Introduction
to
Administration
of an
Internet-based
Local Network

C R

C S
Computer Science Facilities Group
C I

L S

RUTGERS
The State University of New Jersey
Center for Computers and Information Services
Laboratory for Computer Science Research

23 September 1988

This is an introduction for people who intend to set up or administer a network based on the Internet networking protocols (TCP/IP).

Table of Contents

This document is intended to help people who are planning to set up a new network based on the Internet protocols, or to administer an existing one. It assumes a basic familiarity with the TCP/IP protocols, particularly the structure of Internet addresses. A companion paper, "Introduction to the Internet Protocols", may provide a convenient introduction. This document does not attempt to replace technical documentation for your specific TCP/IP implementation. Rather, it attempts to give overall background that is not specific to any particular implementation. It is directed specifically at networks of "medium" complexity. That is, it is probably appropriate for a network involving several dozen buildings. Those planning to manage larger networks will need more preparation than you can get by reading this document.

In a number of cases, commands and output from Berkeley Unix are shown. Most computer systems have commands that are similar in function to these. It seemed more useful to give some actual examples that to limit myself to general talk, even if the actual output you see is slightly different.

1. The problem

This document will emphasize primarily "logical" network architecture. There are many documents and articles in the trade press that discuss actual network media, such as Ethernet, Token Ring, etc. What is generally not made clear in these articles is that the choice of network media is generally not all that critical for the overall design of a network. What can be done by the network is generally determined more by the network protocols supported, and the quality of the implementations. In practice, media are normally chosen based on purely pragmatic grounds: what media are supported by the particular types of computer that you have to connect, the distance you have to go, and the logistics of installing various kinds of cable. Generally this means that Ethernet is used for medium-scale systems, Ethernet or a network based on twisted-pair wiring for micro networks, and specialized high-speed networks (typically token ring) for campus-wide backbones, and for local networks involving super-computer and other very high-performance applications.

Thus this document assumes that you have chosen and installed individual networks such as Ethernet or token ring, and your vendor has helped you connect your computers to these network. You are now faced with the interrelated problems of

- configuring the software on your computers
- finding a way to connect individual Ethernets, token rings, etc., to form a single coherent network
- connecting your networks to the outside world

My primary thesis in this document is that these decisions require a bit of advance thought. In fact, most networks need an "architecture". This consists of a way of assigning addresses, a way of doing routing, and various choices about how hosts interact with the network. These decisions need to be made for the entire network, preferably when it is first being installed.

1.1 Some comments about terminology

I am going to use the term "IP" throughout this document to refer to networks designed to carry TCP/IP. IP is the network-level protocol from the Internet (TCP/IP) family of protocols. Thus it is common practice to use the term "IP" when referring to addresses, routing, and other network-layer items. In fact the distinction is not always very clear. So in practice the terms Internet, TCP/IP, and IP may appear to be almost interchangeable.

The terms "packet" and "datagram" are also almost interchangeable. Ideally, "packet" is used for the lowest-level physical unit, whereas "datagram" refers to a unit of data at the level of IP. However these are identical for most media, so people have nearly stopped making the distinction. I have tried to use the terms correctly, even though these days it may sound a bit pedantic. The term "packet" seems to be winning out in common speech. For example, gateway speeds are generally given in "packets per second." I have used the more technically accurate "datagrams per second," since it is really datagrams that are being counted.

I use the term "gateway" where some other authors use "router." "Gateway" is the original Internet term. Unfortunately, the ISO community has begun using the same word with a rather different meaning. People have started using "router" because it doesn't have this ambiguity. I am continuing to use "gateway" because, like the companion Introduction to the Internet Protocols, this document is intended to help you make sense of the Internet specifications. Those specifications use "gateway."

2. Routing and Addressing

Many of the decisions that you need to make in setting up an IP network depend upon routing, so it will be best to give a bit of background on that topic now. I will return to routing in a later section when discussing gateways and bridges. In general, IP datagrams pass through many networks as they are going between the source and destination. Here's a typical example. (Addresses used in the examples are taken from Rutgers University.)

```
              network 1                network 2     network 3
               128.6.4                  128.6.21      128.121
    ============================   ==========   ================
       |              |        |    |      |    |         |
    ___|______    ______|____   __|____|__   __|____|____   ___|________
    128.6.4.2     128.6.4.3    128.6.4.1    128.6.21.1    128.121.50.2
                               128.6.21.2   128.121.50.1
    __________    __________   __________   ____________   ____________
    computer A    computer B    gateway R     gateway S     computer C
```

This diagram shows three normal computer systems, two gateways, and three networks. The networks might be Ethernets, token rings, or any other sort. Network 2 could even be a single point to point line connecting gateways R and S.

Note that computer A can send datagrams to computer B directly, using network 1. However it can't reach computer C directly, since they aren't on the same network. There are several ways to connect separate networks. This diagram assumes that gateways are used. (In a later section, we'll look at alternatives.) In this case, datagrams going between A and C must be sent through gateway R, network 2, and gateway S. Every computer that uses TCP/IP needs appropriate information and algorithms to allow it to know when datagrams must be sent through a gateway, and to choose an appropriate gateway.

Routing is very closely tied to the choice of addresses. Note that the address of each computer begins with the number of the network that it's attached to. Thus 128.6.4.2 and 128.6.4.3 are both on network 128.6.4. Next, notice that gateways, whose job is to connect networks, have an address on each of those networks. For example, gateway R connects networks 128.6.4 and 128.6.21. Its connection to network 128.6.4 has the address 128.6.4.1. Its connection to network 128.6.21 has the address 128.6.21.2.

Because of this association between addresses and networks, routing decisions can be based strictly on the network number of the destination. Here's what the routing information for computer A might look like:

network	gateway	metric
128.6.4	none	0
128.6.21	128.6.4.1	1
128.121	128.6.4.1	2

From this table, computer A can tell that datagrams for computers on network 128.6.4 can be sent directly, and datagrams for computers on networks 128.6.21 and 128.121 need to be sent to gateway R for forwarding. The "metric" is used by some routing algorithms as a measure of how far away the destination is. In this case, the metric simply indicates how many gateways the datagram has to go through. (This is often referred to as a "hop count".)

When computer A is ready to send a datagram, it examines the destination address. It gets the network number from the beginning of the address, and then looks in the routing table. The table entry indicates whether the datagram should be sent directly to the destination or to a gateway.

Note that a gateway is simply a computer that is connected to two different networks, and is prepared to forward datagrams between them. In many cases it is most efficient to use special-purpose equipment that are designed as gateways. However it is perfectly possible to use ordinary computers, as long as they have more than one network interface, and their software is prepared to forward datagrams. Most major TCP/IP implementations (even for microcomputers) are designed to let you use your computer as a gateway. However some of this software has limitations that can cause trouble for your network.

Note that a gateway has several addresses -- one for each network that it's attached to. This is a difference between IP and some other network protocols: each **interface** from a computer to a network has an address. With some other protocols, each computer has only one address, which applies to all of its interfaces. A gateway between networks 128.6.4 and 128.6.21 will have an address that begins with 128.6.4 (for example, 128.6.4.1). This address refers to its connection to network 128.6.4. It will also have an address that begins with 128.6.21 (for example, 128.6.21.2). This refers to its connection to network 128.6.21.

The term "network" probably makes you think of things like Ethernet, which can have many machines attached. However it also applies to point to point lines. In the diagram above, networks 1 and 3 could be in different cities. Then network 2 could be a serial line, satellite link, or other long-distance point to point connection between the two locations. A point to point line is treated as a network that just happens to have only two computers on it. As with any other network, the point to point line has a network number (in this case 128.6.21). The systems connected by the line (gateways R and S) have addresses on that network (in this case 128.6.21.1 and 128.6.21.2).

It is possible to design routing software that does not require a separate network number for each point to point line. In that case, the interface between the gateway and the point to point line doesn't have an address. This can be useful if your network is so large that you are in danger of running out of network numbers. However such "anonymous interfaces" can make network management somewhat more difficult. If there is no address, network management software may have no way to refer to the interface. Thus you may not be able to get data on throughput and errors for that interface.

3. Choosing an addressing structure

The first comment to make about addresses is a warning: Before you start using an IP network, you must get one or more official network numbers. IP addresses look like this: 128.6.4.3. This address is used by one computer at Rutgers University. The first part of it, 128.6, is a network number, allocated to Rutgers by a central authority. Before you start allocating addresses to your computers, you must get an official network number. Unfortunately, some people set up networks using either a randomly-chosen number, or a generic number supplied by the vendor. While this may work in the short run, it is a very bad idea for the long run. Eventually, you will want to connect your network to some other organization's network. Even if your organization is highly secret and very concerned about security, somewhere in your organization there is going to be a research computer that ends up being connected to a nearby university. That university will probably be connected to a large-scale national network. As soon as one of your datagrams escapes your local network, the organization you are talking to is going to become very confused, because the addresses that appear in your datagrams are probably officially allocated to someone else.

The solution to this is simple: get your own network number from the beginning. It costs nothing. If you delay it, then sometime years from now you are going to be faced with the job of changing every address on a large network. Network numbers are currently assigned by the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025 (telephone: 800-235-3155). You can get a network number no matter what your network is being used for. You do not need authorization to connect to the Defense Data Network in order to get a number. The main piece of information that will be needed when you apply for a network number is the address class that you want. See below for a discussion of this.

In many ways, the most important decision you have to make in setting up a network is how you will assign IP addresses to your computers. This choice should be made with a view of how your network is likely to grow. Otherwise, you will find that you have to change addresses. When you have several hundred computers, address changes can be nearly impossible.

Addresses are critical because IP datagrams are routed on the basis of their address. For example, addresses at Rutgers University have a 2-level structure. A typical address is 128.6.4.3. 128.6 is assigned to Rutgers University. As far as the outside world is concerned, 128.6 is a single network. Other universities send any datagram whose address begins with 128.6 to the nearest Rutgers gateway. However within Rutgers, we divide up our address space into "subnets". We use the next 8 bits of address to indicate which subnet a computer belongs to. 128.6.4.3 belongs to subnet 128.6.4. Generally subnets correspond to physical networks, e.g. separate Ethernets, although we will see some exceptions later. Systems inside Rutgers, unlike those outside, contain information about the Rutgers subnet structure. So once a datagram for 128.6.4.3 arrives at Rutgers, the Rutgers network will route it to the departmental Ethernet, token ring, or whatever, that has been assigned subnet number 128.6.4.

When you start a network, there are several addressing decisions that face you:

- Do you subdivide your address space?
- If so, do you use subnets or class C addresses?
- How big an address space do you need?

3.1 Should you subdivide your address space?

It is not necessary to use subnets at all. There are network technologies that allow an entire campus or company to act as a single large logical Ethernet, so that no internal routing is necessary. If you use this technology, then you do not need to subdivide your address space. In that case, the only decision you have to make is what class of address to apply for. However we recommend using either a subnet approach or some other method of subdividing your address space in most networks:

- In section 6.2 we will argue that internal gateways are desirable for all networks beyond the very simplest.
- Even if you do not need gateways now, you may find later that you need to use them. Thus it probably makes sense to assign addresses as if each Ethernet or token ring were going to be a separate subnet. This will allow for conversion to real subnets later if it proves necessary.
- For network maintenance purposes, it is convenient to have addresses whose structure corresponds to the structure of the network. For example, when you see a stray datagram from system 128.6.4.3, it is nice to know that all addresses beginning with 128.6.4 are in a particular building.

3.2 Subnets vs. multiple network numbers

Suppose that you have been convinced that it’s a good idea to impose some structure on your addresses. The next question is what that structure should be. There are two basic approaches. One is subnets. The other is multiple network numbers.

The Internet standards specify the format of an address. For addresses beginning with 128 through 191 (the most common numbers these days), the first two octets form the network number. E.g. in 140.3.50.1, 140.3 is the network number. Network numbers are assigned to a particular organization. What you do with the next two octets is up to you. You could choose to make the next octet be a subnet number, or you could use some other scheme entirely. Gateways within your organization must be set up to know the subnetting scheme that you are using. However outside your organization, no one will know that 140.3.50 is one subnet and 140.3.51 is another. They will simply know that 140.3 is your organization. Unfortunately, this ability to add additional structure to the address via subnets was not present in the original IP specifications. Thus some older software is incapable of being told about subnets.

If enough of the software that you are using has this problem, it may be impractical for you to use subnets. Some organizations have used a different approach. It is possible for an organization to apply for several network numbers. Instead of dividing a single network number, say 140.3, into several subnets, e.g. 140.3.1 through 140.3.10, you could apply for 10 different network numbers. Thus you might be assigned the range 140.3 through 140.12. All IP software will know that these are different network numbers.

While using separate network numbers will work just fine within your organization, it has two very serious disadvantages. The first, and less serious, is that it wastes address space. There are only about 16,000 possible class B addresses. We cannot afford to waste 10 of them on your organization, unless it is very large. This objection is less serious because you would normally ask for class C addresses for this purpose, and there are about 2 million possible class C addresses.

The more serious problem with using several network numbers rather than subnets is that it overloads the routing tables in the rest of the Internet. As mentioned above, when you divide your network number into subnets, this division is known within your organization, but not outside it. Thus systems outside your organization need only one entry in their tables in order to be able to reach you. E.g. other universities have entries in their routing tables for 128.6, which is the Rutgers network number. If you use a range of network numbers instead of subnets, that division will be visible to the entire Internet. If we used 128.6 through 128.16 instead of subdividing 128.6, other universities would need entries for each of those network numbers in their routing tables. As of this writing the routing tables in many of the national networks are exceeding the size of the current routing technology. It would be considered extremely unfriendly for any organization to use more than one network number. This may not be a problem if your network is going to be completely self-contained, or if only one small piece of it will be connected to the outside world. Nevertheless, most TCP/IP experts strongly recommend that you use subnets rather than multiple networks. The only reason for considering multiple networks is to deal with software that cannot handle subnets. This was a problem a few years ago, but is currently less serious. As long as your gateways can handle subnets, you can deal with a few individual computers that cannot by using "proxy ARP" (see below).

