5143

HOST-HOST PROTOCOL

Document No. 1

Steve Crocker
Chairman,
Network Working Group

3 August 1970

3804 Boelter Hall
UCLA
Los Angeles, California
90024

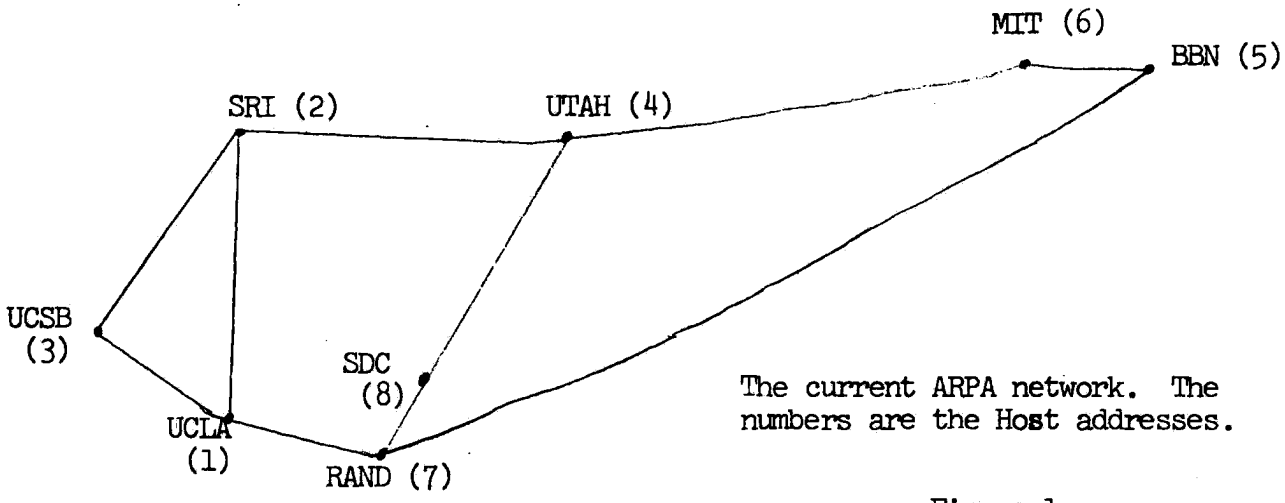
INTRODUCTION

The ARPA network is a store-and-forward message processing system which connects together a variety of computers across the country. A computer attached to the network is called a Host.

The network itself consists of a set of identical, small computers, which are located close to the Hosts and are connected to each other by 50 kilobit-per-second leased lines. These small computers are slightly modified Honeywell DDP-516 computers, and are called Interface Message Processors (IMPs).

Each Host is connected to a single IMP but an IMP may be connected to as many as four Hosts.

At the time of this writing, eight Hosts are connected to eight IMPs. A diagram of the current network is shown in figure 1.



The current ARPA network. The numbers are the Host addresses.

Figure 1

The prime contractor for the network is Bolt, Beranek and Newman, Inc., (BBN). BBN has specified how IMPs and Hosts are connected and what the functional specifications of the IMPs are. These specifications are contained in BBN report #1822 [1]. Broader information is contained in the set of five papers presented at the Spring Joint Computer Conference in May, 1970 [2, 3, 4, 5, 6].

The BBN specifications detail how information is transmitted from one Host to another. We call these specifications the first level protocol or Host-IMP protocol.

Since the Hosts are predominately run under a time-sharing or multiprogramming discipline, additional specifications are necessary to extend

the protocol to the process or user level. The Host organizations have the responsibility for these additional specifications and they have formed a Network Working Group (NWG). The NWG has formulated a second level or Host-Host protocol which extends the first level protocol. The development of this protocol has been documented informally in Network Working Group Requests for Comments. The purpose of this document is to present the version of the Host-Host protocol which has been agreed upon by the Network Working Group. BBN report #1822 is prerequisite to understanding this document, but this document should be otherwise self-contained.

Prospective implementers of this protocol are invited to contact the network staff at UCLA if assistance is desired, and we strongly recommend that contact be made in any case.

