.
The hardware interface detects the end of the block transfer.
Messages vary in size up to the 8080 bit limit. The first six-
teen bits of each message which a Host sends to an IMP for a
transmission are prescribed by the standard network protocol as
follows:

> Eight bits are allocated to the link identification
> number, five bits are allocated to identifying the
> destination Host, one bit is presented for tagging
> selected messages which are to be traced through
> the network, and two bits are reserved as spares.
> The tracing is discussed more fully in a later sec-
> tion. The format for these 16 bits of Host infor-
> mation is illustrated in Fig. 4.

The HOST/IMP Interface transfers bits serially from the Host and
forms them into 16 bit IMP words. The IMP program takes groups
of successive words in segments and stores them in separate
buffer regions until the end of the message has been recognized.
The first buffer accepts up to 64 IMP words from the Host (1024
bits including the 16 bits of Host information). Each succeed-
ing buffer accepts up to 63 words (1008 bits). Thus, the maxi-
mum Host message of 8080 bits will be taken by the IMP in ex-
actly 8 segments.

11
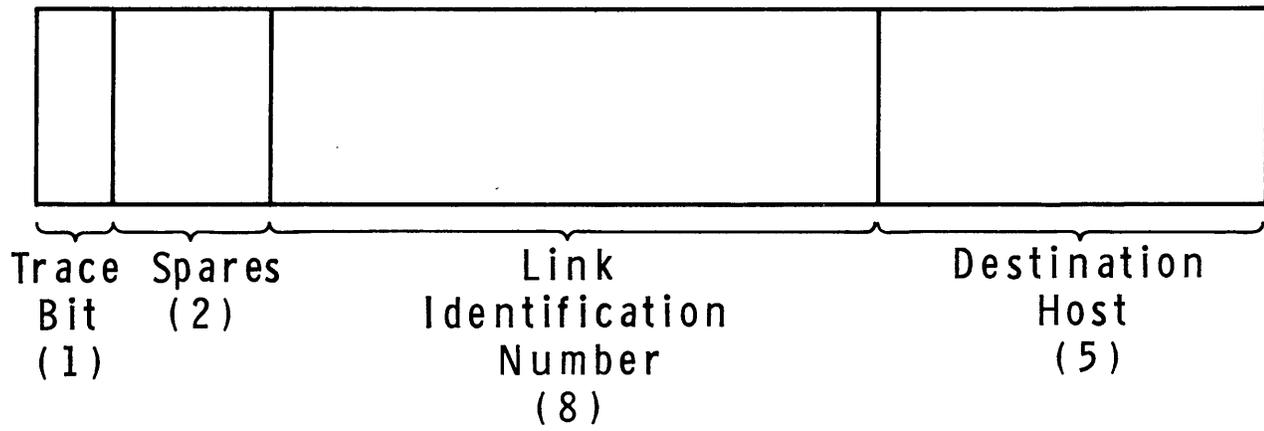
FIG. 4   HOST-TO-IMP INFORMATION FORMAT.

The IMP now formats each segment into a *packet* for transmission into the network. The structure of a formatted packet as it appears in the originating IMP memory is shown in Fig. 5. The output hardware prefaces the packet into the phone line with the character pair DLE STX to mark the packet beginning for the receiving channel hardware. The packet is then transmitted serially over the communcation lines beginning with the left most bit of the first header word and proceeding through the header and the text. The channel hardware computes 24 parity check digits, which it attaches after the packet, immediately following two ASCII control characters DLE ETX to mark the end of the packet for the receiving channel hardware.

A continuous stream of the ASCII control character SYN is transmitted by the channel hardware between packet transmissions. These are used to separate packets and to obtain character synchronization in the receiving channel hardware. Thus the packet appears on the communication line as shown in Fig. 6.

The receiving channel hardware locks into character synchronization on a bit-by-bit search for an 8 bit SYN code. Once synchronization has been obtained, the channel hardware looks for the first occurrence of DLE STX and succeeding characters are fed into the IMP memory until the DLE ETX at the end of the packet is detected. The hardware also computes a 24 bit error check based upon the received data, which should equal zero if no errors have occurred in transmission.

The received data between the STX and the DLE is written into the IMP memory and appears in the buffer as shown in Fig. 7.

I/I Ack — ⌐RFNM

Last
Packet
in
Message

Packet
for Imp

		Acknowledgment Pointer
	Packet No.	Message No.
	Spares	Source
T	Spares	Link# or Control Code	Destination

⎫
⎬ Header
⎭

Text

⎫
⎬ 63
⎭ Words
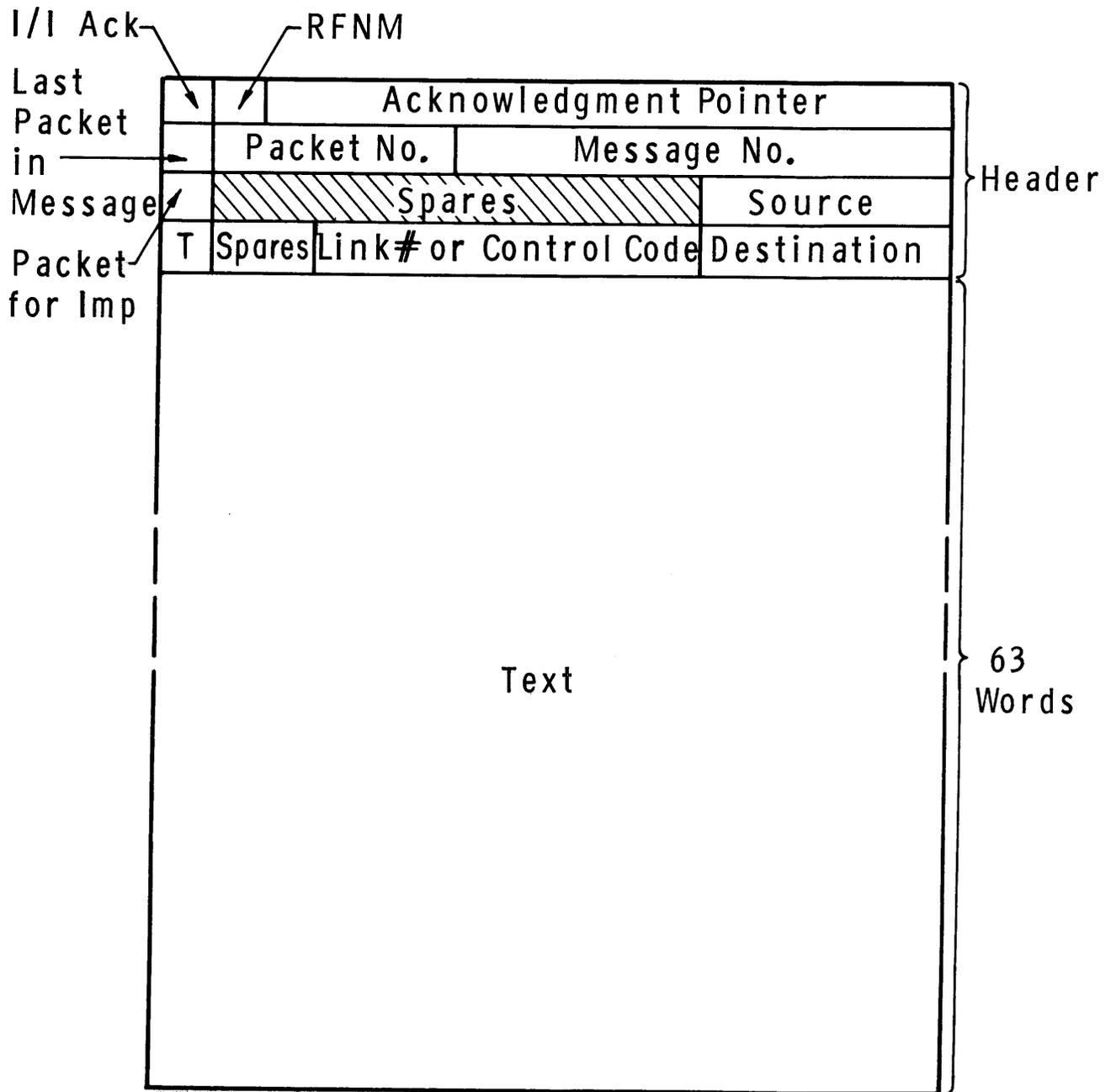
FIG. 5   ORIGINATING IMP PACKET STRUCTURE.

```
      S S D S                          D E C C C S S
· · · Y Y L T    Header      Text     L T C C C Y Y · · ·
      N N E X_____    _____E X 1 2 3 N N
```
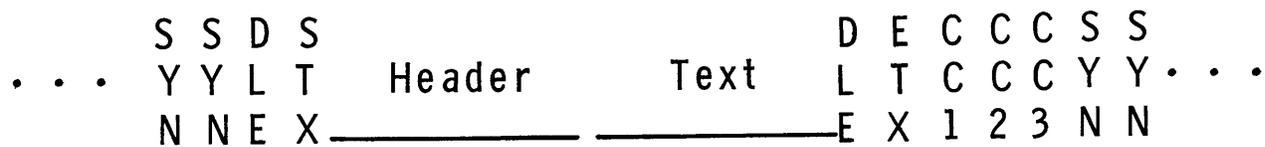
FIG. 6   COMMUNICATION LINE PACKET FORMAT.
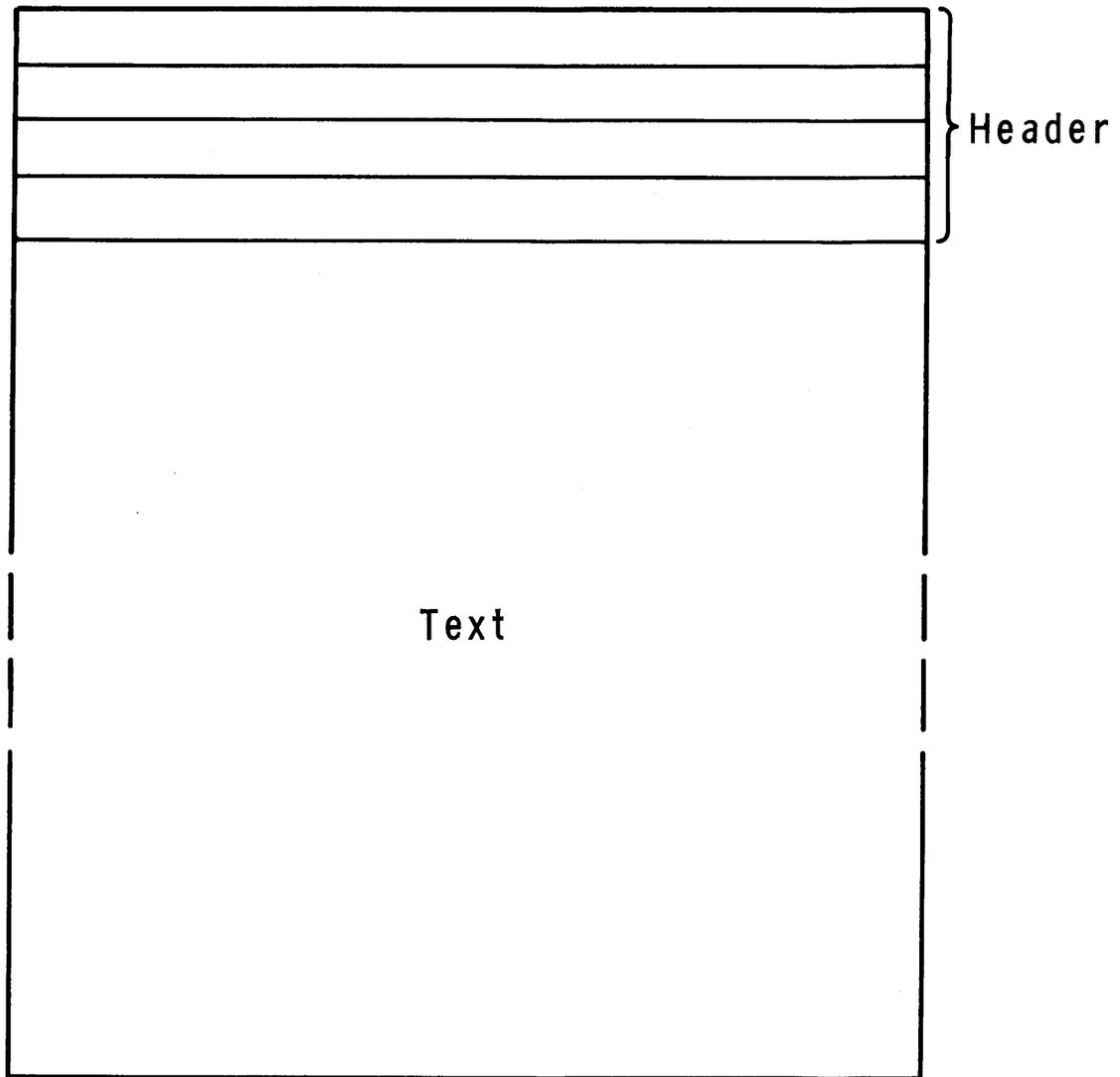
FIG. 7   PACKET FORMAT AS RECEIVED FROM MODEM INTERFACE.