One warning about subnets: Your subnets must all be "adjacent". That is, you can’t have a configuration where you get from subnet 128.6.4 to subnet 128.6.5 by going through some other network entirely, e.g. 128.121. For example, Rutgers has campuses in New Brunswick and Newark. It is perfectly OK for the networks in both cities to be subnets of 128.6. However in that case, the lines beween New Brunswick and Newark must also be part of 128.6. Suppose we decided to use a regional network such as JvNCnet to talk between our two campuses, instead of providing our own lines. Since JvNCnet is 128.121, the gateways and serial lines that they provide would have addresses that begin with 128.121. This violate the rules. It is not allowable to have gateways or lines that are part of 128.121 connecting two parts of 128.6. So if we wanted to use JvNCnet between our two campuses, we’d have to get different network numbers for the two campuses. (This rule is a result of limitations in routing technology. Eventually gateway software will probably be developed that can deal with configurations whose networks are not contiguous.)

3.3 How to allocate subnet or network numbers

Now that you have decided to use subnets or multiple network numbers, you have to decide how to allocate them. Normally this is fairly easy. Each physical network, e.g. Ethernet or token ring, is assigned a separate subnet or network number. However you do have some options.

In some cases it may make sense to assign several subnet numbers to a single physical network. At Rutgers we have a single Ethernet that spans three buildings, using repeaters. It is very clear to us that as computers are added to this Ethernet, it is going to have to be split into several separate Ethernets. In order to avoid having to change addresses when this is done, we have allocated three different subnet numbers to this Ethernet, one per building. (This would be handy even if we didn’t plan to split the Ethernet, just to help us keep track of where computers are.) However before doing this, make very sure that the software on all of your computers can handle a network that has three different network numbers on it. This issue is discussed in more detail in section 3.4.

You also have to choose a "subnet mask". This is used by the software on your systems to separate the subnet from the rest of the address. So far we have always assumed that the first two octets are the network number, and the next octet is the subnet number. For class B addresses, the standards specify that the first two octets are the network number. However we are free to choose the boundary between the subnet number and the rest of the address. It's very common to have a one-octet subnet number, but that's not the only possible choice. Let's look again at a class B address, e.g. 128.6.4.3. It is easy to see that if the third octet is used for a subnet number, there are 256 possible subnets and within each subnet there are 256 possible addresses. (Actually, the numbers are more like 254, since it is generally a bad idea to use 0 or 255 for subnet numbers or addresses.) Suppose you know that you will never have more than 128 computers on a given subnet, but you are afraid you might need more than 256 subnets. (For example, you might have a campus with lots of small buildings.) In that case, you could define 9 bits for the subnet number, leaving 7 bits for addresses within each subnet. This choice is expressed by a bit mask, using ones for the bits used by the network and subnet number, and 0's for the bits used for individual addresses. Our normal subnet mask choice is given as 255.255.255.0. If we chose 9 bit subnet numbers and 7 bit addresses, the subnet mask would be 255.255.255.128.

Generally it is possible to specify the subnet mask for each computer as part of configuring its IP software. The IP protocols also allow for computers to send a query asking what the subnet mask is. If your network supports broadcast queries, and there is at least one computer or gateway on the network that knows the subnet mask, it may be unnecessary to set it on the other computers. However this capability brings with it a whole new set of possible problems. One well-known TCP/IP implementation would answer with the wrong subnet mask when queried, thus leading causing every other computer on the network to be misconfigured. Thus it may be safest to set the subnet mask explicitly on each system.

3.4 Dealing with multiple "virtual" subnets on one network

Most software is written under the assumption that every computer on the local network has the same subnet number. When traffic is being sent to a machine with a different subnet number, the software will generally expect to find a gateway to handle forwarding to that subnet. Let's look at the implications. Suppose subnets 128.6.19 and 128.6.20 are on the same Ethernet. Consider the way things look from the point of view of a computer with address 128.6.19.3. It will have no problem sending to other machines with addresses 128.6.19.x. They are on the same subnet, and so our computer will know to send directly to them on the local Ethernet. However suppose it is asked to send a datagram to 128.6.20.2. Since this is a different subnet, most software will expect to find a gateway that handles forwarding between the two subnets. Of course there isn't a gateway between subnets 128.6.19 and 128.6.20, since they are on the same Ethernet. Thus you have to find a way to tell your software that 128.6.20 is actually on the same Ethernet.

Most common TCP/IP implementations can deal with more than one subnet on a network. For example, Berkeley Unix lets you use a slight modification of the command used to define gateways. Suppose that you get from subnet 128.6.19 to subnet 128.6.4 using a gateway whose address is 128.6.19.1. You would use the command

```
route add 128.6.4.0 128.6.19.1 1
```

This says that to reach subnet 128.6.4, traffic should be sent via the gateway at 128.6.19.1, and that the route only has to go through one gateway. The "1" is referred to as the "routing metric". If you use a metric of 0, you are saying that the destination subnet is on the same network, and no gateway is needed. In our example, on system 128.6.19.3, you would use

```
route add 128.6.20.0 128.6.19.1 0
```

The actual address used in place of 128.6.19.1 is irrelevant. The metric of 0 says that no gateway is actually going to be used, so the gateway address is not used. However it must be a legal address on the local network.

Note that the commands in this section are simply examples. You should look in the documentation for your particular implementation to see how to configure your routing.

3.4.1 Dealing with Multiple Subnets by Turning off Subnetting

There is another way to handle several subnets on one physical network. This method involves intentionally misconfiguring your hosts, so it is potentially dangerous if you don't watch what you are doing. However it may be easier to deal with when you have lots of subnets on one physical network. An example of this is a site that uses bridges, and uses subnets simply for administrative convenience. The trick is to configure the software on your hosts as if you were not using subnets at all. In this case your hosts will not make any distinction between the subnets, and they'll have no trouble dealing with all of them. Now the only problem is how to talk to subnets that are *not* on this multi-subnet network. However if your gateways handle proxy ARP, they will solve that problem for you. This approach is likely to be convenient when the same network is carrying many subnets, particularly if additional ones are likely to be added later. However it has two problems:

If you have any hosts with multiple interfaces, you will have to be very careful. First, only one interface should be on the multi-subnet network. For example, suppose you have a "network" that is made up of several Ethernets connected by bridges. You can't have a machine with interfaces on two of those Ethernets. However you can have a system with one interface on the multi-subnet network and another on some totally separate subnet. Second, any machine with multiple interfaces will have to know the real subnet mask, and will need to be told explicitly which subnets are on the multi-subnet network. These restrictions come about because a system with multiple interfaces has to know which interface to use in any given case.

You will have to be careful about the ICMP subnet mask facility. This is a facility that allows systems to broadcast a query asking what the subnet mask is. If most of your hosts think the network is not subnetted, but your gateways and multi-interface hosts think it is, you've got a potential for confusion. If a gateway or multi-interface host happens to send an ICMP subnet mask reply giving the real subnet mask, some of your other hosts may pick it up. The reverse is possible as well. This means that you will either have to

- disable ICMP subnet mask replies on all of the systems that know the real subnet mask. (This may be easy if only gateways know it.)
- make sure that your hosts ignore ICMP replies

According to the most recent documents, as long as you set the subnet mask explicitly, hosts are supposed to ignore the ICMP subnet mask mechanism. So you should be able to set different masks on different hosts without causing any problem, as long as you set the mask explicitly for all of them. However we have noticed that some IP implementations will change their subnet mask when they see an ICMP subnet mask reply.

3.4.2 Multiple Subnets: Implications for Broadcasting

When you have more than one subnet on the same physical network, you need to give some thought to broadcast addresses. According to the latest standards, there are two different ways for a host on subnet 128.6.20 to send a broadcast on the local network. One is to use address 128.6.20.255. The other is to use address 255.255.255.255. 128.6.20.255 says explicitly "all hosts on subnet 128.6.20". 255.255.255.255 says "all hosts on my local network". Normally these have the same effect. However they do not when there are several subnets on one physical network. If subnet 128.6.19 is on the same Ethernet, it is also going to receive messages sent to 255.255.255.255. However hosts with numbers 128.6.19.x will not listen to broadcasts to 128.6.20.255. The result is that the two different forms of broadcast address will have somewhat different meanings. This means that you will have to exercise some care in configuring software on networks such as this, to make sure that broadcasts go where you intend them to go.

3.5 Choosing an address class

When you apply for an official network number, you will be asked what class of network number you need. The possible answers are A, B, and C. This affects how large an address space you will be allocated. Class A addresses are one octet long, class B addresses are 2 octets, and class C addresses are 3 octets. This represents a tradeoff: there are a lot more class C addresses than class A addresses, but the class C addresses don't allow as many hosts. The idea was that there would be a few very large networks, a moderate number of medium-size ones, and a lot of mom-and-pop stores with small networks. Here is a table showing the distinction:

```
class  range of first octet   network   rest  possible addresses
  A       1 - 126               p      q.r.s    16777214
  B       128 - 191             p.q      r.s    65534
  C       192 - 223             p.q.r      s    254
```

For example network 10, a class A network, has addresses between 10.0.0.1 and 10.255.255.254. So it allows 254**3, or about 16 million possible addresses. (Actually, network 10 has allocated addresses where some of the octets are zero, so there are a few more addresses possible.) Network 192.12.88, a class C network has hosts between 192.12.88.1 and 192.12.88.254, i.e. 254 possible hosts.

In general, you will be expected to choose the lowest class that will provide you with enough addresses to handle your growth over the next few years. Organizations that have computers in many buildings will probably need and be able to get a class B address, assuming that they are going to use subnetting. (If you are going to use many separate network numbers, you would ask for a number of class C addresses.) Class A addresses are normally used only for large public networks and for a few very large corporate networks.

3.6 Dialup IP and Micro gateways: Dynamically assigned addresses

In most cases, each of your computers will have its own permanent IP address. However there are a few situations where it makes more sense to allocate addresses dynamically. The most common cases involve dialup IP, and gateways intended primarily for microcomputers.

3.6.1 Dialup IP

It is possible to run IP over dialup lines. The protocol for doing so is called SLIP ("serial line IP"). SLIP is useful in at least two different circumstances:

- As a low-cost alternative to permanent point to point lines, for cases where there isn't enough traffic to justify dedicated lines.
- As a way to connect individual PC's into a network when they are located in buildings that don't have Ethernets or other LAN technology.

I am going to use the term "SLIP server" to refer to a computer system that has modems attached, which other systems can connect to using SLIP. Such a system will provide a gateway into your network for PC users or for other networks that connect using SLIP.

If you have a number of individual PC's dialing up with SLIP, it is often not practical to assign each PC its own IP address. For one thing, there may just not be enough addresses. In order to keep the routing straight, the dialup systems have to get addresses on the same subnet as the SLIP server. Generally there are only 256 or so addresses available on each subnet. If you have more PC's than that, you can't give each one its own address. If you have SLIP servers on more than one subnet, this will make permanent addresses even more difficult. If a user wanted to be able to call both servers, his PC would need two addresses, one for each subnet.

In order to avoid these problems, many SLIP implementations assign addresses dynamically. When a PC first connects to the SLIP server, the server finds an unused IP address and assigns it to the PC. The simplest way to manage this is to give each SLIP server a range of IP addresses that it keeps track of and can assign.

When you use such a scheme, your SLIP software has to include some way for the server to tell the PC what address to use. If each PC has a permanent address, you have the reverse problem: when a PC connects to a server, there has to be a way for the PC to tell the server what its address is. Some care is needed. Otherwise someone could have his PC claim to be yours and steal all your files.

Unfortunately, there is no standard way to manage these addressing issues with SLIP. There are several SLIP implementations that handle them, but there isn't a single standard yet. Until such a standard is developed, you need to check out SLIP software carefully. Make sure that it assigns addresses the way you want, and that your SLIP server and your PC's agree on how to figure out the PC's address.

I recommend giving the PC's permanent addresses in cases where other computers have to be able to tell which PC they are talking to. This would be the case if the PC is going to receive private computer mail, or engage in other sensitive transactions. I recommend using dynamic addresses where you have a lot of PC's, and where the applications that they access over the network do their own security checking.

When you are using SLIP to connect two networks, you have three choices for handling addressing (although not all SLIP software can handle all three choices):

- Treat SLIP connections like point to point lines that just don't happen to be up all the time. If you call more than one computer, each pair of computers that talks has a separate network number which they use only when they talk to each other.
- Use routing software that allows anonymous interfaces. In that case no address is needed at all.
- Assign addresses dynamically when the connection is opened, just as you would for a PC that is dialing up.

If you make connections only to one or two other systems, it is quite reasonable to use a network number for each connection. This method makes it easy to keep usage and error statistics.

If you have many different connections, it is probably best to use anonymous interfaces. You would probably use dynamic address allocation only if your routing technology did not support anonymous interfaces.

3.6.2 Micro gateways

It is perfectly possible for microcomputers to participate in an IP network. However there seems to be a tendency for micros to use somewhat different network technology than larger systems. This is because many micro users start with specialized network software whose design is tailored specifically to the needs of micros, or even some particular type of micro. Micro users quite naturally want to be able to start using TCP/IP without having to abandon any special micro network that they are already using. For that reason there is a growing number of gateway products that allow PC's to access both some micro-oriented network product and TCP/IP.

In this section, Apple's AppleTalk is used as an example. This is because gateways for it have existed for some time, and are in widespread use. However similar products exist for several other micro network technologies. Note that the term AppleTalk refers to the Apple network protocols, whereas LocalTalk refers to the specific twisted-pair technology on which AppleTalk was initially implemented. Thus AppleTalk is analogous to the TCP/IP protocols, whereas LocalTalk is analogous to the Ethernet medium.

Several vendors supply gateways to connect AppleTalk running over a LocalTalk network with IP running over Ethernet. Although there are several products of this kind, most of them supply the following services:

- TCP/IP applications on the PC can connect to TCP/IP systems on the Ethernet. Special facilities are defined to allow

IP datagrams to be carried over LocalTalk between the PC and the gateway. TCP/IP applications on the PC have to be written using a special library that uses a mixture of AppleTalk and TCP/IP. The AppleTalk facilities are needed to get the datagrams to the gateway, where they are transformed into pure TCP/IP before being put out onto the Ethernet. Thus the TCP/IP systems on the Ethernet don't know they are talking to micros.

- AppleTalk applications can be written for larger systems, so that PC's can use them as servers. These applications are written using a special library that is more or less the reverse of the one just described. Again, it uses a mixture of AppleTalk and TCP/IP. But this time TCP/IP facilities are needed to get the datagrams to the gateway, where they are transformed into pure AppleTalk before being put onto the LocalTalk network to communicate with the PC's. Thus the PC's can access applications on the larger systems, without knowing that they are on the Ethernet rather than an Apple network.
- A campus or corporate IP network can be used to connect AppleTalk networks at different locations. Gateways at each location wrap up AppleTalk datagrams inside IP datagrams, and send them over the main IP network.

