IEN 183

Logical Addressing

Bolt Beranek and Newman Inc.

Eric C. Rosen

May 1981

Bolt Beranek and Newman Inc.
Eric C. Rosen

Table of Contents

1 Introduction..... 1
2 Translating Logical to Physical Addresses..... 7
2.1 Translation Locus..... 7
2.2 Translation Methodology..... 10
3 Organizing the Translation Tables..... 17
4 Initializing the Translation Tables..... 23
5 Updating the Translation Tables..... 30
6 Operational and Implementation Considerations..... 49
7 Network Access Protocol Implications..... 53

Bolt Beranek and Newman Inc.
Eric C. Rosen

1 Introduction

This paper is a slightly modified extract from BBN Report No. 4473, "ARPANET Routing Algorithm Improvement, Volume 1," by Rosen, et al. It proposes a scheme for implementing logical addressing in the ARPANET. However, the issues dealt with are quite similar to the issues raised by the problem of implementing logical addressing in the Catenet (several IENs are currently in preparation in which we argued for this assertion at great length.) It is hoped, therefore, that the discussion will be of interest to internet workers as well.

In the current ARPANET, in order for a user to transmit data to a particular host, he must know the PHYSICAL ADDRESS of the host. That is, he must know which node the host is connected to, and he must know which port on that node is used to connect that host. Furthermore, this is the ONLY means a user has of identifying a host. In many respects, the physical address of a host computer can be compared to a person's telephone number. The problems inherent to physical addressing in a computer network are similar to those we would experience in ordinary interpersonal communication if a person's telephone number were the only means we had of identifying him. Dialing a particular telephone number allows us to establish a communications channel to a particular location. This works well as long as the person

with whom we wish to communicate remains at that particular location. When the person changes his location, though, the phone number becomes virtually useless, and the physical address of a host computer becomes equally useless if the computer's location within the network changes. In the context of interpersonal communication, this gives rise to the calling of a "wrong number". In the context of computer networking, this can give rise to the more serious phenomenon of mis-delivery of data. Furthermore, when a computer changes location within a network, it is quite difficult to carry out a procedure which reliably informs all users of its new physical address. There are difficulties in identifying all users, difficulties in contacting them once they are identified, difficulties in making sure that the information receives the proper level of attention once contact is made, and if all these difficulties are resolved, it is still difficult to make the necessary changes to computer software so that the new physical address is actually put into use.

There is another sort of problem would occur in interpersonal communication if our only means of identifying a person were by his phone number. It is very common for several people to share a phone number. If we identified people only by phone numbers, we could not distinguish among several people at

the same location. This problem arises in computer networking if several computers share the same port. There are, in fact, several reasons why it may be desirable to allow several computers to share the same port. One reson is simply the need to get by with a less than optimal amount of equipment, either due to economics or to shortage. If some administration has two computers, each of which needs to be on the network only part of the day, but which do not need to be on the network at the same time, sharing a single port may be the best solution. The increasing cost-effectiveness of such devices as port expanders and local networks may also make it more and more desirable to have several computers sharing the same port. A related problem arises if one thinks of a network of computers as consisting of "logical hosts," rather than physical hosts. Whereas a physical host would be a particular piece of hardware, a logical host would be the instantiation of a particular function. Thus a physical host which supported (for example) time-sharing during the day and batch processing at night could be regarded as two logical hosts which share the port on a time-division basis. A related application is the situation in which the functionality of two (small) physical hosts is combined into one (larger) physical host, which then can be thought of as consisting of two logical hosts. It could be very useful to have the flexibility to move logical hosts freely around the network, perhaps changing

the correspondence between particular logical and physical hosts, without having to inform all users of the new physical addresses.

Of course, the reasons these problems in computer networking are more serious than the analogous problems in interpersonal communication is that telephone numbers are not the only, or even the primary, means we have of identifying other people. We can identify other people by NAME, and this greatly facilitates our ability to get in touch with people even as they change location. It also enables us to specify the individual we wish to talk to, in the situation where several people share a telephone. Similar advantages would accrue if we could identify computers by name rather than by physical address. In order to get our data to the appropriate computer, its physical address would still have to be determined. But the user should be able to tell the network the name of the appropriate computer (perhaps a logical rather than a physical host), and let the network itself map the name to its physical address. For speed and reliability, the mapping function should be accomplished automatically (i.e., by software) with minimal need for human intervention. Schemes to accomplish this are known as LOGICAL ADDRESSING schemes, and the name of a computer is usually referred to as its "logical address" (though in fact, logical addresses are not addresses at all, since they, like names, are

independent of location).

A good logical addressing scheme should allow more flexibility than may be already apparent. We have spoken of the need to be able to identify a computer in a way which is independent of location, and of the need to be able to map several distinct names onto a single physical address. There is also a need to be able to map a single name onto several physical addresses. To carry out the analogy with interpersonal communication, this would correspond to the case where a single person has several telephones, with different phone numbers, possibly at different locations. This adds reliability to the communications process, since if the person cannot be reached at one phone number, perhaps he can be reached at another. If he can be reached at each of the numbers, he now can handle several conversations simultaneously, i.e., he has increased throughput. In computer networking, this sort of application is known as "multi-homing." In multi-homing, a single computer connects to the network through several ports, usually (though not necessarily) at several different network nodes. This allows the computer to remain on the network even though one of its access lines, ports, or home nodes fails, thereby increasing reliability. In the case where it is more economical (or otherwise more practical) to obtain several low-speed access

lines than to obtain a single high speed line, multi-homing can also allow a given host computer to obtain higher throughput at less cost.

Another sort of application which requires a single name to map into several physical addresses can be compared to a business which has several branch offices, each with a different phone number, but whose customers do not care which branch office they reach. This can be useful in certain sorts of internetting applications. Suppose, for example, that an ARPANET user wants to send a packet to SATNET, but that there are several equally good gateways between the two networks. It may be convenient for the user to simply specify a name like "Gateway-to-SATNET" and let the network choose which of the several gateways (i.e., physical addresses) to use. A related sort of possible application has to do with distributed processing and resource sharing. If some particular resource is available at any of several locations around the network, it may be desirable to allow the user to specify the resource by name, and allow the network to map that name onto some particular physical address according to criteria that the user need not be aware of. (Such a service was formerly offered by the ARPANET TIPs. By typing @N, a user would be connected to a "Resource-Sharing Executive" on one of several network TENEX systems. The user, however,

would have no way of knowing which system he was actually on.)

