MIT/LCS/TR-270

THE DESIGN OF A ROUTING SERVICE FOR CAMPUS-WIDE INTERNET TRANSPORT

Vineet Singh

This blank page was inserted to preserve pagination.

The Design of a Routing Service for Campus-Wide Internet Transport

Vineet Singh

August 1981

This research was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by the Office of Naval Research under contract number N00014-75-C-0661.

Massachusetts Institute of Technology
Laboratory for Computer Science
Cambridge, Massachusetts 02139

This empty page was substituted for a blank page in the original document.

The Design of a Routing Service for Campus-Wide Internet Transport

by

Vineet Singh

Submitted to the
Department of Electrical Engineering and Computer Science
on August 26, 1981 in partial fulfillment of the requirements
for the Degree of Master of Science

Abstract

A campus-wide network requires many subnetworks connected by gateways and it has a relatively loose administration. Modularization of network implementation is important in this environment to make efficient use of ever-improving technologies and protocols. The need for modularization makes it desirable to separate a routing and target identification scheme from gateway implementation—a facility that source routing provides. Moreover, removing routing and target identification responsibilities from the gateways leads to their simplicity and, therefore, a better chance that gateways will not be bottlenecks in the high-bandwidth network. This thesis focuses on the design of a Routing Service to support source routing in the campus environment.

The Routing Service is designed to find paths from a requester's attachment point to a node specified by the requester. The Routing Service accepts hints from the requester about the destination node's location in the network to limit the search involved. The Routing Service also provides user-control of paths and diagnosis for faulty paths. The design of the Routing Service places strong emphasis on scaleability with respect to the size of the network. Reliability and simplicity are two other key features of the Routing Service.

Key Words: computer networks, routing, servers

This empty page was substituted for a blank page in the original document.

Acknowledgments

I am indebted to Professor Jerry Saltzer for his constant encouragement and for the numerous enlightening discussions that I was privileged to have with him. I would also like to thank all the members of the Laboratory for Computer Science, especially the CSR people, for providing an environment that was both friendly and conducive to research. Finally, I would like to thank Kathy Wood for painstakingly proofreading the entire manuscript several times and for typing most of this thesis at whirlwind speeds. (I figure in the list of the ten worst typists of the decade.)

To my Mother and my Father

Table of Contents

Table of Figures

Chapter One

Introduction

Computer communications networks are beginning to play an increasingly important role in society. A major motivation for networking is the need to share resources related to communication facilities, computing facilities, and information. Going by present trends it is expected that, in the not too distant future, almost everyone in society will have access to some sort of computer-based network to cater to his communication, computing, and information needs. Present and envisioned applications of these networks include such diverse fields as electronic mail, office automation, health care, computerized commerce and management systems, military applications, news, and education. In fact, information processing is expected to be the backbone of future societies. Current projections indicate that more than half of the U.S. economy will be based on activities related to information processing by the year 2000.

Pioneering work in the area of computer networks was initiated by the design of the ARPANET. Since then, the field of networking has grown to the level where it is now a major preoccupation of scientific and technological endeavors. Research in the area has indicated that different types of networks or network architectures are suitable for different sets of applications. For example, radically different network characteristics are suitable for a long-haul network like the ARPANET, a packet radio network like the PRNET, and a local area network like the ETHERNET. Similarly, a campus environment, with its special characteristics, has its own set of requirements for a data communication network.

A campus-wide network will extend beyond a single building, but it permits low cost, high-bandwidth transmission media to be installed because the network exists within

a single cohesive political and administrative organization. Within a campus, several thousand nodes (i.e., data sources and data sinks) will require interconnection. Since local area network technologies like the ETHERNET can only support tens through hundreds of nodes and can extend for only thousands of meters, these technologies are not sufficient to support a campus-wide network. It is expected that a campus-wide network will consist of a number of local networks, interconnected by gateways with a group of nodes being attached to each local network.

To maintain high-bandwidth communication spanning one or more gateways, it is necessary for the gateways to be fast; otherwise, the gateways will be communication bottlenecks in the campus-wide network.

Another key feature of the campus environment is the diversity of communication technologies and protocols likely. Also, since these technologies and protocols are constantly improving, the network should be designed to evolve with them. Modularity of network functions is crucial to achieve the objective of adaptability to new network characteristics.

A combination of the need for simpler gateways and the need to modularize network functions is the major motivation for using source routing in the campus-environment. Source routing allows target identification and routing decisions to be removed from the responsibility otherwise assigned to gateways. This redistribution of responsibilities, in one stroke, accomplishes the aim of simpler gateways and modularity (to the extent that target identification and routing schemes need not be tied with gateway implementation).

For source routing to be successful, one cannot allow target identification and routing responsibilities to fall on individual nodes. Target identification and routing for the entire network, or even for the subset of the network a node may be interested in, may be too complex for simple nodes (like microprocessors) to undertake. A more reasonable approach is for network-wide services to cater to the target

identification and routing needs of network users. A network-wide service also leads to economy of scale through sharing. Forcing each node to implement the service within itself may, on the other hand, lead to a lot of redundant processing. The purpose of this thesis is to design a network-wide service to cater to the need for routing information.

1.1 Goals

The Routing Service described in this thesis was designed keeping certain goals in mind. A short description of each major design goal is given below. First, since messages in a computer network contribute significant delays (even in a high-bandwidth network), it is desirable to minimize the delay due to messages required for the correct functioning of the service. Second, the Routing Service should be reliable—robust in the face of arbitrary changes in the network. Third, the service should be reasonably fast—fast enough to avoid being a bottleneck. Fourth, the service should avoid any unnecessary dependencies between its functioning and other network functions. Fifth, the service should scale gracefully for larger networks. Sixth, the routing server should maintain as good a user interface as possible, keeping the other goals in mind. Additional goals ensure that the Routing Service faces up to changing network configurations[1], mobile hosts, artificial partitioning, multi-homing, and shared access.

1.2 Related Work

Telephone systems until two decades ago used source routing to connect a dialing line to the dialed line. Dial pulses would be followed to physically connect appropriate switching devices to complete the connection (i.e., the numbers dialed were used to select an appropriate route through the switching network). In fact, a

[1] This refers to the hierarchical configuration of the network that is used to make the service scaleable. The configuration is subject to change.

primitive sort of "Routing Service" was also in existence. If a given number routed a call from the east coast to the west coast via Chicago and the operator had reason to suspect that the connection to Chicago was faulty, then the operator might try another number that would route the call via Detroit instead of Chicago. In the campus-wide network, the switching devices correspond to gateways, the telephone numbers correspond to a source route, and the operator corresponds to the Routing Service. Clearly, there is a limit to which the analogy can be pursued, mainly because the "Routing Service" in the telephone system context was human whereas the Routing Service in the campus-wide network runs on computers.

Source routing has also been used in the ARPA packet radio network (or PRNET) [10]. Packet radio technology enables packet switching, which has been used for point-to-point communication lines, to be applied to broadcast radio also. The development of packet radio technology was directly motivated by the need to provide a communication network for terminals and computers in motion. Broadcasting is a natural way to avoid the need to control rapidly changing routes. PRNET also offers point-to-point routing that can be used when routes are not expected to change very rapidly; source routing is used to support this point-to-point communication.

There are two main differences between routing as implemented in the PRNET and as it will be implemented in the campus-wide network with a Routing Service. The first difference relates to the methods used for collecting topology information. A station in the PRNET is the entity that collects topology information and computes routes; there may be several stations in the PRNET. Each packet radio in the network periodically announces its existence by transmitting to each station its PR neighbor table. A PR neighbor table for a packet radio typically contains information on which other packet radios can be heard along with information about the quality of the links. In the campus-wide network, a similar approach would imply that nodes and gateways in the network periodically send topology information to the Routing Service. Since gateway simplicity is one of the major motivations of using source

routing with a Routing Service, even this function has been removed from gateways (and nodes). The Routing Service in the campus-wide network assumes the responsibility itself of contacting nodes and gateways to gather topology data.

The second major difference between the point-to-point routing strategies of the PRNET and the campus-wide network concerns scaleability. The PRNET point-to-point routing strategy is not designed to scale gracefully for large networks. Each station has to be aware of all operational packet radios in the net and each station must also compute all routes. The Routing Service for the campus-wide network, on the other hand, is designed so that it actually consists of a hierarchical structure of several Routing Servers. Any given Routing Server has only to gather a limited amount of topology information and compute a limited number of routes. A number of Routing Servers may have to cooperate to find a given route. The hierarchical structure of Routing Servers is designed to scale very gracefully as the network size increases.

1.3 Outline of Thesis

This thesis is only a paper design of a Routing Service—no implementation has been attempted. The lack of an implementation is partially compensated by a comprehensive evaluation of the paper design.

Chap. 2 describes the campus environment and the suitability of source routing for campus-wide internet transport. The function of a Routing Service is explained and a set of requirements laid down for the design of the Routing Service. Chap. 3 begins by addressing questions about the configuration of the campus-wide network, an implementation of source routing, and the kind of routes to be computed. Next, each function of the Routing Service is described together with the method used by the Routing Service to achieve it. Chap. 4 evaluates the design of the Routing Service. Each requirement from Chap. 2 is examined in turn to see how it influenced the

design. Finally, Chap. 5 gives a summary of the Routing Service design and the areas for further improvement and research.

Chapter Two

The Requirements for a Routing Service

Before the requirements for a Routing Service can be discussed, it is necessary to look at some of the important properties of a campus environment and to look at an appropriate mechanism to support internet routing.

Section 2.1 describes the campus environment in order to gain some insight into the choice for an internet routing mechanism. Section 2.2 explains why source routing is a good idea in a campus environment. Section 2.3 describes the function of a Routing Service and section 2.4 lays down some basic requirements for a Routing Service. Finally, section 2.5 looks at some advanced problems that will be encountered in a large, multi-network system and section 2.6 advocates a more flexible approach that should be taken to solve these problems.

2.1 The campus environment

The campus environment has been characterized in considerable detail in [18]. A network in a campus environment will typically span several buildings and will not be subject to a strong central administration. The properties that characterize a campus-wide network could equally well apply to a network at a corporate site, a government complex, etc.

2.1.1 Allows installation of low-cost, high-bandwidth transmission medium

The key property of a campus-wide network is that although the network is loosely administered, it does lie completely within the domain of one political body. This property, along with the limited geographical extent of a campus-wide network,

permits installation of a low cost, high-bandwidth communications medium throughout the network. This is in contrast with a long-haul network like the ARPANET that must resort to transmission over a common carrier.

2.1.2 Local network technologies will not work

A campus-wide network is expected to have several thousand nodes and, therefore, local interconnection strategies like [13, 14, 4, 23, 7, 24], which can support only tens through hundreds of data node interconnections, are not feasible solutions. Moreover, the geographic scope of local networks is limited to a restricted area such as a building or a cluster of buildings. Hence, a local network cannot cover a very large campus. A campus network will have several different local networks connected together by gateways. See Fig. 2-1 for a view of a typical campus-wide network. The part that is enclosed by dotted lines is the campus-wide network. An actual campus-wide network will, of course, have many more local networks. Ring Net, Ethernet, Chaos Net, and Cambridge Digital Communication Ring are the names of local network technologies; ARPANET, TELENET, and TYMNET are the names of long-haul networks.

2.1.3 Diverse technologies and protocols likely

As mentioned before, the network will have a loose administration. This, along with the fact that there is an extremely diverse range of technologies available in the computer communications world today, will ensure a similar diversity to be reflected in the campus network. In fact, the protocols used in a campus network are likely to be just as diverse. The reason for this diversity is that there is no consensus yet in the protocol community on how protocols should be layered and how the various functions should be divided. However, there is also likely to be a push towards commonality in the campus-wide network despite pressures leading to diversity. The reason for this push towards commonality is the desire to facilitate communications between any two nodes in the campus-wide network.

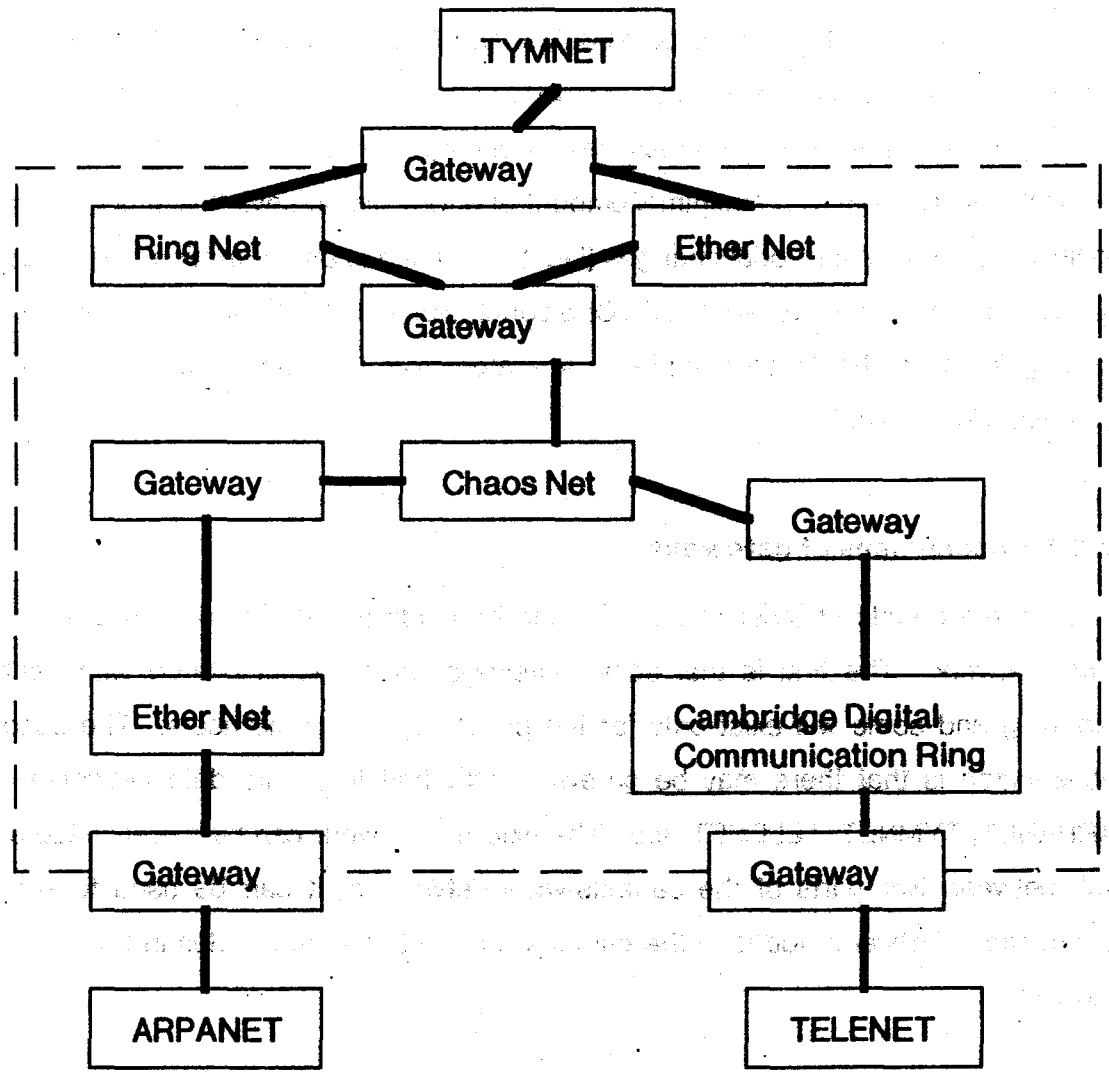

Figure 2-1: A typical campus-wide network

Due to the diversity likely in a campus-wide network, it is a good idea to provide, at the lowest layer possible, a campus-wide protocol that unifies all the diverse protocols below it and provides a uniform interface to all the higher levels of protocol.

2.1.4 Technologies in a state of flux

Not only is there a diversity of communications technologies and protocols, these technologies and protocols are also in a state of flux. Ideally, a campus-wide network should be able to evolve with these ever-changing technologies and make the best possible use of them with a minimal amount of effort. To achieve this, it is necessary to limit the effect of any local changes (i.e., to modularize each function performed in a campus interconnection strategy). One conclusion of this line of thinking is that the routing function should be completely liberated from the addressing involved in a campus-wide network.

2.1.5 Special uses of gateways

There are a couple of other observations to be made about gateways in a campus-wide network. The first is that some gateways will be administered with central planning and some will exist only for the private use of some users. The second observation is that there may be gateways attached to public data networks like ARPANET, TYMNET, TELENET, etc. The external network can be merely used as a link between two parts of the campus-wide network or it can be used to set up connections between nodes in the campus-wide network and nodes in the external network.

2.2 Source Routing for a Campus-Wide Network

This section first describes how internet routing works in general and then goes on to describe the mechanics of source routing. Finally, this section also discusses some advantages of source routing in a campus environment.

2.2.1 How does internet routing work

Source routing has been proposed in [18] as the mechanism to support internet transport. Source routing among subnetworks takes place at a low level in the hierarchy of network protocols. It makes use of the local transport layer that is below it in the hierarchy. The local transport layer is used to transport packets over local networks such as the Ethernet or the Ring network. Packets that have to be sent over more than one local network have an internet target identifier attached at the front of the packet. All the gateways along the way will refer to the target identifier (and will perhaps look up some routing tables) to determine the next part of the route.

The target identifier can be of different types. It can be an unstructured unique identifier. In this case, every node on campus has a unique identifier as its target identifier. Every gateway has in its possession a routing table that contains information on the next part of the route for packets bound for every destination in the network. This approach is also called "step-by-step routing" or "hop-by-hop routing" because each gateway decides the next "step" or "hop" in the path.

Another approach is to treat the target identifier as a hierarchical address. One possibility is to have two fields in the hierarchical address—one for the subnetwork to which the node is attached, and one for the node itself. Now, every routing table only contains information on the next "step" or "hop" towards every possible subnetwork in the network. The main attraction of this scheme is that the table size is reduced. Hierarchical addresses (for target identifiers) with more fields lead to even shorter tables.

2.2.2 The mechanics of Source Routing

The approach that source routing takes is one in which the target identifier is replaced by a variable-length string of local transport addresses. When the packet first leaves the source node, the first local transport address (if not the destination address itself) is treated as the address to the first gateway on the path. The second

local transport address is used as the address on the next local network to which the packet should be sent and so on. The rest of the variable-length string gets interpreted similarly along the path until the packet eventually reaches its destination. The important thing to notice is that the routing tables needed by gateways in previous schemes just vanish. The gateways are "commanded" by the appropriate part of the variable-length target identifier to choose the correct next step. Target identification decisions are no longer made at gateways. Therefore, the gateways can now be extremely simple and it is less likely that they will be bottlenecks in a high-bandwidth campus-wide network. One implementation of source routing that dynamically constructs reverse routes is described in [18].

2.2.3 Where do routes come from?

The next logical question to ask is, "Where do routes come from?" It was explained before that the source node places a variable-length string or source route in front of internet packets but it has not been described yet where the source route comes from. There are two ways of solving this problem. The first way is to have every node in the network compute the routes to every other node in the network. This would be a crushing load on small nodes and it is not required. The second way is for each node to know how to contact some place in the network that knows the routes. Once a route has been found, it can be encached and used until it no longer works or until a better route is found.

