J. Davidson
BBN

IEN 62
Section 2.3.3.20
27 October 1978

ENRICHED INTERNET ADDRESSING OF ARPANET RESOURCES

- AN INTERIM PROPOSAL -

This note discusses a problem which has arisen within the ARPANET because of the way in which we've formatted the 24-bit <REST> component of the internet source and destination address fields. The problem was pointed out to me by Jack Haverty who expressed his concern relative to TCP. The problem is that the syntax we have chosen is not rich enough to allow addressing of all the kinds of resources which may be available on a given ARPANET host. In particular, it does not allow for addressing more than a single protocol module of a given type: to use Jack's example, it does not allow for addressing each of two TCP programs on a given host.

(It is fairly clear that we would like to sometimes be able to have more than one TCP program running on a given host -- for example in cases where a transition from one version to another is being made and both versions must be available simultaneously for some period, or where testing of a new implementation is being done, or where both secure and non-secure versions of the same protocol may be needed, etc. Note too that to hosts outside the ARPANET, an inside-the-ARPANET port expander, which allows several hosts to share a single ARPANET address, appears just as a single host which has multiple TCPs running on it.)

Here is the problem in more detail. The <REST> field for the ARPANET is defined to be 16 bits of IMP address followed by 8 bits of host address. This format is particularly suited for gateways which have to map internet addresses into ARPANET addresses for construction of an ARPANET leader. Unfortunately, the format is not so well suited for addressing the various protocol modules at the next level down within the designated host. (And, as stated before, it is not even useful in selecting one of the several hosts hanging off a port expander.) To account for this, we added the <PROTOCOL> field in the internet header. Right off, this suggests that the <PROTOCOL> field is really an address component which has been perhaps wrongly positioned (in the header) with respect to the address it qualifies.

The <PROTOCOL> field has not been setup as a pure address component, however. Its value is defined in a internet-wide fashion. The benefit of this is that in order to address an "official" instance of a given protocol module in any internet host, a source program has only to plunk the official value into the <PROTOCOL> field. This presumably is a lot easier than doing a table lookup to determine what the <PROTOCOL> value should be, depending on the name of the destination host. On the other hand, the fact that the value is internet-wide prevents a particular host from having its own "unofficial" versions running in parallel with (or even instead of) the official protocol module, because they aren't allowed to specify their own <PROTOCOL> values.

There are several ways we might get around these problems. An easy way would be to make the IMP and HOST fields in <REST> be a little shorter and thus free up some of the 24 bits for use as "address qualifiers" that are interpreted only by the destination host. The gateways would have to be a little smarter, but not much, in order to extract the smaller IMP and host specifications and extend them before placing them in the ARPANET leader.

A second way to get these address qualifiers is to change the ARPANET IMP to not use all 16 bits of the IMP or all 8 bits of the host specifications. Then the gateway need not change.

These address qualifiers could be used in several ways. The one which affects the use of the <PROTOCOL> field the least, is to consider them to designate "pseudo-hosts" at the destination. (This view is consistent with the notion of an internal gateway within each internet host, and also adheres to the requirement of the Internet Protocol Specification that a given physical host be able to look like several distinct (logical) hosts.) Each pseudo-host could support a single instance of each "official" protocol. Pseudo-host zero would support the true internet-wide version, while the others could support the experimental ones, say.

This solution then allows one real host to have multiple instances of a given protocol module. Some special partitioning of the pseudo-host numbers would be needed in order to allow each of the hosts on a port expander to also have multiple instances of a protocol module, but this is easily legislated by the owners of the port expander. (When a host has a new protocol module available for general use, it must publish its address: each host which wants to use the new module may want to provide its users a name-to-address lookup to enable them to retrieve the new address. These are considerations of some importance to implementers, but not to this paper.)

Interestingly, it appears that the ARPANET people may step in and save us from this problem even if we do nothing about it. (In view of this, the proposals made above may simply be though of as interim solutions.) In late spring 1979, they intend to implement a number of enhancements to their host addressing capabilities such as broadcast addressing, multi-homing, and logical addressing. With logical addressing, several ARPANET-style addresses will map into one physical host address. Thus, if our internet <REST> field specifies an ARPANET logical address, then a number of "pseudo-hosts" can be identified at the physical host automatically.

Well, not quite automatically. First, each host which wants to have multiple instances of a single protocol will have to implement ARPANET logical host addressing. This means that it must learn to use the new ARPANET leaders which are required for logical addressing. This may involve a significant programming effort. In addition, it is not clear that 24 bits will be sufficient to specify a logical host address (or a group address, later on); in this event, gateways may have to be changed to map from the 24-bit <REST> specification to the longer logical host specification. This is tougher than simply doing the field expansion required to implement pseudo-hosts under the current scheme. (However, I doubt that more than 24 bits will really be needed for any of these new addressing modes.)

I think the important point to be made here is simply that we should make sure to provide a little richer addressing capability in our 24-bit <REST> field than we have to date. This goes retroactively for the ARPANET, but is a concern for all other nets as well. We should also decide whether the use use of pseudo-hosts is a satisfactory mechanism for supporting multiple copies of a given protocol module on a single host.