Inquiry should be addressed to

Steve Crocker
3804 Boelter Hall
UCLA
Los Angeles, California 90024

phone (213) 825-2368

IMP PROTOCOL

The reader is assumed to be familiar with the concepts of messages, leaders, marking, padding, links, RFNM's, regular messages and control messages as presented in the BBN report. In order to avoid ambiguity, I use the term irregular message instead of control message to mean messages with non-zero type. To be precise about whether a message is traveling from a Host to its IMP, or from the IMP to a Host, I use the notation xHI to refer to messages sent from Host to IMP, and xIH to refer to messages sent from IMP to Host. The "x" is replaced by the message type. Thus a Host receives a RFNM if it receives a 5IH message.

In regular messages, the Host and link field tell where the message came from or goes to and over which link. In most irregular messages the Host and link field refer to a particular link but the message itself travels only between a Host and its (locally connected) IMP. The sole exception is the message type 5. 5IH messages are RFNM's, and these originate in foreign IMPs. 5HI messages are treated exactly like 0HI messages except they do not enter the foreign Host.

As an aid to the Host-Host protocol, BBN has agreed to add the new irregular messages 11HI and 12IH, and to modify the meaning of message 11IH. These changes are recent and have not yet appeared as revisions to BBN report #1822. These new specifications are presented below in the section on flow control.

In order to provide greater flexibility for Hosts sending regular messages, all regular messages have marking. Marking is a (possibly null) sequence of 0 bits followed by a 1 bit, and it comes immediately after the leader. Immediately following the marking is the data-carrying portion of the message, or text. 0HI messages consist of leader, marking and text while 0IH messages consist of leader, marking text and padding.

DESIGN CONCEPTS

Although there is little uniformity among the Hosts in either hardware or operating systems, the notion of multiprogramming dominates all of the systems. Each Host can concurrently support several users, with each user running one or more processes. Many of these processes may want to use the network concurrently, and thus a fundamental requirement of the Host-Host protocol is to provide for process-to-process communication over the network. Since the Host-IMP protocol only takes cognizance of Hosts, and since the several processes in execution within a Host are usually independent, it is necessary for the Host-Host protocol to provide a richer addressing structure.

Another factor which influenced the protocol design is the belief that typical process-to-process communication is not based on solitary messages, but upon a sequence of messages. One example is the sending of a large body of information such as a file from one process to another. Another example is an interactive conversation between two processes with many exchanges.

These considerations led to the notions of connections, a Network Control Program, a control link, control commands, and sockets.

A connection is an extension of a link. A connection connects two processes so that output from one process is input to the other. Connections are simplex, so two connections are needed if two processes are to converse in both directions.

Processes within a HOST communicate with the network through a Network Control Program (NCP). In most HOSTs, the NCP will be part of the executive, so that processes will use sysgen calls to communicate with it. The primary function of the NCP is to establish connections, break connections, and control flow.

In order to accomplish its tasks, a NCP in one HOST must communicate with a NCP in another HOST. To this end, a particular link between each pair of HOSTs has been designated as the control link. Messages received over the control link are always interpreted by the NCP as a sequence of one or more control commands. Messages sent over the control link are called control messages.* As an example, one of the kinds of control commands is used to assign a link and initiate a connection, while another kind carries notification that a connection has been terminated.

---

*Note the difference in usage. Here, control messages are regular messages sent over the control link. IMPs take no notice of control messages. We use the term irregular message for messages of non-zero type. Irregular messages are treated specifically by IMPs; BBN uses the term control message for these.

A major issue is how to refer to processes in a foreign HOST. Each HOST has some internal naming scheme, but these various schemes often are incompatible. Since it is not practical to impose a common internal process naming scheme, an intermediate name space was created with a separate portion of the name space given to each HOST. It is left to each HOST to map internal process identifiers into its name space.

The elements of the name space are called sockets. A socket forms one end of a connection, and a connection is fully specified by a pair of sockets. A socket is identified by a Host number and a 32 bit socket number. The same 32 bit number in different Hosts represents different sockets.

