Jerry

M.I.T. Laboratory for Computer Science

Local Network Note No. 22
March 11, 1980

# Source Routing for Campus-Wide Internet Transport

by Jerome H. Saltzer

This note proposes that for the internet addressing layer of a campus-wide local area network, the source routing mechanism suggested by Farber and Vittal [1] and discussed by Sunshine [2] may have several advantages over hop-by-hop routing schemes based on universal or hierarchical addresses. The campus environment, as defined and discussed in local network note 21, requires many subnetworks connected by gateways, and it has a variety of other special properties of administration. The primary advantage of source routing in this environment is simplicity of implementation of the gateways that interconnect subnetworks with consequent improvement in cost, maintenance effort, recovery time, ease of trouble location, and overall management effort. Secondary advantages of source routing when applied to the campus environment include: 1) a clearer separation of physical addressing from logical naming mechanisms in protocol design, 2) elimination of stability, oscillation, and packet looping considerations, 3) ability for a source to control precisely a route so as to optimize a particular service goal (e.g., response time, reliability, bandwidth, usage policy, or privacy), 4) deferment to a higher protocol level of the detailed design of the fragmentation/reassembly strategy required to pass through intermediate networks with small maximum packet sizes, and finally, 5) the ability to accomodate both official and unofficial gateways between subnetworks.

Two disadvantages of source routing are: 1) that the route used will tend to be relatively static and therefore cannot optimize use of communication facilities as well as the potentially more dynamic hop-by-hop route selection system, and 2) route selection must be accomplished somehow, and since the mechanism to do this selection is not specified by this protocol level, some additional mechanism must be designed to provide this function. The argument made here is that the first disadvantage is not serious in an environment such as a campus, in which the low cost of high bandwidth communication can make optimization less important. The second disadvantage may be less serious than it appears when one considers that a higher-level name resolution service is required in any case, and that service can also provide route selection service. In fact it may be possible to turn this need into an advantage, since there can be more than one such route selection service, one of which is based on simple global or hierarchical network names, while another, perhaps experimental or research service, provides an elaborate interactive directory search facility or a private route pattern.

## How Source Routing Works

Source routing among a collection of subnetworks is a mechanism that comes into play at the next-to-bottom layer of protocol, sometimes called the "internet" layer. Figure one illustrates this two-layer arrangement. The bottom layer, which we may call the "local transport" layer, is a protocol for delivery of a packet within a local subnetwork such as a single ETHERNET, CHAOSNET, or L.C.S. Ringnet. Routing within the local transport protocol is usually accomplished by physically broadcasting the packet to all nodes on one subnetwork; any node that recognizes its own local transport address at the front of the packet will receive it.