This situation calls for a Routing Service in the network. Using a network service for routing to be shared by each node in the network brings the benefit of economy of scale. It will be the function of the service to maintain an internal representation of the network and to act as an identity resolver and a routing information dispenser.

2.2.4 The advantages of Source Routing

The advantages of using source routing in a campus environment are described below.

1. The main advantage is that it provides an opportunity to separate target identification and routing. One consequence of this is that gateways are a lot simpler. Also, the modularization of network function provided by source routing is a big asset in itself. It allows gateways to be implemented without having to fix any network-wide target identification. Therefore, Source Routing allows the coexistence of several different experimental routing policies with different target identification schemes. Also, since target identification is no longer associated with an internet packet's route, paths can lead anywhere. They can traverse "external" networks or they may traverse a "virtual path" inside a node to identify a "socket" or a particular "process".

2. Another important observation about a source routing strategy is that it allows a source to precisely control the path that a packet is going to take. This control can help in several ways.

   a. **Trouble location:** If a path is faulty but it is not known which part of the path is the culprit, it is easy to pin down the blame with a source routing scheme. A test packet can be sent up to some part of the route and then back. If this is done several times, each time with a different chunk of the route, the part of the route at fault can be pinpointed.

   b. **Class-of-service implementation:** An internet connection can have several properties of interest to the end users—error rate, transport delay, bandwidth, security rating, etc. It is possible to choose a route with the properties desired if the Routing Service is sophisticated enough.

   c. **Policy implementation:** Again, the precise control of routes implies that certain routes may be selectively used and some policy can be enforced to decide when they are used.

   d. **FIFO streams:** If it is assumed that packets are routed by gateways in the order in which they arrive there, then packets reach the destination in the order in which they left the source (unless of course, some do not reach at all due to some reason). This property means that FIFO streams can now be easily

implemented.

e. The precise control on routes helps in several other ways described in [18]. Looping of packets is no longer a problem; fragmentation/reassembly strategies are easily implementable, and multi-homing is less of a problem at least for gateways (because the Routing Service can take care of it).

3. Another important property of source routing is that since a source route is a variable-length string of local transport addresses, it can be spliced together from several different parts. Each part can be computed separately, perhaps by servers that only know part of the topology of the network.

2.3 What exactly does a Routing Service do?

There is still a lot of confusion as to what names, addresses, and routes are supposed to mean in a network. One convenient way to look at some network functions, including the function of a Routing Service, is to view them as name resolution functions. This section describes the function of a network Routing Service as one of several name resolutions required to be able to set up a connection with a network service.

2.3.1 Confusion over names, addresses, and routes

Shoch [20] attempted to give clear and concise definitions of names, addresses, and routes. However, there is still a lot of confusion about the exact meanings of these terms. Saltzer [19] tries to explain in his paper how most of the confusion is generated by tight associations between network objects and the common ways of referring to them.

Shoch defined a name as something that identifies what you want, an address as something that identifies where it is, and a route as something that tells you how to get there.

2.3.2 The four important network entities

Saltzer's terminology will be followed here. According to Saltzer, there are basically four important network objects.

1. **Services and Users:** These services are functions available on the network and the users are the clients that use the services.

2. **Nodes:** Nodes are computers that can run a service or run user programs.

3. **Network attachment points:** These are the electrical connectors of a network. Nodes are attached to the network through network attachment points.

4. **Paths:** Paths are the routes between network attachment points. A path is given in terms of the nodes and communication links (or gateways) on the way from one network attachment point to another.

2.3.3 Naming requirements in terms of bindings

Each of the network objects mentioned before can have a name. The bindings possible among these four types of objects are listed below.

1. A service can be run at several different nodes and has an identity independent of the node.

2. A node may be connected to several network attachment points and has an identity independent of the network attachment points to which it is connected.

3. A network attachment point can have several paths from it to another network attachment point and both have identities independent of the paths between them.

Therefore, the bindings that must be made in order to send a message to a service are:

1. Finding a node on which the service is running,

2. Finding a network attachment point for the node, and

3. Finding a path from the source network attachment point to this attachment point.

2.3.4 The function of a conventional Routing Service

The service that performs the name resolution described in the third part above is normally referred to as the Routing Service. The first two bindings are done by a service name resolution service and a node name location service respectively.

Since there are several choices at each level of name resolution, it may be necessary to backtrack and choose a different binding at a previous level if it is found useful in certain circumstances.

2.3.5 A slightly different Routing Service

The Routing Service described in this thesis will be somewhat different in function from one that only identifies paths leading from the network attachment point of the requester to another specified network attachment point. The Routing Service will now attempt to identify a path from the requester's attachment point to a node specified by the requester. However, the Routing Service will accept hints from the requester about the network attachment points to which the destination node may be connected. (A node name location service (or name server) may be used to find the names of the network attachment points to which a node is connected.)

Some of the basic requirements that a Routing Service, which performs the function described above, should satisfy are described below.

2.4 The basic requirements for a Routing Service

Some basic requirements for a Routing Service in a campus environment are described below.

2.4.1 The Routing Service has to work in a distributed environment

All nodes in the network are pretty much autonomous and independent entities and they communicate with each other by sending messages over the network to which they are connected. This structure must be reflected in the design of the Routing Service. Although a campus wide network is expected to have a lot of inexpensive bandwidth, sending messages over the network can still be quite expensive in terms of the amount of time taken for a message to travel from one end of the network to another. The number of messages that a Routing Service must send and receive for it to function properly should be kept low especially when this directly affects the time that it takes to respond to users.

Although local network routing and long-haul network routing also operate in a distributed environment, significant design differences can arise because bandwidth in the campus environment is a more scarce commodity than in the local network case and a less scarce commodity than in the long-haul network case.

2.4.2 The Routing Service should be reliable

This requirement might be partially met by keeping the Routing Service as simple as possible. Therefore, in the design of the Service an effort will be made to keep the "extras" out. If at all any "extras" are retained, they will be those that will be extremely hard to incorporate in the design once the Routing Service is implemented. Those "extras" whose design would require an effort quite orthogonal to the current design effort are most likely to be discarded. Simplicity in the design of the Routing Service is also likely to effect an improvement in maintenance cost, recovery time, trouble location, and so on.

Reliability of the Routing Service also means that it should stand up to any changes in the network topology or its connectivity. Reliability is a necessary requirement even in the local network and long haul case but the kinds of changes possible and their frequency are different in the campus environment. These changes are listed

below.

1. A gateway, subnetwork, or node breaks down or comes up again.

2. A new node is installed or removed from the network.

3. A new gateway or subnetwork is installed or removed from the network.

Therefore, routes should not be established only once at the time of network installation but rather the Routing Service should be capable of updating routes based on new information. However, it is entirely plausible that some changes may be such rare events that it might be wiser to just restart the Routing Service instead of building into it the ability to respond sensibly when such a change does occur. For example, a new gateway or subnetwork is not likely to be installed every day and, therefore, it is is not at all necessary for the Routing Service to be incrementally responsive to such changes.

2.4.3 The Routing Service should be reasonably fast

It was mentioned before that one of the advantages of Source Routing is that it makes gateways almost trivially simple. This simplicity might help to make good use of the cheap and abundant bandwidth available. If Routing Service were inordinately slow, it would wipe out all the advantages of trying to speed up routing through gateways. In fact, this brings up another point. In the design process, the option will exist at several stages to make the Routing Service vary in sophistication along various axes, e.g. the information gathered by the Routing Service may range from just topology or connectivity information to detailed class of service information about the traffic, gateways, subnets, etc. Another option that can be exercised is the amount of computation required to compute "good", "better", or "best" routes. In all these cases, it should be remembered that the philosophy behind a campus environment and a long haul environment (like the ARPANET) is most drastically affected by the fact that in the first case there is lot of bandwidth to utilize and in the second case bandwidth is a critical resource. Therefore, there is no need to squeeze

out the last ounce of bandwidth in the campus environment. The Routing Service can afford to be sub-optimal in conserving bandwidth if in the process we have bought ourselves simplicity, or we have decreased the time that it takes the Routing Service to process queries. To sum up this requirement, the Routing Service should be designed so as not to be a bottleneck in the campus environment.

2.4.4 The Routing Service should require minimal support from the rest of the system

This requirement is mainly to ensure that the opportunity to modularize the routing function is taken advantage of. As mentioned before, target identification and routing can be separately performed in the campus environment. However, it must also be ensured that no built-in dependencies creep into the design of the Routing Service. Moreover, trouble location and recovery are facilitated by keeping the dependencies low. Distributed systems, in general, have a potential for being more reliable than other systems based on central processors. Making the Routing Service self-supportive to as great an extent as possible will go a long way towards making the Service robust and modular.

2.4.5 The Routing Service should scale gracefully for larger networks

A serious attempt should be made to provide scaleability in performance (in terms of response time, reliability, etc.) for the Routing Service.

It is claimed that distributed systems are intrinsically more reliable than centralized systems. This claim is only justified if the system is designed to exploit the existing potential. If, for example, the system is designed to scale gracefully, redundancy can be used to make the system more reliable.

It is entirely plausible that the campus-wide network may grow much too large for one Routing Service to handle efficiently. One approach to this problem is to partition the network into smaller units that single Routing Servers can handle. There may be

other good reasons, in fact, for the network to be partitioned into several smaller units. Much like telephone zones, it is likely that there will be zones consisting of several adjacent subnets where most of the traffic originating in those zones will be directed to nodes within the respective zones. It is quite wasteful in this situation to force the Routing Server in one such zone to maintain information on all the nodes in the network. It is also not wise to compute routes to all the nodes in the network if, for example, 98% of the routes to nodes outside a zone are not used at all. There is another scenario in which it makes good sense to partition the network into smaller units. Consider the case of two large campus-wide networks—one in M.I.T. and another in Harvard—connected together by exactly one gateway. It clearly does not make sense to require the Routing Service in M.I.T.'s campus-wide network to know about the Harvard network in any intricate detail if all the messages from the MIT network are going to go through a single gateway connecting the two networks anyway. Moreover, the Harvard administration may not want outsiders to know about the innards of their network. Now that a good case for partitioning the network has been constructed, how can the system actually function with a different Routing Server for each region?

Fig. 2-2 shows a network partitioned into ten different regions. Each region consists of a number of subnetworks. Different regions are connected by any number of gateways. It is useful to consider the general case of a network in which some region is isolated from the rest of the system. The network may be designed to be completely connected but failures may cause regions to be isolated temporarily. Also, assume that each region is administered separately by a Routing Server and that the ten regions are administered by one higher level Routing Server. The Routing Server for region 5, for example, will only compute routes from nodes within 5 to any other node in 5 and the higher level Routing Server will only compute routes in terms of the smaller regions that make up the network. The higher level Routing Server may decide, for instance, that all routes from attachment points in region 4 to other attachment points in region 3 should go from region 4 to region 5 and then to region 3.

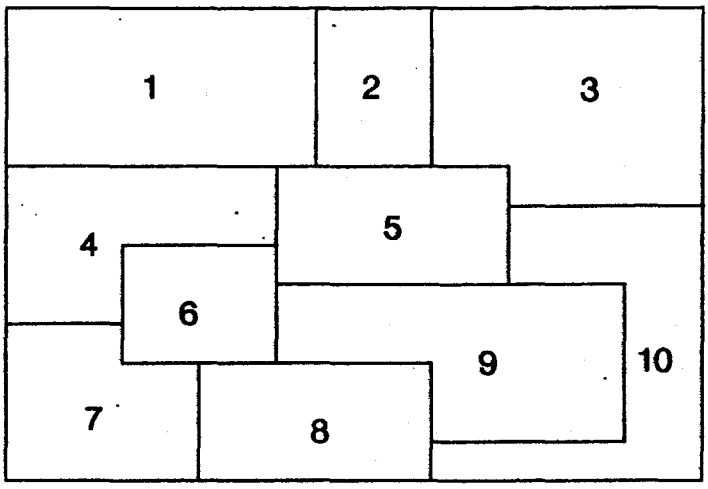

Figure 2-2: The campus-wide network partitioned into ten parts

Now, consider one way to handle routing queries in such a system. Suppose a node (call it 'abcd') connected to an attachment point in region 5 wants to communicate with a node (call it 'efgh') that is connected to an attachment point in region 7. The first node (i.e., the source node) will query its Routing Server to find out the route to 'efgh'. (The query should include the network attachment point name of 'efgh'[2]. The network attachment point name of node 'efgh' should have enough information in it to let the Routing Server know that the attachment point lies in region 7. For example, the attachment point name could be hierarchical.) Since the network attachment point to which 'efgh' is attached does not belong to region 5, the Routing Server of region 5 will pass on the routing query to the higher level Routing Server. The higher level Routing Server may then decide that the best route from region 5 to

[2]As explained before, a name server is invoked to map node names to network attachment point names.

region 7 should go via region 6. The higher level Routing Server will also ask the Routing Servers of regions 5, 6, and 7 to construct the parts of the routes that pass through their territory. After the complete route has been assembled by the higher level Routing Server, it will pass the route to the Routing Server of region 5, which, in turn, will pass it on to the node that originated the query. The problem is to coordinate the whole process so that it works in all situations and also to generalize it to the case where there may be several levels of Routing Servers.

2.4.6 The Routing Service should maintain a good user interface

As far as possible, the Routing Service should pamper the user. For example, suppose that the Routing Service maintains class-of-service information about subnets and gateways. The user should be able to inspect the class-of-service information about the gateways and subnets on any route. If, for example, the user wanted to get a route that did not pass through a certain gateway because its class-of-service information did not meet the user's requirement, it should be possible to do so.

However, having a good user interface will only be a secondary objective. This requirement will lose out if it conflicts with the requirement of reliability or of the Routing Service being fast, etc.

2.5 Some additional requirements

Some very basic requirements for a Routing Service were described in the previous section. There are some additional requirements for a Routing Service that arise mostly from concerns about large networks.

2.5.1 Towards a flexible meaning for network attachment point names

To allow for any communication in a computer network, one must be able to name the nodes of interest and their network attachment points before being able to find routes to them. This is usually not a problem in a small computer network where it is possible to keep track of remote nodes fairly easily. For example, if communication is restricted to a local network over which the normal way of sending messages is by broadcast [4, 13], then only unique identifiers need to be used as names for nodes; no names are required for network attachment points unless one node can be attached at two different points on the same local network. It is not necessary to know the connection point of any node and packets can be sent with the unique identifier of the destination node at their front. As the packet goes by each node in the local network, the destination node (i.e., the node whose unique identifier matches the destination unique identifier of the packet) picks up the packet. Things are a little more complicated as the size of the system increases. It is not as easy for each user or application program to know about the complete system configuration. Moreover, if nodes are allowed to be mobile and the configuration of the system is flexible, it is difficult to know the exact name of the network attachment point of a destination node. This situation demands a more flexible meaning for names of network attachment points in the network.

2.5.2 Hierarchies to combat problems of scale

Hierarchical methods are used all the time to keep problems associated with large systems manageable. The essential idea is that if the number of attachment points grows very large, it becomes useful to divide the attachment points up into several regions to simplify naming of network attachment points. There is a server in each region to take care of routing within that region. Each network attachment point now has a hierarchical name that consists of the identifier of the region that it belongs to and the identifier of the node within the region. Routes will now have to be computed from region to region and within regions. It is no longer necessary to compute routes

between every pair of network attachment points. The same approach can, of course, be applied recursively and there will then be several levels in the hierarchy.

There are two main motivations for using hierarchical names for network attachment points. The first is that routing becomes a lot simpler. The second reason is that with hierarchical names, the authority for assigning or acquiring names can be distributed.

2.5.3 Problems....

Some of the problems associated with using a strict hierarchy for network attachment point names are described below[3].

One of the most obvious problems is that routes are less optimal when a hierarchy is used. This may happen because one or more levels of detail are skipped over while computing routes that span more than one region. The other problems are described below:

1. **Changing Network Configurations**: A large network will be broken up into several regions on the basis of expected and observed traffic patterns. Regions are chosen so that most of the traffic originating from them is directed to attachment points within the same region. This is done because inter-region communications is more costly than intra-region communication. Another reason for choosing regions may be for administrative reasons. Due to a change in any one of the reasons mentioned above, it may be necessary to change the hierarchy of attachment points. Yet another reason for changing configurations may arise if a region grows too large or too small due to a lot of nodes being moved in or moved out. It may then be necessary to either split up a region that has grown too large or it may be necessary to merge together two adjacent small regions. For all the reasons cited above, it is not advisable to fix a hierarchy for the network once and for all.

2. **Mobile Hosts**: It is possible that a node may be moved from its current attachment point and attached at some other point in the network. In this

---

[3] [21] also discusses some routing problems associated with large, multi-network systems

case, it is necessary to ensure that the nodes preserve their identity after the change in connection points. In a strict hierarchy, the mobile host has to inform all other nodes in the network interested in it about the transfer and all ongoing computations must be able to change the hierarchical address of the mobile host they refer to.

3. **Artificial Partitioning**: It is possible to envisage a situation in which the route to the server of a region may not work due to a faulty gateway. Therefore, it may not be possible to communicate from outside with any of the nodes in the region although routes may actually exist to some of the nodes. This is a case of partitioning being artificially forced on the network.

4. **Multihoming**: A node may be connected to more than one place in the network. These connection points may be in different regions of the hierarchical network and, therefore, different routes exist to these various connection points from other parts of the network. There are two ways to handle the problem of a source node desiring to find the best route to a destination node that is connected to more than place. The first way is for the network routing service to be smart enough to figure out the best way given one of the hierarchical addresses of the destination or maybe the unique identifier of the destination. The other way is for the source to be aware of the different connection points, to ask for the paths to all these, and to then choose among the paths available. The first calls for a sophisticated network service and the second for a sophisticated source node.

   In fact, this brings up another way in which a source may not be able to communicate with a destination although a path to the destination may exist. This can happen if the destination has several connection points but all the paths (to the connection points of the destination), that the source knows about, are down.

5. **Shared Access**: It is possible that two or more nodes may share a single connection point in the network. This is possible if there is a shortage of connection points. Another reason may be that the nodes involved would rather share the cost of the network interface than have complete access to the network. This may be especially true of small computers. In a strict hierarchy, both of them will have the same network hierarchical address and there will be a lot of confusion when a message gets sent to the shared connection point.

2.6 A more flexible approach to Routing

The problems described above require special attention by a Routing Service. One approach to attack the issues raised is described below.

Firstly, it was emphasized while discussing mobile hosts, shared and changing configuration access, that it is important to preserve the identity of a node if it shares its attachment point with other nodes or if the attachment point for the node either changes or even just changes its name. One way to solve this problem is to assign a unique identifier to every node and to have every internet message contain the unique identifier of the destination node, perhaps as part of the source route of the message. Now, the unique identifier of every message sent to any attachment point can be checked against the unique identifiers of the nodes currently attached to the connection point.

However, this still does not solve the problem of mobile hosts completely. It is not possible under many circumstances to know the complete hierarchical name of the attachment point of a node. The Routing Service should be able to do some kind of a search for the destination node in the network. However, it may be impractical to search the entire network. The Routing Service should, therefore, be able to accept hints (from the node requesting the information) to limit the scope of the search.

In fact, the procedure described above is one way of solving the multi-homing problem, too. If a search is made for some destination node, several attachment points for the same node may show up and the best among them can be chosen.

As for the case of artificial partitioning mentioned before, one can either provide alternative routes to servers or actually have redundant servers for each region.

