CHAPTER 3

Design Types

A large-scale network design is composed of several common building blocks. Every LAN, of whatever size, has to have an Access system by which the end stations connect to the network. There are several inexpensive options for LAN connections, such as Ethernet and Token Ring. As a philosophical principle, the network should be built using basic commonly available technology. The design shouldn't have to reinvent any wheels just to allow the machines to talk to one another.

So, just as basic commonly available technologies exist for connecting end stations to LANs, there are common methods for interconnecting LAN segments. Once again, these technologies and methods should involve the most inexpensive yet reliable methods. But in this stage of interconnecting, aggregating, and distributing traffic between these various LAN segments, the designer runs into some serious hidden problems.

There may be thousands of ways to connect things, but most of these methods result in some kind of reliability problems. This book intends to establish general methodologies for designing networks so that designers can avoid these sorts of problems.

Basic Topologies

There are four basic topologies used to interconnect devices: bus, ring, star, and mesh. In a large-scale LAN design, the ultimate goal includes a number of these segments. Figures 3-1 to 3-4 show these four basic topologies.

Basic Concepts

Before getting into the solutions, I want to spend a little bit of time making sure that the potential problems are clear. What are the real goals of the network design? What are the options? Ultimately, I want to help point you toward general approaches that can save a lot of worry down the road.

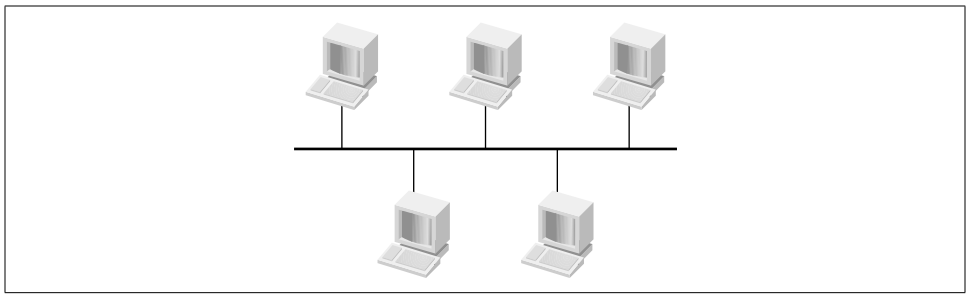

Figure 3-1. Bus topology

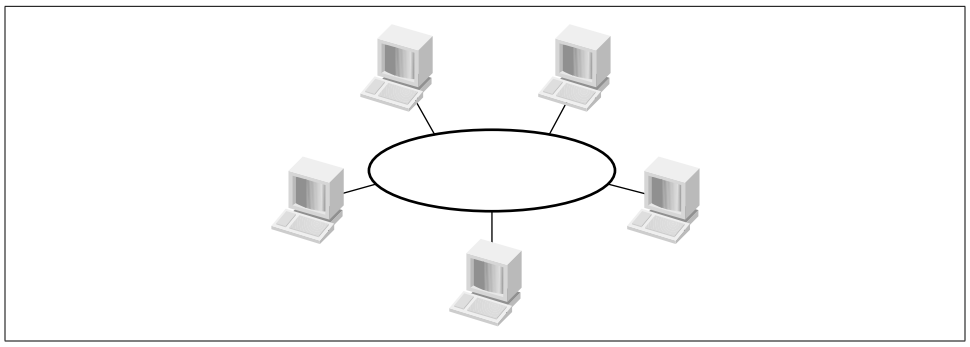

Figure 3-2. Ring topology

The main goal is to build an infrastructure that allows end devices to communicate with one another. That sounds simple enough. But what is an end device? I don't include network devices when I talk about end devices. This fact sounds pedantic, but it's important. A network device is one that cares about the lower layers of the protocol stack. It exists to facilitate the flow of traffic between end devices. End devices are the devices that care about Layer 7. End devices run applications, request data from one another, present information to humans, or control machinery; most importantly, end devices should never perform network functions.

Why do I make this point? I believe that a number of common practices on networks are dangerous or at least misguided, and they should be stopped. Here are some examples of cases in which end devices are permitted to perform network functions (such as bridging or routing) at the expense of network stability:

- File servers with two LAN NIC cards, configured to bridge between the two interfaces
- Application servers with one or more WAN cards in them that allow bridging or routing
- Servers with any number of NIC cards taking part in dynamic routing protocols such as RIP or OSPF

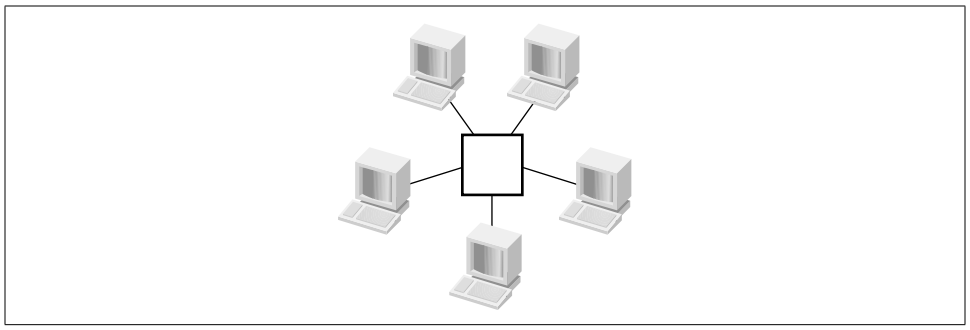

Figure 3-3. Star topology

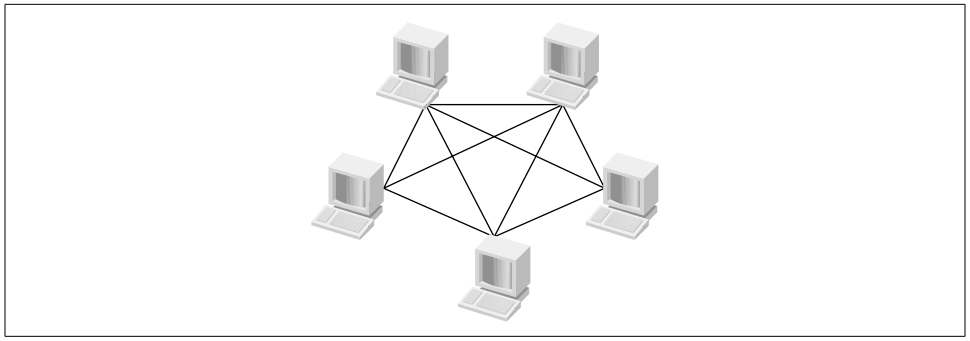

Figure 3-4. Mesh topology

In each of these cases (which I am quite sure will get me in trouble with certain vendors), an end device is permitted to perform network functions. No file or application server should ever act as a router or a bridge. If you want a router or a bridge, buy a real one and put it in. Note that I am not talking about devices that just happen to have a PC form factor or use a standard PC CPU. For example, a dedicated firewall device with a specialized secure operating system is a network device. As long as you refrain from using it as a workstation or a server, you're fine. But in no case should a file or application server act as a bridge or a router.

The concern is that any device that is not dedicated to performing network functions should not be permitted. Furthermore, with the exception of highly specialized security devices such as firewalls and similar gateways, using any general-purpose computing device in a network function is a bad idea. So, even if you only use the Linux PC as a router, and that's the only thing it does, it is still going to be less reliable and probably more expensive than using a device designed from the outset as a router. I don't like home-rolled network equipment. Real routers and switches are not very expensive, and trying to build your own is not going to save you any money in the long run, no matter how good of a programmer you are. At some point in the distant future, somebody else will inevitably have to deal with it and will fail to understand the unique system.

The same thing is true in reverse. Network devices should not perform Layer 7 functions. No router should run an email server. No firewall should be a web or email server. Sometimes you will run applications on your network devices, but they are never Layer 7 functions. For example, running a DHCP server from a router might be expedient. Or, having a web server on a router is often worthwhile if it is used only for the purposes of managing the router itself in performing its network functions. Having a Network Time Protocol (NTP) server running on your network equipment, with all other devices synchronizing their clocks to "the network" is also useful. But these are all very specific exceptions, and none of them are really user applications.

Failing to separate network functions from application functions creates so many problems that it is hard to list them. Here are a few of the most compelling:

- Generally, network engineers are not properly trained to deal with application issues. In most organizations, there are staff members who are better equipped to manage applications and servers. These people can't do their jobs properly if the network staff controls the resources. For example, if the corporate web site is housed inside of the corporate firewall, how effectively will the web mistress work with it? What if a bug is in the web server? Upgrading code could mean taking the whole Internet connection offline.

  The same situation is true of devices that include email functions such as POP servers with network functions. Such devices, if also central components of the network, make maintenance on the email server extremely difficult.
- Running applications is hard work. Running network functions is also hard work. Doing both at the same time often creates serious memory and CPU resource problems. These problems tend to occur during the most busy peak periods of the day, thereby breaking not just the application, but the entire network when it is most needed.
- I've already indicated that the network must be more reliable than any end device. If the network is an end device, then it presents an inherent reliability problem.
- If an end device takes part in a dynamic routing protocol such as RIP or OSPF, and it is either misconfigured or suffers a software bug, then that one end device can disrupt traffic for the entire network. This is why no end device should ever be permitted to take part in these protocols. There are much more reliable ways of achieving redundancy, which I will discuss throughout this book.
- Finally, it is common for file servers with multiple NICs to be configured for bridging. Having multiple NICs can be very useful—it might allow the server to exist simultaneously on several segments, or it might allow the server to handle significantly more traffic. But if these NICs are also permitted to bridge or route traffic between them, they can easily create network loops that disrupt traffic

flows. These bridging and routing functions should always be disabled on servers. Consult your server vendor for information on how to ensure that these functions are disabled.

With respect to running dynamic routing protocols on an end device, a device might passively listen to a routing protocol (particularly RIP) but not send out routing information. This situation is certainly less dangerous than allowing the end device to affect network routing tables, but it is still not a good idea; in a well-designed network, no end device should ever need to care how the network routes its packets. It should simply forward them to a default gateway and forget about them. Part of the problem here is that RIP in particular can take a long time to update after a failure. In general, allowing the network to take full responsibility for traffic flow is more reliable.

Bus topology

In a bus topology, there is a single communication medium, which I often call "the wire." It actually doesn't need to be a physical piece of wire, but a wire is a useful image. In fact, 10Base2 Ethernet looks exactly like Figure 3-1, with a long 50 Ω (50 ohm characteristic impedance) coaxial cable connecting all of the devices. Because of the analogy with 10Base2, it is customary to draw an Ethernet segment like this, with a straight line intersected at various points by the connections (sometimes called "taps") to the various devices. In the drawing, this line (the wire, or bus) extends beyond the last device at each end to symbolize the fact that the bus must be terminated electrically at both ends.

On a bus, any device can communicate directly with any other device and all devices see these messages. This is called a "unicast."* Similarly, any device can send a single signal intended for all other devices on the wire. This is a "broadcast."

If every device sees every signal sent by all other devices, then it's pretty clear that there's nothing fancy about a broadcast. To get point-to-point unicast communication going, however, there has to be some sort of address that identifies each device uniquely. This is called the MAC address.

There also has to be some sort of mechanism to ensure that all devices don't try to transmit at the same time. In Ethernet the collision detection algorithm (CSMA/CD), which I will talk about more in Chapter 4, prevents such a problem. The other network standard that employs this basic topology is called "token bus," which works by passing a virtual "token" among the devices. Only the device that holds the token is allowed to transmit. The term "token bus" is not used much anymore, so I will not cover it in detail in this book.

* This odd word, "unicast," comes from the word "broadcast." A broadcast is sent to everybody, a "mulitcast" is sent to several recipients, and a "unicast" is sent to just one recipient.

There are a few common failure modes in a bus topology. It is possible to have cable break in the middle, thereby isolating the two sides from each other. If one side holds the router that allows devices on the segment to get off, then the devices on the other side are effectively stranded. More serious problems can result if routers are on both sides of the break.

The other problem that often develops in bus architectures is loss of one of the bus termination devices. In the case of 10Base2, this termination was a small electrical resister that cancelled echoes from the open end of the wire. If this terminator was damaged or removed, then every signal sent down the wire was met by a reflected signal. The result was noise and a seriously degraded performance.

Both of these problems are avoided partially by using a central concentrator device such as a hub or a switch. In fact, new Ethernet segments are usually deployed by using such a device.

Ring topology

The second basic segment architecture is a *simple ring*. The most common example of the simple ring architecture is Token Ring. SONET and FDDI are based on double ring architectures.

In Token Ring, each device has an upstream and a downstream neighbor. If one device wants to send a packet to another device on the same ring, it sends that packet to its downstream neighbor, who forwards it to its downstream neighbor, and so on until it reaches the destination. Chapter 4 describes the Token Ring protocol in more detail.

Token Ring relies on the fact that it is a ring. If a device sends a frame on the network, it expects to see that frame coming around again. If it was received correctly, then this is noted in the frame. Thus, the ring topology allows a simple verification that the information has reached its destination.

The closed ring also facilitates token passing and ensures that the network is used efficiently. Thus, a broken ring is a serious problem, although not as serious as a broken bus, since the Token Ring protocol has a detailed set of procedures for dealing with physical problems such as this.

It might look like each device taking part in the Token Ring acts as a bridge, forwarding each frame from its upstream neighbor to the downstream neighbor. But this is not really accurate, since the network interface cards in each device passes the Layer 2 frames along, regardless of their content. Even if the frame is intended for the local device, it still must pass along a copy, although it will change a bit in the header to indicate that it has been received.

FDDI uses another ring architecture that gets around this broken ring problem in a rather clever way. In FDDI, two rings run at all times. The tokens on these two rings

travel in opposite directions, so the upstream neighbor on one ring is the downstream neighbor on the other. However, in normal operation, only one of these rings is used. The second ring acts as a backup in case of a failure, such as a broken ring.

Figure 3-5 shows what happens when the rings break. If the connection between devices A and B breaks, then the devices know about it immediately because there is two-way communication between them, and they have now lost contact with one another. They respond by closing the ring. Now when device A receives a token from device E on the clockwise-rotating ring, instead of sending it on to B, it turns around and sends it back to E on the counterclockwise-rotating ring. The token doesn't get lost because the rings have healed around the fault. The same thing happens if one of the devices taking part in the FDDI ring disappears.

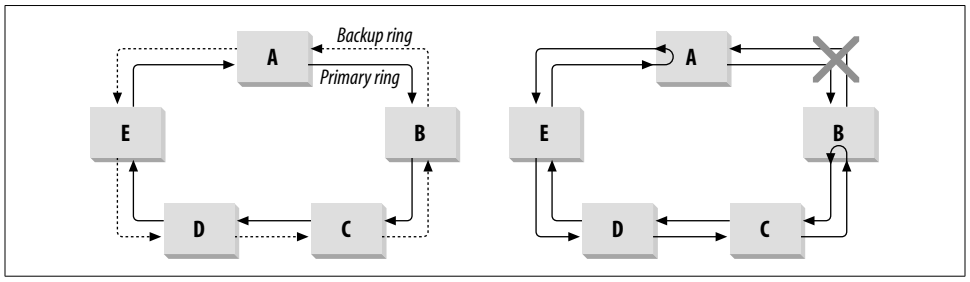

Figure 3-5. Fault tolerance of a dual-ring architecture

Star topology

In practice, most Ethernet and Token Ring LANs are implemented in a star topology. This implementation means that a central device connects to all of devices. All devices communicate with one another by passing packets first to this central device.

In one option for a star topology, the central device aggregates the traffic from every device and broadcasts it back out to all other devices, letting them decide for themselves packet by packet what they should pay attention to. This is called a *hub*. Alternatively, the central device could act as a *switch* and selectively send traffic only where it is intended to go.

The star topology is often called *hub and spoke*, as an analogy to a bicycle wheel. This term can be misleading because sometimes the hub is a hub and sometimes it's a switch of some kind. So I prefer the term *star*.

Most modern LANs are built as stars, regardless of their underlying technology. There are many reasons for this. It's certainly easier to upgrade a network by upgrading only the device in the closet, without having to change the expensive cabling to every desk. It's also much easier to make fast switching equipment in a small self-contained box than it would be to distribute the networking technology throughout the work area.

Even when Token Ring and Ethernet are implemented using a star topology, they still obey their own rules internally. For example, a Token Ring MAU transmits frames to each port in succession, waiting each time until it receives the frame back from the port before transmitting it to the next port. In Ethernet, however, the hub simultaneously transmits the frame to all ports.

The prevalence of star topology networks has made it possible to build general-purpose structured cable plants. The cable plant is the set of cables and patch panels that connect all user workspaces to the aggregation point at the center of the star.

With a structured cable plant of Category 5 cabling and IBDN patch panels, it's relatively easy, for example, to switch from Token Ring to Ethernet or from Ethernet to Fast Ethernet. Executing a change like this means installing the new equipment in the wiring closet, connecting it to the rest of the network in parallel with the existing infrastructure, and then changing the workstations one by one. As each workstation is changed, the corresponding cable in the wiring closet is moved to the new switching equipment.

Chapter 4 discusses structured cable plants in more detail.

When it comes to fault tolerance, however, star topologies also have their problems. The central aggregation device is a single point of failure. There are many strategies for reducing this risk, however. The selection and implementation of these strategies are central to a good network design.

Mesh Topology

A mesh topology is, in some ways, the most obvious way to interconnect devices. A meshed network can be either *fully meshed* or *partially meshed*. In a fully meshed network, every device is connected directly to every other device with no intervening devices. A partial mesh, on the other hand, has each device directly connected to several, but not necessarily all of the other devices.

Clearly, defining a partial mesh precisely is a bit more difficult. Essentially, any network could be described as a partial mesh with this definition. Usually, a mesh describes a network of multiple point-to-point connections that can each send and receive in either direction. This definition excludes descriptions of both the ring and bus topologies because the ring circulates data in only one direction and the bus is not point-to-point.

Since a mesh has every device connected to every other device with nothing in between, the latency on this sort of network is extremely low. So why aren't mesh networks used more? The short answer is that mesh networks are not very efficient.

Consider a fully meshed network with N devices. Each device has to have (N–1) connections to get to every other device. Counting all connections, the first device has (N–1) links. The second device also has (N–1) links, but the one back to the first

device has already been counted, so that leaves (*N*–2). Similarly there are (*N*–3) new links for the third device, all the way down to (*N*–*N* = 0) for the last device (because all of its links were already counted). The easiest way to see how to add these devices up is to write it in a matrix, as shown in Table 3-1.