If the receiving IMP is not the final destination, the header
and the following text is fed to the appropriate output channel
hardware. The channel hardware recomputes 24 parity check digits
and appends these as described earlier, together with the DLE
STX and the DLE ETX.

Eventually, the packet will arrive at the destination IMP. In
fact, eventually all the packets of the message will arrive at
the destination IMP, although not necessarily in the order of
transmission.

The destination IMP sorts received packets according to the link
identification as specified in the header. When all packets of
the message have arrived, it delivers them in the proper order
to its Host.

Packets within a given message are numbered sequentially by the
transmitting IMP in the second word of the header and the last
packet is specially marked by an identifying bit in the same
word. This allows the receiving IMP to determine the order of
the packets and to know when all packets have been received.

The receiving IMP strips off the header from each packet before
sending it on to the Host. Furthermore, 16 bits are sent to the
Host preceding the text of the first packet. The Host network
program uses these bits to identify the link in sorting incoming
messages. The format for these 16 bits is shown in Fig. 8.

Thus, the complete message is finally delivered to the destina-
tion Host in the same form as it left the transmitting Host, with
the source in place of the destination in the Host information.
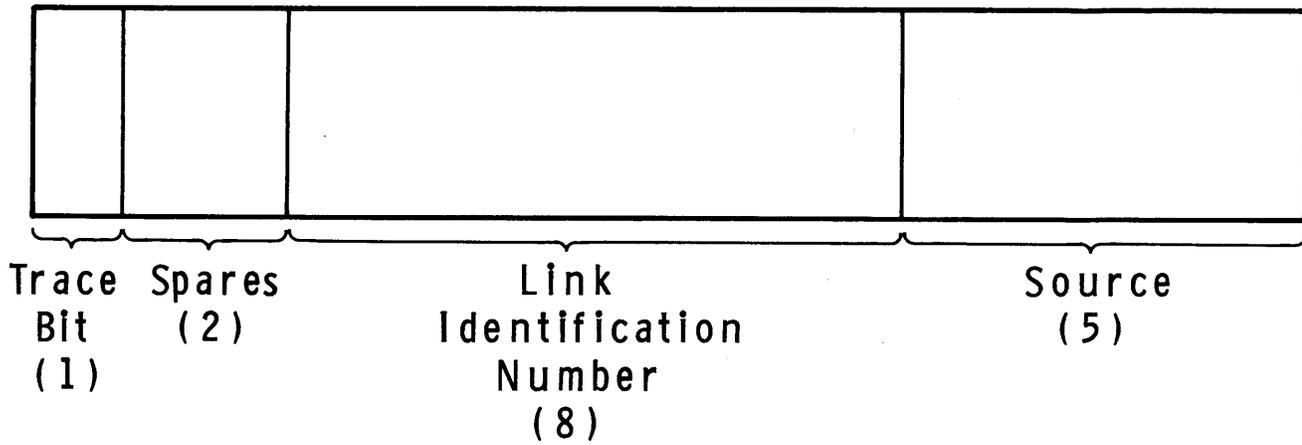
FIG. 8   IMP-TO-HOST INFORMATION FORMAT.

E. Acknowledgment Procedures

We now discuss two kinds of messages which will be used to con-
trol flow in the network: "IMP-to-IMP acknowledgments," and
end-to-end "Requests For Next Message."

1. IMP-to-IMP acknowledgment of packets

The process of communicating a message from the source to the
destination IMP uses the store and forward services of inter-
mediate IMPs. As a packet moves from one IMP to the next, it
is stored in each IMP until a positive IMP-to-IMP acknowledg-
ment message is returned from the succeeding IMP. This ackow-
ledgment indicates that the packet was received without error
and was accepted. The acknowledgment is returned over the same
line on which the packet arrived. A 14 bit acknowledgment
pointer, containing the memory address of the first word of the
transmitted packet, is included in the header of the packet to
simplify the process of releasing that packet when acknowledged.
(The packet identity data are checked before releasing the
packet; the acknowledgment pointer simply avoids searching.)

To send an acknowledgment of a received packet, an IMP simply
returns a packet (without text) whose header is an exact copy of
the header of the received packet, but with the first bit of the
first word changed to a one. This bit is called the IMP-to-IMP
acknowledgment bit and is the first item sensed by the IMP pro-
gram upon receipt of every packet. (The source and destination
do not apply in the usual way to the acknowledgment message it-
self.)

Once an IMP has accepted a packet and returned a positive ac-
knowledgment, it hangs on to that packet tenaciously until it,
in turn, receives an acknowledgment. Under no other circum-
stances (except Host or IMP malfunction) will an IMP discard a
packet after it has generated a positive acknowledgment. How-
ever, an IMP is always free to discard a packet by simply not
returning a positive acknowledgment. It may do this for any of
several reasons: the packet may have been received in error,
the IMP may be busy, the IMP buffer storage may be full, and so
forth.

Packets which are not recognized by the receiving channel hard-
ware, which incur errors in transmission, or which are not ac-
cepted for whatever reason, are not acknowledged. At the trans-
mitting IMP, the situation is readily detected by the absence of
a returned acknowledgment within a reasonable time interval.
Such packets are simply retransmitted.

Acknowledgments are themselves not acknowledged, although of
course they are error checked in the usual fashion. Loss of an
acknowledgment results in the eventual retransmission of the
packet. The resulting duplication is sorted out at the destina-
tion IMP by use of the message number and packet number in the
header.

There are no negative acknowledgments in our proposed design.
They cannot be relied on to induce retransmission. If a nega-
tive acknowledgment is lost, one must resort to a time out pro-
cedure, in which case, the negative acknowledgment becomes re-
dundant. Since the time out procedure must, therefore, always
be used, we include it in our design.

2. Request-For-Next-Message (RFNM)

A central concern of network protocol is the problem of conges-
tion at a destination IMP. This congestion must be reflected
back into corrective quenching of the flow toward that point
from other parts of the net. Otherwise, it would give rise to
the discard of packets at the destination, blockage of those
packets at the contiguous IMPs and the congestion would rapidly
propogate back through the network. If the sources of packets
for that destination continue sending, this congestion would
rapidly affect the flow of other messages within the net.

There are at least two kinds of quenching which could be adopted.

1) We could limit the *degree* of congestion of remote IMPs that
    can be caused by any particular congested Host or link. For
    example, if each IMP only accepted, say, two messages for
    any given destination, the congestion would be limited to
    that amount and, eventually, the source would be unable to
    transmit additional new packets toward the troublesome
    destination.

2) We could try to limit congestion at the source directly by
    shutting off any new packets directed toward the trouble-
    some destination. This action could be accomplished in
    either of two ways: a control message could be dispatched
    when congestion actually has occurred, or successive trans-
    missions could routinely require a "clear-to-send" indica-
    tion from the destination.

Although we have tried to avoid control messages in our design
wherever possible, we decided in this case initially to use the
control message technique. *We propose to avert congestion, by*

*only allowing a source IMP to send one message at a time over a
given link.* After sending a message over a link, a source IMP
must delay sending the next message until a "Request for next
message over link X" (RFNM) packet is end-to-end returned from
the destination IMP. (Note that all packets of a single message,
and/or messages over different links between the same two Hosts,
may be sent into the net without delay.) The RFNM is passed
along to the Host, who may use it to schedule the servicing of
links. This technique only quenches individual links and there-
fore a limit is placed on the total number of links which a
transmitting IMP will accept from its Host.

This technique has several important advantages and two disadvan-
tages. The advantages are:

1) The demand for reassembly storage at the destination IMP for
   use by a given link is limited to eight packets.

2) When congestion occurs, flow is *automatically* quenched with-
   out any control messages. If source IMPs do *not* get new
   RFNM's, they do *not* send new messages.

3) Since the flow is quenched at the source, large numbers of
   packets from a given link neither enter the net nor flow
   about the net trying to get to the congested destination.
   Thus, congestion of other parts of the net by a single link
   is avoided.

Obviously, the main disadvantage is that waiting for RFNM packets
may reduce the effective rate over a given single link. We have
examined this disadvantage and have decided that it is not seri-
ous, for the following reasons:

1) Depending upon the number of active links, there may or may
    not be a reduction of the effective rate between two Hosts.
    When several links are established in a given Host computer,
    the messages will be time multiplexed. The RFNM delay in
    that case may already naturally appear in the system.

2) Since the message length will probably be bi-modal (very
    short or very long) and since very short packets are prob-
    ably generated by humans, the RFNM delay is insignificant
    for processes at human rates. For very long messages, in
    the worst case of no time multiplexing and an unoccupied
    line, we estimate the reduction in effective rate to be
    only 30%.

A second disadvantage is the increase in number of control mes-
sages. Since RFNM's are very short, however, we feel that this
effect is also not serious.

The use of an RFNM control message is a very clean, simple, and
positive way to avoid some nasty and confusing problems. We are
not fully satisfied that the doctrine is optimum, but, so far,
we have been unable to see a clearly superior alternative. We
therefore propose to use RFNM control of congestion in the
initial design. During the implementation and testing, we will
continue to consider this issue in an attempt to determine
whether other alternatives appear to be more advantageous.

F. Examples of Message Flow

The chart on the following pages shows the flow of packets in-
volved in transmitting a message from one Host to another. The

EVENT				STATE OF THE NETWORK				
Comments	Packet	From	To	h1	i1	i2	i3	h3
Host 1 has two packets for Host 3				21				
	1	h1	i1	2	1			
	1	i1	i3	2	1		1	
	2	h1	i1		21		1	
Acknowledgment returned	1a	i3	i1		2		1	
	2	i1	i3		2		21	
	2a	i3	i1				21	
	12	i3	h3				r	21
RFNM goes back to h1	r	i3	i1		r		r	
RFNM also acknowledged	ra	i1	i3		r			
	r	i1	hi	r				
Host 1 has two packets for Host 3				21				
	1	h1	i1	2	1			
	2	h1	i1		21			
	1	i1	i3		21		1	
Packet 1 acknowledgment lost	1a	i3	i1		21		1	
	2	i1	i3		21		21	
	2a	i3	i1		1		21	
Packet 1 rerouted*	1	i1	i2		1	1	21	
	1a	i2	i1			1	21	**
Packet 1 arrives second time	1	i2	i3			1	21	
	1a	i3	i2				21	
	12	i3	h3				r	21
	r	i3	i1		r		r	
	ra	i1	i3		r			
	r	i1	hi	r				

EVENT				STATE OF THE NETWORK					
Host 1 has two packets for Host 3				21					
	1	h1	i1	2	1				
Error on line (i.e., Packet 1 does not get to i3)	1	i1	i3	2	1				
Packet 1 rerouted*	1	i1	i2	2	1	1			
	2	h1	i1		21	1			
	2	i1	i3		21	1	2		
	2a	i3	i1		1	1	2		
	1a	i2	i1			1	2		
	1	i2	i3			1	12		
	1a	i3	i2				12		
Packets 1 & 2 get sorted	12	i3	h3				r	21	
	r	i3	i1	r			r		
	ra	i1	i3	r					
	r	i1	h1	r					

LEGEND:

21 = 12 =	Packet 1 and Packet 2
1	Packet 1
2	Packet 2
1a	Packet 1 acknowledgment
2a	Packet 2 acknowledgment
h1	Host 1
i3	IMP 3
r	Ready for next message
ra	RFNM acknowledgment

*A time out period elapses before Packet 1 is rerouted. In the third example, other events which are not shown (because they are irrelevant for this example) prevent Packet 2 from being transferred from Host 1 to IMP 1 during this interval.