2 Translating Logical to Physical Addresses

2.1 Translation Locus

It should be obvious that any implementation of logical addressing would require the network to maintain a translation table. The user would specify a logical address, and the network would use this logical address as an index into the translation table in order to obtain the physical address (or list of physical addresses) to which it corresponds. The network would use the physical address internally to determine the routing of user messages. The need to maintain a translation table gives rise to a multitude of design issues. The first question that needs to be answered is, where should the translation table be located? Should every node maintain a copy of the whole translation table, or should there be just a few copies of the table scattered around the network in strategic locations? If there are only a few copies scattered around the network, then nodes which do not contain the tables would have to query the nodes that do in order to perform the translation function. This is less efficient (both in terms of overhead and response time) than placing the table in every node. Considerations of

reliability and survivability also favor placing the table in every node. This eliminates the possibility of finding that, due to network partition, all copies of the translation table are inaccessible. It eliminates the need to have some nodes serving as hot or cold standby for the nodes which do have the tables. This is an important advantage, since the protocols needed to implement "standby" tend to be slow and cumbersome, or else unreliable. Furthermore, if all nodes maintain identical copies of the translation table, there is no need to go through any special initialization procedure for creating the table when a node first comes up. Typically, a node which is just coming up has been reloaded from a neighboring node. If all nodes have an identical copy of the table, a node coming up can simply have its table reloaded from its neighbor, i.e., can copy its neighbor's table. (Under certain unusual conditions this may give rise to a race condition, but as we shall see later, it is a race that can be easily remedied and one that will not have any bad effect other than to slow the reload process.)

There is, however, a possible disadvantage to having the tables in every node. That is simply the need to have enough memory in every node to hold the table. In certain networks, particularly commercial ones, the network nodes may be of widely varying sizes and capabilities, and the smaller nodes just may

not be able to hold the complete translation table. Such networks would necessarily have a hierarchical structure (probably with some form of hierarchical routing), and a node would not be able to hold a translation table unless it were at or above a certain level in the hierarchy.

Strictly speaking, only those nodes which will ever need to translate a logical address to a physical address will need to have a copy of the translation table. Whether that is all nodes or only a subset of the nodes depends on the translation methodology that we adopt. We have a choice between requiring translation to be done only at the source node, or requiring it to be done also at tandem nodes. In the former case, when the user presents some data to the network and specifies its destination with a logical address, the source node looks in the translation table, gets the physical address, places the physical address in the packet header, and sends the packet on its way. Tandem nodes do not look at the destination logical address at all, but only at the physical address. In the other case, each tandem node looks at the logical address, does its own translation to physical address, and routes the packet on that basis. The packet header would not even have to contain the physical address. If we do source translation only, the only nodes which need to contain translation tables are those that

connect to hosts which use logical addressing. If these nodes are only a subset of all the nodes, then it may be feasible to configure these nodes with more memory than the others. Therefore, we must look at the advantages of source node vs. tandem node translation.

2.2 Translation Methodology

Clearly, in the case where a logical address maps to a unique physical address, source node translation is superior to tandem node translation. As long as there is only one possible physical address for that logical address, all nodes will produce exactly the same mapping. There is thus no advantage to performing the mapping several times, and the scheme which does it only once is more efficient. There is, however, one exception to this. When the translation tables need to be updated, we cannot expect all copies to be updated simultaneously. There will necessarily be some short interval of time when not all of the copies of the table around the network are identical, and during this interval, tandem node translation may yield different results than source node translation. It will certainly be necessary to design some mechanism to deal with this problem, and we shall propose one shortly for the ARPANET environment. Tandem node translation, however, is not the right solution to this

problem. During the transient period, some copies of the table will be right (up to date) and some wrong (out of date). But the copies at the tandem nodes will be no more likely to be right than the copy at the source node, so tandem node translation would be as likely to amplify the problem as to reduce it. The solution to this problem lies elsewhere.

In the case where a logical address maps to several physical addresses (multi-homing), tandem node translation might well give different results than source node translation. However, we must now distinguish between virtual circuit and datagram traffic. If virtual circuit traffic is logically addressed, all translation must be performed at the source node. In fact, the translation must be performed only once, at connection set-up time. This is the only way to ensure that all traffic on a given circuit is sent to the same physical address, which in turn is the only way to provide the sequencing and duplicate detection that is the RAISON D'ETRE of virtual circuit traffic. (Additional issues having to do with logically addressed virtual circuit traffic will be discussed later.) Thus the only possible advantage of tandem node translation would have to do with datagram traffic which is destined to a multi-homed logical address. To understand the differences, though, between source node and tandem node translation, we must first discuss

the criteria which a node uses to pick one physical address out of the several that are available. Even though a host is multi-homed, any particular packet for it can go to only one of its physical addresses; some criterion for choosing the proper one in each particular case must therefore be available. Several possible criteria come readily to mind:

a) When there are several physical addresses corresponding to a given logical address, it may be desirable to send packets to the physical address which is closest to the source node, according to some metric of distance. For example, if we are interested in minimizing delay, we may want to choose the physical address to which the delay from the source is least. If SPF routing is used, this information is readily available. If we are interested in minimizing the use of network resources by a particular packet (i.e., in maximizing throughput while still using only a single path), we may want to choose the physical address which is the least number of hops from the source. (For these purposes, ties can be broken arbitrarily.) Again, this information can be made readily available by the SPF routing algorithm (although it is not readily available from the ARPANET's particular implementation of that algorithm, since a table of hop-counts is not saved). If either of these criteria is used, tandem node translation can result in better route

selection for the logically addressed datagram. If delay changes or topology changes take place while a packet is in transit, it may happen that the "closest" physical address to some tandem node is different from the physical address that was closest to the source node when the packet first entered the network. It is easy to prove that, if SPF routing is used, this cannot result in looping, except as a transient phenomenon while a routing update is traversing the network. That is no particular disadvantage, since a packet may be subject to that sort of transient looping even when its destination physical address does not change. However, it is not clear that tandem node translation provides much of an advantage either, especially when one takes into account the additional overhead of doing the re-translation at each tandem node. Doing translation at tandem nodes will necessarily increase the nodal delay and decrease the nodal throughput. These negative effects may outweigh the positive effects of improved route selection for those relatively rare cases in which delay or topology changes significantly while a packet is in transit. We must remember that although real improvements in route selection would only occur rarely (since delay and topology changes are very infrequent when compared with average network transit times), re-translation would have to be done for EVERY logically addressed datagram. Unfortunately, all these effects are extremely difficult to quantify with any degree

of confidence. Our (somewhat intuitive) conclusion is that, under the selection criterion of choosing the closest physical address, tandem node translation offers at best a small improvement over source node translation, and at worst a severe degradation.

