

AD 686811

Report No. 1783

April 1969

INTERFACE MESSAGE PROCESSORS FOR  
THE ARPA COMPUTER NETWORK

QUARTERLY TECHNICAL REPORT No. 1  
2 January 1969 to 31 March 1969

Principal Investigator: Mr. Frank E. Heart  
Telephone (617) 491-1850, Ext. 470

Sponsored by  
Advanced Research Projects Agency  
ARPA Order No. 1260

Contract No. DAHC15-69-C-0179  
Effective Date: 2 January 1969  
Expiration Date: 28 February 1970  
Contract Amount: \$1,077,727.00

Submitted to:

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

DDC  
RECEIVED  
APR 30 1969  
B

Reproduced by the  
CLEARINGHOUSE  
for Federal Scientific & Technical  
Information Springfield Va. 22151

Report No. 1783

Bolt Beranek and Newman Inc

INTERFACE MESSAGE PROCESSORS FOR  
THE ARPA COMPUTER NETWORK

QUARTERLY TECHNICAL REPORT No. 1  
2 January 1969 to 31 March 1969

Principal Investigator: Mr. Frank E. Heart  
Telephone (617) 491-1150, Ext. 470

Sponsored by  
Advanced Research Projects Agency  
ARPA Order No. 1260

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

INTERFACE MESSAGE PROCESSORS  
FOR THE ARPA COMPUTER NETWORK

QUARTERLY TECHNICAL REPORT NO. 1  
2 January 1969 to 31 March 1969

Submitted to:

Advanced Research Projects Agency  
Washington, D.C. 20301

Attn: Dr. L.G. Roberts

This research was supported by the  
Advanced Research Projects Agency  
of the Department of Defense under  
Contract No. DAHC 15-69-0179

TABLE OF CONTENTS

	<u>PAGE</u>
1. INTRODUCTION.....	1
2. HARDWARE DESIGN.....	3
3. SOFTWARE DEBUGGING.....	5
4. MULTIPLE HOSTS.....	7
5. ROUTING ALGORITHM.....	10

## 1. INTRODUCTION

A contract has been awarded to Bolt Beranek and Newman (BBN) to design, fabricate and install interface message processors (IMPs) for the ARPA computer network. Work on this contract began on 2 January 1969 and is proceeding according to schedule: initial field installations will take place in the Fall of 1969. This Quarterly Report No. 1 describes several aspects of our technical progress during the first quarter of 1969. [BBN Report No. 1763 describes the initial IMP design and we assume a general familiarity with that document. Since we have adhered quite closely to the initial design plans, we have therefore included in this report only new material and modifications to the initial design.]

Work is proceeding on both the hardware and the software design. Section 2 describes the status of the hardware design. Intensive work has also begun on the software package. In the first quarter, we placed some emphasis on building software *tools* as well as the main program. In Section 3 of this report, we describe our plans for software debugging. A stand-alone debugging program (DDT) has been written and will be modified to run in conjunction with the operational IMP program and to assist in *network* debugging.

We have made provision for the connection of up to four Hosts to a single IMP by enlarging the number of Host tables, by expanding the Host routines, and by a change in the method of handling links. We discuss these multiple Host provisions in Section 4.

We have designed, and are currently implementing, a routing algorithm. Each IMP regularly estimates the minimum delay path through the network to each destination and stores the information in an updated routing table. This algorithm and its implementation are discussed in Section 5.

## 2. HARDWARE DESIGN

The prototype interface hardware is now complete. The IMP/Host and the IMP/Modem interface designs that have been realized by Honeywell are functionally identical to those described in our initial design. We have made modifications to the hardware at a detailed level to permit its construction using standard Honeywell micro-pacs.

Some additional protective features have been added. For example, in order to protect against noise in the Modem clock signal, we have inhibited clock transitions from occurring too frequently. Special test and crosspatching features incorporated into the hardware design permit the operational program to perform loop tests on each interface and to indicate detected error conditions.

In addition to the interfaces, several other features have been defined and incorporated into the design:

- 1) Memory protection, under switch control, has been provided for a 1000 (octal) word block of the memory and for the two specific memory locations associated with the Watchdog timer (see 2 below) and power fail interrupts.
- 2) The Watchdog timer is normally held off by a periodic command from the program. Should the program fail in such a way as to stop issuing these commands, the timer will run out, causing a "program failure" interrupt via a

protected location (see 1 above) into a special restart routine in the block of protected memory. When the timer runs out, it will restart itself. After four successive time-outs occur without a successful program restart, an alarm will be presented and the computer will be shut off.

