Network Working Group
Request for Comments: 714
NIC: 35144

A. McKenzie
BBN-NCC

April 1976

A Host/Host Protocol for an ARPANET-Type Network

Recently we have been involved in the planning of a network, which, if implemented, would use ARPANET IMPs without modification, but would allow re-specification of Host/Host (and higher level) Protocol. The remainder of this document is a slightly edited version of our recommendation for Host/Host protocol; we thought that it might be of interest to the ARPANET Community.

I. INTRODUCTION

The Host/Host Protocol for the ARPANET was the first such protocol designed for use over a packet-switched network. The current version has been in existence since early 1972 and has provided for the transportation of billions of bits over tens or hundreds of thousands of connections. Clearly, the protocol is adequate for the job; this does not mean that it is ideal, however. In particular, the ARPANET Host/Host protocol has been criticized on the following grounds (among others):

(1) It is specified as a simplex protocol. Each established connection is a simplex entity, thus two connections (one in each direction) must be established in order to carry out an exchange of messages. This provides great generality but at a perhaps unacceptable cost in complexity.

(2) It is not particularly robust, in that it cannot continue to operate correctly in the face of several types of message loss. While it is true that the ARPANET itself rarely loses messages, messages are occasionally lost, both by the network and by the Hosts.

(3) Partly because of the simplex nature of connections, the flow control mechanisms defined in the ARPANET protocol do not make efficient use of the transactional nature of much of data processing. Rather than carrying flow control information (in the form of permits, or requests for more information) in the reverse traffic, a separate channel is set up to convey this information. Thus, for transactional systems, up to twice as many messages are exchanged (half for flow control information and half for data) as would be needed for data alone.

(4) Prohibition against the multiple use of a connection termination point makes the establishment of communication with service facilities extremely cumbersome.

The International Federation for Information Processing (IFIP) Working Group 6.1 (Packet-switched Network Internetworking) has recently approved a proposal for an internetwork end-to-end protocol. The IFIP Protocol is based on experience from the ARPANET, the (French) Cyclade Network, and the (British) NPL Network, as well as the plans of other networks. Thus, one would expect that it would have all of the strengths and few (or none) of the weaknesses of the protocols which are in use on, or planned for, these networks.

In fact, the IFIP Protocol avoids the deficiencies of the ARPANET protocol mentioned above. Connections are treated as full-duplex entities, and this decision permits flow control information to be carried on the reverse channel in transaction-oriented systems where there is reverse channel traffic occurring naturally. In addition, the IFIP Protocol is to some extent self synchronizing; in particular, there is no type of message loss from which the Protocol does not permit recovery in a graceful way.

The IFIP Protocol makes a minimal number of assumptions about the network over which it will operate. It is designed to permit fragmentation, as a message crosses from one network to another, without network reassembly. It anticipates duplication, or non-delivery, of messages or message fragments and provides ways to recover from these conditions. Finally, it permits delivery of messages at their destination Host in a completely different order from the order in which they were input by the source Host. Unfortunately, it achieves these advantages at a relatively high overhead cost in terms of transferred bits. The complete source and destination process addresses are carried in every message, 24-bits of fragment identification are carried with each fragment and 16-bits of acknowledgement information are else carried in every message.

When considering channel capacities of hundreds of kilobits (or more), message overhead of a few hundred bits is a modest price to pay in order to achieve great flexibility and generality. However, for a stand-alone network of the type under consideration, and especially in view of the anticipated use of many circuits of 10kbs capacity, the IFIP Protocol offers far more generality than is needed, for which a fairly severe overhead price is paid.

The virtual circuit protocols currently being debated within the International Telegraph and Telephone Consultative Committee (CCITT) are a step in the opposite direction. Virtual circuit protocols attempt to make a packet switching network indistinguishable (from a

customer's point of view) from a switched circuit network, except possibly in regard to error or delay characteristics. Thus, virtual circuit protocols generally place responsibility for end-to-end communications control within the network rather than within the Hosts. For example, when a receiving Host limits the rate at which it accepts data from the network, the network in turn limits the rate of input from the Host which is transmitting this data stream. Host protocols which are designed for virtual circuit networks can be quite simple, if somewhat inflexible. For example, the Host might give the network a "link number" or "index" and ask the network to set up a virtual circuit to some other Host to be associated with this number, and report back if and when the circuit is established. However, significant development would be required to add a virtual circuit capability to the ARPANET IMP software; the required changes would seem to be more expensive and carry greater uncertainty than they are worth.

In light of the above, our approach in defining this proposed protocol has been to start with the ARPANET Host/Host Protocol and modify it according to some of the concepts of the IFIP Protocol in order to remedy its major deficiencies. The remainder of this document specifies the protocol, which we have designed for this purpose.