b) It is possible that some multi-homed hosts will want to establish an inherent ordering to their ports. That is, they may prefer to receive all their traffic on port A, unless that port is inaccessible to the source of the traffic, in which case they prefer to receive all traffic on port B, unless that port is inaccessible to the source of the traffic, in which case they prefer to receive all traffic on port C, etc. This sort of strategy may be appropriate if certain of the host access lines are charged according to a volume-based tariff, while others are not. It may also be appropriate if certain of the access lines can be used more efficiently (i.e., can be serviced with less host CPU bandwidth) than others. (An example might be a host which can access the ARPANET through an 1822 line and a VDH line. It might be desirable to avoid the VDH line, unless absolutely necessary, since VDH lines tend to be used less efficiently.) In either case, the idea would be to reduce cost by favoring certain ports over others, using the more expensive ports only when needed for purposes of reliability or availability. Thus we may

want the logical addressing scheme to support an inherent ordering among the several physical addresses which correspond to a given logical address. With this scheme, there is no advantage to doing tandem node translation. There will be only one ordering for the set of physical addresses corresponding to a logical address, so tandem nodes should always pick the same physical address as the source node picked, and re-translation would simply be a waste of resources. There are only two exceptions to this. The first exception would arise in the situation where a particular physical address becomes inaccessible as a packet routed to that address is traversing the network. However, since this can happen no matter what criteria are used for choosing among the physical addresses, we put it off for later consideration. The second exception would arise if the translation tables were being updated while a packet is in transit. Clearly, some procedure to deal with this case must be devised, since the update which is taking place may invalidate the translation which was done at the source. Tandem node translation is not the proper solution, however, since there is in general no reason to believe that the tandem node is more up-to-date than the source node. We will return to this issue when we discuss the table updating procedures.

c) Certain multi-homed hosts may have a preference for

receiving certain kinds of traffic over particular ports. Thus a dual-homed host may consider one of its ports more suitable for receiving batch traffic, and another more suitable for receiving interactive traffic (perhaps the first port offers a higher speed but a longer delay than the second). However, if one of the ports is inaccessible from a particular source node, that node would send all its traffic (both kinds) to the port which remains accessible. With this criterion, we see once again that tandem node translation offers no benefits, since, barring the case of a port becoming inaccessible while a packet is in transit, all tandem nodes would select the same physical address as the source node.

d) Some multi-homed hosts may wish to try to keep their several access lines as equally loaded as possible. One possible way to do this would be to establish an inherent ordering to the ports (as in b, above), but to make the ordering different for different source nodes (or hosts). Clearly, this scheme requires source node translation; tandem node translation would only serve to defeat it. Another possible way to achieve some sort of load leveling would be for each source node to send traffic to the various physical addresses on a round-robin basis. This would be a very crude form of control, but might work reasonably well for particular traffic patterns. This scheme also requires source

node translation. In fact, tandem node translation could actually cause packets to loop endlessly.

We see then that none of the suggested selection criteria give any very large advantage to tandem node translation, and some of the criteria are actually in conflict with re-translation at tandem nodes. Thus it is not necessary for all notes to hold the translation tables; only those nodes connected to hosts which use logical addressing need hold the tables. Of course, it is still possible that the tables will be too large to be held in any one node, in which case we would have to use distributed database techniques to split the tables among several nodes. We will not consider that further here, but we will consider it in a future IEN.

3 Organizing the Translation Tables

Another issue having to do with the translation tables is the way in which the tables should be organized. Clearly we do not want the entries in the table to be in random order, necessitating a lengthy linear search each time a translation must be done. Basically, there are two possible ways to order the table. We can sort the table by logical address, and do a binary search of the table whenever we need to do a logical-to-

physical address translation. This is a rather efficient form of searching, but it causes inefficiencies when table entries have to be inserted or deleted, since that would require a potentially time-consuming expansion or compression of the table. There are, however, ways of reducing the overhead involved in insertions/deletions. Deletions could be made "logically", i.e., by marking an entry deleted, rather than physically compressing the table. New entries could be inserted into an overflow area, which itself would be searched linearly whenever a particular logical address could not be found in the main table. Actual compression/expansion of the main table would be done only when the overflow area filled. Note, however, that this sort of strategy would necessarily complicate the search algorithm, and this might actually do more harm than good, especially if insertions and deletions are rare events. We shall see later, when we discuss the conditions under which insertions and deletions are required, that these are indeed rare events. We conclude tentatively that, if a sorted table with binary searching is used, the use of an overflow area is probably not necessary.

The other possibility for organizing the table is to use hashing. A good hashing algorithm (i.e., one which minimizes collisions) provides very efficient insertions and very efficient

searches. Deletions are not quite so efficient, but are still more efficient, in general, than the table compression required if binary searching is used. However, hashing has certain inherent problems which may make it less suitable. Choosing a hashing algorithm which both minimizes collisions and is computationally efficient is not a simple matter. One must be sure that the time needed to perform the hashing is really less than the average time needed to find an entry in a sorted table by means of binary searching; otherwise, the efficiency is lost. Furthermore, the number of collisions generated by a particular hashing algorithm will depend on exactly which set of logical addresses are in use. The set of logical addresses in use during a network's lifetime will be a slowly changing set, and a hashing algorithm which is excellent at one time may give poor performance at another. Hashing algorithms are also subject to undetected programming bugs in a way in which binary search algorithms are not. A bug which is inserted into a hashing algorithm, which, for example, causes all entries to hash into the same bucket, might go undetected for years, although it would cause a significant performance degradation by reducing the efficiency of the hashing technique to that of linear searching. A bug in the binary search algorithm, however, would be more likely to come to someone's attention. Its probable result would not be performance degradation, but rather, failure to find

certain entries. This would cause inability to deliver traffic to certain logical addresses, and this would certainly come to the users' attention very quickly. These qualitative reasons would seem to indicate that binary searching is preferable to hashing. It would also be useful to do some quantitative analysis; that may be done at a later stage of our research.

Someone may wonder why we have not considered a much simpler method of organizing the tables. Logical addresses, after all, are just numbers (or at least are representable as numbers). If the set of logical addresses in use at any given time is a contiguous set of numbers, then the addresses can be used as indexes directly into a translation table, with no need either for hashing or for sorting. The problem, of course, lies in the requirement that the set of logical addresses form a contiguous set of numbers. Assigning numbers to hosts contiguously may not be a problem in itself, but it does cause a problem as soon as some host is removed from the network. Its number (or numbers, if it has several logical addresses) cannot be left unused, or the size of the translation table would be determined not by the number of logical addresses currently in use in the network, but rather by the number of logical addresses that have ever been in use in the network, a number which may be much larger, and which in fact has no bound. Yet it is not

acceptable simply to reassign the same logical addresses as hosts enter or leave the network. We all have some experience with moving to a new location, getting a new telephone number, and finding ourselves frequently getting calls intended for the person who previously had that number. Such calls may persist for years, especially if the number previously belonged to a business. Receiving phone calls or mail intended for someone else who happens to have the same name as we do is also a familiar occurrence. We would expect analogous problems if logical addresses are reassigned (at least, if they are reassigned without some very long waiting period), especially if the logical address previously belonged to a large service host. When a user tries to address a host which is no longer on the network, he should receive some indication of that fact; he should NOT have his data mis-delivered to some other host which has been assigned the same name. Thus it is preferable to have a logical addressing scheme which does not depend on the logical addresses forming a contiguous set of numbers.