In addition, some newer gateways will translate at the application level. For example one gateway will translate between the Apple filing protocol and Sun's Network File System. This allows a PC to access a Unix file system, with the PC using the Apple filing protocol, and the final access to the Unix system being done using Sun's Network File System.

Unfortunately the flexibility of products like this also means that they are complex. Addressing issues are particularly complicated. For the same reasons as SLIP, these gateways often use dynamic IP address allocation. A range of IP addresses is assigned to each gateway. When a PC attempts to open its first TCP/IP connection, the gateway picks a free IP address and assigns it to the PC. As with SLIP, you will often need to choose whether you want addresses to be assigned this way, or you want each PC to have its own address. Again, this depends upon how many PC's you have and whether you have applications which must be able to use the IP address to identify the particular PC that is talking to it.

Addressing is further complicated by the fact that AppleTalk has its own addressing structure. So you must define a mapping between AppleTalk and IP network numbers. There must also be a mapping between individual IP addresses and AppleTalk addresses, but this mapping is maintained dynamically by the gateways.

4. Network-wide Services, Naming

If you are going to have a TCP/IP network, there are certain things that you are going to have to do centrally. Some of them are simply administrative. The most important is that you will a central registry of names and IP addresses. The DDN Network Information Center performs this role for the Internet network as a whole. If you are connected to the international Internet, your administrator will need to register with the DDN Network Information Center, so that queries from other institutions about your hosts are forwarded to your servers.

You will want to maintain a database containing information about each system on your network. At a minimum, you need to have the host name and IP address for each system. Probably the central registry will assign IP addresses. If your network is subnetted, or if you use multiple class C network numbers, the registry will probably assign network numbers to new networks or subnets. Most commonly, individual host administrators will be allowed to choose their own host names. However the registry must at least verify that there are no duplicate names. If you have a very large network, you may choose to delegate some of these tasks to subregistries, possibly one for each department.

We suggest that you assign numbers in the simplest way: starting from 1. Thus if your network is 128.6, you would assign 128.6.1 as your first subnet, 128.6.2 as the second, etc. IP addresses for individual hosts should probably start at 2. This allows you to reserve 1 on each subnet for use by a gateway. Thus the first host on subnet 128.6.4 would be 128.6.4.2, the next 128.6.4.3, etc. There is a specific reason for keeping addresses as small as possible. If you have a large organization, you may run out of subnet numbers. If you do, and if your host numbers are small, you can assign another bit for the subnet. For example, we use the entire third octet as a subnet number. As long as all of our host numbers are less than 128, we will be able to expand to 9-bit subnet numbers. For example, subnet 128.6.4 would be split into two separate subnets, 128.6.4.0 and 128.6.4.128. If we had assigned host numbers above 128, this split would be impossible.

Host names need not be so systematic. They can start with almost any word made up of letters numbers, and hyphens. It is safest for the first character to be a letter. It will be easier for users if the name is fairly short. (We have seen software that has trouble dealing with names longer than 16 characters.) Many times departments or projects choose a theme, and pick names that are consistent with them. For example, the machines used by computer science graduate students at Rutgers are named after rock bands: STEELEYE, BAND, TREX, DEVO, etc. Our math department uses famous mathematicians: GAUSS, FERMAT, etc. If your institution does not have any connection with the outside world, such one-word names are all you need.

If you are connected to with the international Internet, your organization will need to get a "domain name." This is assigned to you by the DDN Network Information Center, just as your network number is. Unlike the network number, you can get along without one if your network is isolated. If you find later that you need one, it is easy to add a domain name. (We recommend that you start with an official network number from the beginning because changing network numbers later can be traumatic.) Domain names normally end in .EDU for educational institutions, .COM for companies, etc. For example, Rutgers University has a domain name of .RUTGERS.EDU A full domain-style host name consists of your one-word internal name followed by your organization's domain name. For example, the computer I normally use is known internally as ATHOS. It's full name is ATHOS.RUTGERS.EDU If you have a large organization, it is possible to have sub-domains. For example, you might have a subdomain for each department. This adds another period to your names. For example, the computer science department might have decided to create a subdomain. In this case, my computer would probably be called ATHOS.CS.RUTGERS.EDU Once you get a domain name assigned to you, it is wise to change all of your configuration files so that the full form of name is used. However your software can be set up so that the one-word versions are accepted as nicknames. That way your users don't have to type out the long form.

If you have more than one or two systems, you are going to need some way to keep host information up to date on all of your systems. TCP/IP software needs to be able to translate host names into IP addresses. When a user tries to connect to another system, he wants to be able to refer to it by name. The software has to translate the name into the IP address in order to open the connection. Most software provides two ways to do this translation: a static table or a name server. The table approach is probably easier for small organizations, as long as they are not connected to any other network. You simply create a file that lists the names and addresses of all your hosts. Here's part of our host table:

```
HOST: 128.6.4.2, 128.6.25.2 : ARAMIS.RUTGERS.EDU,ARAMIS : SUN-3-280 : UNIX ::
HOST: 128.6.4.3 : GAUSS.RUTGERS.EDU,GAUSS : SUN-3-180 : UNIX ::
HOST: 128.6.4.4, 128.6.25.4 : ATHOS.RUTGERS.EDU,ATHOS : SUN-4-280 : UNIX ::
```

This format has one line for each system, and lists its addresses, names, and other information about it. Note that aramis and athos are both on two networks, so they have two addresses. They have both primary names, e.g. ARAMIS.RUTGERS.EDU, and nicknames, e.g. ARAMIS. Since we are attached to the Internet, our primary name is a full domain name. We supply brief nicknames to make it easier for our users. There is one other commonly-used format for the host table. Here's an example of that format:

```
128.6.4.2   aramis.rutgers.edu aramis
128.6.25.2  aramis.rutgers.edu aramis
128.5.4.3   gauss.rutgers.edu gauss
128.6.4.4   athos.rutgers.edu gauss
128.6.25.4  athos.rutgers.edu gauss
```

In this format, each line represents a single IP address. If a system has two interfaces, there are two lines in the table for it. You should try to put the address first that is likely to be used more often. The documentation for your systems should indicate what format they want the host information to use.

In the simplest setup, every computer has its own copy of the host table. If you choose to use the setup, you will want to set up procedures to make sure that systems get updated copies of the host table regularly.

Larger sites, and all sites that are connected to the Internet, should use name servers instead of individual host tables. A name server is a program that you run on a few of your systems to keep track of names. When a program needs to look up a name, instead of looking for a copy of the host table, it sends a network query to the name server. This approach has two advantages:

- For a large site, it is easier to keep tables up to date on a few name servers than on every system.
- If your site is connected to the Internet, your name server will be able to talk to name servers at other organizations, and look up names elsewhere.

Using a name server is the only way to have access to complete host information about the rest of the Internet.

It is important to understand the difference between a name server and a resolver. A name server is a program that accesses a host database, and answers queries from other programs. A resolver is a set of subroutines that can be loaded with your program. It generates queries to the name server, and processes the responses. Every system should use the resolver. (Actually, the resolver is generally loaded with each program that uses the network, since it's simply a set of subroutines.) You only need a few name servers. Many people confuse these two concepts, and come to believe that every computer needs to run a name server.

In order to use a resolver, each computer will need a configuration file or other option that specifies the address of a name server where queries should be sent. Generally you should specify several name servers, in case one of them is down. If your system cannot reach any name server, much of your software is likely to misbehave. Thus you should be very careful to have enough name servers around that every system can always reach at least one name server.

Name servers generally have a number of configuration options. Rather than giving advice here on setting up a name server, I am going to refer you to two official Internet standards documents. Both are available from the DDN Network Information Center, SRI International, 333 Ravenswood Avenue, Menlo Park, California 94025 (telephone: 800-235-3155). RFC 1032 contains instructions for getting a domain name from the Network Information Center, including the necessary forms. RFC 1033 contains instructions on how to set up a name server. Like this document, these documents are conceptual. You will also need documentation for the specific name server software that you are going to use. [This paragraph is a cop-out. Future editions of this document will contain some advice on setting up a name server. However RFC 1033 is almost unique in that it is directed at administrators rather than networking experts. Thus it is reasonable to direct people there for the moment.]

In some cases you may need to use both fixed tables and name servers. If you have some TCP/IP implementations that do not include resolvers, then you will have to have host tables for those systems. If your network is connected to the international Internet, you are going to have problems with systems that don't have resolvers. The Internet is too big for there to be a host table that lists all of its hosts. Thus you will have to put together a host table that lists those hosts that your users tend to use. The DDN Network Information Center maintains a host table that will be a good starting point. However it is by no means complete. So you will have to add your users' favorite hosts to it. Systems that use a resolver will not have this problem, since the name servers are able to translate any legal host name.

Host name and number allocation is the only facility that has to be done centrally. However there are other things that you may prefer to do centrally. It is very common to have one or two computers that handle all computer mail. If are on the Internet, it is easy for every one of your computers to talk directly to any other computer on the Internet. However most institutions want to communicate with systems on other networks, such as Bitnet and Usenet. There are gateways between the various networks. But choosing the right gateway, and transforming computer mail addresses correctly is a rather specialized business. Thus many sites set up the appropriate software only one place and direct all external mail (or all external mail to hosts that are not on the Internet) through this system.

5. Setting up routing for an individual computer

All TCP/IP implementations require some configuration for each host. In some cases this is done during "system generation". In other cases, various startup and configuration files must be set up on the system. Still other systems get configuration information across the network from a "server". While the details differ, the same kind of information needs to be supplied for most implementations. This includes

- parameters describing the specific machine, such as its IP address.
- parameters describing the network, such as the subnet mask (if any)
- routing software and the tables that drive it
- startup of various programs needed to handle network tasks

Before a machine is installed on your network, a coordinator should assign it a host name and IP address, as described above. Once you have name and address, you are ready to start configuring your computer. Often you have to put the address and name into a configuration file on the computer. However some computers (particularly those without permanent disks on which configuration information could be stored) get this information over the network. When such a system starts, it broadcasts a request over the network. In effect, this request says "who am I?" If you have any computers like this, you will have to make sure that some system on your network is ready to answer these questions. The obvious issue is: how can another system tell who you are? Generally this is done based on Ethernet address (or the analogous address for other types of network). Ethernet addresses are assigned by the computer manufacturer. It is guaranteed that only one machine in the entire world has any particular Ethernet address. The address is normally stored in ROM somewhere in the machine. The machine may not know its IP address, but it does know its Ethernet address. Thus the "who am I" request includes the Ethernet address. Systems that are set up to answer such requests have a table that lists Ethernet addresses and the corresponding IP address. This lets them know how to answer. Unfortunately, you have to set this table up manually. Generally you know the IP address, because your address coordinator has assigned it. The only problem in constructing the table will be finding out the Ethernet address for each computer. Generally, computers are designed so that they print the Ethernet address on the console shortly after being turned on. However in some cases you may have to find a way to bring the computer up and then type a command that displays information about the Ethernet interface.

Generally the subnet mask should be specified in a configuration file associated with the computer. (For Unix systems, the "ifconfig" command is used to specify both the Internet address and subnet mask.) However there are provisions in the IP protocols for a computer to broadcast a request asking for the subnet mask. The subnet mask is an attribute of the network. It is the same for all computers on a given subnet. Thus there is no separate subnet table corresponding to the Ethernet/Internet address mapping table used to answer address queries. Ideally, only a few authoritative computers will answer queries about the subnet mask. However many TCP/IP implementations are set up so that any machine on the network that believes it knows the subnet mask will answer. If your TCP/IP is like this, an incorrect subnet mask setting on one machine can cause confusion throughout the network.

Normally the startup files do roughly the following things:

- load any special device drivers that may be necessary. (This is particularly common with PC's, where network access is likely to depend upon add-on controller cards and software that is not part of the original operating system.)
- enable each of the network interfaces (Ethernet interface, serial lines, etc.) Normally this involves specifying an Internet address and subnet mask for each, as well as other options that will be described in your vendor's documentation.
- establish network routing information, either by commands that add fixed routes, or by starting a program that obtains them dynamically.
- turn on the domain system (used for looking up names and finding the corresponding Internet address -- see the section on the domain system in the Introduction to TCP/IP). Note that the details of this will depend upon how the domain system is configured. In most cases only a few hosts actually run domain name servers that must be started. Other hosts simply need configuration files that specify where the nearest name server is located.

- set various other information needed by the system software, such as the name of the system itself.
- start various "daemons". These are programs that provide network services to other systems on the network, and to users on this system. In the case of PC's, which often cannot run multiple processes, similar facilities may be provided by so-called "TSR"'s, or they may be built into the device drivers.

It is not practical to document these steps in detail, since they differ for each vendor. This section will concentrate on a few issues where your choice will depend upon overall decisions about how your network is to operate. These overall network policy decisions are often not as well documented by the vendors as the details of how to start specific programs. Note that some care will be necessary to integrate commands that you add for routing, etc., into the startup sequence at the right point. Some of the most mysterious problems occur when network routing is not set up before a program needs to make a network query, or when a program attempts to look up a host name before the name server has finished loading all of the names from a master name server.

5.1 How datagrams are routed

If your system consists of a single Ethernet or similar medium, you do not need to give routing much attention. However for more complex systems, each of your machines needs a routing table that lists a gateway and interface to use for every possible destination network. A simple example of this was given at the beginning of this section. However it is now necessary to describe the way routing works in a bit more detail. On most systems, the routing table looks something like the following. (This example was taken from a system running Berkeley Unix, using the command "netstat -n -r". Some columns containing statistical information have been omitted.)

```
Destination         Gateway             Flags       Interface

128.6.5.3           128.6.7.1           UHGD        il0
128.6.5.21          128.6.7.1           UHGD        il0
127.0.0.1           127.0.0.1           UH          lo0
128.6.4             128.6.4.61          U           pe0
128.6.6             128.6.7.26          U           il0
128.6.7             128.6.7.26          U           il0
128.6.2             128.6.7.1           UG          il0
10                  128.6.4.27          UG          pe0
128.121             128.6.4.27          UG          pe0
default             128.6.4.27          UG          pe0
```

The example system is connected to two Ethernets:

```
  controller  network   address     other networks
     il0      128.6.7   128.6.7.26    128.6.6
     pe0      128.6.4   128.6.4.61    none
```

The first column shows the name for the Ethernet interface. The second column is the network number for that Ethernet. The third column is this computer's Internet address on that network. The last column shows other subnets that share the same physical network.