- 3) A set of 24 status indicator lights has been provided on the front of the machine.
- 4) Provision has been made for a program-generated interrupt.
- 5) A relative time clock has been provided. The program can enable or disable the clock and can also read the time into the accumulator. The clock causes a program interrupt to facilitate program event timing.
- 6) Upon the return of power following a power failure, an automatic restart feature will restart the program. In combination with the regular power failure interrupt (which stops the program "cleanly"), the auto restart feature should permit the resumption of normal duties after a power outage, without the need for human intervention.

### 3. SOFTWARE DEBUGGING

We have begun intensive work on the software package. Organization and function of the operational program is essentially as described in our initial design. One part of the software package which has already been built is a stand-alone debugging program (DDT) that will be used as a tool for debugging IMP programs.

The stand-alone DDT program will be used in the initial debugging phase. It occupies approximately 600 registers and can be used to examine and modify storage registers in the IMP or to start programs. To provide maximum isolation between itself and the IMP programs, the stand-alone DDT program runs without interrupts. Teletype input and output are handled by direct testing of the Teletype interface hardware.

We felt that it was also desirable to have a debugging program from which the operational program could be run. Consequently, we have made a provision to incorporate the stand-alone DDT into an operational DDT program that can run in conjunction with the operational IMP program.

This operational DDT program will include most of the stand-alone DDT program along with a Teletype interrupt routine and a software interface with the operational program. The interface will provide a mechanism for transferring characters in both directions between the operational IMP and DDT programs.

The operational DDT program will provide debugging assistance at a single location and, in addition, may be used in *network* testing

and in *network* debugging and maintenance. Using DDT a programmer at the IMP Teletype will have complete control over the operational program and, in particular, he will be able to modify unprotected registers while the operational IMP program is running. More important, a programmer will be able to construct a packet within the IMP and place it on an output queue. *This packet need not necessarily conform to a standard packet format*; for example, it could be a test packet specifically constructed to contain one or more format errors. However, it also could be a packet intended for the operational DDT program at another IMP. With this last capability, it will be possible to display on the Teletype at one IMP the contents of storage registers at another IMP. To accomplish this, an appropriate packet would be sent by the DDT program at one IMP to the DDT program at the other IMP, whereupon a packet containing the desired information would be returned. In a similar fashion it would be possible to modify the contents of storage registers at other IMPs.

We are aware of the risks associated with such powerful debugging tools and we will attempt to restrict access to the DDT when the system is in operational use.

#### 4. MULTIPLE HOSTS

Our initial design incorporated a single Host and a maximum of six communication channels. The operational program has now been modified to accept a maximum of *four* Hosts connected to a single IMP. Since each Host will require a separate Host/IMP interface to replace one IMP/Modem interface, an IMP with four Hosts can have no more than three communication channels.

We thoroughly reconsidered the program timing and organization for the cases of one to four Hosts, and, for each case, we determined that the program timing is still satisfactory, although program size and complexity increases with the number of Hosts.

Timing primarily depends on the maximum rate at which interrupts must be handled. An increase in the number of effective I/O channels increases the maximum rate at which interrupts can arrive. Since each additional IMP/Host interface replaces one IMP/Modem interface, the maximum number of effective I/O channel is unchanged and the program does not encounter substantially different timing conditions than it did in the one-Host case.

The increase in program complexity is due primarily to a change in the method of handling links and to an expansion of the Host routines for disabling interrupts, maintaining multiple tables and for otherwise insuring the ability of the Host routines to be shared. The increase in program size is primarily due to the need for additional tables.

A separate pair of ASCII conversion tables are being considered for each Host. All Host packets marked with a special ASCII bit

will be placed on the appropriate input or output ASCII queue to await conversion in the "background". A limit will be placed on the total number of packets allowed on the ASCII queues.