Whatever means of organizing the table is chosen, it may still be useful to maintain a smaller table for use as a "cache." The cache would contain the n most recently used logical addresses (where n is some small number), together with a pointer to the absolute location of that logical address entry in the

main translation table. When it is necessary to do translation, the cache would be searched before the main table. The assumption is that once one packet for a particular logical address is received from a source host, many more will follow. Thus it pays to optimize the search for that particular logical address. Choosing the optimum size for the cache, and the means of searching it (linear or binary) are issues left for later resolution. These issues must be dealt with carefully, however; one would not want to find that searching the cache takes as long or longer than searching the main table. It is worth emphasizing that the cache should contain a pointer into the main translation table, rather than a copy of the list of physical addresses associated with a particular logical address. For multi-homed logical addresses, this is more efficient, since it involves less copying. Also, if there are variables associated with the physical addresses, this enables unique copies of the variables to be kept in the main table. (Suppose, for example, that packets are to be sent on a round-robin basis to the several physical addresses corresponding to a multi-homed logical address. This requires a variable to be associated with each physical address, indicating whether that physical address was the last one to be sent data. This variable must be kept in the main table, not the cache, since one cannot rely on a particular logical address always being present in the cache.) This means,

of course, that the cache must be cleared whenever there are insertions or deletions into the main translation table, but that should not be very expensive as long as such insertions and deletions are relatively rare.

4 Initializing the Translation Tables

We turn now to the issue of how the translation table entries are to be set up in the first place. That is what procedure is to be used for establishing that a particular logical address is to map to a particular set of physical addresses. One possibility for the ARPANET, of course, is to have all the mappings set up by the Network Control Center (NCC). This is quite reasonable in certain cases. If some user wants his computer to be addressable by some new logical address (i.e., by a logical address not previously in use), it makes sense to have him contact the NCC directly. If the user has proper authorization, the NCC can then take action to set up the new entry in all the translation tables. A similar procedure would also be appropriate if some logical address is to be totally removed from the translation tables (i.e., that logical address will no longer be in use in that network). This procedure would also be appropriate when a particular computer is moved from one location to another, necessitating a change in its logical-to-

physical mapping, or if the functionality of two computers is combined into one, so that two logical addresses which formerly mapped to distinct physical addresses now map to the same physical address. What all these cases have in common is that they are relatively infrequent (i.e., occurring on the order of days, rather than on the order of minutes), and they require considerable advance planning. The first of these characteristics ensures that NCC personnel will not be swamped with translation table changes. The second of these characteristics makes it feasible to coordinate such changes in advance with the NCC. Unfortunately, not all translation table changes have these characteristics. For example, we have suggested that a good logical addressing scheme should facilitate port-sharing. That is, some user might want to unplug one of his computers from the network and use that port for another computer. He should be able to do this without much advance planning, and without having to explicitly coordinate with the NCC. As soon as the change is made, users who are logically addressing the first computer should be told that it is no longer on the network; only the logical address of the second computer should map to this port. If this change in the mapping does not take place immediately, the result can be mis-delivery of data, as packets which are logically addressed to the first computer get mis-delivered to the second. A similar situation arises if

some computer consists of several "logical hosts." Logical hosts may come and go quite frequently, with no advance planning at all. The logical addressing system should be able to adapt immediately to such changes, without any need for human intervention. A related situation arises in the case of logical addresses which are multi-homed. We have already discussed various possible criteria for choosing among the several physical addresses associated with a single multi-homed logical address. But before applying these criteria, any physical addresses which are "inaccessible" from the source node must be excluded. If some host has two access lines into the network, and one of them is inaccessible from a particular source node, then all traffic from that source node should be directed to the other access line. Indeed, this is one of the most important purposes of multi-homing. This implies that a source node must have some way of knowing that a certain physical address is not currently accessible. There are basically two classes of reasons why a given physical address might be inaccessible from a source node. The first is that there is no path from the source node to the destination node (either because the network is partitioned, or because the destination node is down). This information is readily available from the routing tables, and need not be kept in the translation tables. It is simple enough to check the routing tables when choosing one from a set of physical

addresses. The other reason why a physical address might be inaccessible is that the port itself, or the access line from the port to the host, has failed. Functionally, this is the equivalent of unplugging the host from the port. It may happen quite frequently, however, and certainly with no advance planning. As long as the node itself is up, the routing algorithm will give no indication that the port is inaccessible; this information must somehow get into the translation tables. Clearly, we do not want to depend on human intervention to ensure that this sort of change gets made in the translation tables. What is needed here is a quick and reliable means of making changes in the translation tables, not the cumbersome and unreliable method of contacting the NCC. The same problem arises when the inaccessible port becomes accessible again. One wants to be able to begin using this port again as soon as possible, without having to wait until NCC personnel have time to make the appropriate changes in the translation tables. So although certain sorts of changes to the translation tables can be made by the NCC, many sorts of changes will occur suddenly and unexpectedly, and need to become effective immediately. So the procedure of having all translation table changes made by the NCC is not satisfactory.

There is another sort of problem with having translation

table changes made by the NCC. The problem is that carelessness, either by the NCC or by host site personnel, can result in mis-delivery of data if changes are made by the NCC. Suppose, for example, that a network controller makes a typographical error, associating a logical address with an incorrect physical address. If there is no further check on the validity of that mapping, one computer may receive data intended for another. A good logical addressing scheme should prevent this sort of simple typographical error from resulting in mis-delivery. The same situation can occur if one computer is carelessly plugged into the wrong port. In this case, networks which use only physical addressing might also mis-deliver data. However, with physical addressing, one must expect mis-delivery if some computer is plugged into the wrong port (i.e., given the wrong physical address) due to carelessness. With logical addressing, this is not inevitable, and a good scheme should give better protection against carelessness.