Table 3-1. Connections in a meshed network

	1	2	3	4	…	N
1	x	1	1	1		1
2		x	1	1		1
3			x	1		1
4				x		1
…					…	…
N						x

An "x" runs all the way down the diagonal of the matrix because no device talks to itself. The total number of boxes in the matrix is just N^2. The number of entries along the diagonal is *N*, so there are (N^2–*N*) links. But only the upper half of the matrix is important because each link is only counted once (the link from a → b is included, but not b → a, because that would be double counting). Since there is exactly the same number above the diagonal as below, the total number of links is just $N(N–1)/2$.

Making a fully meshed network with 5 devices requires 5(5-1)/2 = 10 links. That doesn't sound so bad, but what happens if this number is increased to 10 devices? 10(9)/2 = 45 links. By the time you get to a small office LAN with 100 devices, you need 100(99)/2 = 4950 links.

Furthermore, if each of these links is a physical connection, then each of the 100 devices in that small office LAN needs 99 interfaces. It is possible to make all those links *virtual*—for example, with an ATM network. But doing so just moves the problem and makes it a resource issue on the ATM switching infrastructure, which has to keep track of every virtual circuit.

The other reason why meshed networks are not particularly efficient is that not every device needs to talk to every other device all of the time. So, in fact, most of those links will be idle most of the time.

In conclusion, a meshed topology is not very practical for anything but very small networks. In the standard jargon, it doesn't scale well.

Scalability

This discussion has just looked at certain basic network topologies. These concepts apply to small parts of networks, to workgroups, or to other local groupings. None of the basic topologies mentioned is particularly useful for larger numbers of users,

however. A mesh topology doesn't scale well because the number of links and ports required grow too quickly with the number of devices. But ring and bus architectures also don't scale particularly well.

Everybody seems to have a different rule about how many devices can safely connect to the same Ethernet segment. The number really depends on the traffic requirements of each station. An Ethernet segment can obviously support a large number of devices if they all use the network lightly. But in a Token Ring network, even devices that never talk must take the token and pass it along. At some point, the time required to pass the token all the way around the ring becomes so high that it starts to cause timeouts. The number of ring members required to achieve this state is extremely high, though. Other types of problems generally appear first.

Both Ethernet and Token Ring networks have theoretical upper limits to how much information can pass through them per second. Ethernet has a nominal upper limit of 10Mbps (100Mbps for Fast Ethernet and 1000Mbps for Gigabit Ethernet), while 4, 16, and 100Mbps Token Ring specifications are available. Clearly, one can't exceed these nominal limitations. It actually turns out that the practical limits are much lower, though, particularly for Ethernet.

The collision rate governs throughput on an Ethernet network. Thus, the various rules that people impose to set the maximum number of devices in a particular collision domain (i.e., a single Ethernet segment) are really attempts to limit collision rates. There is no generally reliable rule to decide how many devices can go on one segment.

This fact is easy to deduce from a little calculation. Suppose you have an Ethernet segment with N devices. Each device has a certain probability, P, of wanting to use the network at any given moment. The probability of having k simultaneous events is:

$$ {}_kP_N = \frac{N!P^k(1-P)^{N-k}}{k!(N-k)!} $$

Thus, for two devices, both wanting to talk at the same time, $k = 2$.

$$ {}_2P_N = (1/2)N(N-1)P^2(1-P)^{N-2} $$

$$ \sim (1/2)N^2P^2 $$

Taking this equation a step further to work out real numbers is more difficult because it would require a detailed discussion of collision back-off algorithms. One would also have to be very careful about how P was calculated, as a collision is only counted when two devices actually send packets simultaneously. Usually, one sends first and the second device simply buffers its packet and waits until the wire is free. But the most important result is already here. The probability that two devices want to talk at the same time is proportioned to N^2, where N is the number of devices on the segment.

Interestingly, the probability goes like P^2. P is the probability that a particular device will want to use the network (in a suitable unit of time, such as the MTU divided by the nominal peak bandwidth). This probability is clearly going to be proportional to the average utilization of each device. The probability ${}_2P_N$ is essentially the probability that a device will have to wait to transmit because another device is already transmitting. Since the probability of having to wait is proportional to P^2, a small increase in the average utilization per device can result in a relatively large increase in the collision rate. But the real scaling problem is because of the factor of N^2, which rises very quickly with the number of devices.

This is why there are so many different rules for how many devices to put on an Ethernet segment. The number depends on the average utilization per device. A small increase in this utilization can result in a large increase in the collision rate, so it is not safe to trust these general rules.

Remember that collision rates cause the effective throughput on an Ethernet segment to be significantly smaller than the nominal peak. You will never get a 10Mbps throughput on a shared 10BaseT hub. You will never get 100Mbps on a Fast Ethernet hub, either. In fact, if there are more than 2 or 3 devices you probably can't get close to that nominal peak rate. Typically, the best you will be able to get on a shared Ethernet segment is somewhere between 30 to 50%. Sometimes you can do better, but only if the number of talking devices is very small. This is true for both Ethernet and Fast Ethernet hubs, but it is not true for switches.

Each port on an Ethernet switch is a separate collision domain. If every device is connected to its own switch port, then they are all on their own collision domains. Now they can all talk at the same time, and the switch will make sure that everything gets through.

Token Ring, on the other hand, has a much simpler way of avoiding contention. If two devices want to talk at the same time, they have to wait their respective turns. If another device is inserted into the ring, then everybody has to wait slightly longer. The average amount of time that each device has to wait is roughly proportional to the number of devices on the ring, N. This result is much better than N^2.

Also note that in Ethernet, the collision rate goes up proportionally to the square of the average utilization of each device. In Token Ring, the average wait time between each device's transmission bursts is the corresponding rate limiting factor. This factor scales roughly to the average per device utilization, not its square.*

As a result, a Token Ring "segment" can hold more devices than an Ethernet segment before contention becomes a serious problem. It's also much safer to rely on

* Some people say that Token Ring is deterministic because of this property, meaning that you can readily calculate how the traffic from a group of devices will aggregate on the entire ring. But you can do similar calculations for Ethernet if you understand how to combine probabilities and how the collision mechanisms work. It's just a harder calculation. Since everything is measured statistically anyway, having a deterministic model for your network is actually not much of an advantage.

general rules for how many devices to put on a ring. Even with Token Ring, there is an upper limit of how many devices can take part in a particular segment. Efficiency usually demands that you break up your rings through a bridge or a switch, exactly the same as for Ethernet.

You have seen that all of the basic LAN building blocks have different types of scaling problems. A 16Mbps Token Ring can hold more devices than a 10Mbps Ethernet segment, but in both cases there is a practical upper limit to how many devices you can put on the network before you start having performance problems. I have already alluded to one practical solution that allows us to continue growing our network beyond these relatively small limitations: switches.

You can connect a large number of Ethernet segments or Token Rings with a central switch. This switch will create a single point of failure, as I discussed in the previous chapter, but it will also move the problem up only a level. Now, instead of having a limit of *N* devices per segment, there is a limit of *N* devices times the number of ports on the switch. Expanding beyond this new upper limit is going to create a new problem.

Solving this new problem is what this whole book is about.

Reliability Mechanisms

Before moving on to larger-scale topologies, it is important to review some of the systems for automated fault recovery that are used in large networks. Just inserting backup switches and routers connected with backup links is not enough. The network has to be able to detect problems quickly with its primary paths and activate the backup devices and links.

There are two main methods for doing this, and most large-scale networks use both. You can detect and repair the fault at either Layer 2 or at Layer 3. The Layer 2 mechanism employs a special IEEE standard called Spanning Tree. As an IEEE standard, Spanning Tree is applicable across a wide range of Layer 2 networks, including the commonly used Ethernet and Token Ring protocols.

Conversely, there are many different ways of detecting and working around faults at Layer 3. Selecting among these different possibilities depends on what the Layer 3 protocols on the network are and on the scope of the fault tolerance. There are purely local mechanisms as well as global ones.

Spanning Tree

Spanning Tree, also called STP or IEEE 802.1d, is a Layer 2 protocol that is designed to accomplish two important tasks. It eliminates loops and it activates redundant links for automated fault recovery. Figure 3-6 shows a simple bridged network that employs Spanning Tree for both of these purposes.

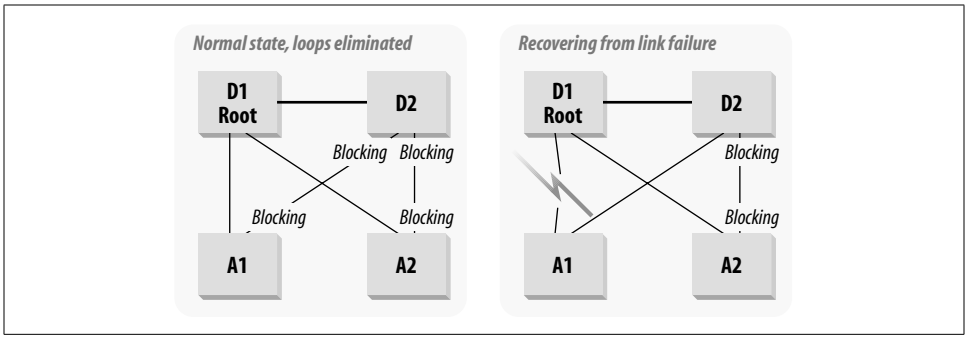

Figure 3-6. Spanning Tree is used to eliminate loops and activate backup links

Figure 3-6 has four switches. D1 and D2 are the central Distribution switches, while A1 and A2 are downstream Access switches that connect to end-device segments. The Spanning Tree priorities have been adjusted to give D1 the lowest value, making it the "Root Bridge."

Now, suppose D1 has a packet intended for a device on A1. It has several ways to get there. It can go directly, or it can go over to D2, which also has a link to A1. Or, it can go to A2, back to D2, and over to A1. The worst thing it could do is to go around in circles, which it can also do in this diagram. In fact, every device in the picture except the one containing the end device wants to helpfully forward the packet along to any other device that might be able to deliver it. This forwarding results in a big mess of loops. Spanning Tree removes this problem.

Spanning Tree eliminates loops

Each port taking part in Spanning Tree can be in one of five different states: blocking, forwarding, listening, learning, or disabled. Blocking means that Spanning Tree is preventing this port from forwarding packets. Each switch looks at its neighbors and inquires politely whether they are the Root Bridge or whether they can help it get to the Root Bridge. Only one Root Bridge is allowed in a broadcast domain, and that device is the logical center of the network. This is why the priorities have been set manually to force the network to elect D1 as the Root Bridge. D2 is configured to be the second choice in case D1 fails. You never want a switch that serves the end-device Access level of the network to be Root Bridge.

In this way, the network is always able to designate a Root Bridge. This device serves as the main Distribution point for all packets that a switch can't otherwise deliver itself. Every switch keeps track of the next hop that will take it to the Root Bridge. In effect, the network, with all of its redundant cross-connections, becomes a simple tree topology, which eliminates the loops.

Spanning Tree activates backup links and devices

Now suppose the link from A1 to D1 suddenly fails, as shown in the diagram. A1 knows it has lost its Root Bridge connection because it stops exchanging hello packets on that port. These packets exist purely for this reason—to keep checking that everything is working properly. When the link breaks for any reason, A1 remembers that it had another link to the Root Bridge via D2, and so it tentatively activates that port. It isn't certain yet that this way is correct, so it doesn't start forwarding data; instead, it goes into a "listening" state. This state allows it to start exchanging Spanning Tree information over this other port—to see if this is the right way.

Once A1 and D2 have established that they can use this link as a valid backup, both ports go into a *learning* state; they still do not forward data packets; they first must update their MAC address tables. Until the tables are updated, switch D2 doesn't know what devices are on A1. Then, once they have synchronized all of this information, both switches set this new port to a *forwarding* state, and the recovery process is complete.

In this picture, all switches are connected directly to the Root Bridge, D1. All links that do not lead to Rome are set to the blocking state, so A1 and A2 both block their links to D2. At the same time, D2 sets all the links it has that do not lead to D1 to blocking as well. The other links—the ones that do lead to the Root Bridge—are all set to their forwarding state.

The thick line connecting D1 and D2 is a higher bandwidth link. Suppose the thin lines are 100Mbps Ethernet links, while the thick line is a 1000Mbps Gigabit Ethernet link. Clearly the thick line is a better link to the Root Bridge than one of the slower links. So, the engineer sets the priority on this port so that, if there is a choice between what link to use, the switch always chooses the faster one.

Having a link between D1 and D2 is important. Imagine what would happen if it were not present and the link between A1 and D1 failed. A1 would discover the new path to the Root Bridge through D2, exactly as before. However, D2 doesn't have its own link directly to D1, so it must instead pass traffic through A2 to get to the Root Bridge. This means that traffic from A1 to the Root Bridge must follow the circuitous path—A1 to D2 to A2 to D1. In this simple example, the traffic passes through every switch in the network! This situation is clearly inefficient.

But wait—it gets worse. Now suppose a third switch, A3, is connected to both D1 and D2, and the link from A1 to D1 fails. A1 will again try to use D2 as its backup, but D2 now has two possible paths to the Root Bridge—one through A2 and the other through A3. It picks one of these links at random as the best path—say A2—and it blocks the other. Now, because of an unrelated failure elsewhere in the network, A2 has extra traffic load and A3 has no redundancy.

In the hierarchical network designs that this book recommends, the configuration with two redundant Core (or Distribution) switches connected to each of several Access switches will be common. Therefore, it is important to include a separate trunk connecting the two central switches each time this configuration is used.

It can take several seconds for conventional Spanning Tree to activate a backup link. This may not sound like a long time, but it can be a serious problem for some applications. Fortunately, trunk failures don't happen very often, but techniques for improving recovery time are available.

Spanning Tree has three adjustable timers that can be modified to make convergence faster. These times are called *hello*, *forward delay*, and *maximum age*. All bridges or switches taking part in Spanning Tree send out hello packets to their neighbors periodically, according to the hello timer. All neighboring devices must agree on this interval so that they all know when to expect the next hello packet. If the timers do not agree, it is possible to have an extremely unstable network, as the switch with the smaller timer value thinks that its trunks are continuously failing and recovering.

The forward delay timer determines how long the switch will wait in the listening and learning states before it sets a port to the forwarding state. The maximum age timer determines how long the switch should remember old information.

By reducing the hello and forward delay timers, you can improve your convergence time in a failure, but there are limits to how far you can push these numbers. The forward delay timer exists to prevent temporary loops from forming while a network tries to recover from a serious problem. The switch has to be certain that the new link is the right one before it starts to use it.

For example, suppose your Root Bridge fails. In this case, all switches must elect a new Root Bridge. In the example, the priorities are adjusted so that, if D1 fails, D2 becomes the Root Bridge. D2 has to realize that D1 has failed and has to alert every other device that it is now the Root Bridge. The forward delay timers on all of these switches have to be long enough to allow this process to complete.

Having a short hello interval is the easiest way to speed up the convergence of a Spanning Tree network. But even this process has to be done carefully. Remember that a packet is sent in both directions over all of your production trunks once every time interval. If the interval becomes too short, then link congestion and CPU loading problems can result. If hello packets are dropped for these reasons, Spanning Tree may assume that links have failed and try to find alternate paths.

The best set of Spanning Tree timers vary from network to network. By default, the values of the hello and forward delay timers will be approximately a few seconds each. The best way to determine the appropriate values is to start with the defaults and then try reducing them systematically. Then try deliberately failing links to verify that these settings result in a stable network. In most cases, the default parameters

are very close to optimal. Since timer changes must be made on all devices, it is generally best to use the defaults unless there is a compelling requirement to improve convergence efficiency.

Some switch vendors have implemented additional Spanning Tree features that facilitate faster convergence and greater stability. Generally, these features work by allowing ports that are nominally in blocking states to behave as if they are in a perpetual learning state. This way, in the event of a simple failure, they can find the new path to the Root Bridge more quickly. Of course, in the case of a Root Bridge failure, the network still has to calculate a new topology, and this calculation is difficult to speed up.

Layer 3 Recovery Mechanisms

There are two main methods of implementing fault tolerance at Layer 3. You can either take advantage of the dynamic routing protocol to reroute traffic through the backup link or you can use an address-based redundancy scheme, such as HSRP (Hot Standby Router Protocol) or VRRP (Virtual Router Redundancy Protocol). The choice depends on the location.

I have already said that running a dynamic routing protocol on any end devices is a bad idea. If the problem is to allow end devices to stop using a failed default gateway on their LAN segment and use its backup instead, the dynamic routing protocol can't help. Instead, you need to use HSRP or VRRP. There is considerable similarity between these two protocols, which is why I mention them together. HSRP is a Cisco proprietary system defined in RFC 2281, and VRRP is an open standard defined in RFC 2338. In general, it is not a big problem to use the proprietary standard in this case because, if two routers are operating as a redundant pair, the chances are good that they are as nearly identical as possible; they will almost certainly be the same model type and probably have the same software and card options. This is one of the relatively rare cases in which the open standard doesn't matter very much.

Both of these protocols work by allowing end devices to send packets to a default gateway IP address that exists on both routers. However, end devices actually send their packets to the Layer 2 address associated with that default gateway IP address in their ARP cache. They don't use the default gateway address directly. When the backup router takes over for the primary router's default gateway functions, it must adopt both the IP address and the Layer 2 MAC address. Both VRRP and HSRP have quick and efficient methods of making this change. When one router fails, the other takes over and the end stations on that segment are not even aware that a problem has occurred.

On segments that do not have any end stations, particularly router-to-router segments, there is no need for HSRP or VRRP. In these cases, all devices can take part in the dynamic routing protocol (such as OSPF). In these places, using HSRP or VRRP

is not a good idea because it has the potential to confuse the routing tables of the other routers on the segment. These routing protocols are very good at maintaining lists of alternative paths and picking the one that looks the best. If two paths have the same "cost," then most routers simply use both, alternating packets between them. If one router fails, the other routers quickly drop it out of their routing tables and start using the remaining path exclusively.

VLANs

VLAN is an acronym for "Virtual LAN." This name gives a good picture of what it is. A VLAN is, in effect, a *logical* LAN segment. But physically, it is spread throughout a larger network. The term VLAN also refers to a LAN port grouping within a single switch. If ports 1, 2, 5, and 12 are all part of the same broadcast grouping on an Ethernet switch, then this segment is also often called a VLAN. However, this designation is used mainly for simplicity when this switch is connected to another switch and they share this VLAN between them.

