Ann - Killeach

IEN 188

ISSUES IN INTERNETTING

PART 3: ADDRESSING

Eric C. Rosen

Bolt Beranek and Newman Inc.

June 1981

Bolt Beranek and Newman Inc.
Eric C. Rosen

ISSUES IN INTERNETTING

PART 3: ADDRESSING

3. Addressing

This is the third in a series of papers that discuss the issues involved in the design of an internet. The initial paper was IEN 184, familiarity with which is presupposed.

In this paper, we will deal with two basic issues. The first has to do with the Network Access Protocol. It is concerned with the sort of addressing information which a source Host has to supply, along with its data, to a source Switch (gateway, in the Catenet context), in order to enable the Switch to get the data delivered to the proper destination Host. The second issue has to do with the question of how the Switches (both source Switch and the intermediate Switches) are to interpret and act upon the addressing information supplied by the source Host. We begin by stating generally the sort of addressing scheme we envision (which is by no means original), and by comparing it to the very different sort of addressing currently in use in the Catenet. Next we will discuss some of the issues and details that arise in considering how to make such a scheme work reliably. We will then show how this scheme lends itself quite naturally to the solution of certain problems which are very difficult to handle in the current Catenet architecture. Although addressing and routing are rather intimately bound up,

we will avoid routing considerations here whenever possible. Routing in the internet will be the topic of a longer paper which will be the next to appear in this series.

3.1 Logical Addressing / Flat Addressing

For maximum flexibility and robustness of operation, a source Host should be able to simply "name" the destination Host it wants to reach, where a "name" is just an arbitrary identifier for a Host. That is, the source Host should not need to know anything about the physical location of the destination Host, NOT EVEN WHAT NETWORK IT IS ON. In other words, the internet should have logical addressing. The advantages of logical addressing are thoroughly discussed in IEN 183, and that discussion shall not be repeated here. IEN 183 presents a logical addressing scheme which was designed with the ARPANET in mind. However, since we regard the internet as a Network Structure whose Switches are gateways and whose Hosts are generally multi-homed to the gateways, most of the ideas presented in IEN 183 can be carried over directly to the internet environment. The present IEN will emphasize those aspects of the logical addressing scheme which are specific to the internet environment, but the proposed scheme is basically the same as the one discussed in IEN 183. Anyone with a real interest in these issues will want to become familiar with that document.

The basic idea of logical addressing is that a source Host should name the destination Host, and the Switches should map that name into a physical address that is meaningful within the Network Structure of the Switches. The mapping between names and (physical) addresses will, in general, be many-many. That is, one name may refer indeterminately to several distinct physical addresses, either because some one physical machine is multi-homed, or because the user does not care which of several physical machines he reaches. Similarly, one physical machine may have several names, which may either be synonyms, or may be used for further multiplexing within the destination Host. (This may be particularly important when a Host within one Network Structure is really a Switch, e.g., a port expander or local network, within another.)

Logical addressing tends to result in a flat addressing space, rather than a hierarchical one. This may seem surprising in the context of the internet, since an internet is a hierarchical structure, and internet routing is almost certainly going to be some form of hierarchical routing. However, it simply does not follow that the addressing space used in the internet Network Access Protocol must be a hierarchical addressing space. In fact, since the form of the addressing space has an effect on the Network Access Protocol, and hence on Host-level software, whereas the routing algorithm is a purely internal matter to the Network Structure, proper protocol

layering would seem to require that the form of the addressing and the form of the routing be independent. We would like to be able to change the internal routing algorithm of the Network Structure without requiring corresponding changes in Host software, i.e., without changing the form of the addressing.

What we are proposing is quite different from the way addressing is done in the current Catenet Network Access Protocol, IP. IP uses both physical addressing and hierarchical addressing. (Note that physical addressing within a hierarchical Network Structure will almost certainly be hierarchical addressing, whereas logical addressing allows the internal structure of the Network Structure to be better hidden from the users. This is one of its main advantages.) The first component of the address is a network number, and the second component is a physical address which is meaningful within that network. In IEN 183, we discuss a number of reasons for the superiority of logical over physical addressing. Other criticisms of the Catenet's current addressing scheme have been voiced by other authors. For example, the way in which hierarchical addressing is incorporated into Catenet addressing mechanisms has recently come under criticism in IEN 177 by Danny Cohen, who focuses his criticism on the particular case of the ARPANET. His main criticism is that it does not allow enough hierarchical levels. That is, with the presence of local nets or port expanders which appear to the ARPANET as Hosts, there is really another level of

hierarchy after the ARPANET. He suggests, therefore, that ARPANET addressing (1822-level) be changed to provide this additional hierarchical level, and that end-users (or at least Host software modules) fill in this additional level.