A socket is either a receive socket or a send socket, and is so marked by its low-order bit (0 = receive; 1 = send). This property is called the socket's gender. The sockets at either end of a connection must be of opposite gender.

Except for the gender, this protocol places no constraints on the assignment of socket numbers within a Host. However, the following ideas have gained currency in the Network Working Group and have motivated the protocol design. Some of these ideas probably will be included in the next layer of protocol.

A socket number is envisioned as the concatenation of a high-order, 24-bit user number and a low-order 8 bit tag. Each user is assigned a 24-bit user number which uniquely identifies him throughout the network. Generally this will be the 8-bit HOST number of his home HOST, followed by 16 bits which uniquely identify him at that HOST. Provision can also be made for a user to have a user number not keyed to a particular HOST, an arrangement desirable for mobile users who might have no home HOST or more than one home HOST. This 24-bit user number is then used in the following manner. When a user signs onto a HOST, his user number is looked up. Thereafter, each process the user creates is marked with his user number. When the user signs onto a foreign HOST via the network, his same user number is used to mark processes he creates in that HOST. The foreign HOST obtains the user number either by consulting a table at login time, as the home HOST does, or by noticing the identification of the caller. Thus all the processes which a user creates are marked with his user number. At each Host, there are 128 sockets of each gender for each user number. We envision that a process may request the use of only those sockets whose high-order 24 bits match the process' user number. A request is granted if the socket is not otherwise in use.

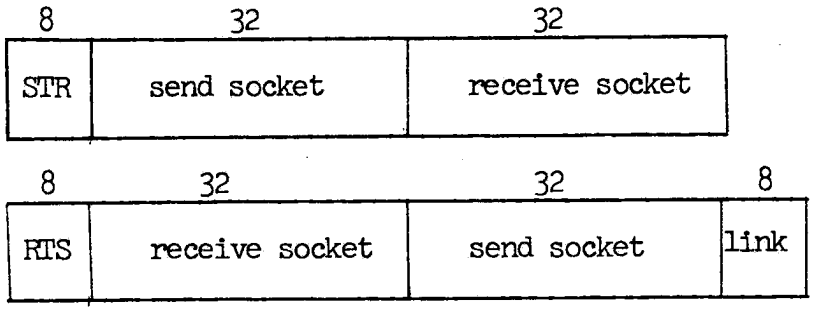
We now make two observations. First, a local socket requested by a process cannot already be in use unless it is by some other process owned by that user. Second, processes marked with the same user number need only know the low-order 8 bits of foreigh socket numbers. It is thus fairly easy for a user to connect his processes across the network; yet the ability to connect to other users' processes remains.

NCP FUNCTIONS

The functions of the NCP are to establish connections, terminate connections, control flow, transmit interrupts, and respond to test messages. These functions are explained in this section, and control commands are introduced as needed. In the next section, the formats of all the control commands are presented together.

Connection Establishment

The commands used to establish a connection are STR and RTS.



103 The STR command is sent from a prospective sender to a prospective receiver, and the RTS from a prospective receiver to a prospective sender. The send socket field names a socket local to the prospective sender; the receive socket field contains a socket local to the prospective receiver. In the RTS command, the link field assigns a link. These two commands are referred to as requests-for-connection (RFC). A STR and an RTS match if the receive socket fields match and the send socket fields match. A connection is established where a matching pair of RFC's have been exchanged. Hosts are prohibited from establishing more than one connection to any local socket.

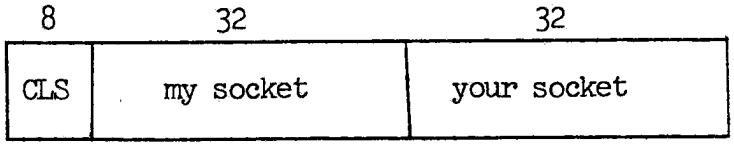
104 With respect to a particular connection, the Host containing the send socket is called the sending Host and the Host containing the receive socket is called the receiving Host. Some control information such as RFNM's, other IMP-to-Host messages, and flow control commands(see below) travel from "receiving" Hosts to "sending" Hosts. A Host may connect one of its receive sockets to one of its send sockets, thus becoming both the sending Host and the receiving Host for that connection.