There is another way in which partitioning may be forced on the network although a path may exist between the two supposedly "partitioned" parts. This can happen if no path exists between two attachment points in the same region but a path may exist

if the path from the source goes outside the region and comes in again to a part of the region from which the destination may be reached. This is solved by using a backup and retry scheme. Help has to be asked of the Server of the region that is hierarchically one level above the region in which the source and destination exist.

This chapter has concentrated on laying down a set of basic requirements for a Routing Service and on developing a general approach to solve some more advanced problems. The next chapter delves into more details about the Routing Service.

This empty page was substituted for a blank page in the original document.

Chapter Three

The Routing Service

This chapter describes the design of a Routing Service in detail. Before describing the details of the design, it is necessary to explain how the campus-wide network is configured. It is also necessary to describe a particular implementation of source routing and to decide what kind of routes should be computed by the Routing Service. These preliminaries are taken care of in the first three sections. After that, Sec. 3.4 describes the topology-finding algorithm of the Routing Service. Sec. 3.5 deals with the algorithms that are used to compute paths in the network and Sec. 3.6 lays out the procedure for asking the Routing Service for routing information. Sec. 3.7 discusses the types of changes in configuration that are useful in a campus-wide network. The next three sections describe strategies for user control of paths, for responding to faults in the network, and for congestion control.

3.1 The Configuration of the Campus-wide Network

As discussed in Chapter 2, a hierarchical approach will be taken to break down the problems of routing for large networks into manageable sub-units. The configuration of the network is described below.

The campus-wide network will be essentially a large number of local networks (or subnetworks) connected together by gateways. The lowest level of the hierarchy will consist of a number of adjacent subnetworks[4]. This lowest level in the hierarchy will be called a level-1 region. One such level-1 region is shown in Fig. 3-1. It is also a

[4] A subnetwork is adjacent to another if the two subnetworks are connected together by at least one gateway. A bunch of subnetworks are adjacent if one can find a path (that passes through gateways and other subnetworks in the same region) between any two subnetworks in the region.

requirement that two different level-1 regions may not overlap and that the entire network be divided into level-1 regions (i.e., every subnetwork will belong to one and only one level-1 region).

In a similar manner, several adjacent level-1 regions may be grouped together to form a level-2 region. If this depth in the hierarchy is deemed necessary, it is enforced over the entire network. In other words, the entire level-1 space is divided into non-overlapping level-2 regions.

In a completely recursive fashion, level-2 regions may be grouped into level-3 regions and so on. Fig. 3-2 shows what a level-i region looks like in terms of level-(i – 1) regions. The number of levels in the hierarchy will depend on the actual size of the network and will increase or decrease with the size of the network.

Before going any further, it is necessary to emphasize that regions are chosen primarily on the basis of the following two factors:

1. Regions are chosen so that traffic originating from the region is mostly directed to attachment points within the region. The reason for doing this, as mentioned earlier, is that routing decisions involving routes that span more than one region are expensive. This will become apparent when the Routing Service is described in detail.

2. Regions may be decided upon to respect administrative and political boundaries. If regions are divided up this way, it is easier to assign responsibilities for maintenance in the network.

Each region has a Routing Server associated with it. A Routing Server is said to be level-i if it looks after a level-i region. A level-i Routing Server should know its place in the hierarchy (i.e that it is level-i) and it should know how to communicate with all the Routing Servers of level-(i – 1) below it in the hierarchy (unless i = 1) and with the Routing Server above it in the hierarchy (if one in fact exists). A Routing Server for a level-i region only computes routes between the level-(i – 1) regions that exist in the level-i region. A level-1 Routing Server only computes routes between attachment points that belong to the level-1 region.

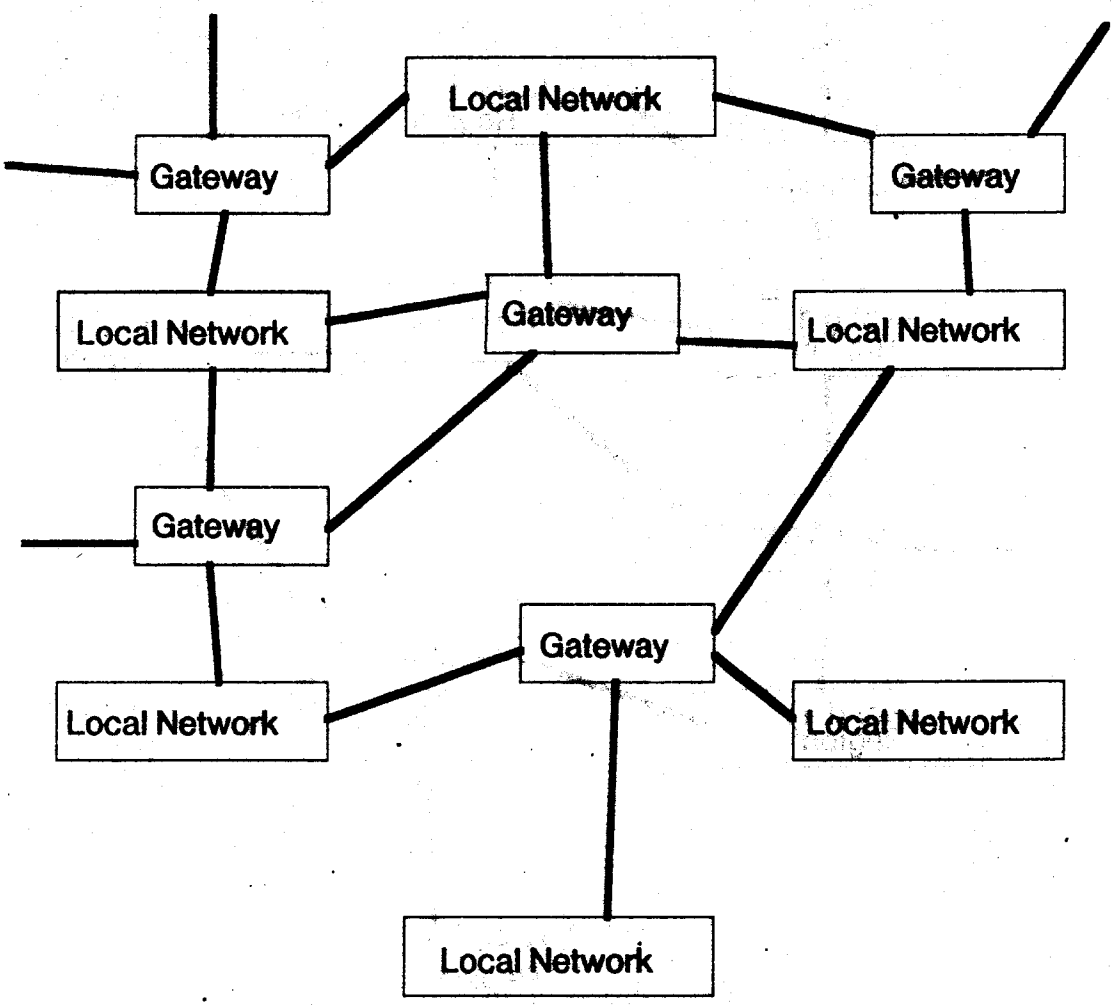

Figure 3-1: A level-1 region

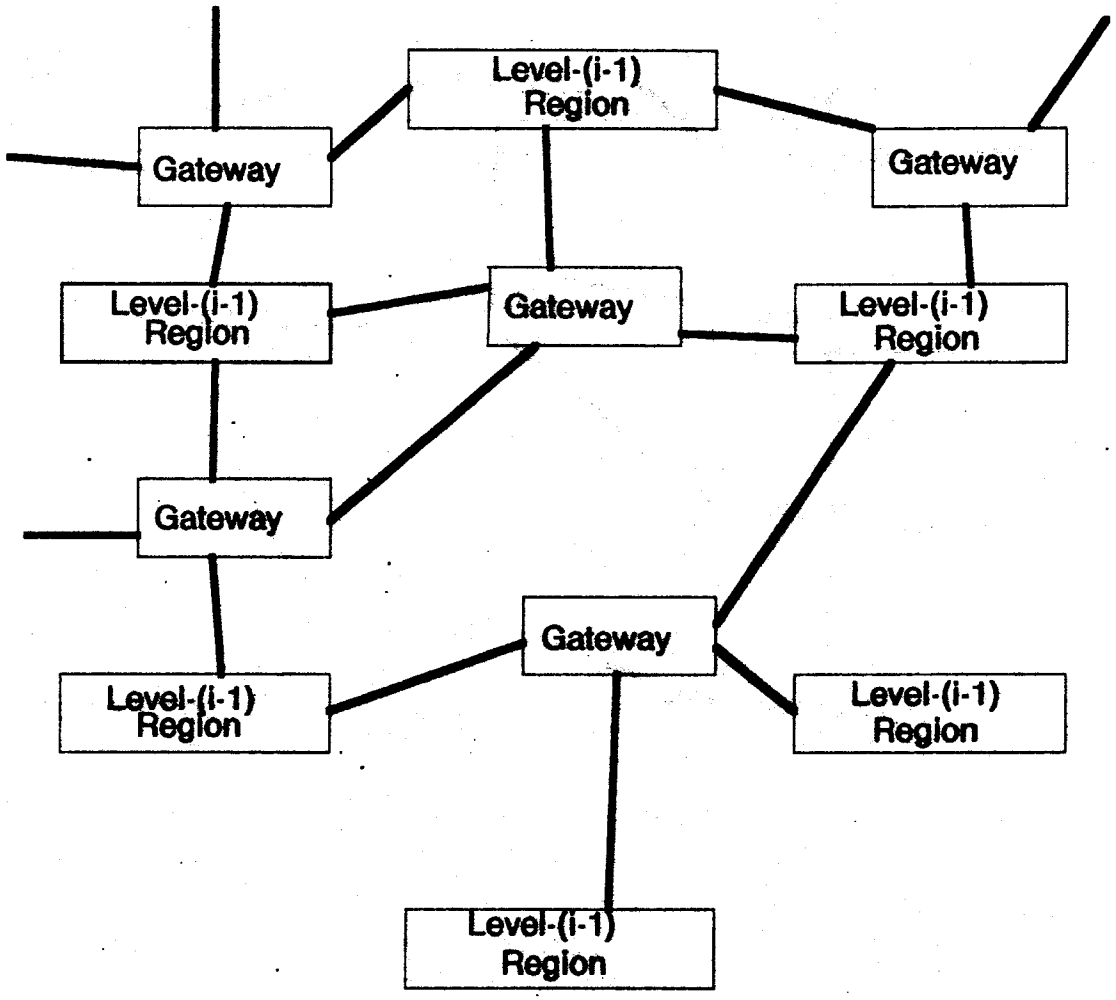

Figure 3-2: A level-i region

Each region, local net, gateway, and node in the network has a unique identifier associated with it. Every node and gateway remembers its own unique identifier. The unique identifier of a subnetwork is remembered by the gateways on the subnetwork. The unique identifier of a region is remembered by the gateways that exist on its boundary.

The reason for assigning unique identifiers to nodes was discussed in Chapter 2 (i.e., to preserve the identity of a node if it shares its attachment point with other nodes or if the attachment point for the node changes).

The reasons for assigning unique identifiers to subnetworks, gateways, and regions are slightly different. There are essentially two reasons.

1. Since class-of-service information will be maintained for subnetworks, gateways, and regions, it is necessary to be able to identify these network entities easily and unambiguously.

2. Due to several reasons (flow control, user control of path, or temporary malfunctioning) it may be necessary to find a path between two attachment points in the network such that the path does not pass through certain subnetworks, gateways, or regions. For this reason, too, it is necessary to be able to point to each of these network entities easily and unambiguously.

Attachment points in the network do not need to have unique identifiers assigned to them. Attachment points may have identifiers that are unique only within the subnetwork in which they exist[5]. In this thesis, attachment point names may be sometimes referred to equivalently as local transport addresses (or simply addresses) of the nodes that are connected to the attachment points.

---

[5]There are a couple of exceptions. Nodes and gateways will each have a common broadcast address as will be seen later.

3.2 An Implementation of Source Routing

To permit explicit discussion of the Routing Service, it is necessary to look at an actual implementation of Source Routing. The implementation described here is essentially the same as the one described in [18]. One of the features of this implementation is that it dynamically constructs a reverse route. This feature has been retained in the implementation because it is extensively used by the Routing Service. The implementation is described below.

The internet source route field is shown in Fig. 3-3. It consists of two one-octet numerical fields and a variable (but constant for the life time of the packet) number of octets of route. The first field contains a count of the number of octets of route while the second field points to the next unused octet of the route. The first field remains constant for the life time of the packet but the second one is updated at each gateway and also by the source node and the destination node.

Assume for now that every gateway connects exactly two subnetworks. The operation of a gateway that gets a packet using the local transport protocol of the incoming subnetwork and wants to send it out on some outgoing subnetwork is described next. The gateway uses the second numerical field (which points to the next unused octet of route) to find the next local transport address[6]. The assumption here is that the gateway knows the number of octets required by a local transport address of this subnetwork. This local transport address is placed in the local transport address field for the outgoing subnetwork. Also, this local transport address is replaced by the gateway's own local transport address (after reversing the address octet by octet). The gateway then increments the second numerical field by the number of octets it extracted from the route, and it uses the local transport protocol to send it out on the outgoing subnetwork. This routing strategy assumes that all paths are bi-directional and that all local transport addresses on a subnetwork

[6]Note that the local transport address of a node on some subnet is not the same as the name of the node. Node names are unique identifiers but local transport names are unique only within local nets.

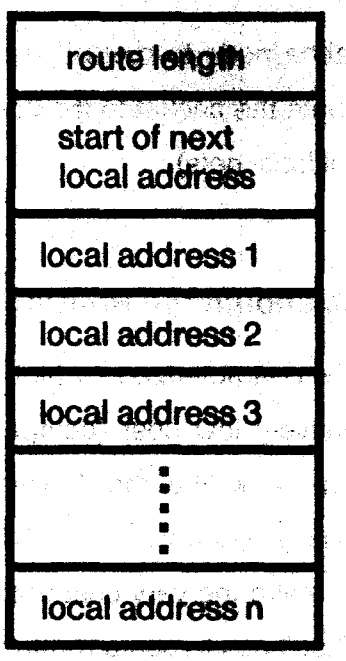

Figure 3-3: The internet source route field

are of the same size. Also, the reverse route comes out upside down and will have to be reversed before it can be used.

In case a gateway interconnects more than two subnetworks, it behaves as if it were another subnetwork. The next local transport address[7] in the source route is used to choose the outgoing subnetwork. Finally, to make it consistent and simple, even gateways that interconnect just two subnetworks will be made to go through this

[7] Just as local transport addresses for nodes are not unique identifiers, this local transport address should not be confused with the unique identifier that is associated with the subnet. Each gateway uses some names to identify the different subnets to which it is connected. These names, which are treated as local transport addresses on the conceptual subnetwork of the gateway, are unique only within the conceptual network.

step.

The operation described above is repeated not only by each gateway but may be also repeated inside the destination node to route the packet to the correct activity and inside the source node to route the packet to the correct local network (since a node may be connected to several local nets).

3.3 What kind of routes should the Routing Service compute?

The campus-wide network is characterized by high-bandwidth communication lines that are available at a relatively low cost (compared to the communication links available for a long-haul network like the ARPA network). It is, therefore, not worthwhile to look for extremely sophisticated routing strategies that may use a lot of computational resources to make the very best use of bandwidth. Since there is an abundance of bandwidth, it suffices to use a routing strategy that computes shortest hop paths. By choosing a shortest hop routing strategy, the cost of making the complicated routing calculations required for long-haul networks like the ARPANET will not be incurred.

By pursuing the line of reasoning given above a little further, one might argue that routing calculations should be cut down even further by calculating any routes whatever (i.e., routes that are not necessarily shortest in length). However, there are at least two good reasons for finding shortest routes as contrasted with finding any routes. The first argument in favor of using shortest routes is the reliability argument. The smaller the number of subnets and gateways that a message has to pass through, the better are the chances of it getting through intact to the other end (on the average, at least). The second reason for using shortest paths and not any paths whatever is as follows. If any paths whatever are used, then there will be more traffic generated in the network due to each message (on the average) because the message may have to go over a higher number of subnets and gateways as compared to the case when shortest paths are used. Consequently, the number of

different messages per unit time that the network can sustain will be decreased, which is a disadvantage.

If shortest hop paths are the paths that the Routing Service will compute, then the only thing that needs to be monitored is the topology of the network and its connectivity, both of which are subject to change. Traffic conditions on communication links need not be monitored, as in the ARPANET, to make sophisticated routing decisions. The Routing Service that is described later in this chapter has been designed keeping this in mind.

Given the structure of the campus-wide network in which various subnets are connected by means of gateways, it is logical to define a path-length to be as many hops as the number of gateways and subnets that it spans. Therefore, nodes on the same network will be one hop away, nodes on different subnets connected by a common gateway are three hops away and so on. There is another way to define hops that is simpler although it may be less intuitive. The number of hops in a path can be defined to be the number of gateways on the path. A little bit of thought will show that shortest hop paths, which are computed by using either of the two definitions, mean the same thing. (The reason for this is that the number of hops in a path, using the first definition, is a monotonically increasing function of the number of hops in the same path, using the second definition.) The second definition of path length will be used in this thesis.

3.4 Finding the topology of the Campus-wide Network

Finding the topology of the network is the first task of the Routing Service. As explained before, a hierarchical approach has been taken to split up the routing problems into manageable units. For convenience of discussion, suppose that the network is itself a level-n region, where n is some number greater than zero.

This section is organized as follows. First, some special features that are required for

the topology finding functions of the Routing Server have been explained. Next, a very crude description of the topology finding operation of the Routing Server has been given. This has been done to give the reader a higher level insight into the algorithm before introducing a lot of details.

There are two ways to read this section. If the reader only wants to get a feel for the algorithm, it is recommended that he should skim over the description of the special features and then he should only read the crude description of the algorithm. On the other hand, for the reader interested in more details, the following is recommended. The reader should first skim over the description of the special features and the crude description of the algorithm. Next, the reader should read the exact algorithm—referring to the description of the special features, if necessary.

This thesis will not attempt to describe the details of coding the various messages that are required for the operation of the Routing Service. Also, no attempt will be made to describe in detail how the Routing Servers, the gateways, or the nodes organize the storage of state associated with the operation of the Routing Service except where absolutely necessary.

3.4.1 Eleven special features required for topology-finding

1. *Gateways respond to a common local transport address*: Every gateway has a local transport address on each subnet that it is connected to. The local transport address of the gateway on every subnet is unique over the subnet. However, it is required that all gateways on a subnet also respond to some other common local transport address. The common address for gateways on a subnet may be different from the common address for gateways on another subnet. The common address for gateways will be used to broadcast messages to all gateways on a subnet and it is for this reason that the the common local transport address may be referred to equivalently as the broadcast address for gateways.

2. *Nodes respond to a common local transport address*: Just as gateways respond to a common address other than their regular local transport

address, all nodes on a subnet respond to a common local transport address other than their regular address. It is possible to make the common address for nodes identical to the common address for gateways on the same subnet. Software can then be used to differentiate between broadcasts meant for nodes from those meant for gateways. However, it will be assumed, for ease of discussion, that the common address for nodes is different from the common address for gateways on the same subnet.

3. *Remote Broadcast* message: Before describing this message, it is necessary to explain the standard representation of a message in this thesis. This standard representation is given below.