It is not obvious, though, that a single additional level of addressing will do for all applications. If we are sending data not just to a local net, but to an internet of local nets, maybe several additional levels of hierarchy are needed. We may also need more hierarchy on the "front end" of the address. A protocol which begins the internet address with a field which is supposed to identify the destination network (e.g., IP) assumes that there is no need to establish a hierarchy among the networks themselves. (This is equivalent to assuming that all Switches can "know about" all networks.) As long as we have only a small number of networks, it may be reasonable enough to assume that destination network addresses need not themselves be hierarchical. However, it is not difficult to imagine a very large internet composed of thousands of networks, where before specifying a network, we must first specify, say, a continent. So maybe our protocol for hierarchical addressing needs a "continent address" field before the network address field. It begins to look as if the addressing structure needs to be INFINITELY EXTENSIBLE in both directions. In fact, in IEN 179 Cohen proposes a scheme which seems intended to provide this sort of infinite extensibility. That seems both an inevitable

consequence of hierarchical addressing, and a reductio ad absurdum of it.

It is also worth noting that a given number of Hosts can generally be addressed with fewer bits in a flat addressing scheme than in a hierarchical addressing scheme. Given, say, 32 bits of addressing, flat addressing can represent 2**32 Hosts. However, if these 32 bits are broken into four 8-bit fields, hierarchically, fewer Hosts can be represented, since in general, not every one of the four fields will actually take on the full 256 values. Inevitably, one finds that at least one field must take on 257 values, while at least one other turns out to have a smaller number of values than expected. This tends to lead to the feeling that the address field needs "just one more level" of hierarchy. It also tends to lead to the use of funny escape values and multiplexing protocols so that different fields can be divided up in different ways by different applications. The same problems usually reappear, however, in a few years, as the need for "just one more level" is proclaimed yet again. Yet the alternative of making the address fields arbitrarily long, hence infinitely extensible, is rather infeasible, if bandwidth considerations are taken into account.

The need for infinite extensibility at the Host interface can be avoided by using logical addressing (although this is only one of its many advantages). We can then identify a single Host

by using a single, structure-less, unique name which is meaningful at each level of internet hierarchy. That is, the Switches at each level of the hierarchy would be able to recognize the name, and to map it into a physical address that is meaningful at that level of hierarchy. Neither the end-user nor the source Host would be responsible for determining the physical addresses at each level of a never-ending hierarchy. Of course, neither these arguments, nor those of IEN 183, can be regarded as finally settling the "flat vs. hierarchical" issue. In networking, no one issue can ever be settled in isolation, and attempts to do so result only in endless and unproductive arguments. A network (or internet) is a whole whose performance and functionality result from the combination of its protocols, addressing schemes, routing algorithm, hardware and software architecture, etc. Particular addressing schemes can only be judged when it is seen how they actually fit into particular designs. The only real argument in favor of a particular addressing scheme is that it fits naturally into a network architecture which provides the needed functionality and performance. It is hoped that the addressing scheme we propose will be judged as part of the architecture we are developing in this series of papers, rather than in isolation.

3.2 Model of Operation: An Overview

The model of operation we are proposing is as follows. A source Host submits a packet to a source Switch, naming (not addressing) the destination Host. THE SOURCE SWITCH THEN TRANSLATES (OR MAPS) THAT NAME INTO A PHYSICAL SWITCH ADDRESS WHICH IS MEANINGFUL WITHIN ITS OWN NETWORK STRUCTURE; THAT WILL BE THE ADDRESS OF THE DESTINATION SWITCH WITHIN THAT NETWORK STRUCTURE. The data is then routed through the Network Structure to the destination Switch so addressed. The name (logical address) of the destination Host is also carried through the Network Structure along with the data and the physical address of the destination Switch. When the destination Switch receives the data, it forwards it to the destination Host over (one of) its Pathway(s) to that Host. If the Pathway is itself a network or internet configuration with logical addressing, the name of the destination Host is passed on via the Pathway Access Protocol. If logical addresses or names are not unique across all component networks of an internet, translation from the internet logical address to the Pathway logical address would have to be done at this point. If the network or internet underlying the Pathway does not even have logical addressing, the Host name will have to be translated into a Pathway physical address by the destination Switch.

Note that, at any particular hierarchical level (i.e., within any particular Network Structure), the ADDRESSABLE ENTITIES are the Switches at that level (which are physically addressed), and all the Hosts (which are logically addressed, or named). Component networks of the internet are treated as structure-less Pathways, AND NEITHER THE COMPONENT NETWORKS THEMSELVES NOR THE SWITCHES OF THE COMPONENT NETWORKS ARE INDEPENDENTLY ADDRESSABLE. Furthermore, a name (logical address) which adequately identifies the destination Host is present at each level of the hierarchy. Of course, a particular name only needs to be unique at a single level of the internet hierarchy, within a particular Network Structure. The names can change as we travel up and down the hierarchy of Network Structures that make up the internet.