105 A Host sends an RFC either to request a connection, or to accept a foreign Host's request. Since the same command is used for both requesting and accepting the establishment of a connection, it is possible for both sides to initiate the connection. One consequence is that a family of processes may be created with connection initiating actions built-in, and the processes within this family may be started up in arbitrary order.

106 There is no prescribed lifetime for an RFC. Hosts are permitted to queue incoming RFC's and withold a response for an arbitrarily long time. It is reasonable, for example, for an NCP to queue an RFC which refers to some currently unused socket until some process grabs the socket and tells the NCP to accept or reject the request. Of course, the Host which sent the RFC (or the process within that Host) may not be willing to wait for an arbitrarily long time, so it may abort the request.

Connection Termination

The command used to terminate a connection is CLS.



The my socket field contains the socket local to the sender of the CLS command. The your socket field contains the socket local to the receiver of the CLS command.

Each side must send and receive a CLS command before a connection is completely terminated and the sockets are free to participate in other connections. It is not necessary that both RFC's be exchanged before a connection is terminated.

After a connection is established, CLS commands sent by the receiver and sender have slightly different effects. CLS commands sent by the sender indicate that no more messages will be sent over the connection. This command must not be sent if there is a message in transit over the connection.

A CLS command sent by the receiver acts as a demand on the sender to terminate transmission. However, since there is a delay in getting the CLS command to the sender, the receiver must expect more input.

If a Host wishes to refuse a request for connection, it sends back a CLS instead of a matching RFC. The refusing Host then waits for the initiating Host to acknowledge the refusal by returning a CLS.

Similarly, if a Host wishes to abort its outstanding request for a connection, it sends a CLS command. The foreign Host is obliged to acknowledge the CLS with its own CLS.

Under all circumstances, a Host should quickly acknowledge an incoming CLS so the foreign Host can purge its tables.

Because the CLS command is used both to initiate closing, aborting and refusing a connection, and to acknowledge closing, aborting and refusing a connection, the various race conditions do not lead to ambiguous or erroneous results. Suppose, for example, that Host A sends Host B a request for connection, and then A sends a CLS to Host B because it is tired of waiting for a reply. However, just when A sends its CLS to B, B sends A a CLS to refuse the connection. A will "believe" B is acknowledging the abort, and B will "believe" A is acknowledging its refusal, but the outcome will be correct.

Flow Control

After a connection is established, the sending Host sends messages over the agreed upon link to the receiving Host. The receiving Host takes messages from its IMP and queues them for its various processes. Since it may happen that the messages arrive faster than they are disposed of, some mechanism is required which permits the receiving Host to quench the flow from the sending Host.

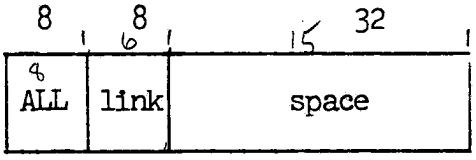
The flow control mechanism requires the receiving Host to allocate buffer space for each connection and to notify the sending Host of how much space is available. The sending Host keeps track of how much room is available and never sends more data than it believes the receiving Host can accept.

To implement this mechanism, the sending Host keeps a counter associated with each connection. The counter is initialized to zero when the connection is established, is increased by ALL control commands sent from the receiving Host (see below), and is decremented by 100 plus the text length in bits of any message sent over the connection. The sending Host may also return all or part of its allocation with a RET command (see below). The purpose of the constant 100 is to approximate the per-message overhead in the receiving Host; since the receiving Host presumably keeps a counter which is incremented and decremented according to the same rules, the receiving Host can be cognizant of small errors between the value of the counter and the actual space available and easily compensate for them.

The sending Host never sends a message over a connection if it would cause the counter to go below zero. There are thus two conditions the NCP must check before sending a message: whether the RFNM is back for the required link, and whether the counter is high enough.