Now let's look at the routing table. For the moment, let us ignore the first 3 lines. The majority of the table consists of a set of entries describing networks. For each network, the other three columns show where to send datagrams destined for that network. If the "G" flag is present in the third column, datagrams for that network must be sent through a gateway. The second column shows the address of the gateway to be used. If the "G" flag is not present, the computer is directly connected to the network in question. So datagrams for that network should be sent using the controller shown in the third column. The "U" flag in the third column simply indicates that the route specified by that line is up. (Generally a route is assumed to be up unless attempts to use it consistently result in errors.)

The first 3 lines show "host routes", indicated by the "H" flag in column three. Routing tables normally have entries for entire networks or subnets. For example, the entry

```
128.6.2          128.6.7.1       UG      il0
```

indicates that datagrams for any computer on network 128.6.2 (i.e. addresses 128.6.2.1 through 128.6.2.254) should be sent to gateway 128.6.7.1 for forwarding. However sometimes routes apply only to a specific computer, rather than to a whole network. In that case, a host route is used. The first column then shows a complete address, and the "H" flag is present in column 3. E.g. the entry

```
128.6.5.21       128.6.7.1       UHGD    il0
```

indicates that datagrams for the specific address 128.6.5.21 should be sent to the gateway 128.6.7.1. As with network routes, the "G" flag is used for routes that involve a gateway. The "D" flag indicates that the route was added dynamically, based on an ICMP redirect message from a gateway. (See below for details.)

The following route is special:

```
127.0.0.1        127.0.0.1       UH      lo0
```

127.0.0.1 is the address of the "loopback device". This is a dummy software module. Any datagram sent out through that "device" appears immediately as input. It can be used for testing. The loopback address can also handy for talking to applications that are on your own computer. (Why bother to use your network to talk to a program that is on the same machine you are?)

Finally, there are "default" routes, e.g.

```
default          128.6.4.27      UG      pe0
```

This route is used for datagrams that don't match any other entry. In this case, they are sent to a gateway with address 128.6.4.27.

In most systems, datagrams are routed by looking up the destination address in a table such as the one just described. If the address matches a specific host route, then that is used. Otherwise, if it matches a network route, that is used. If no other route works, the default is used. If there is no default, the user should get an error message such as "network is unreachable".

The following sections will describe several ways of setting up these routing tables. Generally, the actual operation of sending datagrams doesn't depend upon which method you use to set up the routes. When a datagram is to be sent, its destination is looked up in the table. The different routing methods are simply more and less sophisticated ways of setting up and maintaining the tables.

5.2 Fixed routes

The simplest way to set up routing is to use fixed commands. Your startup files contain commands to set up the routing table. If any changes are needed, you make them manually, using commands that add and delete entries in the routing table. (When you make such a change, don't forget to update the startup files also.) This method is practical for relatively small networks, particularly if they don't change very often.

Most computers automatically set up some routing entries for you. Unix will add an entry for the networks to which you are directly connected. For example, your startup file might contain the commands

```
ifconfig ie0 128.6.4.4 netmask 255.255.255.0
ifconfig ie1 128.6.5.35 netmask 255.255.255.0
```

These specify that there are two network interfaces, and your addresses on them. The system will automatically create routing table entries

```
128.6.4          128.6.4.4       U       ie0
128.6.5          128.6.5.35      U       ie1
```

These specify that datagrams for the local subnets, 128.6.4 and 128.6.5, should be sent out the corresponding interface.

In addition to these, your startup files would contain commands to define routes to whatever other networks you wanted to reach. For example,

```
route add 128.6.2.0 128.6.4.1  1
route add 128.6.6.0 128.6.5.35 0
```

These commands specify that in order to reach network 128.6.2, a gateway at address 128.6.4.1 should be used, and that network 128.6.6 is actually an additional network number for the physical network connected to interface 128.6.5.35. Some other software might use different commands for these cases. Unix differentiates them by the "metric", which is the number at the end of the command. The metric indicates how many gateways the datagram will have to go through to get to the destination. Routes with metrics of 1 or greater specify the address of the first gateway on the path. Routes with metrics of 0 indicate that no gateway is involved -- this is an additional network number for the local network.

Finally, you might define a default route, to be used for destinations not listed explicitly. This would normally show the address of a gateway that has enough information to handle all possible destinations.

If your network has only one gateway attached to it, then of course all you need is a single entry pointing to it as a default. In that case, you need not worry further about setting up routing on your hosts. (The gateway itself needs more attention, as we will see.) The following sections are intended to provide help for setting up networks where there are several different gateways.

5.3 Routing redirects

Most TCP/IP experts recommend leaving routing decisions to the gateways. That is, it is probably a bad idea to have large fixed routing tables on each computer. The problem is that when something on the network changes, you have to go around to many computers and update the tables. If changes happen because a line goes down, service may not be restored until someone has a chance to notice the problem and change all the routing tables.

The simplest way to keep routes up to date is to depend upon a single gateway to update your routing tables. This gateway should be set as your default. (On Unix, this would mean a command such as "route add default 128.6.4.27 1", where 128.6.4.27 is the address of the gateway.) As described above, your system will send all datagrams to the default when it doesn't have any better route. At first, this strategy does not sound very good if you have more than one gateway. After all, if all you have is a single default entry, how will you ever use the other gateways in the cases where they are better? The answer is that most gateways are able to send "redirects" when they get datagrams for which there is a better route. A redirect is a specific kind of message using the ICMP (Internet Control Message Protocol). It contains information that generally translates to "In the future, to get to address XXXXX, please use gateway YYYYY instead of me". Correct TCP/IP implementations use these redirects to add entries to their routing table. Suppose your routing table starts out as follows:

```
Destination          Gateway              Flags        Interface

127.0.0.1            127.0.0.1            UH           lo0
128.6.4              128.6.4.61           U            pe0
default              128.6.4.27           UG           pe0
```

This contains an entry for the local network, 128.6.4, and a default pointing to the gateway 128.6.4.27. Suppose there is also a gateway 128.6.4.30, which is the best way to get to network 128.6.7. How do you find it? Suppose you have datagrams to send to 128.6.7.23. The first datagram will go to the default gateway, since that's the only thing in the routing table. However the default gateway, 128.6.4.27, will notice that 128.6.4.30 would really be a better route. (How it does that is up to the gateway. However there are some fairly simple methods for a gateway to determine that you would be better off using a different one.) Thus 128.6.4.27 will send back a redirect specifying that datagrams for 128.6.7.23 should be sent via 128.6.4.30. Your TCP/IP software will add a routing entry

```
128.6.7.23           128.6.4.30           UDHG          pe0
```

Any future datagrams for 128.6.7.23 will be sent directly to the appropriate gateway.

This strategy would be a complete solution, if it weren't for three problems:

- It requires each computer to have the address of one gateway "hardwired" into its startup files, as the initial default.
- If a gateway goes down, routing table entries using it may not be removed.
- If your network uses subnets, and your TCP/IP implementation does not handle them, this strategy will not work.

How serious the first problem is depends upon your situation. For small networks, there is no problem modifying startup files whenever something changes. But some organizations can find it very painful. If network topology changes, and a gateway is removed, any systems that have that gateway as their default must be adjusted. This is particularly serious if the people who maintain the network are not the same as those maintaining the individual systems. One simple appoach is to make sure that the default address never changes. For example, you might adopt the convention that address 1 on each subnet is the default gateway for that subnet. For example, on subnet 128.6.7, the default gateway would always be 128.6.7.1. If that gateway is ever removed, some other gateway is given that address. (There must always be at least one gateway left to give it to. If there isn't, you are completely cut off anyway.)

The biggest problem with the description given so far is that it tells you how to add routes but not how to get rid of them. What happens if a gateway goes down? You want traffic to be redirected back to a gateway that is up. Unfortunately, a gateway that has crashed is not going to issue Redirects. One solution is to choose very reliable gateways. If they crash very seldom, this may not be a problem. Note that Redirects can be used to handle some kinds of network failure. If something fails in a distant part of the network, your current route may no longer be a good one. As long as the gateway to which you are talking is still up and talking to you, it can simply issue a Redirect to the gateway that is now the best one. However you still need a way to detect failure of one of the gateways that you are talking to directly.

The best approach for handling failed gateways is for your TCP/IP implementation to detect routes that have failed. TCP maintains various timers that allow the software to detect when a connection has broken. When this happens, one good approach is to mark the route down, and go back to the default gateway. A similar approach can also be used to handle failures in the default gateway. If you have marked two gateways as default, then the software should be capable of switching when connections using one of them start failing. Unfortunately, some common TCP/IP implementations do not mark routes as down and change to new ones. In particular, Berkeley 4.2 Unix does not. However Berkeley 4.3 Unix does do this, and as other vendors begin to base products on 4.3 rather than 4.2, this ability is expected to become more common.

5.4 Other ways for hosts to find routes

As long as your TCP/IP implementations handle failing connections properly, establishing one or more default routes in the configuration file is likely to be the simplest way to handle routing. However there are two other routing approaches that are worth considering for special situations:

- spying on the routing protocol
- using proxy ARP

5.4.1 Spying on Routing

Gateways generally have a special protocol that they use among themselves. Note that redirects cannot be used by gateways. Redirects are simply ways for gateways to tell "dumb" hosts to use a different gateway. The gateways themselves must have a complete picture of the network, and a way to compute the optimal route to each subnet. Generally they maintain this picture by exchanging information among themselves. There are several different routing protocols in use for this purpose. One way for a computer to keep track of gateways is for it to listen to the gateways' messages among themselves. There is software available for this purpose for most of the common routing protocols. When you run this software, your computer will maintain a complete picture of the network, just as the gateways do. The software is generally designed to maintain your computer's routing tables dynamically, so that datagrams are always sent to the proper gateway. In effect, the routing software issues the equivalent of the Unix "route add" and "route delete" commands as the network topology changes. Generally this results in a complete routing table, rather than one that depends upon default routes. (This assumes that the gateways themselves maintain a complete table. Sometimes gateways keep track of your campus network completely, but use a default route for all off-campus networks, etc.)

Running routing software on each host does in some sense "solve" the routing problem. However there are several reasons why this is not normally recommended except as a last resort. The most serious problem is that this reintroduces configuration options that must be kept up to date on each host. Any computer that wants to participate in the protocol among the gateways will need to configure its software compatibly with the gateways. Modern gateways often have configuration options that are complex compared with those of an individual host. It is undesirable to spread these to every host.

There is a somewhat more specialized problem that applies only to diskless computers. By its very nature, a diskless computer depends upon the network and file servers to load programs and to do swapping. It is dangerous for diskless computers to run any software that listens to network broadcasts. Routing software generally depends upon broadcasts. For example, each gateway on the network might broadcast its routing tables every 30 seconds. The problem with diskless nodes is that the software to listen to these broadcasts must be loaded over the network. On a busy computer, programs that are not used for a few seconds will be swapped or paged out. When they are activated again, they must be swapped or paged in. Whenever a broadcast is sent, every computer on the network needs to activate the routing software in order to process the broadcast. This means that many diskless computers will be doing swapping or paging at the same time. This is likely to cause a temporary overload of the network. Thus it is very unwise for diskless machines to run any software that requires them to listen to broadcasts.

5.4.2 Proxy ARP

Proxy ARP is an alternative technique for letting gateways make all the routing decisions. It is applicable to any broadcast network that uses ARP or a similar technique for mapping Internet addresses into network-specific addresses such as Ethernet addresses. This presentation will assume Ethernet. Other network types can be acccomodated if you replace "Ethernet address" with the appropriate network-specific address, and ARP with the protocol used for address mapping by that network type.

In many ways proxy ARP it is similar to using a default route and redirects, however it uses a different mechanism to communicate routes to the host. With redirects, a full routing table is used. At any given moment, the host knows what gateways it is routing datagrams to. With proxy ARP, you dispense with explicit routing tables, and do everything at the level of Ethernet addresses. Proxy ARP can be used for all destinations, only for destinations within your network, or in various combinations. It will be simplest to explain it as used for all addresses. To do this, you instruct the host to pretend that every computer in the world is attached directly to your local Ethernet. On Unix, this would be done using a command

```
route add default 128.6.4.2 0
```

where 128.6.4.2 is assumed to be the IP address of your host. As explained above, the metric of 0 causes everything that matches this route to be sent directly on the local Ethernet. Alternatively, some systems will allow you to get the same effect by setting a subnet mask of 0. If you do this, you may have to take precautions to make sure that it isn’t reset by an ICMP subnet mask broadcast by a system that knows the real subnet mask.

When a datagram is to be sent to a local Ethernet destination, your computer needs to know the Ethernet address of the destination. In order to find that, it uses something generally called the ARP table. This is simply a mapping from Internet address to Ethernet address. Here’s a typical ARP table. (On our system, it is displayed using the command "arp -a".)

```
FOKKER.RUTGERS.EDU (128.6.5.16) at 8:0:20:0:8:22 temporary
CROSBY.RUTGERS.EDU (128.6.5.48) at 2:60:8c:49:50:63 temporary
CAIP.RUTGERS.EDU (128.6.4.16) at 8:0:8b:0:1:6f temporary
DUDE.RUTGERS.EDU (128.6.20.16) at 2:7:1:0:eb:cd temporary
W20NS.MIT.EDU (18.70.0.160) at 2:7:1:0:eb:cd temporary
OBERON.USC.EDU (128.125.1.1) at 2:7:1:2:18:ee temporary
gatech.edu (128.61.1.1) at 2:7:1:0:eb:cd temporary
DARTAGNAN.RUTGERS.EDU (128.6.5.65) at 8:0:20:0:15:a9 temporary
```

Note that it is simply a list of IP addresses and the corresponding Ethernet address. The "temporary" indicates that the entry was added dynamically using ARP, rather than being put into the table manually.

If there is an entry for the address in the ARP table, the datagram is simply put on the Ethernet with the corresponding Ethernet address. If not, an "ARP request" is broadcast, asking for the destination host to identify itself. This request is in effect a question "will the host with Internet address 128.6.4.194 please tell me what your Ethernet address is?". When a response comes back, it is added to the ARP table, and future datagrams for that destination can be sent without delay.

This mechanism was originally designed only for use with hosts attached directly to a single Ethernet. If you need to talk to a host on a different Ethernet, it was assumed that your routing table would direct you to a gateway. The gateway would of course have one interface on your Ethernet. Your computer would then end up looking up the address of that gateway using ARP. It would generally be useless to expect ARP to work directly with a computer on a distant network. Since it isn't on the same Ethernet, there's no Ethernet address you can use to send datagrams to it. And when you send an ARP request for it, there's nobody to answer the request.