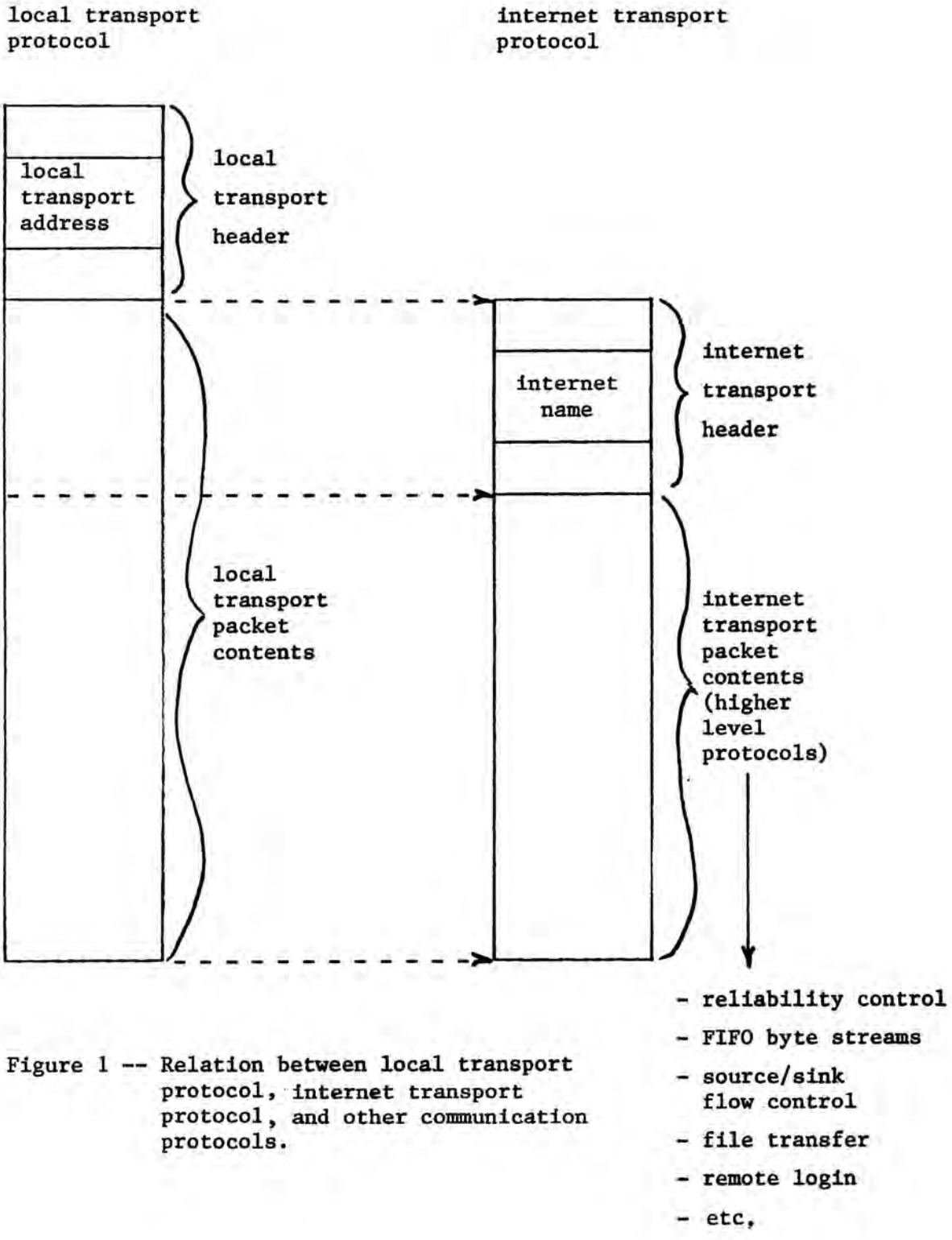


Figure 1 -- Relation between local transport protocol, internet transport protocol, and other communication protocols.

The next-to-bottom, internet layer is a protocol for delivery of a packet between any pair of nodes on the campus. One starts a packet on its way by placing the address of a gateway in the local transport address field, and what may be called the "internet name" of the target node in the internet name field. The local transport medium carries the packet to the gateway, which examines the internet name field to determine what local transport address to use to get to the next gateway. In turn, the internet name field is again interpreted by successive network gateways to determine which local transport address should be used for the next step of this packet's journey.

There have been suggested several alternatives for the interpretation of internet names. Three of these are:

1) Unstructured unique identifier. Every node on the campus-wide net has as its internet name a permanent unique identifier. Each gateway has a set of tables or other rules that allow it to determine the appropriate next step in the route to every possible named node. (Thus this approach is sometimes called "step-by-step" or "hop-by-hop" routing.) In its most general form, the unique identifier provides no routing information whatsoever. Finally, the unique identifier may be interpreted either as the name of the node or as the name of the point on the network to which the node is attached, depending on the network's convention on what happens to the name if a node is disconnected and reattached to a different place.

2) Hierarchical identifier. In this alternate form of hop-by-hop routing, the internet name of each node is a multi-part field. For example, a two-part hierarchical identifier might consist of an identifier of the

subnetwork to which the node is attached and a node number (usually the local transport address) of the node on that subnet. For this kind of internet name, each gateway has a set of tables or rules that allow it to determine the appropriate next step in the route to every possible named subnetwork. Since there are many fewer subnetworks than nodes, these tables should be much smaller than in the case of the unstructured unique identifier. Reduction in table size is the chief attraction of the hierarchical identifier, and the argument can be extended to identifiers of more than two parts, network groups, and still smaller tables. Because the hierarchical identifier contains components that are names of parts of the network, this kind of network name is almost always thought of as naming the network attachment point, rather the node that is attached to it.