We have changed the method of handling links to avoid placing a constraint on the ability of the ARPA network to grow and to increase the number of available links at a multiple Host site. The initial method considered for the handling of links was to maintain in each IMP a sparsely filled link table with 32 links to each destination. For 16 IMPs, this table occupied 512 words of core. However, additional IMPs in the network would have required a corresponding increase in the size of the table. For four Hosts at a single site, the number of links on which a given Host could receive packets was reduced from 32 by a factor of 4. This number of links could have remained at 32 via a fourfold increase in the size of the link table to over 2000 words of core; however, this size was considered too large a portion of memory to be allocated to this function in a machine with only 12K of core. Moreover, we considered 8 links per Host at a four-Host site only marginally sufficient, especially since it is still unclear how the Hosts might decide to use the links.

Consequently, the new method selected for handling links avoids placing any internal constraints, such as table size, on the number of IMPs that a network may have and also provides a large *pool* of links which may be shared among all the Hosts at each Host site.

A table will be allocated to handle 64 transmit links and 64 receive links at each IMP. These links are for the combined use of all Hosts at the site and may form a common pool or be legislatively apportioned by the Hosts among themselves. The link

number contains eight bits. Thus, at a four-Host site, 64 unique link numbers could be assigned to each Host if desired. The IMP will accept any 8-bit link number that the Host includes at the beginning of a message and establish that link in its transmit link table. The link is identified in the table by the link number, the destination IMP, and the destination Host.

A link is established in the receive link table of the destination IMP when the first packet on that link arrives. The only limitation placed on the Hosts is that at any one time they may not combine to use a total of more than 64 links for transmitting packets, nor may they receive a combined total of more than 64 links at any one time.

If an established link is not used for a given period of time (currently planned to be 5 or 10 minutes), a procedure will be initiated to delete it from the transmit and receive link tables. Of course, if the link subsequently reappears in use, it will be automatically reestablished — provided fewer than 64 links are then established in the link table.

A Host may request and will receive link status information from its IMP. The Host will have the option to delete transmit links, although there appears to be no need for it to exercise this option under normal circumstances.

A provision has also been made to indicate up to four dead Hosts at a given site. In effect, a dead Host simply means that messages to that Host will be discarded. A dead Host at a multiple Host site will not affect the status of other Hosts at that site.

## 5. ROUTING ALGORITHM

A routing algorithm has been designed for the ARPA network to direct each packet from its source to its destination along a path for which the total estimated transit time through the network is smallest. A path originates at a source IMP, terminates at a destination IMP, and may include one or more intermediate IMPs. A path is therefore composed of one or more IMP-to-IMP segments called legs. Each IMP estimates the best route from itself to a given destination and selects the corresponding leg.

A path is constructed for each packet one leg at a time, beginning at the source IMP. If the first leg terminates at the destination IMP, the packet encounters no intermediate IMPs and the routing is completed. If the first leg terminates at an intermediate IMP, a second leg is selected by that IMP upon receipt of the packet, and the packet is transmitted along that new leg. Each intermediate IMP, in turn, selects the next leg of the path until one leg finally terminates at the destination IMP and the routing is completed.

At each IMP, the next leg is determined straightforwardly by a fast and simple table lookup procedure. For each possible destination, an entry in the table designates the appropriate next leg to be selected. This table is called the routing table.

Each IMP constructs its own routing table. The entries for this table are determined from information transmitted to the IMP from its neighbors and from estimates of its own internal delay. The information from its neighbors consists of delay information,

connectivity information, and dead Host information, which have propagated backward into the network from each destination. The connectivity and dead Host information are monitored for status changes. The received delay information is stored in a table, called the "neighbor's delay table" (NDT), and is used to construct a "minimum delay table" containing minimum delay estimates to each destination.

The NDT at the IMP is the set of minimum delay tables that are transmitted to it by its neighbors. An updated minimum delay table is transmitted (asynchronously) by each IMP to all its neighbors approximately every half second during a time-out routine. Upon arrival, this new table is used to overwrite a portion of the NDT. The entries in the minimum delay table are updated, as are the entries in the routing table, just prior to this transmission.