Proxy ARP is based on the concept that the gateways will act as proxies for distant hosts. Suppose you have a host on network 128.6.5, with address 128.6.5.2. (computer A in diagram below) It wants to send a datagram to host 128.6.4.194, which is on a different Ethernet (subnet 128.6.4). (computer C in diagram below) There is a gateway connecting the two subnets, with address 128.6.5.1 (gateway R):

```
          network 1                network 2
           128.6.5                  128.6.4
  ============================   ==================
     |              |        |    |       |     |
  ___|______    _____|____  __|____|__  __|____|____
 128.6.5.2    128.6.5.3    128.6.5.1   128.6.4.194
                           128.6.4.1
 __________   __________   __________  ____________
 computer A   computer B    gateway R   computer C
```

Now suppose computer A sends an ARP request for computer C. C isn't able to answer for itself. It's on a different network, and never even sees the ARP request. However gateway R can act on its behalf. In effect, your computer asks "will the host with Internet address 128.6.4.194 please tell me what your Ethernet address is?", and the gateway says "here I am, 128.6.4.194 is 2:7:1:0:eb:cd", where 2:7:1:0:eb:cd is actually the Ethernet address of *the gateway*. This bit of illusion works just fine. Your host now thinks that 128.6.4.194 is attached to the local Ethernet with address 2:7:1:0:eb:cd. Of course it isn't. But it works anyway. Whenever there's a datagram to be sent to 128.6.4.194, your host sends it to the specified Ethernet address. Since that's the address of a gateway R, the gateway gets the datagram. It then forwards it to the destination.

Note that the net effect is exactly the same as having an entry in the routing table saying to route destination 128.6.4.194 to gateway 128.6.5.1:

```
128.6.4.194          128.6.5.1          UGH          pe0
```

except that instead of having the routing done at the level of the routing table, it is done at the level of the ARP table.

Generally it's better to use the routing table. That's what it's there for. However here are some cases where proxy ARP makes sense:

- when you have a host that does not implement subnets
- when you have a host that does not respond properly to redirects
- when you do not want to have to choose a specific default gateway
- when your software is unable to recover from a failed route

The technique was first designed to handle hosts that do not support subnets. Suppose that you have a subnetted network. For example, you have chosen to break network 128.6 into subnets, so that 128.6.4 and 128.6.5 are separate. Suppose you have a computer that does not understand subnets. It will assume that all of 128.6 is a single network. Thus it will be difficult to establish routing table entries to handle the configuration above. You can't tell it about the gateway explicitly using "route add 128.6.4.0 128.6.5.1 1" Since it thinks all of 128.6 is a single network, it can't understand that you are trying to tell it where to send one subnet. It will instead interpret this command as an attempt to set up a *host* route to a host whose address is 128.6.4.0. The only thing that would work would be to establish explicit host routes for every individual host on every other subnet. You can't depend upon default gateways and redirects in this situation either. Suppose you said "route add default 128.6.5.1 1". This would establish the gateway 128.6.5.1 as a default. However the system wouldn't use it to send datagrams to other subnets. Suppose the host is 128.6.5.2, and wants to send a datagram to 128.6.4.194. Since the destination is part of 128.6, your computer considers it to be on the same network as itself, and doesn't bother to look for a gateway.

Proxy ARP solves this problem by making the world look the way the defective implementation expects it to look. Since the host thinks all other subnets are part of its own network, it will simply issue ARP requests for them. It expects to get back an Ethernet address that can be used to establish direct communications. If the gateway is practicing proxy ARP, it will respond with the gateway's Ethernet address. Thus datagrams are sent to the gateway, and everything works.

As you can see, no specific configuration is needed to use proxy ARP with a host that doesn't understand subnets. All you need is for your gateways to implement proxy ARP. In order to use it for other purposes, you must explicitly set up the routing table to cause ARP to be used. By default, TCP/IP implementations will expect to find a gateway for any destination that is on a different network. In order to make them issue ARP's, you must explicitly install a route with metric 0, as in the example "route add default 128.6.5.2 0", or you must set a subnet mask of 0.

It is obvious that proxy ARP is reasonable in situations where you have hosts that don't understand subnets. Some comments may be needed on the other situations. Generally TCP/IP implementations do handle ICMP redirects properly. Thus it is normally practical to set up a default route to some gateway, and depend upon the gateway to issue redirects for destinations that should use a different gateway. However in case you ever run into an implementation that does not obey redirects, or cannot be configured to have a default gateway, you may be able to make things work by depending upon proxy ARP. Of course this requires that you be able to configure the host to issue ARP's for all destinations. You will need to read the documentation carefully to see exactly what routing features your implementation has.

Sometimes you may choose to depend upon proxy ARP for convenience. The problem with routing tables is that you have to configure them. The simplest configuration is simply to establish a default route, but even there you have to supply some equivalent to the Unix command "route add default ...". Should you change the addresses of your gateways, you have to modify this command on all of your hosts, so that they point to the new default gateway. If you set up a default route that depends upon proxy ARP (i.e. has metric 0), you won't have to change your configuration files when gateways change. With proxy ARP, no gateway addresses are given explicitly. Any gateway can respond to the ARP request, no matter what its address.

In order to save you from having to do configuration, some TCP/IP implementations default to using ARP when they have no other route. The most flexible implementations allow you to mix strategies. That is, if you have specified a route for a particular network, or a default route, they will use that route. But if there is no route for a destination, they will treat it as local, and issue an ARP request. As long as your gateways support proxy ARP, this allows such hosts to reach any destination without any need for routing tables.

Finally, you may choose to use proxy ARP because it provides better recovery from failure. This choice is very much dependent upon your implementation. The next section will discuss the tradeoffs in more detail.

In situations where there are several gateways attached to your network, you may wonder how proxy ARP allows you to choose the best one. As described above, your computer simply sends a broadcast asking for the Ethernet address for a destination. We assumed that the gateways would be set up to respond to this broadcast. If there is more than one gateway, this requires coordination among them. Ideally, the gateways will have a complete picture of the network topology. Thus they are able to determine the best route from your host to any destination. If the gateways coordinate among themselves, it should be possible for the best gateway to respond to your ARP request. In practice, it may not always be possible for this to happen. It is fairly easy to design algorithms to prevent very bad routes. For example, consider the following situation:

```
      1              2             3
    -------  A  ----------  B ----------
```

1, 2, and 3 are networks. A and B are gateways, connecting network 2 to 1 or 3. If a host on network 2 wants to talk to a host on network 1, it is fairly easy for gateway A to decide to answer, and for gateway B to decide not to. Here's how: if gateway B accepted a datagram for network 1, it would have to forward it to gateway A for delivery. This would mean that it would take a datagram from network 2 and send it right back out on network 2. It is very easy to test for routes that involve this sort of circularity. It is much harder to deal with a situation such as the following:

```
              1
       ---------------
         A        B
         |        | 4
         |        |
       3 |        C
         |        |
         |        | 5
         D        E
       ---------------
              2
```

Suppose a computer on network 1 wants to send a datagram to one on network 2. The route via A and D is probably better, because it goes through only one intermediate network (3). It is also possible to go via B, C, and E, but that path is probably slightly slower. Now suppose the computer on network 1 sends an ARP request for a destination on 2. It is likely that A and B will both respond to that request. B is not quite as good a route as A. However it is not so bad as the case above. B won't have to send the datagram right back out onto network 1. It is unable to determine there is a better alternative route without doing a significant amount of global analysis on the network. This may not be practical in the amount of time available to process an ARP request.

5.4.3 Moving to New Routes After Failures

In principle, IP routing is capable of handling line failures and gateway crashes. There are various mechanisms to adjust routing tables and ARP tables to keep them up to date. Unfortunately, many major implementations of TCP/IP have not implemented all of these mechanisms. The net result is that you have to look carefully at the documentation for your implementation, and consider what kinds of failures are most likely. You then have to choose a strategy that will work best for your site. The basic choices for finding routes have all been listed above: spying on the gateways' routing protocol, setting up a default route and depending upon redirects, and using proxy ARP. These methods all have their own limitations in dealing with a changing network.

Spying on the gateways' routing protocol is theoretically the cleanest solution. Assuming that the gateways use good routing technology, the tables that they broadcast contain enough information to maintain optimal routes to all destinations. Should something in the network change (a line or a gateway goes down), this information will be reflected in the tables, and the routing software will be able to update the hosts' routing tables appropriately. The disadvantages are entirely practical. However in some situations the robustness of this approach may outweight the disadvantages. To summarize the discussion above, the disadvantages are:

- If the gateways are using sophisticated routing protocols, configuration may be fairly complex. Thus you will be faced with setting up and maintaining configuration files on every host.
- Some gateways use proprietary routing protocols. In this case, you may not be able to find software for your hosts that understands them.
- If your hosts are diskless, there can be very serious performance problems associated with listening to routing broadcasts.

Some gateways may be able to convert from their internal routing protocol to a simpler one for use by your hosts. This could largely bypass the first two disadvantages. Currently there is no known way to get around the third one.

The problems with default routes/redirects and with proxy ARP are similar: they both have trouble dealing with situations where their table entries no longer apply. The only real difference is that different tables are involved. Suppose a gateway goes down. If any of your current routes are using that gateway, you may be in trouble. If you are depending upon the routing table, the major mechanism for adjusting routes is the redirect. This works fine in two situations:

- where the default gateway is not the best route. The default gateway can direct you to a better gateway
- where a distant line or gateway fails. If this changes the best route, the current gateway can redirect you to the gateway that is now best

The case it does not protect you against is where the gateway that you are currently sending your datagrams to crashes. Since it is down, it is unable to redirect you to another gateway. In many cases, you are also unprotected if your default gateway goes down, since routing starts by sending to the default gateway.

The situation with proxy ARP is similar. If the gateways coordinate themselves properly, the right one will respond initially. If something elsewhere in the network changes, the gateway you are currently issuing can issue a redirect to a new gateway that is better. (It is usually possible to use redirects to override routes established by proxy ARP.) Again, the case you are not protected against is where the gateway you are currently using crashes. There is no equivalent to failure of a default gateway, since any gateway can respond to the ARP request.

So the big problem is that failure of a gateway you are using is hard to recover from. It's hard because the main mechanism for changing routes is the redirect, and a gateway that is down can't issue redirects. Ideally, this problem should be handled by your TCP/IP implementation, using timeouts. If a computer stops getting responses, it should cancel the existing route, and try to establish a new one. Where you are using a default route, this means that the TCP/IP implementation must be able to declare a route as down based on a timeout. If you have been redirected to a non-default gateway, and that route is declared down, traffic will return to the default. The default gateway can then begin handling the traffic, or redirect it to a different gateway. To handle failure of a default gateway, it should be possible to have more than one default. If one is declared down, another will be used. Together, these mechanisms should take care of any failure.

Similar mechanisms can be used by systems that depend upon proxy ARP. If a connection is timing out, the ARP table entry that it uses should be cleared. This will cause a new ARP request, which can be handled by a gateway that is still up. A simpler mechanism would simply be to time out all ARP entries after some period. Since making a new ARP request has a very low overhead, there's no problem with removing an ARP entry even if it is still good. The next time a datagram is to be sent, a new request will be made. The response is normally fast enough that users will not even notice the delay.

Unfortunately, many common implementations do not use these strategies. In Berkeley 4.2, there is no automatic way of getting rid of any kind of entry, either routing or ARP. They do not invalidate routes or ARP entries based on failures. If gateway crashes are a significant problem, there may be no choice but to run software that listens to the routing protocol. In Berkeley 4.3, routing entries are removed when TCP connections are failing. ARP entries are still not removed. This makes the default route strategy more attractive for 4.3 than proxy ARP. Having more than one default route may also allow for recovery from failure of a default gateway. Note however that 4.3 only handles timeout for connections using TCP. If a route is being used only by services based on UDP, it will not recover from gateway failure. While the "traditional" TCP/IP services use TCP, network file systems generally do not. Thus 4.3-based systems still may not always be able to recover from failure.

In general, you should examine your implementation in detail to determine what sort of error recovery strategy it uses. We hope that the discussion in this section will then help you choose the best way of dealing with routing.

There is one more strategy that some older implementations use. It is strongly discouraged, but we mention it here so you can recognize it if you see it. Some implementations detect gateway failure by taking active measure to see what gateways are up. The best version of this is based on a list of all gateways that are currently in use. (This can be determined from the routing table.) Every minute or so, an echo request datagram is sent to each such gateway. If a gateway stops responding to echo requests, it is declared down, and all routes using it revert to the default. With such an implementation, you normally supply more than one default gateway. If the current default stops responding, an alternate is chosen. In some cases, it is not even necessary to choose an explicit default gateway. The software will randomly choose any gateway that is responding. This implementation is very flexible and recovers well from failures. However a large network full of such implementations will waste a lot of bandwidth on the echo datagrams that are used to test whether gateways are up. This is the reason that this strategy is discouraged.

6. Bridges and Gateways

This section will deal in more detail with the technology used to construct larger networks. It will focus particularly on how to connect together multiple Ethernets, token rings, etc. These days most networks are hierarchical. Individual hosts attach to local-area networks such as Ethernet or token ring. Then those local networks are connected via some combination of backbone networks and point to point links. A university might have a network that looks in part like this:

```
 ______________________________
|   net 1      net 2    net 3  |       net 4          net 5
| ---------X---------X-------- |      --------       --------
|                         |    |         |                |
|  Building A             |    |         |                |
|               ----------X--------------X----------------X
|                              |  campus backbone network  :
|______________________________|                          :
                                                   serial :
                                                     line :
                                                   -------X-----
                                                       net  6
```

Nets 1, 2 and 3 are in one building. Nets 4 and 5 are in different buildings on the same campus. Net 6 is in a somewhat more distant location. The diagram above shows nets 1, 2, and 3 being connected directly, with switches that handle the connections being labelled as "X". Building A is connected to the other buildings on the same campus by a backbone network. Note that traffic from net 1 to net 5 takes the following path:

- from 1 to 2 via the direct connection between those networks
- from 2 to 3 via another direct connection
- from 3 to the backbone network
- across the backbone network from building A to the building in which net 5 is housed
- from the backbone network to net 5

Traffic for net 6 would additionally pass over a serial line. With the setup as shown, the same switch is being used to connect the backbone network to net 5 and to the serial line. Thus traffic from net 5 to net 6 would not need to go through the backbone, since there is a direct connection from net 5 to the serial line.

This section is largely about what goes in those "X"'s.