**In this example, the duplicate of Packet 1 merely overlays the one in IMP memory, effectively deleting it. The reassembled message could have entered the Host any time in the bracketed interval, before the arrival of the duplicate packet. In this case, the message number of the duplicate allows it to be discarded.

packets of the message, the acknowledgment packets, and the ready
for next message packet are indicated assuming that the message
being transmitted contains two packets.

The chart includes three examples: in the first, transmission is
completed without any problem; in the second, an IMP-to-IMP ac-
knowledgment for one packet is lost; and in the third, a packet
encounters difficulty due to line error. Although the events
within the examples are ordered, we emphasize that most of the
events occur asynchronously and could be ordered in many other
ways. Equal time does not pass between events.

The relevant portion of the network assumed for the examples is:
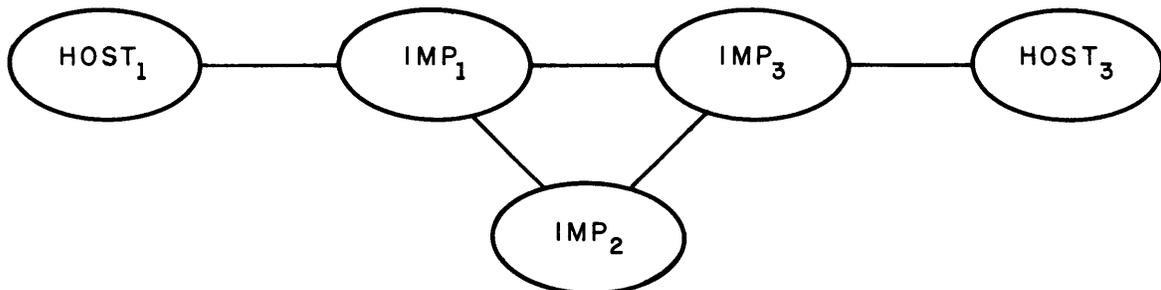
G. Word Length Mismatch

We discuss two aspects of word length mismatch: first, the ob-
vious need for formatting that occurs between computers of dif-
ferent word length; and second, since mismatched words may lead
to messages that end in the middle of words, the need for mark-
ing the exact beginning and ends of a message to permit unambigu-
ous recognition.

There are several logical ways in which the reformatting of a
word length mismatch might conceivably be handled. One may de-
cide upon a word-by-word algorithm, where transfers from long to
short machines involve truncation, and where transfers from short
to long machines deposit a partial word. Unfortunately, there
are many slightly different ways to do this and, worse, it is
very undesirable in many applications. A second possibility is
to list a number of kinds of reformatting and have a given mes-
sage carry a code for the required type of reformatting. We
feel that such a plan would be unreasonable for a 19 node net.
Finally, one may beg the question and just send a bit stream,
leaving to the individual Hosts the task of reformatting.

We have decided to adopt almost this latter position. Our de-
sign guarantees that between Hosts of identical word length the
natural word boundaries are preserved. (This is not as easy as
it sounds.) But, reformatting in general will be initially left
to the Hosts. At a later time, the IMP program might be used
to alleviate further this set of problems.

The second problem is that of recognizing the end of a message
at the receiving Host. There are two general solutions to this,
one of which is to locate the last bit in the message by count-
ing from the beginning (using either a transmitted count or an
agreed upon fixed value). The other general solution requires
that the ends be marked in an unambiguous way. We have chosen
the latter scheme, which marks the end of the message by ap-
pending a "one" followed by zeroes after the last bit in the
message. This process is called *padding* and is accomplished by
the hardware in the HOST/IMP interfaces. The receiving Host can
therefore identify the end of the message.

As a message passes from the transmitted Host to its IMP, the
hardware appends a one to the bit string when it receives the
end of message signal. This bit may fall, in general, in any
position of an IMP word somewhere in the last packet. The hard-
ware then fills any remaining bits of this word with trailing
zeros.   The format of the last packet of a message as it thus
appears in the IMP memory is shown in Fig. 9.

The packet appears in the destination IMP in exactly the same
format.

As the last packet is serially shifted into the Host through the
interface, the last bit from the IMP (which in our example is
the fifth trailing zero in the padding) will fall, in general,
somewhere in the middle of the receiving Host's final word.
The remaining bits in this word are filled in by the Host's
special interface hardware with additional trailing zeros.
(Note that a one is purposely omitted here.) Thus the packet
appears in the receiving Host with a one immediately following
the last bit in the message, followed by a string of zero or
more trailing zeros that terminate at a Host word boundary.
The last word in the receiving bit stream does not necessarily
contain the last bit in the message, as it may contain nothing
but padded zeros.

Another occasion for inserting a form of marking data arises at
the beginning of a message. The transmitting Host, in general,
arranges that the text of a message begins at a word boundary.
Since the network protocol requires the first 16 bits of a mes-
sage to contain Host information, there will thus, in general,
be a gap between the end of that identification and the beginning
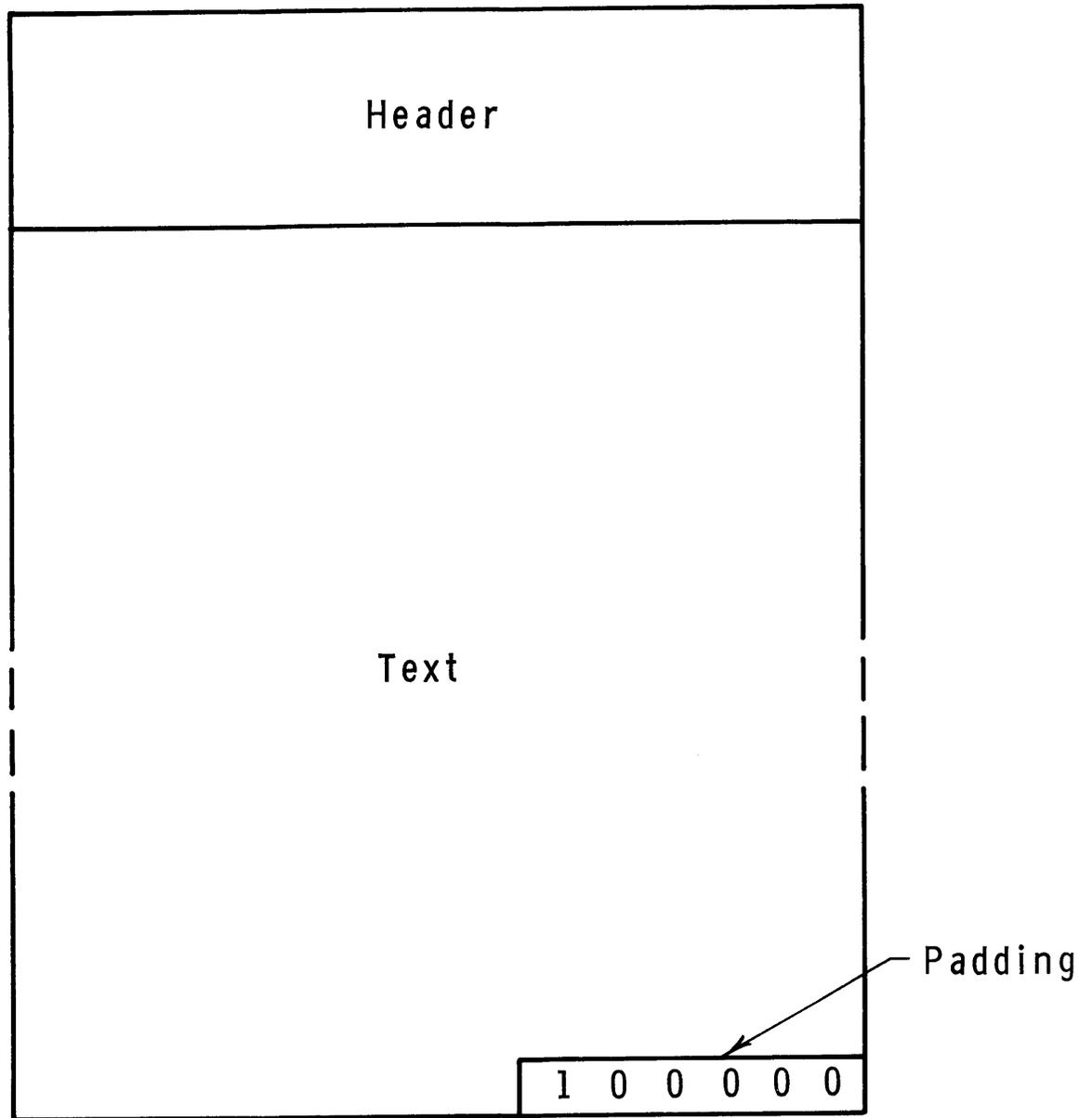
FIG. 9   FORMAT OF LAST PACKET OF A MESSAGE.

of the text. This gap is preserved in transmission to the desti-
nation Host and must be marked in a way which the destination
Host can recognize as not forming part of the message. This
marking must be inserted by the transmitting Host's software,
and consists of a one preceding the first bit of the text and,
in turn, preceded by a zero or more zeros to fill up the gap.

In Fig. 10 we illustrate one complete set of Host and IMP
buffers, corresponding to a message of slightly under two full
packets. We have selected in our example a 22 bit source Host
word length and a 20 bit destination Host. We have specifically
indicated both the padding and the marking in the figure.

H. Hardware Description and Interface Operation

A block diagram of the IMP computer and its interfaces to the
Host and phone line modems is shown in Fig. 11. The area be-
tween the heavy vertical lines shows the IMP system itself; the
area to the left is specialized Host equipment; the area to the
right is phone line equipment. There are from one to six full-
duplex IMP/MODEM interface units and one (or optionally two) HOST/
IMP interface unit. The DMC provides the only direct access to
and from memory, other than that for the CPU itself. The function-
ing of these units is described briefly in this section.

The IMP/MODEM Interface Unit is full duplex. It serializes and
deserializes data for the Modem to and from memory. In the
absence of outgoing messages, it loads a continuous string of
SYN characters onto the line. It does special formatting for
output, and character sensing for the beginning and end of input
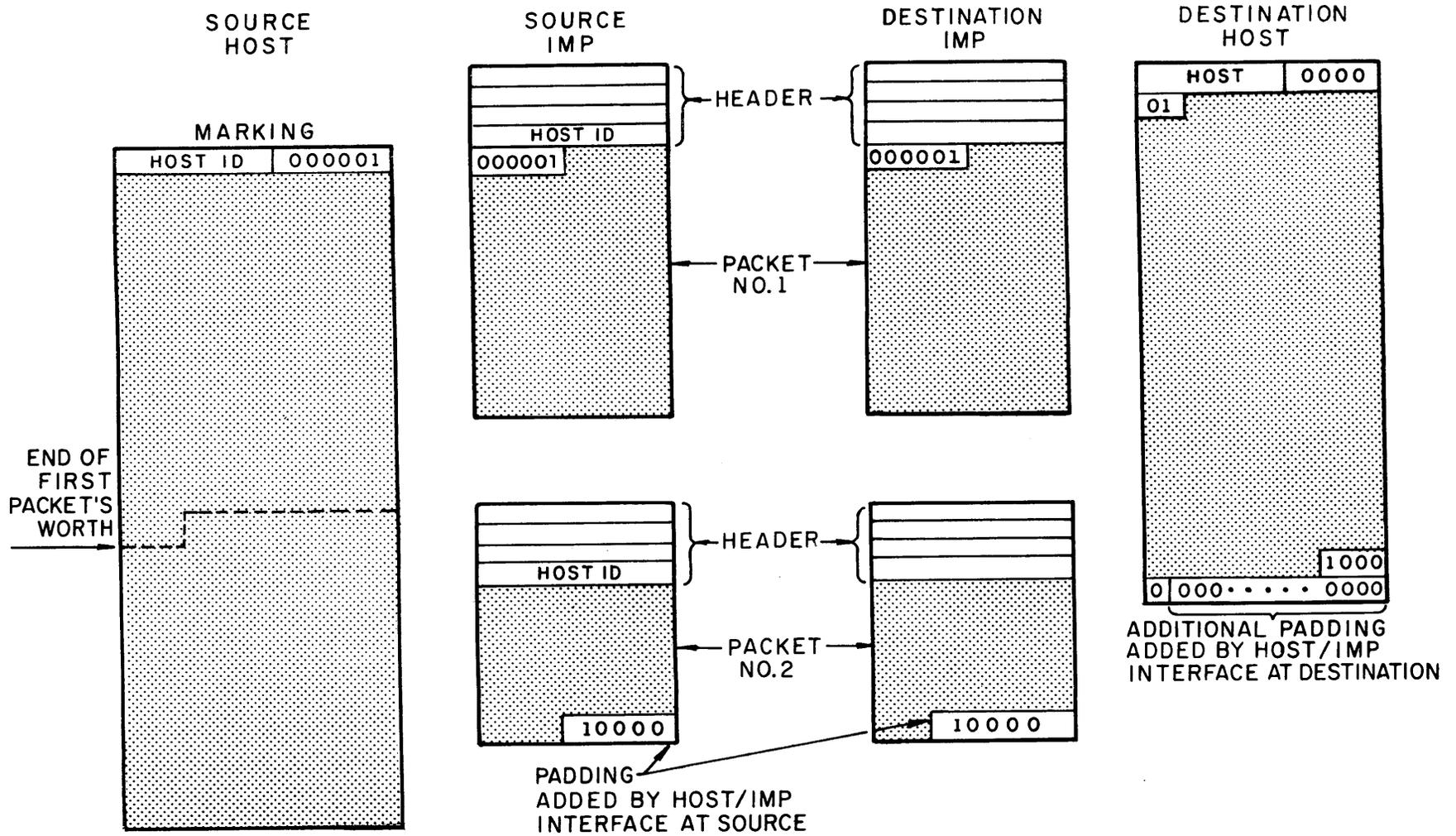
FIG. 10   HOST AND IMP BUFFER FORMAT FOR A TWO-PACKET MESSAGE.