3.3 Some Issues in Address Translation

In order to do the sort of translation from logical to physical address that we have been discussing above, the Switches must have translation tables. Many of the issues involved in the design of a robust translation table mechanism are discussed in IEN 183, and much of that discussion applies without change to the internet. We will confine our discussion here, therefore, to issues which are not considered in that note, or which are more specific to the internet environment.

The main problem with the model of operation we have proposed is a very mundane one, but unfortunately a very important one. If there may be thousands of Hosts on an internet, each one with an unlimited number of different names, and if a source Switch must be able to map any name to the address of a destination Switch, then each Switch will have to have a very large table of names to drive this translation function. By itself, this is not much of a problem. To be sure, in the past, it has been considered important to keep the gateways as small as possible. It now seems to be more generally accepted that the current Catenet gateways provide inadequate performance, and that building a robust operational internet system requires us to build Switches that are large enough to handle the required functionality at a reasonably high level of performance. We would expect Switches built in the future to be much larger than the current gateways are. However, it is one thing to require large tables, and quite another thing to require tables which may grow without bound. Since the number of Hosts on the internet may grow without bound, it does not seem feasible to require the Switches to have tables with one or more entries for each and every Host in the internet.

If we cannot fit the complete set of translation tables into each Switch, a natural alternative is to turn the tables into a DISTRIBUTED DATA BASE, with each Switch having only a subset of the complete set of tables. For each Switch, there would be a

subset of logical addresses for which the Switch would have complete physical addressing information. These logical addresses would fall into one of two classes:

1) Those logical addresses which refer to Hosts which are homed (in some Network Structure) directly to that Switch.

2) Those logical addresses which refer to distant Hosts which are in FREQUENT communication with the Hosts which are directly homed to that Switch.

The logical addresses in these two classes are the ones for which the Switch will be most often called upon to do logical-to-physical address translation, and for best efficiency, the information needed to do the translation ought to be present in the Switches. For other logical addresses, which are less often seen, all that is needed is for the Switch to know where the address translation information can be found. Then if a packet with an infrequently-seen logical address is encountered, it can be forwarded to a place where the proper information is known to reside, or else the packet can be held while the information is obtained. (We may want to have a scheme which is a hybrid of these two alternatives. For example, packets with logical addresses that are not contained in the resident tables can be forwarded to a place with more addressing information, and this can in turn cause the needed addressing information to be

sent back to the source Switch, so that additional packets with the same address can be handled directly by the source Switch. That is, the source Switch might maintain, in addition to its permanently resident tables, a cache of the most recently needed addressing information.)

It is important to note that the two classes defined above may vary dynamically, and we may want a procedure for altering the members of those classes in some specific Switch depending upon the traffic that the Switch is actually seeing in real time.

Unfortunately, any such scheme would seem to require the inclusion of at least one additional level of hierarchy in the addressing structure, since when a Switch sees a logical address for which it does not have complete information, it must be able to determine how to get that complete information. The scheme would be self-defeating if it meant that we had to have a table of all the logical addresses, with an indication for each one of which other Switch has the complete information. Rather, we need to be able to group the logical addresses into "areas", of which there will be a bounded number. Then each Switch will be able to keep a table indicating which other Switches contain the complete translation information for each area. This table of areas would then be the only part of the complete set of translation tables that had to be resident at ALL Switches. While this is much more feasible than requiring each Switch to keep a table containing

all the logical addresses, it does means that the destination address provided by the source Host must include not only a destination Host identifier, but also an "area code" for that logical address.

If we are going to organize the logical addresses of all internet Hosts into a relatively small set of "areas", we would like to find some means of organization which is fairly optimal. Unfortunately, there are a number of fairly subtle considerations which make this quite tricky to do. Certain intuitively attractive ways of organizing the internet into these areas will result in various sorts of significant and quite annoying sub-optimalities. Suppose, for example, we treated "area" as meaning "home network", much as in the present Catenet IP (where network number is part of the address that the Hosts must specify.) Then we would require all and only the ARPANET gateways to contain the logical-to-physical addressing information for the ARPANET Hosts, all and only the SATNET gateways to contain the tables for the logical addresses of the SATNET Hosts, etc. The user, in addressing a particular Host, would not only name it, but also name its "home network", and the source Switch would choose some Switch which interfaces directly to the home network of the destination Host from which to obtain the translation information. This method of organization, however, has several unsatisfactory consequences. One problem is that if any Host is on two "home networks", we want the Switches, not the Hosts, to