II. COMMUNICATION CONCEPTS

The IMP subnetwork imposes a number of physical restrictions on communications between Hosts. These restrictions are presented in BBN Report No. 1822. In particular, the concepts of leaders, messages, padding, message ID's and message types are of interest to the design of Host/Host Protocol. The following discussion assumes that the reader is familiar with these concepts.

The IMP subnetwork takes cognizance only of Hosts, but in general a Host connected to the network can support several users, several terminals, or several independent processes. Since many or all of these users, terminals, or processes will need to use the network concurrently, a fundamental requirement of the Host/Host Protocol is to provide process-to-process communication over the network. Thus, it is necessary for the Host/Host Protocol to provide a richer addressing structure than is required by the IMP subnetwork.

Processes within a Host are envisioned as communicating with the rest of the network through a network control program (NCP) resident in that Host, which implements the Host/Host protocol. The primary functions of an NCP are to establish connections, break connections, and control data flow over connections. A connection couples two processes so that the output from one process is input to the other

and vice versa. The NCP may be implemented either as part of the Host's operating system or a separate user process, although it must have the capability of communicating with all of the processes or routines which are attempting to use the network.

In order to accomplish its tasks, the NCP of one Host must communicate with the NCPs of other Hosts. To this end, a particular communication path between each pair of Hosts has been designated as the control connection. Messages transmitted over the control connection are called control messages, and must always be interpreted by an NCP as a sequence of one or more control commands. For example, one kind of control command is used to initiate a connection while another kind carries notification that a connection has been terminated.*

* Note that in BBN Report No. 1822, messages of non-zero type are called control messages, and are used to control the flow of information between a Host and its IMP. In this document the term "control message" is used for a message of type zero transmitted over the control connection. The IMPs take no special notice of these messages.

The maximum size of a message is limited by the IMP subnetwork to approximately 1000 8-bit bytes, and in fact may be further limited by the receiving Host for flow control reasons, as described later.

Accordingly, the transmitting process, or its Network Control Program, must take responsibility for fragmenting long interprocess messages into messages of a size conforming to the Host/Host and Host/IMP protocols. For this reason, it is impossible for a sending Host to guarantee that any significance should be attached to message boundaries by receiving processes. Nevertheless, message boundaries will occur naturally, and should be used in a reasonable way wherever possible; that is, a sending process or its NCP should not act arbitrarily in deciding to fragment messages. For example, this protocol specifies that each control message must contain an integral number of control commands and no single control command will be split into two pieces which are carried through the network in separate messages.

A major concern of the Host/Host Protocol is the definition of the method for references to processes in other Hosts. In order to facilitate this, a standard name space is used, with a separate portion of the name space allocated to each Host. Each Host therefore must map internal process identifiers into its portion of this 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, one in each Host. A socket is

identified by a Host number and a 16-bit socket number. The same 16-bit socket number in different Hosts represents difference sockets. In order to avoid the transmission of a pair of 16-bit socket numbers in each message between these sockets, the process of connection establishment allows each Host to define a mapping, valid for the lifetime of the connection being established, from the 32 bits which specify the socket pair to an 8-bit number.

No constraints are placed on the assignment of socket numbers; however, since a pair of socket numbers defines a unique connection, it is clear that in assigning socket numbers, a Host must ensure that for each new connection at least one of the socket numbers is unique. For example, a Host which supports many terminals might choose to use a terminal's physical interface number as a portion of the socket number involved in any connection established on behalf of that terminal. This would insure uniqueness at the terminal end. Thus, no conflict would occur if several terminals attempted to access a common resource (identified by its own unique socket number).

From the foregoing it should be clear that the Host/Host protocol allows a single socket to participate in several connections simultaneously. This is quite similar to what happens in the telephone system, where a company, as well as an individual, can be identified with a phone number. As seen from the outside, the phone number of a company is sharable, since several conversations can proceed at the same time and the caller does not have to worry about the already existing conversations. Conversely, the phone number of an individual is not sharable, since he can process only one conversation at a time; the same is generally true of a connection to a terminal which might be using the network.

A final major concept which should be explained is the "windowing" concept, which is used for flow control. This concept is adapted from the IFIP protocol with some appropriate modifications for use in an ARPANET-type network. When a connection is established, a sequence number is initialized to some specified starting point and the receiver allocates a certain number of credits to the sender. Each credit entitles the sender to transmit one message; that is, the receiver agrees to provide buffering for the number of messages specified by the number of credits granted. If one thinks of sequence numbers advancing from left to right, the initial sequence number defines the left edge of a window into the entire sequence number space and the credit, when added to the initial sequence number, defines the right edge of the window. The transmitting process is permitted to send as many messages and as would fill the window, but not more.