IMP

CONTROL

CPU

MEMORY

CONTROL

HOST MEMORY

HOST SPECIAL INTERFACE

STD IMP/HOST INTERFACE

DMC

IMP/MODEM INTERFACE

MODEM

IMP/MODEM INTERFACE
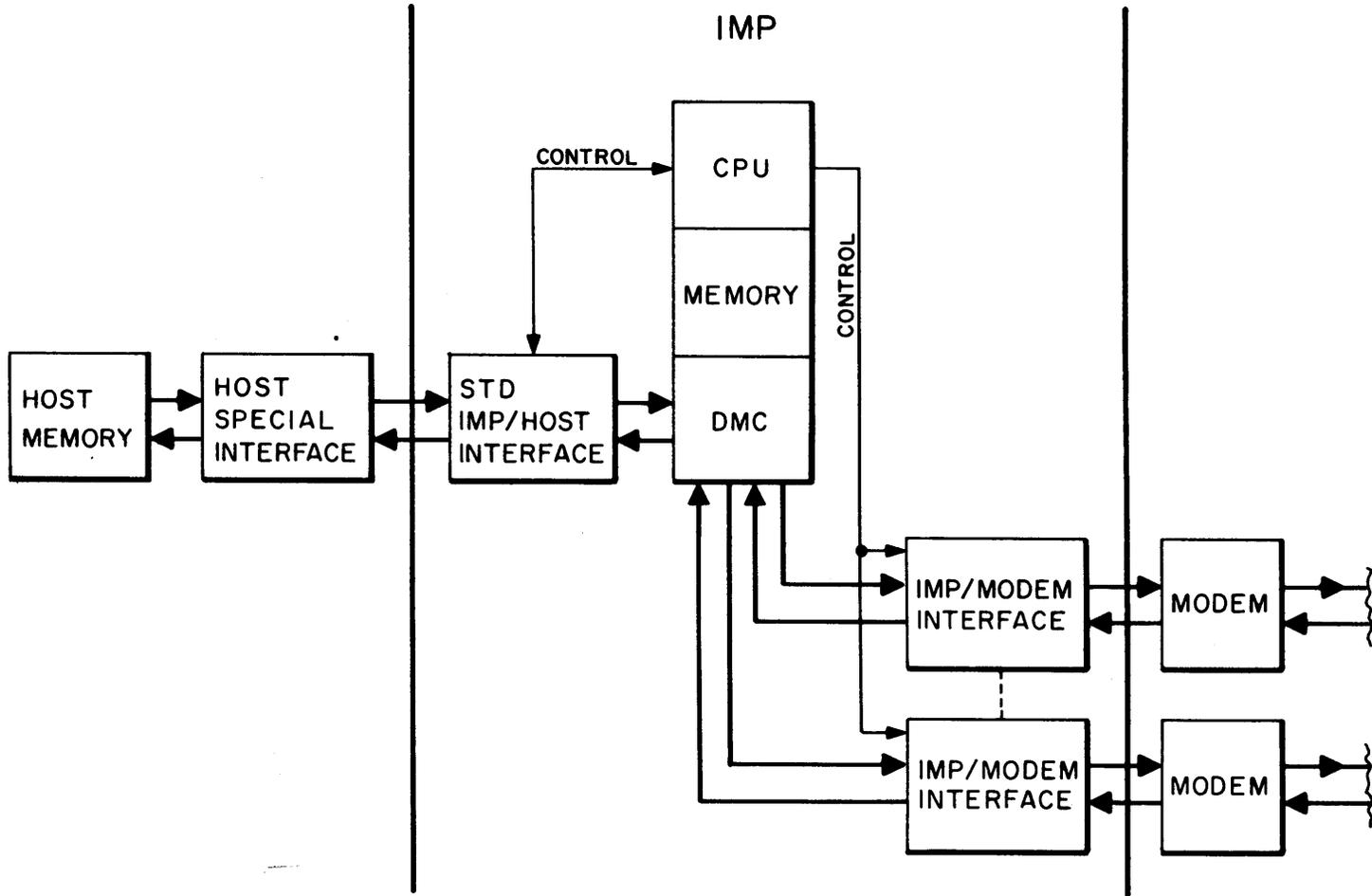
MODEM

FIG. 11   GENERAL VIEW OF A TYPICAL IMP SYSTEM.

messages. It includes construction and testing of parity check digits and fault detection and reporting. Its timing is controlled primarily by the Modem.

The standard HOST/IMP Interface Unit is full duplex and passes messages bit-serially to and from the Host special interface. It also deserializes and serializes words to and from the IMP memory. Communication across the interface with the Host is asynchronous to allow for maximum flexibility.

The relative-time clock is a 16-bit counter indexed every 20 μs and may be read into the Accumulator. The full clock count repeats approximately every 1.3 sec and an Interrupt is generated on the turnover of an appropriate high order bit. This bit is selected to give an interrupt frequency which is convenient for use by the program in performing time outs for retransmission of packets.

1. The HOST/IMP interface unit

There is no general rule whereby the HOST/IMP Interface Unit can determine in which direction (Host-to-IMP or IMP-to-Host) information will next have to be processed. The equipment must therefore be capable of starting a transmission in either direction. Transmission requests arrive asynchronously for the two directions and, rather than trying to sort them out for processing over a half duplex channel, a full duplex channel is provided. The primary advantage of this is simplicity and it also provides the capability for concurrent transmission in both directions. The HOST/IMP Interface is thus divided logically into two

parallel channels — one for either direction — as indicated in
the following figure.
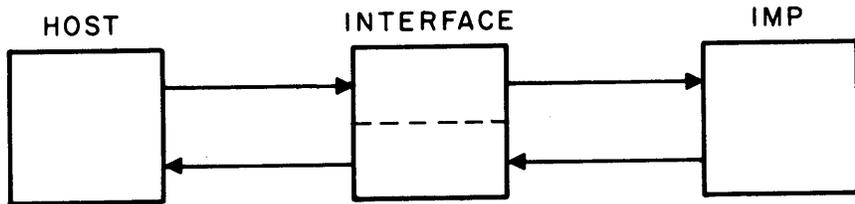
HOST   INTERFACE   IMP

Because Hosts vary in word length, signal forms, and logic for
receiving and transmitting information, we further subdivide
"vertically" the HOST/IMP Interface, into two separate units:

HOST   SPECIAL   STANDARD   IMP
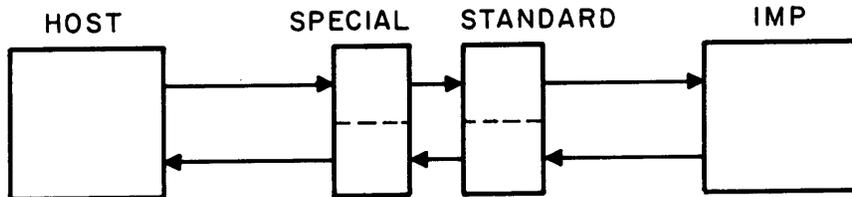
The right hand Unit contains logic that is standard for all
HOST/IMP Interfaces. The left hand unit contains the special
equipment for interfacing directly to the particular Host.
Standard signals pass between these two halves; all special
logic and signal adjustments (which vary from Host to Host) are
handled in the left hand portion. Power for the standard unit
is directly connected to the IMP's power — i.e., its power is
turned on whenever IMP power is turned on. Power for the
special unit is derived from the Host power system (or a separate
supply) and will probably have a separate on/off switch.

Each participating Host will be responsible for the design and
building of its own special unit that will mate to the standard

unit according to fixed rules. In general, this special unit
serves to serialize and deserialize information in whatever
manner best suits the particular Host. The IMP-to-Host section
of the special unit must perform the "padding with zeros" func-
tion discussed earlier.

Two levels of hardware handshaking take place between a Host and
its IMP. At the meta-level, each needs to know whether the
other is turned on and operational. The standard unit provides
to the special unit (and it in turn to the Host in whatever way
is appropriate) a signal which indicates that IMP power is up
and that the IMP program has turned on a Ready indicator. The
special unit presents a similar Host ready signal to the stan-
dard unit, and thence to the IMP. Each unit automatically moni-
tors the readiness of the other, and if the other's readiness
state changes, the unit will notify its parent computer; in the
case of the IMP, by an interrupt. Thus, for example, should
the Host computer fail or drop power, the IMP will be inter-
rupted and can take appropriate action. Only when the Host
returns to Ready, which requires not only reinstating power but
also program turn on of the Host ready indicator in the special
unit, will communications with the Host be re-established.
Under normal operation, when either computer detects that the
other has become ready, it will prepare to receive information.
Thus, with both Host and IMP ready, each will be waiting for
the other to transmit. As soon as information is provided by
either one, it will flow across the Interface.

Thus, when the Host ready indicator comes on, the operational
IMP program prepares to receive from its Host by setting up a
pair of pointers used by the standard Host-to-IMP interface

channel of the DMC. These pointers delineate a packet-sized
buffer in the IMP memory. After they have been set, the IMP
program issues an ACCEPT* command to the interface. Thereafter,
when information becomes available from the Host, the standard
interface unit takes it in serially and forms it into 16 bit-
IMP words in an input buffer register. These words are stored
into successive locations of the IMP memory buffer until the
buffer area becomes full or until the message end is indicated
by the Host. When either of these happens, information flow
ceases and the IMP program is interrupted. In the case where
the Host message ends, the hardware appends a trailing "one"
followed by any "zeros" necessary to pad out a full 16-bit word.
The interrupt routine will normally reset the pointers to an-
other buffer location and restart the interface with a new
ACCEPT command. Serial transmission makes the standard unit
independent of Host word size, and requires only one data line
driver and receiver. The interface unit is designed to accept
bits from the Host at 1 MHz maximum rate (5 MHz circuits are
used). The Host, of course, can slow this rate by controlling
the flow of bits. Memory references in both computers will
slow the rate well below the maximum.

When the IMP has set up memory pointers and is ready to transmit
a packet into the Host, it starts the transmission via a GO
command. The first word is then loaded from the IMP memory into
the interface and the Host unit takes the bits serially. Each
time 16 bits have been taken in, a new word is fetched from the
IMP memory. When the buffer has been emptied, the program is

---

*Control commands to devices are delivered by execution of as-
 signed OCP instructions. These instructions deliver appro-
 priate control signals.

interrupted and normally prepares for the next transmission to the Host if any more buffers are waiting. When the IMP is ready to transmit the last packet of a message, it executes a special END command before starting the transmission with the GO. In this case, when the last bit of the packet is taken into the special Host unit, an end-of-message signal is also sent to the unit. This causes the special Host unit to pad the remaining bits of its final word with zeros before passing it to the Host with the "that's all" indication.