3) Source route. The internet transport layer contains, instead of a network name, a variable-length string of local transport addresses, with the property that each gateway merely takes the next local transport address from the string, moves that address to the local transport protocol address field, and sends the packet on its way. With this approach, a gateway needs no knowledge of network topology, so the tables required for hop-by-hop routing vanish. A source route unquestionably identifies a network attachment point, quite independently of what node is attached to that point. Any attempt to make an interpretation that a source route identifies a node rather than an attachment point would be strained at best.

Note that if the network is arranged as a two-level hierarchy, with a single "supernet" acting as the only communication path among all the remaining

subnetworks, then the two-part hierarchical identifier taken together with the local address of the nearest gateway to the supernet is an example of a source route and the gateways can become very simple. However, the hierarchical identifier can be used even if the network topology is not hierarchical, by providing an appropriate routing algorithm in the gateways. In that case, only the final part of the hierarchical identifier might be directly usable as part of the route; even it might actually be interpreted or mapped by the final gateway.

Note also, that it is common for a single node to have several activities underway at once. For example, a time-sharing system may have many logged-in users, several of which are using the network for communication between their terminal and the time-sharing system. The receiving network software in the time-sharing system then finds that it is acting as a kind of gateway, between the campus network on the one hand and the array of activities inside the node on the other. As a result it is commonly proposed that the internet name not identify a node but rather a particular activity within that node. This proposal usually takes the form of an additional field in a hierarchical internet name, known as a "socket number" or "link". There is a controversy over what level of protocol should recognize this socket number, and how big it should be. For our purpose, it is sufficient to observe that the socket number is a kind of route for use by the receiving node.

The mechanics of operation of a source-routing gateway as a packet passes through are quite simple; this simplicity is the chief attraction of source routing. There are several alternative detailed approaches; to permit

explicit discussion one implementation will be described here.* This implementation dynamically constructs a reverse route. It works as follows:

1) The internet source route field is structured as shown in figure two, with two one-byte numerical fields and a variable (but constant for the lifetime of the packet) number of bytes of route. Each local transport address uses an integral number of bytes, typically one or two. The first count is the number of bytes in the route. The second count is the position of the next unused byte of the route. The first count remains constant for the lifetime of the packet; the second is updated at each gateway.

2) A gateway receives a packet using the local transport protocol of one network (call it network A) and wants to send it out on a second network (call it network B). For the moment, assume that a gateway interconnects exactly two nets; generalization for a multinet gateway involves a simple conceptual extension described later.

3) The gateway parses the internet source route field using the "start of next local address" count to obtain the next step of the route. (We presume that the gateway is endowed with the knowledge of how many bytes of route are required by network B.) It extracts the appropriate bytes and places them in the local transport address field for network B. Then it replaces those bytes of the internet source route with its own local transport address on network B, thus contributing its part of the reverse route. Finally, it increments the "start of next address" field by the

---

* This implementation is only a slight variation of the one proposed by Farber and Vittal.

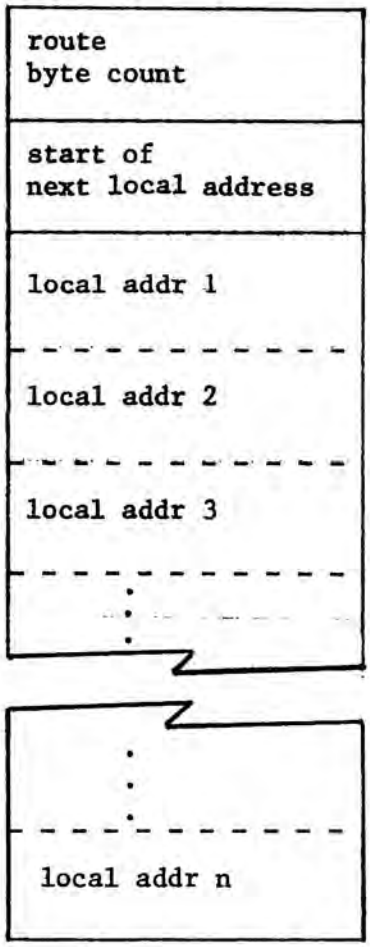


Figure 2. Possible implementation of an internet source route.

number of bytes it extracted from the route, and it invokes the local transport level to send the packet out on network B. (Note that this reverse route construction strategy assumes that all paths are bi-directional and that all local transport addresses on any single network are of the same size.)