When a receiver receives a message whose sequence number is at the left window edge (or several consecutive messages extending rightward from the left window edge) the receiver returns an acknowledgement for the rightmost such message, along with a new credit, and advances his own window; its new left edge immediately follows the last acknowledged message and it's new right edge is at the location defined by adding the new credit to the new left window edge. Similarly, when a sender receives an acknowledgement he advances his own left window edge to the location in the sequence number space specified by the acknowledgement and his own right window edge to the location specified by adding the new credit allocation to the left window edge. Fields are reserved in each data message to carry an acknowledgement and a credit for traffic flowing in the reverse direction. Thus, in the case of interactive or transactional exchanges, no control messages need to be sent.

In the event that a sender does not receive acknowledgements for previously transmitted messages within some timeout period, the messages are transmitted again, using the same sequence number as was previously assigned. This allows straightforward recovery from the situation of lost messages. On the other hand, if it is the returning acknowledgement which is lost, the fact that the retransmitted message carries an identical sequence number allows the receiver to discard it. However, the receiver should notice that at the time of retransmission the sender had not received an acknowledgement; therefore, the receiver should re-acknowledge this (and any subsequently received messages) by transmitting an acknowledgement bearing the current left window edge. Thus, in both the case of lost data messages and the case of lost acknowledgements the protocol remains synchronized.

The primary difference between this protocol and the IFIP Protocol is in the size of the sequence number field. The IFIP Protocol is designed for interconnections of many networks with huge variabilities in delay and with a strong possibility that messages will not be delivered at the destination in the same order in which they were transmitted by the source. Thus, the IFIP Protocol uses a 16-bit sequence number field which, even at megabit per second rates cannot be completely cycled through in less than several hours. However, the proposed ARPANET-type network has the characteristic that delays are typically short, messages are rarely lost, and they are always delivered in the same order in which they were sent if they are delivered at all. Therefore, this Host/Host Protocol uses only a 4-bit sequence number field which, of course, is cycled through every 16 messages. This imposes the constraint that a window may never be larger than eight messages. Since the sequence number is contained in a 4-bit field, it is also possible to use only four

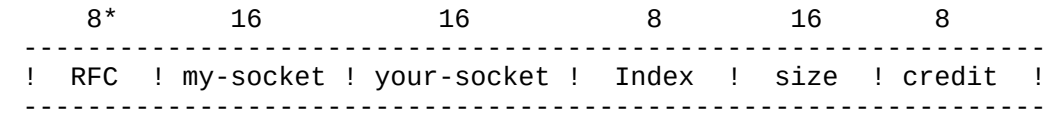
bits for each of the credit and acknowledgement fields; thus, this protocol uses only 12 bits in each message header rather than 40 bits used under the IFIP Protocol.

III. NCP FUNCTIONS

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

Connection Establishment

The command used to establish the connection is the RFC (request for connection).



* The number shown above each control command field is the length of that field in bits.

The RFC command either requests the establishment of a connection between a pair of sockets or accepts a previously received request for connection. Since the RFC command is used both for requesting and accepting the establishment of a connection, it is possible for either of two cooperating processes to initiate connection establishment. Even if both processes were to simultaneously request the establishment of a connection, each would interpret receipt of the RFC sent by the other as an acceptance of its own RFC, and thus the connection would be established without difficulty. The my-socket and your-socket fields in the RFC identify the sockets which terminate the ends of the connection at each Host. The index field of the RFC specifies an index number which will be contained in each data transmission sent over this connection from the "my-socket" to the "your-socket" end of the connection. The size field of the RFC specifies the maximum number of 8-bit bytes which are permitted to be sent from the "your-socket" to the "my-socket" end of the connection in any one message. The credit field of the RFC specifies the initial size (in the range 0-7) of the window in the "your-socket" to the "my-socket" direction of the connection. A pair of RFCs exchanged between two Hosts matches when the my-socket field of one equals your-socket field of the other, and vice versa. The connection is established when a matching pair of RFCs has been exchanged.

Connections are uniquely specified by the sockets which terminate the connection; thus, a pair of socket numbers cannot be used to identify two different connections simultaneously. Similarly, the index is used to specify which connection a data message pertains to; thus, an index value cannot be reused while the connection to which it was first assigned is still active or in the process of being established. For example, consider an RFC sent from Host A to Host B whose my-socket field contains the value X, your-socket field contains the value Y, and index contains the value Z. Until the requested connection has been closed (even if it is never established) or reinitialized, Host A is prohibited from sending a different RFC to Host B whose my-socket field and your-socket fields are X and Y, or whose index field is Z. Note that the prohibition against the reuse of the values X and Y treats them as a pair; that is, another RFC may be sent from Host A to Host B, whose my-socket field contains the value X so long as the your-socket field contains some value other than Y.