2. The IMP/MODEM interface unit

Each IMP connects to several (up to 6) telephone line modems each of which has a separate IMP/MODEM Interface unit. This unit converts outgoing information into serial form and assembles incoming serial information into 16-bit words which it places in the IMP memory. It also computes 24 parity check bits, which it transmits at the end of a packet and checks upon receiving a packet. As shown in Fig. 12, a modem consists of two logi-cal halves, each producing clock signals and containing a single data line, one in and one out. The interface unit correspond-ingly contains two logically distinct sections, one dedicated to transferring output from the IMP to the modem and the other dedicated to transferring in the other direction. In the ab-sence of outgoing messages, the output section sends a continu-ous stream of SYN characters to the modem. Fig. 13 shows a typical packet buffer in the IMP memory from both the output and input points of view. In this presentation, only those elements of particular concern to the hardware are separated out. Thus header and text are not distinguished.

IMP/MODEM
INTERFACE

PHONE → MODEM → DATA → ◇ INPUT SECTION

LINES ← MODEM ← CLOCK →

CLOCK → OUTPUT SECTION

DATA
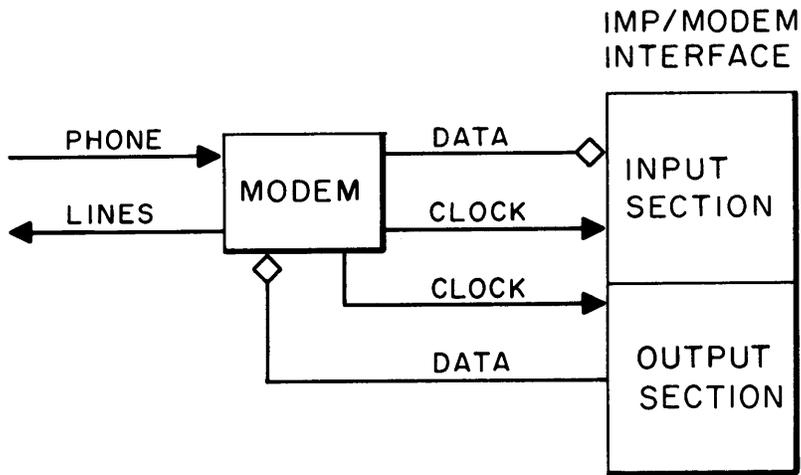
FIG. 12   LOGIC VIEW OF MODEM.

OUTPUT		INPUT		COMPOSITE	

SYN   SYN ◄── MANUFACTURED
DLE   STX ◄── BY OUTPUT
              HARDWARE

FROM IMP
MEMORY

SYN   SYN
DLE   STX ──► WAKEUP
              INPUT
              MECHANISM

STORED
INTO IMP
MEMORY

SYN   SYN
DLE   STX

SETTING OF   SETTING OF
POINTERS   POINTERS
FOR OUTPUT   FOR INPUT

DLE   HARDWARE PUTS
ETX   LAST MEMORY
CC1   WORD ON LINE AND
CC2   THEN ADDS DLE, ETX,
CC3   CC1, CC2, CC3
SYN   AND AT LEAST ONE
SYN   PAIR OF SYN SYN
             CHARACTERS

DLE   DLE ETX
ETX   IDENTIFIES
CC1   END OF
CC2   PACKET
CC3
SYN
SYN

DLE
ETX
CC1
CC2
CC3
SYN
SYN

FIG. 13   PACKET BUFFER FORMAT.

After setting the output pointers, as shown, the IMP program
notifies the output hardware that a packet is ready to be trans-
mitted. The hardware then sends the character pair DLE STX and
follows this with the data words taken from the IMP memory ac-
cording to the pointers. When the DMC indicates that the entire
packet has been sent, the hardware appends the character pair
DLE ETX followed by the check digits and at least one pair of
SYN characters. A string of SYN characters then follows until
another transmission is initiated.

Additionally, the hardware monitors the data from memory for
DLE characters and, upon finding one, immediately inserts an-
other character, thus averting confusion resulting from a DLE
within the packet. The receiving input unit deletes these extra
DLEs. Of course, extra DLEs are not inserted with the hardware-
generated DLEs.

The input hardware detects the DLE STX, which marks beginning of
a message and loads into the IMP memory all characters between
(but not including) the STX and the DLE of the final DLE ETX
character pair. The three check digits which follow the DLE ETX
are never brought into memory. Any error indicated by the
parity check is signaled to the computer. Note that the STX
is not itself fed into memory but serves only to cue the input
hardware to the start of the packet on the line. The bottom
input pointer points to *one location beyond* the point where the
last data word of a maximum-sized legal packet would be put.
Normally, the input hardware recognizes the end of input by
spotting the DLE ETX at the end of the packet. To assure that,
if it misses this, input does not proceed to flood the IMP
memory, input is cut off if the allocated IMP buffer fills up —

i.e., if *one more* than the expected maximum number of words ar-
rives in a packet. An error is indicated to the IMP program in
this case. Since the receiving input unit recognizes when a
packet begins and ends by the DLE STX and DLE ETX characters
enclosing the packet, there is no possibility of confusing the
start or end of a message since DLE STX or DLE ETX character
pairs can never occur *within* a message without being preceded
by another DLE. The receiving input unit deletes the extra DLE's.

J. Organization of IMP Storage

Message packets are read into buffers in IMP storage as we have
already discussed. Each incoming packet is allocated one free
buffer selected from a free buffer pool. Pointers are set by
the CPU to the beginning and end of the buffer and an input
transfer is enabled. When a packet is read into memory, an inter-
rupt signals the program upon completion of the transfer. If
an error is detected, the buffer is returned to the free buffer
pool. The packet, in effect, is discarded, since the buffer is
now free to be overwritten. Otherwise, the packet is assumed
to be correct.

*Within an IMP, a packet is never moved from one buffer to an-
other.* It is read into one location in memory with a set of in-
put pointers and taken out of the same location with a set of
output pointers.

Approximately six thousand words of memory will be occupied by
programs and the remainder will be available for buffers and
program expansion. Each of the buffers contains about 70 words.

One of these is a free word allocated at the end of the buffer
to detect the case where the buffer is about to be overflowed,
due to the loss of the end of message indication. An interrupt
will be generated during input if the moving pointer ever co-
incides with a pointer to this last cell. Approximately two
additional words at the beginning of each buffer are used for
holding queue pointers as discussed below.

We distinguish between three types of packets in the IMP which
we call store and forward packets, packets for the Host and
packets for the IMP. A store and forward packet is one whose
destination is another site. A packet for the IMP, defined
implicitly, is handled by special IMP routines and does not re-
quire lengthy storage since the buffer is quickly released back
into the free buffer pool.

The Host computer generates only store and forward packets or
packets for its IMP. Packets that arrive over the communication
lines may be either store and forward packets, packets for the
Host, or packets for the IMP.

A packet for the Host computer may be a single packet message
or part of a multiple packet message. Single packet messages,
which are uniquely identified by the last-packet-in-message bit
on packet number one, clearly require no reassembly and may be
directly transmitted to the Host computer. When the first
packet is received for a multiple packet message, seven addi-
tional buffers are removed from the free buffer pool and re-
served. As each additional packet of this message arrives and
is stored in a free buffer, one of the reserved buffers is re-
leased into the free buffer pool. When all packets of the

message have been reassembled, the remaining unused reserved
buffers are released and the complete message is sent to the
Host. Waiting until the full message is assembled avoids the
risk of typing up the channel to the Host in the middle of a
message. The storage for these packets is called reassembly
storage.

Each communication line has a buffer assigned to it which is un-
assigned upon receipt of an incoming error-checked packet, where-
upon another buffer from the free buffer pool is assigned in its
place.

A correctly received store and forward packet is placed on a
queue for transmission over the first choice output communica-
tion line. An IMP with three communication lines has three
such queues, one assigned to each line. Packets on each of the
three queues are transmitted sequentially over the communication
lines. There is also a similar queue for reassembled messages
going to the Host.

We now discuss the maintenance of these queues. Upon arrival,
each store and forward packet is placed at the end of a first
choice queue which is determined from an entry in a routing
table. Each queue is linked in the forward direction and three
pointers into the queue are kept. These pointers locate the
current service position on the queue, the last entry into the
queue, and the position of the packet expected to be acknowledged
next. In addition, the last packet in the queue is linked to
the first packet, thus forming a circular queue. The last posi-
tion on each circular queue is defined to be the position just
behind the current service position.

There are certain packets which, upon arrival or generation, may
be placed at the head of a queue at the current service position
where they will be next in line for transmission. These may in-
clude all packets for IMPs and all short packets.

K. Buffer Congestion

We now discuss the subject of buffer congestion and the techniques
that we have introduced to deal with it. We indicate the prin-
ciple causes of buffer congestion, describe the kinds of diffi-
culties which are caused by it and develop a number of simple
strategies which either attempt to prevent buffer congestion
from occurring or ensure the recovery from it.

Certain Host computers will be primary receivers of network mes-
sages and their corresponding IMPs will have a substantial por-
tion of the buffer storage containing messages for the Host com-
puter. Other IMPs will function essentially in the store and
forward mode, containing significantly fewer messages for their
own Host computers than for other IMPs in the network. IMPs
such as these, which primarily store and forward messages, are
critical links in the network. When they become congested, they
affect the overall pattern of traffic flow.

An IMP is said to be congested whenever the contents of the free
buffer pool falls below a level equal to the number of communi-
cation lines. There are several different causes of buffer
congestion, the most serious of which is a malfunction. We
discuss the effects of a malfunction later in the chapter. How-
ever, congestion can also occur during normal operation of the

network due to transmission errors, line concentration, or re-
assembly.

Line errors may be expected to occur on the order of seconds
apart. At 50,000 bits per second and line bit error probability
of 10^{-5}, one error is expected every two seconds. However, the
errors will undoubtedly be clustered so that the interval be-
tween error bursts will probably be over 10 seconds on the aver-
age. An IMP stores packets from the time they arrive until an
acknowledgment is returned. Sufficient storage has been allo-
cated to handle the reasonable peak loads of offered traffic
and to allow for line errors.

Line concentration refers to the situation when messages arrive
on several different communication lines and are intended for
transmission over the same outgoing channel. Since a packet
must be transmitted contiguously in time over a communication
line, two packets cannot be simultaneously transmitted and there-
fore at least one of the packets must wait.

Buffer congestion may also occur if insufficient reassembly stor-
age is available. For example, if 10 network users are logged
into one system, all messages have 8 packets, and a buffer is 70
16-bit words, then 5K of core would be needed for reassembly
alone, with all users simultaneously being reassembled. We may
expect to be confronted from time to time with the situation
where the IMP simply does not have enough buffers to do reassem-
bly. Furthermore, if a Host computer does go down or if messages
are fed to it over many links, the backup of packets into the
rest of the network could cause the entire network to overload.
The process of automatic rerouting which takes place when

messages fail to get through on a primary route (as discussed in
the following section) will tend to alleviate this situation.

In Section E (above) we already discussed the use of RFNM's for
averting congestion. We now discuss several more techniques de-
signed for coping with buffer congestion. To prevent buffer
congestion from affecting reassembly, we lock in (i.e., reserve)
seven more buffers for reassembly at the destination IMP when
the first packet of a message arrives. A reassembly packet is
accepted only if the addition of the seven additional buffers
will not trespass on the 25% minimum store and forward buffer
space. Buffer storage is conceptually divided into two sections,
one to hold messages to and from the Host computer and the other
used for store and forward packets. There is no fixed allocation
of buffers into one category or the other. The amount of stor-
age allocated to each is adjusted to meet the network demands.
However, some fixed minimum percentage of the total number of
buffers is always reserved for store and forward traffic. That
is, an IMP is never allowed to block network traffic by assign-
ing all its buffers for reassembly packets and outgoing messages
from its Host. The minimum number of buffers that must always
be available to the rest of the network for store and forward
packets is an IMP program parameter. Initially, we will dedi-
cate at least one quarter of the IMP buffers for such store and
forward packets.