4) If a gateway interconnects three or more subnetworks, it simply behaves as though it is itself a subnetwork with three or more gateways to other subnetworks. The next byte of route is interpreted as a local address on this hypothetical subnetwork. The reverse route is constructed as usual.

The operation described above is repeated at every gateway, and may also be repeated one or more times inside the target node to dispatch the packet to the correct activity within that node. Similarly, when a packet originates, it may go through one or more route selection steps before it actually is placed on the first subnetwork.*

## Where Routes Come From

For source routing to work, the source of a message must somehow know what route to place in the internet header of a packet before it launches the packet into the internet environment. This requirement superficially implies that every source of packets be very knowledgeable, which sounds like a terrible burden to small nodes--every node on the network would have to be

* From a viewpoint of telephone terminology, a source route system is a kind of electronically implemented step-by-step switch, with each subnetwork, multi-tailed gateway, as multi-activity host acting as a multi-position switch. However, because it is electronically implemented and thus not restricted to ten-position mechanical switches, this step-by-step switch does not have the limitations of the corresponding telephone technology.

able to create or deduce suitable routes. In fact, that implication is unwarranted--all that is really required is that every source of messages know of a place in the network to ask to obtain routes. Once a source has learned of a suitable route to a particular target, it can encache that fact and reuse it as often and as long as it wants--until the route fails to work or there is a reason for it to believe that a better route exists.

The most general form of route selection would come by providing one (or more, for reliability, quick response, or administrative convenience) routing server in the network. A routing server is a specialized node whose function is to maintain an internal representation of the topology of network interconnection (along with any useful class-of-service information about various subnetworks and gateways) and also to act as a name resolver. The desired target must, of course, have some name, perhaps the unstructured unique identifier or hierarchical identifier earlier suggested as an alternative internet name. The routing server then implements a map from internet names to routes.

There are two independent dimensions along which this routing server may be more or less sophisticated: in its name-resolution abilities, and in its route-choosing abilities. To begin with, let us assume a particular fixed, fairly simple name resolution scheme--say a hierarchical identifier--with the understanding that this choice has little or no bearing on routing sophistication. The routing choice mechanism, then, can range from a simple fixed table of routes from all possible sources to all possible targets (perhaps cleverly compressed with knowledge of the actual net topology) to a dynamic mechanism based on frequent exchanges of traffic statistics with gateways and other routing servers throughout the network.

Thus, to get started, a node that wants to originate messages needs to know one route: a route that can be used to send a request to a routing server to obtain other routes. It would be possible, though poor practice, to embed this "route to the nearest routing server" in the software of every node; a more general and flexible approach would be for a newly-arrived node to use either a broadcast or a breath-of-life strategy to discover this one route. In the broadcast strategy, a node broadcasts on its local transport network a request for the "route to the nearest routing server". For this particular broadcast route request, at least one gateway on every subnetwork is prepared to act as a rudimentary routing server. In the breath-of-life strategy each gateway periodically (say once every ten seconds) broadcasts over its local subnetwork a packet containing the route to the nearest routing server. A newly-operating node waits for the next breath-of-life packet before it can request its first route.

Having found a route to a target node, if that node carries on more than one activity it may be necessary to hold a further negotiation with the target to learn how the target wants the source to identify the particular activity in which it is interested at the target. This negotiation probably takes place by sending a rendezvous packet to the host and receiving in return a packet that contains some extra routing steps to be appended to the route originally obtained from the routing server. (Note that this protocol step is just the source-routing variation on a negotiation that takes place in every such protocol; it is <u>not</u> an extra step introduced by source routing.)