Another possibility for setting up the translation table entries is to have each host, as it comes up on the network, tell the network which logical addresses it wants to be addressed by over each of its (physical) ports. This would require augmentation of the host access protocol to include a "Logical Address Declaration" (LAD) message. A given host could put as

many logical addresses as it wanted in each LAD message. Multi-homed hosts would send the same LAD message over each of their ports. The logical addresses specified in the LAD message received over a given port would all be mapped to that particular physical address. Hosts would be allowed to change their logical addresses at any time by sending a LAD message to the network. Since a host may wish to add or delete logical addresses for itself at any time, there would have to be two options for the LAD message -- "add" and "delete." Whenever a particular port goes down (either because the port itself fails to function properly, or because the access line between the network and the host fails, or because the host itself crashes), all mappings of logical addresses to that port would be cancelled. When the host can once again communicate with the network through that port, it would have to redeclare its logical addresses with a LAD message before it could receive any logically addressed traffic.

Allowing each host to set up its own logical-to-physical address mappings in this manner has several advantages over having all the mappings set up by the NCC. This procedure allows sudden and unplanned mapping changes to take effect immediately, with no need for advance planning and coordination with the NCC. Since the mappings are cancelled immediately when a port goes down, this procedure helps to ensure that, if one of a multi-

homed host's ports is down, all data which is logically addressed to that host will go to the other ports. If one host is unplugged from a given port, and another plugged in its place, the procedure ensures that the mapping for the first host is cancelled, while the mapping for the second host becomes effective. When a host goes down, there is no assumption that the same host will return in the same location. Hence carelessness on the part of site personnel or NCC personnel cannot result in mis-delivery of data; data which is logically addressed to a certain host could only be delivered to a host which has declared itself to have that logical address.

There are, however, two quite serious problems with this procedure. The first problem is that of spoofing. That is, this procedure offers no protection against the situation where one host declares itself to be addressable by a logical address which is supposed to be the logical address of a different host. Thus the procedure allows one host to "steal" traffic intended for another, simply by declaring itself to have the same logical address as the other. This sort of spoofing might be done by a malicious user, who is really trying to steal someone else's data, or it might happen accidentally, as a result of programmer or operator error. In either case, we would like to have some procedure which is less prone to spoofing. The other serious

problem with this procedure is that it can easily cause the translation tables to overflow in size. If every host can specify an uncontrolled and unlimited number of logical addresses for itself, there is no bound on the size of the translation tables. Since only a finite amount of memory will be available for the translation tables, it is clearly not acceptable to allow each host to specify an arbitray number of logical addresses for itself.

5 Updating the Translation Tables

We have examined two different procedures for setting up the logical-to-physical address mappings, and have found that they both have problems. Many of these problems can be resolved, however, by a combination of the two procedures. Let us define two characteristics of a logical-to-physical address mapping, which we will call "authorized" and "effective." A mapping from a particular logical address to a particular physical address is "authorized" if a host which connects to the network at that physical address is allowed to use that logical address. Authorizations would change very infrequently, and only after considerable advance planning. Hence it is appropriate for authorizations to be determined (i.e., added and deleted) by the NCC. A mapping from a particular logical address to a particular

physical address would be said to be "effective" from the perspective of a given source node if (1) that mapping is authorized, (2) that physical address is accessible from that source node, and (3) the host at that physical address has, by means of a LAD message, declared itself to have that logical address. When a port goes down, all mappings to it will become ineffective, until they are made effective again by means of a LAD message. Logically addressed traffic will not be delivered to a particular physical address unless the mapping between that logical address and that physical address is effective. Changes in the effectiveness of a mapping will occur automatically, in real-time, with no need for intervention by NCC personnel. This facilitates multi-homing, since if there are two authorized mappings of a logical address to a physical address, and only one is effective, that one can be chosen all the time until the second becomes effective also. It facilitates sharing of ports (either by physical or by logical hosts), since each host has control over the effectiveness (though not the authorization) of the mappings that affect it. Carelessness by NCC personnel can cause the wrong mappings to become authorized, but it is rather unlikely that an incorrectly authorized mapping could become effective -- that would require carefully planned malicious intent. Therefore, such carelessness might prevent delivery of data to some host, but would not cause mis-delivery of data.

Carelessness by site personnel, such as plugging a host into the wrong port, would not cause mis-delivery of data, since the mapping of that host's logical address to that particular port would not be authorized. The possibility of spoofing is greatly reduced; since host A cannot pretend to be host B unless it is at a port which is already authorized for host B. The size of the translation table cannot increase without bound, since that is determined by the number of authorized mappings, and cannot be increased by LAD messages. This means, of course, that the network access protocol must be further modified so that it can provide positive and negative acknowledgments for the LAD messages. For each logical address that a host specifies for itself in a LAD message, the network must return either a positive or a negative acknowledgment. The positive acknowledgment would indicate that the mapping is authorized and has become effective. The negative acknowledgment would indicate that the mapping is not authorized.

It must be emphasized that the suggested procedures are not intended to provide security in any very strict sense. For networks in which security is a very important issue (e.g., AUTODIN II), further study of these issues should be carried out by security experts.

It should also be emphasized that these procedures will

allow the logical addressing scheme to continue to function normally even if the NCC facilities are down. It does require centralized intervention to add or delete authorizations, and this could not be done if the NCC were down. For a fixed set of authorized mappings, however, no centralized intervention is required to determine the effectiveness of the mappings. That is, the real-time functionality and responsiveness of the logical addressing scheme does not depend in any way on the proper functioning of the NCC.

We have argued that the authorization of a mapping should be determined by the NCC, and the effectiveness of a mapping should be determined by the network node which contains the physical address (port) to which the mapping is made, in cooperation with the host that is connected to that port. We have also argued that a full translation table (i.e., a table containing all the effective mappings) should be stored at each network node (or more precisely, at each network node which serves as an access point for a host which can be either a source or a destination of logically addressed traffic). However, we have not yet discussed the algorithm by which the effectiveness or ineffectiveness of a particular logical-to-physical address mapping is communicated to all network nodes. We turn now to this issue. We will discuss two very different methods for

building up the translation tables at all nodes.

The first method is based upon an extension of the SPF routing algorithm, wherein each logical address is treated like a stub node. In this method, each node is initialized with a partial translation table. This table contains a list of all the logical addresses which are AUTHORIZED to map TO that node. (i.e., all the logical addresses which correspond to ports at that node). Each of these logical addresses is associated with a particular port or ports at that node. At initialization time, each of these logical addresses is treated just as if it is a neighboring node which is down, and the node sends an update (similar to a routing update) to all other nodes, indicating that all authorized mappings to itself are ineffective. When a host comes up over a particular port, it declares its logical address(es) by means of one or more LAD messages. The node then checks its table of authorized mappings, and acknowledges to the host (either positively or negatively) each logical address mentioned in the LAD message. Whenever a logical address is positively acknowledged, it becomes effective, and the node must broadcast an update to all other nodes declaring that mapping to be effective. Whenever a host declares (via a LAD message) that it no longer wants to be addressable by a particular logical address, an update must be generated declaring that mapping to be

ineffective. Whenever a port goes down, all logical addresses mapping to it become ineffective, and an update indicating this must be broadcast. If the protocol used to disseminate these updates is the same as the protocol used in the ARPANET to disseminate the updates of the SPF routing algorithm, then all nodes will be able to build dynamically an up-to-date table of effective mappings, just as the routing updates enable them to build an up-to-date topology table. (The procedure used to build the topology tables is described in [1]. The updating protocol is described in [2] and [3].) In effect, this procedure extends the routing algorithm to treat the hosts (or more precisely, the mappings of logical addresses to physical addresses) as stub nodes, and the ports as lines, except that there is no delay associated with a port, but only an up/down status.