L. Line Quality Determination and Rerouting

We define the *quality* (Q) of a line as the time varying relation
of received acknowledgments of a line to the total number of

packets requiring acknowledgment transmitted over the line. Thus,
the quality is a simple and direct measure of transmission suc-
cess on the line. The quality of a broken line will rapidly drop
to a very low value. Similarly, the quality of a line to a con-
gested IMP which does not regularly acknowledge packets will also
drop. This quality factor is used in two ways: to detect dif-
ficulties with the functioning of a line for statistics gather-
ing and trouble reporting, and *as a criterion for rerouting*. In
addition to the line quality, there is an *a priori* weighting of
the lines that reflects the desirability of using each line to
reach a given destination. This weighting is designated by the
letter K. The determination of K for each line to each destina-
tion is a complex judgmental matter, reflecting not only the
topology of the net but also knowledge, as it is gained, about
known average traffic patterns. Such information comes from
human analysis of network performance. The values of K are thus
selected in advance, loaded into the IMP as required, and kept in
a routing table.

Unless a line is disabled, when a packet first arrives in an IMP,
ready to be sent to some other IMP, the packet is placed on a
queue for the line with largest value of K. The line quality is
thus not normally used in the initial transmission, thereby
guaranteeing that lines are tried frequently in order to maintain
an up-to-date estimate of Q. Of course, routing for *retrans-*
mission is based on both the line quality and the K factor.

Regular checks are made on the status of all entries in the
queues as part of a time out procedure, in order to consider the
possiblity of retransmission. The algorithm which selects
packets for retransmission works as follows: Each buffer on a

queue has a "sent" bit which is set to one when the contents of
the buffer have been transmitted. The bit is reset to zero if
the buffer is to be retransmitted. During each time out proce-
dure, a check is made to determine if a time out has occurred
since the packet was last transmitted. If the packet was trans-
mitted but has not timed out, the sent bit is left on. If the
packet has timed out, a calculation is made to determine the
most desirable route and the packet is routed accordingly. The
calculation will be a simple function of the line quality and
the preassigned weighting of the line.

We have not attempted to specify the alternate routine algorithm
in greater detail at this time for two primary reasons. First,
any reasonable algorithm will perform acceptably in the initial
net since the connectivity is so limited. Secondly, we did not
want to include as part of our proposed design, an *ad hoc* solu-
tion to a problem upon which the network performance will be
critically dependent under heavy load. We plan to provide an
algorithm which is adaptive, free from recurring loops, and re-
flects our best judgment on this matter.

We have designed and operated a network simulation program on our
940 computer. The program drives a CRT display that may be used
to assist in the testing and simulation of various algorithms.
This simulation will be a valuable instrument in studying im-
proved routine algorithms. The algorithms can then be tested
by actual network experimentation.

M. Network Introspection

As the network operates to service Hosts, it must monitor its
own performance to detect faults, take corrective actions as re-
quired, and report on its own activity to various points in the
network. The reporting function includes urgent messages about
malfunctions, prompt comments about changing conditions, and
more leisurely periodic summaries of statistical performance.
In order to permit such monitoring, fault recovery, and report-
ing by the program, adequate "test points" must be built into
the hardware and the operational software. In addition, decisions
must be made as to where reports of various types should be sent:
reports might go to a local Host, or to a "special" IMP run by
the network contractor, or to ARPA, or to a particular special
Host, or to some combination of these places. We do not feel
that the choice of destinations is a crucial issue at this time,
and for purposes of discussion we have assumed the existance of
a "network measurement center" (NMC). This NMC is presumed to
be a particular interested Host.

In the remainder of this section, we first discuss detection,
reporting, and recovery from three kinds of faults, namely, Host
faults, line faults and IMP faults. We then discuss the tech-
niques to be used for gathering detailed information about net-
work performance, and the reporting of that performance; finally
we summarize the kinds of abnormal messages which will be gen-
erated in these processes.

1. Faults

1.1 Host Faults

If a Host actually goes off the air, either voluntarily or through
a traumatic failure such as loss of power, a special Host ready
indicator which resides in the IMP/Host Interface will be turned
off. Any change of state of this indicator produces an interrupt
of the IMP; thus, the IMP program may note the change and take
action. If the shutdown was voluntary, the IMP may have been
notified previously and therefore suitably modified its tables.
If no prior notification has been received, the IMP informs the
current remote users. A message saying "My Host is down" will be
sent to users who try to login at unavailable Hosts. The normal
result of a traumatic Host failure is not only the immediate
quenching of additional messages from the sources, but a dis-
carding of all packets in the net addressed to that Host upon
their arrival at the destination IMP. When a Host comes back
up after a down period, the ready status will change to on and
the IMP will note this change. Test messages may also be used
in this case to confirm proper operation of the channel to the
Host.

A more difficult case occurs when the Host fails in some way
which does not change its ready status, but which nonetheless
destroys its ability to interact with the network. Such fail-
ures, for example, may be caused by software bugs, or minor
hardware transients, which can cause programs to loop. In order
for the IMP to detect such a situation, it will keep an indicator
of the quality of communication with the Host. If normal IMP-
Host message flow is greatly diminished for some comparatively

long time, the IMP will assume that the Host is down and will
take the same action as if the ready indicator had been turned
off. To determine when the Host is again available involves the
use of test messages from the IMP to the Host. The outage of
the Host, even for extended periods, does not in any way affect
the IMPs role in storing and forwarding other network messages.

1.2 *Line Failures*

The normal operational IMP program maintains up-to-date indica-
tions of the quality of every incoming and outgoing line. If
the estimate of quality on a given line falls below a preset
clip level (a program parameter), the IMP will inform local per-
sonnel by changing lights in the lights register, and will in-
form the NMC by producing a trouble report. This provides a
relatively straightforward and positive procedure for keeping
track of line troubles.

Checks of the lines will also be done during initialization of
the IMP program, and also during scheduled and unscheduled
maintenance of the line. A special IMP program will be able to
cross patch each line under program control and test the Modem
and Interfaces of each line. It is conceivable that such cross-
patch testing could be built into the operational program at a
later stage in the development of the network, but we do not
plan to include it initially.

1.3 *IMP Faults*

Despite the extreme provisions for reliability built into the
IMPs, faults will sometimes occur. Detection of these faults is

necessary to ensure smooth operation of the network. In some
cases (such as total failure), an IMP will be unable to detect
trouble itself. Provision must be made for neighboring IMPs
(which do detect such failure) to report this. Communication
outside the network channel (e.g., by phone) will then be used
to inform personnel at the site of the IMP of that IMP's mal-
function.

On the other hand, the majority of IMP failures should be able
to be detected at the IMP itself by making the operating program
periodically reset a timing device. Failure to reset the timer
before it times out will set a failure indicator.

This internal failure detector can communicate the failure to
the failed IMP or to a maintenance person without resort to ex-
ternal communication. For this reason, we have included an
internal failure detector utilizing a time-out period.

Having detected failure, there are several methods for imple-
mentating a restart. Certainly the simplest to implement at the
outset is to arouse the Host operator with an alarm and allow
him to load the system via the paper tape reader following the
same *simple* procedure employed in start-up of new program ver-
sions. As the system evolves, automatic restart procedures
could reduce the outage time caused by transient failure. Ideal-
ly, the IMP could restart automatically from an auxillary stor-
age device capable of multiple restarts. Alternatively, one
could restart by automatically reloading the IMP from its Host.
(We do not favor involving the Host with this task.) Still an-
other alternative is to reload one IMP from another by causing
a loader to be put into operation in the failed IMP. This IMP,

in turn, requests and checks the reloading of the operational
program from a neighboring IMP.

We would tend to order these automatic restart alternatives on
the basis of IMP autonomy and simplicity, and would thus tend
to favor first an auxillary storage device, followed by restart
from a neighboring IMP and, lastly, restart from the Host. The
actual choice and implementation of automatic restart should be
the subject of further study and experiment in the 4 node net-
work. Initially, the IMPs should be restarted manually with
paper tape following a hardware alarm. The 4 node IMP equipment
will support experimental investigation of alternative automatic
restart methods; the IMP will have a limited amount of protected
memory and a suitable timer for this purpose.

An IMP which fails may be a critical node which cuts off some
existing links. For example, a destination IMP failure cuts off
all links to its Host. The network must respond appropriately
to such an outage. All links through the IMP will quickly be
blocked since no RFNM messages will get back to the sources.
Packts trying to get through a down IMP will circulate in the
system, trying to circumvent it. When the IMP comes back on
the air, the messages will eventually reach the destination and
be discarded.

Should an IMP be down for an extended period, some sort of mech-
anism is required to purge the system of undeliverable packets.
We have not settled on a particular technique but have considered
two possibilities. The first of these is to include in each
packet a handover number that would increase on every IMP-to-
IMP transfer and that would allow a discard of the packet when

a (high) clip level is reached. An alternate approach is to have a
Host generate special messages for this purpose.

2. Performance measurements

We propose two main techniques for gathering performance infor-
mation on the operation of the network: (1) Regular measurement
by each IMP of its internal performance; and transmission of
that information on a periodic basis to the NMC and (2) the trac-
ing of messages through the system, resulting in the generation
of report packets about that message proceeding to the NMC for
reconstruction of the message path.

a) Regular Data Gathering

Each IMP will include in its operational program a routine that
will be run on a clock interrupt. Thus the program will run
periodically independent of the load on the IMP at that time.
This program will sample some program parameters and either save
the values or running averages of these values. The following
list provides examples:

    1. Empty buffer count

    2. Number of messages being reassembled

    3. Queue length of output queues

    4. Number of sent but not acknowledged buffers in each queue

    5. Quality measures

    6. Rate of inputs

The list of sample parameters will then be included in a special
report message directed to the NMC. We believe that this regular
technique of reporting will provide a comprehensive history of
what the IMPs are doing. It naturally assumes some attention on
the part of the NMC, but obviously remains a matter of choice.

b) Tracing

The other data gathering facility, which we believe will be ex-
ceptionally useful, we call tracing. A common notion in computer
programming, tracing allows one to obtain either a small amount
of information or a large amount of information as the trace
proceeds. We believe that our network trace feature has the
same extremely desirable flexibility.

Any or all messages may include a trace bit in the header. Mes-
sages with trace bits may be initiated by the NMC or by other
Hosts. For example, trace bits could be put in some set frac-
tion of each Host's messages. In fact, we can think of a number
of techniques whereby trace bits could be added to messages on
a sample basis. To give one more example, each IMP could be
asked to include a trace bit in every mth IMP message. We be-
lieve this technique will permit occasional sampling or complete
tracing of messages in the network.

When an IMP receives a message that includes a trace bit, it in-
curs the additional task of noting in detail how it handles that
particular message. When the IMP has finally released that
message, it must generate the special report about that message
and send the report to the NMC. The NMC will thus receive a

sequence of report messages for each message that contains a
trace bit. It should then be possible for the NMC to generate
a good representation of the path taken by that message, or by
a group of messages in the network.

3.   Summary of abnormal messages

Results of the introspection discussed above are transmitted by
"abnormal" messages that are generated by IMPs for these special
purposes; these abnormal messages are not part of the normal
flow of data between Hosts. We believe that there will be a
large number of packets of this type, but it is impossible to
list them now with any confidence. However, we can distinguish
between several kinds of packets, and provide an initial esti-
mate of what types might exist.

We group the class of special packets into three categories.
The first category contains those packets which only cross
IMP/MODEM Interfaces and contain all IMP-to-IMP messages. The
second category contains those messages which only cross an
IMP/HOST Interface. The third category defines messages which
cross one HOST/IMP Interface and one or more IMP/MODEM Inter-
faces. (If two HOST/IMP Interfaces are crossed, the message is
a Host-to-Host message and considered to be part of the Host
protocol.)