## Separation of routing and naming

The main difference between source routing and its alternatives is that the responsibilities both of route choice and of name resolution are moved from the internet gateways to some other agent. In turn, this responsibility change allows the internet transport protocol to be defined and the gateways to be implemented without freezing a particular form of network-wide naming. A commitment to a particular form of network-wide name is made in the design of the name resolution part of a routing server, and since it doesn't matter to a gateway where a route comes from (the gateway cares only that the next step works,) there can be more than one kind of name resolution going on at the same time, perhaps implemented by different routing servers. Practically, one would expect that there might be one centrally administered and widely-used naming method implemented by standard routing servers, and in addition some experimental or special-purpose routing servers developed for special applications or to experiment, for example, with interactive resolution of catalogued service names, or multi-casting protocols. These latter ideas, while likely of interest for the future, seem inappropriate to embed now in the internet transport protocol layer on grounds of inexperience. But they can be tried in the environment of a source-routing internet transport strategy without disruption and without change to the gateways. It is even possible for one routing server to have a different view of the extent of the network from others. Overlapping virtual networks are thus implementable with this strategy. This feature might be used, for example, to segregate "local" communication paths from "long-distance" paths that involve routes through external, tariffed, networks.

At the same time, the source route field format places little constraint on the format of the local transport addresses for any particular subnetwork--only that there be an integral number of bytes whose number is known by the gateway that moves the packet to that subnetwork. This flexibility means that paths can go almost anywhere: in particular they can traverse "outside" networks no matter what their addressing or internal routing strategy, so long as at the far end of the outside network is a gateway that understands how to continue the packet on its journey.

Separation of the mechanics of routing from the functions implemented by a naming or addressing system has the advantage of clarifying some frequent protocol design arguments that boil down to how much naming function should be embedded in the lowest protocol layers. For example, it is usually proposed that an extra field, for use within the target node, be carried along as part of the internet address. This field is known as a "link" field in the ARPANET, the "channel" in X.25, and the "socket" in ARPA's Internet for TCP and in the internet layer of the Xerox PUP. The argument develops over how big this field should be--just large enough to distinguish among the activities or connections a host carries on at one time, or generously large enough to distinguish among all activities or connections the host will ever carry on. The former choice takes the view that the field in question is merely the last step in a route, the latter choice makes the socket number a unique identifier, which is handling a naming function for the host, perhaps allowing it to distinguish old connections from current ones.

The source routing strategy finesses this argument in that it allows the design of the packet format at the level of the internet transport layer address to be frozen without forcing a decision about socket number size. As

many bytes of route as the target host needs to distinguish among its current connections can be included as part of the source route and learned as part of the initial negotiation with the target host using a well-known route to its negotiator. A unique identifier for a connection can be returned as part of that negotiation, and it can be included in a connection identifier field of the next higher level of protocol, to insure that packets arriving over a route are part of a current connection.

## Gateway simplicity and network maintenance

With the source routing scheme just described, a gateway makes no decisions (possibly it should check to insure that the route byte count hasn't been exceeded) and it remembers nothing after the packet goes by. This simplicity of operation and lack of memory means that one can in principle implement such a gateway with a small amount of random logic and a pair of packet buffers interconnecting two local network hardware interfaces. Such an implementation, since it does not involve a stored program, has an exceptionally simple recovery strategy: a hardware reset to a standard starting state will always suffice. In practice, at least a microprocessor would probably be used to collect statistics and respond to trouble diagnosis requests, but the basic principle that recovery is trivial remains intact.

There is one way in which a source-routing gateway is more complex than its hop-by-hop counterpart. Every packet that arrives may have a different source route size and different next step offset, so a small amount of lookup is needed to perform the forwarding operation. A related consequence is that higher-level protocols find that their headers don't always start in the same position within the packet.

To create a gateway that can sustain a through transmission rate comparable to that of the subnetworks involved requires careful budgeting of the machine cycles involved. For example, a bandwidth of 8 Mbits/sec. requires being able to pass 1000 1000-byte packets/second, leaving a time budget of 1 ms. per packet. If a 0.5 MIPS processor is used for the gateway, there must be fewer than 500 instructions executed for each packet, with the implication that whatever routing scheme is used, it must be extremely simple. The source routing approach makes meeting this budget a realistic possibility.