6.1 Alternative Designs

Note that there are alternatives to the sort of design shown above. One is to use point to point lines or switched lines directly to each host. Another is to use a single-level of network technology that is capable of handling both local and long-haul networking.

6.1.1 A mesh of point to point lines

Rather than connecting hosts to a local network such as Ethernet, and then interconnecting the Ethernets, it is possible to connect long-haul serial lines directly to the individual computers. If your network consists primarily of individual computers at distant locations, this might make sense. Here would be a small design of that type.

```
computer 1                computer 2             computer 3
    |                          |                      |
    |                          |                      |
    |                          |                      |
computer 4 -------------- computer 5 ----------- computer 6
```

In the design shown earlier, the task of routing datagrams around the network is handled by special-purpose switching units shown as "X"'s. If you run lines directly between pairs of hosts, your hosts will be doing this sort of routing and switching, as well as their normal computing. Unless you run lines directly between every pair of computers, some systems will end up handling traffic for others. For example, in this design, traffic from 1 to 3 will go through 4, 5 and 6. This is certainly possible, since most TCP/IP implementations are capable of forwarding datagrams. If your network is of this type, you should think of your hosts as also acting as gateways. Much of the discussion below on configuring gateways will apply to the routing software that you run on your hosts. This sort of configuration is not as common as it used to be, for two reasons:

- Most large networks have more than one computer per location. In this case it is less expensive to set up a local network at each location than to run point to point lines to each computer.
- Special-purpose switching units have become less expensive. It often makes sense to offload the routing and communications tasks to a switch rather than handling it on the hosts.

It is of course possible to have a network that mixes the two kinds of techology. In this case, locations with more equipment would be handled by a hierarchical system, with local-area networks connected by switches. Remote locations with a single computer would be handled by point to point lines going directly to those computers. In this case the routing software used on the remote computers would have to be compatible with that used by the switches, or there would need to be a gateway between the two parts of the network.

Design decisions of this type are typically made after an assessment of the level of network traffic, the complexity of the network, the quality of routing software available for the hosts, and the ability of the hosts to handle extra network traffic.

6.1.2 Circuit switching technology

Another alternative to the hierarchical LAN/backbone approach is to use circuit switches connected to each individual computer. This is really a variant of the point to point line technique, where the circuit switch allows each system to have what amounts to a direct line to every other system. This technology is not widely used within the TCP/IP community, largely because the TCP/IP protocols assume that the lowest level handles isolated datagrams. When a continuous connection is needed, higher network layers implement it using datagrams. This datagram-oriented technology does not match a circuit-oriented environment very closely. In order to use circuit switching technology, the IP software must be modified to be able to build and tear down virtual circuits as appropriate. When there is a datagram for a given destination, a virtual circuit must be opened to it. The virtual circuit would be closed when there has been no traffic to that destination for some time. The major use of this technology is for the DDN (Defense Data Network). The primary interface to the DDN is based on X.25. This network appears to the outside as a distributed X.25 network. TCP/IP software intended for use with the DDN must do precisely the virtual circuit management just described. Similar techniques could be used with other circuit-switching technologies, e.g. ATT's DataKit, although there is almost no software currently available to support this.

6.1.3 Single-level networks

In some cases new developments in wide-area networks can eliminate the need for hierarchical networks. Early hierarchical networks were set up because the only convenient network technology was Ethernet or other LAN's, and those could not span distances large enough to cover an entire campus. Thus it was necessary to use serial lines to connect LAN's in various locations. It is now possible to find network technology whose characteristics are similar to Ethernet, but where a single network can span a campus. Thus it is possible to think of using a single large network, with no hierarchical structure.

The primary limitations of a large single-level network are performance and reliability considerations. If a single network is used for the entire campus, it is very easy to overload it. Hierarchical networks can handle a larger traffic volume than single-level networks if traffic patterns have a reasonable amount of locality. That is, in many applications, traffic within an individual department tends to be greater than traffic among departments.

Let's look at a concrete example. Suppose there are 10 departments, each of which generates 1 Mbit/sec of traffic. Suppose futher than 90% of that traffic is to other systems within the department, and only 10% is to other departments. If each department has its own network, that network only needs to handle 1 Mbit/sec. The backbone network connecting the department also only needs 1 Mbit/sec capacity, since it is handling 10% of 1 Mbit from each department. In order to handle this situation with a single wide-area network, that network would have to be able to handle the simultaneous load from all 10 departments, which would be 10 Mbit/sec.

However this example was carefully constructed to be favorable to the hierarchical design. If more of the traffic in the department is going to other departments, then the backbone will need a higher bandwidth. For example, suppose that a campus has a few centralized resources, e.g. mainframes and other large systems in a computing center. If most of the network traffic is from small systems attempting to get to the central system, then the argument above does not work. In this case a hierarchy may still be useful. However it doesn't reduce the bandwidth required for the long-haul network. In the example above, if all 10 departments communicated primarily with systems at the computer center, the backbone would have to be able to carry all of their traffic, 10Mbits per second. The computer center would either attach its systems directly to the backbone, or it would have a "departmental" network with a capacity of 10Mbits per second rather than the 1Mbits per second needed by the other departments.

The second limitation on single-level networks is reliability, maintainability and security. Wide-area networks are more difficult to diagnose and maintain than local-area networks, because problems can be introduced from any building to which the network is connected. They also make traffic visible in all locations. For these reasons, it is often sensible to handle local traffic locally, and use the wide-area network only for traffic that actually must go between buildings. However if you have a situation where each location has only one or two computers, it may not make sense to set up a local network at each location, and a single-level network may make sense.

6.1.4 Mixed designs

In practice, few large networks have the luxury of adopting a theoretically pure design.

It is very unlikely that any large network will be able to avoid using a hierarchical design. Suppose we set out to use a single-level network. Even if most buildings have only one or two computers, there will be some location where there are enough that a local-area network is justified. The result is a mixture of a single-level network and a hierachical network. Most buildings have their computers connected directly to the wide-area network, as with a single-level network. However in one building there is a local-area network which uses the wide-area network as a backbone, connecting to it via a switching unit.

On the other side of the story, even network designers with a strong commitment to hierarchical networks are likely to find some parts of the network where it simply doesn't make economic sense to install a local-area network. So a host is put directly onto the backbone network, or tied directly to a serial line.

However you should think carefully before making ad hoc departures from your design philosophy in order to save a few dollars. In the long run, network maintainability is going to depend upon your ability to make sense of what is going on in the network. The more consistent your technology is, the more likely you are to be able to maintain the network.

6.2 An introduction to alternative switching technologies

This section will discuss the characteristics of various technologies used to switch datagrams between networks. In effect, we are trying to fill in some details about the black boxes assumed in previous sections. There are three basic types of switches, generally referred to as repeaters, bridges, and gateways, or alternatively as level 1, 2 and 3 switches (based on the level of the OSI model at which they operate). Note however that there are systems that combine features of more than one of these, particularly bridges and gateways.

The most important dimensions on which switches vary are isolation, performance, routing and network management facilities. These will be discussed below.

The most serious difference is between repeaters and the other two types of switch. Until recently, gateways provided very different services from bridges. However these two technologies are now coming closer together. Gateways are beginning to adopt the special-purpose hardware that has characterized bridges in the past. Bridges are beginning to adopt more sophisticated routing, isolation features, and network management, which have characterized gateways in the past. There are also systems that can function as both bridge and gateway. This means that at the moment, the crucial decision may not be to decide whether to use a bridge or a gateway, but to decide what features you want in a switch and how it fits into your overall network design.

6.2.1 Repeaters

A repeater is a piece of equipment that connects two networks that use the same technology. It receives every data packet on each network, and retransmits it onto the other network. The net result is that the two networks have exactly the same set of packets on them. For Ethernet or IEEE 802.3 networks there are actually two different kinds of repeater. (Other network technologies may not need to make this distinction.)

A simple repeater operates at a very low level indeed. Its primary purpose is to get around limitations in cable length caused by signal loss or timing dispersion. It allows you to construct somewhat larger networks than you would otherwise be able to construct. It can be thought of as simply a two-way amplifier. It passes on individual bits in the signal, without doing any processing at the packet level. It even passes on collisions. That is, if a collision is generated on one of the networks connected to it, the repeater generates a collision on the other network. There is a limit to the number of repeaters that you can use in a network. The basic Ethernet design requires that signals must be able to get from one end of the network to the other within a specified amount of time. This determines a maximum allowable length. Putting repeaters in the path does not get around this limit. (Indeed each repeater adds some delay, so in some ways a repeater makes things worse.) Thus the Ethernet configuration rules limit the number of repeaters that can be in any path.

A "buffered repeater" operates at the level of whole data packets. Rather than passing on signals a bit at a time, it receives an entire packet from one network into an internal buffer and then retransmits it onto the other network. It does not pass on collisions. Because such low-level features as collisions are not repeated, the two networks continue to be separate as far as the Ethernet specifications are concerned. Thus there are no restrictions on the number of buffered repeaters that can be used. Indeed there is no requirement that both of the networks be of the same type. However the two networks must be sufficiently similar that they have the same packet format. Generally this means that buffered repeaters can be used between two networks of the IEEE 802.x family (assuming that they have chosen the same address length and maximum packet size), or two networks of some other related family. A pair of buffered repeaters can be used to connect two networks via a serial line.

Buffered repeaters share with simple repeaters the most basic feature: they repeat every data packet that they receive from one network onto the other. Thus the two networks end up with exactly the same set of packets on them.

6.2.2 Bridges and gateways

A bridge differs from a buffered repeater primarily in the fact that it exercizes some selectivity as to what datagrams it forwards between networks. Generally the goal is to increase the capacity of the system by keeping local traffic confined to the network on which it originates. Only traffic intended for other networks goes through the bridge. So far this description would also apply to a gateway. Bridges and gateways differ in the way they determine what datagrams to forward. A bridge uses only the OSI level 2 address. In the case of Ethernet or IEEE 802.x networks, this is the 6-byte Ethernet or MAC-level address. (The term "MAC-level address" is more general. However for the sake of concreteness, examples in this section will assume that Ethernet is being used. You may generally replace the term "Ethernet address" with the equivalent MAC-level address for other similar technologies.) A bridge does not examine the datagram itself, so it does not use the IP address or its equivalent for routing decisions. In contrast, a gateway bases its decisions on the IP address, or its equivalent for other protocols.

There are several reasons why it matters which kind of address is used for decisions. The most basic is that it affects the relationship between the switch and the upper layers of the protocol. If forwarding is done at the level of the MAC-level address (bridge), the switch will be invisible to the protocols. If it is done at the IP level, the switch will be visible. Let’s give an example. Here are two networks connected by a bridge:

```
       network 1          network 2
        128.6.5            128.6.4
 ==================  ================================
   |            |      |            |             |
___|______    __|______|__   _______|___   _______|___
128.6.5.2      bridge         128.6.4.3     128.6.4.4
__________    ____________   ___________   ___________
computer A                   computer B    computer C
```

Note that the bridge does not have an IP address. As far as computers A, B, and C are concerned, there is a single Ethernet (or other network) to which they are all attached. This means that the routing tables must be set up so that computers on both networks treat both networks as local. When computer A opens a connection to computer B, it first broadcasts an ARP request asking for computer B’s Ethernet address. The bridge must pass this broadcast from network 1 to network 2. (In general, bridges must pass all broadcasts.) Once the two computers know each other’s Ethernet addresses, communications use the Ethernet address as the destination. At that point, the bridge can start exerting some selectivity. It will only pass datagrams whose Ethernet destination address is for a machine on the other network. Thus a datagram from B to A will be passed from network 2 to 1, but a datagram from B to C will be ignored.

In order to make this selection, the bridge needs to know which network each machine is on. Most modern bridges build up a table for each network to which they are connected, listing the Ethernet addresses of machines known to be on that network. They do this by watching all of the datagrams on each network. When a datagram first appears on network 1, it is reasonable to conclude that the Ethernet source address corresponds to a machine on network 1.

Note that a bridge must look at every datagram on the Ethernet, for two different reasons. First, it may use the source address to learn which machines are on which network. Second, it must look at the destination address in order to decide whether it needs to forward the datagram to the other network.

As mentioned above, generally bridges must pass broadcasts from one network to the other. Broadcasts are often used to locate a resource. The ARP request is a typical example of this. Since the bridge has no way of knowing what host is going to answer the broadcast, it must pass it on to the other network. Some bridges have user-selectable filters. With them, it is possible to block some broadcasts and allow others. You might allow ARP broadcasts (which are essential for IP to function), but confine less essential broadcasts to one network. For example, you might choose not to pass rwhod broadcasts, which some systems use to keep track of every user logged into every other system. You might decide that it is sufficient for rwhod to know about the systems on a single segment of the network.

Now let's take a look at two networks connected by a gateway

```
              network 1                   network 2
               128.6.5                     128.6.4
        ====================      ==================================
           |              |          |              |             |
        ___|_______    ____|__________|_____   ________|___   ________|___
        128.6.5.2     128.6.5.1  128.6.4.1     128.6.4.3     128.6.4.4
        __________    ____________________   ___________   ___________
        computer A           gateway            computer B    computer C
```

Note that the gateway has IP addresses assigned to each interface. The computers' routing tables are set up to forward through appropriate address. For example, computer A has a routing entry saying that it should use the gateway 128.6.5.1 to get to subnet 128.6.4.

Because the computers know about the gateway, the gateway does not need to scan all the packets on the Ethernet. The computers will send datagrams to it when appropriate. For example, suppose computer A needs to send a message to computer B. Its routing table will tell it to use gateway 128.6.5.1. It will issue an ARP request for that address. The gateway will respond to the ARP request, just as any host would. From then on, datagrams destined for B will be sent with the gateway's Ethernet address.

6.2.3 More about bridges

There are several advantages to using the MAC-level address, as a bridge does. First, every packet on an Ethernet or IEEE network has such an address. The address is in the same place for every packet, whether it is IP, DECnet, or some other protocol. Thus it is relatively fast to get the address from the packet. A gateway must decode the entire IP header, and if it is to support protocols other than IP, it must have software for each such protocol. This means that a bridge automatically supports every possible protocol, whereas a gateway requires specific provisions for each protocol it is to support.

However there are also disadvantages. The one that is intrinsic to the design of a bridge is

- A bridge must look at every packet on the network, not just those addressed to it. Thus it is possible to overload a bridge by putting it on a very busy network, even if very little traffic is actually going through the bridge.