In general there is no prescribed lifetime for an RFC. A Host is permitted to queue incoming RFCs and withhold a response for an arbitrarily long time, or, alternatively, to reject requests immediately if it has not already sent a matching RFC. Of course, the Host which originally sent the RFC may be unwilling to wait for an arbitrarily long time so it may abort the request.

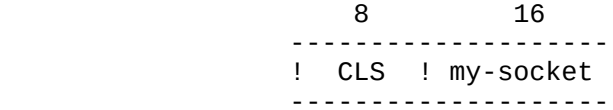
The decision to queue or not to queue incoming RFCs has important implications which must not be ignored. Each RFC which is queued, of course, requires a small amount of memory in the Host doing the queuing. If the incoming RFC is queued until a local process takes control of the local socket and accepts (or rejects) the RFC, but no local process ever takes control of the socket, the RFC must be queued "forever". On the other hand, if no queuing is performed, the cooperating processes which may be attempting to establish communication may be able to establish this communication only by accident.

The most reasonable solution to the problems posed above is for each NCP to give processes running in its own Host two options for attempting to initiate connections. The first option would allow a process to cause an RFC to be sent to a specified remote socket, with the NCP notifying the process as to whether this RFC was accepted or rejected by the remote Host. The second option would allow a process to tell its own NCP to "listen" for an RFC to a specified local socket from some remote socket (the process might also specify the particular remote socket and/or Host it wishes to communicate with) and to accept the RFC (i.e., return a matching

RFC) if and when it arrives. Note that this also involves queuing (of "listen" requests) but it is internal queuing, which is susceptible to reasonable management by the local Host.

Connection Termination

The command used to terminate a connection is CLS (close).



The my-socket field and your-socket field of the CLS command identify the sockets which terminate the connection being closed. Each side must send and receive a CLS command before the connection termination is completed and prohibitions on the reuse of the socket pair and index value are ended.

It is not necessary for connection to be established (i.e., for both RFCs to be exchanged) before connection termination begins. For example, 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 connection it sends a CLS command. The foreign Host is obliged to acknowledge the CLS with its own CLS. Note that even though the connection was never established, CLS commands must be exchanged before the prohibition on the reuse of the socket pair or the index is completely ended. Under normal circumstances a Host should not send a CLS command for a connection on which that Host has unacknowledged data outstanding. Of course, the other Host may have just transmitted data so the sender of the CLS command may expect to receive additional data from the other Host.

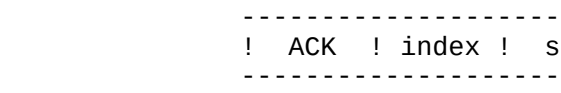
The Host should quickly acknowledge an incoming CLS so that the foreign Host can purge its tables. In particular, in the absence of outstanding unacknowledged data a Host must acknowledge an incoming close within 60 seconds. Following a 60 second period, the Host transmitting a CLS may regard the socket pair and the index as "unused" and it may delete the values from any tables describing active connections. Of course, if the foreign Host malfunctions in such a way that the CLS is ignored for longer than 60 seconds, subsequent attempts to establish connections or transmit data may lead to ambiguous results. To deal with this possibility, a Host should in general "reinitialize" its use of connection parameters before attempting to establish a new

connection to any Host which has failed to respond to CLS commands. Methods for reinitializing connection parameter tables are described below.

Acknowledgement

As described in the previous section, flow control is handled by a windowing scheme, based on sequence numbers. Credits and acknowledgements can be piggybacked on data traveling over the reverse channel. Thus, in general, acknowledgement of the receipt of messages will take place over the data connection rather than over the control connection. However, there are some cases when it may be desirable to pass acknowledgements over the control connection (for example, when there is no data to be returned in the reverse direction). In addition, for efficiency it may be desirable to negatively acknowledge data transmissions known not to have been delivered, rather than waiting for the timeout and retransmission mechanism to cause such messages to be retransmitted. [Note that such negative acknowledgement is not required, since timeout and retransmission is always sufficient to guarantee eventual delivery of all data, but may be used to increase the efficiency of communication.] Since the frequency of use of the negative acknowledgement system over an ARPANET-type network will be extremely low, it is undesirable to leave space for negative acknowledgements in the header of every data message. Thus, negative acknowledgement can be most conveniently handled by control messages.

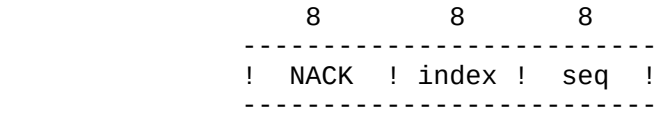
There are two commands dealing with acknowledgements.



