Comments on Host-Host Protocol Document No. 1 (S. Crocker - 8/3/70)

Page 3.   Eliminate marking. Instead, make all regular messages into
two message: The first containing just the leader and indicating that
the data follows in the second (next) message. Do this both from the
source Host to its IMP and from the destination IMP to its Host.
Thus, no more hunting for the beginning of the data is necessary.
Once this adjustment is made, an additional simplification is
available. If the maximum message length is a common multiple of the
word sizes of all the computers in the network (perhaps 2880*2 bits),
successive messages of long files can be dropped in place without
shifting.

Page 4.   Control messages should be sent to and from the _control
socket_ -- not over the control link. The concept of the control
link causes a great big, unnecessary special case.

Page 5.   Assigning sockets permanently to certain network resources
should be encouraged and a directory of the socket/resource
associations should be available somewhere in the network, perhaps in
physical book form at each site.

Page 6. Links have no Host-Host purpose other than identifying a
connection so that socket numbers don't have to be included in all
messages and to simplify table look-ups in the NCPs. However, since
there are possibly 512 links* with the same number, links don't aid
table look-ups very much. Also finding the next available link to a
particular destination is very ugly . Therefore, I suggest limiting
the number of links to a total of n (where n = 32, 64, or 256 or some
other good number) for all destinations. In other words, a
particular link is only in use to one destination at a time(actually
from one destination at a time since the receiver picks the link to
be used for a connection). This change makes picking the next
available link very simple and,I feel,is a worthwhile change if only
for this reason. The question of simplifying table look-ups is a
little more complex. It is easy to use the link directly as an index
into tables in the receive portion of the NCP since the receiver
picks the link. But a hash table or linear search or something is
still necessary in the send portion of the NCP. This too can be
fixed with the following changes. Add to STR a _pseudo link_ chosen
by the sender. This link is sent in all non-control messages in the 8
----
*A destination number is 9 bits.

bits to the right of the link in the leader. The IMP must preserve
these bits and return them with RFNMs and the receiver must use the
pseudo link instead of the link in RET and INR. The extra memory
necessary to store the pseudo link in the NCP receive tables (which
are indexed by link) and the link in the NCP send tables (which are
indexed by pseudo link) is certainly less than the overhead necessary
to maintain associative tables.

Page 8.   The allocate mechanism seems very inconvenient for the
receive portion of the NCP to use. The receiver wants the allocation
to be used up in units of the receiver's buffer size not in units of
sender messages which may be variable length. Otherwise the receiver
has a memory compaction problem.

Page 9.   The new irregular message to make the "cease" mechanism
work are unnecessary, I think. The sender can keep track (probably
with a one bit counter) of ALLs and GVBs and ignore GVB 0s for which
resume ALLs have already arrived.   This the receiver need not know
whether the cease has been sent or not.

Page 15. If I implemented an NCP, all ERRs would be treated like
NOP. As an error control mechanism ERR is complicated and
insufficient. Who wants to debug a complicated mechanism which only
catches bugs due to the primary mechanism being undebugged. The one
error control mechanism I would provide is a receive process to send
process acknowledgment on every message. If this is not received for
too long, the send process can send the message again if it has been
saving it. This acknowledgment catches errors causing message loss
at the process/NCP, NCP/NCP, Host/IMP, IMP/IMP, etc. levels.
Currently the Host/IMP interface is particularly lacking in useful
error controls. I wouldn't worry about kinds of errors check-sums
are designed to pick up. If dropped and picked up bits ever become a
problem either add hardware to more interfaces or let the receive
process not send the process to process acknowledgment if a software
checksum does not check.

The page 3 and page 6 comments involve a change to the IMP program.
I feel a tiny bit guilty suggesting changes I don't have to implement
any more. However, I trust Crowther and Cosell will, as always,
resist bad changes while making sensible ones. The page 9 comment is
aimed at avoiding a change in the IMP program.

        [This RFC was put into machine readable form for entry]
          [into the online RFC archives by Luke Hollins 8/99]