The routing table is dynamically updated to adjust for changing traffic conditions in the network. For each leg and destination, an estimated delay is formed from two components, namely an estimated delay along the leg and an estimated delay from the end of the leg to the final destination. The sum of the two components is simply called the *delay* for the particular leg and destination and, for each destination, the smallest of the delays is entered into the minimum delay table. The second of the two components is contained in the NDT. The IMP program computes the first component at  $kD$  where  $k$  is a scale factor and  $D$  the solution to the equation

$$D = Q_D(1-P_r) + (100+D+Q_D)P_r ,$$

for which  $Q_D$  is the queue delay in milliseconds with no retransmission;  $P_r$  is an estimate of the probability of retransmission; and 100 msec is the elapsed time between retransmission attempts. The above equation may be solved for  $D$  to yield

$$D = \frac{Q_D + 100P_r}{1 - P_r} .$$

The estimate for the quantity  $Q_D$  is taken to be  $20L$ , where  $L$  is equal to the length of the output queue awaiting transmission on that line. The estimate for  $P_r$  is  $E/20$  where  $E$  is equal to the recorded number of retransmissions among twenty consecutive packets transmitted on that line.  $L$  is updated with every packet and  $E$  is updated every twenty packets. If  $E$  equals 20, the value for 19 is used to avoid dividing by zero.

One or more IMPs are disconnected from the rest of the network when no path exists between them and the other IMPs. This condition may be attributed to the presence of dead lines that connect the IMPs with the rest of the network. A *dead line* is defined by the sustained absence on that line of either received messages or acknowledgements. If no packets arrive on a line for approximately two and a half seconds, the line is defined to be dead and no outgoing packets will be routed onto it. In addition, any packets still remaining on the queue for that line will be rerouted to another queue.

A dead line may reflect trouble in either the communication facilities, in the IMP hardware, or possibly in the IMP itself. Normal

outages on a line due to dropouts, impulse noise, or other error conditions are not expected to result in a dead line since they typically last only a few milliseconds, and only occasionally last as long as a few tenths of a second. Therefore, we expect that a line will be defined as dead only when serious trouble conditions occur.

In the absence of regular packets to transmit over a line, the IMP program transmits *hello* packets at half second intervals. The acknowledgement for a hello packet is called an *I heard you* packet.

A dead line must remain dead long enough for all the IMPs to learn of the condition, since a dead line may cause one or more IMPs to be disconnected from the rest of the network. When thirty consecutive *hello* packets have been acknowledged (an event which consumes at least 15 seconds), the line will be defined to be alive once again and will subsequently be included in the normal routing procedure.

After a line is defined to be dead, it is not permitted to be used as the next leg of a path. Subsequent entries into the composite delay table are generated without reference to routing information from that line. Routing tables in the network are adjusted automatically to reflect the loss of a line as the composite delay tables are continually transmitted between neighboring IMPs.

If every path leading to a destination IMP contains one or more dead lines, it is impossible for packets to reach that IMP and they will therefore be discarded. The routing table will be modified to indicate no path to such an IMP.

Disconnected IMPs can not be rapidly detected from the delay tables that arrive from neighboring IMPs. Consequently, additional information is transmitted between neighboring IMPs to help detect this condition. Each IMP transmits to its neighbors the length of the shortest existing path from itself to each destination. To the smallest such received number per destination, the IMP adds one. This incremented number is the length of the shortest path from that IMP to the destination. If the length ever reaches 15 to any destination, the destination is assumed to be unreachable, all packets to that destination are discarded.

Messages intended for dead Hosts (which are not the same as dead IMPs) cannot be delivered and therefore require special handling to avoid indefinite circulation in the network and possible spurious arrival at a later time. Such messages are purged from the network either at the source IMP or at the destination IMP.

Information that identifies dead Hosts is regularly transmitted between IMPs, accompanying the routing information. A Host computer is only notified about another dead Host when it attempts to send a message to that Host.

## DOCUMENT CONTROL DATA - R &amp; 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) Bolt Beranek and Newman Inc 50 Moulton Street Cambridge, Massachusetts 02138		2a. REPORT SECURITY CLASSIFICATION UNCLASSIFIED	
		2b. GROUP	
3. REPORT TITLE QUARTERLY TECHNICAL REPORT No. 1 2 January 1969 to 31 March 1969			
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 April 1969		7a. TOTAL NO. OF PAGES 17	7b. NO. OF REFS
8a. CONTRACT OR GRANT NO. DAHC 15-69-C-0179		9a. ORIGINATOR'S REPORT NUMBER(S) BBN Report No. 1783	
b. PROJECT NO. 1260		9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report)	
c.			
d.			
10. DISTRIBUTION STATEMENT			
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						