The ACK (acknowledgement) command carries three data fields. The index value is the index used by the sender of the acknowledgement to identify the connection. The sequence ("seq") field contains the sequence number of the highest-numbered sequential data message correctly received over the connection. [The very first data message to be transmitted over a newly established connection will have the sequence number one; until this data message is correctly received, any acknowledgement commands transmitted for this connection (for example, to change the credit value) will have the sequence field set to zero. This applies whether the "acknowledgement" is carried by an ACK command or is contained in data messages being sent to the foreign Host over the connection.] The credit ("crd") field contains a number, in the range 0-7,

which gives the size of the receive window. This number, when added to the "seq", gives the sequence number of the highest numbered message which is permitted to be transmitted by the foreign Host. Thus, a credit of zero says that the Host transmitting the ACK command is currently not prepared to accept any messages over the connection; and a credit of 7 says the Host is prepared to accept up to 7 messages over the connection. Of course, since the sequence number is contained in a 4-bit field, the addition of the sequence number and the credit value must be performed modulo 16 (sequence number zero immediately follows sequence number 15).

As noted above, the ACK command is intended for use with data connections where there is no data flow in one direction, for example, the transmission of a file to a line printer. In fact it should be clear that, since transmission of control messages is not synchronized with transmission of data messages (either in the network or, more importantly, in the transmitting NCP), ACK commands should not be sent for any connection over which data is flowing in the same direction. Thus, if an ACK command is generated, the NCP which transmits it must insure that the control message which contains it is transmitted prior to the transmission of new data messages for the same connection.

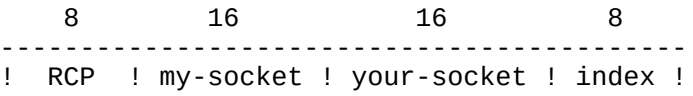


The NACK (negative acknowledgement) command contains two data fields. As with the positive acknowledgement command described above, the first field is the index number assigned to this connection by the sender of the NACK. However, the second field contains only the 4-bit sequence number, right justified in an 8-bit field, of the data message for the connection in question which is being negatively acknowledged. As previously noted, the NACK serves no vital function in the protocol but may occasionally allow more efficient communication. The NACK is intended to be used when the window width is greater than one, the message at the left window edge has not been correctly received, and messages toward the right of the window have been correctly received. A timeout will eventually cause the retransmission of the missing message, at which point the left window edge can be moved forward several messages. Use of the NACK, however, could trigger the immediate retransmission of the missing message and thus reduce the delay. Of course, if more than one message is missing it may

be desirable to send several NACKs for one index in a single control message; the protocol permits this, although it is extremely unlikely to occur.

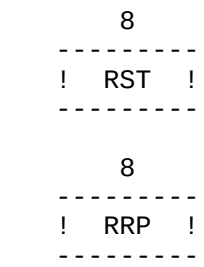
Re-initialization

Occasionally, due to lost control messages, system crashes, NCP errors, or other factors, communication between two NCPs will be disrupted. One possible effect of any such disruption might be that neither of the involved NCPs could be sure that its stored information regarding connections with the other Host matched the information stored by the NCP of the other Host. In this situation, an NCP may wish to reinitialize its tables and request that the other Host do likewise. This re-initialization may be requested for a particular index and/or socket pair, or globally for all connections possibly established with the other Host. For these purposes, the protocol provides three control commands as described below:



The RCP (reinitialize connection parameters) command contains three data fields. The my-socket and your-socket fields contain a pair of socket numbers, which define a connection; the index field contains a value which would identify data messages over a connection. When this command is received by an NCP it should purge its tables of any reference to a connection identified by the socket pair or any reference to a connection for which received data would be identified by the specified index value; of course, only connections using these values with the Host sending the RCP would be purged. In effect, the Host sending the RCP command is saying: "I am about to send you an RFC using this socket pair and this index to identify a data connection, which I hope we can agree to establish. I do not believe that any use of this socket pair or this index conflicts with any previous use, but if you believe it does, please record the fact (for later examination) as an error and then delete from your tables the conflicting information so that we may proceed to establish the connection."

In case more global difficulties or loss of state information are suspected, the protocol provides the pair of control commands RST (reset) and RRP (reset reply).



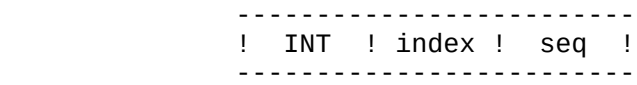
The RST command is to be interpreted by the Host receiving it as a signal to purge its tables of any entries which arose from communication with the Host which sent the RST. The Host sending the RST should likewise purge its tables of any entries which arose from communication with the Host to which the RST was sent. The Host receiving the RST should acknowledge receipt by returning an RRP. Once the first Host has sent an RST to the second Host, the first Host should not communicate with the second Host (except for responding to RST) until the second Host returns an RRP. If both NCPs decide to send RSTs at approximately the same time, each Host will receive an RST and each must answer with an RRP even though its own RST has not been answered.