Maintenance is directly aided by having such a simple gateway mechanism. With little to go wrong, failures should be relatively rare and diagnosis and repair should be straightforward. Even in the case where a gateway is actually implemented by software in a node attached to two local transport networks, the simplicity of action required of a gateway means that the program required is short, the cycles required are few, and that therefore the program is not only likely to be trouble-free but also it is acceptable to embed it in the innermost part of the supervisor, where it is less likely to fail because of interference by other programs in the same node. Perhaps even more important in the case of a software gateway, the simplicity of the source-routing approach means that the software required can be quick to implement.

## Route Control

One of the more interesting opportunities that arises when source routing is used is that the node that is the source of a message can, if appropriate, control precisely the route through the internet that outgoing packets follow. This control can be applied to solve several problems, as follows:

a) Trouble location. If trouble develops in a network gateway, it will be noticed first as failure of packets routed through that gateway to arrive at their destination. Starting at any node that notices such a problem, one can route a test packet "out and back", through some set of gateways and back to the originating node. A series of such tests, tracing successive steps in the route that failed, should quickly locate the troublesome gateway. One can also imagine extending this idea to route a message into a target node and back out again, as a check on the operation of the lower levels of that node's operating system. An interesting aspect of this approach to trouble location is that any user, if sufficiently desperate, can undertake network diagnosis; trouble location is not restricted to a network maintenance center that has some particular address or special hardware.

b) Policy implementation: Some local networks may be paid for by a supporting organization that wants to have a say in their usage policy. (For example, use of the ARPA network is supposed to be restricted to government-sponsored business.) If such a network has gateways to two other networks, it could be used as an intermediate transport link on some packets flowing between those networks. If source routing is used, the node that originates a packet can control whether the packet is routed through the network in question or, alternatively, avoids that network. (Obviously, sophisticated help from routing servers is needed to actually implement such a policy, but the opportunity is there.)

c) Class-of-Service implemention. There are a variety of properties that an internet connection can have, and that may be different on different routes: error rate, transport delay, probability of wiretapping,

bandwidth. Again, assuming considerable knowledge on the part of a routing server, with source routing one can choose a route that has class-of-service properties that are tailored to the application.

d) FIFO streams. Assuming that all gateways along a given route relay packets in the same order that they are received, if the same source route is used on several packets, those packets will arrive at their target in the same order that they left the source, eliminating any need for the target to restore order in what is intended to be a FIFO stream. In a hop-by-hop dynamic routing system, FIFO delivery cannot be easily insured, so the source and target must work harder if that is the function they require.

Finally, in an inter-network environment that includes both public and private gateways, the precise route control provided by source routing seems to be a key to effective use; private gateways can be used by their owners while being ignored by everyone else; flaky gateways can be bypassed by wary users no matter what administration is responsible for them.

Other observations

There are a variety of other observations that one can make about source routes. These are, in no particular order:

1) Source routing avoids several problems that can accompany more dynamic, highly optimal routing schemes. There is no danger of packets circulating in a loop forever, so techniques such as hop counts are not needed. There is little concern for startup transients, stability, or oscillation in the dynamics of route selection. Extra traffic to

exchange traffic statistics among gateways is not involved, and one does not have to worry about the interaction between the reliability of that traffic and the stability of the network. There is no requirement that each gateway maintain a table that has a number of entries proportional to the size of the network.

2) Development of network software for a new node can take an important shortcut by assembling hand-constructed routes at first. As long as the network topology does not change faster than the software gets debugged, this technique can be used to get a primitive connection operational without the need to program a routing server protocol. For quick debugging of a new microprocessor this ease of programming the first network connection would be quite useful.

3) For certain very simple applications (e.g., trouble logging, or data collection) one could imagine leaving them permanently in place with a fixed, hand-selected route to their target. (Such an approach would have to be weighed carefully against the disruption that a change in network topology might cause. The point is that this opportunity to exploit source routing for simple applications does exist.)