This procedure is attractive from a conceptual point of view, but it is not really cost-effective. That is, it seems to be too expensive to be practical. One reason is that it is hard to place a bound on the size of the updates. The updating protocol of the ARPANET routing algorithm is quite efficient, because the updates are so small. The maximum update size is only 216 bits (from a node with 5 neighbors). The logical addressing updates might be much longer, since there is no limit on the number of logical addresses that may map to a given node.

The updates would also have to be sent periodically, even when there in no change in state. These features are necessary to ensure reliability in the face of such events as partitions, node crashes, updates received out of order, etc. With no restriction on the number of logical addresses which can map to a given node (and it seems unwise to build in such a restriction), there is no restriction on update size, and hence no bound on the bandwidth needed for updating, or on the extra delay which may be imposed on data packets due to the need to transmit the updates.

The updating protocol which was designed for the SPF routing algorithm was designed to get the updates to all nodes very quickly, and with 100% reliability, even in the face of various types of network failures. This extreme speed and reliability is necessary for routing updates, since rapid and reliable updating of the routing tables is necessary to ensure the integrity of the network. Routing failures, after all, can make the network completely unusable, and can be very difficult to recover from, since most recovery techniques depend on the NCC's ability to communicate with the nodes, which in turn depends upon the integrity of the routing algorithm. Fortunately, the protocol used for disseminating the logical addressing updates does not need all the functionality of the updating protocol used for routing, since the integrity of the

logical addressing scheme is not quite as critical as the integrity of the routing algorithm. This enables us to use a simpler and less expensive method of maintaining the translation tables, which we will now discuss.

In this second method, each node is initialized with a translation table containing ALL the authorized mappings. This table would have an entry for every logical address that can be used in the network. Each logical address would be associated in the table with all the physical addresses to which it has an authorized mapping. Associated with each of these physical addresses would be a Boolean variable indicating whether that particular logical-to-physical address mapping is effective or ineffective. At initialization time a node would mark all mappings to itself as INEFFECTIVE, and all mappings to other nodes as EFFECTIVE. Whenever a host declares itself, via a LAD message, to have a certain logical address, the node looks in the translation table to see if that mapping is authorized. (This is just an ordinary table look-up, indexed off the logical address.) If not, a negative acknowledgment is sent to the host. If the mapping is authorized, a positive acknowledgement is sent, and the entry in the translation table is marked EFFECTIVE. Whenever a port goes down, the node marks all mappings to that port as INEFFECTIVE. Of course, this also requires a "reverse" search of

the table (i.e., a search based on a physical, rather than logical address). To make this more efficient, when the initial reverse search is done at initialization time, the node can save a list of pointers into the translation table. Each pointer would correspond to a physical address entry for that node. If a separate table of pointers is kept for each port, the node will be able to find in a very efficient manner entries which map to a particular port. Using this methodology, each node's translation table will correctly indicate, for each of the logical addresses that map to IT, whether or not that mapping is effective. (Of course, these pointers would have to be adjusted whenever table insertions or deletions are made.)

When a source host sends a logically addressed datagram packet into the network, the source node will search the translation table for the correct mapping. If that logical address cannot be found, i.e., its use is not authorized, an error message indicating this fact should be returned to the host, and the packet discarded. If that logical address is found, but all the corresponding physical addresses are either marked INEFFECTIVE, or else are unreachable (according to the routing algorithm), then the packet should be discarded, and the host informed of that fact. If some of the physical addresses are both reachable (according to routing) and marked EFFECTIVE,

then one should be chosen, according to some set of criteria (perhaps one of those which we discussed above). The chosen physical address should be placed in the packet header, along with the logical address. The packet should then be forwarded to its destination; in doing the forwarding, tandem nodes will look only at the physical address. According to the procedure described in the previous paragraph, all mappings to REMOTE ports will be initially marked EFFECTIVE. To see how such mappings can get marked INEFFECTIVE, we must see what happens when a logically addressed packet reaches its destination node.

When a logically addressed datagram packet reaches its destination node, the node looks up that logical address in its translation table. It is possible, of course, that that logical address will not be found at all, or that it will be found, but that there will be no authorized mapping to this particular destination node. This would indicate some sort of disagreement between the translation tables at the source and destination nodes. There are three possible causes of this disagreement: (1) NCC error in setting up the translation tables, (2) deletion of the authorization for that particular logical address while the packet was in transit, or (3) a race condition, whereby a translation table update authorizing the new logical address is taking place, but the update has not reached that destination

node yet. In any case, the data packet should be discarded without delivery, and an error message should be sent to the NCC indicating receipt of a packet with an unauthorized logical address. This will alert NCC personnel to a possible error. If the authorization for that logical address was deleted while the packet was in transit, however, then the NCC need not take any action; having the destination node simply discard the packet is the correct procedure. If, on the other hand, that logical-to-physical mapping is really authorized, but the update making the authorization has not yet reached the destination node, then we want to take the same action as we would take for a packet delivered according to an authorized but ineffective mapping. This action shall be described in the next paragraph.

Suppose that, upon looking up the logical address in the translation table, the destination node does find an authorized mapping to itself, but that mapping is marked ineffective. Then there are two actions to take. The first action is to try to re-address and then re-send the message. Of course, this can only be done if the destination logical address is multi-homed, and at least one of the corresponding physical addresses is effective. If this is not the case, the packet must be discarded. The second action is to send a special message back to the source node of that datagram packet. We will call this

message a "DNA message" (for "Destination Not Accessible"). The DNA message will specify that the particular logical-to-physical address mapping used for that packet is not an EFFECTIVE mapping. The DNA message should also be sent in response to datagrams which appear to have unauthorized mappings (see previous paragraph). For reliability, each logically addressed datagram must carry the physical address of its source node (though not of its source host), so that the DNA message can be physically addressed to the source node. It is not enough for the packet simply to carry the logical address of its source host, for two reasons. The first reason is that if the source host is multi-homed, the destination node will not know which source node the packet came from, and hence will not know where to send the DNA message. The second reason has to do with the fact that one situation in which DNA messages may have to be sent is the situation in which the translation table at the destination node has been set up erroneously. In this case, we do not want to have to rely on the integrity of the translation table to ensure proper delivery of the DNA message.