A Host should not send an RRP when an RST has not been received. Further, a Host should send only one RST (and no other commands) in a single control message and should not send another RST to the same Host until either 60 seconds have elapsed or a command which is not an RST or RRP has been received from that Host. Under these conditions, a single RRP constitutes an answer to all RSTs sent to that Host and any other RRPs arriving from that Host should be discarded.

Interrupts

It is sometimes necessary in a communication system to circumvent flow control mechanisms when serious errors or other important conditions are detected. For example, the user of a time sharing terminal who creates and begins the execution of a program which contains an erroneous infinite loop may need to "attract the attention" of the operating system to ask it to cancel the execution of his program, even though the operating system may normally "listen" to the terminal only when the program in execution asks for input. Similarly, in a computer communication network, where flow control may prevent the transmission of data from one process to another, under certain extraordinary conditions it may be necessary to pass a signal from one process to another. Since the channel between the NCPs of two Hosts is not subject to the flow control mechanisms imposed on the data

connections, it is possible to transmit such an "out-of-band" signal over the control connection, and for this purpose the INT (interrupt) command is provided.

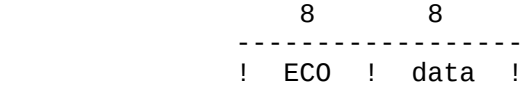


The INT command contains two data fields. The index field identifies the data connection to which the "interrupt" pertains; the sequence number ("seq"), which is four bits right-justified in an eight-bit field, gives the sequence number of the first data message which should "come after" the interrupt. In other words, the INT command notifies the receiving NCP of an exception condition which must be synchronized with the data stream, and the sequence number provides the necessary synchronization. Any data messages with sequence numbers to the left of the specified sequence number were generated before the exception condition arose.

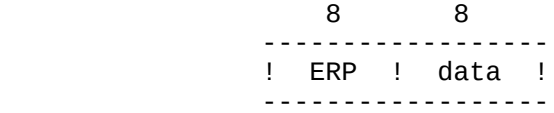
An NCP which receives an INT command should advance the right window edge of the specified data connection so that the window contains at least the sequence number specified in the interrupt command. (It may be necessary to acknowledge data messages which were not correctly received or were not buffered in order to be able to advance the window to this point; justification is provided by the assumption that the INT was sent only because the flow control mechanisms were preventing the transmission of important information.) Of course, the interrupt or exception signal itself is subject to the interpretation of the Host receiving the signal, but should have a meaning equivalent to: "notify the process in execution, or that process' superior, that something exceptional has happened and that the data now buffered is an important message."

Test Inquiry

It may sometimes be useful for one Host to determine if some other Host is carrying on network conversations. The control command to be used for this purpose is ECO (echo).



The data field of the ECO command may contain any bit configuration chosen by the Host sending the ECO. Upon receiving an ECO command, an NCP should respond by returning the data to the sender in an ERP (echo reply) command.



A Host should respond (with an ERP command) to an incoming ECO command within a reasonable time, here defined as sixty seconds or less. A Host should not send an ERP when no ECO has been received.

IV. DECLARATIVE SPECIFICATIONS

Message Format

All Host-to-Host messages which conform to this protocol shall be constructed as follows:

Bits 1-96: Leader - This field is as specified in BBN Report No. 1822, with the following additional specifications.

Bits 38-40: Maximum Message Size - This field should be zero for all control messages. For messages sent over data connections, the value of this field should be calculated from the size received in the RFC which established the connection.

Bits 65-76: Message-id - This field is subdivided into eight bits giving the index of the connection of which the message is a part, and four bits giving the sequence number of the message. The index is contained in bits 65-72, and the sequence number in bits 73-76.

Bits 97-100: Acknowledgement - This field contains the four-bit sequence number of the highest-numbered data message to the left of the window for this connection; that is, the sequence number identifying the highest-numbered of the sequence of consecutively numbered (none missing) data messages which have been correctly received over this connection. If no data messages have been received since the connection was established, this field must contain the value zero. This field is not used (i.e., may have any value) in control messages.

Bits 101-104: Credit - This field contains a number in the range 0-7. Adding this number (modulo 16) to the sequence number in the acknowledgement field (bits 97-100) gives the highest sequence number which the foreign Host is permitted to send over this data connection. Thus, a value of zero in this field indicates that no new data messages should be sent, and a value of seven indicates that the foreign Host may send up to seven messages beyond the message whose sequence number is specified by the acknowledgement bits. Since flow control does not apply to messages sent over the control connection, this field may have any value in control messages.