Figure 3-7 shows two switches connected by a trunk. Each switch has three VLANs. Switch A has VLAN 1, VLAN 2, and VLAN 3, while Switch B has VLAN 1, VLAN 2, and VLAN 4. Designating VLANs with numbers in this way is common. Ports 1, 2, 5, and 12 of Switch A are assigned to VLAN 1. On Switch B, VLAN 1 consists of ports 3, 5, and 7. Since these two switches are connected through a trunk, all seven ports can now communicate as if they were all part of the same LAN segment.

In an IP network, the ports from the same VLAN can all be part of the same IP subnet. In an IPX network, then they share the same IPX network number. Other ports on both switches are unable to communicate with any of these ports except through a router. They must all be on different IP or IPX networks.

Similarly, all ports assigned to VLAN 2 on Switch A are part of the same logical network as the VLAN 2 ports on Switch B. To make things a little more interesting, I have also included a VLAN 3 and a VLAN 4. VLAN 3 only appears on Switch A, while VLAN 4 is only visible on Switch B. Since these two VLANs are both entirely local to their respective switches, they do not use the trunk. If I were to define a new VLAN 3 on Switch B and assign a port to it, it could also use the trunk to allow the VLAN 3 ports on both sides to communicate.

So that's a VLAN; it's a simple but exceptionally powerful and useful concept. Like all powerful concepts, it can be used well or abused horribly.

The advent of VLAN technology was a mixed blessing to large-scale LANs. On the one hand, it has made it much easier to build a rational hierarchical network with a minimal number of components, which is very cost effective. On the other hand, VLANs make it easy to construct extremely bad network designs.

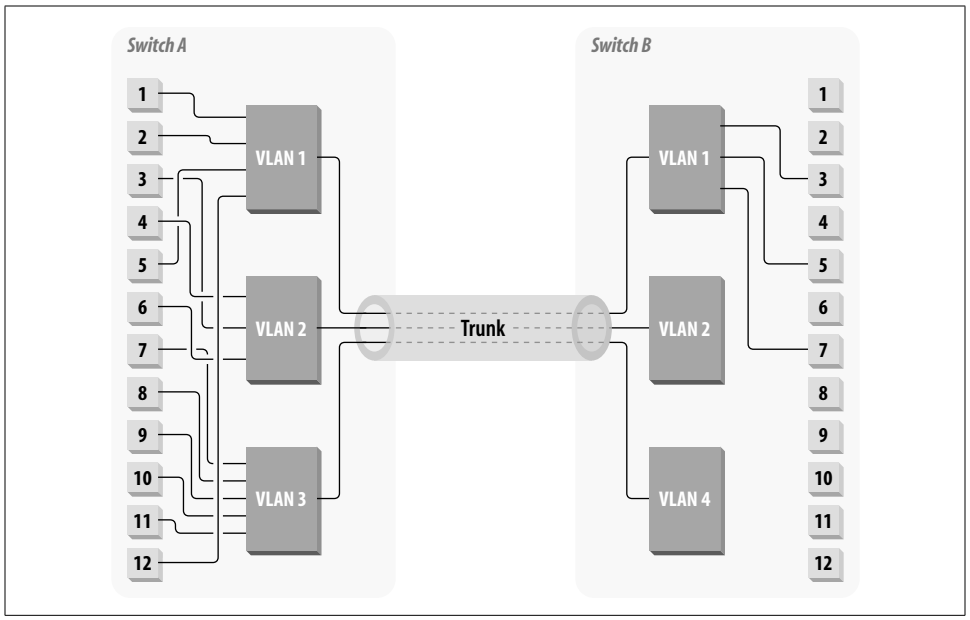

Figure 3-7. VLANs are shared through trunks

Avoid Spaghetti VLANs

The worst thing you can do in any network is build a random mess of spaghetti. With VLAN technology, you can create a completely rational and logical physical network of switches and trunks and then ruin it by superimposing an irrational VLAN design on top of it. You can assign one port on each of a hundred switches in a dozen different buildings to the same VLAN. All VLANs can exist everywhere simultaneously.

But why would this situation be bad? It sounds like it could be a great thing. You could have a highly integrated office real-estate plan so a working group or department may have members spread throughout the campus. Well, there are two main problems with this sort of topology. First, it's hard to manage and troubleshoot problems in such a network. Second, it leads to terrible problems with latency and trunk congestion.

Consider the first problem. Suppose you have to troubleshoot a problem in which two devices cannot communicate. The first step in such a problem is to figure out where these devices are in both a Layer 2 and a Layer 3 picture. Once you have located these devices, you try to figure out where things are broken. Is there an intermediate device that can get to one of these devices? Can other devices on the same IP subnet as one of the problem devices communicate with it? Can this third device communicate with the other end?

Those are the questions an engineer always starts with. In a network of spaghetti VLANs, however, it is entirely possible that this third device is in a completely different part of the physical network. If it can't communicate, you may have only proved that there is a physical problem. Is it part of the same physical problem or a different one? You have to determine where these devices are both logically and physically and figure out what trunks they use to communicate through. Unraveling the spaghetti of the VLANs can be extremely frustrating and time consuming.

Now, suppose you have a large network in which every VLAN is present on every switch. A broadcast from one device on one of these VLANs must be sent to all other devices on the same VLAN. That means that a broadcast has to go to every other switch and traverse every trunk. This scenerio is at least as inefficient as building a huge bridged network where every device is effectively part of the same single VLAN.

Lesson number one in building a network with VLANs is to use them sparingly and thoughtfully. VLAN is an extremely powerful concept with wide-ranging benefits to a network designer. But power usually comes with risks, and I want to help you to realize the benefits while minimizing these risks.

An old rule of network design, the 80/20 rule, is intended to keep loads down on routers. Some designers have used 60/40, 70/30, or even 90/10, but just about everybody has such a rule. It says that 80% of your traffic is local and only 20% need to cross the Core. Clearly, the less traffic that has to cross through the Core, the happier and less congested it will be, but making these sorts of rules isn't always practical. As network designers, we have very little control over how the applications are used, but we can exercise some direction. If most user traffic is destined for one central mainframe device, then there is no way we will ever be able to make such rules.

This rule is useful in VLAN construction, particularly in deciding which users will be in which VLAN groupings. But it is important to weigh this rule against the Spaghetti Factor. The point of the 80/20 rule is to try to reduce loading on the routers that direct your VLAN-to-VLAN traffic. In some organizations this is not practical, sometimes the only way to create well-segmented VLANs is by adding too many devices to the VLAN or by making every VLAN present on every switch. In such situations, remember that the point is to create a stable, reliable network; in a conflict, the 80/20 rule should be sacrificed before reliability and manageability.

Protocol-Based VLAN Systems

I need to talk about one other extremely serious VLAN-related trouble pit before moving on to specific VLAN topologies. There are really two main ways that a switch can color packets to associate them with a particular VLAN. The switch can say that every packet coming from any given switch port is automatically on only one VLAN. Or, alternatively, it can look at each packet and decide what VLAN to put it on based on what it sees.

Some switch vendors have implemented their VLAN technology with some clever protocol-dependent VLAN tagging features. Each time one type of packet (e.g., a particular application) is sent out, the switch assigns that packet to VLAN 1 and forwards it appropriately. Another packet, corresponding to a different application, would be forwarded as if it were on VLAN 2.

This feature sounds, on the surface, like it should be extremely useful and clever. It is definitely clever. But please use it with extreme caution. Many potential disasters are hiding in a feature this clever.

First, suppose that both protocols are IP-based. Then the network has a serious problem. How is it supposed to handle the IP addressing of these two VLANs? Which one will the default router for this IP address range take part in? It could be set up to take part in both. This way, the packets used by a particular application are shunted off onto a different VLAN so they can use higher speed trunks. But this leads to serious troubleshooting and fault-tolerance problems. So it should be avoided.

This sort of topology might be useful if a semitrusted external information vendor's server is to be placed on the network. Then, when workstations communicate with that server, the traffic is segregated from the rest of the network. This segregation could have important security benefits because this special server could then be prevented from taking part in the regular VLAN for these users. In other words, the protocol-based VLAN tagging feature is a sort of security filter. However, if you want a security filter, why not just use a filter? It is simpler conceptually to just implement a security filter on the one VLAN so packets from the vendor's switch port are treated specially.

Suppose you want to have your IP traffic all use one VLAN and have all other protocols use a different VLAN. This situation is actually more useful and sensible than the all-IP case. You might use some nonroutable protocol such as NetBEUI or a legacy LAT application. Then you could construct your network so everybody takes part in the same VLAN for the nonroutable traffic, but your IP traffic would be segregated. This is, in fact, the only time when I would recommend using this sort of feature.

You must be extremely careful when you try to troubleshoot this network. You have to remember that the nonroutable protocol is on a different VLAN than the IP traffic. So traditional troubleshooting tools such as *ping* and *traceroute* are not going to provide useful information on this other VLAN. A *ping* may work fine over the IP VLAN, but that has nothing to do with how the same network handles the NetBEUI VLAN. In general, this sort of feature should be considered dangerous and avoided unless there is absolutely no other way to accomplish the design goals. Even then it should be used with extreme care.

Usually, the best way to implement VLANs is by switch port. This way, each device is a member of only one VLAN, regardless of protocol. Overlaying several different logical topologies on the same network will always cause confusion later when troubleshooting unrelated network problems.

Toward Larger Topologies

Until now, this chapter looked at small-scale LAN structures and described some of the concepts, such as VLANs and reliability mechanisms, that allow designers to glue these small-scale concepts together into a large network. Now I'd like to move on to talk about how these basic building blocks are used to put together large-scale networks. To do this, I need to put many of these ideas into their historical context. New technology has allowed larger and more stable networks. It is useful to talk about the simpler creatures that evolved into more sophisticated modern networks. By reviewing how we got where we are, I hope to prevent you from making old mistakes or reinventing old wheels.

Collapsed Backbone

There are many ways to create larger networks from basic LAN segments. The simplest is to just interconnect several Ethernet segments or Token Rings via a single switch. This type of large-scale LAN architecture is called a *Collapsed Backbone*. Although it may sound like the painful result of a highway accident, the Collapsed Backbone topology gets its name from the concept of a network backbone that interconnects various segments.

In general, the backbone of the network can be either collapsed or distributed. I use the general term backbone to refer to a high-capacity part of the network that collects traffic from many smaller segments. It can gather traffic from several remote LANs onto a network backbone that connects to a central computer room.

The network backbone concept also works well in more peer-to-peer networks where there is no central computer room, but there is communication among the various user LANs. Figure 3-8 shows a simple example of a traditional network backbone design. In the early days of LAN design there was no such thing as the collapsed backbone—it was itself just some sort of LAN.

Why collapse a backbone?

The various user LANs connect to some sort of shared medium that physically runs between the separate areas. This medium could be some flavor of Ethernet, in which case the little boxes making these connections could be bridges, switches, or repeaters of some kind. Or the backbone could be a completely distinct network technology

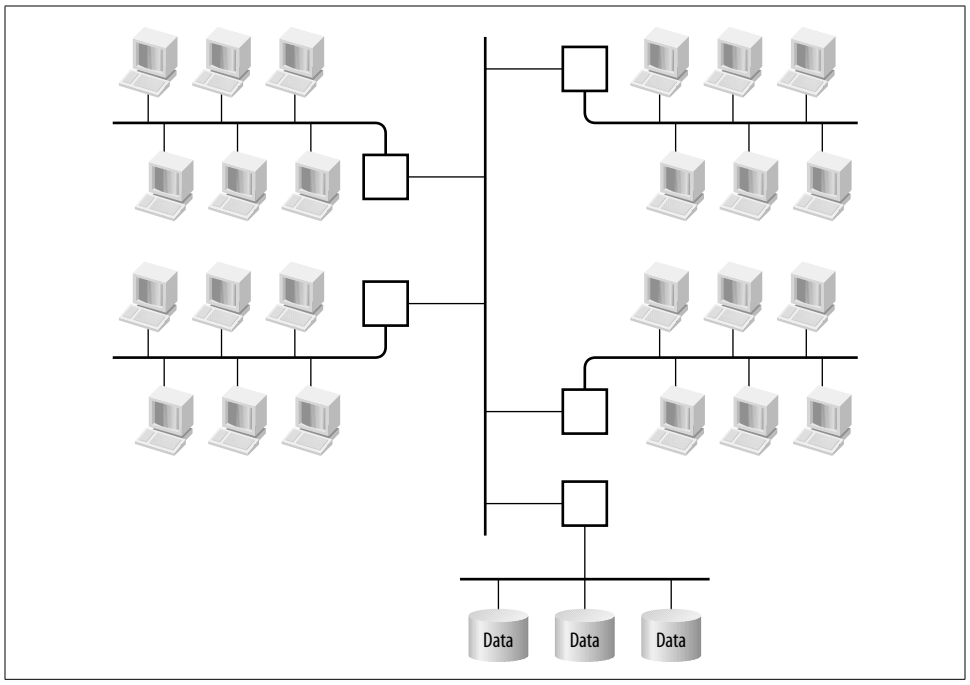

Figure 3-8. A simple network backbone technology

such as ATM or FDDI, in which case the little boxes must be capable of interconnecting and converting between these different network types.

Figure 3-9 shows the same diagram with a collapsed backbone. Here, some sort of central router or switch has long-haul connections to the various user areas. Typically, these connections would be fiber connections. Note that there is still a backbone, exactly the same as in the previous diagram, but here the backbone is contained inside the central concentrator device.

The two diagrams look essentially similar, but there is a huge potential performance advantage to the collapsed backbone design. The advantage exists because the central concentrator device is able to switch packets between its ports directly through its own high-speed backplane. In most cases, this means that the aggregate throughput of the network is over an order of magnitude higher.

The essential problem is that all network segments must share the bandwidth of the backbone for all traffic crossing it. But how much traffic is that? If the separate segments are relatively autonomous, with their own file and application servers, there may be very little reason to send a packet through the backbone. But, in most large LAN environments, at least one central computer room contains the most heavily used servers. If everybody shares these servers, then they also share the backbone. Where will the bottleneck occur?

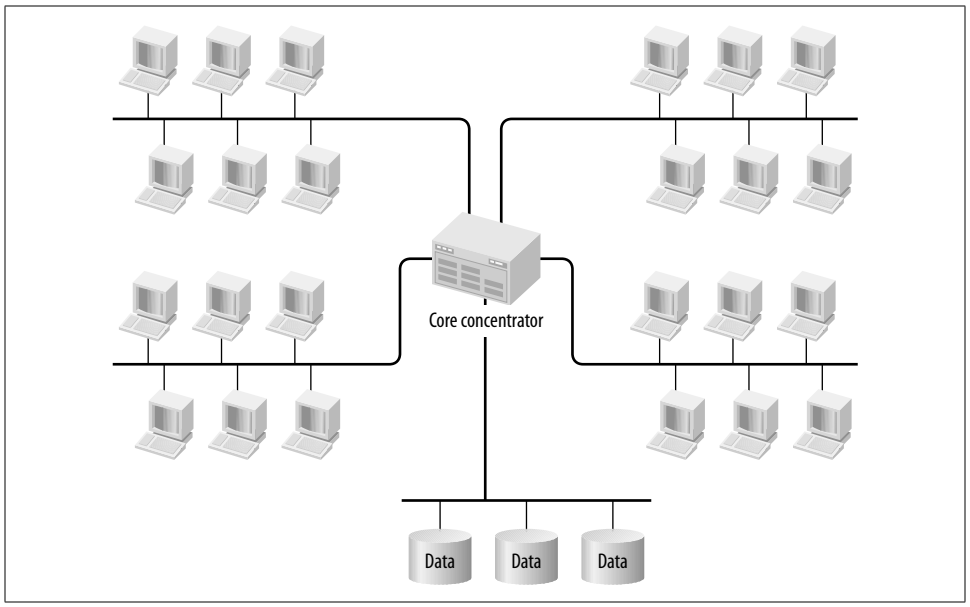

Figure 3-9. A collapsed backbone technology

Backbone capacity

In the diagram shown, bottleneck is actually a bit of a moot point because there is only one central server segment. If all traffic crossing the backbone goes either to or from that one segment, then it's fairly clear that all you need to do is control backbone contention a little bit better than on the server segment and the bottleneck will happen in the computer room. But this is not the usual case. Drawing the central server segment with all of those servers directly connected to a Fast Ethernet switch at full duplex would be more realistic. With just three such servers (as in the drawing), the peak theoretical loading on the backbone will be 600Mbps (100Mbps for Fast Ethernet times two for full duplex times three servers).

Clearly that number is a maximum theoretical burst. In the following section I will discuss how to appropriately size such trunk connections. The important point here is that it is very easy to get into situations in which backbone contention is a serious issue.

This is where the collapsed backbone concept shows its strength. If that central concentrator is any commonly available Fast Ethernet switch from any vendor, it will have well over 1000Mbps of aggregate throughput. The backplane of the switch has become the backbone of the network, which provides an extremely cost effective way of achieving high throughput on a network backbone. The other wonderful advantage to this design is that it will generally have significantly lower latency from end to end because the network can take advantage of the high-speed port-to-port packet switching functions of the central switch.

In Figure 3-8, each user segment connects to the backbone via some sort of Access device. The device may be an Ethernet repeater, a bridge, or perhaps even a router. The important thing is that any packet passing from one segment to another must pass through one of these devices to get onto the backbone and through another to get off. With the collapsed backbone design, there is only one hop. The extra latency may or may not be an issue, depending on the other network tolerances, but it is worth noting that each extra hop takes its toll.

Backbone redundancy

The biggest problem with this collapsed backbone design should already be clear. The central collapse point is also a single point of failure for the entire network. Figure 3-10 shows the easiest way around this problem, but forces me to be more specific about what protocols and technology the example network uses.

