Network Working Group
Request for Comments # 54
June 18, 1970

Steve Crocker (UCLA)
Jon Postel (UCLA)
John Newkirk (Harvard)
Mike Kraley (Harvard)

An Official Protocol Proffering

I. INTRODUCTION

As advertised in NEW/RFC #53, we are submitting the protocol herein for criticism, comments, etc. We intend for this protocol to become the initial official protocol, and will, therefore, be happiest if no serious objections are raised. Nevertheless, we will entertain all manner of criticism until July 13, 1970, and such criticism should be published as a NWG/RFC or directed to the first author.

After July 13, a decision will be made whether to adopt this protocol (or slight variation) or whether to redesign it and resubmit it for criticism.

Only the Protocol

In preceding discussions of protocol, no clear distinction has been made between the network-wide specifications and local strategies. We state here that the only network-wide issues are message formats and restrictions on message content. Implementation of a Network Control Program (NCP) and choice of system calls are strictly local issues.

This document is constrained to cover only network-wide issues and thus will not treat system calls or NCP tables; nevertheless, a protocol is useless without an NCP and a set of system calls, so we have expended a great deal of effort in deriving a protypical NCP. This effort is reported in NWG/RFC #55, and the reader should correlate the protocol presented here with the suggestions for using it presented there. It is important to remember, however, that the content of NWG/RFC #55 is only suggestive and that competitive proposals should be examined before choosing an implementation.

Flow Control

In the course of designing this current protocol, we have come to understand that flow control is more complex than we imagined. We now believe that flow control techniques will be one of the active areas of concern as the network traffic increases. We have, therefore, benefitted from some ideas stimulated by Richard Kaline and Anatol Holt and have modified the flow control procedure. (Defects in our scheme are, of course, only our fault). This new

procedure has demonstrable limitations, but has the advantages that it is more cleanly implementable and will support initial network use. This is the only substantive change from the protocol already agreed upon.

The new flow control mechanism requires the receiving host to allocate buffer space for each connection and to notify the sending host of how much space in bits is available. The sending host keeps track of how much room is available and never sends more text 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, increased by control commands sent from the receiving host, and decremented by the text length of any message sent over the connection. The sending host is prohibited from sending text longer than the value of the counter, so the counter never goes below zero.

Ideally, the receiving host will allocate some buffer space as soon as the connection is established. The amount allocated must never exceed what the receiver can guarantee to accept. As text arrives, it occupies 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.

The control command which allocates space is

ALL <link> <space>

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

This formulation of flow control obviates the RSM and SPD commands in NWG/RFC #36, and the Host-to-Imp message type 10 and Imp-to-Host message types 10 and 11 in the current revision of BBN Report 1822.

The obvious limitation in this scheme is that the receiving host is not permitted to depend upon average buffer usage -- worse case is always assumed. If only a few connections are open, it is unlikely that there would be much savings. However, for more than a few connections, average buffer usage will be much less than allocated buffer space. We have looked at extensions of this protocol which would include adaptive allocation, and we believe this to be feasible. For the present this limited scheme seems best, and we

look forward to discussing more sophisticated schemes later. The old scheme of special RFNM's, etc. also remains under discussion.

In order to answer questions and discuss details, we will hold a pair of network meetings. The first will be on June 29 at Harvard and the second on July 1 at UCLA. We request that no more than on programmer per host attend a meeting and that hosts be represented at only one of these meetings. Two of us (J.N. and S.C.) will be at both meetings.

To make reservations to attend the Harvard meeting, contact

Mrs. Margi Robison
(617) 495-3989
or 495-3991

To make reservations to attend the UCLA meeting, contact Mrs. Benita Kirstel (213) 825-2368.

II. THE PROTOCOL

The notion of a connection as explained in NWG/RFC #33 pervades the protocol. A connection is a simplex communication path, intended to be between two processes.

The primary function of the protocol is to provide for
(1) establishment of connections,
(2) regulation of flow over connections, and
(3) termination of connections.

In addition, the protocol provides some ancillary functions such as sending simulated interrupt pulses and echoing test messages.

To provide a path for exchanging information about connections, we designate specific links, i.e. link one between each pair of hosts to be control links. Traffic on control links consists only of control commands, defined below.

Connections are named by a pair of sockets. Sockets are 40 bit names which are known throughout the network. Each host is assigned a private subset of these names, and a command which requests a connection names one socket which is local to the requesting host and one local to the receiver of the request.

Sockets are polarized; even numbered sockets are receive sockets; odd numbered ones are send sockets. One of each is required to make a connection.