   To-<destination>(<Type of message>, <parameter 1>, <parameter 2>, <parameter 3>,...,<parameter n>)

   Here <destination> is the destination for this kind of packet. Examples of <destination> are gateways, nodes, etc. In an actual message, of course, this corresponds to the source route that is used to send the message to the appropriate destinations (which are of the type given in the <destination> field of the description).

   <Type of message>, <parameter 1>, etc. are all fields that correspond, in an actual message, to pieces of information that are coded into the internet transport packet. The details of coding this information are not of importance to this thesis.

   If this type of message has a standard reply message associated with it, it will look like this:

   From-<destination>(<Type of message>, <parameter a>, <parameter b>, ...)

   From-<destination> corresponds, in an actual message, to a source route that leads from the destination to the originator of the To-<destination> message. (The dynamic reverse route construction strategy, which was described earlier, is used to get the reverse route required above.)

   <Type of message>, <parameter a>, etc. now correspond to pieces of information that are coded into the internet transport packet that is used for the reply.

   Now, to get back to the original discussion, nothing has been said so far

about the mechanism that is used to send a packet over a local network using the local transport protocol. Indeed, each local network will have its own local transport protocol that may be different from any other local transport protocol. However, one assumption must be made about the local transport protocol. The assumption is that if there are several attachment points on a subnet with the same local transport address and if a packet having the common local transport address is delivered to the subnet, then the packet will eventually be delivered to all the attachment points on the subnet with the common local transport address.

The *remote broadcast* message looks like this:

To-gateway(Remote broadcast, destination, local transport address of subnet, packet)

The "destination" field of the message can contain either "nodes" or "gateways". This field essentially tells the recipient of the message whether the packet included in the "packet" field should be broadcast to all the gateways or all the nodes on the subnet identified in the "local transport address"[8] field. Suppose, for example, that the "destination" field of the message says "nodes". The gateway will then place the packet contained in the message on the appropriate subnet using the correct local transport protocol (after putting the common address for nodes on the subnet in the local transport address field of the packet). The gateway will also be responsible for arranging the internet source route field of the packet that it will send so that it looks as if the originator of the packet is the same as the originator of the *remote broadcast* message sent to the gateway.

4. *Gateway Descriptor*: Each gateway has a gateway descriptor associated with it. The descriptor is essentially supposed to store information about where the gateway belongs in the hierarchical configuration of the network, the routes to the various Routing servers of interest, and the class of service information about the gateway and the various regions and local nets around it. To be more precise, a gateway descriptor consists of the following pieces of information:

   a. The unique identifier of the gateway.

   b. The local transport addresses, on the conceptual subnet of the

[8] This local transport address is the address of the subnet on the conceptual subnet of the gateway.

gateway, of the subnets that the gateway is connected to.

c. The local transport address on each subnet.

d. The class-of-service information on the gateway.

For each pair of local networks connected to the gateway:

e. The highest level of the gateway between the local networks. (If both the local networks belong to the same level-1 region then the highest level of the gateway between them is taken, by convention, to be 0. It should be noticed that if this highest level is i, then the gateway is also a level-j gateway between the two local networks for $0 < j < i$.)

For each local network connected to the gateway

For each level region on that side

f. The unique identifier of the region.

g. The class-of-service information on the region.

h. The route to the Routing Server of the region[9].

It is not necessary for the gateway to have all this information at all times. In fact, when the gateway first begins operation, it is only necessary for it to know b, c, and e. The rest of the information gets filled in as part of the operation of the Routing Service as will be seen later. Also, b, c, and e have to be kept in some kind of stable storage because it is required that this information be retained by the gateway even if it crashes and comes up again.

5. *Node descriptor*: Each node has a node descriptor associated with it. The node descriptor consists of the following pieces of information:

a. The unique identifier of the node.

b. The class-of-service information on the node.

---

[9]For the level-0 region (i.e., the local network), this is not applicable. Another exception to this rule is that the a level-0 gateway also contains information on the unique identifier of the level-1 region and the route to the level-1 Routing Server.

c. The local network addresses of the attachment points to which it is connected.

d. The unique identifiers of the local networks to which it is connected.

Clearly, c and d change whenever a node is moved from one local network to another. Information on c is acquired by the node when it is connected to a new place in the network. The mechanism for learning d dynamically will be described later.

6. *What-is-your-descriptor?* message: There are two variations on the *What-is-your-descriptor?* message. The two variations are shown below:

   a. To-gateway (What-is-your-descriptor?)

      From-gateway (What-is-your-descriptor?, gateway descriptor)

      The gateway descriptor here is the same as the standard one described before.

   b. To-node (What-is-your-descriptor?)

      From-node (What-is-your-descriptor?, node descriptor)

      This node descriptor is the same as the one described earlier.

7. *Fill-in-your-descriptor* message: This message appears as follows:

   To-gateway (Fill-in-your-descriptor, <information pair 1>, <information pair 2>,...)

   The information pairs mentioned above consist of two parts themselves— the first is the kind of information that must be filled in and the second is the information itself.

   A variation of this message also exists:

   To-node <Fill-in-your-descriptor, <information pair a>, <information pair b>, ...)

   Sometimes, an originator of this message might want the gateway to pick up the reverse route as one of the pieces of information to be filled in.

For example, a Routing Server level-i might want to fill in some gateway descriptor that needs the route to the Routing Server. The Routing Server might only know the route from itself to the gateway. In that case, the information pair field corresponding to this route (from the gateway to the Routing Server) may refer to the reverse route constructed by the dynamic reverse route construction strategy.

8. *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message.

This message looks as follows:

To-Routing Server(Give-me-descriptor-of-gateways-at-the-edge-of-your-region, hierarchical address of attachment point where I want routes from, number of last update)

The reply to this message looks as follows:

From-Routing Server (Give-me-descriptor-of-gateways-at-the-edge-of-your-region, <information changed?>, <information on gateways>)

The information that the Routing Server is asked to furnish concerns the gateways that lie on the boundary of the region that the Routing Server administers. This information involves not only the descriptors of the gateways but also routes to the gateways from the point in the network specified in the request. Each update by the Routing Server or information about the gateways is numbered. Therefore, if the request contains the latest update number, the reply puts *no* in the "information changed?" field and makes the "information on gateways" field blank. If, however, the update number is different now, then the "information changed?" field has *yes* in it and the "information on gateways" field is filled with the appropriate information. The "information on gateways" field, in fact, consists of the following sub-fields:

a. List of descriptors: This is a list of descriptors of the gateways on the edge of the region.

b. List of routes to the gateways: This is a list of the routes to the gateways mentioned in the "list of descriptors" field in the same order. The routes are given from the attachment point whose hierarchical address was specified in the request. The way to specify a hierarchical address is described later in this thesis.

c. Number of this update: This field specifies the update number of

the information on the gateways sent back in this request.

9. *Here-is-the-route-to-Routing Server-above-you* message: This message looks as follows:

   To-Routing Server (Here-is-the-route-to-the-Routing Server-above-you, route, unique identifier of Routing Server, list of unique identifiers of Routing Servers above the one mentioned, list of Routes to Routing Servers above the one mentioned from the originator of this message)

   This message will be used by some Routing Server level-i (where $i > 1$) and will be sent to one of the Routing Servers level-$(i-1)$ under the level-i Routing Server. The route specified in the message may refer to the reverse route that is normally constructed. Also, if the level-i Routing Server has information (regarding unique identifiers and routes) on Routing Servers above it in the hierarchy, then that information is sent, too.

10. *I-would-like-to-get-information-on-Routing Server level-i* message: This message looks as follows:

    To-Routing Server (I-would-like-to-get-information-on-Routing Server level-i)

    From-Routing Server(I-would-like-to-get-information-on-Routing Server level-i, highest level $j \leq i$ of Routing Servers for which I have information, list of unique identifiers of Routing Services above me until level-j, list of routes to Routing Servers above me until level-j)

    All fields in this message type are self-explanatory. The routes to Routing Servers are from the Routing Server to which the message was directed.

11. *What-is-the-route-from-me-to-you?* message: This message looks as follows:

    To-Routing Server (What-is-the-route-from-me-to-you?, my hierarchical address)

    From-Routing Server (What-is-the-route-from-me-to-you?, route)

    Clearly, the originator of the message must have a route to the Routing Server to be able to send the message in the first place. However, the

route may not be a good one and it may go past several other attachment points unnecessarily. The message described above may then be used to find a relatively shorter route.

Assume for now that the Routing Server to which this query is directed is capable of coming up with a route of the type specified. The exact algorithm for finding this route or any other route, for that matter, will be described later.

It should be noted that most of the messages described in this section had some kind of reply associated with them. The replies to those messages also act as acknowledgments for the messages themselves and, therefore, there is no need to send explicit acknowledgments. There are two messages, however, which do not have any replies associated with them—namely, the *Fill-in-your-descriptor* message and the *Here-is-the-route-to-Routing Server-above-you* message. It may be necessary to send explicit acknowledgments for these messages although none have been mentioned in the description of these messages.

3.4.2 A crude description of the algorithm

One assumption of this algorithm is that a level-i Routing Server exists within the level-i region that it serves or, to be more precise, the node on which the Routing Server runs is connected to an attachment point within the level-i region that it serves. This is not an unreasonable assumption considering that the main motivation for dividing up the network into regions is to provide better service to each region. It is, therefore, natural to place the Routing Server as close as possible to all the nodes or Routing Servers below it in the hierarchy.

Another thing to notice is that level-1 Routing Servers are a little different from level-i Routing Servers, where $i > 1$. The reason is that level-1 Routing Servers do not gather their topology information from lower level Routing Servers but instead directly from the nodes and gateways in the level-1 region. The operation of a level-1 Routing Server will, therefore, be described first and then the operation of other Routing Servers will be described.

Also, note that the operations that are described are actually repeated over and over again. The frequency of the repetitions should be high enough so that changes get noticed fairly quickly but the frequency should not be so high that the Routing Service gets slowed down unduly by just this part of its responsibility.

3.4.2.1 The operation of a level-1 Routing Server

Assume that when a level-1 Routing Server first begins operation, it knows absolutely nothing about the topology of the network. It will only be required that the Routing Server should know that it is level-1 and also to know the unique identifier of the region that it will serve[10]. Now, it is known that the node on which the Routing Server runs is attached to a point in the same level-1 region[11]. The idea is that the Routing Server should find out first about the nodes and gateways on the subnet on which it exists and then expand its knowledge to nodes and gateways in neighboring subnets until the boundary of the level-1 region is finally reached.

Therefore, to begin with, the Routing Server sends out a *What-is-your-descriptor?* packet on its subnetwork directed to all the nodes there. This is done by putting the common address for nodes in the local transport address field of the *What-is-your-descriptor?* packet. Next, the Routing Server sends out a *What-is-your-descriptor?* packet to all the gateways on its subnet. The replies that the Routing Server gets back from nodes and gateways give it information on all the nodes and gateways on the subnet (except those that are down temporarily[12]).

The gateway descriptor that the Routing Server receives will inform it of other subnets in the region that it has not explored yet. Using remote broadcasts, the

---

[10] In fact, the Routing Server need not even know this unique identifier. It can get a unique identifier from some network-wide unique identifier dispensing service or some such means before proceeding.

[11] Assumption made earlier.

[12] There is no need to despair, however. Information on these will get collected in later iterations of this algorithm.

Routing Server will put *What-is-your-descriptor?* messages for both gateways and nodes on all subnets that it has not explored yet. Again, information from nodes and gateways is collected and organized appropriately. As replies from gateways come in, the Routing Server gets to know about other subnets in the region that it was not aware of before. All such subnets are then explored until no more subnets are left unexplored.

It should be noted that there may be several ways in which a subnet may be revealed to the Routing Server (i.e., through different chains of gateways). However, a single subnet needs to be explored just once through remote broadcasting *What-is-your-descriptor?* messages (for nodes and gateways) on it. Discovering that each subnet is explored just once is a responsibility of the Routing Server and it is possible because each local network has a unique identifier that is part of the descriptors of gateways that are connected to the subnet.

Another important observation concerns gateways that lie on the boundary of the region and that are connected to several subnets. Clearly, one of the subnets to which such a gateway is connected lies within the level-1 region (because the Routing Server became aware of the gateway through a broadcast on some subnet). Also, for the gateway to be on the boundary, one of the other subnets must be in another level-1 region (i.e., the gateway must be level-1 between these two subnets). The important thing is that the other subnets may be either within the level-1 region of the Routing Server in question or they may be out of it. If they are within the region, they must be explored (unless they have already been explored). The point is that a gateway may be on the edge of a region and yet some of the subnets that it leads to may be within the region and these should be explored as usual.

Also, note that it has not yet been described how the information collected from nodes and gateways is organized. The purpose of describing the topology-finding algorithm is only to convince the reader that the topology can be found (and efficiently).

The next thing that will be described is the operation of a level-i Routing Server, where i > 1. As explained earlier, the differences in the algorithms are due to the fact that this Routing Server will get the topology information from Routing Servers below it and not directly from nodes and gateways below it in the hierarchical configuration.

3.4.2.2 The operation of a level-i Routing Server

Just like a Routing Server level-1, a Routing Server level-i has no knowledge of the topology when it first begins operation. All it knows is that it is a level-i Routing Server and that it knows the unique identifier of the region. The idea here is to contact all the Routing Servers of level-(i – 1) below the level-1 Routing Server in the hierarchy and to gather information from them about the gateways of level-(i – 1) at the edge of their regions.

To do this, the Routing Server of level-i must first contact the Routing Server of level-(i – 1) in whose region it lies. This is done by first getting to the Routing Server level-1 (in whose region it lies) and working up to the Routing Server level-(i – 1). This, in turn, is done by first sending a *What-is-your-descriptor?* packet to all the gateways on the subnet on which the Routing Server level-i is located. Any one of the gateway descriptors received in reply will give the level-i Routing Server the route to Routing Server level-1 and the unique identifier of the level-1 regions. Next, the Routing Server level-i sends an *I-would-like-to-get-information-on-Routing-Server level-(i – 1)* message to the level-1 Routing Server[13]. The route to the Routing Server level-2 is then computed by concatenating the route to the level-1 Routing Server with the route from the level-1 Routing Server to the Routing Server level-2. An *I-would-like-to-get-information-on-Routing Server level-(i – 1)* is then sent to the level-2 Routing Server and so on until finally the route to the level-(i – 1) Routing Server is found. It should be noted that short-cuts can be taken if some Routing Server in the chain up

[13] As will be seen later, a level-s Routing Server sends to each level-(s – 1) Routing Server below it the route from the level-(s – 1) Routing Server to itself as part of its topology-finding algorithm. Therefore, the level-1 Routing Server is expected to have the route from it to the level-2 Routing Server above it.

from the level-1 Routing Server to the level-(i – 3) Routing Server knows of more than one Routing Server above it. (This was explained in the *I-would-like-to-get-information-on Routing Server level-i* message.)

This route that the Routing Server level-i now knows to reach Routing Server level-(i – 1) may not be the best route because it will go by several other Routing Servers along the sequence from Routing Server level-1 to Routing Server level-(i – 2). To get a good route, the Routing Server level-i sends a *What-is-the-route-from-me-to-you?* message to the Routing Server level-(i – 1). Using this route the Routing Server level-i then sends a *Here-is-the-route-to-Routing Server-above-you* message to Routing Server level-(i – 1) and it also sends a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the same Routing Server level-(i – 1).

Now, the Routing Server level-i can, by inspection of the gateway descriptors received from Routing Server level-(i – 1), find out the routes to the Routing Servers of the level-(i – 1) regions lying beyond the level-(i – 1) region to which the level-i Routing Server belongs but lying within the level-i region being considered. This is done as follows. Each gateway on the boundary of the level-(i – 1) region to which the level-i Routing Server belongs will have routes to the Routing Servers level-(i – 1) lying beyond it. To get the route to any of those level-(i – 1) Routing Servers, the Routing Server level-i has to merely concatenate the route from itself to the gateway with the route from the gateway to the other level-(i – 1) Routing Server.

Using these routes to other level-(i – 1) Routing Servers (within the level-i region), the level-i Routing Server can send *Here-is-the-route-to-Routing Server-above-you* and *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the level-(i – 1) Routing Servers. This can then be repeated for each level-(i – 1) region that lies in the level-i region until all of the level-(i – 1) regions in the level-i regions have been explored.

Again, the details of organizing the topology data will not be described now. For

present purposes, the claim is made that the level-i Routing Server will, at the end of the topology finding operation described above, know about the connections of the level-(i – 1) regions in the level-i regions and it will know the descriptors of the level-(i – 1) gateways that connect the level-(i – 1) regions.

The way the algorithm has been described so far, a level-i Routing Server will get information from a level-(i – 1) Routing Server below it only when the level-i Routing Server sends a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message. However, it might be useful to provide a feature whereby a level-(i – 1) Routing Server could bring to the attention of the level-i Routing Server above it any changes that might have occurred in the level-(i – 1) region since the last update that was sent to the level-i Routing Server[14]. In fact, one can go even further and make a level-(i – 1) Routing Server always report to the level-i Routing Server immediately after an iteration of its topology-finding operation. These changes can be reported to the level-i Routing Server in the form of a reply to a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message (even though, in fact, no such message may have been sent).

This is the end of the crude description of the topology-finding algorithm. The rest of this section gives a detailed description of the algorithm.

3.4.3 The full algorithm for finding the topology of the network

First, the algorithm that a level-1 Routing Server uses for topology-finding and an informal explanation about how it works is given. This is followed by a similar description for the algorithm that a level-i Routing Server uses for topology-finding and by an informal argument for its correctness.

---

[14]Note that we are only referring to changes in the level-(i – 1) region that the level-i Routing Server is interested in.

3.4.3.1 The level-1 topology-finding algorithm

The algorithm that will be described uses four different lists and a queue. All the lists and the queue start out empty for each iteration of the topology-finding algorithm. The lists are:

1. *List of node descriptors*

2. *List of gateway descriptors*

3. *List of gateways on the boundary of the region*: This list only contains the unique identifiers of the gateways on the boundary.

4. *List of local net descriptors*: A local net descriptor will consist of the following two items:

    a. The unique identifier of the local net in question.

    b. The unique identifiers of the gateways on the local net.

The queue that the algorithm uses will be called the *exploration queue*. Each item on the queue will consist of the unique identifier of some local net and also the route to the local net from the Routing Server.

The topology-finding algorithm is given below.

1. Put the local net to which the Routing Server is connected on the exploration queue.

   For each local net on the exploration queue do:

2. Send a *What-is-your-descriptor?* message to all the gateways on the local net. Use the remote broadcast feature with "gateways" in the "destination"' field and the contents of a regular *What-is-your-descriptor?* message in the "packet" field of the *remote broadcast* message. Send the remote broadcast message to the last gateway on the path to the local net with an appropriate "local network address of subnet" field.

3. Send a *What-is-your-descriptor?* message to all the nodes on the local net by using the remote broadcast feature.

4. For each reply from 2 do:

a. Check to see if the gateway descriptor contains the following information:

- the unique identifier of the gateway

- the unique identifiers of the local networks to which the gateway is connected

- the route to the Routing Server

If the unique identifier of the local net (currently being explored), as contained in the descriptor, is different from the unique identifier of the local net in the descriptor of the last gateway on the path of the remote broadcast sent in step 2, then do the following. Check the exploration queue to see if it contains an entry corresponding to a local net with the unique identifier that the present gateway descriptor contains. If there is such an entry, remove it from the exploration queue.