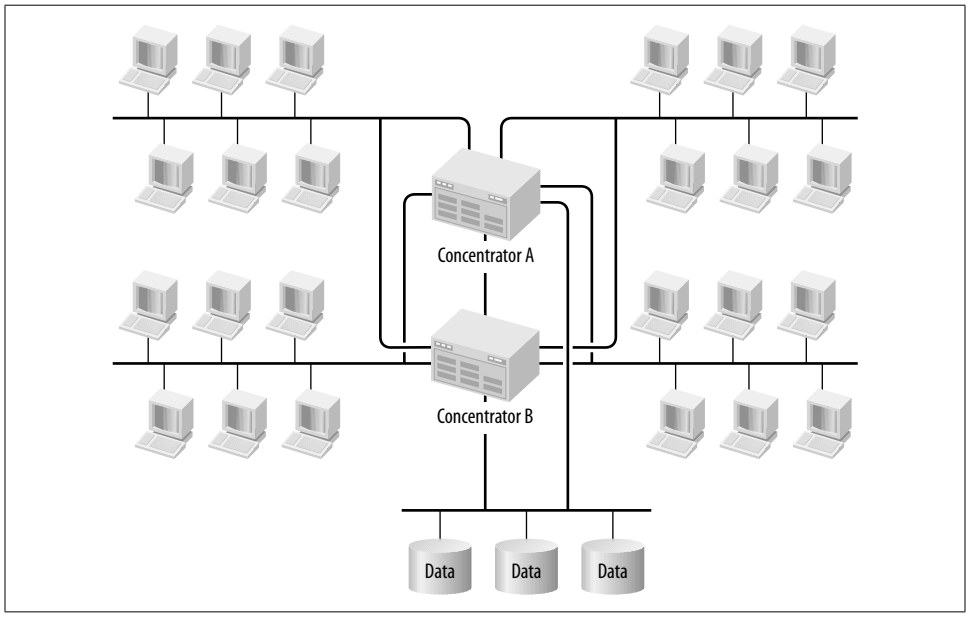

Figure 3-10. A collapsed backbone with redundancy

The most common way to collapse a LAN backbone is through a layer 2 Ethernet switch. So let's suppose that each user segment is either Ethernet or Fast Ethernet (or perhaps a combination of the two). The central device is a multiport Fast Ethernet switch with an aggregate backplane speed of, say, 1Gbps (this number is much lower than what is currently available in backbone switches from any of the major vendors, but it's high enough for the example). Each user LAN segment connects to these central switches using two fiber optic Fast Ethernet connections, one to each switch.

Then the two switches can be configured to use the Spanning Tree protocol. This configuration allows one switch to act as primary and the other as backup. On a port-by-port basis, it is able to ensure that each user LAN segment is connected to only one of the two switches at a time. Note that a switch-to-switch connection is indicated in the diagram as well. This connection is provided in case LAN segment 1 is active on Switch A and segment 2 is active on Switch B. If this happens, there needs to be a way to cross over from one switch to the other.

There are several important redundancy considerations. First, it may seem more complicated to use port-by-port redundancy rather than redundancy from one whole switch to the other. After all, it means that there will probably be complicated switch-to-switch communication, and seems to require the switch-to-switch link that wasn't previously required. But this is actually an important advantage. It means that the switch can suffer a failure affecting any one port without having to flip the entire backbone of the network from one switch to the other. There are a lot of ways to suffer a single port failure. One could lose one of the fiber transceivers, or have a cut in one of our fiber bundles, or even have a hardware failure in one port or one card of a switch. So minimizing the impact to the rest of the network when this happens will result in a more stable network.

This example specified Ethernet and Spanning Tree, but there are other possibilities. If all LAN segments used Token Ring, for example, you could use two central Token Ring switches and the Token Ring flavor of Spanning Tree. Exactly the same comments would apply.

Alternatively, for an IP network you could have done exactly the same thing at Layer 3 by using two central routers. In this case, you could use the Cisco proprietary HSRP protocol or the RFC 2338 standard VRRP protocol. These protocols allow two routers to own this address, but only one is active at a time. The result provides exactly the same port-by-port redundancy and collapsed backbone properties using routers instead of switches.

Distributed Backbone

The alternative to the Collapsed Backbone architecture is a Distributed Backbone. Later in this chapter, I describe the concept of hierarchical network design. At that point, the implementation of distributed backbone structures will become clearer. For now, I need to discuss some general principles.

A Distributed Backbone just indicates more than one collapse point. It literally distributes the backbone functions across a number of devices. In a network of any size, it would be extremely unusual to have a true single collapsed backbone. A large network with a single collapsed backbone would have a terrible single point of failure. It would also probably suffer from serious congestion problems if all inter-segment

traffic had to cross through one point. Even if that collapse point had extremely high capacity, it would probably be difficult to get a high enough port density for it to be useful in a large network.

All practical large-scale networks use some sort of distributed backbone. Moving the backbone functions outside of a single chassis introduces two main problems: trunk capacity and fault tolerance.

Trunk capacity

Suppose you want to distribute your backbone-switching functions among two or more large switches. The central question is how much capacity should you provide to the trunk? By a trunk I mean any high-speed connection that carries traffic for many end-device segments. In this book, I often use the term trunk to refer specifically to a connection that carries several VLANs. I want to consider the more general case here.

A naïve approach would be simply to add up the total burst capacity of all segments feeding this trunk. If you had, for example, 5 Fast Ethernet (100Mbps half-duplex) LAN segments flowing into one trunk, then you would need 500Mbps of trunk capacity. But this scenerio presents a serious problem. How do you practically and inexpensively get this much bandwidth? Do you really have to go to a Gigabit Ethernet or an ATM just because you're trying to run a few trunks? Even load sharing isn't much of an option because you would need as many Fast Ethernet trunks as you have segments, so why use trunks at all in that case?

Needless to say, this approach is not very useful. You have two options for more efficient ways to think about trunk sizing. You could either develop some generally useful rules of thumb, or you could give up completely and just keep throwing bandwidth at it until the congestion goes away. You could actually take a rigorous approach to this second idea by using simulation tools. In the end, you will always have to monitor your trunks for congestion and increase their capacity when you start to get into trouble. A few good rules would give a useful starting point. Trunks should have more capacity than the average utilization. The only question is how much of a peak can the network deal with. Congestion on these trunk links is not a disaster in itself. Later in this book I talk about prioritization schemes to ensure that the important data gets through no matter how heavy the flow is. But there needs to be enough capacity for the normal peak periods, and this capacity needs to be balanced against cost because the higher speed technologies are significantly more expensive to implement.

The key to this discussion is the fact that all end segments are not statistically expected to peak at once. Most of the time, there will be an average load associated with all of them. Every once in a while, one or (at most) two experience a burst to full capacity. The basic rule for sizing trunks is to make sure that they have enough

capacity for two end (shared) segments to peak at the same time plus 25% of capacity for all the remaining end segments. If the trunk has full-duplex transmission, consider the directions separately.

For an example, look at Figure 3-8. A central trunk connects four user segments with a server segment. First assume that this is a half-duplex trunk and that all end segments are 10Mbps Ethernet segments. Then the rule says to allow for two times 10Mbps plus 25% of three times 10Mbps, which works out to be 27.5Mbps. It would be completely safe to use a Fast Ethernet trunk in this case.

If the trunk technology is capable of full-duplex transmission, then you need to consider the two directions separately. Suppose that all traffic is between the users and the servers, with little or no user segment to user segment communication. This situation will help to establish the directions. For the user-to-server direction, there are four 10Mbps Ethernet segments. If two of these segments burst to capacity at the same time, the other two reach 25% of their capacity, and the trunk load will be 25Mbps in this direction. In the other direction, there is only one segment, so if it bursts to capacity, then it will have only 10Mbps in the return direction. As a side benefit, this activity shows that upgrading the server segment to full-duplex Fast Ethernet doesn't force an upgrade on the full-duplex Fast Ethernet backbone.

But this rule doesn't work very well for LANs that have every PC connected to a full-duplex Fast Ethernet port of its own. The rule allows two PCs to burst simultaneously and add 25Mbps to the trunk for every other PC on the network. 50 PCs connected in this way would need a full-duplex trunk with 1.4Gbps in either direction. This doesn't make much sense.

Individual workstations do not behave like nice statistical collections of workstations. The problem is not in assuming that two will burst simultaneously, but rather in the 25% of capacity for the rest. When workstations are connected to a switch like this, the typical utilization per port looks like silence interspersed with short hard bursts. A completely different sort of rule is necessary to express this sort of behavior.

A simple way to say it is that some smal percentage of the workstations will operate at capacity, while the rest do nothing. The actual percentage value unfortunately changes radically depending on the organization and even on the department. A graphic design group that spends its time sending large graphic image files might have a relatively high number. A group that only uses the network for printing the occasional one-page document will have a much smaller number. A general rule requires a reasonable mid-point number that is useful for Distribution trunks in a large network. A fairly safe number for this purpose is 5%. This percentage may be a little on the high side for many networks, so you can consider reducing it to 2.5%. Bear in mind that the smaller this number is, the less capacity for expansion allowed in your network.

Consider another example to demonstrate this rule. Suppose the end-segments in the network shown in Figure 3-8 have switched full-duplex Fast Ethernet to every desk. Suppose that 25 workstations are in each of the four groups. Then, for the user to server traffic, the trunk should allow for 5% of these $4 \times 25 = 100$ workstations to burst to their full 100Mbps capacity simultaneously. Thus, the trunk will operate at 500Mbps in at least this direction. Gigabit Ethernet or ATM can achieve these bandwidths, as can various vendor-proprietary Ethernet multiplexing technologies.

But wait, there's a twist in this example. So far, the discussion has assumed that all traffic is between the users and the servers. So what good does it do if the network can burst to 500Mbps on the trunk for traffic destined for the server segment, if the server segment can't deal with this much traffic? If 5 or more servers are all connected similarly to full-duplex Fast Ethernet switch ports, then this is possible. But the burst would have to be conveniently balanced among these servers. In this case, because traffic patterns are known very precisely, it is possible to reduce the trunk capacity to save money. The point is that this rule is just a starting point. You should always re-evaluate according to your own network conditions. Also note that the rule doesn't apply at all on the server side because you should always expect the servers to work the network very hard.

Trunk fault tolerance

A trunk, like any other part of the network, can fail. If it happens to carry all traffic from some part of the network at the time, though, it could be disastrous. Since trunk failures are potentially serious, it is always wise to include some sort of redundancy in every trunk. In fact, in most organizations I have seen personally, trunk failure is more common than hardware failure on key network equipment. This information is anecdotal, and I have no statistics on it, but it makes sense that delicate strands of optical fiber stretching long distances might be more vulnerable than a tank-like Ethernet switch chassis. If that switch is located in a locked room while the fiber has to run through a conduit shared with other building tenants, there's an even stronger reason to worry about the fiber. In some cases, it is physically damaged while technicians are doing other work. But even if fiber is never touched and the conduit remains sealed forever, eventually it degrades due to a host of environmental hazards, such as background radiation.

All of this information is intended to scare the reader into worrying about trunk failures. In most network designs, the trunks are the first things I would want to provide redundancy for. There are many ways to do so. The actual redundancy mechanism depends on trunk type. If the trunk is itself a multiplexed collection of links (like Cisco's EtherChannel or Nortel's MultiLink Trunking), then redundancy is inherent in the design. In this case, it would be wise to employ an *N*+1 redundancy system. This means that the trunk capacity should be sized as discussed in the previous section, and then increased by one extra link. This way, there is still sufficient capacity if any one link fails.

However, if a single fiber pair carries the trunk, then the only useful way to add redundancy is by running a second full-capacity trunk link. Since one of the main concerns is environmental or physical damage to the fiber, putting this second link through a different conduit makes sense.

The only remaining question is whether to make the backup trunk link a hot standby or to have it actively share the load with the primary link. And the answer, unfortunately, depends on what you can get with the technology you're using. In general, if you can do it, load sharing is better for two reasons:

- In case you inadvertently underestimate your trunk capacity requirements, or in case those requirements grow over time, load sharing gives you extra bandwidth all the time.
- If the primary can fail, so can the backup. The difference is that you notice when the primary fails, and you don't necessarily know when the backup fails. If traffic goes through it all the time, then you'll usually know pretty quickly that you've had a failure of your backup link.

Switching Versus Routing

In the discussion of backbone designs, I mentioned that the same general design topologies are applicable to both Layer 2 and Layer 3 implementations. Thus, at many points the designer can choose to either bridge or route. There are philosophical reasons for choosing one or the other in many cases, but there are also several practical reasons for favoring either switching (bridging) or routing implementations.

Ancient history

The old rule for designing large-scale LANs was "bridge on campus, route off campus." There were good reasons for this rule, but many of these reasons are less relevant today than they once were. Figure 3-11 shows an example of a LAN designed using this rule. It consists of a number of separate Ethernet-based work groups, all interconnected via an FDDI ring. I don't call this an "old-style" design to disparage it. In its day, this was cutting-edge technology. Although I modify the basic rule later in this chapter, the general design concept points out some important principles of network design that are still applicable.

Suppose that the network protocol in this diagram was TCP/IP. The entire campus, then, would have been addressed from the same large address range, such as a Class B or Class A. In fact, because all of these segments were bridged together, there would have been no technical requirement to break down the user segments into their own specific address ranges. The whole campus looked like one gigantic common flat network at the IP layer.

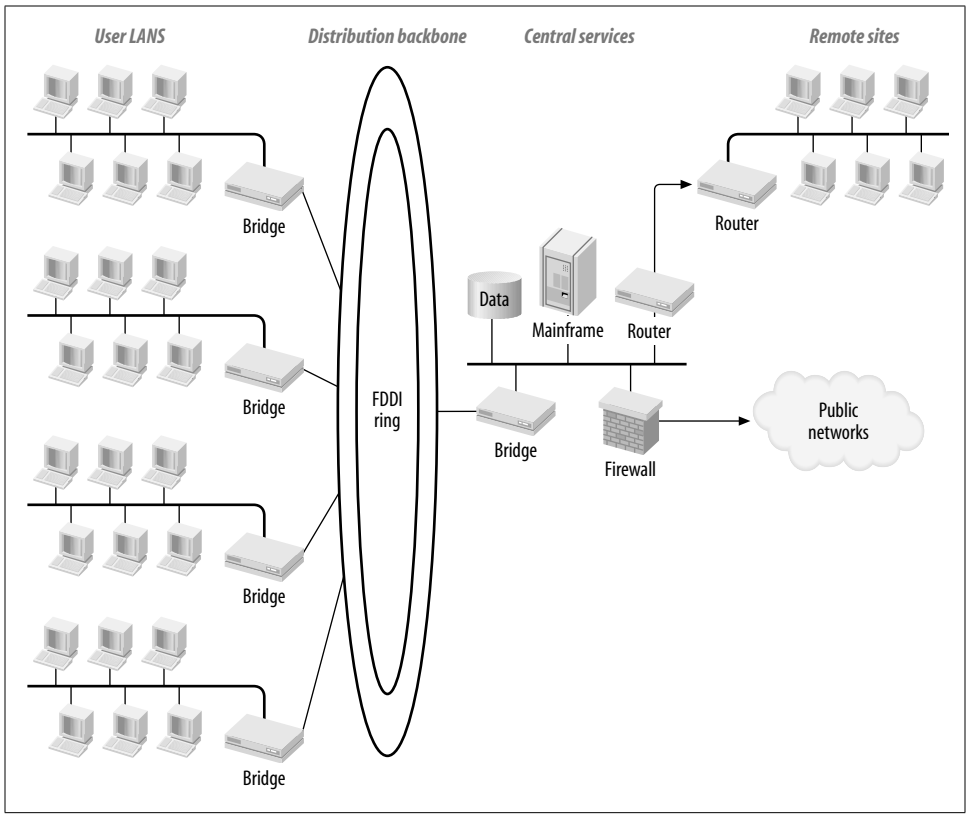

Figure 3-11. Old-style "bridge on campus, route off campus" design

In practice, however, most network administrators would have broken down their larger address range into subranges, and allocated these subranges to different user LAN segments. This allocation would be done purely for administrative reasons and to make troubleshooting easier.

In this old-style design, if someone on one of the user LAN segments wants to access the central database, they first look up the IP address (probably using DNS). They then send out an ARP (Address Resolution Protocol) packet to find the Ethernet MAC address associated with this IP address. This ARP packet goes out through the bridge and onto the FDDI backbone ring. Every other bridge on the ring forwards this packet onto its local segment. Eventually, the packet reaches the database server, which responds appropriately.

This approach immediately points out one of the important limitations of this design principle. Broadcast packets (like the ARP packet in the example) are sent to every distant corner of the network. This may be fine if there is very little broadcast traffic,

but some broadcasts, like ARP, are a Core part of the network protocol. Every station sends broadcasts. There are necessarily limits to how big one can make a bridged network before routine broadcast traffic starts to choke off production application traffic.

This model does a nice job of segregating the regular application traffic, though. Suppose a user on the left side of the picture talks to a server on the right with regular unicast packets. Each packet on both sides of the conversation contains the Ethernet MAC address of the destination device. All bridges are smart enough to keep track of the MAC addresses on each port. So, a packet heading for the database server enters the FDDI ring because the user's local bridge knows to find that MAC via the ring. Then every other bridge on the ring simply leaves the packet alone until it reaches the one that has that MAC address on its LAN segment. Thus, normal application traffic takes an efficient direct route.

Now consider traffic destined for the remote site shown on the far right-hand side of the picture. Two rules of networks are almost immutable. The first is that bandwidth costs money; the second is that distance costs money. From these two rules, it is safe to conclude that high bandwidth over long distances costs a lot of money. Whatever technology is used to connect to the remote site, it almost certainly has much lower bandwidth than any LAN element.

This point is important because the rule was "bridge on campus, route off campus." In other words, it says that you should bridge where bandwidth is cheap and route where it's expensive. Bridging allows all broadcast chatter to go everywhere throughout the bridged area. You simply want to avoid letting this chatter tie up your expensive WAN links. On the LAN, where bandwidth is cheaper, you will want to use the fastest, cheapest, most reliable technology that you can get away with. At least in earlier times, that meant bridging.

A bridge is generally going to be faster than a router because the decisions it makes are much simpler. The manipulations it does to packets as they pass through it are much simpler as well. In the example, these bridges interconnect Ethernet and FDDI segments, so the Layer 2 information in the packets needs to be rewritten. This is a simpler change, though, than what a router needs to do with the same packet.

Modernizing the old rule

This old rule has merit, but it needs to be modernized. It is still a good idea to keep broadcast traffic off of the WAN, for exactly the same reasons that it was important 10 to 15 years ago. However, two current trends in networking are leading network designers away from universally bridging throughout a campus. First, many more devices are being connected to the network than there ever were in the past. Second, certain changes in network technology have changed the way things scale.

Let me explain what I mean by this second point. In the old-style network of Figure 3-11, user workstations were connected to shared 10Mbps Ethernet segments. All segments were interconnected via a 100Mbps FDDI ring. If you have a

dozen active devices sharing a 10Mbps Ethernet segment, the collision overhead limits the total throughput on the segment to somewhere between 3 and 5Mbps in practice. So each of these dozen devices can use a steady state bandwidth of a few hundred kbps and a burst capacity of a fewMbps.