choose which "destination network" to use. This is necessary if we want the routing algorithm to be able to choose the "best" path to some destination Host, and is really the only way of ensuring that packets can be delivered to a Host over some path, if one of the Host's home networks is down but the other is up. (This is jumping ahead a bit, since a full discussion of the "partitioned net" problem will not appear until section 3.4. The point, though, is that the choice of "home network" to use when sending traffic to a particular destination Host is a ROUTING PROBLEM, NOT AN ADDRESSING PROBLEM. Therefore it ought to be totally in the province of the Switches, which are responsible for routing, and not at all in the province of the Hosts, which must participate in the addressing, but not the routing.)

Another problem arises as follows. Suppose we have adopted the scheme of sending packets for a certain area to a Switch in that area, depending on that Switch to do the further logical-to-physical translation. It is possible that when this further translation is done, we will find that the route which the packet travels from that Switch takes it back through the source Switch. This could mean a very lengthy and delay-producing "detour" for the packet. It might at first appear that this is not very likely. If a packet is going to some ARPANET Host, and we send it to some Switch which is directly connected to the ARPANET, surely we have sent it closer to its final destination, not further away. Unfortunately, that

just is not necessarily true. Network partition or congestion may force a packet for an ARPANET Host to travel from an ARPANET gateway to a gateway (or series of gateways) outside the ARPANET, back around (through a potentially long route) to another ARPANET gateway. (Consider the partitioned net and the expressway problems.) In such cases, the Network Structure may already be in a condition of stress which is likely to result in below par performance. We do not want to make things even worse by adding any further unnecessary but lengthy detours just because we cannot keep all the addressing information at the source Switch.

One way of helping to avoid these sorts of problems is to separate the notion of "area" from any physical meaning. The purpose of adding the notion of area to the logical addressing scheme is just to enable us to distribute the data base needed to do logical-to-physical address translation. There is no reason to suppose that the addressing information needed for some particular Host ought to be contained only in Switches that are "near" that Host. That would be a mistake. Rather, the addressing information ought to be somewhere which is "near" the SOURCE Host, not somewhere which is near the destination Host. This maximizes the chances that the necessary address translation will be done as soon as possible after the packet enters the Network Structure. The sooner we do the address translation, the more information we have which we can make use of to improve the routing of the packet, and the less likely any unnecessary detours will be.

One might think that at least Hosts which are on the same home network should be grouped into the same area. This will work until the first time a Host is moved from one network to another. Since the area codes are given by the individual Host or user as part of the address in the Network Access Protocol of the internet, changing a Host's area code would involve changing Host-level software or tables, which has to be avoided. (Avoiding the need to make such changes when Hosts move physically is one of the main reasons for using logical addressing.) So we really have to think of "areas" as random collections of Hosts.

What we are proposing is a truly distributed logical address translation table, rather than a scheme where each Switch maintains only local information. To make this more concrete, consider how this might be done in the Catenet. All the information about logical addresses which refer to Hosts on the ARPANET would be contained not only in all the gateways which are directly connected to the ARPANET, but also in a set of additional gateways which are uniformly scattered around the internet. Then, although the addressing information would not be in every potential source Switch, it would be somewhere close to every potential source Switch, and packets would not have to travel a long distance only to find out that they are going in the wrong direction.

3.4 Model of Operation: More Detail

Let's assume that a source Host has given a message to a source Switch, with a logical address and an "area code" indicating the destination Host. If the source Switch does not have the complete address translation information in its tables, it will look in its table of area codes. The given area code will be associated in the latter table with some set of Switches (within the same Network Structure). The sequence of operations that we envisage is the following:

1) The source Switch picks one of these Switches, and sends the message to it. There must be enough protocol between these two Switches so that the chosen Switch knows that it is not the final destination Switch, but only an intermediate Switch, and that it is expected to complete the address translation and then to forward the message further.

2) The chosen Switch must be able to recognize the logical address of the destination Host, and associate it with one or more possible destination Switches. The message will be forwarded to one of these Switches. Furthermore, the addressing information can be sent back to the source Switch where it can be held in a cache in case the message is followed by a flood of additional messages for the same logical address.

In the case where the source Switch does contain complete address translation information for the destination logical address, that logical address will be associated with some set of potential destination Switches. The source Switch will choose one, and send the message directly to it.