1D21 Ideally, the receiving Host will allocate some buffer space as soon as the connection is established. As messages arrive, they occupy the allocated buffer space. When the receiving process absorbs the waiting text from the buffer, the NCP fires back a new allocation of space for that connection. The NCP may allocate space even if the receiving process has not absorbed waiting text if it believes that extra buffer space is appropriate. Similarly, the NCP may decide not to reallocate buffer space after the receiving process makes it available.

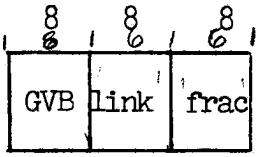
1D21A The control command which allocates space is



This command is sent only from the receiving Host to the sending Host.

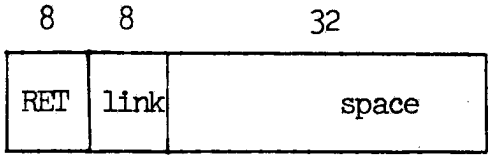
In order to reclaim allocated space, because the space is not being used or for any other reason, the receiving Host may request that the sending Host return all or part of its current allocation.

The control command for this request is



This command is sent only from the receiving Host to the sending Host. The frac is an eight-bit binary fraction, and it specifies what part of the sender's current allocation the sender may keep. Thus a fraction of zero means the sender must return its whole allocation.

Upon receiving a GVB command, the sending Host must return at least the requested allocation. The command for doing so is



Some Hosts will allocate only as much space as they can guarantee for each link. These Hosts will tend to use the GVB command only to reclaim space which is being filled very slowly or not at all. Other Hosts will allocate more space than they have, so that they may use their space more efficiently. Such a Host will then need to use the GVB command when the input over a particular link comes faster than it is being processed.

Naturally, the efficacy of this strategy depends upon how quickly the GVB command can be sent to the sending Host. In order to notify the sending Hosts in minimal time, the IMP's provide a special service which utilizes returning RFNM's to convey flow control information.

A receiving Host may send its IMP a 10HI message, which instructs the IMP to set a "cease bit for the named link. The IMP examines the cease bit whenever it constructs RFNM; if the cease bit is set, the IMP sends back a special RFNM and sends a 12IH message to the local (receiving) Host. When the special RFNM reaches the sending IMP, it is converted to a 10IH message instead of the standard 5IH message.

The sending Host interprets the 10IH message as equivalent to both a RFNM (5IH) and a GVB command with a fraction of zero. Upon receiving a 10IH message, the sending Host returns all of its current allocation for the named link by sending a RET command.

This mechanism of "piggybacking" a GVB on a RFNM is faster, in general, than sending a regular GVB command. However, it may happen that the sending Host suspends transmission at the same time the receiving Host is sending its 10HI message. Therefore, some means is desirable to reset the cease bit in the IMP. The IMP resets the cease bit if it has not

been examined for approximately a minute or if it receives an 11HI message. In response to receiving the 11HI message or in response to timing out the cease bit, the IMP sends an 111H message to the HOst. The IMP will not return an 111H message if it receives an 11HI message and the cease bit is already off.

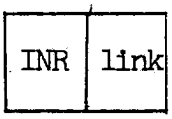
There are some race conditions which can develop, and to resolve them, it is necessary to know the order in which the IMP sends messages to its Host. The IMP maintains three queues of messages for Host, two for regular messages (OHI) and one higher priority queue for "irregular" messages. All irregular messages (message type is non-zero) consists of only a leader and padding. No regular message is sent to a HAst if there are any irregular messages waiting. Of course, irregular messages may be put into the queue for the Host while a regular message is moving from the IMP into the Host, and the irregular message will not be sent until the regular message is finished.

Of particular interest is the timing of RFNM creation and the consequent examination of the cease bit. A RFNM is built using the buffer of the first packet of the message. Thus RFNM's are reutrned after the message has left the IMP for one packet messages, but are returned before the message has left the IMP for multi-packet messages. The cease bit is examined when the RFNM is built.