Today it is common to connect end devices directly to 100Mbps Fast Ethernet switch ports, and backbone speeds are several Gbps. Thus, each station has access to a steady state bandwidth of 100Mbps sending and receiving simultaneously. Each station is therefore able to use 200Mbps of backbone capacity, with the lack of local contention increasing the tendency for routine traffic to burst from very low to very high instantaneous loads. This is almost a factor of 1000 higher than in the older style of network, but our backbone speed has only increased by a factor of between 10 and 100.

In other words, each station is now able to make a much larger impact on the functioning of the network as a whole. This is why traffic prioritization and shaping (flattening out the bursts) have become so much more critical in network design. If more cars are on the road, there is a limit to how much the flow rate can be improved by just increasing the number of lanes. New methods of traffic control are needed as well.

Hierarchical Design

What's really valuable about the old-style design shown in Figure 3-11 is that it leads to the useful and practical concept of hierarchical network design. Figure 3-12 and 3-13 show what a hierarchical network design is and how it works. At this point, however, whether this network is basically bridged or routed is still questionable.

Figure 3-12 is a conceptual drawing of the hierarchical design model. There are three main levels, the Core, Distribution, and Access. These terms are widely used. End stations are connected at the Access level. You will sometimes see a drawing like this in which central servers are connected at the Core. If end node devices are connected at the Core, then the model is not strictly hierarchical. It may be some sort of hybrid. Or, more likely, the diagram could be an application-oriented diagram rather than a network diagram.

Figure 3-12 shows how connections are made. End devices connect at the outside edges of the diagram. They are connected to the Access Level of the network. This level exists primarily to give a place for these end devices to connect to the network. At the center of the diagram is the Core Level, which performs the main traffic switching functions, directing packets from one part of the network to another. The Distribution Level exists to connect the Access and Core Levels.

The name "Distribution Level" is appropriate for a couple of reasons. First, this level is what allows the network to spread out the distributed backbone. Second, the Distribution Level distributes data from the Core out to the Access Levels of the network.

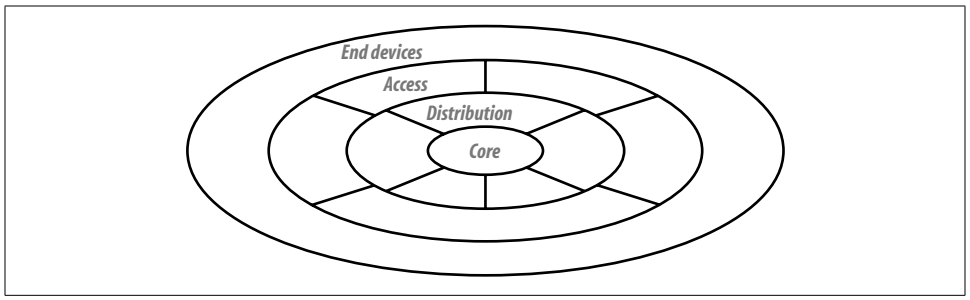

Figure 3-12. Hierarchical network-design concept

The basic idea is to separate the different functions of the network and hopefully make them more efficient. What does a network do? It directs traffic (Core), it conveys packets from one place to another (Distribution), and it provides connection points for end devices (Access). In a small network these functions could all be performed in one box, or even a simple piece of wire. But the larger the network, the more these component functions have to be separated for efficiency.

There are usually important cost advantages to using a hierarchical model. For example, the Access Level needs to give a high port density with a low cost per port. At the Core Level, it is more important to have high throughput devices with a few high-speed ports. Expecting one type of device to fill both of these categories isn't always reasonable.

Figure 3-13 shows a more specific example of how to think about hierarchical design models. In the middle is the Distribution Level, which carries traffic between the various Access groups and the Core. Two new ideas here were not shown in Figure 3-12. The first is the addition of some redundancy; the second is the implication that not all traffic needs to cross the Core.

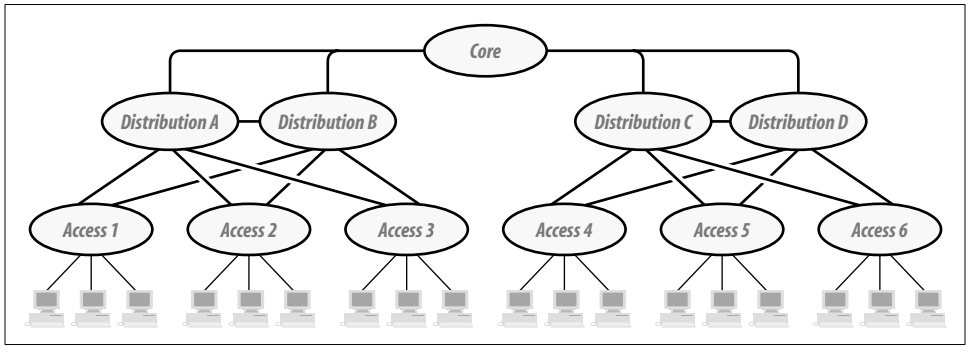

Figure 3-13. Hierarchical network-design model

Each of the Access Level devices is connected to two devices at the Distribution Level. This connection immediately improves the network throughput and reliability. Doing this has effectively eliminated the Distribution Level devices as single

points of failure. For example, if Distribution "cloud" A broke then all three of the Access groups using it can switch over to Distribution "cloud" B transparently.

I am deliberately leaving the contents of these clouds vague for the moment. Notice that I have included a connection between Distribution clouds A and B so that the Core connection for either can break and traffic will simply shift over to the other.

Now consider the traffic patterns. Suppose an end device connected to Access cloud 1 wants to talk to another end device in the same cloud. There is no need for the packets to even reach the Distribution Level. Similarly, if that same device wants to talk to an end node connected to Access cloud 2, it doesn't need to use the Core. The packets just go through Distribution clouds A and B to get from Access cloud 1 to Access cloud 2. It needs to cross the Core only when the packet needs to go further afield, to another Access cloud that is not connected to the same Distribution cloud.

This principle is important because, if used carefully, it can drastically reduce the amount of traffic that needs to cross the Core. Because everybody shares the Core, the design principle needs to be used as efficiently as possible.

Recall the 80/20 rule that I mentioned earlier in this chapter. This rule is particularly applicable to the Distribution Level. If certain groups of users tend to use the same resources, then it makes sense to group them together with these resources. It's best to group into the same VLAN. But putting them into the same Distribution groups also saves traffic through the Core. In most large companies, separate business divisions have their own applications and their own servers. Try to consider these relationships when deciding how to divide your Distribution and Access groups.

To look more deeply into the various clouds shown in Figure 3-13, I need to first tell you where the network routes and where it uses bridging or switching.

Routing Strategies

Relaxing the "bridge on campus, route off campus" rule opens up the question of where to use routers. Designers could use them at every level of the LAN, including the Access Level, if they wanted to. Or, they could use them just at the Distribution Level and use switches in the Core and Access Levels. How do they decide what's right?

Well, you need to start by remembering what routers do. A router is a device that connects two or more different Layer 3 addressing regions. So, by the same token, routers break up Layer 2 broadcast domains. A router is also a convenient place to implement filtering, since it has to look much further into the packet than a switch does.

There are also negative aspects of routers. Every packet passing through a router has to be examined in much more detail than the same packet passing through a switch. The Layer 2 MAC addresses and framing have to be rewritten for every packet. Thus, latency through a router is necessarily going to be higher than through a switch.

Furthermore, Layer 3 dynamic routing protocols such as OSPF, RIP, and EIGRP must all be considered every time a router is installed. The designer has to ensure that the dynamic routing protocol will be stable and will converge quickly and accurately whenever the state of a network connection changes. The more routers in a network, the more difficult this process becomes.

Because of these negative aspects of routing, I would happily bridge the whole LAN if I could get away with it, but I've already discussed the inherent problems in this strategy. What else can be done?

When the requirements include filtering for security, the answer is easy; use a router. If a sensitive part of the network needs to be separated from the rest (for example, the Payroll Department or the Corporate Finance Department of a brokerage company), the designer should make sure that it's behind a router.

For the rest of the network, the routers are used only for breaking up broadcast domains. The improved congestion control properties from installing routers have to be balanced against the extra latency that they introduce. At the same time, you have to be careful of how you implement your dynamic routing protocols.

One-armed routers and Layer 3 switches

One way of implementing a router into the Core of a network is to use a so-called *one-armed router*. This picturesque term refers to a router that connects to several logical networks via a single physical interface. One clever modern way of accomplishing this feat is by making the router a card in a Layer 2 switch. This card, called a *Layer 3 switch*, then makes a single physical connection to the shared backplane of the switch. This backplane is generally an extremely high-speed proprietary medium. Attaching the router directly to it resolves several problems simultaneously.

First, you don't need to pay a huge amount of money to install a super high-speed network media module in the switch just to run the connection out to an external router. Instead, you can bring the router directly to the backplane of the switch. Second, the high bandwidth available on the backplane drastically reduces congestion problems that often plague one-armed router constructions. Third, because the Layer 3 switch module only has to form packets for the proprietary backplane of the switch, it is able to drastically reduce overhead required when routing between different media types. It only needs to know one Layer 2 protocol, which is the proprietary protocol used internally on the backplane.

It is possible to make a one-armed router act as a Layer 3 switch and achieve many of the same benefits. The single port on the router can be configured to support several VLANs, looking like a trunk connection to the switch. If this router-to-switch connection is sufficiently fast, such as a Gigabit or ATM link, then it is almost the same as a Layer 3 switch. Specifically, it has the benefit of being able to flip packets between different VLANs all using the same Layer 2 protocol.

This construction can be a useful way of getting the benefits of a Layer 3 switch when using equipment that doesn't support integrated Layer 3 switching, or for which the performance of these switches is poor. However, I would expect to see better performance from an integrated Layer 3 switch that is able to access the higher capacity backplane directly.

Using a construction in which several different Layer 3 networks converge on a single point makes sense. In a network like the one in Figure 3-11, putting a one-armed router on the FDDI backbone would have been fairly common. Then the various bridged Ethernet segments shown could be on different IP subnets. The FDDI interface on the router would also have an address from each of the various subnets. Although this scenerio was not uncommon, there are several deficiencies in a network built this way.

Network designers put routers into networks to separate broadcast domains. If they are just going to bridge everything together and have a single one-armed router in the middle, then they haven't separated the broadcast domains. Furthermore, they've made the network one step worse because they have introduced a new, artificial single point of failure for the entire network.

The same criticism is not necessarily true for Layer 3 switches, though. If the network consists of many VLANs, then the trunks between the switches ensure that all VLANs are visible on the backplane of the switch. Thus, the Layer 3 switch will not only route, but will also flip the packets between the various VLANs. This step can be done very efficiently, and the problem of failure to segregate the broadcast domains largely disappears (however, as I will discuss later in this chapter, it is possible to make bad design decisions for the VLAN structure that will negate this advantage).

The question remains, where should you use these sorts of devices? One obvious answer is the Core Level of the network. At the Core you have the greatest need for speed, and the greatest potential number of converging VLANs. But this second point is only true if you have no (or few) routers in the Access and Distribution Levels of the network. Figure 3-14 shows a hierarchical LAN design in which all VLANs converge on the Core of the network. In the two Core switches at the center, a pair of Layer 3 switches handles all routing for the network. Everything is redundant at the Core and Distribution Levels.

In this picture, there are four Access switches for each pair of Distribution switches. A total of four user LANs converge on the Core switches from above, and another four converge from below. Now the designer has to make important decisions about how to handle the VLAN trunks, which affect how routing is handled in the Core. There are many options. One option is to simply make everything one large VLAN, in which case there is no need to route anything. Or, one could make several small VLANs, all of which are visible everywhere in the network. Once again, this means that there is very little advantage to having the routers because all (bridged) VLANs must send their local traffic through the entire network anyway.

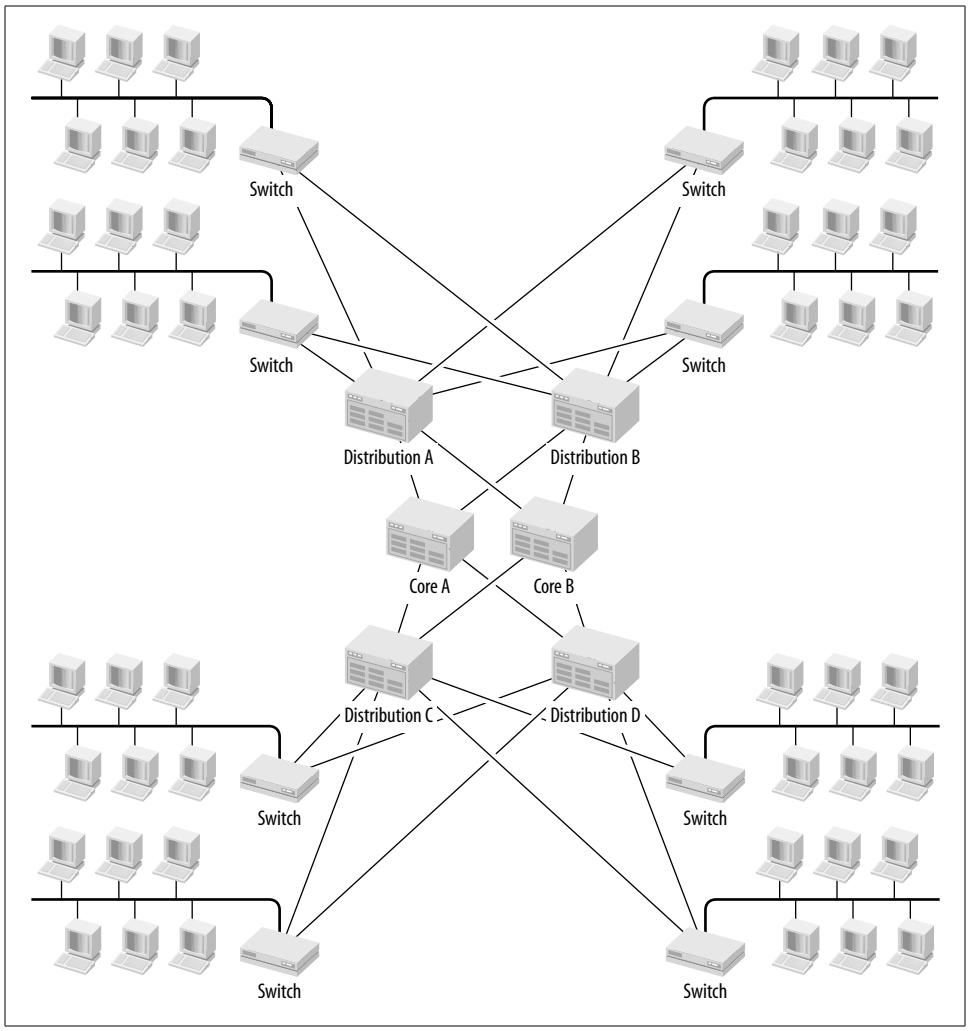

Figure 3-14. Hierarchical LAN with central routing

Always bear in mind that one of the key points in putting in routers is to limit the chances of broadcast traffic from one part of the network causing congestion someplace else. A VLAN is a broadcast domain, so you might think that making lots of VLANs results in small broadcast domains and eliminates your broadcast problems. This is only partially true, however. Remember that each trunk probably contains several VLANs. If an Access Level trunk circuit holds all VLANs for the entire network, it has to carry all of the broadcast packets. The effect is the same as if you had done no VLAN segregation at all, only with more inherent latency.

In Figure 3-14, I assume that I have been able to reduce the traffic so that the upper two Distribution switches carry completely different VLANs than the lower two. The

only way to get between them is through the Layer 3 switches contained in the two Core switches. These Layer 3 switches also have to handle the inter-VLAN routing within each of these two groups of VLANs. Figure 3-15 shows the same picture at the Network Layer. In this case, it is easy to see the pivotal role played by the Layer 3 switch. For symmetry, I have shown four VLANs for both the upper and lower pair of Distribution switches (see Figure 3-14). However, as I will discuss later in this chapter, there is no need for the VLANs to correspond to the physical Access switches.

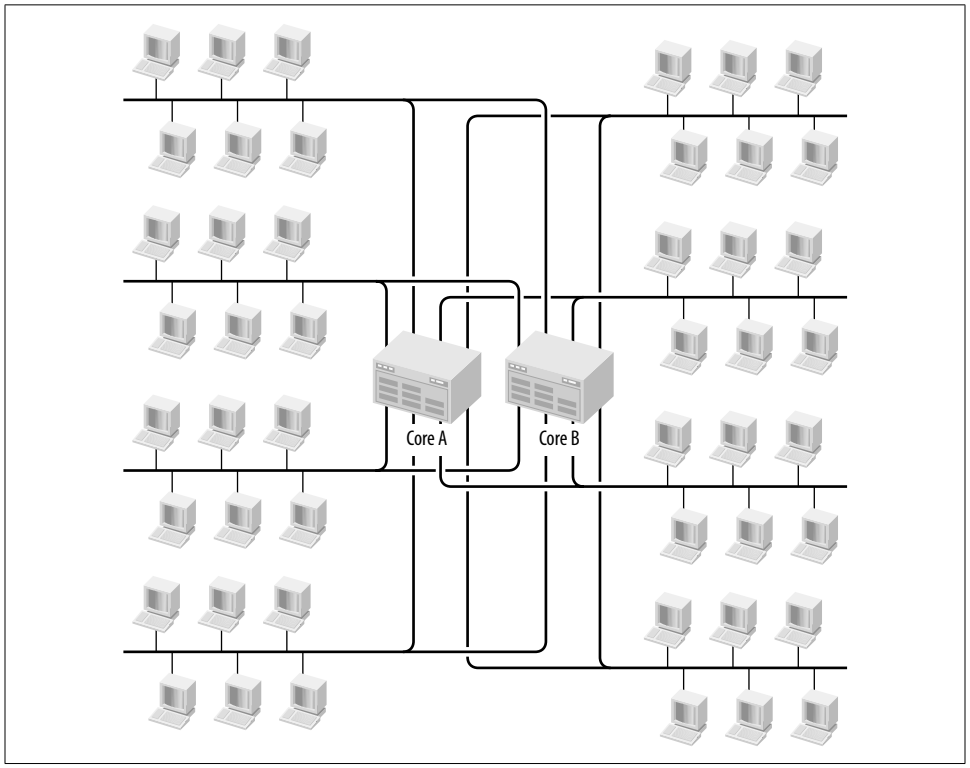

Figure 3-15. Hierarchical LAN with central routing—network-level view

The important thing to note from Figure 3-15 is that a total of eight VLANs converge on the redundant pair of Layer 3 switches. It is not possible for traffic to cross from any VLAN to another without passing through one of them. Obviously, redundancy is an important concern, as I will discuss in a moment. But there's another important feature. Because all off-segment traffic for each segment must pass through these devices, they tend to become serious network bottlenecks if congestion is not controlled carefully. The fact that they are connected directly to the backplane of the two Core switches starts to look like a necessity. This relatively small example collapses eight separate full-duplex 100Mbps feeds from the various Access switches.