If any of the above pieces of information that are checked for in the gateway descriptor are incorrect or missing, send a *Fill-in-your-descriptor* message to the gateway to change the descriptor suitably.

b. Add this new updated gateway descriptor to the list of gateway descriptors for the region. If a descriptor for this gateway already exists in the list, then it is replaced by this new descriptor.

c. If the gateway happens to be level-1 between the present local net and some other local net to which it is connected, then add it to the list of gateways on the boundary of this level-1 region.

d. Add the local nets to which the gateway leads and that lie within the same level-1 region to the end of the exploration queue (unless the local nets in question are already in the queue or unless there is an entry for the local net in the list of local net descriptors).

end of do

5. For each reply from 3 do:

a. Check the node descriptor to see if it has the following pieces of

information:

- the unique identifier of the node

- the unique identifier of the local net being explored

If either piece of information is missing[15], then send a *Fill-in-your-descriptor* message to the node to complete it.[16]

b. Add this new updated node descriptor to the list of node descriptors for the region. If it is already there, just replace the old descriptor with the new one.

end of do

6. Complete the local net descriptor for the local net just explored and add it to the list of local net descriptors for the region.

end of do

7. Send a reply to a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the Routing Server above this one (although no such message may have been sent by the higher level Routing Server).

3.4.3.2 How does the level-1 topology-finding algorithm work?

No formal proof for the correctness of the algorithm will be given. However, some less-than-obvious properties of the algorithm will be clarified.

The exploration queue is used to store leads to local nets that have not been explored yet. The most important property of the exploration queue is that it promotes a breadth-first search of the level-1 region as opposed to a depth-first search (if depth is defined in terms of number of hops). This ensures that the local

---

[15] It might seem a little exotic to get a unique identifier for a node but this facility is provided to take care of the rare circumstance when a node does not have its own unique identifier due to some reason. Normally, every node is expected to come with its own unique identifier.

[16] The unique identifier for the local net that is given to the node is the same as the unique identifier that the gateway descriptors were given for this local net.

net zero hops away from the Routing Server (i.e., the local net on which the Route Server resides) is searched first. Next, all local nets one hop away are searched and then all local nets two hops away, and so on[17]. Therefore, the path that is used first to get to a local net is the minimum-hop path from the Route Server. This is the reason why the route from gateways to the Route Server is put in the gateway descriptors as the local net is being explored. It is not necessary to wait for the completion of the algorithm to find the shortest hop path from each gateway to the Route Server.

An attempt has been made in the algorithm to explore all local nets only once. This is the reason why a local net is not put in the exploration queue if it is already somewhere in the queue or if it is in the list of local net descriptors. However, this property may not always be true and sometimes a local net may be explored more than once if care is not exercised. Consider the case of a local net that does not have a unique identifier yet. Also, assume that the local net is connected by gateways to two of the local nets that are about to be explored. Since the gateways that lead to the local net in question will not have a unique identifier for it, the Routing Server will get unique identifiers for the local net for each of the gateway descriptors not knowing that it is for the same local net. Since the unique identifiers are different, the local net will be placed on the exploration queue twice. The algorithm is, however, smart enough to recover from this situation. When the local net is first explored through one of the gateways that leads to it, the Routing Server will notice that some other gateway has an inconsistent unique identifier for the same local net. The exploration queue is scanned to check if there is an entry with the inconsistent unique identifier. If there is, then it is removed from the exploration queue. Also, the gateway descriptor for the gateway with the inconsistent unique identifier is corrected accordingly, as usual.

There is another question that is worth dwelling on for a little while. Is the topology

---

[17] In this context, "all local nets" refers to all local nets that lie in the level-1 region being considered.

information really complete? The claim is that the topology data is complete. The reason—to put it very intuitively—is that every lead to a local net is pursued (if it lies within the same level-1 region) and, therefore, each local net in the region is explored. Also, since all node descriptors and gateway descriptors on each local net are collected, complete topology information for the level-1 region exists.

3.4.3.3 The level-i topology-finding algorithm

The topology-finding algorithm of a level-i Routing Server (where i > 1) uses three different lists and a queue. All the lists in the queue start out empty for each iteration of the topology-finding algorithm. The lists are:

1. *List of gateway descriptors*: The gateways in the list are all the gateways at the edges of the various level-(i – 1) regions in the level-i region.

2. *List of gateways on the boundary of the level-i region.*

3. *List of level-(i – 1) region descriptors*: A level-(i – 1) region descriptor consists of the following three items:

   - The unique identifier of the level-(i – 1) region.

   - The unique identifiers of the gateways on the boundary of the level-(i – 1) region.

   - The route to the Routing Server of the region.

The queue that the algorithm uses will be called the *exploration queue*. Each item on the queue will consist of the unique identifier of some level-(i – 1) region and the route to the Routing Server of the level-(i – 1) region.

The topology-finding algorithm is given below.

1. If the route to the level-(i – 1) Routing Server is known, then go to step 8, else go to step 2.

2. Send a *What-is-your-descriptor?* message to all the gateways on the local net to which the Routing Server is connected by putting the the

common local transport address for gateways in the local transport address field of the packet.

3. Send an *I-would-like-to-get-information-on-Routing Server level-(i – 1)* message to the level-1 Routing Server. (The route to the level-1 Routing Server is picked out from any of the gateway descriptors received as replies to 2.)

4. Check to see if the reply to step 3 contains information on the route to the level-(i – 1) Routing Server[18]. If it does then the route to the level-(i – 1) Routing Server from the level-i Routing Server is constructed by concatenating the route from the level-i Routing Server to the level-1 Routing Server with the route from the level-1 Routing Server to the level-(i – 1) Routing Server. Now, go to step 8.

   On the other hand, if the highest level that the level-1 Routing Server knows about = j < (i – 1), then construct the route to the level-j Routing Server from the level-i Routing Server.

5. Send an *I-would-like-to-get-information-on-Routing Server level-(i – 1)* message to the level-j Routing Server.

6. Check to see if the reply to 3 contains information on the route to the level-(i – 1) Routing Server. If it does then the route to the level-(i – 1) Routing Server from the level-i Routing Server is constructed by concatenating the route from the level-i Routing Server to the level-j Routing Server with the route from the level-j Routing Server to the level-(i – 1) Routing Server. Now, go to step 8.

   Otherwise, construct the route to the highest level Routing Server (known to the level-j Routing Server) and make j = the level of this Routing Server. Now, go to step 7.

7. Send a *What-is-the-route-from-me-to-you?* message to the level-(i – 1) Routing Server. (Note that the hierarchical address to the sender has to be included in a *What-is-the-route-from-me-to-you?* message. The level-i Routing Server knows the unique identifiers of the regions in which it

[18] A level-s Routing Server will always know the route from it to the level-(s + 1) Routing Server after the level-(s + 1) Routing Server goes through the topology-finding algorithm. See step 11 to find out the reason.

resides from level-0 to level-(i – 1)[19]. Only the information corresponding to these fields is put in the hierarchical address (of the level-i Routing Server) that is placed in the *What-is-the-route-from-me-to-you?* message.

8. Put the level-(i – 1) region in which the level-i region lies on the exploration queue.

   For each level-(i – 1) region on the exploration queue do:

9. Send a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the Routing Server of the level-(i – 1) region. If this does not work for some reason, then go to step 1.

10. For each gateway descriptor in the reply to 9 do:

    a. Check to see if the gateway descriptor contains information on the unique identifier of the level-i region and the route to the level-i Routing Server.

       If the unique identifier of the level-(i – 1) region (currently being explored), as contained in the descriptor, is different from the unique identifier of the region, as contained in the entry for the level-(i – 1) region in the exploration queue, then do the following. Check the exploration queue to see if it contains an entry corresponding to a level-(i – 1) region with the unique identifier that the present gateway descriptor contains. If there is such an entry, remove it from the exploration queue.

       If any of the pieces of information checked for above is missing or incorrect, send a *Fill-in-your-descriptor* message to the gateway to update that information.

    b. Add this new updated gateway descriptor to the list of gateway descriptors for the region. If a descriptor for the gateway already exists in the list, then it is replaced by this new descriptor.

    c. If the gateway happens to be level-i between the present level-(i – 1) region and some other level-(i – 1) region to which it is

---

[19] It gets this information from the reply to 2 and the replies to the *I-would-like-to-get-information-on-Routing Server level-(i – 1)* messages that it sends out.

connected, then add it to the list of gateways on the boundary of this level-i region.

d. Add the level-(i – 1) regions, which the gateway leads to and which lie within the level-i region, to the end of the exploration queue (unless the level-(i – 1) regions in question are already in the queue or unless there is an entry for the level-(i – 1) region in the list of level-(i – 1) region descriptors).

end of do

11. Send a *Here-is-the-route-to-Routing Server-above-you* message to the level-(i – 1) Routing Server with the relevant information.

12. Complete the region descriptor for the level-(i – 1) region just explored and add it to the list of level-(i – 1) region descriptors for the level-i region.

end of do

13. Send a reply to a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the Routing Server above this level-i Routing Server (although no such request may have come from the higher level Routing Server).

3.4.3.4 How does the level-i topology-finding algorithm work?

There are only a few differences between the topology-finding algorithm for a level-i (i > 1) Routing Server and the topology-finding algorithm of a level-1 Routing Server. Essentially, a level-(i – 1) region is treated just like a local net was treated before. There are two important differences, however.

The first difference is that the gateway descriptors for a lower level region are not collected by using remote broadcasts (as before) but by sending a *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* message to the lower level Routing Server.

A second difference lies in the fact that now the level-i Routing Server has to find a

route to the Routing Server for the level-(i – 1) region, in which it resides, before any topology information can be collected. This is done by getting the route to the level-1 Routing Server from a gateway on the same local net and then by sending a series of *I-would-like-to-get-information-on-Routing Server level-(i – 1)* messages to higher and higher level Routing Servers until the route to the level-(i – 1) Routing Server is found. An optimization that allows the level-i Routing Server to avoid contacting all Routing Servers in the sequence from the level-1 Routing Server up to the level-(i – 2) Routing Server is tucked in.

Apart from these two major differences, this algorithm is basically the same as the one for the level-1 Routing Server. Hence, the same correctness arguments apply.

3.5 Computing routes in the campus-wide network

It was mentioned in Sec. 3.3 that it is advantageous to use shortest hop paths in the campus-wide environment. The Routing Service will, therefore, attempt to convert its topology information into an equivalent graph and then apply a shortest hop algorithm to find shortest hop paths. Inventing shortest hop path algorithms is not the subject of this thesis; there are plenty available already. The shortest hop algorithm described here is simpler than several well-known shortest path algorithms that compute shortest paths in graphs with directed or variable-cost edges. For example, Dijkstra's algorithm [1, 2, 8, 16, 5] finds shortest paths in a directed graph with no negative-cost (but variable) edges. The simplicity of the shortest hop algorithm, which the Routing Service will use, arises from the fact that in this case all paths are bi-directional and all hops are treated equivalently (i.e., no variable costs are assigned to hops).

As mentioned before, the Routing Service has first to convert its topology information into an equivalent graph before applying its shortest hop algorithm. A level-i (where i is 1 or greater) Routing Server will treat each of the level-(i – 1) regions in its territory as a node in the graph. Furthermore, each link between any two level-(i – 1) regions

will be treated as an edge in the graph. Therefore, one gateway in the region can correspond to more than one edge in the graph. For example, if a gateway connects three level-(i – 1) regions, then each level-(i – 1) region will correspond to a node in the graph and the gateway will correspond to three edges— one between each pair of nodes in the graph.

After constructing the graph for its region, a level-i Routing Server will apply a shortest hop algorithm to find shortest hop paths between all pairs of nodes in the graph. This, in turn, is done by first constructing shortest hop paths from one node to all other nodes in the graph and then repeating the same algorithm for each other node. The algorithm that finds shortest hop paths from one node to all other nodes is described below. It essentially does a breadth-first search of the graph starting from some source node and outputs the shortest hop routes (as well as the number of hops in the routes) from the source node to the other nodes as it does the search. The shortest hop algorithm will use a FIFO queue.

The shortest hop algorithm

1. Enqueue each edge from the edge list of the starting node, with # of hops = 1 and path to the node set as this edge.

2. Dequeue first edge. Output node with its associated path and # of hops. Also, insert the path in the table of shortest hop paths for the region.

3. For the node in the previous step, check its edge list. For each node in the edge list, do the following. If the node in the edge list has already been output, then do nothing. If, on the other hand, the node has not been output yet, then enqueue it with # of hops equal to one plus the # of hops output in the last step. Also, the path to the node should be set to the path output in the last step with this last edge added on at the end.

4. Repeat steps 2 and 3 till the queue is empty.

The algorithm described above only finds shortest paths from one node to all the other nodes. The algorithm can be repeated #nodes times to find shortest hop

paths between all pairs of nodes.

The algorithm described so far has one flaw. If one edge is deleted, the whole algorithm has to be repeated all over again. A complete repeat can be avoided by finding a path between the nodes on the two sides of the deleted edge[20]. This path can then be spliced into any path in place of the deleted edge. The resulting path may not be a shortest hop path but the path will serve as a temporary measure until shortest hop paths are computed again between all pairs of nodes. In fact, the same principle can be used to find paths that do not go through certain specified nodes or edges even if they may not have been deleted. In terms of the campus-wide network, the previous technique is applicable to finding routes that avoid certain regions or gateways. The technique will be used to implement user control of paths and to respond to faults in the network as will be seen later.

3.6 Answering queries about routes

This section is organized as follows. A description will be given first of the specification of hierarchical addresses. Next, some special features that are necessary for users to ask the the Routing Service for routing information will be explained. A description of the exact procedure to be employed by both users and the Routing Service to process queries for routing information will follow.

3.6.1 Specification of hierarchical addresses

A great deal of flexibility has been provided in the way names of attachment points can be specified. As explained before, these names are hierarchical.

---

[20] In fact, the algorithm described above can be used to do this if the source node is set to one of the nodes on either side of the edge. There is another reasonable approach to this problem. It is possible to use shortest path algorithms that require only an incremental calculation rather than a complete recalculation of all shortest path routes for a single change in network topology, e.g. the new ARPANET routing algorithm [12]

For a network of level-n, a *completely specified hierarchical address* for an attachment point would look as follows:

$$UID_{(n-1)}/UID_{(n-2)}/.../UID_1/UID_0/LTA,$$

where $UID_{(n-1)}$ is the unique identifier of the level-(n – 1) region in which the attachment point lies, $UID_{(n-2)}$ is the unique identifier of the level-(n – 2) region in which it lies and so on. Similarly, UID_0 is the unique identifier of the level-0 region (i.e., the local network) in which it lies, and LTA is the local transport address of the attachment point on the local network.

However, a hierarchical address need not be completely specified as shown above. A hierarchical address may have an *implicit prefix*. An *implicit prefix* can be used if the higher level fields that are left out correspond to the higher level regions to which the originator of the query belongs. For example, a hierarchical address might look as follows:

$$UID_i/UID_{(i-1)}/.../UID_1/UID_0/LTA,$$

where i is less than (n – 1).

There is another variation on hierarchical addresses. A hierarchical address may have *omitted components*. This is used when the unique identifiers of some regions are not known. Those fields that are unknown may be omitted. It is not allowed to omit the highest level field of the hierarchical address after an *implicit prefix*. An example of a hierarchical address with *omitted components* is as follows:

$$UID_i/UID_{(i-1)}//UID_{(i-3)}/.../UID_0/LTA.$$

In the example shown above, the unique identifiers of the level-(n – 3) and level-1 regions were not known and they were omitted. In fact, the example shown above illustrates another feature; it is possible that a hierarchical address may have an *implicit prefix* and *omitted components* at the same time.

3.6.2 Some special features required for answering queries

There are essentially two special messages required. The format for their description is the same as before.

1. *What-is-the-route-to-the-Routing Service?* message:

   To-gateway(What-is-the-route-to-the-Routing Service?)

   From-gateway(What-is-the-route-to-the-Routing Service?, route, unique identifier of the local network)

   The route that is returned in the reply is the route to the level-1 Routing Server.

2. *What-is-the-route-from-the-source-to-the-destination?* message:

   To- Routing Server(What-is-the-route-from-the-source-to-the-destination?, hierarchical address of source, hierarchical address of destination, unique identifier of destination)

   From- Routing Server(What-is-the-route-from-the-source-to-the-destination?, status of query, route, actual hierarchical address of destination)

   The hierarchical address of the source must be specified up to the level-0 address only (i.e., only the local net field and the LTA field need to be filled in).

   The "status of query" field can say any of the following things: "Route found", "No such destination found", or "No path to destination found". Each possibility for the "status of query" field is self-explanatory. If a route is found, then it is sent back in the "route" field of the reply. Also, if the message to the Routing Server contains omitted components, and the exact address (of the destination) is discovered by the Routing Server, then it is returned in the "actual hierarchical address of the destination" field of the reply.

   At this point, it is reasonable to ask if it is plausible for a source node to know the hierarchical address of the destination node. It is definitely not plausible for each node to find the hierarchical address of all other nodes by itself. However, it is possible for a source node to ask some node

name location service in the network to map a destination node name[21] into the hierarchical addresses of the attachment points to which the destination node is connected.

3.6.3 How are routes looked up?

To begin with, assume that a hierarchical address may have an implicit prefix but it may not contain any omitted fields.

To make a routing query, a node must first know the route to the level-1 Routing Server. This is accomplished by sending a *What-is-the-route-to-the-Routing Service?* message to the gateways on the local net of the node[22]. The route that is sent back by the gateways is the route from any node on the local net to the level-1 Routing Server. (This information is retrieved by the gateway from its descriptor). Another piece of information that is sent back is the unique identifier of the local net. This information is required by the node when it makes a request to the Routing Server. (The node may already have this information as part of its node descriptor if it has been there long enough for the Routing Service to have updated the descriptor.)

The next thing that the node should do is send a *What-is-the-route-from-the-source-to-the-destination?* message to the level-1 Routing Server. The only fields required in the hierarchical address of the source are the unique identifier of the local net and the LTA of the attachment point.

On receiving the request, the level-1 Routing Server checks the level-1 unique identifier field of the destination hierarchical address. If the unique identifier is the same as the unique identifier of the level-1 region of the Routing Server, then the Routing Server looks up its routing tables to find the route from the source to the

---

[21] This node name need not be the unique identifier that has been referred to before.

[22] This is done by putting all zeros (the common address for gateways) in the local transport address field of the message.

destination. Three possibilities exist—a route is found to the destination, no such destination is found, or no route is found to the destination (if the destination is found to exist but no paths lead to it due to temporarily broken gateways or subnets). Depending on the above three cases, the "status of query" field of the reply to the *What-is-the-route-from-the-source-to-the-destination?* message is appropriately filled in. If a route is found, then it is sent back in the reply. The field that contains the actual destination address may be left blank for now.

When the Routing Server checks the level-1 field of the destination address, it is possible that the level-1 field may not correspond to the level-1 region of the Routing Server. In this case, the Routing Server checks higher level fields to see if they correspond to the unique identifiers of any higher level Routing Server that it has information on[23]. For the first such field, as the Routing Server scans from the lower level fields up to the higher level ones, the Routing Server does the following: Suppose that the field in question is the level-j field. It was explained before that if the level-1 Routing Server has information on the level-j Routing Server above it, then it also has information on all the levels of Routing Servers between it and the level-j Routing Server. The level-1 Routing Server will now send a *What-is-the-route-from-the-source-to-the-destination?* message to the level-j Routing Server after filling in the hierarchical address of the source up to the level-j field.