However there is another disadvantage that is based on the way bridges are usually built. It is possible in principle to design bridges that do not have this disadvantage, but I don't know of any plans to do so. It stems from the fact that bridges do not have a complete routing table that describes the entire system of networks. They simply have a list of the Ethernet addresses that lie on each of its networks. This means

- Networks that use bridges cannot have loops in them. If there were a loop, some bridges would see traffic from the same Ethernet address coming from both directions, and would be unable to decide which table to put that address in. Note that any parallel paths to the same destination constitute a loop. This means that multiple paths cannot be used for purposes of splitting the load or providing redundancy.

There are some ways of getting around the problem of loops. Many bridges allow configurations with redundant connections, but turn off links until there are no loops left. Should a link fail, one of the disabled ones is then brought back into service. Thus redundant links can still buy you extra reliability. But they can't be used to provide extra capacity. It is also possible to build a bridge that will make use of parallel point to point lines, in the one special case where those lines go between a single pair of bridges. The bridges would treat the two lines as a single virtual line, and use them alternately in round-robin fashion.

The process of disabling redundant connections until there are no loops left is called a "spanning tree algorithm". This name comes from the fact that a tree is defined as a pattern of connections with no loops. Thus one wants to disable connections until the connections that are left form a tree that "spans" (includes) all of the networks in the system. In order to do this, all of the bridges in a network system must communicate among themselves. There is an IEEE proposal to standardize the protocol for doing this, and for constructing the spanning tree.

Note that there is a tendency for the resulting spanning tree to result in high network loads on certain parts of the system. The networks near the "top of the tree" handle all traffic between distant parts of the network. In a network that uses gateways, it would be possible to put in an extra link between parts of the network that have heavy traffic between them. However such extra links cannot be used by a set of bridges.

6.2.4 More about gateways

Gateways have their own advantages and disadvantages. In general a gateway is more complex to design and to administer than a bridge. A gateway must participate in all of the protocols that it is designed to forward. For example, an IP gateway must respond to ARP requests. The IP standards also require it to completely process the IP header, decrementing the time to live field and obeying any IP options.

Gateways are designed to handle more complex network topologies than bridges. As such, they have a different (and more complex) set of decisions to make. In general a bridge has fairly simple decision to make: should it forward a datagram, and if so which interface should it send the datagram out? When a gateway forwards a datagram, it must decide what host or gateway to send the datagram to next. If the gateway sends a datagram back onto the same network it came from, it should also issue a redirect to the source of the datagram telling it to use a better route. Many gateways can also handle parallel paths. If there are several equally good paths to a destination, the gateway will alternate among them in round-robin fashion. (This is done by some bridges also, though it is less common there. In both cases, there are some issues raised by round-robin alternation. It tends to lead to datagrams arriving in an order different than the order in which they were sent. This can complicate processing by the destination host. Some older TCP/IP implementations have bugs in handling out of order datagrams.)

In order to handle these decisions, a gateway will typically have a routing table that looks very much like a host's. As with host routing tables, a gateway's table contains an entry for each possible network number. For each network, there is either an entry saying that that network is connected directly to the gateway, or there is an entry saying that traffic for that network should be forwarded through some other gateway or gateways. We will describe the "routing protocols" used to build up this information later, in the discussion on how to configure a gateway.

6.3 Comparing the switching technologies

Repeaters, buffered repeaters, bridges, and gateways form a spectrum. Those devices near the beginning of the list are best for smaller networks. They are less expensive, and easier to set up, but less general. Those near the end of the list are suitable for building more complex networks. Many networks will contain a mixture of switch types, with repeaters being used to connect a few nearby network segments, bridges used for somewhat larger areas, and gateways used for long-distance links.

Note that this document so far has assumed that only gateways are being used. The section on setting up a host described how to set up a routing table listing the gateways to use to get to various networks. Repeaters and bridges are invisible to IP. So as far as previous sections are concerned, networks connected by them are to be considered a single network. Section 3.4 describes how to configure a host in the case where several subnets are carried on a single physical network. The same configuration should be used when several subnets are connected by repeaters or bridges.

As mentioned above, the most important dimensions on which switches vary are isolation, performance, routing, network management.

6.3.1 Isolation

Generally people use switches to connect networks to each other. So they are normally thinking of gaining connectivity, not providing isolation. However isolation is worth thinking about. If you connect two networks and provide no isolation at all, then any network problems on other networks suddenly appear on yours as well. Also, the two networks together may have enough traffic to overwhelm your network. Thus it is well to think of choosing an appropriate level of protection.

Isolation comes in two kinds: isolation against malfunctions and traffic isolation. In order to discuss isolation of malfunctions, we have to have a taxonomy of malfunctions. Here are the major classes of malfunctions, and which switches can isolate them:

- Electrical faults, e.g. a short in the cable or some sort of fault that distorts the signal. All types of switch will confine this to one side of the switch: repeater, buffered repeater, bridge, gateway. These are worth protecting against, although their frequency depends upon how often your cables are changed or disturbed. It is rare for this sort of fault to occur without some disturbance of the cable.
- Transceiver and controller problems that general signals that are valid electrically but nevertheless incorrect (e.g. a continuous, infinitely long packet, spurious collisions, never dropping carrier). All except the simple repeater will confine this: buffered repeater, bridge, gateway. (Such problems are not very common.)
- Software malfunctions that lead to excessive traffic between particular hosts (i.e. not broadcasts). Bridges and gateways will isolate these. (This type of failure is fairly rare. Most software and protocol problems generate broadcasts.)
- Software malfunctions that lead to excessive broadcast traffic. Gateways will isolate these. Generally bridges will not, because they must pass broadcasts. Bridges with user-settable filtering can protect against some broadcast malfunctions. However in general bridges must pass ARP, and most broadcast malfunctions involve ARP. This problem is not severe on single-vendor networks where software is under careful control. However sites with complex network environments or experimental network software may see problems of this sort regularly.

Traffic isolation is provided by bridges and gateways. The most basic decision is how many computers can be put onto a network without overloading its capacity. This requires knowledge of the capacity of the network, but also how the hosts will use it. For example, an Ethernet may support hundreds of systems if all the network is used for is remote logins and an occasional file transfer. However if the computers are diskless, and use the network for swapping, an Ethernet will support between 10 and 40, depending upon their speeds and I/O rates.

When you have to put more computers onto a network than it can handle, you split it into several networks and put some sort of switch between them. If you do the split correctly, most of the traffic will be between machines on the same piece. This means putting clients on the same network as their servers, putting terminal servers on the same network as the hosts that they access most commonly, etc.

Bridges and gateways generally provide similar degrees of traffic isolation. In both cases, only traffic bound for hosts on the other side of the switch is passed. However see the discussion on routing.

6.3.2 Performance

Absolute performance limits are becoming less of an issue as time goes on, since the switching technology is improving. Generally repeaters can handle the full bandwidth of the network. (By their very nature, a simple repeater must be able to do so.) Bridges and gateways often have performance limitations of various sorts. Bridges have two numbers of interest: packet scanning rate and throughput. As explained above, a bridge must look at every packet on the network, even ones that it does not forward. The number of packets per second that it can scan in this way is the packet scanning rate. Throughput applies to both bridges and gateways. This is the rate at which they can forward traffic. Generally this depends upon datagram size.

Normally the number of datagrams per second that a unit can handle will be greater for short datagrams than long ones. Early models of bridge varied from a few hundred datagrams per second to around 7000. The higher speeds are for equipment that uses special-purpose hardware to speed up the process of scanning packets. First-generation gateways varied from a few hundred datagrams per second to 1000 or more. However second-generation gateways are now available, using special-purpose hardware of the same sophistication as that used by bridges. They can handle on the order of 10000 datagrams per second. Thus at the moment high-performance bridges and gateways can switch most of the bandwidth of an Ethernet. This means that performance should no longer be a basis for choosing between types of switch. However within a given type of switch, there are still specific models with higher or lower capacity. And there may still be differences in price/performance. This is particularly true at the low end. The least expensive bridges are currently less than half the price of the least expensive gateway.

Unfortunately there is no single number on which you can base performance estimates. The figure most commonly quoted is packets per second. Be aware that most vendors count a datagram only once as it goes through a gateway, but that one prominent vendor counts datagrams twice. Thus their switching rates must be deflated by a factor of 2. Also, when comparing numbers make sure that they are for datagrams of the same size. A simple performance model is

processing time = switching time + datagram size * time per byte

That is, the time to switch a datagram is normally a constant switching time, representing interrupt latency, header processing, routing table lookup, etc., plus a component proportional to datagram size, representing the time needed to do any datagram copying. One reasonable approach to reporting performance is to give datagrams per second for minimum and maximum size datagrams. Another is to report limiting switching speed in datagrams per second and throughput in bytes per second, i.e. the two terms of the equation above.

6.3.3 Routing

Routing refers to the technology used to decide where to send a datagram next. Of course for a repeater this is not an issue, since repeaters forward every packet.

The routing strategy for a bridge turns into two decisions: (1) enabling and disabling links in order to maintain the spanning tree, and (2) deciding whether it should forward any particular packet, and out what interface (if the bridge is capable of handling more than two interfaces). The second decision is usually based on a table of MAC-level addresses. As described above, this is built up by scanning traffic visible from each interface. The goal is to forward those packets whose destination is on the other side of the bridge. This algorithm requires that the network configuration have no loops or redundant lines. Less sophisticated bridges leave this up to the system designer. With these bridges, you must set up your network so that there are no loops in it. More sophisticated bridges allow arbitrary topology, but disable links until no loops remain. This provides extra reliability. If a link fails, an alternative link will be turned on automatically. Bridges that work this way have a protocol that allows them to detect when a unit must be disabled or reenabled, so that at any instant the set of active links forms a "spanning tree". If you require the extra reliability of redundant links, make sure that the bridges you use can disable and enable themselves in this way. There is currently no official standard for the protocol used among bridges, although there is a standard in the proposal stage. If you buy bridges from more than one vendor, make sure that their spanning-tree protocols will interoperate.

Gateways generally allow arbitrary network topologies, including loops and redundant links. Because of their more general routing algorithms, gateways must maintain a model of the entire network topology. Different routing techniques maintain models of greater or lesser complexity, and use the data with varying degrees of sophistication. Gateways that handle IP should generally support the two Internet standard routing protocols: RIP (Routing Information Protocol) and EGP (External Gateway Protocol). EGP is a special-purpose protocol for use in networks where there is a backbone under a separate administration. It allows exchange of reachability information with the backbone in a controlled way. If you are a member of such a network, your gateway must support EGP. This is becoming common enough that it is probably a good idea to make sure that all gateways support EGP.

RIP is a protocol designed to handle routing within small to moderate size networks, where line speeds do not differ radically. Its primary limitations are:

- It cannot be used with networks where any path goes through more than 15 gateways. This range may be further reduced if you use an optional feature for giving a slow line a weight larger than one.
- It cannot share traffic between parallel lines (although some implementations allow this if the lines are between the same pair of gateways).
- It cannot adapt to changes in network load.
- It is not well suited to situations where there are alternative routes through lines of very different speeds.
- It may not be stable in networks where lines or gateways change a lot.

Some vendors supply proprietary modifications to RIP that improve its operation with EGP or increase the maximum path length beyond 15, but do not otherwise modify it very much. If you expect your network to involve gateways from more than one vendor, you should generally require that all of them support RIP, since this is the only routing protocol that is generally available. If you expect to use a more sophisticated protocol in addition, it may be useful for the gateways to translate between their own protocol and RIP. However for very large or complex networks, there may be no choice but to use some other protocol throughout.

More sophisticated routing protocols are possible. The primary ones being considered today are cisco System’s IGRP, and protocols based on the SPF (shortest-path first) algorithms. In general these protocols are designed for larger or more complex networks. They are in general stable under a wider variety of conditions, and they can handle arbitrary combinations of line type and speed. Some of them allow you to split traffic among parallel paths, to get better overall throughput. Some newer technologies may allow the network to adjust to take into account paths that are overloaded. However at the moment I do not know of any commercial gateway that does this. (There are very serious problems with maintaining stable routing when this is done.) There are enough variations among routing technology, and it is changing rapidly enough, that you should discuss your proposed network topology in detail with all of the vendors that you are considering. Make sure that their technology can handle your topology, and can support any special requirements that you have for sharing traffic among parallel lines, and for adjusting topology to take into account failures. In the long run, we expect one or more of these newer routing protocols to attain the status of a standard, at least on a de facto basis. However at the moment, there is no generally implemented routing technology other than RIP.

One additional routing topic to consider is policy-based routing. In general routing protocols are designed to find the shortest or fastest possible path for every datagram. In some cases, this is not desired. For reasons of security, cost accountability, etc., you may wish to limit certain paths to certain uses. Most gateways now have some ability to control the spread of routing information so as to give you some administrative control over the way routes are used. Different gateways vary in the degree of control that they support. Make sure that you discuss any requirements that you have for control with all prospective gateway vendors.

6.3.4 Network management

Network management covers a wide variety of topics. In general it includes gathering statistical data and status information about parts of your network, and taking action as necessary to deal with failures and other changes. The most primitive technique for network monitoring is periodic "pinging" of critical hosts. Pinging is a monitoring technique that depends on an "echo" datagram. This is a specific type of datagram that requests an immediate reply. Most TCP/IP implementations contain a program (usually called "ping") that sends an echo to a specified host. If you get a reply, you know that the host is up, and that the network connection to the host works. If you don’t get a reply, you know that something is wrong with one of the other. By pinging a reasonable sample of hosts, you can normally tell what is going on. If all the hosts on a network suddenly stop returning pings, it is reasonable to conclude that the connection to that network has gone bad. If one host stops returning pings, but other hosts on the same network still do, then it is reasonable to conclude that the host has crashed.

More sophisticated network monitoring requires the ability to get specific status and statistical information from various devices on the network. These should include various sorts of datagram counts, as well as counts of errors of various kinds. This data is likely to be most detailed in a gateway, since the gateway classifies datagrams using the protocols, and may even respond to certain types of datagram itself. However bridges and even buffered repeaters can certainly have counts of datagrams forwarded, interface errors, etc. It should be possible to collect this data from a central monitoring point.

There is now an official TCP/IP approach to network monitoring. The first stages use a related set of protocols, SGMP and SNMP. Both of these protocols are designed to allow you to collect information and to make changes in configuration parameters for gateways and other entities on your network. You can run the corresponding interface programs on any host in your network. SGMP is now available for several commercial gateways, as well as for Unix systems that are acting as gateways. There is a limited set of information which any SGMP implementation is required to supply, as well as a uniform mechanism for vendors to add information of their own. By late 1988, the second generation of this protocol, SNMP, should be in service. This is a slightly more sophisticated protocol. It has with it a more complete set of information that can be monitored, called the MIB (Management Information Base). Unlike the somewhat ad hoc collection of SGMP variables, the MIB is the result of numerous committee deliberations involving a number of vendors and users. Eventually it is expected that there will be a TCP/IP equivalent of CMIS, the ISO network monitoring service. However CMIS, and its protocols, CMIP, are not yet official ISO standards, so they are still in the experimental stages.