Redundancy

Another key feature shown in Figure 3-14 is redundancy. The two Core switches are completely redundant. If all of the traffic aggregates onto a single router that handles the whole enterprise, then that's one colossal single point of failure. With some manufacturers, you have the option of putting a redundant Layer 3 switch module in the same chassis. This option is certainly an improvement, as I showed in the previous chapter. It's still necessary to do all of the MTBF calculations to figure out how much of an improvement it gives, though, and to show how the result compares with having a completely separate chassis plugged into different power circuits.

Unfortunately, I can't do this calculation for every possible switch type because vendors implement new switches with awe-inspiring regularity. You need to watch out for devices with which the failure of some other component, such as a controller module, affects functioning of the Layer 3 switch module or its redundancy options. Every switch seems to do these things differently.

The bottom line is that, to achieve good redundancy with a single chassis device, there can be no single points of failure within the device. Typical MTBF values for the chassis of most switches is sufficiently long not to be a serious concern. If you are going to implement a single-chassis solution, however, it has to have redundant Layer 3 switch modules, redundant power (*N*+1 redundancy is usually sufficient), and redundant connections to all Distribution Level devices. It may also require redundant CPU modules, but in some designs the CPU module is used only for reconfiguring otherwise autonomous media modules. Be careful, though, because such a design might mean that redundancy of Layer 3 switch modules will not work in the event of a CPU module failure. In this case, the net MTBF needs to be calculated. Even in this case, I am talking about a multiple failure situation (CPU module plus one of the Layer 3 switch modules), for which the aggregate MTBF should still be quite high.

The other solution, which is conceptually simpler, is to use two separate chassis, as shown in Figure 3-14 and Figure 3-15. However, in this case you have to use a router redundancy protocol to allow one of these devices to take over for the other. In the most common (and most stable) configuration, end devices send all of their off-segment traffic to a default gateway. In the one-armed router model, this default gateway is the same physical device for all of the segments. To make the module in the second switch chassis become the active default gateway for all segments, it has to somehow adopt the same IP address.

This adoption is most easily accomplished by means of either the Cisco proprietary HSRP protocol or the open standard VRRP protocol.

Router-to-router segments. When two or more routers or Layer 3 switches function between the same set of LAN segments, it is common to implement an additional segment just for the routers. This construction is shown in Figure 3-16.

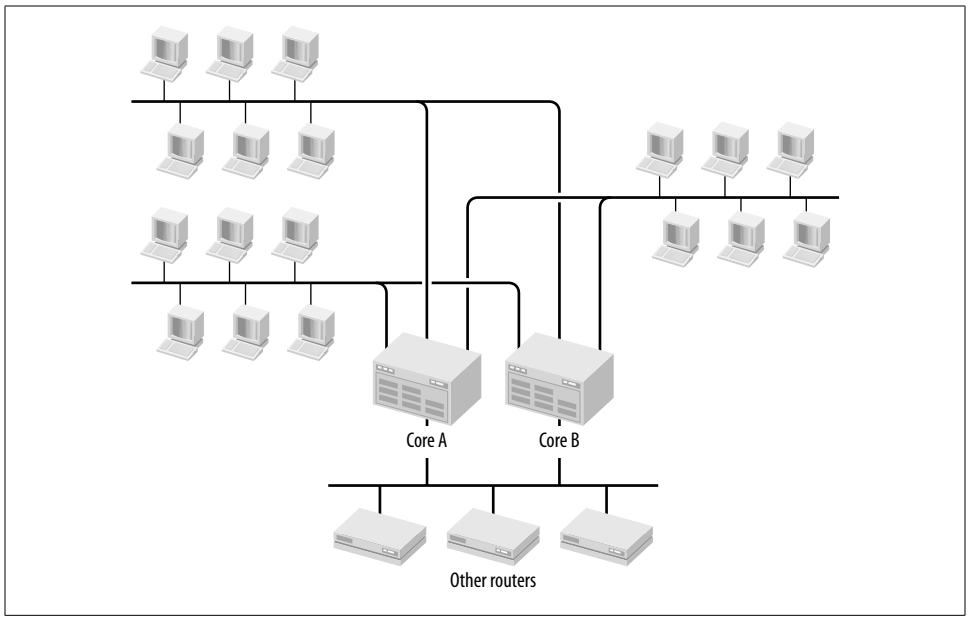

Figure 3-16. Implementation of a router-to-router segment

The diagram shows three user LAN segments connecting to the two routers called Core A and Core B. These two routers also interconnect using the special router-to-router segment. If other routers are in this location, then they would be connected here as well. These other devices might be WAN routers or other special function routers like tunnel termination points for X.25 or SDLC sessions, for example.

The router-to-router segment serves two main purposes. First, and most obvious, if special function routers are at this location, it provides a place to connect them where they will not suffer interference from user LAN traffic. In many cases, these legacy protocols exist because the applications using them are extremely critical to business. Routine word processing and low priority LAN traffic should not be allowed to disrupt the more important tunneled traffic, so it needs to go on its own segment. And, conversely, end-user devices don't need to see dynamic routing traffic.

To see the second reason, however, suppose that these other devices are not present and the network consists of the two Core routers and three user segments. Now suppose the first Ethernet connection on Core A breaks. HSRP or VRRP kicks in promptly, making Core B the default gateway for this segment. Core A is still the default gateway for the other two segments, though. Now consider the flow of traffic between Segment 1 and Segment 2.

A user on Segment 1 sends a packet to its default gateway, Core B. Core B forwards this packet out its port for Segment 2 and the user on this segment receives it. The response, however, takes a very different route. This packet goes to the default

gateway for Segment 2, which is Core A, but Core A doesn't have an active port on Segment 1 because it's broken. It has to somehow send this packet over to Core B. I'll presume for the moment that there is a good dynamic routing protocol, so the two routers know how to get to one another and know which ports are functioning properly.

Core A sends the packet through one of the user LAN segments over to the Core B router. From there, it is sent out and received by the right user. So, there are two possibilities in this case. Either the packet was forwarded back out on Segment 2 to get over to the other router, or it was sent across on segment three. If it went via Segment 2, then that packet had to appear on this LAN segment twice, which could have a serious affect on overall congestion. If it went via segment three, then it potentially causes congestion on a completely unrelated user segment where it has no business being. This could be a security issue, but it is more likely just a congestion problem.

The easiest way around this sort of problem is to implement a special router-to-router segment. The routing protocols must then be carefully adjusted so that this segment is always preferred whenever one router needs to access the other.

Some network designers consider this problem aesthetic and ignore it. If all router ports are connected to high-bandwidth full-duplex switch ports, then the problem is much less dangerous. Another thing to remember is how VLAN trunks might be loaded in failure situations. For example, if the router-to-router segment is carried in the same physical trunk as the user segments, then it doesn't prevent congestion.

Physical diversity. As long as I'm talking about router redundancy, I need to mention a special side topic because it can be quite dangerous. On the surface it sounds like putting those two routers in different physical locations would be a good idea. For example, they might be in different rooms, on different floors, or even in different buildings. This arrangement could save the network in the case of a fire or large-scale power problem. But it could also make some simpler types of problems much worse.

To see why, look at Figure 3-16 and suppose that the left half of the picture including router Core A and two user segments are all in one building and everything else is in another building. Now, suppose that you have a relatively simple and common problem—a fiber cut between the two buildings. I'll go one step further and assume that both routers have some other form of connection back to a central network. Perhaps this is actually part of the Distribution level of a larger network, for example. The problem still exists without this added twist, but I think this example makes it a little easier to see it.

When the fiber was cut, VRRP or HSRP kicked in and made sure that all three segments still have a default gateway, so all inbound traffic from the user LAN segments will be delivered properly. The problem is with the return path. Look at the

ports for LAN segment number 1. Both routers Core A and Core B have valid connections to this segment, but only one of them actually contains the particular user expecting this packet. Which one is right?

In many cases, if the central router has two paths available with the same cost, it just alternates packets between the two. The first one gets to the correct destination. The second one goes to the other router—the one that has a valid connection to the segment that has the right IP address but just doesn't have this user on it because the connection between the two sides is broken. So the packet is just tossed off into the ether and lost forever.

Different routers implement this in different ways. For example, some routers work based on *flows*. A flow is a single session. This concept is important to Quality of Service, so it is discussed in detail in Chapter 8. In this case, the router handles each flow separately, routing all packets belonging to a particular session through the same path.

This just means that some sessions will work and others will try to follow the path that is broken. Also, for applications that do not use a Layer 4 connection, such as those built using UDP, it is not possible to divide applications into unique flows. In these cases, some of the packets will be randomly lost.

This will happen for all of the user segments. So a measure that was intended to give better reliability in a rare failure mode has actually reduced the reliability in a more common failure mode.

If you really want to use physical diversity in this way, it has to be combined with path redundancy. Instead of running all of your LAN segments through the same fiber conduit so they could all break together, you could have another fiber conduit. In this second conduit, you would run redundant connections for all segments. Then, to complete the picture, you would use Layer 2 switches with Spanning Tree to switch to the backup fiber in case the primary breaks.

Figure 3-17 shows how this concept might work. In this figure, I've only drawn one of the segments for simplicity. The thick dashed lines represent the backup fiber pairs, which go through the second conduit. For symmetry, I've also included a backup connection from Switch A to the user segment, even though this segment is within the same building. The connection between Switch A and Switch B is required for Spanning Tree to work properly, as I discussed earlier in this chapter.

The Core A and Core B routers are assumed to be directly connected to their respective switches, so you don't need to worry about extra redundancy in these connections. Spanning Tree is configured on Switches A and B so that when the primary fiber stops working, the secondary one is automatically engaged. The same procedure would be followed on all other segments, including the router-to-router segment, if applicable.

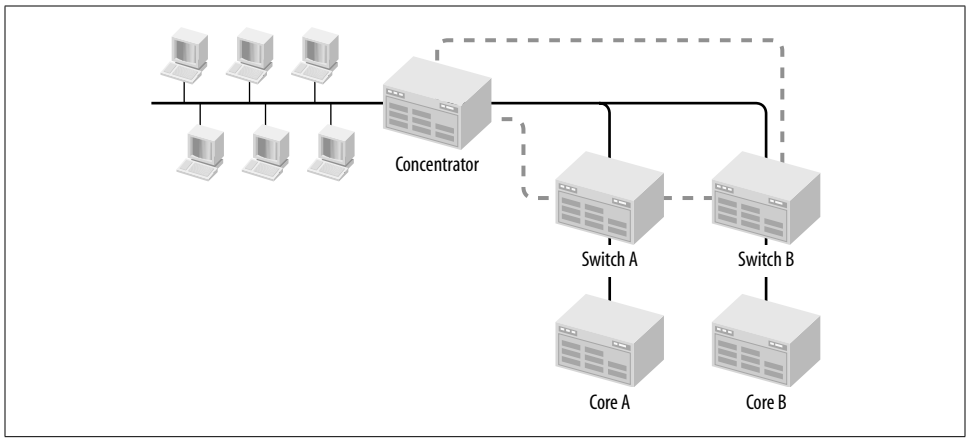

Figure 3-17. Physical diversity the safe way

In this picture the local floor connection is shown as a concentrator. The actual technology is irrelevant, however. It could be a hub, or a switch, or even a piece of 10Base2 cable connected to the fiber pairs by means of transceivers.

Filtering

There are three reasons why you might want to implement filtering on a router:

- Security
- Clean up for ill-behaved applications
- Policy-based routing

If you really want hard security on an IP network, you should probably be looking at a proper firewall rather than a router. But, in many cases, you just want a little security. In an IPX network, a router may be the only practical option for implementing security precautions.

You can do several different types of security-based filtering on a router:

- Filtering based on source or destination IP address
- Filtering based on UDP or TCP port number
- Filtering based on who started the session
- Filtering based on full IPX address or the external network number

The decision about which combination of these different filters to use depends on what you're trying to accomplish. So, I want to look at some different examples and see how different filter rules might apply.

Filtering for security. It is fairly common, particularly in financial companies, to have an external information vendor such as a news or stock quote service. The vendor's service involves putting a box on the client's internal LAN to allow them to access

real-time information. The security problem is obvious: the external vendor theoretically has full access to the client LAN. Since financial companies usually have strict rules about access to their internal networks, they need to provide a mechanism that allows the information vendor's box to see only the genuine application data that it is supposed to see.

Assume that the vendor's special application server is hidden on a special segment behind a router. Now what sorts of filters can be implemented on this router?

The first type of filter, based on source or destination address, is probably not going to be useful here. There could be many internal users of this service, and you don't want to have to rewrite your filter rules every time somebody new wants access. It doesn't do any good to filter based on the address of the server because that's the only device on the special segment anyway.

The second type of filter, based on TCP or UDP port number, on the other hand, should be quite useful here. Since the application probably uses a designated port number (or perhaps a range), this could be a good way to identify the application packets.

The third type of filter is only useful if the application is TCP-based. If it is UDP-based, then the router cannot discern a session, so it can't tell who started the conversation. If it is TCP-based, and if the application starts with the user logging in (which is common), then this filter will help you to prevent the vendor's box from being used to initiate an attack on the client LAN.

What you really want is to combine the second and third filter types. You can do this on a Cisco router just adding the "established" keyword to an Access list for the required TCP port number.

The other example concerns the IPX filter. It's fairly common to have a special Novell server for sensitive data like personnel and payroll records, or other secret information. The payroll server makes a good example. The company might have this server on the Human Resources segment and use standard Novell authentication systems to ensure that only authorized people can see secret files.

But the organization may be concerned that these measures are not sufficient to prevent people from trying to give themselves a special pay bonus. To help prevent this, you can keep this server on a special segment and configure the router to disallow any access from off-segment. The trouble is that members of the Human Resources staff still need to get to the other corporate Novell servers. The CEO or other high-ranking corporate officials that it is supposed to seemight need access to the Human Resources server. So you can build a special filter that allows only the CEO's full IPX address (which includes the workstation's MAC address) to connect to the full IPX network number (including internal and external network numbers) of the server. Then you can allow all other internal network numbers to leave the segment. Consult your router vendor's documentation for information about constructing IPX filters.

Filtering for application control. Some applications do not behave in a friendly manner on a large network. An application might try to do any number of unfriendly things. For example, it might try to register with a server on the Internet. Or, it might send out SNMP packets to try and figure out the topology of the network. Sometimes a server tries to probe the client to see what other applications or protocols it supports. From there, the list branches out to the truly bizarre forms of bad behavior that I'd rather not list for fear of giving somebody ideas.

The trouble with most of these forms of bad behavior is that, if you have several hundred workstations all connecting simultaneously, it can cause a lot of irrelevant chatter on your network. If you don't have the spare capacity, this chatter can be dangerous. The SNMP example is particularly bad because a number of applications seem to think that they should have the right to poll every router on the network. In general, you don't want your servers to know or care what the underlying network structure looks like. It can actually become a dangerous problem because SNMP queries on network gear often use excessive CPU and memory resources on the devices. If several servers try to gather the same information at the same time, it can seriously hamper network performance. I have seen this problem cripple the Core of a mission-critical network during the start-of-day peak.

If you suspect that you have a problem like this, you need to use a protocol analyzer to get a good picture of what the unwanted information looks like. You also need to prove experimentally that this information is really unwanted. Some applications may just work in mysterious ways.

Once you have established what the unwanted data looks like and where it's coming from, then you can start to filter it out. Usually, it's best to put the filters close to the offending server (hopefully it's the server and not the client that is to blame) to help contain the unwanted traffic.

Policy-based routing. Policy-based routing is a Cisco term. Some other vendors' routers have similar capabilities, but I have to admit I learned this stuff first while using Cisco gear, so I still think in Cisco terms. This term means that the router is able to make routing or prioritization decisions based on whether a particular packet matches predefined characteristics. Perhaps it is a source or destination IP address, or perhaps a TCP, a UDP port number, or a packet size. By whatever mechanism, you define rules for what happens when the router receives packets of this type.

The rule may specify that you tag the packet with a special priority code so that every other device in the network will know that this packet is important and will forward it first (or last, or whatever). Or, the rule may be that certain types of packets use the high-speed trunk, while others use the low-speed trunk.

This last case, in which a routing decision is made based on the policy, is what gives the concept its name. It warrants special comment, though. In general, it is extremely dangerous to do this kind of thing for three reasons. First, it can interfere with redundancy mechanisms. Second, it makes troubleshooting unnecessarily difficult. (The

low-priority ping packet gets through, but the application doesn't work. Is it the server or the high-priority trunk that's down?) Third, it has a nasty tendency to run up the CPU on your router (although this tendency is less likely in IOS Version 12 and higher because of support for FastSwitching of policy-based routing). Yes, it will work, but it's an extremely bad idea in most real world networks. Having said this, however, using the same feature to tag packets for priority works extremely well.

One final comment on filtering on a router: it's important to watch your CPU utilization. Modern routers tend to try to offload most routing decisions onto hardware associated with the port itself, so most packets never have to hit the CPU. This situation results in much faster and more efficient routers. But, depending on the router and the specific type of filter you are implementing, you may be forcing a lot of the processing back to the CPU. The result could be that your powerful expensive router is no longer able to handle even modest traffic volumes. So, when implementing filters, always take care to understand what it will do to the processing flow through the router. Often the best way to do this is simply to mock up the change in a lab and see what happens to your CPU statistics.

Switching and Bridging Strategies

In a LAN, every connection that isn't routed must be either bridged or repeated. I won't discuss repeaters much in this book. In modern LAN technology, there is rarely a good reason to use them. In nearly all cases, a switch is a better choice, both for cost and functionality. For that matter, conventional bridges are also increasingly rare, having been replaced by switches.

Of course, these comments are mostly semantics. People still use hubs. And what is a hub but a multi-port repeater? People still use switches, which are really multi-port bridges.

If you are dealing with a portion of a LAN that is all logically connected at Layer 3, then you have two main choices for our Layer 2. You can use a hub or a switch. This is true regardless of whether the LAN technology used at Layer 2 is Ethernet, Fast Ethernet, or Token Ring. It is also true for Gigabit Ethernet, although in this case I question the usefulness of Gigabit Ethernet hubs, preferring switches in all cases. Fortunately, it appears that the market agrees with me, as I am not aware of any major network hardware vendor who has implemented the hub part of the Gigabit Ethernet specification.