Logical-to-physical address translation should be done by only one Switch; either the source Switch or the Switch chosen by the source Switch on the basis of the area code. There is no need to allow intermediate Switches to do any logical-to-physical address translation. (There is only one exception to this, namely the case where a message arrives at an intermediate Switch only to discover that the destination Switch chosen by the source Switch is no longer accessible. In this case, re-translation is the alternative to dropping the message entirely.) Remember that many Hosts will be multi-homed (in the internet, virtually every Host is multi-homed, since most networks will have at least two internet gateways connected to them), so that there will in general be more than one possible destination Switch. By prohibiting re-translation at intermediate Switches, we avoid the problems of looping that might arise if different intermediate Switches make different choices of destination Switch. As we shall see, this also simplifies our approach to the partitioned net problem, and at any rate, there is no great advantage to allowing intermediate Switch translation (cf. IEN 183).

We suggested above that if a source Switch does not recognize a particular logical address, and hence must send a message to another Switch (as determined by the area code), the latter Switch should send the addressing information back to the source Switch, to be kept temporarily in a cache. We have to emphasize "temporarily." The source Switch should time out the addressing information which it keeps in the cache, and then discard it. If it later receives from any of its source Hosts any subsequent messages for the same destination logical address, it will have to reobtain the information. The reason for this is that it will be necessary, from time to time, to change the translation tables. It is not that hard to develop an updating procedure which ensures consistent updating of all Switches where the information about a logical address normally resides. But it might be more difficult to develop a procedure which ensures consistent updating of all the temporary (cached) copies of that information. Timing out the temporary copies of the addressing information will prevent out-of-date information from being preserved in inappropriate places. (Though the use of an out-of-date translation is not so terrible, since it would elicit a DNA message, rather than causing mis-delivery of data. See IEN 183 for details. In this sense, out-of-date information is self-correcting.)

When either a destination Host name (logical address) or an area code maps into several Switches, the source Switch must

apply some criterion to choose one from among them, since in general we will want to send only one copy of the message to its destination. (Though there may indeed be cases in which we want to send a copy of the message to each possible destination Switch, in order to increase the reliability of the system, or to be sure that we get the message to its destination Host as fast as possible.) There are several possible criteria that we might consider using:

a) We might always choose the "closest" Switch, according to some particular distance metric (which might or might not be the same distance metric used by the routing algorithm).

b) The list of potential destination Switches might have a "built-in" ordering, so that the first one is always used unless it is down, in which case the second one is always used, unless it is down, in which case the third one is used, etc.

c) If the set of potential destination Switches has the right sort of topological distribution, we might try to round-robin them in order to achieve some sort of load-splitting.

d) If we can obtain some information about the relative loadings of the various Switches, we can try to choose

the one with the smallest load (to try to avoid causing congestion within the destination Switches), or we might try to trade off the increase in load that we will cause at the destination Switch with the distance we have to travel to get there.

e) Certain possible destination Switches might be favored for certain classes of traffic (as determined by the "type of service" field, or by access control considerations). That is, certain destination Switches might be favored for interactive traffic, and certain others (with more capacity?) for bulk traffic. Or there might be administrative access control restrictions which prohibit certain classes of traffic from being sent to certain Switches. (This may be particularly applicable in an internet context where different Switches are under the control of different administrations. It is possible, though, to imagine applications of this sort of access control even in a single-administration Network Structure. For example, we might want to prohibit military traffic from entering certain Switches, in order to preserve capacity for important university traffic.)

f) It is possible to combine some of the above criteria, e.g., choose the closest (i.e., shortest delay) Switch for interactive traffic and the most lightly loaded one for bulk traffic.

Remember that in the internet case, all the Hosts on some network are considered to be homed to all the gateways on that network, so that in general most Hosts will be multi-homed, and the way we select the destination Switch could have a significant effect on internet performance.

Of course, a destination Switch might itself have two or more Pathways to a particular destination Host. Perhaps the Switch is a gateway on two networks, and the Host is also on those two networks. Or perhaps the Switch is multi-homed onto the network of the Host. In such cases, a further choice remains -- the destination Switch must choose which of several possible Pathways to the destination Host it should use for sending some particular packet. Each (destination) Switch, therefore, will have to have a second logical-to-physical address translation table, which it accesses in order to choose the proper Pathway to a destination Host. This second translation table, however, contains information which is only useful locally. In addition to containing information needed to map the logical address onto one of the Switch's access lines, it must also contain any information needed in order to specify the address of the destination Host in the Pathway Access Protocol. In some cases, the logical address of the Host in its "home network" may be the same as its logical address in the internet, in which case no additional information is needed. If this is not the case, or if the "home network" does not have logical

addressing, the local translation tables must contain information for mapping the internet logical address to an ad (logical or phys which is meaningful in the "home network." The issues of choosing one from among a set of possible Pathways according to sriteria are basically the same as those we have been discussing from the perspective of the source Switch, howeve

An inteng little issue: suppose that traffic for Host H can be sent to either Switch A or B, but that the route to Switch B contains Switch A as an intermediate Switch. Does this mean that the traffic should always be sent to A, rather than B? Not necessarily. Perhaps A has plenty of bandwidth available for forwarding traffic to other Switches, but only a little available for sending traffic directly to a Host. Or the Pathway from Switch A to Host H may itself have such a long delay that it is quicker to send the traffic through A to B and then on B's Pathway to H. While it may turn out to be very difficult to take account of such factors, we ought not to rule them out by a priori considerations, and we ought not to design a system in which such factors cannot be considered.