To facilitate transmission of information over a connection, a unique link is assigned to each connection. One of the steps in establishing a connection, therefore, is the assignment of a link. Of the non-control links, zero is reserved for intra-network use, and links 32 to 255 are reserved for experiment and expansion. Thus only links 2 through 31 are available for regular use. Link assignment must either always be done by the receiver or always by the sender. We have (almost) arbitrarily chosen this to be the receiver's responsibility.

All regular messages consist of a 32 bit leader, marking, text, and padding. Marking is a (possibly null) sequence of zeroes followed by a 1; padding is a 1 followed by a (possibly null) sequence of zeroes.

A regular message sent over the control link (link 1) is called a control message. Its text is an integral (possibly zero) number of control commands in the form described below, and this text must end on a command boundary.

The commands used to establish a connection are STR and RTS. The STR command is sent from a prospective sender to a prospective receiver. Its <my socket> field contains a send socket local to the prospective sender; its <your socket> field contains a receive socket local to the prospective receiver. The RTS command is the dual, but is also contains a <link> field for link assignment. These two commands are referred to as requests-for-connection (RFC). A STR and an RTS match if the <my socket> field of one is identical to the <your socket> field of the other and vice versa. 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. Therefore, a host may not use a socket for the <my socket> field of an RFC if that socket is mentioned in a previous RFC and the connection is not yet terminated.

The command used to terminate a connection is CLS. 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. More details on termination are given below.

After a connection is established, the receiving host sends a ALL command which allocates space for the connection. The sender keeps track of how much space is available in the receiving host and does not transmit more text than the receiving host can accept, as explained above. A sender is also constrained by the local IMP from sending a message over a connection until the RFNM from the previous

message is received.

After a connection is established, CLS commands sent by the receiver and sender have slightly different effects. CLS command 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.

CLS commands sent by the receiver act as demands on the sender to terminate transmission. However, since there is a delay in getting the CLS command to the sender, the receiver must expect its buffers to fill to the limit provided in ALL commands.

While a connection is established, either side may send INR or INS commands. The interpretation of these commands is a local matter, but in general they will provide and escape function.

Note that the ALL, INR and INS commands may be sent only after the connection is established and before a CLS command is sent.

A very simple test facility is provided by the ECO and ERP commands. Upon receiving a ECO command, a host must change the first eight bits to ERP and return it. These commands have no relationship to connections.

A NOP command is included for convenience. It is coded as zero to facilitate command message construction.

Finally, an ERR command is included for notifying a foreign host it has (apparently) made an error. At present, no specific list of errors is defined, and no action is defined for the receipt of ERR commands. Hosts should log ERR commands upon receipt so that system programmers can diagnose the trouble. A host may generate an ERR command at any time and for any reason, but it is advised that each host publish an exhaustive list of the ERR commands it may sent and their interpretations.

NETWORK CONTROL COMMANDS

The following is a detailed description of the structure and format of each of the control commands.

To facilitate and clarify socket descriptions, the following conventions have been adopted:

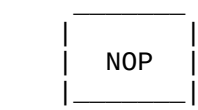
<my socket> and <your socket> are used in the command descriptions.

<my socket> is local to the originator of the command.

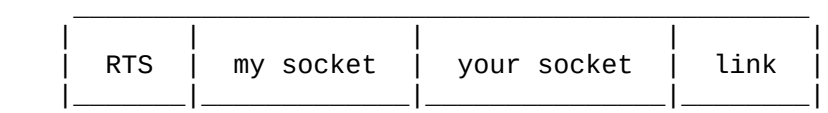
<your socket> is local to the receiver of the command.