We list some of the special messages in each of these three
categories:

    1.   Across IMP/MODEM INTERFACES
        a.   Query
        b.   Response

        c.   IMP going down

        d.   IMP back up

        e.   Acknowledgment

        f.   Ready for next message

        g.   My Host is down

  2. ACROSS IMP/HOST INTERFACES

        a.   Query

        b.   Response

        c.   I am going down

        d.   Ready for next message

3A. IMP TO REMOTE HOST

        a.   Fault detected

        b.   Report generation

 B. HOST TO REMOTE IMP

        a.   Change routing table

The above list contains some entries such as "My Host is down."
In connection with messages such as these, we wish to here intro-
duce the notion of busy signals. In making a telephone call,
there is no indication, at the telephone and before the call
is tried, that a line will be busy, out of order, or not an-
swered. We feel that this is a powerful concept as applied to
the network. For example, when an actual user at a Host site
tries to use the network to call some other Host, *at that time*
the network should try the call and then send back a message,
finally reaching that user, which says, "Sorry, the Host you
just tried to call is down." This arrangement has the advantage
that as a given Host goes up and down it is not necessary for
large numbers of control messages to flow around the network.
To keep everyone informed of the instantaneous status of that

Host. Instead the status is made available "on request." This
approach can be applied to many situations within the network,
and we propose to apply it where possible. Naturally some
status information will, in fact, be kept distributed, but we
will try to minimize the number of different kinds of status
tables that must be kept up-to-date.

N. The Operational IMP Program

Inasmuch as the operational program implements the strategy and
protocol of the network, some discussion of general philosophy
and its significant features is in order.

Because of the experimental nature, the diffuse geography and
the multiplicity of Host types of the network, it is essential
that the program be simple and crisp. The program should be
divisible into clearly defined functional units with as few
interconnecting pathways as possible. This approach will greatly
simplify the debugging of the software. Since the network will
evolve as we learn more about networks and their uses and con-
straints, the program must be designed to allow for changes
and modifications.

To cope with a wide range of real-time data rates, particular
attention must be paid to timing requirements. In addition,
since much of the IMP memory is given over to buffer storage
(both to and from the local Host and for store and forward),
the program must be as compact as possible. Of the 12K of
memory, we expect the program will eventually occupy approxi-
mately one-half to two-thirds. The network software is out-
lined in this section.

We feel that the only sensible language in which to write the
IMP software is DDP-516 assembly language. This will enable the
IMP programs to be as compact and efficient as possible, which
is something a higher level language typically subverts. Opti-
mum efficiency is essential here; when a program must deal with
low level hardware considerations in real time, a high level
language becomes more of a nuisance than a convenience. Al-
though a high level language makes programs more readable and
easier to debug, we do not feel we can afford the luxury.

Figure 14 is a schematic diagram outlining the control logic
of the operational program. It has five basic pieces: an
initialization routine, interrupt routines, task routines,
shared subroutines, and background routines. The program is
started at the initialization routine, which first goes through
a machine and interface checking routine. It then sets up in-
puts for all input channels (from Host and phone line Modems)
such that, when an input is complete, an interrupt will occur.
It also enables the clock interrupt and does all other initiali-
zation that is necessary and then turns control over to the
background loop.

The routines of the background loop are cycled through repeatedly
until an interrupt switches control to some other routine. When
all interruptions have been serviced, control is returned to the
instruction in the background routine which was about to be exe-
cuted when the first interrupt occurred.

START ⟶ INITIALIZATION ⟶ BACKGROUND-LOOP

(Pulse from clock
every timeout period)

Interrupt routines {

INPUT-FROM-NETWORK ✱ ●

OUTPUT-TO-NETWORK ✱ ●

INPUT-FROM-HOST ✱ ●

OUTPUT-TO-HOST ✱ ●

CLOCK-INTERRUPT ✱ ●

These routines
lock out further
interrupts in
their category
and call
ENTER-TASK

Task routines
corresponding to
interrupt routines {

NETWORK-INPUT ○ ✗

NETWORK-OUTPUT ○ ✗

HOST-INPUT ○ ✗

HOST-OUTPUT ○ ✗

TIMEOUT ○
Line quality and
rerouting
calculation
✗

Shared subroutines
(interrupts must be
locked when these
are called) {

ENTER-TASK ○
If task list is
empty, initiate
task interrupt

Enter task at
end of task list
✗

GET-EMPTY-BUFFER ○
If there are no
free buffers,
execute last task
on task list until
buffer is found
✗

RELEASE-BUFFER ○
✗

TASK-INTERRUPT ✱
Execute tasks in
order off task list

●

(This interrupt
routine may be
interrupted by
any other)

✱ Interrupt entrance
● Return from interrupt
○ Subroutine call
✗ Return from subroutine

FIG. 14   IMP PROGRAM CONTROL LOGIC

When an interrupt occurs, a call to the routine associated with
that interrupt is executed. This call saves the point of inter-
ruption so that control can later be returned to the proper
place. The interrupt routine also saves the state of the ma-
chine for restoration upon return. An example of an interrupt
condition is the completion of the input of a packet from a
neighboring IMP. The input hardware calls the interrupt routine,
which sets up another input, rearms the interrupt line and
designates the received packet for subsequent processing. The
input interrupt routines are indicated just below the initiali-
zation routine in the diagram. These interrupt routines prohibit
calls *of themselves* while they are running by locking out fur-
ther interrupts of the same kind upon entry to the routines.*
Consequently, these routines must be very fast so that inter-
rupts can be re-enabled quickly and not be missed. Most of the
time-consuming work is taken out of the interrupt routines by
having them merely stack calls to other routines (called task
routines) on a task queue which will be executed in what is,
in some sense, high priority background time. This allows some
time buffering of packet handling if the handling routines take
more than real time for a short period.

The question arises as to how the tasks contained in the task
queue are ever processed, since the interrupt routines return
control to another interrupt routine (if interruption occurred
there) or to the background routines when all interrupts have
been serviced. This is done as follows: each time a task is
entered onto the task list, a check is made to see whether there

---

*The DDP-516 provides for this with a convenient interrupt se-
lection mask and enable scheme.

are any previous tasks on the queue. If not, a special hardware
feature is used for a program-initiated interrupt (called the
"task interrupt"), which is set so that, when the "normal" in-
terrupt routine returns to the background loop and re-enables
interrupts, the "task interrupt" will take control and allow
entries to be processed in the task queue. (The ENTER-TASK and
TASK-INTERRUPT routines are shown in the bottom left and the
bottom right of Fig. 14.) When the task list is empty, con-
trol is returned to the point of interruption in the background
loop. The interrupt routine which executes tasks can be inter-
rupted by any other interrupt routine but will never interrupt
itself. Because calls of the task routines are executed se-
quentially, there is no need to make the task routines re-
entrant and indeed this is the fundamental reason for queueing
tasks. Appendix F includes an example of the use of task and
interrupt routines.

There remains a set of routines called the shared subroutines.
These are the routines that make entries on the task list, the
routines that handle empty buffers, etc. Other interrupts which
may call these routines are locked out when these routines are
called.

In summary, then, there are really three levels of priority,
each corresponding to programs which perform a particular type
of function:

1) interrupt routines that interrupt task routines and back-
    ground routines and even some other interrupt routines;

2) task routines which (in some sense) interrupt the back-
    ground routines; and

3) background routines.

The interrupt routines for the interfaces are activiated as buf-
fers fill, or are emptied. In general these routines reset
pointers, make entries in the task queue for handling filled buf-
fers and releasing emptied ones, and reactivate the interface
in question. The clock interrupt routine indexes a higher order
clock counter which is maintained in core memory and adds to the
task list the task that tests for packet time out. Some of the
task routines are: allocating and reclaiming empty buffer stor-
age; handling short buffers with high priority; timing out for
IMP-to-IMP acknowledgments and retransmitting (when appropriate);
processing end-to-end Requests-For-Next-Message; locating the
next buffer to send; identifying incoming messages and placing
them on the proper queue for transmittal either to the Host or
into the proper output line; transmitting IMP-to-IMP acknowledg-
ments; reassembling messages for the local Host and transmitting
Requests-For-Next-Message after reassembly is complete; breaking
off destination information from the top of messages from the
local Host and fabricating and attaching link identification;
and other header information to outgoing packets of a message.

The concept of three priority levels, and the availability of
the background loop permits the IMP to perform much more exten-
sive computations on an occasional basis. This is particularly
important if the need arises for word-rate jobs on occasional
packets. If Host-peculiar programs are required for ASCII con-
version, or for other data transformation tasks, such jobs may
be accomplished without disrupting the tight timing of the in-
terrupt routines or the task queue. Background programs also
include such jobs as transmitting and checking received network
test messages and miscellaneous statistics gathering.

1. Summary of IMP program routines

Initialization

>   Checks hardware of machine and interfaces, sets
>   up initial inputs, enables interrupts, and does
>   other necessary initialization.

Background loop

>   Set of routines executed cyclicly, in order when
>   not interrupted.

Execute task

>   Executes entries on task list in order.

Input from network

>   Answers interrupt, sets up new input from
>   network line, and enters task on task list.

Output to network

>   Answers interrupt and enters task on task list.

Input from host

>   Answers interrupt and enters task on task list.

Output to host

>   Answers interrupt and enters task on task list.

Timeout

>   Answers interrupt and enters task on task list.

Interrupt
Routines
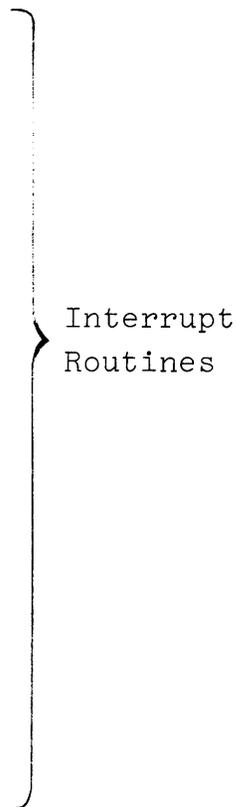
Input from network

> Puts acknowledgment on output queue and
> dispatches* to the input processing routines.

Output from network

> Finds next unused buffer, marks it sent and
> sets up output.

Input from Host

> Appends header to buffer, etc., puts buffer on
> output queue, and sets up new input.

Output to Host

> Sets up output to next buffer to Host.

Timeout

> Searches output queues for any unacknowledged
> buffers and reroutes them.

Task
Routines

Enter task

> If task list is empty, initiates program inter-
> rupt and enters task on task list.

Get empty buffer

> Calls Execute task if no empty buffers remain
> and returns buffer.

Return empty buffer

Rerouting

Shared
Subroutines

---

*These routines do most of the work of IMP program.

We feel that the program structure just described meets the goals
discussed earlier. The program is constructed of functional
modules that are logically independent, thus giving them a sim-
plicity that will make their coding, debugging, and understanding
easy. Such modularity also enables natural and easy addition and
deletion of functional modules.

Recursion (i.e., reentrancy), which is costly in time, is elimi-
nated through use of the task list that also provides a single
consistant manner of calling and passing arguments to subroutines.
Speed is also attained by moving pointers rather than buffers and
by keeping buffers on doubly linked lists for easy insertion and
deletion from queues.

While the proposed program structure does not waste space, it is
not designed to be as short as possible. We feel it is not worth
the additional complexity that results from routines which share
short pieces of common code, especially since the routines run on
interrupts and interrupt each other. Of course within a routine
we will use all of the cleverness at our disposal.

2. Timing and space considerations

In this section we estimate the running time of the crucial rou-
tines of the IMP program, review the consequences of these times,
and estimate the storage requirement of the IMP program.

A study of the various IMP program routines yields our timing
estimates. We first consider in detail the running time of the
INPUT-FROM-NETWORK interrupt routine (we actually coded sample
routines).

The coding requires 40 instructions with an average time of 2.5 μs/instruction. We next estimate quite closely the running time of the NETWORK-INPUT task routine, including the STORE-AND-FORWARD input processing routine which we feel approximates an average path through the NETWORK-INPUT task routine. This we also estimate to be 40 instructions.

We also estimate that the OUTPUT-TO-NETWORK interrupt routine and the NETWORK-OUTPUT routine will each take about 20 instructions.

The time required to handle the Host is under the IMP's control and is also down by a factor of four from the time required to handle the four modem lines and may thus be temporarily discounted; rerouting happens rarely, as it is clocked.

Thus, the bulk of the work may be tabulated:

               40 instructions — INPUT-FROM-NETWORK
               40 instructions — NETWORK-INPUT
               20 instructions — OUTPUT-TO-NETWORK
               20 instructions — NETWORK-OUTPUT

Since the number of instructions required to pass a packet into an IMP is 80 and the number of instructions required to pass a packet out is 40, we take the average number to handle a packet to be 60 instructions. Adding a factor of one-half to take into account things we have forgotten (overheads of various types, Host routines, and a share of the rerouting time for each packet), we arrive at an estimate of ninety instructions required, on the average, to pass a packet across an IMP boundary.