A variant on this issue can arise as follows. Suppose Host H1 wants to send some data to Host H2, and H1 puts this data into the internet by submitting it to source Switch S. Now S will look in its address translation table to find the possible

destination Switches for H2. Let's suppose that there are two such possible destination Switches, one of which is D, and the other of which is S itself. That is, S has a choice of sending the data directly to H2 (over a Pathway with no intermediate Switches), or of sending it to D, so D can transmit it directly to H2. Nothing in the proposed scheme constrains S to choose itself as the destination Switch. If we want, we can have S make the choice of destination Switch without taking any special cognizance of the fact that it itself is a possible destination Switch. Or we might even require that S not choose itself as the destination Switch. That is, when a gateway on the ARPANET, for example, gets some data from an ARPANET Host which is destined for another ARPANET Host, maybe we want the data to be sent through another gateway, rather than just sending it right back into the ARPANET. This possibility might be crucial to solving the "expressway" problem. While we are not at present making any proposals for allowing the internet to be used as an "expressway" between two Hosts on a common, but very slow, network, we are trying to ensure that nothing in our proposed addressing scheme will make this impossible. This is a very important difference between our proposed scheme and the scheme presently implemented in the Catenet, where a source Switch which is also a potential destination Switch is highly constrained to pick itself as the actual destination Switch. Of course, for this to work, there must be enough protocol so that a Switch which receives some data

can know whether it is getting it directly from a source Host, or whether it is getting it from another Switch.

When we say that a particular Host name maps onto a set of possible Switches, what w really saying is that each member of that set of Switches has a Pathway to the Host. Remember the definition of "Pathway" -- a Pathway in Network Structure N between two Switches of Network Structure N or between a Switch and a Host of Network Structure N is a communications path between the two entities which does not contain any Switches of Network Structure N. The logical-to-physical address translation tables will not map a Host name to a particular set of destination Switches unless each of those Switches has a Pathway to that Host. But we must remember that at any particular time, one or more of these Pathways may be down. Before we apply the above criteria (or others) to the set of possible destination Switches in order to choose a particular one, we must first eliminate from the set any Switches whose Pathway to the destination Host is down. This is a non-trivial task which breaks down naturally into two sub-tasks. First, the destination Switch must be able to determine which of the Hosts that are normally homed to it is reachable at some particular time. Second, this information must be fed back to the source Switch. Each of these sub-tasks raises a number of interesting issues.

In IEN 187, we discussed the importance of having a Pathway up/down protocol run between each Host and each Switch to which it is homed, so that a source Host can know which source Switches it has a currently operational Pathway to. Now we see the other side of the coin -- each destination Switch must be able to determine which Hosts it currently has an operational Pathway to. Many of the considerations discussed in IEN 187 apply here too, and need not be mentioned again. Basically, the Switch will have to run a low-level up/down protocol which relies on the network which underlies the Pathway to tell it whether a particular Host is reachable (e.g., the ARPANET returns an 1822 DEAD Reply to any ARPANET source Host which attempts to send a non-datagram message to an unreachable destination Host), and the Switch will also have to run a higher-level up/down protocol whereby it queries the Host and infers that the Host is unreachable if no replies to the queries are received. Of course, if some Pathway consists of a simple datagram-oriented network that provides no feedback to the source, then a higher-level protocol will have to be used alone.

Assuming that the Switches have some way of determining whether their Pathways to particular Hosts are operational, we have the following subsidiary issue -- should these determinations be made on a regular basis, for all Hosts that might be reachable, or should they be made on an exception basis, with the information obtained only as needed? Let's consider the

analogous operation in the ARPANET. In the ARPANET, the up/down status of each Host is maintained continuously, as a matter of course, by the IMP to which that Host is homed. This information, however, is not generally maintained at other IMPs. If a packet for a dead Host (on a live IMP) is submitted to some source IMP, the packet will always be sent to the destination IMP, which will (unless the packet is a datagram) return an 1822 DEAD reply. The source IMP receives the DEAD reply, signals it to the source Host, and then discards the information. IMPs do not maintain status information about remote Hosts, but the information is available to them as they need it (i.e., on an exception basis). On the other hand, each IMP always maintains complete, accurate, and up-to-date information about the reachability of each other IMP. Whenever any IMP goes down or comes up, this information is broadcast to all other IMPs in an extremely quick and reliable manner. If a source Host attempts to send a packet to a Host on an unreachable IMP, no data is sent across the network at all; the source IMP already knows that the destination IMP cannot be reached, and tells the source Host immediately.