In general terms all of these protocols accomplish the same thing: They allow you to collect critical information in a uniform way from all vendors’ equipment. You send commands as UDP datagrams from a network management program running on some host in your network. Generally the interaction is fairly simple, with a single pair of datagrams exchanged: a command and a response. At the moment security is fairly simple. It is possible to require what amounts to a password in the command. (In SGMP it is referred to as a "session name", rather than a password.) More elaborate, encryption-based security is being developed.

You will probably want to configure the network management tools at your disposal to do several different things. For short-term network monitoring, you will want to keep track of switches crashing or being taken down for maintenance, and of failure of communications lines and other hardware. It is possible to configurate SGMP and SNMP to issue "traps" (unsolicited messages) to a specified host or list of hosts when some of these critical events occur (e.g. lines up and down). However it is unrealistic to expect a switch to notify you when it crashes. It is also possible for trap messages to be lost due to network failure or overload. Thus you can’t depend completely on traps. You should also poll your switches regularly to gather information. Various displays are available, including a map of your network where items change color as their status changes, and running "strip charts" that show datagram rates and other items through selected switches. This software is still in its early stages, so you should expect to see a lot of change here. However at the very least you should expect to be notified in some way of failures. You may also want to be able to take actions to reconfigure the system in response to failures, although security issues make some managers nervous about doing that through the existing management protocols.

The second type of monitoring you are likely to want to do is to collect information for use in periodic reports on network utilization and performance. For this, you need to sample each switch perodically, and retrieve numbers of interest. At Rutgers we sample hourly, and get the number of datagrams forwarded for IP and DECnet, a count of reloads, and various error counts. These are reported daily in some detail. Monthly summaries are produced giving traffic through each gateway, and a few key error rates chosen to indicate a gateway that is being overloaded (datagrams dropped in input and output).

It should be possible to use monitoring techniques of this kind with most types of switch. At the moment, simple repeaters do not report any statistics. Since they do not generally have processors in them, doing so would cause a major increase in their cost. However it should be possible to put network management software in buffered repeaters, bridges, and gateways. Gateways are the most likely to contain sophisticated network management software. Most gateway vendors that handle IP are expected to implement the monitoring protocols described above. Many bridge vendors make some provisions for collecting performance data. Since bridges are not protocol-specific, most of them do not have the software necessary to implement TCP/IP-based network management protocols. In some cases, monitoring can be done only by typing commands to a directly-attached console. (We have seen one case where it is necessary to take the bridge out of service to gather this data.) In other cases, it is possible to gather data via the network, but the monitoring protocol is ad hoc or even proprietary.

Except for very small networks, you should probably insist that any switch more complex than a simple repeater should collect statistics and provide some way of querying them remotely. Portions of the network that do not support such operations can be monitored by pinging. However ping simply detects gross failures. It does not allow you to look at the noise level of a serial line and other quantities needed to do high-quality maintenance. In the long run, you can expect the most software to be available for standard protocols such as SGMP/SNMP and CMIS. However proprietary monitoring tools may be sufficient as long as they work with all of the equipment that you have.

6.3.5 A final evaluation

Here is a summary of the places where each kind of switch technology is normally used:

- Repeaters are normally confined to a single building. Since they provide no traffic isolation, you must make sure that the entire set of networks connected by repeaters can carry the traffic from all of the computers on it. Since they generally provide no network monitoring tools, you will not want to use repeaters for a link that is likely to fail.
- Bridges and gateways should be placed sufficiently frequently to break your network into pieces for which the traffic volume is manageable. You may want to place bridges or gateways even in places where traffic level alone would not require them for network monitoring reasons.
- Because bridges must pass broadcast datagrams, there is a limit to the size network you can construct using them. It is probably a good idea to limit the network connected by bridges to a hundred systems or so. This number can be increased somewhat for bridges with good facilities for filtering.
- Because certain kinds of network misbehavior will be passed, bridges should be used only among portions of the network where a single group is responsible for diagnosing problems. You have to be crazy to use a bridge between networks owned by different organizations. Portions of your network where experiments are being done in network technology should always be isolated from the rest of the network by gateways.
- For many applications it is more important to choose a product with the right combination of performance, network management tools, and other features than to make the decision between bridges and gateways.

7. Configuring Gateways

This section deals with configuration issues that are specific to gateways. Gateways that handle IP are themselves Internet hosts. Thus the discussions above on configuring addresses and routing information apply to gateways as well as to hosts. The exact way you configure a gateway will depend upon the vendor. In some cases, you edit files stored on a disk in the gateway itself. However for reliability reasons most gateways do not have disks of their own. For them, configuration information is stored in non-volatile memory or in configuration files that are uploaded from one or more hosts on the network.

At a minimum, configuration involves specifying the IP address and address mask for each interface, and enabling an appropriate routing protocol. However generally a few other options are desirable. There are often parameters in addition to the IP address that you should set for each interface.

One important parameter is the broadcast address. As explained above, older software may react badly when broadcasts are sent using the new standard broadcast address. For this reason, some vendors allow you to choose a broadcast address to be used on each interface. It should be set using your knowledge of what computers are on each of the networks. In general if the computers follow current standards, a broadcast address of 255.255.255.255 should be used. However older implementations may behave better with other addresses, particularly the address that uses zeros for the host number. (For the network 128.6 this would be 128.6.0.0. For compatibility with software that does not implement subnets, you would use 128.6.0.0 as the broadcast address even for a subnet such as 128.6.4.) You should watch your network with a network monitor and see the results of several different broadcast address choices. If you make a bad choice, every time the gateway sends a routing update broadcast, many machines on your network will respond with ARP's or ICMP errors. Note that when you change the broadcast address in the gateway, you may need to change it on the individual computers as well. Generally the idea is to change the address on the systems that you can configure to give behavior that is compatible with systems that you can't configure.

Other interface parameters may be necessary to deal with peculiarities of the network it is connected to. For example, many gateways test Ethernet interfaces to make sure that the cable is connected and the transceiver is working correctly. Some of these tests will not work properly with the older Ethernet version 1 transceivers. If you are using such a transceiver, you would have to disable this keepalive testing. Similarly, gateways connected by a serial line normally do regular testing to make sure that the line is still working. There can be situations where this needs to be disabled.

Often you will have to enable features of the software that you want to use. For example, it is often necessary to turn on the network management protocol explicitly, and to give it the name or address of a host that is running software to accept traps (error messages).

Most gateways have options that relate to security. At a minimum, this may include setting password for making changes remotely (and the "session name" for SGMP). If you need to control access to certain parts of your network, you will also need to define access control lists or whatever other mechanism your gateway uses.

Gateways that load configuration information over the network present special issues. When such a gateway boots, it sends broadcast requests of various kinds, attempting to find its Internet address and then to load configuration information. Thus it is necessary to make sure that there is some computer that is prepared to respond to these requests. In some cases, this is a dedicated micro running special software. In other cases, generic software is available that can run on a variety of machines. You should consult your vendor to make sure that this can be arranged. For reliability reasons, you should make sure that there is more than one host with the information and programs that your gateways need. In some cases you will have to maintain several different files. For example, the gateways used at Rutgers use a program called "bootp" to supply their Internet address, and they then load the code and configuration information using TFTP. This means that we have to maintain a file for bootp that contains Ethernet and Internet addresses for each gateway, and a set of files containing other configuration information for each gateway. If your network is large, it is worth taking some trouble to make sure that this information remains consistent. We keep master copies of all of the configuration information on a single computer, and distribute it to other systems when it changes, using the Unix utilities make and rdist. If your gateway has an option to store configuration information in non-volatile memory, you will eliminate some of these logistical headaches. However this presents its own problems. The contents of non-volatile memory should be backed up in some central location. It will also be harder for network management personnel to review configuration information if it is distributed among the gateways.

Starting a gateway is particularly challenging if it loads configuration information from a distant portion of the network. Gateways that expect to take configuration information from the network generally issue broadcast requests on all of the networks to which they are connected. If there is a computer on one of those networks that is prepared to respond to the request, things are straightforward. However some gateways may be in remote locations where there are no nearby computer systems that can support the necessary protocols. In this case, it is necessary to arrange for the requests to be routed back to the network where there are appropriate computers. This requires what is strictly speaking a violation of the basic design philosophy for gateways. Generally a gateway should not allow broadcasts from one network to pass through to an adjacent network. In order to allow a gateway to get information from a computer on a different network, at least one of the gateways in between will have to be configured to pass the particular class of broadcasts used to retrieve this information. If you have this sort of configuration, you should test the loading process regularly. It is not unusual to find that gateways do not come up after a power failure because someone changed the configuration of another gateway and made it impossible to load some necessary information.

7.1 Configuring routing for gateways

The final topic to be considered is configuring routing. This is more complex for a gateway than for a normal host. Most TCP/IP experts recommend that routing be left to the gateways. Thus hosts may simply have a default route that points to the nearest gateway. Of course the gateways themselves can't get by with this. They need to have complete routing tables.

In order to understand how to configure a gateway, we have to look in a bit more detail at how gateways communicate routes. When you first turn on a gateway, the only networks it knows about are the ones that are directly connected to it. (They are specified by the configuration information.) In order to find out how to get to more distant parts of the network, it engages in some sort of "routing protocol". A routing protocol is simply a protocol that allows each gateway to advertise which networks it can get to, and to spread that information from one gateway to the next. Eventually every gateway should know how to get to every network. There are different styles of routing protocol. In one common type, gateways talk only to nearby gateways. In another type, every gateway builds up a database describing every other gateway in the system. However all of the protocols have some way for each gateway in the system to find out how to get to every destination.

A metric is some number or set of numbers that can be used to compare routes. The routing table is constructed by gathering information from other gateways. If two other gateways claim to be able to get to the same destination, there must be some way of deciding which one to use. The metric is used to make that decision. Metrics all indicate in some general sense the "cost" of a route. This may be a cost in dollars of sending datagrams over that route, the delay in milliseconds, or some other measure. The simplest metric is just a count of the number of gateways along the path. This is referred to as a "hop count". Generally this metric information is set in the gateway configuration files, or is derived from information appearing there.

At a minimum, routing configuration is likely to consist of a command to enable the routing protocol that you want to use. Most vendors will have a prefered routing protocol. Unless you have some reason to choose another, you should use that. The normal reason for choosing another protocol is for compatibility with other kinds of gateway. For example, your network may be connected to a national backbone network that requires you to use EGP (exterior gateway protocol) to communicate routes with it. EGP is only appropriate for that specific case. You should not use EGP within your own network, but you may need to use it in addition to your regular routing protocol to communicate with a national network. If your own network has several different types of gateway, then you may need to pick a routing protocol that all of them support. At the moment, this is likely to be RIP (Routing Information Protocol). Depending upon the complexity of your network, you could use RIP throughout it, or use a more sophisticated protocol among the gateways that support it, and use RIP only at the boundary between gateways from different vendors.

Assuming that you have chosen a routing protocol and turned it on, there are some additional decisions that you may need to make. One of the more basic configuration options has to do with supplying metric information. As indicated above, metrics are numbers which are used to decide which route is the best. Unsophisticated routing protocols, e.g. RIP, normally just count hops. So a route that passes through 2 gateways would be considered better than one that passes through 3. Of course if the latter route used 1.5Mbps lines and the former 9600 bps lines, this would be the wrong decision. Thus most routing protocols allow you to set parameters to take this sort of thing into account. With RIP, you would arrange to treat the 9600 bps line as if it were several hops. You would increase the effective hop count until the better route was chosen. More sophisticated protocols may take the bit rate of the line into account automatically. However you should be on the lookout for configuration parameters that need to be set. Generally these parameters will be associated with the particular interface. For example, with RIP you would have to set a metric value for the interface connected to the 9600 bps line. With protocols that are based on bit rate, you might need to specify the speed of each line (if the gateway cannot figure it out automatically).

Most routing protocols are designed to let each gateway learn the topology of the entire network, and to choose the best possible route for each datagram. In some cases you may not want to use the "best" route. You may want traffic to stay out of a certain portion of the network for security or cost reasons. One way to institute such controls is by specifying routing options. These options are likely to be different for different vendors. But the basic strategy is that if the rest of the network doesn't know about a route, it won't be used. So controls normally take the form of limiting the spread of information about routes whose use you want to control.

Note that there are ways for the user to override the routing decisions made by your gateways. If you really need to control access to a certain network, you will have to do two separate things:

- Use routing controls to make sure that the gateways use only the routes you want them to.
- Use access control lists on the gateways that are adjacent to the sensitive networks.

These two mechanisms act at different levels. The routing controls affect what happens to most datagrams: those where the user has not specified routing manually. Your routing mechanism must be set up to choose an acceptable route for them. The access control list provides an additional limitation which prevents users from supplying their own routing and bypassing your controls.

For reliability and security reasons, there may also be controls to allow you to list the gateways from which you will accept information. It may also be possible to rank gateways by priority. For example, you might decide to listen to routes from within your own organization before routes from other organizations or other parts of the organization. This would have the effect of having traffic use internal routes in preference to external ones, even if the external ones appear to be better.

If you use several different routing protocols, you will probably have some decisions to make regarding how much information to pass among them. Since multiple routing protocols are often associated with multiple organizations, you must be sure to make these decisions in consultation with management of all of the relevant networks. Decisions that you make may have consequences for the other network which are not immediately obvious. You might think it would be best to configure the gateway so that everything it knows is passed on by all routing protocols. However here are some reasons why you may not want to do so:

- The metrics used by different routing protocols may not be comparable. If you are connected to two different external networks, you want to specify that one should always be used in preference to the other, or that the nearest one should be used, rather than attempting to compare metric information received from the two networks to see which has the better route.
- EGP is particularly sensitive, because the EGP protocol cannot handle loops. Thus there are strict rules governing what information may be communicated to a backbone that uses EGP. In situations where EGP is being used, management of the backbone network should help you configure your routing.
- If you have slow lines in your network (9600 bps or slower), you may prefer not to send a complete routing table throughout the network. If you are connected to an external network, you may prefer to treat it as a default route, rather than to inject all of its routing information into your routing protocol.