Interrupts

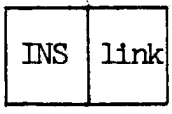
In addition to the simplex transmission path provided by a connection, a facility for sending "interrupts" in either direction. By "interrupt" it is meant that a control message is sent from one Host to another and this control message refers to an established connection. The NCP which receives this control command should notify the process attached to the socket that an interrupt has arrived; no specific action is defined.

This facility permits easy simulation of dial-up terminals which use a long-space or break to as an interrupt. The interrupt command sent from receiving Host to sending Host is



;

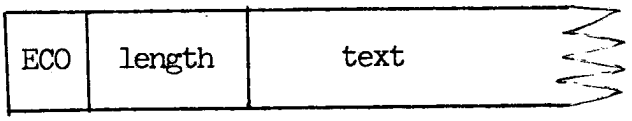
the interrupt command sent from receiving Host to sending Host is



.

Echoes

Whenever the NCP receives a control command of the form



where the length is the length in bits of the text portion, the NCP must return the text in an ERP command to the sender of the ECO. The ERP command is identical in format to the ECO command except for the opcode.

DECLARATIVE SPECIFICATIONS

Message Format

All regular messages shall have marking; the marking may be of any length and should be chosen to suit the sender.

Control messages must consist of an integral number of control commands.

The text of a message sent over a connection is part of a continous data stream, and the message boundry may not be meaningful. Provision is made for definition of message data types which are 8 bit codes the sender includes in the text of messages and which notify the receiver what the meaning of message boundries is and which character set is being used. The only currently agreed upon message data type is zero, and it specifies that message boundries convey no information, that the data is an arbitrary bit string, and that these conditions will prevail for the life of the connection. It is thus sufficient for the first eight bits of the text of the first message to be zero in order to satisfy the conventions of message data types. Possible future message data types might guarantee that a line of input text ended on a message boundary, or that the ASCII character set is in use.

Link Allocation

Link 1 is the control link.

Links 2 through 31, inclusive, are assignable for connections under this protocol.

Links 192 through 255, inclusive, are available for any experimental purpose.

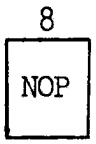
Link 0 is used by the IMP programs and will not be assigned.

Links 32 through 191 are reserved for expansion and should not be used.

Control Commands

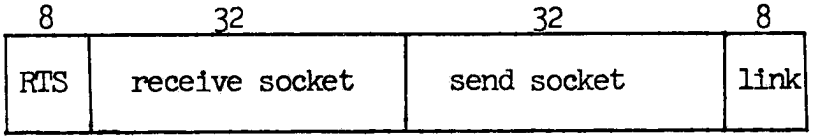
Each control command begins with an 8 bit opcode. These opcodes have values of 0, 1, etc. to permit table lookup upon receipt. Private experimental protocols should be tested using opcodes of 255, 254, etc. Most of the control commands are more fully explained in the various sections under NCP FUNCTIONS.

No Operation



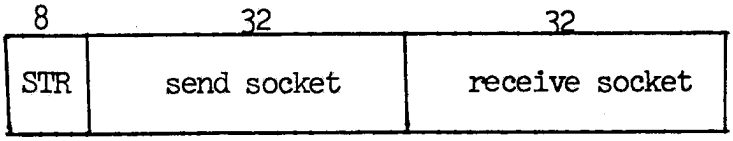
The NOP command may be sent at any time and should be discarded upon receipt. It consists only of zeroes and may thus be useful for formatting command messages.

Request Connection, Receiver to Sender



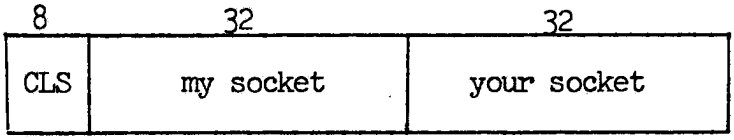
The RTS command is used to establish a connection and is sent from the Host containing the receive socket to the Host containing the send socket. The link is assigned with this command; the link must be between 2 and 31, inclusive.

Request Connection, Sender to Receiver



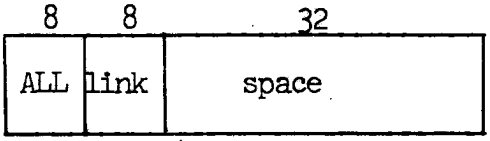
The STR command is used to establish a connection and is sent from the Host containing the send socket to the Host containing the receive socket.