4) Source routing is consistent with at least two proposed fragmentation/reassembly strategies. Fragmentation can be done by a gateway on entry to a subnetwork that has a small maximum packet size: by using the same route for all fragments of a given packet reassembly can be accomplished either at the gateway leaving that subnetwork or by the target node. Fragmentation can also be done by a fragmentation server, which might be a node whose address appears "in the middle" of a

route unbeknownst to the source, target, or intervening gateways. If it receives a packet that it believes is too large to get through some intermediate subnetwork, it can fragment that packet and also reroute the fragments through a reassembly server on the other side of the bottleneck. Finally, one might successfully finesse fragmentation completely by sending big packets over a longer or less desirable route that allows big packets, while sending small ones the short, desirable way.

5) In a manner similar to the fragmentation/reassembly servers just described, one can place other specialized servers along a route to act as filters, translators, etc. This idea has not been explored, but it seems to represent an interesting kind of opportunity.

6) Attachment of multi-tailed hosts (the "multi-homing problem") is simplified. In a complex internet installation, one might expect to find some hosts that have attachments to two or more different subnetworks of the internet, perhaps for added reliability or for assured bandwidth to services found on different subnetworks. If the several attachment points are functionally equivalent, then when another node tries to send a message to such a host, there is a question of to which one of the several attachment points the message should go. A hop-by-hop routing scheme in which gateways interpret internet names would require that either the different attachment points be assigned different internet names (so the originator has the burden of choosing which internet name to use) or else a single internet name is used for all the attachment points of the multi-tailed host and the gateways add this topological fact to their storehouse of routing knowledge and make the choice on the

fly. With source routing, the burden of choice can move to the routing server, where the topological information is available to choose a route from the originator to the nearest attachment point of the multi-tailed host. Neither the originator nor the internet gateways need realize that the target has several attachment points.

In this last case, as in some others, one can argue that some of the apparent simplifications or advantages obtained by using source routing are actually only shifts of the underlying problem over to the routing server. This argument has some validity, but it overlooks two points:

1) Separation of two tangled problem areas, naming and routing, into two distinct and largely independent mechanisms simplifies and clarifies design, algorithms, and code.

2) When one implements routing as a service supplied by a server, it becomes possible to introduce variations on the service by changing just the server, or providing an alternate server. When the function of routing is distributed among the gateways, changes in the service require changing all of the gateways, an undertaking that is more difficult and hazardous.

## Conclusions

The premise of this note is that source routing is particularly well-suited to the campus environment. The argument goes as follows: in the campus environment, one can install high bandwidth lines at low cost, since reliance on common-carrier offerings is not required and physical facilities are under common control. This high bandwidth permits using strategies, such

as source routing, that may waste some part of the communications capacity by not being optimal. The campus administrative environment calls for diversity in protocol, for which source routing caters by providing a lowest campus-wide transport protocol with a minimum amount of predetermined function that might constrain higher level protocol choices. The campus administrative environment also calls for diversity in administration, for which source routing caters by permitting precise control of complete routes for particular messages, and multiple strategies for resolving service names or network addresses, as required. It also permits messages to flow through an internetwork arrangement despite some of its topology not being centrally planned. Source routing allows particularly easy trouble location and source routing gateways are exceptionally simple, two properties that are important when one assumes a central administration that must be cost-conscious or even under-funded. Thus, from these arguments one can conclude that, at least for the campus-wide internetwork case, source routing is an attractive scheme well worth considering.

Acknowledgements

This note records a series of intensive discussions with David Reed, David Clark, Kenneth Pogran, and Noel Chiappa. It also borrows ideas and terminology from working papers of the ARPA internet project by Dan Cohen, Jon Postel, David Clark, and John Shoch and from working papers of the M.I.T. AI Laboratory Chaosnet project by David Moon. Welcome comments on early drafts were made by Dan Cohen and John Shoch.

## References

[1] Farber, D.J., and Vittal, J.J., "Extendability Considerations in the Design of the Distributed Computer System (DCS)," Proc. Nat. Telecomm. Conf., (November, 1973), Atlanta, Georgia, pp. 15E-1 to 15E-6.

[2] Sunshine, Carl A., "Source Routing in Computer Networks," Computer Communication Review 1, 7, (January, 1977) pp. 29-33.