When a source node receives a DNA message indicating that a certain logical-to-physical address mapping is ineffective, it must find the proper entry in its translation table, and mark that mapping as ineffective. Henceforth, incoming packets with

that particular logical address will not be sent to that particular physical address. If the logical address is multi-homed, packets will be sent to one or more of the other physical addresses, unless all the mappings for that logical address are ineffective. If this is the case, packets for that logical address will be discarded by the source node, which should also return some sort of negative acknowledgment to the source host. We see then that the DNA messages provide a feedback mechanism which enables a source node to tell when a mapping to a remote port is ineffective. The source node has no way to tell whether this is the case, until it sends a packet to that port. After sending the packet, it will be explicitly told by the DNA message if the mapping is ineffective. If it receives no DNA message, it assumes that the mapping is effective. This may mean, of course, that some logically addressed packets are sent to a wrong physical address. However, if there are other possible physical addresses corresponding to that logical address, and the original destination node has one of those other mappings marked as effective, the packet will be re-addressed and re-delivered, so there is no data loss. Note that there are two possible reasons why a given logical-to-physical address mapping might be ineffective: (1) the physical port might not be operational, or (2) the host at that physical address might not have declared itself addressable with that logical address. If desired, the

DNA message can indicate which of these two reasons is applicable in the particular case in hand. This information can be stored in the source node's translation table, and passed on to source hosts which try to use a logical address for which all the mappings are ineffective.

This procedure enables all nodes to find out when a particular authorized mapping is ineffective. We also need a procedure to enable the nodes to find out when an ineffective mapping becomes effective again (i.e., a port comes back up, or a new LAD message is received at some remote site). A simple but effective method is the following. At periodic intervals (say, every 5 or 10 minutes) each node will go through its translation table and mark ALL the entries which map to REMOTE ports to be effective. (Entries which map to local ports will be marked effective or ineffective according to procedures already discussed. The current procedure will not apply to such entries.) This enables mappings to be used again shortly after they become effective. Of course, this scheme will result in some packets being sent to the wrong physical address. When that happens, however, a DNA message will be elicited, causing that mapping to be marked ineffective again in that source node. Furthermore, this scheme does not cause any unnecessary data loss, since packets sent to the wrong physical address will be

re-addressed and re-delivered, if possible.

Although this method requires all nodes to periodically mark all mappings to remote ports effective, it is important to understand that it does not require any time-synchronization among the various nodes. Also, there is no reason why all the mappings have to be marked effective at the same time. For example, if the translation table contains 600 mappings, rather than marking all of them effective every 10 minutes, it may be more efficient to mark one mapping effective each second, thereby cycling through the table every ten minutes (though if this method is used, it must take account of table compressions and expansions which may occur as the NCC adds or deletes authorizations).

There is also an issue as to the exact methodology to be used to send the DNA messages. The simplest method is for a destination node to send a DNA message to the source node of each packet which arrives as the result of an ineffective mapping. If this method is used, there is no need to use a reliable transport protocol in sending the DNA messages. If, for some reason, a DNA message fails to get through to the source node, more packets will arrive at the wrong destination node, causing more DNA messages to be sent, until one of them finally gets through. This method, however, might generate a virtually unbounded number

of DNA messages, particularly in pure datagram networks with no flow or congestion control. This in turn might contribute to network congestion. In order to gain better control of the throughput due to DNA messages, one could implement a scheme which ensures that only one DNA per ineffective mapping per source node can be sent within a certain time interval. This scheme would have a significant cost in table space, however. Also, it would require some sort of reliable transport protocol (e.g., positive acknowledgments from the source node when it receives the DNA message) to protect against the case where a DNA message is lost in transit. This issue would have to be carefully considered before any implementation is done.

The procedure to follow with virtual circuit traffic is very similar. In the ARPANET, a single virtual circuit or "connection" is individuated by the source and destination physical addresses. The user takes no part in setting up a connection; whenever a user sends a packet to the network which is not a datagram, the network checks to see if a connection from the user's physical address to the destination physical address that he specified is already in existence. If not, the IMPs automatically run a protocol to set up such a connection. With logical addressing, we would want to redefine the notion of a connection so that connections are individuated by source and

destination LOGICAL address, rather than physical address. However, translation would be done only at connection setup time. Thereafter, all virtual circuit packets received by a given source node with the same source logical address and destination logical address would be sent on the same connection. If a destination node receives a connection setup message for a logical address whose mapping is ineffective, it will refuse the connection, just as it would refuse a setup message for a physical connection to a dead port. When the source node receives the refusal message, it will mark that particular logical-to-physical mapping as ineffective. If the destination logical address is multi-homed, the source node can attempt to set up the connection again, but with a different physical destination address. If a mapping becomes ineffective after a connection has already been set up, the destination node will take action to reset the connection, also informing the source node that the mapping is now ineffective.

Note that logically addressed virtual circuit packets need not carry in their headers the logical addresses of either the source or destination hosts, since that information can be stored in the connections tables at the source and destination nodes. Of course, all packets sent on a particular logical connection will go to the same physical destination port.

However, if the destination node or port goes down, and the destination host is multi-homed, the above procedure automatically ensures that a new connection will be opened to one of the ports which is not down. Since the ARPANET connection protocol is transparent to the user, the user need never know that this has happened.

It is interesting to compare this procedure (based on DNA messages) with the previously discussed procedures (based on an updating protocol similar to that used for the SPF routing algorithm). The latter procedure would ensure that all nodes always agree (except during some very short transient period) on precisely which mappings are effective and which are not. Mappings would be marked effective (or ineffective) almost as soon as they become so. There would be no need for the source nodes to probe the destination nodes by sending data packets to possibly incorrect physical addresses. The procedure we are recommending does not have these features. In the recommended procedure, different nodes' translation tables would not necessarily be in agreement all the time as to which mappings are or are not effective, and probing is necessary. This is an acceptable situation though, since the sort of universal and immediate agreement which is necessary to ensure the proper functioning of a routing algorithm just is not needed to ensure

the proper functioning of the logical addressing scheme. However, THE LACK OF UNIVERSAL AGREEMENT DOES REQUIRE THAT TRANSLATION BE DONE AT SOURCE NODES RATHER THAN TANDEM NODES, since the DNA-based procedure, while designed to keep the tables at source nodes up-to-date, will not necessarily have the same effect at tandem nodes. (That is, DNA messages are sent to the source nodes, NOT to tandem nodes.)