The response of the level-j Routing Server to a *What-is-the-route-from-the-source-to-the-destination?* message is described next. The Routing Server will check to see if the level-j field of the destination address matches the unique identifier of its level-j region. Since the two match in this case, the Routing Server knows that both the source and the destination lie in its level-j region and that there is no need to send the message any further up the hierarchy.

---

[23]Having information on Routing Servers is understood, in this context, as knowing the unique identifier and the route to the Routing Server.

The level-j Routing Server looks up its routing tables to find the route from the level-(j – 1) region in which the source lies to the level-(j – 1) region in which the destination lies. Consider the example in Fig. 3-4 of a level-j region. Assume that the source lies in region F and the destination in region C. One possible route that the level-j Routing Server could find in its routing tables is from region F to region D via gateway K and then from region D to region C via gateway J. Once the level-j Route Server decides on this route, then it has to send messages down to lower level Routing Servers to construct the different pieces of the route. In the present case, the level-j Routing Server will send *What-is-the-route-from-the-source-to-the-destination?* messages to the level-(j – 1) Routing Server of region F (to find the route from the source to gateway K), to the level-(j – 1) Routing Server of region D (to find the route from gateway K to gateway J), and to the level-(j – 1) Routing Server of region C to find the route from gateway J to the destination. After the replies from the Routing Servers of regions F,D, and C are received they will be concatenated in the right order to produce the complete route. Note that if (j – 1) > 1 (i.e., level-(j – 1) is not the same as level-1) then the Routing Servers of regions F,D, and C will have to send further *What-is-the-route-from-the-source-to-the-destination?* messages to lower level Routing Severs to find the routes asked of them.

After the complete route has been found by the level-j Routing Server, it is sent down to the level-1 Routing Server as the reply to the *What-is-the-route-from-the-source-to-the-destination?* message that was sent by the level-1 Routing Server to the level-j Routing Server. The route is, in turn, passed down by the level-1 Routing Server to the source as a reply to the *What-is-the-route-from-the-source-to-the-destination?* message that was sent by the source to the level-1 Routing Server[24].

The description above relates to what happens if the level-1 Routing Server is able to

---

[24]Note that it is possible to introduce an optimization here. The complete route can be sent down by the level-j Routing Server directly to the level-1 Routing Server. The same can be done later for the complete algorithm when there will be several different intermediate Routing Servers through which the complete route will pass. However, this optimization will be omitted for ease of discussion.

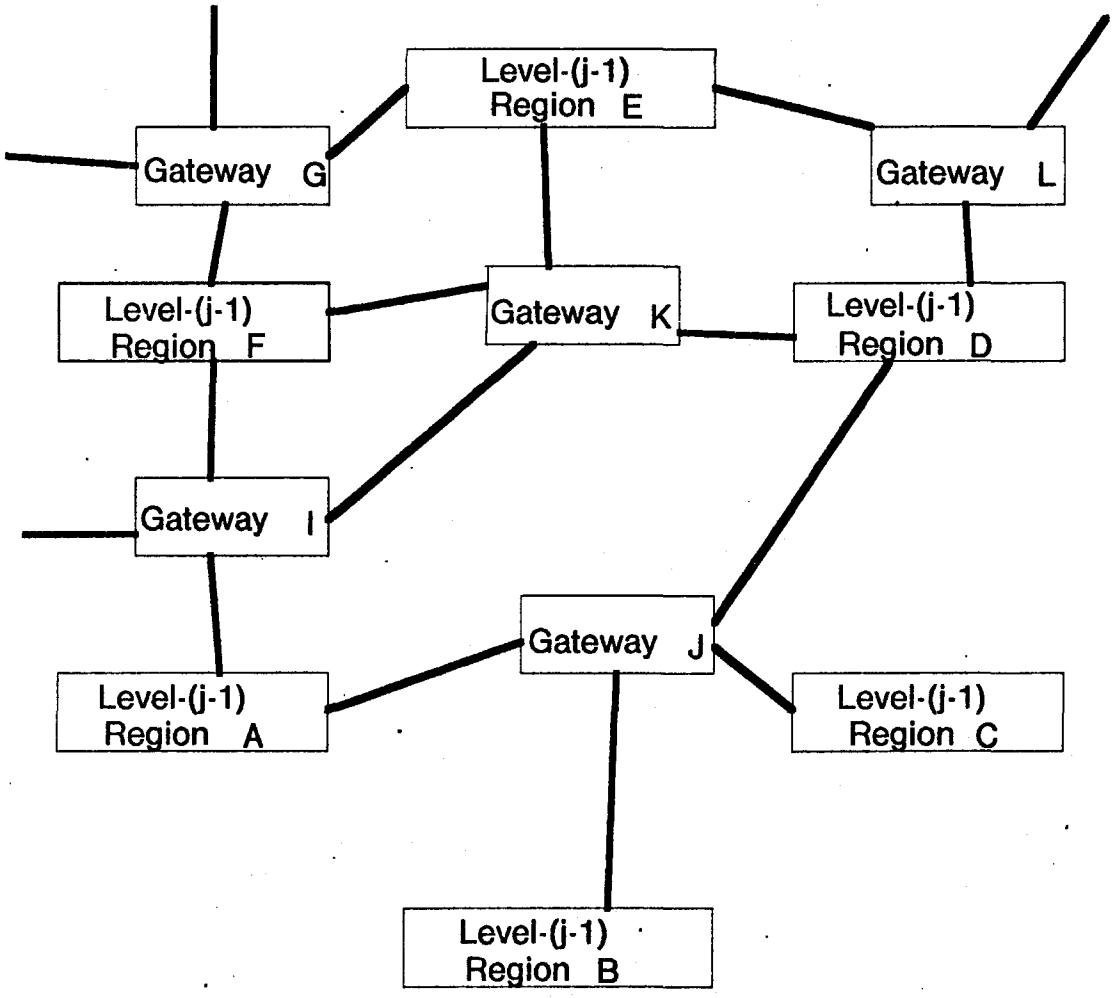

Figure 3-4: A level-j region

recognize some field of the hierarchical address of the destination as corresponding to some Routing Server that it has information on. If, however, no such field is recognized, then the level-1 Routing Server sends a *What-is-the-route-from-the-source-to-the-destination?* message to the level-k Routing Server (where $k \leq (i+1)$ and is the highest number such that the level-1 Routing Server has information on the level-k Routing Server). The level-k Routing Server now does what the level-1 Routing Server did before (i.e., it scans the hierarchical address of the destination up from the level-(k + 1) field to the level-i field and sends a *What-is-the-route-from-the-source-to-the-destination?* message to an appropriate Routing Server above it in the hierarchy). The level-k Routing Server uses the same algorithm as the level-1 Routing Server did to decide which level Routing Server to send a *What-is-the-route-from-the-source-to-the-destination?* message to.

The algorithm that is used by a level-j Routing Server to handle requests for routing information is given below. It essentially puts down in algorithmic form what was described above.

1. On getting a *What-is-the-route-from-the-source-to-the-destination?* message, check the level-j field of the destination address. If the level-j field does not exist (i.e., the destination address was only specified up to the level-(j – 1) field), then go to step 3, else go to step 2.

2. Check the level-j field of the destination to see if it matches with the unique identifier of the level-j region. If it does, then go to step 3. If it does not, then go to step 5.

3. Check the level-(j – 1) field of the destination address to confirm that the unique identifier corresponds to the unique identifier of one of the level-(j – 1) regions below it in the hierarchy. If it does not, then compose a reply to the *What-is-the-route-from-the-source-to-the-destination?* message (that was received earlier) by putting "No such destination found." in the "status of query" field and then send it to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.

   However, if the level-(j – 1) field corresponds to the unique identifier of one of the level-(j – 1) regions below it, then go to step 4.

4. If j = 1, then look up the routing tables to find the route from the source to the destination and send it to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.

   If j > 1, then find the route in terms of level-(j – 1) interconnections and send appropriate *What-is-the-route-from-the-source-to-the-destination?* messages to lower level Routing Servers. After receiving all replies, concatenate the parts of the routes received in the right order to construct the complete route. Send this route to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message. (If it turns out that lower level Routing Servers send back "No such destination found" or "No path to destination found" in the "status of query" field of their replies, then do the same to the reply sent to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.) Exit from the algorithm.

5. Look at the destination address and find the lowest k > j such that the level-k field of the destination address corresponds to the unique identifier of the level-k Routing Server that it knows about. If no such k is found, then make k equal to the highest level of Routing Servers that it has information on such that $k \leq (i+1)$, where i is the highest level specified in the destination address. Send a *What-is-the-route-from-the-source-to-the-destination?* message to the level-k Routing Server after filling in the "actual hierarchical address of destination" field up to level k. Get the reply to this *What-is-the-route-from-the-source-to-the-destination?* message and pass it on to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message that was sent to the level-j Routing Server. Exit from the algorithm.

Note that the algorithm described above works only if the hierarchical address of the destination does not contain any omitted fields. The algorithm for handling routing queries when the destination address may contain omitted fields is somewhat different from the previous algorithm.

Consider the case of a level-j Routing Server that is asked to construct a route from a source to a destination when the level-(j – 1) field of the destination address is omitted. The level-j Routing Server handles this by sending an appropriate *What-is-the-route-from-the-source-to-the-destination?* message (to find the route from some place in the network to the destination) to each of the level-(j – 1) Routing Servers

below it instead of to just one (which is what would happen if the level-(j – 1) field were specified).

One way to look at this situation is that the Routing Server basically treats this case as *m* different routing requests, where *m* is the number of level-(j – 1) Routing Servers below it (one for each possible level-(j – 1) field of the destination address). After all the routing servers below it send their replies to the *What-is-the-route-from-the-source-to-the-destination?* message, the level-j Routing Server constructs its own reply to the *What-is-the-route-from-the-source-to-the-destination?* message that *it* received.

If only one of the replies from the level-(j – 1) Routing Servers sent back a route to the destination, then the level-j constructs its reply just as it would if it had known all along that the destination existed in that particular level-(j – 1) region and if it had sent a *What-is-the-route-from-the-source-to-the-destination?* message to only that level-(j – 1) Routing server. Observe that due to multi-homing, more than one level-(j – 1) Routing Server may send back a route. In that case, the best (i.e., the one with the shortest hops) is chosen.

Even if no level-(j – 1) Routing Server sends back a route to the destination, one or more may say, "No path to destination found." In this case, any one of them can be thought of as the place where the destination exists. The reply to the *What-is-the-route-from-the-source-to-the-destination?* message that the level-j Routing Server received earlier will now also say, "No path to destination found."

If none of the replies sent back either a route or said "No path to destination found", then the reply that the level-j Routing Server sends back will say, "No such destination found" in its "status of query field."

The modified algorithm for a level-j Routing Server is presented below:

1. On getting a *What-is-the-route-from-the-source-to-the-destination?* message, check the level-j field of the destination address. If the level-j

field does not exist (i.e., the destination address was only specified up to the level-(j – 1) field), then go to step 3, else go to step 2.

2. Check the level-j field of the destination to see if it matches with the unique identifier of the level-j region. If it does, then go to step 3. If it does not or if the field is omitted, then go to step 6.

3. Check the level-(j – 1) field of the destination address to confirm that the unique identifier corresponds to the unique identifier of one of the level-(j – 1) regions below it in the hierarchy. If it does not, then compose a reply to the *What-is-the-route-from-the-source-to-the-destination?* message (that was received earlier) by putting "No such destination found." in the "status of query" field and then send it to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.

   If the level-(j – 1) field corresponds to the unique identifier of one of the level-(j – 1) regions below it, then go to step 4.

   If the level-(j – 1) field is omitted, then go to step 5.

4. If j = 1, then look up the routing tables to find the route from the source to the destination and send it to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.

   If j > 1, then find the route in terms of level-(j – 1) interconnections and send appropriate *What-is-the-route-from-the-source-to-the-destination?* messages to lower level Routing Servers. After receiving all the replies, concatenate the parts of the routes received in the right order to construct the complete route. Send this route to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message. (If it turns out that lower level Routing Servers sent back "No such destination found" or "No path to destination found" in the "status of query" field of their replies, then do the same to the reply sent to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message.) Exit from the algorithm.

5. If j = 1, then this means that the level-0 field or the local net field was omitted. The level-1 Routing Server should use the unique identifier of the destination to find the local net in which the destination exists and then find the appropriate route.

   If j > 1, then treat this case as if there were actually *m* different requests, one for each of the *m* level-(j – 1) Routing Servers below the level-j

Routing Server. Therefore, send a *What-is-the-route-from-the-source-to-the-destination?* message to each of the level-(j – 1) Routing Servers. It is important that the level-j Routing Server should fill in the level-(j – 1) field of each message with the unique identifier of the region to which the *What-is-the-route-from-the-source-to-the-destination?* message is being sent. After receiving all the replies, the level-j Routing Server should put together a reply to the *What-is-the-route-from-the-source-to-the-destination?* that *it* received earlier.

If one or more of the replies contain routes to the destination, then select the best one (i.e., the one with the least hops) as the one to be used for the reply to the *What-is-the-route-from-the-source-to-the-destination?* message that was sent to the level-j Routing Server earlier.

If none of the replies contain a route to the destination, but one or more say "No route to destination found", then send back the same information in the reply to the *What-is-the-route-from-the-source-to-the-destination?* message that was sent to the level-j Routing Server earlier.

If none of the replies contain a route to the destination or say "No route to destination found" (i.e., they all say "No such destination exists"), then send the same information back in the reply to the *What-is-the-route-from-the-source-to-the-destination?* message that was sent to the level-j Routing Server earlier. Exit from the algorithm.

6. Look at the destination address and find the lowest $k > j$ such that the level-k field of the destination address corresponds to the unique identifier of the level-k Routing Server that it knows about. If no such k is found, then make k equal to the highest level of Routing Servers that it has information on such that $k \leq (i+1)$, where i is the highest level specified in the destination address. Send a *What-is-the-route-from-the-source-to-the-destination?* message to the level-k Routing Server after filling in the "actual hierarchical address of destination" field up to level-k. Get the reply to this *What-is-the-route-from-the-source-to-the-destination?* message and pass it on to the originator of the *What-is-the-route-from-the-source-to-the-destination?* message that was sent to the level-j Routing Server. Exit from the algorithm.

3.7 Changing the configuration of the network

There are essentially five different types of changes allowed in the configuration of the network. They are described below along with the procedures that should be used when they occur.

1. *Adding or taking away a region from the network*: Consider the case of a level-(i – 1) region that is to be added to a level-i region. Fig. 3-5 shows exactly such a case. 'R' stands for a level-(i – 1) region and 'GW' stands for a gateway. As can be seen from the figure, the new level-(i – 1) region that is to be added has to be connected to the level-i region through two new gateways. To make this change in the configuration, both gateways must be given their descriptors. Remember that a gateway descriptor need only contain the following information when the gateway is first initialized: the number of local nets that the gateway is connected to, the local transport address on each subnet, and the highest level of the gateway between each pair of local networks that it is connected to.

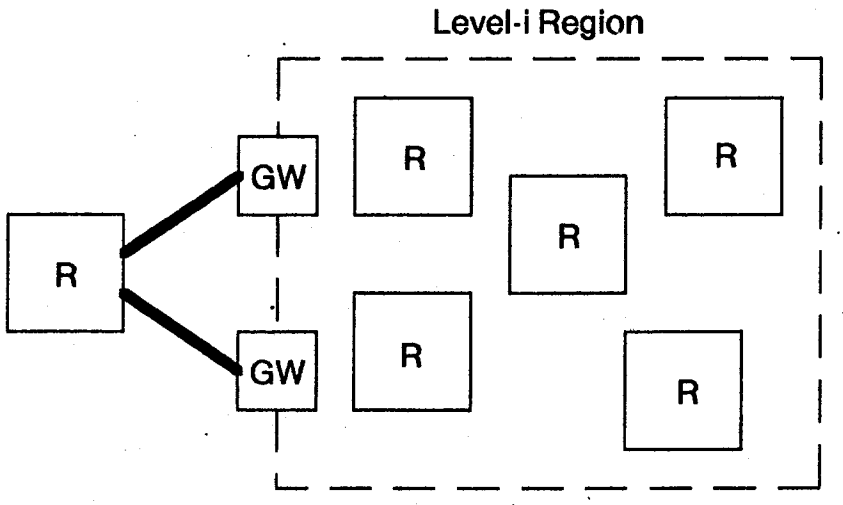

Figure 3-5: Adding or taking away a level-(i – 1) region

In the future, when the Routing Servers on either side of the gateway execute their topology-finding algorithms, they will notice these

gateways and they will change their topology data base accordingly. This will occur after the gateways get their descriptors.

However, it is not necessary to wait for the next iteration of the topology-finding algorithm. The gateways can be made to report their presence to the Routing Service on their own initiative. Each new gateway can send a *What-is-the-route-to-the-Routing Server?* message on each local net to which it is connected. Through this message it can find the route to the level-1 Routing Server on each side, and also the unique identifier of the local net. The new gateways can then send replies to a *What-is-your-descriptor?* message to each of the level-1 Routing Servers (although no *What-is-your-descriptor?* messages were sent by the Routing Servers). When the level-1 Routing Servers receive the messages from the gateways, they will update their data bases accordingly and send *Fill-in-your-descriptor* messages to the gateways, if necessary, to update their descriptors. Next, the level-1 Routing Server will send replies to *Give-me-descriptors-of-gateways-at-the-edge-of-your-region* messages to the level-2 Routing Servers above them just as they would at the end of an iteration of the topology-finding operation. This way the gateways will make their presence known to the level-1 Routing Servers, which will, in turn, inform the level-2 Routing Servers and so on. Information on the new gateways will, therefore, percolate up the hierarchy of Routing Servers.

If on the other hand, the level-(i – 1) region shown outside the dotted line was actually part of the old level-i region but is now to be removed from it, then the following is done. The descriptors of the gateways are changed accordingly and the gateways inform the level-1 Routing Servers on either side about the change by using a reply to the *What-is-your-descriptor?* message. If it turns out that the gateways were only being used to connect the level-(i – 1) region to the rest of the level-i region and will not be needed now, the gateways still have to report to the level-1 Routing Servers on either side about the change. The gateway descriptor that is sent to the Routing Servers will then say that the gateway is not connected to any local networks. In fact, this is the way any gateway informs the Routing Service that it is being removed from the network.

2. *Splitting up a level-i region into two level-i regions*: Consider the example illustrated in Fig. 3-6. As shown in the figure, two gateways presently connect the regions that will later become full-fledged level-i regions.