Why don't IMPs maintain regular status information about all ARPANET Hosts? It's not as if this is against the law, and under certain conditions, it might be advantageous to do so. However, the more entities about which regular status information is maintained, the more bandwidth (trunk and CPU) and memory must be

devoted to handling the information. With a potentially unbounded number of Hosts being able to connect to the ARPANET, it does not seem feasible for all IMPs to maintain this status information for every Host. Fortunately, it just is not as important to maintain status information for Hosts as it is for IMPs. Status information about the IMPs is necessary in order to do routing, so failure to maintain this information regularly would degrade the routing capability, with a consequent global degradation in network service. Since Hosts, on the other hand, are not used for storing-and-forwarding packets, routing does not have to be so aware of Host status, and global degradations due to incorrect assumptions about Host status are less likely.

If we can't expect ARPANET IMPs to maintain regular status information for each Host, we certainly can't expect internet gateways to maintain regular status information for each and every Host in the internet. In fact, in the internet, the situation is even worse. In the ARPANET, each IMP at least maintains regular status information about the few Hosts to which it is directly connected. This is simple enough to do, since the number of Hosts on an IMP is bounded (barring the introduction of local nets or port expanders) and there are machine instructions to detect the state of the Ready Line. However, we can hardly expect a gateway to maintain regular status information about all the Hosts on all the networks to which the gateway is directly connected. So we will suppose that in general, status

information about the Hosts which are homed to a particular Switch will be obtained by that Switch on an exception basis, as needed. Of course, saying that this will be true in general does not mean that it must be universally true. If there are a few Hosts somewhere that are major servers with many many important users scattered around the internet, there is no reason why the Switches to which those servers are homed cannot maintain regular status information about those few Hosts. If the number of such special Hosts is kept small, this would not be prohibitively expensive, and if these Hosts really do handle a large portion of the internet traffic, this might be an important efficiency savings.

If a source Switch knows that a particular destination Host logical address can be mapped to any of a number of destination Switches, then, as we have pointed out, it must be able to tell when, due to some sort of failure or network partition, the destination Host is (temporarily) unreachable via some particular Switch. It must have that information in order to be able to avoid choosing a destination Switch whose Pathway to the Host is non-operational. If we agree that the Pathway up/down status between a particular destination Switch and a particular destination Host which is ordinarily homed to it can only be obtained, on an exception basis, by that destination Switch itself, it follows that this information can also only be obtained by the source Switch on an exception basis. That is,

the only way for a source Switch to find out that a particular Host can temporarily not be reached through a particular destination Switch is to send a message for that Host to that Switch. The destination Switch must then determine that it has no operational Pathway to that Host, and it must send back a control message to the source Switch informing it of this fact. (In IEN 183, we christened these messages "DNA messages", for "Destination Not Accessible.) The source Switch will store this information in its address translation tables, so that from then on it does not choose a destination Switch whose Pathway to the Host is down. (Of course, in addition to sending this control information back to the source Switch, the putative destination Switch should also try to forward the message it received to one of the other Switches to which the destination Host is homed.)

This should work well, unless the Pathway between the original destination Switch and the destination Host comes back up. We must develop some way of informing the source Switch that that destination Switch is now once again usable as a destination Switch for that Host. A simple and robust way to handle this is as follows. When a source Switch is informed, according to the mechanism of the previous paragraph, that a particular destination Switch cannot reach a particular destination Host (without forwarding traffic through additional intermediate Switches), it marks (in its address translation tables) that Switch as UNUSABLE as a destination for that Host. However, this

information is reset periodically, say, every few minutes. In effect, this approach would cause a source Switch which is handling traffic for that destination Host to query the destination Switch periodically to see if it has become usable again. Note that no special control message is needed for the querying. The querying is done simply by sending data addressed to the destination Host to the destination Switch. If the destination Switch is still unusable, no data is lost, since the data can be readdressed by the destination Switch and sent to some other destination Switch which does have an operational Pathway to that destination Host. Note also that with this scheme, not all source Switches will be in agreement as to which destination Switches can be used to reach which destination Hosts at some particular time. But this is not much of a problem, as long as address translation is done only once, and not re-done at each intermediate Switch. Further, any source Switch which tries to use the wrong destination Switch will be told, via a DNA message, to use another one.