Bits 105 - ... : Text and padding - A sequence of 8-bit bytes of text, followed by padding, as specified in BBN Report No. 1822.

Index Assignment

Index values must be assigned (in bits 65-72) as follows:

Number	Assignment
0	Identifies a control connection
1	Reserved for revisions to this protocol
2-191	Identify data connections
192-255	Reserved for expansion or for other protocols

Sequence Number Assignment

Every data message contains a sequence number in bits 73-76. The sequence number is used by the receiver to detect the fact that a transmitted message has been lost, to identify the correct location in the data stream to insert a retransmitted (and therefore probably out of order) message which was previously lost (or to detect the retransmitted message as a duplicate) and to identify acknowledged messages (or sequences of messages) to the sender. The sequence number is also used by the flow control mechanism. Since the IMP subnetwork itself contains elaborate mechanisms to achieve these same goals, it is not anticipated that the error-recovery mechanisms based on the sequence numbers will be called into play frequently, and thus their efficiency is not of primary importance.

Sequence numbers are assigned to the two directions of a connection independently. For a given direction of a connection, the first data message transmitted after the connection is

established must have sequence number one. Subsequent messages are assigned sequentially increasing (modulo 16) sequence numbers; that is, sequence number zero is assigned to the message following message number 15.

Sequence numbers are not assigned to control messages, since the protocol is designed to permit these messages to be delivered out-of-sequence without ill effect, and since flow control cannot be applied to the control link.

Control Messages

Messages sent over the control connection have the same format as other Host-to-Host messages, with the exceptions noted above. However, control messages may not contain more than 120 8-bit bytes of text. Further, control messages must contain an integral number of control commands; a single control command must not be split into parts which are transmitted in different control messages.

Message Transmission and Retransmission

Control messages may be transmitted whenever they are required. Data messages, however, may be transmitted only when permitted by the flow control mechanism; that is, whenever the sequence number assigned to the message is within the "window" for the appropriate direction of the given connection. The "left window edge" (LWE) is defined by the highest sequence number (modulo 16) which has been acknowledged (or zero, if no messages have been acknowledged). The "right window edge" (RWE) is defined by adding (modulo 16) the most recently received credit to the left window edge. [Note that LWE=RWE if the most recently received credit is zero.] A message with sequence number SEQ may be transmitted only if, prior to the (possible) reduction modulo 16 of the SEQ and/or RWE, it is true that

LWE less-than SEQ less-than-or-equal RWE

Messages should be retransmitted whenever any of the following conditions occur:

- The IMP subnetwork has returned an "Incomplete transmission" (type 9) or "Error in Data" (type 8) response to the message (identified by having bits 41-76 of the response equal to those bits of the transmitted message). Note that this condition applies to control messages as well as data messages.

- The sequence number of this message is equal to (LWE + 1), and it has been more than 30 seconds since the message was last transmitted.

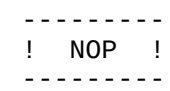
- The sequence number of the message is specifically identified in a NACK command for this connection from the foreign Host.

Since messages may occasionally have to be retransmitted, it is clear that they should not be discarded by the transmitting NCP until they have been acknowledged. A message is considered to be acknowledged when its sequence number, or the sequence number of any message to the right of it in the same direction of the given connection, is returned in the acknowledgement field of a data message transmitted in the other direction over this connection, or is returned in an ACK command for this connection from the foreign Host.

Control Commands

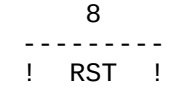
Control commands are formatted in terms of 8-bit bytes. Each command begins with a one byte opcode. Opcodes are assigned the sequential values 0, 1, 2, ... to permit table lookup upon receipt. The conditions underlying the design and anticipated use of the control commands are described in Section III.

NOP - No Operation



The NOP command may be sent at any time and should be discarded by the receiver. It may be useful for formatting control messages.

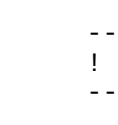
RST - Reset



The RST command is used by one Host to inform another that all information regarding any previously existing connections between the two Hosts should be purged from the NCP tables of the Host receiving the RST. Except for responding to RSTs, the Host which sent the RST should not communicate further with the other Host until an RRP is received in response. When a Host is about to

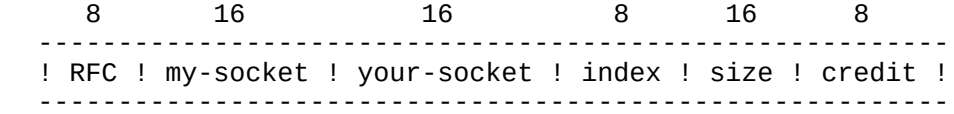
begin communicating (e.g., send an RFC command) to another Host with which it has no open connections, it is good practice to first send an RST command and wait for an RRP command.