So I'll start by discussing where to use hubs and where to use switches in an Ethernet or Token Ring environment.

Switches have three main advantages over hubs:

- Higher throughput
- The ability to communicate at full-duplex (Ethernet)
- Better control over multicast traffic

There are two disadvantages to weigh against these advantages:

- Switches are more expensive
- It is much easier to use diagnostic tools such as protocol analyzers on a hub than a switch

A hub (sometimes called Media Attachment Unit [MAU] in Token Ring literature) is basically a way of sharing the network's Layer 2 medium. This sharing necessarily has overhead. In Ethernet, the overhead comes in the form of collisions. In Token Ring, it appears as token passing latency. In both cases, the system for deciding who gets to speak next takes a toll.

If you replace the hub with a switch instead, then this overhead essentially disappears. There are only two devices on the segment (or ring)—the end device and the switch itself. If it is a Token Ring switch, then every end device gets, in effect, its own token. There is never any waiting for the token, so each device can use the entire 16Mbps capacity of the ring.

If it is an Ethernet switch, on the other hand, the only times you should expect to see collisions are when both the switch and the end device try to talk at once. Even this small collision rate can be eliminated if you go to full-duplex Ethernet. On a large shared Ethernet segment, you can only practically achieve between 30% and 50% of the capacity because of the collision overhead. On a half-duplex switch this jumps well over 90% of capacity for every device and 100% for full-duplex switching. Thus, the net throughput of a switch is considerably higher than a hub with the same number of ports, for both Token Ring and Ethernet.

Most Fast Ethernet and many Token Ring switches can operate in a full-duplex mode. This means that they can send and receive simultaneously without collisions. Obviously this mode only works when a single end device is attached to each switch port. You can't have a full-duplex connection to a hub. Using a full-duplex switch has the effect of theoretically more than doubling the throughput to each device. It more than doubles because a half-duplex port still loses some capacity due to collisions. This advantage is most significant on servers, where it is not unusual to have a high volume of traffic both sending and receiving.

Containing broadcasts

Broadcasts are an integral part of many network protocols including TCP/IP and IPX. However, having too many broadcasts on a network can cause serious problems. The most obvious problem is simply bandwidth utilization. However, it is important to remember that broadcasts are delivered to every end device. Because these broadcast packets are addressed generically, the network interface cards of these end devices cannot tell whether they are important. So they are all passed up the protocol stack to be examined by the main CPU of the end device. Having a lot of broadcasts on a LAN segment can cause CPU loading problems on end devices, even when they are not actively using the network. Thus, broadcasts must be controlled.

A bridge or switch is supposed to forward broadcasts. This is, in fact, one of the most fundamental differences between bridging and routing. Forwarding broadcasts allows devices that are part of the same Layer 3 network to communicate easily. All global information on the network is shared.

A hub can't stop a broadcast without breaking the Layer 2 protocol. Those broadcast packets have to circulate, and stopping one would also throw a wrench into the congestion control mechanism (token passing or collisions). A switch or bridge, however, can choose which packets it forwards.

Normally, the way a switch or bridge makes this decision is by looking at its MAC address table. If the packet has a destination MAC address that the switch knows is on a particular port, then it sends the packet out that port. If the packet has an unknown destination address or if it has a broadcast or multicast destination address, then the switch needs to send it out to every port.

If the network is very large, then the number of packets that need to go out every port can become a problem. Usually, in most networks, the broadcast volume is a relatively small fraction of the total number of packets. Pathological conditions called "broadcast storms" (see the discussion in the previous chapter) can make this broadcast volume suddenly high, though. If these conditions occur frequently, then serious performance problems may occur on the network.

Controlling broadcasts is one of the main reasons why network designers have historically gone from bridged to routed networks. With many modern switches, it is possible to push this decision further because of broadcast control mechanisms available on these devices. Usually, the broadcast control mechanism works by simply monitoring how frequently broadcast packets are seen on a port or on the switch as a whole. When the broadcast volume rises above this high-water mark, the switch starts to throw away broadcast packets.

Clearly, this threshold level has to be high enough that the network rarely loses an important broadcast packet (such as an ARP packet). It also has to be low enough so it doesn't interfere with the normal functioning of the network.

This way of treating broadcast storms is reasonably effective. It doesn't prevent them, of course; there will still be storms of packets. But this kind of simple measure ensures that they don't represent a serious traffic performance problem on the network.

There is an interesting trade-off in the place where the decision is made to start throwing away packets. If the decision is made on a whole switch that happens to be in a broadcast-heavy network, then throttling for broadcast storms can actually interfere with normal network operation. On the other hand, just looking at the per-port broadcast volumes ignores the possibility that the storm has been caused by the interaction between several different devices.

One of the most difficult types of broadcast storms to control starts with a single device sending out a broadcast packet. Then one or more other devices on the network receive this packet and respond to it by either sending out a new broadcast (such as an ARP for the originator's IP address) or forwarding the original broadcast back onto the network. A good example is the old RWHO protocol, which broadcasts periodically.

Some IP stack implementations like to send an ARP packet in response to a broadcast packet from an unknown source. This way, they are able to keep a more complete ARP cache. A large number of different devices that respond like this simultaneously, can choke the network for an instant. RWHO is still run on many network print servers by default for historical reasons (although I will never understand why it is still needed). This problem is actually rather common, and it can be extremely serious if the timeout in the ARP cache is shorter than the interval between RWHO broadcasts.

In this case, the per-port monitoring is not effective at stopping the storm. The storm originates with a single broadcast packet, which is the one that really should be stopped, but it is the response that causes the problem, and that response comes from everywhere.

The moral of this story is that just because you implement broadcast storm controls on your switches doesn't mean that you won't have broadcast storms. However, if you have such controls in place, you will be able to prevent this storm from migrating to another switch. The second switch will see an incoming storm on its trunk port and will block it. The problem is at least partially contained.

Redundancy in bridged networks

Redundancy in bridged networks is important for exactly the same reasons as in routed networks. The only differences are in the methods and protocols for redundancy. Just as in the router case, the first step is to install a second switch that is capable of taking over if the first fails. Thus, it needs an automatic mechanism for this to work effectively.

The most commonly employed fault recovery mechanism in bridged networks is the Spanning Tree protocol. The other type of fault recovery system that I mentioned earlier in the case of trunks is a multiplexed arrangement of individual connections. That type of system works well for trunks, but is very difficult to use to make the switches themselves redundant. It is difficult because the individual connection lines must connect between two specific endpoints. If you have a Distribution level switch connecting to a Core switch, you can use this type of system.

For good redundancy, you should have the Distribution switches connected to two Core switches. If the multiplexed bundle of links is split between two switches, then the packets can be sent in two different ways. Some trunk mechanisms treat the

bundle in parallel and break up each packet into small fragments, which are each sent through different links and reassembled at the other side. Other multilink solutions, such as Cisco's Fast EtherChannel, ensure that each packet is sent through a single link intact. In this case, the extra capacity is achieved by distributing packets among the various links in the bundle.

In any case, splitting one bundle among two different switches makes it much harder for the switches to effectively manage the bandwidth. It is generally simplest to think of the bundle as a single logical trunk and connect it between the two end point switches. Just avoid splitting the bundles.

Filtering

Most organizations do little or no filtering on their switches. For most networks, this is the right amount. It is generally much easier to filter on routers than switches. However, in some cases it is more effective to filter on the switches. In general, the same reasons for filtering on routers also apply here:

- Security
- Cleaning up for ill-behaved applications

The other reason I listed in the router case, policy-based routing, could theoretically apply here as well. But that sort of facility should be used sparingly at best, and where it is used, routers are a more natural place for it, so I do not include it here.

Security filtering is usually handled on switches in two ways. Many vendors offer some sort of port-level security, in which only a specified MAC address is permitted to connect to a particular port. The second type of security filtering typically restricts packets according to their contents, usually allowing only packets with certain source MAC addresses to communicate with sensitive devices.

Port-level MAC address security features allow the switch (or hub, since this feature is also available on some hubs) to lock out any devices except the one specified. If a particular workstation is supposed to be connected to a particular port, then only that workstation will function on that port. If another device is connected, it will have a different MAC address and the switch (or hub) will disable the port, requiring manual intervention.

This sort of feature is provided to prevent people from putting unauthorized equipment on the network. It is not perfect because many types of devices can use a manually configured MAC address instead of their burned-in-address (BIA). But it is a useful measure if this sort of problem is a concern. Note, however, that there is significant administrative overhead comes in maintaining the table of which MAC addresses are permitted on which ports throughout a large network. Generally, I wouldn't use this feature unless a compelling security concern warranted it.

In the second type of security filtering, you instruct the switch to look at the packet before transmitting it. If a sensitive server, for example, is only permitted to communicate with a small list of other MAC addresses, then this information could be programmed into the switch. Not all switches allow this sort of functionality, and it can be difficult to maintain such a switch. Once again, this feature should only be used if there is a strong overriding security concern.

I have already talked about certain broadcast storm problems. These problems are commonly handled with a simple volume filter. In some cases, it may be worthwhile to use a more specific filter. For example, I was once responsible for a network that suffered from the RWHO problem mentioned earlier. I was able to write a special purpose filter to restrict these packets on the switch. As for the security-based filtering, it was also a huge administrative problem. This sort of filtering should be used sparingly, and only where absolutely necessary. Bear in mind that switch manufacturers know this, so they tend not to provide extensive filtering capabilities.

VLAN-Based Topologies

Now that I have discussed how not to use VLANs, I'd like to turn to more positive matters. VLANs are typically used in bridged sections of a LAN, but they give two important advantages over older bridging techniques. First, they allow much more efficient use of trunk links. The ability to combine several segments into one trunk without having to first bridge these segments together allows you to use far fewer physical resources (ports and fiber or copper connections). Second, a VLAN-based architecture built on top of a rational hierarchical structure allows great flexibility in expanding or modifying the network without having to fundamentally change the Core.

Here are a few good ways of employing VLANs in a hierarchical network design. Figure 3-18 shows a rather typical VLAN topology. In this picture, several different segments are visible on the various Access Level switches. These VLANs are collected on the two redundant Distribution Level switches. At the Core, two redundant routers handle the VLAN to VLAN routing.

Although this diagram is a vastly simplified version of what you might find in a real large-scale LAN, it demonstrates some important features for VLAN topologies. First consider the trunk design.

Trunk design

Each Access Level switch has two trunk connections to redundant Distribution switches. This switch provides excellent fault tolerance. For the purposes of this discussion, let's assume that the trunks are configured so that only one trunk is active at a time. The primary trunk must fail completely before the secondary trunk becomes active. This fault tolerance scheme is fairly typical for trunks. Each Access switch has

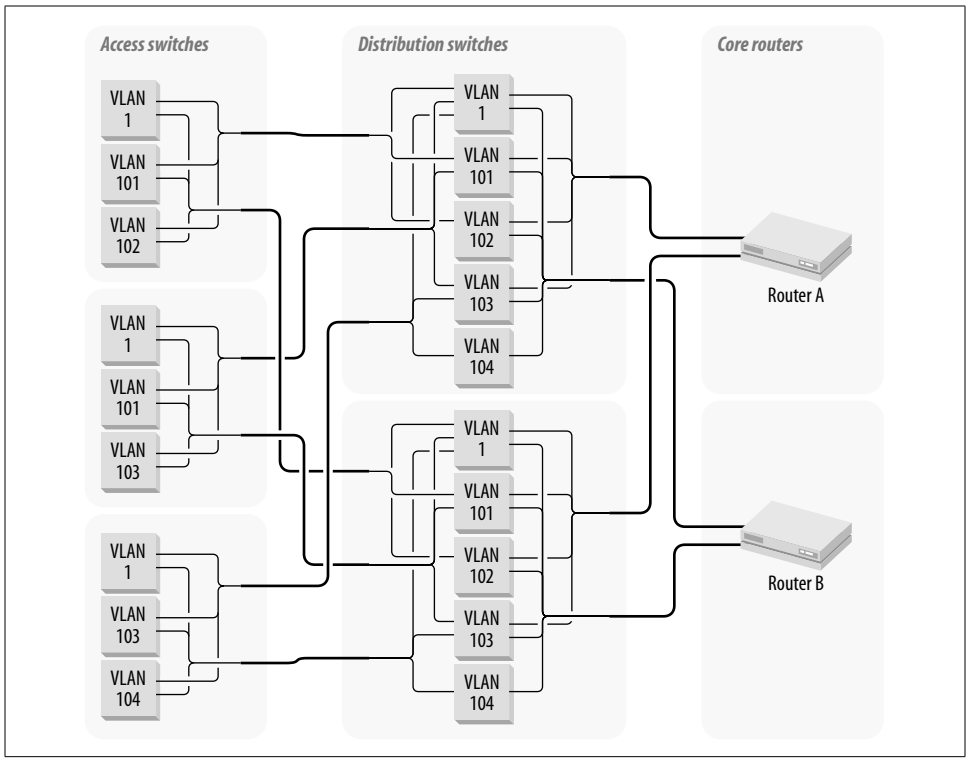

Figure 3-18. VLANs in a hierarchical network design

two trunk connections to provide complete redundancy. Notice that if you had to run a separate link for every VLAN, you would need six links for redundant connections to each Access switch. Worse still, if you added another VLAN on this Access switch, you would need two more ports and two more fiber connections. With the design shown in Figure 3-18 you can keep adding more VLANs to the existing trunks until you start to get congestion problems.

Figure 3-18 has five different VLANs. VLAN 1, the management VLAN, is present on all switches. I will talk about network management considerations in more detail later in this book, but for now I will just point out that separating your management traffic from your business traffic is a good idea. With this sort of VLAN structure, putting the management segment for all switches on the same VLAN is very convenient. In any case, one can generally expect management traffic requirements to be much smaller than for business application traffic.

VLAN 1 is used for network management because some low-end switches require their management IP address to be associated with VLAN 1. Since the VLAN naming convention is globally relevant over large portions of the network, it's a good idea to use VLAN 1 for management on all switches just in case it's required on a device somewhere in the region.

The other four VLANs are all user segments of various types. I have arbitrarily put two such user segments on each Access switch. The actual number of VLANs you should support on each Access switch depends on geography and port density. In general, it is a good idea to keep it fairly low for efficiency on your trunks.

Notice no user VLANs appears on all Access switches. VLAN 101 appears on the first two switches, but is not present on the third. Similarly, VLAN 102 is only configured on the first switch. This configuration is important because of the way it affects trunk utilization. The trunks serving the first Access switch carry no broadcast traffic from VLAN 103 or 104, so that spaghetti VLANs can be avoided. If I had not done this, I would have quickly wound up with Spaghetti VLANs. Remember that one of the main reasons for segregating our traffic is to break up the broadcast traffic. If all VLANs are present on all switches, then all broadcasts traverse all trunks. In such a network, the only benefit to using VLANs is that the end devices don't see as many broadcast packets. VLANs can provide much greater benefits if they are used more carefully, though. Network designers use VLANs for efficiency, so they should not throw that efficiency away on a Spaghetti VLAN topology.

The Distribution switches collect all VLANs. In general, this sort of two-point redundancy is a good idea at the Distribution Level, but there will usually be several pairs of Distribution switches collecting VLANs for large groups of Access switches. For example, this diagram might just show the first two Distribution switches, which collect the first 4 user VLANs (plus the management VLAN) for the first 12 Access switches (of which I have shown only 3). Then the next pair of Distribution switches might collect the next 6 user VLANs for the next 8 Access switches, and so forth. Each group of switches will have a VLAN 1 for management. This VLAN 1 may or may not be the same VLAN 1 throughout the network, but it can be simpler to handle routing if it is.

Trunking through a router

The previous example had the routers at the Core. This location turns out to be one of the most natural places for them in a VLAN-based network design. Suppose, for example, that you wanted to put your routers at the Access Level. Then you necessarily route between user VLANs, so it becomes harder to bridge different user segments via VLANs. The same is true to a lesser extent if you wanted to put the routers at the Distribution Level.

It's more difficult, but possible, to have the same VLAN existing on two different sides of a router. Figure 3-19 shows one way to accomplish this feat. This picture shows three switches interconnected by three different routers. Switch A holds VLAN 102, Switch B holds VLAN 103, and Switch C holds VLAN 104. VLAN 102 has IP address `10.1.102.0`, VLAN 103 has `10.1.103.0`, and VLAN 104 has `10.1.104.0`. So, as long as the three routers know how to route to these three IP addresses, everything will work fine.

But there is a problem with VLAN 101. This VLAN, which has IP address `10.1.101.0`, is present behind all routers. So if a device on VLAN 101 on Switch A wants to communicate with another device on VLAN 101 on Switch B, the packet will hit Router A and won't know where to forward this packet. After all, the IP address range `10.1.101.0` is directly connected to one of its Ethernet ports. The IP address range is broken up behind different routers. Even the VLAN tagging information present on the other three VLANs disappears as soon as it hits the routers.