There is only one situation in which re-translation should be done at the tandem nodes. Suppose a logically addressed packet arrives at a tandem node, and that node, after checking its routing table, sees that the physical destination address of that packet is unreachable. If the packet is a virtual circuit packet, or the tandem node does not implement logical addressing (i.e., does not contain a translation table), the packet must simply be discarded. But if the packet is a datagram, and the tandem node does have a translation table, it should re-translate the destination logical address, and re-address the packet. This procedure can help to prevent unnecessary data loss. Note that this tandem node translation would happen only rarely, and only in situations in which it could not serve to defeat the criteria according to which the source node translation was done.

It should be noted that, with the recommended procedure

(unlike the alternative), the size of the translation table at each node is a function only of the number of authorizations. That is, only changes in the authorizations require insertions or deletions to the table. This justifies our previous claim that insertions and deletions are relatively rare events.

It should also be pointed out that nothing in this procedure prevents a computer from being multi-homed to a single node.

6 Operational and Implementation Considerations

This procedure requires each network node to maintain a full table of authorized mappings. There are operational advantages to requiring all nodes to have precisely the same translation table; this simplifies the process whereby one node can be reloaded from another in case of failure, and reduces the amount of site-dependent information that must be maintained in the nodes. (In general, the more site-dependent information there is, the larger the Mean Time to Repair will be.) We have not spoken explicitly of the way in which NCC personnel add or delete authorizations. This will require some protocol between the nodes and the NCC. This protocol would be similar in some ways to the protocol used to broadcast software patches or

packages to the nodes. However, since we want to be able to make incremental changes to the tables (rather than broadcasting an entire new table each time a change must be made), the node will have to contain routines to add to or delete from the tables. The node may have to inhibit interrupts while modifying its table, so that no translations are done while the table is in a state of flux. Also, no reloads may be done from a node whose table is in a state of flux. These last two restrictions are needed to prevent race conditions; these restrictions are easily implemented.

When the NCC makes a change to the table of authorizations, it will want to receive some sort of positive feedback, indicating that the change has indeed been made. One method of doing this is to associate a sequence number with every "add" or "delete" command. Each node could periodically report to the NCC the sequence number of the last command that it fully executed, and an entry could appear in the log whenever the sequence number is other than expected. If the nodes refuse to execute commands which are received out of sequence, this would enable the NCC to determine whether each node has received the correct sequence of commands.

If memory considerations make it impossible for each node to contain a table of ALL authorized mappings, it is possible to

get by with a shorter table. Strictly speaking, each node's table need contain only the logical addresses which map to that node itself, PLUS those logical addresses which the node's own local hosts are allowed to use. While this smaller table may take less memory, it would, however, increase the operational difficulties of table maintenance. We have not yet said anything explicit about the format of a logical address. We recommend use of a 16-bit field for coding the logical addresses. This should be enough to prevent bit-coding limitations from placing any restriction on network growth. The only other information needed in the translation table is (1) destination node physical address (8 bits should suffice), (2) one bit for the effective/ineffective variable, (3) enough bits to code the port numbers, and (4) enough bits to code any variables needed to implement the selection criteria used for multi-homed hosts. This should not take more than two or three 16-bit words per entry. If space is a problem, it is possible to shorten the tables somewhat by deleting the port numbers. Strictly speaking, port numbers are only needed by the destination nodes, and hence need not appear in each node's copy of the translation table. However, eliminating the port numbers from the common table increases the amount of site-specific information in the tables, which is a disadvantage in itself.

It goes without saying that the use of logical addressing has implications for the network access protocol. We have already discussed one aspect of the network access protocol, viz., the need for the host to send LAD messages to the source node, and the need for positive and negative acknowledgments to be returned to the host. A LAD message from a particular port contains a list of logical addresses, with an indication for each one as to whether the host wants the mapping of that logical address to that port to be effective or not. When a host declares a mapping to be ineffective, the source node must always return a positive acknowledgment, and must mark the mapping ineffective in its translation table. However, if the mapping is not authorized (i.e., not in the translation table), the source node should also return a warning to the source host, since host error is likely. When a host declares a mapping to be effective, the node will return either a positive or negative acknowledgment, depending upon whether the mapping is authorized or not. When a host declares an authorized mapping to be effective, the node must mark it so. The host should be allowed to send LAD messages to the node at any time, and they should take effect immediately.

7 Network Access Protocol Implications

The use of a logical addressing scheme also has implications on the part of the network access protocol that is used for ordinary data transport. When a source host passes a message to its source node, the message leader must indicate whether logical or physical addressing is desired (assuming the network allows both). If logical addressing is desired, the destination logical address must be indicated. The source node must be able to discard that message (with an appropriate negative acknowledgment) if there is no effective mapping for that logical address. The source host must also be able to indicate its own logical address, if it wants to make this known to the destination host. (Since the source host may have several logical addresses, it must explicitly choose one to be carried to the destination host.) Again, the source node must be able to negatively acknowledge and then discard the message if the mapping from that logical address to the source host's physical address is not effective. Alternatively, one may want to return this sort of negative acknowledgment only if the source logical address mapping is unauthorized, and allow the message to be sent if the mapping is authorized but ineffective. If this is done, a particular "logical host" may be allowed to send data, but not to receive it.

When a logically addressed message is passed from the destination node to the destination host, the message header must contain the destination logical address (since the destination host may have more than one logical address), and the source logical address, if any. This implies, of course, that the source and destination logical addresses of datagram packets must be carried across the network in the packet header. (For virtual circuit packets, the logical addresses can be kept in the connection blocks in the source and destination nodes.) The internal packet header must also carry the physical destination node number (for addressing at tandem nodes), and the physical source node number (so that DNA messages can be returned without having to rely on the integrity of the translation tables). However, these physical node numbers need not be passed to the destination host. Note that there is no need for the internal packet header to carry source or destination port numbers, since these are usually determined by the combination of physical node number and logical address. In the case where a host is multi-homed to a single node, the port numbers are not so determined, but the destination node can make a choice of ports "at the last minute," either by choosing according to one of the criteria already discussed, or by choosing the port with the shortest queue.

[1] E.C. Rosen, J.G. Herman, I. Richer, J.M. McQuillan, ARPANET Routing Algorithm Improvements -- Third Semiannual Technical Report, BBN Report No. 4088, April 1979, chapter 2.

[2] J.M. McQuillan, I. Richer, E.C. Rosen, D.P. Bertsekas, ARPANET Routing Algorithm Improvements -- Second Semiannual Report, BBN Report No. 3940, October 1978, chapter 4.

[3] E.C. Rosen, "The Updating Protocol of ARPANET's New Routing Algorithm," Computer Networks, February 1980, Volume 4, p. 11.