RRP - Reset Reply



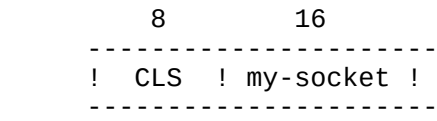
The RRP command must be sent in reply to an RST command.

RFC - Request for Connection



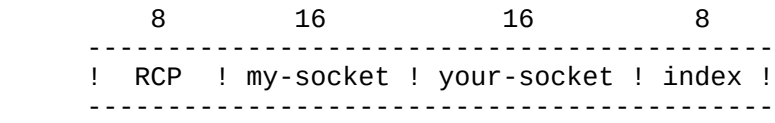
The RFC command is used to establish a connection. The "my-socket" field specifies the socket local to the Host transmitting the RFC; the "your-socket" field specifies the socket local to the Host to which the RFC is transmitted. The "index" field specifies the index value which will be given in bits 65-72 of each data message sent from "my-socket" to "your-socket". The "size" field specifies the maximum number of 8-bit bytes which may be transmitted in any single message from "your-socket" to "my-socket". The "credit" field specifies the size of the initial sequence number window (in the range 0-7) in the "your-socket" to "my-socket" direction.

CLS - Close



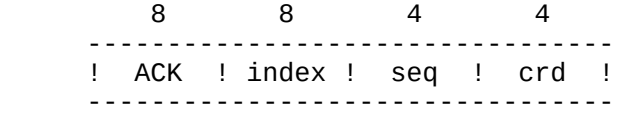
The CLS command is used to terminate a connection. The connection need not be completely established before CLS is sent.

RCP - Re-Initialize Connection Parameters



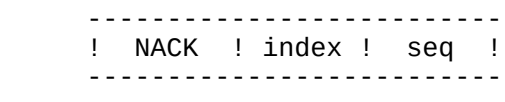
The RCP command is used by one Host to inform another that all information regarding a possibly previously-existing connection between "my-socket" and "your-socket" AND all information regarding a possibly previously-existing connection identified by "index" (between these Hosts) should be purged from the tables of the Host receiving the RCP. The "my-socket", "your-socket", and "index" fields are defined as in the RFC command.

ACK - Acknowledgement



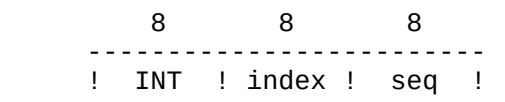
The ACK command may be used to acknowledge received data, or to assign credit, without sending a data message. The value in the index field identifies the data connection which uses the same index value (in the direction from the sender of the ACK to the receiver of the ACK). The eight bits following the index field (the "seq" and "crd" field) have the same meaning as bits 97-104 of the data message identified by the index value.

NACK -- Negative Acknowledgement



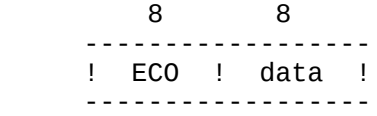
The NACK command informs the receiver of the NACK that it should immediately retransmit the data message identified by the remaining fields. The index field is defined exactly as for the ACK command. The "seq" field gives the 4-bit sequence number (right-justified) which should be immediately retransmitted. Note that the data message to be retransmitted does not have an index value equal to "index", but instead is transmitted over the other direction of the data connection which the Host sending the NACK identifies by "index". No Host is ever required to transmit or act upon a NACK command; however, use of the NACK may occasionally permit a decrease in retransmission delay.

INT - Interrupt



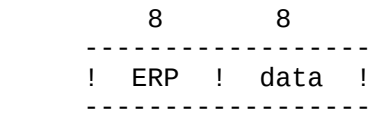
The INT command is sent over the control link to provide an "out-of-band" (and hence not subject to flow control) signal for the data connection denoted by the index field. The index value is the value which would appear in bits 65-72 of a data message sent from the sender of the INT command to the receiver of the INT command. The means of synchronizing this signal with the data being transmitted over the data connection is the inclusion of a 4-bit sequence number (right-justified) in the "seq" field. The number specified by this field denotes the first data message which "follows" the out-of-band signal.

ECO - Echo Request



The ECO command is used only for test purposes. The data field may be any bit configuration convenient to the Host sending the ECO command.

ERP - Echo Reply



The ERP command must be sent in reply to an ECO command. The data field must be identical to the data field in the incoming ECO command.

Opcode Assignment

Opcodes are defined to be 8-bit unsigned binary numbers. The values assigned to opcodes are:

NOP = 0

INT = 1

RFC = 2

CLS = 3

ACK = 4

NACK = 5

RCP = 6

RST = 7

RRP = 8

ECO = 9

ERP = 10