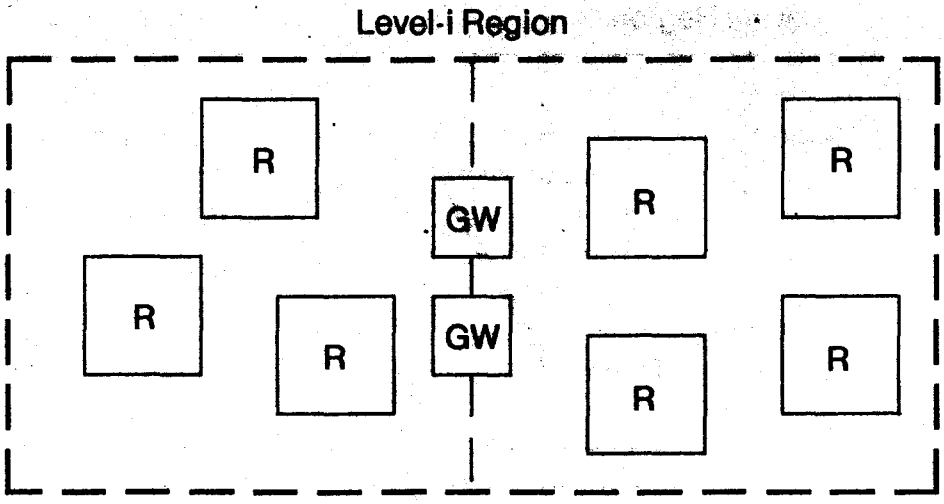

Figure 3-6: Splitting up a level-i region

Both the gateways have to be taken down and given new descriptors. After that, the gateways can either wait for the next iteration of the topology-finding operation of the Routing Service or they can inform the Routing Service of their presence themselves, as before.

It is necessary to take down both gateways before giving any of them a new descriptor. If this is not done, a peculiar situation can arise. Suppose that one of the gateways has got a new descriptor but the other one has not. It is possible that an iteration of the topology-finding algorithm may be completed between the times that the two gateway descriptors are changed. This will result in the Routing Service getting an inconsistent view of the network and it should be avoided.

3. *Increasing one region and decreasing another*: Consider the example illustrated in Fig. 3-7. Gateways 'a','b', and 'c' are used to interconnect the two level-i regions. However, gateway 'a' is the only one connected to region 'h'.

   The idea is to increase the level-i region (to the left) so that it also contains the level-(i – 1) region 'h'. At the same time the level-i region to

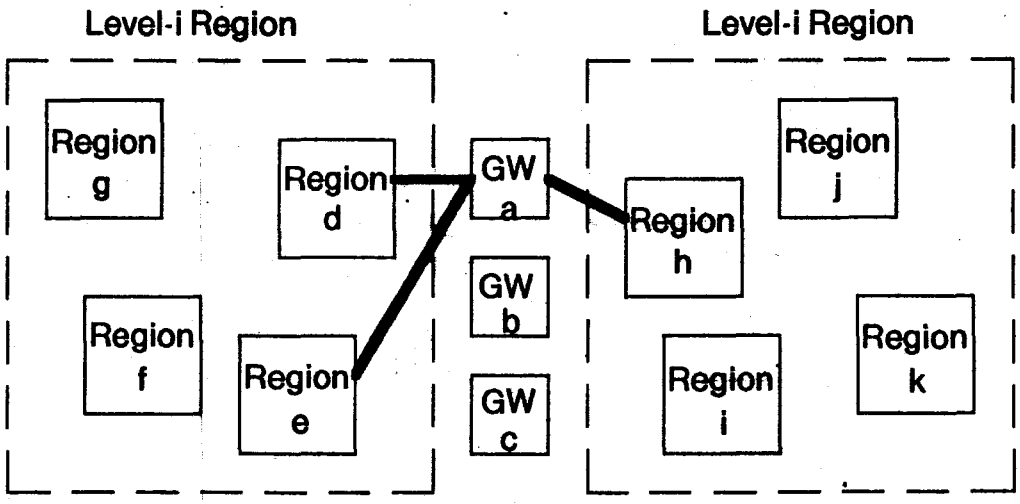

Figure 3-7: Increasing or decreasing a level-i region

the right should lose the level-(i – 1) region 'h'.

This is done by taking down gateway 'a' and all the gateways in the level-i region to the right that are connected to region 'h'. Next, all these gateway descriptors are updated appropriately. These changes are made known to the Routing Service in the same manner as before.

4. *Merging two level-i regions*: Consider the example illustrated in Fig. 3-8. The idea is to merge the two level-i regions shown into one level-i region. At present, the two level-i regions are interconnected by exactly three gateways: gateway 'a', gateway 'b' and gateway 'c'.

   Here, again, each of the gateways is taken down and given a new descriptor to reflect the change in configuration. The Routing Service is informed exactly as before about the changes.

5. *Removing, adding or modifying a gateway/node*: It is possible that a gateway may be removed, added, or merely have its descriptor modified and yet not change the hierarchical configuration of the network. Although this case clearly does not fit in with a list of the basic types of changes in configuration, it has been included here because it is

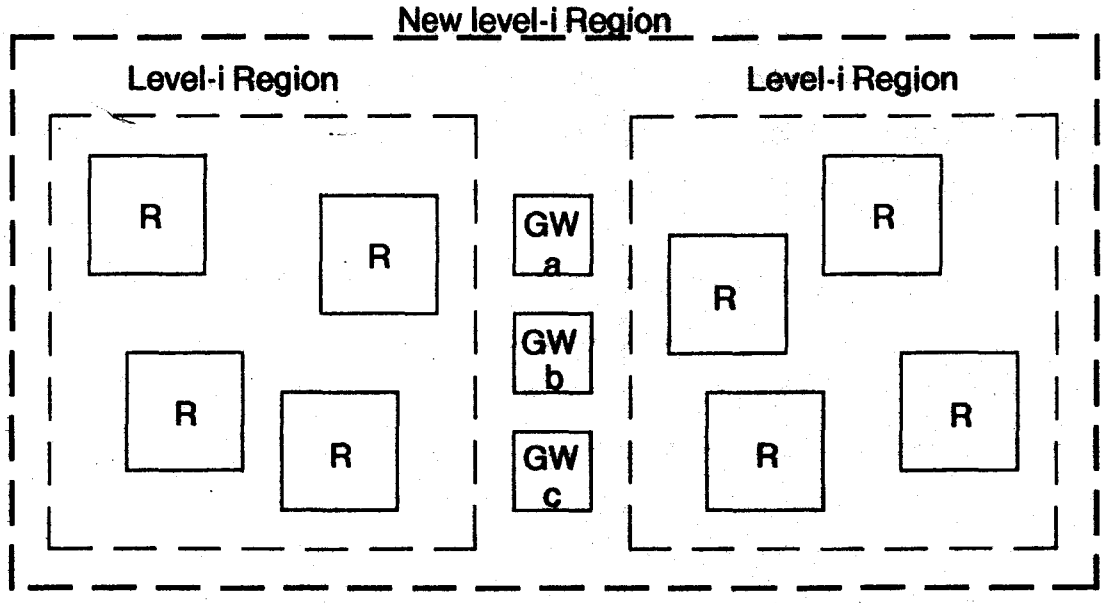

Figure 3-8: Merging two level-i regions

handled in a very similar way.

Whatever change the gateway undergoes, its descriptor is changed accordingly and the new gateway descriptor is reported to the Routing Service in the manner previously described.

Much like the changes that a gateway can undergo, a node may be removed, added, or have its descriptor modified. The change in the node descriptor may be due to the node being connected to a different attachment point in the network. In any event, the node can either wait for the Routing Service to collect topology information or it can report the changes in its descriptor to its level-1 Routing Server immediately; the latter will be done by sending replies to *What-is-your-descriptor?* messages to the level-1 Routing Servers of all the local nets that it is connected to—exactly as a gateway would report a change to the Routing Service.

3.8 User control of paths

The description of the advantages of source routing in Chapter 2 mentioned that in a source-routing environment it is possible to choose a route with a specified class-of-service standard. It is time to admit now that although that was a theoretical possibility, implementation of a class-of-service feature is a very difficult task.

To begin with, it is far from easy to *collect* the class-of-service information on gateways and regions. Consider the wide range of properties of paths that users may be interested in—error rate, transport delay, bandwidth, security rating, etc.. To measure some of these requires sophisticated procedures and takes us beyond the design of a Routing Service. It is for this reason that gateway descriptors and node descriptors contain class-of-service information but it has not been described where they get their information. The reason that class-of-service was introduced at all in gateway and node descriptors was so that future designers or implementors of a Routing Service would have a handle to work with if they decided to incorporate a class-of-service feature.

Not only is it not easy to collect class-of-service information, it is extremely difficult to have the Routing Service automatically select a path to meet certain class-of-service standards. It is relatively easy for the Routing Service to check for certain class-of-service properties that each hop in a path should satisfy, e.g. bandwidth. However, there is no easy way out if the class-of-service rating of a path depends on some kind of aggregate of the rating for each link in the path. For example,

$$\text{The error rate over a path} = 1 - (1 - \text{error rate}_1)*(1 - \text{error rate}_2)*\ldots*(1 - \text{error rate}_n),$$

where error rate_i is the error rate (expressed as a fraction of one) of the i^{th} link in a path. Similarly, the transport delay of a path is the sum of the transport delays over its various parts. As can be seen, the class-of-service rating of a path for various properties can be a radically different function of the rating for each different link in the path. Also, it is not clear exactly which subset (or superset) of these properties is

useful under which circumstances. The Routing Service has, therefore, not been endowed with the ability to select a path given some class-of-service standards (specified by the user). Instead, a more restricted but general way for the user to exercise control over paths has been provided. There is a way for the user to ask for a path (from it to some destination) that does not pass through certain specified gateways and regions.

User control of paths requires a special feature of the Routing Service—a message. The message is a variation on the *What-is-the-route-from-the-source-to-the-destination?* message. The special message looks like this:

To-Routing Server(What-is-the-route-from-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?, list of unique identifiers of gateways to be avoided, list of unique identifiers of regions to be avoided, hierarchical address of source, hierarchical address of destination, unique identifier of destination)

From-Routing Server (What-is-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?, status of query, route, actual hierarchical address of destination)

In fact, the regular *What-is-the-route-from-the-source-to-the-destination?* message can be thought of as a special case of the message given above with the list of unique identifiers of gateways to be avoided and the list of unique identifiers of regions to be avoided left empty.

The message described above is utilized by the user to send a query to the level-1 Routing Server. It is also employed by the Routing Servers for internal communication just as with the *What-is-the-route-from-the-source-to-the-destination?* message. Sec. 3.5 has already described how routes are computed to avoid certain parts of a network.

3.9 Responding to faults in the campus-wide network

As explained in Sec. 3.4, the Routing Service periodically checks on the topology of the network. While doing that, it also picks up any temporary changes in the topology caused by broken gateways or subnets. If the frequency with which the Routing Service updates its topology is high enough, there is really no need for any other mechanism for the Routing Service to detect faults and find alternative paths, if necessary.

However, if there is a considerable delay between topology updates, then it is useful to provide a feature whereby entities in the network that notice broken gateways or subnets can report the same to the Routing Service. Therefore, faults in the network can be noticed faster and alternative routes can be found, if necessary, without waiting for another topology update.

In fact, in the previous section, the means for a user to find paths that do not go through certain specified gateways and regions have already been provided. However, this does not provide a convenient way for a user to find an alternative path if he finds that a path he is trying to use does not work. Before he can ask the Routing Service to find another path, he must pinpoint the fault. This may be beyond the capability of some users. A better way would be to have the source merely report a bad path to the Routing Server and let the Routing Server find a path that avoids the faulty part of the route. This solution to the problem requires some special features, too, and they have been described next. After describing the special features, the way these special features can be used to detect faults and to find alternative routes will be described.

3.9.1 Special features required for responding to faults

There are essentially three messages required to support fault-finding. They are described below.

1. *Path-does-not-work* message:

To- level-1 Routing Server(Path-does-not-work, path, hierarchical address of source, hierarchical address of destination, unique identifier of destination)

From- level-1 Routing Server(Path-does-not-work, path)

The path that the Routing Server returns will be a path to the destination but one that does not go through the faulty part of the previous path.

2. *Return-packet* message:

To-gateway/node(Return-packet, test packet)

From-gateway/node(Return-packet, test packet)

If a gateway/node is sent the above message, then it merely sends back the test packet to the originator of the message. This will be used to test out the path from the originator of the message up to the gateway/node.

3. *Give-me-the-unique identifier-of-a-gateway* message:

To- level-1 Routing Server(Give-me-the-unique identifier-of-a-gateway, hierarchical address of gateway)

From- level-1 Routing Server(Give-me-the-unique identifier-of-a-gateway, unique identifier of gateway)

Only the LTA field and the level-0 field of the hierarchical address of the gateway needs to be specified. The level-1 Routing Server will use that information to find the unique identifier of the gateway by looking up its topology data base.

3.9.2 The procedure for finding alternative paths

When a user suspects that a path does not work, he should do the following to find another working path to the same destination. He should first send a *Path-does-not-work* message to his level-1 Routing Server complaining about the faulty path. It is the duty of the Routing Server now to carry out diagnostic tests on the faulty path to pinpoint the first fault in the path (which may be either a broken gateway or a broken subnet). This is done by using the *Return-packet* message repeatedly. *Return-*

packet messages can be sent down the suspect path to increasing distances until the last subnet or gateway to which a *Return-packet* message is sent does not reply.

Note that lack of response to a *Return-packet* message, in this case, makes the last step in the path suspect, but in an unreliable network packets can get lost anywhere. The correct protocol is repeatedly to send alternate *Return-packet* messages to the last step and the next-to-last step until the probability that a failure elsewhere produces the same result is low enough to ignore. The same technique will be used to resolve any other negative inference problems that arise while the faulty gateway/subnet is being located.

Moreover, lack of response from the last gateway (call it gateway *x*) in the path may imply either that the last gateway is faulty or that the last subnet (over which the packet travelled to get to the last gateway) is faulty. Therefore, the Routing Server will attempt to find out the operational status of the subnet in question. The Routing Server will remote broadcast a *Return-packet* message for gateways on the last subnet. If a reply is received, then the subnet will be assumed to be operational and the fault in the original path will be attributed to gateway *x*. If a reply is not received, it can imply one of three things; the last subnet does not have any other gateways besides the one used to remote broadcast on the subnet and gateway *x*, any other gateways that it does have are faulty[25], or the subnet is faulty. In any event, the subnet cannot be used by an alternative path and, therefore, it is reasonable to attribute the fault in the original path to this subnet. Also, note that although the fault may be attributed to the subnet, gateway *x* or, in fact, any other gateway or subnet later in the original path may be faulty. The faults will, however, get located in later invocations of the algorithm if the alternative paths produced by the Routing Service still go through the faulty gateways or subnets.

---

[25] If there were an alternative path going through the subnet, there would be another working gateway on the subnet.

Now, if the probable fault lies in a subnet, then a *What-is-your-descriptor?* message can be sent to the gateway that was used to get onto the subnet in the first place. The level-1 Routing Server that is carrying out the diagnosis can then find out the unique identifier of the local net by inspecting the contents of the gateway descriptor.

On the other hand, if the probable fault lies in a gateway, then again a *What-is-your-descriptor?* message can be sent to the gateway that comes immediately before the suspect gateway on the faulty path. The level-1 Routing Server can then inspect the contents of the gateway descriptor and find a route to the level-1 Routing Server of the local net on which the last two gateways lie. The Routing Server carrying out the diagnosis can then send a *Give-me-the-unique identifier-of-a-gateway* message to the other Routing Server and find out the unique identifier of the suspect gateway.

After the Routing Server carrying out the diagnosis pinpoints the gateway or subnet that is creating problems on the faulty path, then it can use the *What-is-the-route-from-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?* message to ask higher-level Routing Servers to help it find a path that does not go through the faulty gateway or subnet. Note that it is possible that the destination may lie in the same level-1 region as the source and the level-1 Routing Server may not have to go up the hierarchy of Routing Servers to find the alternative path. Moreover, if the fault is pinned down to the last subnet in the original path (i.e., the one to which the destination node is attached), then clearly no alternative paths can be found to the destination; the Routing Server need not use the *What-is-the-route-from-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?* message to try to find an alternative path.

3.10 Congestion control

Congestion control is still a topic of current research and several possible approaches to tackle congestion control are described in the literature [15]. Before ending the discussion on the design of the Routing Service, it is necessary to point

out that in a source routing environment with a centralized computation of routes, there exists an opportunity to attack the problem of congestion control in yet another way.

Since source routing provides complete control over paths, it is possible to find routes that do not pass through certain parts of the network. The section on user control of paths and the section on fault-finding described ways of computing such paths. A similar thing is possible with congestion control. If congested parts of the network can be pinpointed, then the Routing Service may be asked to find routes not going through the congested areas. However, it is not clear that such a feature is useful. The first reason to suspect that it may not be very helpful is due to the nature of a campus-wide network. Since all communication links are going to be relatively high-bandwidth (relative to a long-haul net), it is likely that congestion will take place one instant and go away shortly after. In other words, congestion is not likely to be a long-term problem in any part of the campus-wide net. If congestion only occurs in short bursts, then it is likely that by the time a Routing Service finds an alternative route, the problem may have disappeared.

Another reason to doubt the usefulness of the congestion control feature that has been described is as follows. Since traffic generated in a campus-wide network by any single node is likely to be in short bursts (unlike the traffic generated by a node in a long-haul network over a virtual circuit), and since gateways are simple and fast in a source-routing environment, it is unlikely that a given gateway will receive traffic from more than one source at a given instant. Therefore, if a gateway does get congested, it is more than likely due to traffic from just one source. It may not be wise to find an alternative path for the traffic from that source because the traffic will probably overload any new gateway put in the path. In such a situation, in fact, cutting down on the traffic generated by the source or sending alternate packets on different routes may be better than having the Routing Service find an alternative route to send all packets on.

One instance when the congestion control feature mentioned above is likely to be useful is when the congested gateway happens to be one of the low-bandwidth gateways that connect the campus-wide network with the long-haul networks. These low-bandwidth gateways are likely to remain congested for a longer time once they do get congested as compared to the other high-bandwidth gateways in the campus-wide network.

Even for low-bandwidth gateways, it is not clear that the congestion-control feature described above is a good way to tackle the problem. In a similar method of congestion-control for hop-by-hop routing, one would expect all alternative routes not passing through the congested area to be used before another solution to the congestion-control (such as source-quenching) is used. However, the congestion-control feature mentioned above merely finds any random path not passing through the congested area and makes no special effort to try out all possible alternatives for paths that do not pass through the congested area. It is, therefore, not easy to figure out when the congestion control feature mentioned should be abandoned in favor of some other congestion control feature.

There are a few other observations that should be made about the difficulty of using routing decisions as a mechanism to combat congestion control. The observations are listed below.

- "to the extent that the initial route selected for a packet was the best one, any other route is likely to consume more network resources" [15];
- "if congestion is due to a persistent cause, diverting traffic onto more routes only delays the cure and is likely to spread the congestion" [15];
- FIFO strategies are easy to implement in a source routing environment (see Chap. 2) but frequent routing changes wipe out the advantage. The constraint of sequential delivery conflicts with frequent routing changes, because more packets arrive out of order at the destination.

Since it is not clear that finding paths that avoid congested areas is a good way to tackle the problem of congestion, such a feature has not been incorporated into the

Routing Service. Higher-level protocols between the source node and the destination node can still be used as usual to control the traffic sent from the source to the destination for any prolonged exchange between the two nodes.

This chapter has described in detail the design of a Routing Service for campus-wide internet transport. The next chapter will evaluate this design to see how well it meets the requirements set out in Chap. 2.

Chapter Four

Evaluation

This chapter will evaluate the design of the Routing Service to see how well it meets the requirements of Chap. 2. Each of the eleven requirements has been examined in turn below to see how it affected design considerations.