Close



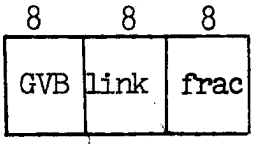
The CLS command is used to terminate a connection.

Allocate



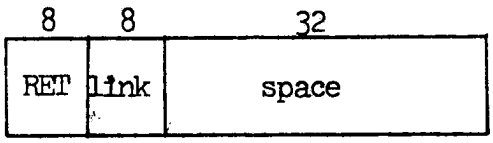
The ALL command is sent from a receiving Host to a sending Host to increase the sending Host's space counter. This command may be sent only while the connection is established.

Give Back



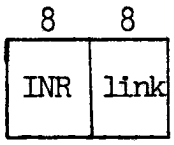
The GVB command is sent from a receiving Host to a sending Host to request that the sending Host return all or part of its space allocation. The frac specifies what portion of the allocation the sending Host may keep. This command may be sent only while the connection is established.

Returning Space



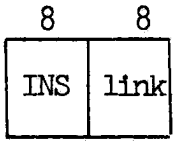
The RET command is sent from the sending Host to the receiving Host to return all or a part of its space allocation, either voluntarily, or in response to a GVB command or a 101H message. This command may be sent only while the connection is established.

Interrupt Sent by Receiving Process



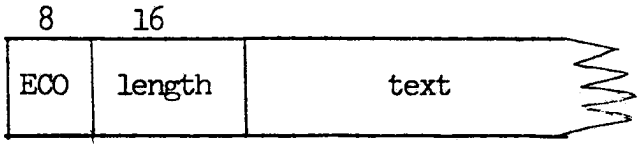
The INR command is sent from the receiving Host to the sending Host when the receiving process wants to interrupt the sending process. This command may be sent only while the connection is established.

Interrupt Sent by Sending Process



The INS command is sent from the sending Host to the receiving Host when the sending process wants to interrupt the receiving process. This command may be sent only while the connection is established.

Echo Request

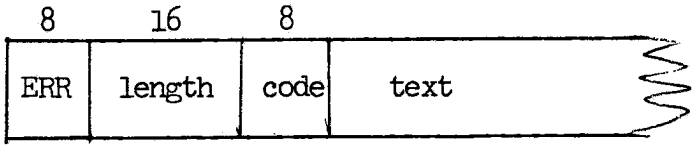


The ECO command is used only for test purposes and does not pertain to the rest of the protocol. A Host may send one at any time, and should expect to receive an ERP command in reply. The length field gives the length in bits of the text field.

Echo Reply

The ERP command is sent in reply to an ECO command. The length field gives the length in bits of the text field. The text field must match the text field of the incoming ECO command.

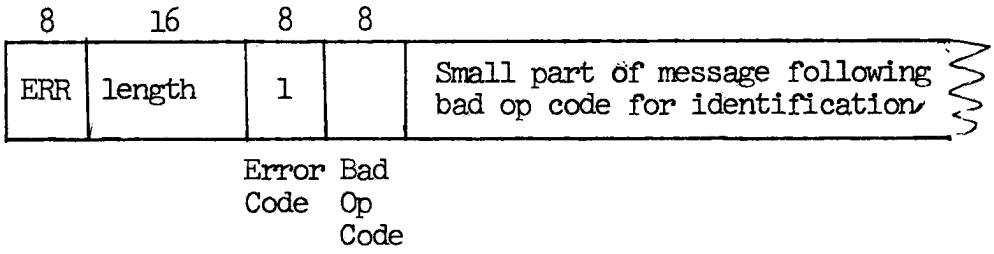
Error Detected



The ERR command may be sent whenever an error is detected in the input from another Host. The length field specifies the length of the code field and the text field. In the case that the error condition has a predefined code, the code field specifies the specific error, and the text field gives parameters. For non-standard errors, the code field is zero and the text field is idiosyncratic to the sender. Implementers of Network Control Programs should publish timely information on their ERR commands in NWG/RFC form.