Lest there be any misunderstanding, we should emphasize that we are not proposing this as a general mechanism for determining which Hosts are homed to which Switches. That information is not to be obtained dynamically at all, but rather is to be installed in the translation tables at each Switch by the Network Control Center (or whatever equivalent of the Network Control Center we devise for the internet.) This mechanism is only used to

determine that a Pathway which ORDINARILY exists between some Switch and some Host is TEMPORARILY out of operation.

If a destination Host happens to be unreachable from EACH potential destination Switch (which will happen if the Host is down), this procedure will eventually result in the source Switch marking all potential destination Switches unusable. Once this happens, the source Switch should discard any data it receives which is destined for that destination Host, and should return some sort of negative acknowledgment to the source Host. The source Host can then try again, every few minutes, to send more data to the destination Host. Since the information marking a destination Switch as unusable (for a particular destination Host) is reset every few minutes, the source Host will be able to establish communication with the destination Host soon after it becomes reachable again. Strictly speaking, a negative acknowledgment from the source Switch is not required, and the current IP makes no provision for such a thing. Yet the information contained in the negative acknowledgment might well help the source Host to choose a suitable retransmission interval. If a destination Host is unreachable, it makes sense for a TCP to retransmit more infrequently than if the TCP has no information at all about why it is not getting any acknowledgments from the destination Host. Also, this information would be useful to the end-user (if the various protocol layers in his Host succeed in passing it back to him.)

A user who is not getting any response from the system may want to take a different action if he knows his destination cannot be reached than if he thinks that the network (or internet) is just slow.

This procedure, which is basically the same as the one we recommended (in IEN 183) for use with logically addressed multi-homed Hosts on the ARPANET, should resolve the partitioned net problem. Our approach is not dissimilar to one proposed by Sunshine and Postel in IEN 135. To quote them:

> A simpler solution to the partitioning problem follows the spirit of querying a database when things go wrong. Suppose there were another database listing networks and all the gateways attached to each net (whether up or down). This database would change slowly only as new equipment was added to the internet system. Further suppose that the gateways and internet routing are totally unaware of network partitions, except that gateways to partitioned nets find out when they cannot reach some Host on their own net. In this case, the gateway would return a Host Unreachable (through me) advisory message to the source. The source could then query the global database to get a list of all gateways to the destination net, and construct explicit source routes to the destination going through each of these gateways, trying each one in turn until it succeeded.

Note, however, that our proposal does not require any source routing, because it is Switches (i.e., gateways) themselves which are the addressable entities in our scheme, rather than networks (though the authors quoted above were considering how to handle the problem in the current Catenet environment, rather than how to design a new environment). The database they propose can be identified with the translation tables we have spoken of. Also,

our proposal handles the situation where a Pathway that was down becomes usable again, a case they don't seem to mention.

It is sometimes claimed that hierarchical addressing requires less table space than flat addressing, since there is no need to have an entry in a translation table for each address. We can see now that this is not true. If we wish to be able to handle multi-homing, and in particular to handle the "partitioned net" problem, we need to maintain table space for the Hosts with which we are in communication. This is true no matter what kind of addressing scheme we adopt.

Let's look now at how our scheme would handle the problem of mobile Hosts, i.e., Hosts which move from one network to another. We distinguish the case of "rapidly mobile" Hosts from the case of "slowly mobile" Hosts. A Host is slowly mobile if its move from one net to another can be made with enough lead time to allow manual intervention to update the logical-to-physical address translation tables. This case is handled simply by the presence of the logical addressing. When the Host moves to another network, it can still be addressed by the same name, but the translation tables are changed so that the logical address is now mapped to a different set of Switches. This creates some work for the internet administration and control center, but is completely transparent to higher level protocols, since the logical address does not change. On the other hand, we consider

a Host to be rapidly mobile if it moves from one net to another too quickly or too frequently to allow the procedure of modifying the address translation tables to be feasible. If we can know in advance that there is some limited set of networks to which that Host might connect, we can map the logical address of that Host onto the set of all gateways which connect to any of those networks. Our procedure for choosing one gateway to use as the destination gateway might be as follows. Try the first gateway on the list. If a DNA message is received, try the second, etc., etc. Once a source gateway begins sending traffic for a mobile Host to a particular destination gateway, it should always continue to use that gateway, until it receives a DNA message, in which case it should try the next one. You will note that this procedure is very similar to that used for non-mobile Hosts. In fact, it might be entirely identical. The only possible difference is that we might want to be much more reluctant to switch from one destination gateway to another in the case of mobile Hosts than in the case of non-mobile Hosts, since we expect that a mobile Host will not generally be reachable through all of the potential destination gateways at every time.