Using these numbers, Appendix A draws the following conclusions:
assuming the RFQ model (i.e., 4 links, 15Kb lines, 344 bit packets,
etc.), 14% of the machine time is used. Assuming the RFQ model,
but with all 50Kb lines, 43% of the machine time is used.

We finally estimate, based on experience rather than actual coding,
that the storage necessary for the main IMP program outlined in
this section — the program which does the hard, fast, "necessary"
work — will fit in 2000 words of DDP-516 storage. The remainder
of the program (the background routines, the special IMP-TO-HOST
message routines, etc.) is much less well defined but we estimate
that it will occupy somewhere around 4000 words. This leaves
about 6000 words for buffers and program expansion.

3. Test programs

Typically, many of these programs are short and simply pump test
patterns through the interfaces for observation on an oscilloscope.
Programs for loop and inter-computer tests in general will not in-
volve complex error analysis although they will include error de-
tection. The more sophisticated test programs transmit and receive
(in loop or inter-computer configuration) random patterns, checking
for identity upon receipt. No program means exists for generating
errors in the cyclic check mechanism of the hardware, but failure
can be introduced by temporarily disabling check character genera-
tion in the sending hardware.

4. Utility programs

The DDP-516 comes with an assembler, a primitive editor, a program
loader and an octal debugger. Assembly of programs will be done

at the test facility on a 516 which will have a high-speed punch.
Programs will be composed and edited on BBN's PDP-ld computer
under time-sharing and will be punched in ASCII for the 516
assembler. This requires the construction of no additional
sophisticated utility programs, allows multiple users access
to program composition facilities, and causes no disturbance
of the standard DDP-516 assembly and debugging system.

O. Optional Site Arrangements

We have given some consideration to three special sorts of site
installations: one with two hosts to be served, one in which
the IMP acts as a terminal controller, and one in which the IMP
services the Host as a data concentrator. For the site with
two hosts, two IMP/HOST hardware interfaces will be required.
While the standard interface is modular in nature and two such
interfaces can be installed in an IMP, this installation creates
a special situation. First of all, either additional priority
interrupts will be required or some of the normal priority in-
terrupt channels will have to be reassigned. In either case,
some special tailoring of the standard program will be required,
at the very least, to enable it to handle the interrupts properly.
The 16 channels of the DMC are sufficient to cover this case.
However, we feel that generalizing the standard program in such
a way as to make it directly suitable for either a one or two
host installation is not sensible: the additional required
sorting and routing is simply too expensive in terms of time
and space to warrant its inclusion in the standard version. On
the other hand, the program is amenable to modifications that will
enable it to handle the two host situation — but with some degra-
dation of performance.

If a proposed network node does not have a Host computer, it may
be useful to put into the IMP those functions of a Host computer
that allow users at Teletypes to converse with distant nodes.

To do this, one might first conceptually partition the IMP com-
puter into two parts — one for the IMP network program and one
for a program similar to the Host network program which each nor-
mal Host has. This partition is easy to make since both programs
will run asyncronously on interrupts. Additionally, a Teletype
scanner must be attached to the I/O channel for the pseudo-Host
network program. This program maintains an input and an output
buffer for each Teletype line and gathers characters for the
buffers as the scanner collects them. When a buffer is full, it
is passed to the IMP network program as a packet. The IMP pro-
gram which normally deals with the Host interface is now no longer
necessary.

This scheme subtracts from the time available for the IMP network
program to service store and forward packets. The method does
not detract from the space available for buffers, since the pseudo-
Host program replaces the IMP Host interface program,and the
pseudo-Host program shares buffer storage with the IMP network
program.

If there is a Host computer at a network node, it might be feasible
to use the IMP as a data concentrator for the Host. In this case,
the pseudo-Host program described above is still necessary; in-
stead of passing packets to the IMP network program, the program
passes them to the Host. The Host can arrange to process these
special packets as, for example, line-at-a-time Teletype input
to the standard Host operating system.

Once again, no timing problems occur since the separate IMP pro-
grams are run asynchronously on interrupts, but the additional
IMP program does subtract from the available space since the IMP/
Host interface program cannot be omitted.

We have not investigated these issues in any real detail. There
are many other possible, perhaps better, methods of simultaneously
using an IMP for a data concentrator or terminal.

APPENDIX A: TIMING COMPUTATIONS

A central computation in the design and evaluation of the network
is the determination of the actual amount of IMP processing time.
It affects the selection of the IMP computer, the performance and
utilization of the chosen computer, and forms a basis for
the model calculations.   It also strongly affects the de-
sign of the hardware interface and, in conjunction with the chosen
computer, forms a principal measure of the expansion capability of
the network.

However, this computation cannot be performed without making some
estimate of the traffic which an IMP is expected to handle. The
results which are obtained are extremely sensitive to the initial
assumptions. In this appendix we will discuss two sets of assump-
tions which we label as A and B.

Assumption A: This is the assumed traffic in the RFQ model. Each
                channel carries 15 kilobits/sec and the Host line
                carries 20 kilobits/sec. The average packet size
                on a channel is 344 bits and the average packet
                size on the Host line is 576 bits. There are four
                channels and one Host line.

Assumption B: This corresponds to a "reasonable" peak load con-
                dition and is identical to assumption A except
                that all channels as well as the Host line are
                assumed to carry 50 kilobits/sec.

We determine the total number of bits per second, R, and the aver-
age number of packets per second, P, that cross an IMP interface
in any direction.

$$A: \quad R = 8 \times 15{,}000 + 2 \times 20{,}000 = 160{,}000 \text{ bits/sec}$$

$$P = \frac{120{,}000}{344} + \frac{40{,}000}{576} = \sim 420 \text{ packets/sec;} \tag{1}$$

$$B: \quad R = 10 \times 50{,}000 = 500{,}000 \text{ bits/sec}$$

$$P = \frac{400{,}000}{344} + \frac{100{,}000}{576} = \sim 1325 \text{ packets/sec.} \tag{2}$$

There are two primary components to the calculation of the IMP processing time, namely the time required for I/O transfers and the time required for internal packet processing. We first consider the total cycle time, T_T, required to do input-output transfers.

We assume four cycles per I/O transfer (core counters are assumed instead of hardware counters for reasons of economy) and set

$$W = \text{Word length in bits}$$
$$C = \text{Cycle time in } \mu s$$
$$I = \text{Instruction time in } \mu s.$$

$$A: \quad T_T = \frac{160{,}000}{W} \times 4C \quad \mu s/sec; \tag{3}$$

$$B: \quad T_T = \frac{500{,}000}{W} \times 4C \quad \mu s/sec. \tag{4}$$

Within each IMP, the bulk of the processing is performed on a per packet basis. We have estimated the average number of instructions required in the IMP program to process these packets. There are four basic components of the processing .

INPUT INTERRUPT ROUTINE - 40 instructions $\left.\right\}$ 80 for
INPUT TASK PROCESSING   - 40 instructions input

OUTPUT INTERRUPT ROUTINE - 20 instructions $\left.\right\}$ 40 for
OUTPUT TASK PROCESSING   - 20 instructions output

We average these quantities to obtain a figure of 60 instructions/
packet in crossing an IMP boundary. We further estimate that all
additional tasks will average another 30 instructions/packet.
Therefore we use the figure of 90 instructions/packet as the aver-
age number of instructions which must be performed by the IMP
program to process each packet which crosses the IMP boundary.
Note that a packet which *traverses* the IMP is thus assigned a
total of 2 × 90 = 180 instructions.

The total program instruction time, T_I, is given by

$$A: \quad T_I = 420 \times 90I = {\sim}38,000I \ \mu s/sec; \tag{5}$$

$$B: \quad T_I = 1325 \times 90I = {\sim}120,000I \ \mu s/sec. \tag{6}$$

We now wish to estimate the individual instruction time, I, for
a small sized computer. It is reasonable to assume that in a
hypothetical 20 bit machine, indirect addressing should never be
required to access any word of memory (in a typical IMP config-
uration of less than 16K). We assume that such a 20 bit machine
requires an average of 2 cycles per instruction and that a ma-
chine with a shorter word length, W, will require approximately
2 × 20/W cycles/instruction due to an increasing frequency of
indirect addressing with decreasing word size.

Thus, we have the following expression for the instruction time

$$I = \frac{20}{W} \times 2C \ \mu s$$

and the total program instruction time, T_I, for handling packets is

$$A: \quad T_I = 38{,}000 \times \frac{20}{W} \times 2C = 1.52 \times 10^6 \ \frac{C}{W} \ \mu s/sec; \qquad (7)$$

$$B: \quad T_I = 120{,}000 \times \frac{20}{W} \times 2C = 4.8 \times 10^6 \ \frac{C}{W} \ \mu s/sec. \qquad (8)$$

On adding Eq. 3 to Eq. 7 and 4 to 8 we obtain an estimate, $T = T_T + T_I$, of the total cycle time required to handle the IMP traffic.

$$A: \quad T = 6.4 \times 10^5 \ \frac{C}{W} + 1.52 \times 10^6 \ \frac{C}{W} = 2.2 \times 10^6 \ \frac{C}{W} \ \mu s/sec; \qquad (9)$$

$$B: \quad T = 2 \times 10^6 \ \frac{C}{W} + 4.8 \times 10^6 \ \frac{C}{W} = 6.8 \times 10^6 \ \frac{C}{W} \ \mu s/sec. \qquad (10)$$

From this point we will simply assume that

$$C = 1$$
$$W = 16$$
$$I = \frac{20}{W} \times 2C = 2.5 \ ,$$

since these are the appropriate values for the DDP-516, and proceed with the computation of the timing and the model.

$$A: \quad T = 2.2 \times 10^6 \times \frac{1}{16} \cong 0.14 \times 10^6 \text{ } \mu\text{s/sec } \textit{or 14\% of capacity};$$

$$\text{(11)}$$

$$B: \quad T = 6.8 \times 10^6 \times \frac{1}{16} \cong 0.43 \times 10^6 \text{ } \mu\text{s/sec } \textit{or 43\% of capacity}.$$

$$\text{(12)}$$

Therefore, under assumption A, only 14% of the machine capacity is used, while at the "reasonable" peak loads of condition B approximately 43% of the machine capacity is used.

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author)	2a. REPORT SECURITY CLASSIFICATION
Bolt Beranek and Newman Inc 50 Moulton Street Cambridge, Massachusetts 02138	UNCLASSIFIED
	2b. GROUP

3. REPORT TITLE

INITIAL IMP DESIGN

4. DESCRIPTIVE NOTES *(Type of report and inclusive dates)*

5. AUTHOR(S) *(First name, middle initial, last name)*

Bolt Beranek and Newman Inc

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
January 1969	80	

8a. CONTRACT OR GRANT NO. DAHC 15-69-C-0179	9a. ORIGINATOR'S REPORT NUMBER(S)
b. PROJECT NO.	BBN Report No. 1763
c. (not yet known)	9b. OTHER REPORT NO(S) *(Any other numbers that may be assigned this report)*
d.	

10. DISTRIBUTION STATEMENT

Unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Advanced Research Projects Agency Washington, D.C. 20301

13. ABSTRACT

The basic function of the IMP computer network is to allow large existing time-shared (Host) computers with different system configurations to communicate with each other. Each IMP (Interface Message Processor) computer accepts messages for its Host from other Host computers and transmits messages from its Host to other Hosts. Since there will not always be a direct link between two Hosts that wish to communicate, individual IMPs will, from time to time, perform the function of transferring a message between Hosts that are not directly connected. This then leads to the two basic IMP configurations —— interfacing between Host computers and acting as a message switcher in the IMP network. The message switching is performed as a store and forward operation. Each IMP adapts its message routine to the condition of those portions of the IMP network to which it is connected. IMPs regularly measure network performance and report in special messages to the network measurement center. Provision of a tracing capability permits the net operation to be studied comprehensively. An automatic trouble reporting capability detects a variety of network difficulties and reports them to an interested Host. An IMP can throw away packets that it has received but not yet acknowledged, transmitting packets to other IMPs at its own discretion. Self-contained network operation is designed to protect and deliver messages from the source Host to the destination Host.

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Computers and Communication						
Store and Forward Communication						
ARPA Computer Network						
Honeywell DDP-516						
IMP						