CONTROL COMMAND FORMATS

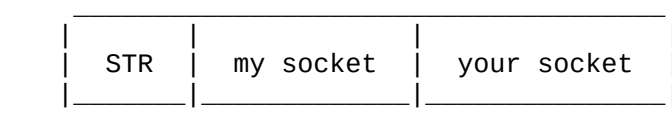
No Operation



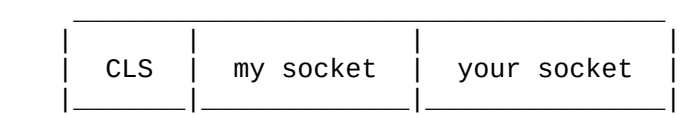
Request Connection, Receiver to Sender



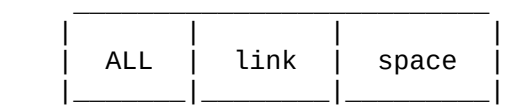
Request Connection, Sender to Receiver



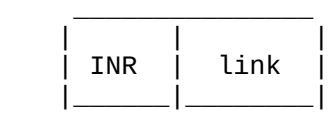
Close



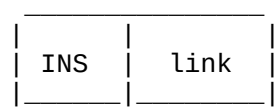
Allocate



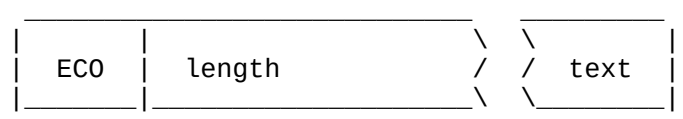
Interrupt Sent by Receiving Process



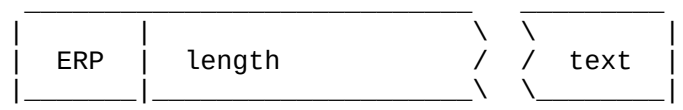
Interrupt Sent by Sending Process



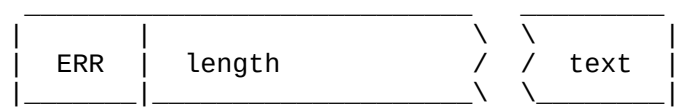
Echo Request



Echo Reply



Error Detected



The host is specified in the leader.

<link> is 8 bits

<space> is 32 bits long and is an unsigned integer.

<length> is an unsigned 16 bit integer.

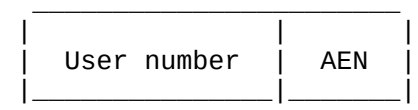
<text> is as long as the length. The command is therefore 24 bits longer that the length. Maximum length is one message, to facilitate command decoding and manipulation.

All control command codes are 8 bit long:

NOP = 0
RTS = 1
STR = 2
CLS = 3
ALL = 4
INR = 5

INS = 6
ECO = 7
ERP = 8
ERR = 9

<my socket> and <your socket> are 32 bits long,



24 bits for user number and 8 bits for AEN.

III. Conclusion

Extensions to the Protocol

Some issues have not been adequately treated in the current protocol. We have in mind the following topics to consider more thoroughly and perhaps experiment with.

1. More Sophisticated Flow Control.

As mentioned above, other schemes for flow control are still being considered. Other than the necessity of providing some form of it, we are completely unsure of the nature of the problem. It may turn out that the present scheme is completely adequate; it may also turn out that we will need a much more complex scheme.

2. Error Detection and Recovery

As we gain some experience with the network, we will develop a better understanding of what errors can occur and, perhaps more importantly, what to do about these errors. We expect the protocol to change as we understand error control.

3. Start Up and Shut Down Procedures

We have not done enough thinking about the problem of the host which participates part-time in the network, which ceases normal network operation but remains on the network for special purposes, or which recovers from a system failure. These issues are critical to robust network operation and are possibly our highest priority. 4. Query and Response

A host-to-host status test would be a valuable tool, but it is not yet clear what is appropriate to provide.

Coming onto the Network

We suggest that hosts come onto the network gingerly. First, each host should thoroughly exercise connections to itself. Then it should arrange experiments with some other host who is already functioning. Finally, it may begin to exercise the facilities of other hosts. It is not clear at this time which host will be in the best position to help other hosts first, but UCLA will attempt to serve this function.

Private Networking

A common ploy is to use the IMP to connect several local computers, one or more of which is not available to the whole network. For example, Harvard is connecting its PDP-1 to its PDP-10 via an IMP; Lincoln Laboratories is connecting its TSP to the 360/67 and the TX2 via an IMP; and UCLA is similarly connecting a XDS 920 to its Sigma-7. In each of these cases, the small machine will not initially provide services to the network.

Although there should be no unwanted traffic to any of these extra hosts, it is desirable that they conform minimally to the network protocol. Provided that they never initiate a connection or send out spurious control commands, it is sufficient for a host to respond to CLS commands with acknowledging CLS commands, and to respond to ECO commands with ERP commands.

Acknowledgments

The work presented above is only a small portion of the concurrent effort. Most of the related effort will be reported in immediately forthcoming RFC’s. A number of people provided extremely valuable aid during the last two weeks. We are particularly grateful to Professor George Mealy of Harvard for supporting four of his students to come westward to work on the network, to Robert Uzgalis for facilitating access to CCN at UCLA, and to the secretarial staff of the Computer Science Division of the University of Utah, and especially Peggy Tueller and Marcella Sanchez, for excellent help in preparing these documents.