4.1 The Routing Service has to work in a distributed environment

While discussing this requirement, it was emphasized that the number of messages required by the Routing Service must be kept low especially if it affects the response time to users. It is due to this requirement that the level-i topology-finding algorithm ensures that each level-(i – 1) region is visited only once in each iteration. Also, the algorithm for answering queries was designed so that the delay due to messages in answering a query is proportional to the hierarchical configuration of the network in the worst case; this claim is defended below. The delay due to message passing for answering the query is composed of the following:

1. The time taken for the query to reach the lowest level Routing Server (call it Routing Server 'a') which contains both the source and the destination in its region,

2. The time taken for the query to percolate down to all the level-1 Routing Servers on the path from the source to the destination,

3. The time taken by the lower level Routing Servers to send up the replies to Routing Server 'a', and

4. The time taken for the complete route to be sent down by Routing Server 'a' to the originator of the query via all the intermediate Routing Servers.

Note that although several different messages can be sent down from level-i Routing

Servers to level-(i – 1) Routing Servers in 2 and, similarly, although several different messages can be sent up from level-(i – 1) Routing Servers to level-i Routing Servers in 3, these are all done in parallel. Clearly, the serial delay involved in all the four times listed above is only proportional to the level of Routing Server 'a'. Therefore, the total serial delay due to message passing in answering the query is proportional to the level of Routing Server 'a', which, in the worst case, is the height of the hierarchical configuration of the network.

4.2 The Routing Service should be reliable

This implies that the Routing Service should be robust in the face of arbitrary changes in the topology or connectivity of the network. The topology-finding algorithms of the Routing Service were designed to pick up all changes in the topology or connectivity of the network. Moreover, since gateway and node-related changes are likely to be more frequent than other changes, a mechanism was devised to allow such changes to be reported to the Routing Service immediately.

A secondary issue connected to the reliability of the Service is the introduction of "extras" or "frills" into the Routing Service design. It was mentioned before that reliability is enhanced by keeping the design simple and free of "extras" that could be incorporated later. It was partly due to this concern that congestion control and class-of-service features were dropped from the design.

4.3 The Routing Service should be reasonably fast

As mentioned before, a hierarchical approach was taken to routing problems mainly to ensure faster service. With the hierarchical approach, no single Routing Server needs to have complete global knowledge of the network.

The Routing Service was also sought to be made simpler by choosing shortest hop routes as the kind of routes to be computed. Compared to sophisticated routing

strategies like the one used in the ARPANET, the Routing Service here spends much less effort on gathering information about the network and on computing routes.

4.4 The Routing Service should require minimal support from the rest of the system (especially gateways)

While describing the design of the Routing Service in Chap. 3, a lot of special features were described. The special features were used by the Routing Service in carrying out its various tasks. Fig. 4-1 contains a list of all the special features that have to be supported by nodes and Fig. 4-2 contains a list of all the special features that have to be supported by gateways.

The demands made on nodes and gateways to support these special features is minimal. Recall that one of the major motivations of using source routing and a Routing Service (to support source routing) is to make the gateways simple. A proper design of the Routing Service has ensured that gateways are kept simple.

4.5 The Routing Service should scale gracefully for larger networks

The three major tasks of the Routing Service are topology-finding, computation of shortest hop routes, and answering queries. The algorithms that were designed to perform the tasks mentioned should scale gracefully as the size of the network increases. Each of these functions of the Routing Service will be examined in turn below to see how well they scale with network size.

Topology-finding is accomplished by letting each level-i Routing Server find the topology of its level-i region in terms of interconnections of level-(i – 1) regions. Clearly, the size of the network will affect the number of Routing Servers in the network but the cost of the topology-finding algorithm of any level-i Routing Server is only dependent on the number of level-(i – 1) regions and gateways in the level-i region.

1. Nodes respond to a common local transport address

2. Node descriptor

3. *What-is-your-descriptor* message

4. *Fill-in-your-descriptor* message

5. *Return-packet* message

Figure 4-1: The special features supported by nodes

1. Gateways respond to a common local transport address

2. Remote broadcast

3. Gateway descriptor

4. *What-is-your-descriptor* message

5. *Fill-in-your-descriptor* message

6. *What-is-the-route-to-the-Routing-Service* message

7. *Return-packet* message

Figure 4-2: The special features supported by gateways

The previous argument holds for the computation of routes, too. A level-i Routing Server only computes routes between the level-(i - 1) Routing Servers in the level-i region. The cost of computing routes, for any Routing Server, is not dependent on the size of the complete campus-wide network.

Of course, one pays a price for using a hierarchical approach to routing. One of the disadvantages is that now it is more complicated to look up a route. A route may have to be constructed now from pieces of information gathered from several different Routing Servers. Answering queries about routes is, in fact, one of the

major tasks of the Routing Service.

Care was taken in the design of the algorithm for answering queries to ensure that the time taken for this function does not blow up as the size of the network increases. As explained before in this chapter, the serial delay due to message passing for answering any query is only proportional to the height of the hierarchical configuration, in the worst case. If each level-i region in the network has the same number of level-(i – 1) regions in it, then the height of the hierarchical configuration only grows as log of the number of subnets in the campus-wide network.

4.6 The Routing Service should have a good user interface

While describing this requirement, it was mentioned that this requirement would lose out if it conflicted with any of the requirements discussed above. There are various messages that a user can invoke to interact with the Routing Service. These messages are listed in Fig. 4-3. As can be seen from the list, the user interface consists only of messages required to support a very austere Routing Service (i.e., no exotic features have been introduced in the user interface).

1. *What-is-the-route-to-the-Routing Service?* message
2. *What-is-the-route-from-the-source-to-the-destination?* message
3. *What-is-the-route-from-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?* message
4. *Path-does-not-work* message

Figure 4-3: The user interface

A feature that would enable users to ask for routes that satisfy certain class-of-service standards was dropped because the area is not well understood yet.

4.7 The Routing Service should face up to changing network configurations

In Chap. 3, various ways of changing the network configuration were described. The basic types of changes were:

1. Adding or taking away a region from the network,
2. Splitting up a level-i region into two level-i regions,
3. Increasing one region and decreasing another,
4. Merging two level-i regions, and
5. Removing, adding or modifying a gateway/node.

All other reasonable changes in the network configuration can easily be broken down into the basic types of changes.

4.8 The Routing Service should face up to mobile hosts

The effect of mobile hosts is that knowing the exact hierarchical address of a node may be difficult. However, if the node is likely to be in a part of the network that can be specified by a hierarchical address with omitted components, then the same hierarchical address can be used to make a query. The Routing Service ensures that a route will be found to the node if the node exists in the specified area. A guarantee for locating the node in the specified part of the network does not carry with it a guarantee of good response time from the Routing Service. If the hierarchical address specified has too many omitted components for the Routing Service to handle efficiently, the query may even be assigned to some background handler for queries.

4.9 The Routing Service should face up to artificial partitioning

It may be possible to avoid most cases of artificial partitioning by introducing multiple routes and redundant Routing Servers. Although multiple routes and redundant Servers do not exist in the design right now, it is straightforward to incorporate them into the design.

There is one class of artificial partitioning problems for which a different approach is needed. Consider the case when no path exists between two attachment points in the same region but a path may exist if the path from the source goes outside the region and comes in again to a part of the region from which the destination can be reached. One way to approach this is for the Routing Server of the region to ask the Routing Server above it (in the hierarchy) for a path between two appropriate points at the edge of the partitioned region such that the path avoids the partitioned region; the *What-is-the-route-from-the-source-to-the-destination-avoiding-the-following-gateways-and-regions?* message can be used for this purpose.

4.10 The Routing Service should face up to multi-homing

Multi-homing can mean that several shortest hop paths exist to a destination node (one to each attachment point to which the destination node is connected). If some or all of the attachment points of the destination node lie in a part of the network that can be specified by a hierarchical address with omitted components, then it is straight forward to find the shortest hop path to the attachment points of the node that lie in the specified part.

The algorithm for answering queries ensures that if several shortest-hop paths exist to a destination node, then the shortest path among the possible paths is chosen. This approach clearly does not solve the problem of multi-homing completely because it does not work very well if the destination node is connected to attachment

points that are very far apart in the hierarchical configuration of the network[26].

4.11 The Routing Service should face up to shared access

It was mentioned while describing the operation of a gateway in a source routing environment that the algorithm executed by a gateway to route packets may be repeated inside the destination node to route packets to the correct activity. In fact, the same algorithm can also be used inside a shared network interface to route packets to the correct node. Of course, an extra field will be required in the source route of the packets for the network interface to be able to send packets to the correct nodes. A possibility is that this extra field could, by convention, be the unique identifier of the destination node.

4.12 Summary

The design of the Routing Service was most strongly affected by the requirement for scaleability. A hierarchical organization of Routing Servers was used and algorithms for topology-finding, computations of routes, and for answering queries were designed to fit in with the hierarchical structure. One effect of the hierarchical structure is that a faulty Routing Server affects only route finding for routes passing through the region of the faulty server. The Routing Service was designed to be reasonably fast to avoid the possibility of the Routing Service being a bottleneck in the network. The other key features of the Routing Service are simplicity and reliability. Reliability in a general sense means that the Routing Service should be able to face up to any event. Therefore, the service has been designed to work in the face of arbitrary changes in the topology of the network or its connectivity and also to

---

[26] "Far apart" should be taken to mean that if a single hierarchical address were used to specify a part of the network which included all the attachment points, then the address would need to have a lot of omitted components.

efficiently respond to changing network configurations, mobile hosts, artificial partitioning, multi-homing, and shared access.

This empty page was substituted for a blank page in the original document.

Chapter Five

Conclusions

This chapter summarizes the design goals for the Routing Service and presents the salient features of the design itself. The chapter also discusses areas for further improvement and research.

5.1 Summary of Routing Service Design

The original motivation of this thesis was clear and compelling—to simplify gateways and to aid modularity by separation of target identification and routing decisions from gateway implementation. Source routing seemed to provide the magic answer for the need described above. However, source routing by itself does not amount to much without a network service to provide routing information.

Several goals were laid down for the Routing Service itself. To begin with, the service should be reliable and it should be fast enough to avoid being a bottleneck in the system. Reliability, in a general sense, covers not only the changes in topology and connectivity but also the ability to deal with changing network configurations, mobile hosts, artificial partitioning, multi-homing, and shared access. Note that in a few cases the efficiency of the Routing Service was as much or more of a concern than reliability. For example, the Routing Service was designed to be efficient in the fact of mobile hosts—the service was not designed to just survive in the face of mobile hosts. Scaleability, or efficiency in the face of networks increasing or decreasing in size, was also a major design goal.

A hierarchical configuration was chosen for the campus-wide network to address the problem of scaleability. Also, an implementation of source routing that computes

reverse routes was chosen to facilitate discussion of the design in concrete terms. Moreover, it was decided that computing shortest hop routes was sufficient in the high-bandwidth campus environment—more sophisticated routes are not needed.

The actual design of the Routing Service was split up into seven parts. The first three parts are the backbone of the Routing Service. They deal with topology finding, computing routes, and answering queries about routes. Algorithms were described for each of the three basic functions. The topology-finding algorithm was described with two variations—one for a level-1[27] Routing Server and one for any other level Routing Server. The topology-finding algorithm was designed to be efficient and also reliable (to cope with any arbitrary topology and connectivity). The algorithms for computing routes was designed to compute shortest hop routes. The algorithm for answering queries was designed to be efficient in the face of various anticipated situations. First, since a number of Routing Servers had to cooperate to produce a route, care was taken to ensure that the time taken to produce a route was not unduly long. Second, the service was designed to be efficient to the extent possible in finding routes to mobile or multi-homed hosts. The algorithm for answering queries was also designed to be flexible enough to find routes when the hierarchical address of the destination is not completed specified.

Apart from the three basic functions of the Routing Service, four other facilities were incorporated into the design. First, several useful ways of changing the configuration of the network were described. Second, user control of paths was provided to an extent by allowing users to specify parts of the network to be avoided by a certain route. Third, the service was geared to diagnose faults in faulty routes and compute alternative routes. Four, it was suggested that congestion control could be provided by using global information about the network to compute appropriate routes.

The design of the Routing Service was evaluated in Chap. 4. The evaluation

---

[27] Level-1 is the lowest level in the hierarchy.

concluded that the goals set out in Chap. 2 were essentially met by the proposed design.

5.2 Future Work

This thesis has been a paper design of a Routing Service for campus-wide internet transport. An implementation of the Routing Service should be attempted as soon as possible to verify the essential correctness of the design. To make the implementation completely convincing, its performance should be monitored under normal as well as stressful conditions. Even if a Routing Service with all its functionality cannot be implemented (because a full-scale campus-wide net does not exist yet), a stripped down version should be implemented. If this approach is taken, it will be easy to build a complete Routing Service when the campus-wide network comes into existence.

There are several useful features that can be added to the Routing Service. Recall that a class-of-service feature was dropped from the current design because the related area is not well understood yet. Further investigation should be done on how to enable the Routing Service automatically to select paths to meet certain class-of-service standards specified by users.

The Routing Service presented in this thesis gives routes to users only when asked for them. These routes are expected to be cached and used until the user discovers that they do not work any more, or suspects that a better route may exist, or if the user loses the route for some reason. In either of the previous cases, the user has to ask the Routing Service for a new route. In some circumstances when the Routing Service discovers a better route between two points in the network, it might be more efficient for the Routing Service to directly send the route to its expected users (instead of waiting for a query). For the scheme to work well, however, it might be necessary for the Routing Service to be able to force routes into the caches of users. In fact, it is even acceptable to provide the Routing Service with a mechanism to

invalidate entries in user caches for routes; users could then ask for new routes for the invalidated entries when the need arises. This second mechanism is being suggested because it is possible that it may be easier to implement than a mechanism to force entries into user caches. Therefore, a mechanism to force routes into caches or a mechanism to invalidate entries in caches should be worked out.

One of the major problems of using routing decisions as a mechanism to spread traffic and optimize network usage is that in most networks only local information is available. Intuitively, it seems far-fetched to be able to achieve a network-wide optimization based only on local information. In fact, a recent study [9] indicates that flow-control power[28] is non-decentralizable. The study lends further credence to the idea that some sort of global information is required to tackle congestion control or flow control meaningfully. Since the Routing Service for the campus-wide network is in a position to choose routes based on global information, it may be possible to implement useful routing strategies for flow control or congestion control. Therefore, this area is open to a lot of interesting research.

Various ways of changing the hierarchical structure of the network were described in Chap. 3. It is conceivable that global information could be used by the Routing Service automatically to reconfigure the network based on network optimization criteria. Automatic reconfiguration of the network is yet another area worth exploring.

[28]Power is defined as the ratio of total throughput to average delay.

Bibliography

[1]

Aho, A.V., Hopcroft, J.E., and Ullman, J.D.
The design and analysis of computer algorithms.
Addison-Wesley, 1974, chapter 5.10.

[2]

Baase, S.
Computer algorithms.
Addison-Wesley, 1978, chapter 3.3.

[3]

Boggs, D.R., Shoch, J.F., and Taft, E.A.
Pup: An Internetwork Architecture.
IEEE Transactions on Communications (COM-28, 4):612-623, April, 1980.

[4]

Clark, D.D., Pogran, K.T., and Reed, D.P.
An Introduction to Local Area Networks.
In *Proceedings of IEEE*, pages 1497-1517. IEEE, November, 1978.

[5]

Even, S.
Graph Algorithms.
Computer Science Press, 1979, chapter 1.5.

[6]

Friedman, D.U.
Communication Complexity of Distributed Shortest Path Algorithms.
Master's thesis, Massachusetts Institute of Technology, Department of Electrical Engineering, Cambridge, Massachusetts, December, 1979.

[7]

Hopkins, G.T.
Multimode Communications on the MITRENET.
In *Proceedings of the Local Area Communications Network Symposium*, pages 169-177. MITRE Corporation, Bedford, Massachusetts, May, 1979.

[8]

Horowitz, E. and Sahni, S.
Fundamentals of Data Structures.
Computer Science Press, 1978, chapter 4.7.

[9]
Jaffe, J.M.
Flow Control Power is Non-decentralizable.
Technical Report RC 8343 (# 36291), IBM T.J. Watson Research Center Yorktown Heights, NY 10598, March, 1980.
Submitted for publication

[10]
Kahn, R.E., Gronemeyer, S.A., Burchfiel, J., Kunzelman, R.C.
Advances in Packet Radio Technology.
In *Proceedings of the IEEE, Vol. 66, No. 11*, pages 1468-1496. IEEE, November, 1978.

[11]
McQuillan, J.M.
Routing Algorithms for Computer Networks—A Survey.
In *Proceedings of the National Telecommunications Conference*, pages 28:1-6. , 1977.

[12]
McQuillan, J.M., Richer, I., and Rosen, E.C.
The New Routing Algorithm for the ARPANET.
IEEE Transactions on Communications (COM-28, 5), May, 1980.

[13]
Metcalfe, R.M., and Boggs, D.R.
Ethernet: Distributed Packet Switching for Local Computer Networks.
Communications of the ACM (19, 7):395-404, July, 1976.

[14]
Okuda, N., and Kunikyo, T.
Ring Century Bus: An Experimental High Speed Channel for Computer Communications.
In *Proceedings of the Fourth International Conference on Computer Communications*, pages 161-166. , September, 1978.

[15]
Pouzin, L.
Methods, Tools, and Observations on Flow Control in Packet-Switched Data Networks.
IEEE Transactions on Communications (COM-29, 4), April, 1981.

[16]
Reingold, E.M., Nievergelt, J., and Deo, N.
Combinatorial algorithms: theory and practice.
Prentice-Hall, 1977, chapter 8.2.6.

[17]
Rudin, H.
On Routing and Delta Routing: A Taxonomy and Performance Comparison of Techniques for Packet-Switched Networks.
IEEE Transactions on Communications (COM-24, 1):43-59, January, 1976.

[18]
Saltzer, J.H.
Source Routing for Campus-wide Internet Transport.
In *Proceedings of the IFIP Working group 6.4 Workshop on Local Area Networks in Zurich*. IFIP, August, 1980.
Yet to be published

[19]
Saltzer, J.H.
On Names in Networks—A Systems Perspective.
Technical Report, Massachusetts Institute of Technology, Laboratory for Computer Science, March, 1981.
Paper in preparation

[20]
Shoch, J.F.
Inter-Network Naming, Addressing, and Routing.
In *Proceedings of COMPCON*, pages 72-79. IEEE, Fall, 1978.

[21]
Sunshine, C.A.
Addressing Problems in Multi-Network Systems.
Technical Report DARPA Internet Experiment Note 178, University of Southern California, Information Sciences Institute, April, 1981.
Unpublished, available from SRI International, Menlo Park, CA

[22]
Svobodova, L., Liskov, B., and Clark, D.D.
Distributed Computer Systems: Structure and Semantics.
Technical Report TR-215, Massachusetts Institute of Technology, Laboratory for Computer Science, March, 1979.

[23]
Thornton, J.E., Christensen, G.S., and Jones, P.D.
A New Approach to Network Storage Management.
Computer Design :81-85, November, 1975.

[24]
Wilkes, M.V., and Wheeler, D.J.
The Cambridge Digital Communication Ring.
In *Proceedings of the Local Area Communications Network Symposium*,
pages 47-61. MITRE Corporation, Bedford, Massachusetts, May, 1979.