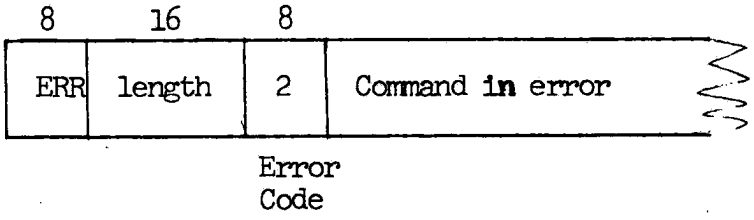
The following codes are defined. Additional codes may be defined later.

1. Illegal Op Code



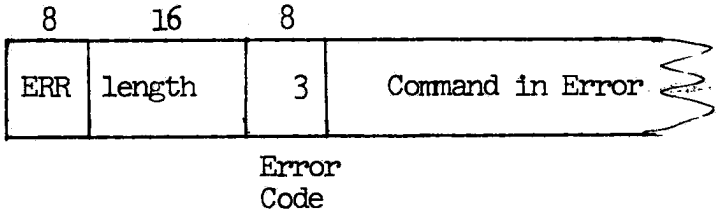
Op code was encountered in operation deblocking.

2. Short Parameter Space



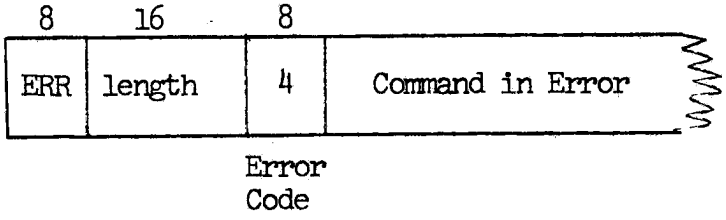
End of message encountered before all expected parameters.

3. Bad Parameter(s)



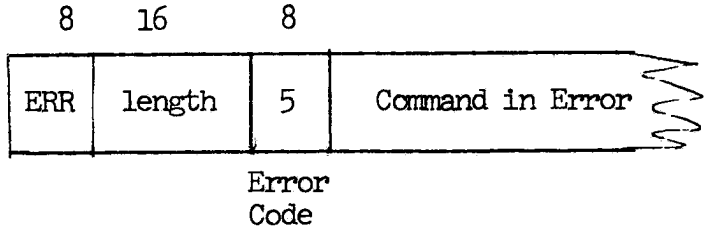
e.g., two receive (send) sockets in an RTS (STR), link number not $1<L<33$, bad socket polarity within command.

4. Request on a Closed (Null) Socket



A request (other than RTS/STR) was made for a non-existent socket.

5. Socket (Link) not Connected



A request which requires a connected socket (link) (i.e., ALL, INR, transmit) was made for an existing but not connected socket (link).

Opcodes

NOP = 0
RTS = 1
STR = 2
CLS = 3
ALL = 4
GVB = 5
RET = 6
INR = 7
INS = 8
ECO = 9
ERP = 10
ERR = 11

BIBLIOGRAPHY

1. "Specifications for the Interconnection of a Host and an IMP." Bolt, Beranak and Newman, Inc. report number 1822, May 1969

2. Roberts, L. G. and B. D. Wessler, "Computer Network Development to Achieve Resource Sharing." AFIPS Proceeding of the 1970 SJCC, pp. 543-549

3. Heart, F. E., et al, "The Interface Message Processor for the ARPA Computer Network." AFIPS Proceedings of the 1970 SJCC, pp. 551-567

4. Kleinrock, L., "Analytic and Simulation Methods in Computer Network Design." AFIPS Proceedings of the 1970 SJCC, pp. 569-579

5. Frank, H., et al, "Topological Considerations in the Design of the ARPA Network." AFIPS Proceedings of the 1970 SJCC, pp. 581-587

6. Carr, C. S., et al "Host-Host Communication Protocol in the ARPA Network," AFIPS Proceedings of the 1970 SJCC, pp. 589-597