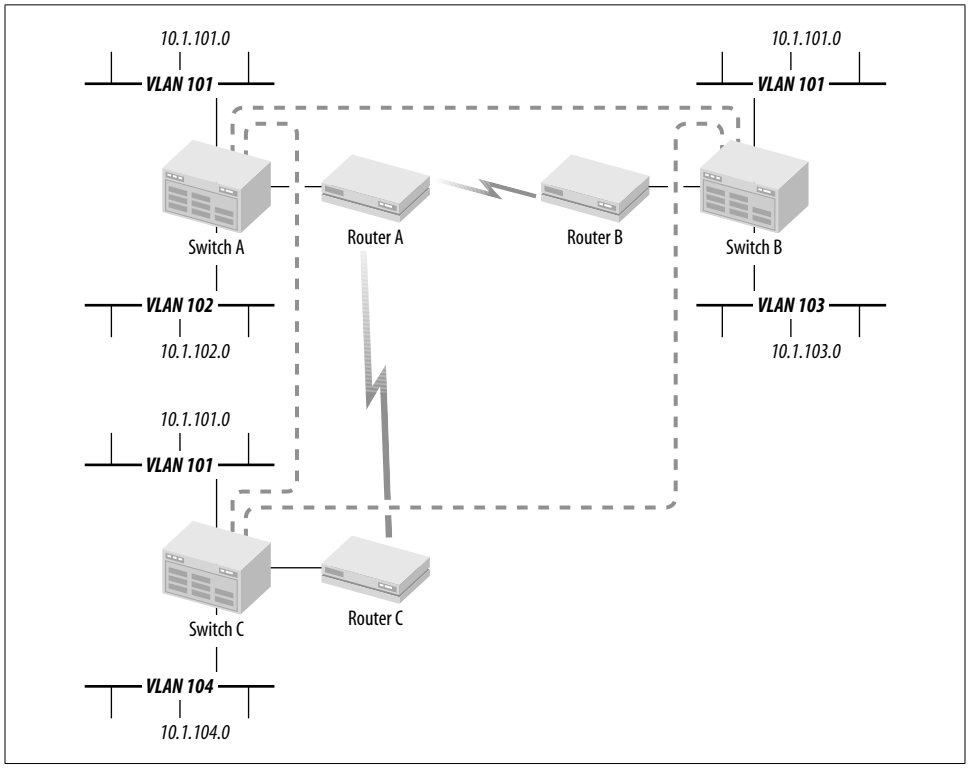

Figure 3-19. A VLAN split by routers

Routers are Layer 3 devices and they forward packets based on Layer 3 protocol information. VLAN information is fundamentally a Layer 2 concept. Thus, the only way to get around this problem is to configure a bridge or a tunnel that emulates Layer 2 between the various routers or switches (it could be done either as a router-to-router tunnel, or a switch-to-switch, or even switch-to-router bridge group). Then, when the device on VLAN 101 on Switch A sends a packet to the device on VLAN 101 on Switch B, the packet enters the tunnel and is transmitted to both Switch B and Switch C automatically. In short, the network has to bypass the routers.

There are many problems with this sort of solution. It is inherently more complicated because of the extra step of setting up tunnels or bridge groups. The designer

has to be extremely careful that whatever fault tolerance systems he has in place supports the tunnel or bridge group transparently. As I have mentioned previously, having an IP subnet broken across two routers is disastrous.

There is also potentially much extra traffic crossing these links. Suppose a device on Switch C, VLAN 104, wants to communicate with a device on Switch A, VLAN 101. The packet first goes to Router C, where it is forwarded to the local Switch C instance of VLAN 101. Then the switch bridges the packet over to Switch A. This packet passes through Router C twice.

Now suppose a device on VLAN 101 on Switch A sends out a broadcast packet to every other device on VLAN 101. This packet has to be duplicated and sent out to both Switches B and C (hopefully they will be configured to not reforward the packet again or it will cause a mess), again passing through the local router twice. The network in this simple picture has effectively doubled whatever broadcast congestion problems it might have otherwise had.

Now suppose that a device on any of these VLAN 101 segments wants to send out a packet to a VLAN 102 device. The destination is not on the local segment, so the source device must send this packet to the default router. But there are three routers on this segment—which one is the default? In fact, it could be any of them, so a device on Switch A may need to send its packets to Router B, which then forwards the packet back to Router A to be delivered to VLAN 102. The backward path is just as convoluted.

The other problem with this configuration is that it makes network management difficult. Suppose there is a problem in the IP address range `10.1.101.0`. The engineer trying to solve the problem still doesn't have any idea where that device is. There could be a problem with any of the three routers or with any of the three switches, and it could affect devices in one of the other locations.

The network designer should try to avoid this situation whenever possible. A good rule is to never try to split a VLAN across a router. It can be done, but the potential for serious problems is far too high. There is, however, one important case when it is unavoidable: when some external network vendor provides the intermediate routed network. The two sides of the same VLAN could be in different buildings on the opposite sides of a city, for example. If the link supplied by the network vendor is provided through a routed network, then there may be no other option but to use such an approach.

Trunks

So far I've talked about trunk links like they had some sort of magical properties, but there is really nothing particularly special about them. A trunk can be any sort of physical medium. Generally, it should support relatively high bandwidth to be effective, but the actual medium could be just about anything. The most common

technology used in trunks is Fast Ethernet, although Gigabit Ethernet is increasingly popular. ATM links are also used frequently. FDDI used to be fairly common, but it is being replaced as a trunk technology because Fast and Gigabit Ethernet systems are cheaper and faster.

What makes a trunk link special is the fact that it carries several distinct VLANs simultaneously. This is done by an extremely simple technique. Each packet crossing through the trunk looks exactly like a normal packet, but it has a couple of extra bytes called the VLAN tag, added to the Layer 2 header information. The tag's precise format and contents depend on the specific trunk protocol.

Trunks are useful because they allow the network designer to economize greatly on switch-to-switch links. If you had to carry three different VLANs (a modest and reasonable number) from an Access switch to a pair of redundant Distribution switches without using trunks, you would need at least six links. But if you did use trunks, you could achieve full redundancy with only two links. Better still, if you suddenly had to set up a new VLAN on that Access switch, you could do it all in software. There is no need to run another pair of uplink fibers to the Distribution switches.

To work as a trunk connecting two switches, both ends must know that the link in question is intended to be a trunk. They must also agree on the trunk protocol (which specifies the VLAN tagging format). This protocol usually has to be configured manually. But then, by default, most switches treat this link as a common trunk for all the VLANs this switch knows about. Some switches allow you to separately specify which VLANs use which trunks. In some ways, this specification is contrary to the spirit of trunks. But it can be a simple method for balancing the loading of your trunks, and in particular a method to divide up the broadcast traffic.

Generally, the trunks connect Access Level switches to Distribution Level switches in hierarchical network designs. Then there may or may not be further trunks connecting Distribution to Core Levels, depending on where the routers are. Extending trunks between two Access Level devices is not usually recommended; one usually wants to keep the relationship between the different levels as clear and clean as possible. Access devices that act as Distribution devices can make troubleshooting network problems difficult.

Trunk protocols. There is an IEEE standard trunk protocol, called 802.1Q. Because this standard was developed and released in 1998, after the requirement for such a protocol appeared, a handful of vendor-proprietary trunk protocols also exist. One of the most common is Cisco's ISL protocol, but several other proprietary trunk protocols are on the market.

ISL and 802.1Q share many similarities. Both protocols feature a generic VLAN header that can support several different standard LAN types. A trunk can contain many different VLANs, each of which can run many different Layer 3 protocols.

Other proprietary trunk protocols have other nice features as well. The Cabletron SmartTrunk system was relatively popular at one time because of its automated fault-recovery and load-sharing properties.

However, I recommend using the open standard wherever possible. All major manufacturers now implement 802.1Q, so there is very little reason to use the proprietary trunk solutions any longer, and I don't recommend doing so. The unique nature of trunking makes it one of the most important areas for using open standards.

Most networks have distinctly different requirements at their Access Level than in the Core or Distribution Levels. Consequently, it is quite likely that the switches at these different levels could come from different vendors. Since the hierarchical design model has most of its trunks running between these different levels and only a small number within a level, there is a good chance that you will have to connect a trunk between switches made by different vendors.

The difference between a regular Ethernet frame and an 802.1Q tagged frame is shown in Figure 3-20. Four extra octets (8-bit bytes) are added to the frame just before the length/type field. To ensure that this tagged frame isn't mistaken for a normal Ethernet frame, the "tag type" field is always the easily identified sequence "81-00" (that is, the first byte is 81 in hex and the second is 00 in hex). Then the remaining two bytes specify the VLAN information. For compactness, these two bytes are broken down into three fields of different bit lengths.

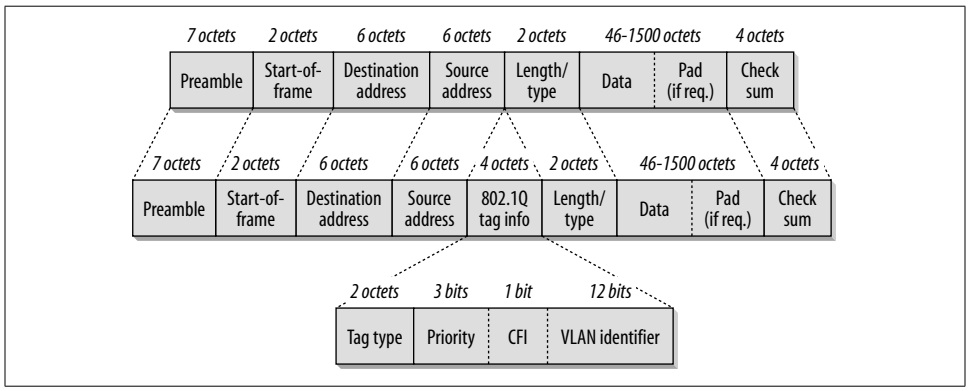

Figure 3-20. Q VLAN tagging format compared with normal Ethernet framing

The priority field is a three-bit number, also called "Class of Service" in some literature. Because it has three bits, this field can have values from 0 to 7. I will talk more about prioritization later in this book. But for now it's important only to note that Class of Service is a MAC-level priority, so it is not the same thing as the higher layer QoS concepts such as the TOS (Type of Service) or DSCP (Distributed Services Control Point) fields in the IP packet header. Putting this new Class of Service field in Layer 2 makes it easier for Layer 2 devices such as switches to use it.

Also note that the priority field is independent from the VLAN identifier field. It is possible to classify priorities on a trunk so that one VLAN has precedence over another and that a particular application on one VLAN has precedence over another application on a different VLAN. This concept will be important when you start to encounter congestion on your trunks.

The one-bit CFI field is the "Canonical Format Indicator." This field is set to 1 if a RIF (Routing Information Field) is in the Data segment of the frame, and 0 if there isn't. A RIF is a piece of information that allows a device to request a particular path through a bridged network. The CFI field makes it easier for switching devices to deal with RIF data by saving them the time of looking for this data when it isn't present.

And then comes the 12-bit VLAN identifier field. Having 12 bits, it could theoretically handle up to 4,094 different VLANs (since there is no VLAN zero and VLAN 4,095 is reserved). But I urge caution in configuring VLAN numbers greater than 1000 because of intervendor compatibility problems. The problem is that some switch vendors implement VLANs internally using their own native proprietary systems and then merely translate to 802.1Q. Some of these internal schemes have trouble with VLAN numbers greater than 1000. Worse still, some early VLAN schemes could only support a few hundred VLAN numbers, so don't assume that it will work until you've tried it.

Always remember that if you share VLAN numbers across a large Distribution Area, every switch in this area must agree on VLAN numbers. This is rarely a serious problem because a Distribution Area containing more than a few hundred VLANs would suffer from serious efficiency problems anyway.

Trunk redundancy. All of our discussion of trunks so far in this chapter has assumed that you will run redundant trunk links everywhere, but, in fact, there are two different ways to handle trunk redundancy. You can use Spanning Tree to keep one entire trunk dormant until there is a failure on its partner. Or, you can run both trunks simultaneously and consider all of the individual VLANs running through them to be distinct virtual links. Then you can run Spanning Tree separately for each VLAN.

In fact, it is not possible to run Spanning Tree separately for each VLAN when using 802.1Q, but it is possible with other trunk protocols, such as Cisco's ISL.

The per-VLAN option is considerably more complex, but it can sometimes be useful. Consider, for example, the network shown in Figure 3-18. The first Access switch has trunk connections to both Distribution switches. Suppose the upstream connections to VLAN 101 on the first Distribution switch were to break. In this case, you would want to use the second trunk, which goes to the second Distribution switch.

This scenario is actually relatively easy to get around. All you need is a trunk link between the Distribution switches. Then the first Distribution switch acquires its lost connection to VLAN 101 via the second Distribution switch through this trunk link.

In fact, it is extremely difficult to come up with examples where this is not the case. In general, since I always prefer simplicity to complexity, I prefer to use Spanning Tree on whole trunks rather than more individual VLANs within a trunk. Further, because many switches do not support running Spanning Tree for individual VLANs, compatibility helps to dictate the best methods as well.

However, this example brings up an important issue. If you run Spanning Tree on the individual VLANs in a network, you should not run it on the trunk as a whole. Conversely, if you run it on the trunk, you should disable it on the individual VLANs. It is very easy to generate serious loop problems by using a mixture of the two approaches.

When considering trunk redundancy, it is important to think through what will happen when a trunk breaks. A good hierarchical design with Spanning Tree should have very few problems recovering from a fault. One thing to beware of is a failure that breaks a Layer 3 network.

Figure 3-21 shows a network that has two routers for redundancy. These networks both serve the same IP subnet and the same IPX network. Assume that they have an automated system for IP redundancy such as VRRP or HSRP. No such system is required for IPX, so if the primary router on the segment fails, the other one will take over.

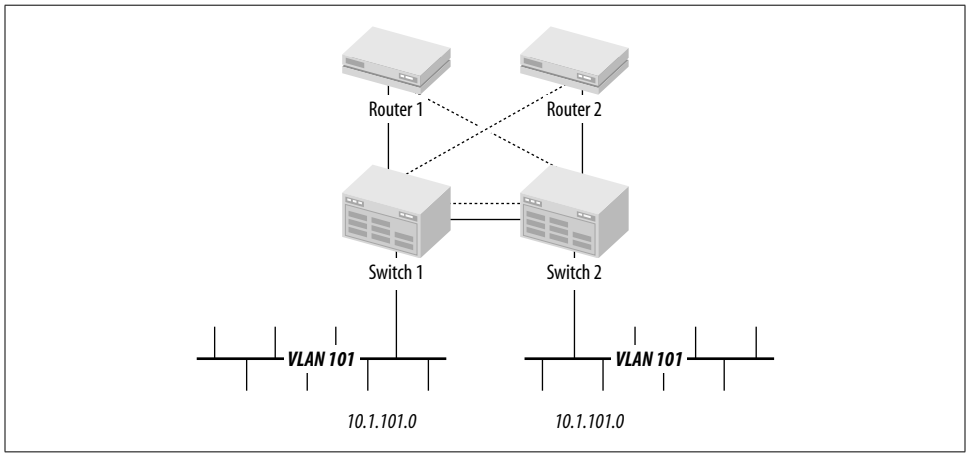

Figure 3-21. When a trunk breaks, it must not fragment a Layer 3 network

The same VLAN, number 101, which has IP address `10.1.101.0`, exists on both switches. Then, for diversity, the first router connects to the first switch and the second router connects to the second switch.

This design is seriously flawed. Consider what happens when the trunk connecting the two switches fails. Suddenly two distinct unconnected LAN segments have the same IP address range and the same IPX network number. Now both routers provide valid routes to these networks. Thus, no communication will work properly to either segment. This is almost exactly the same problem I described earlier with two routers on the same LAN segment, but here you can see that it happens with VLANs as well.

How does one resolve this problem? A couple of different approaches are available. One method connects both routers to both switches, as shown by the addition of the dashed lines in Figure 3-21. This solution is not always practical, depending on the capabilities of the routers, since it implies that both routers have multiple interfaces on the same network.

In fact, the simplest solution is to just run a second trunk between the two switches, as shown with the dotted line. Then you can simply rely on Spanning Tree to activate this link if the primary fails. Furthermore, if you suffer a complete failure of one entire switch, then you lose half of your workstations, but at least the other half continues to work. A failure of one router allows the other to take over transparently, so this is the most acceptable solution.

However, in a good hierarchical design, this sort of problem is less likely to arise because each Access switch connects to two different Distribution switches. Thus, the network would need to have multiple simultaneous trunk failures to get into this sort of problem.

Trunks on servers. Some types of servers support VLAN trunks directly so that you can have a single server with simultaneous presence on several different VLANs, as shown in Figure 3-22.

This is certainly an interesting thing to do, but it's important to understand why you would want to do this before trying it. There are different ways to achieve similar results. For example, many servers support multiple network interface cards (NIC). Installing two NICs in a server and connecting them to different VLANs via different switch ports has the benefit of simpler configurations on both the switch and the server and provides higher net throughput. Alternatively, if you can't afford to use multiple physical ports for whatever reason, then you could just as easily put the server behind a router and let the traffic route to all of the different user segments.

However, this strategy is cost-effective in some cases. For example, if the trunk connection is a Gigabit Ethernet link, it might be significantly less expensive than deploying a router solution, as routers with high-speed interfaces tend to be very expensive. At the same time, Gigabit Ethernet ports on switches can be costly. This strategy may be a convenient way of deploying a server for multiple user VLANs.

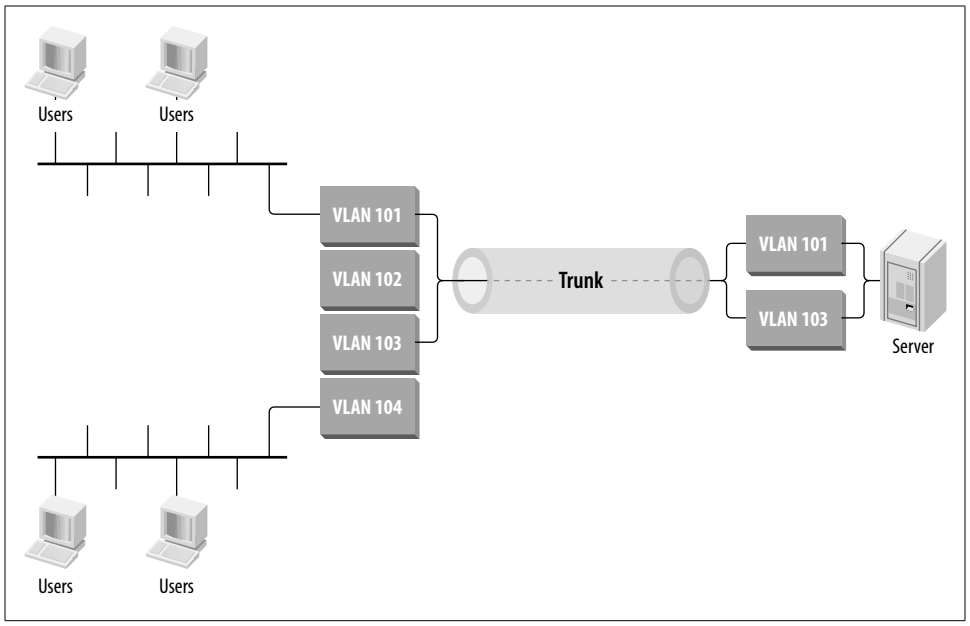

Figure 3-22. Some servers connect directly to trunk links to access several VLANs simultaneously

However, this method does not scale very well. If there will be many such servers, it would likely be less expensive in the long run to build a specialized high-speed server segment behind a router. Because it is a trunk link, the different VLANs will also compete with one another for server bandwidth on this link.

In previous chapters I made the point that only network devices should perform network functions. Therefore, I don't like connecting an end device to multiple VLANs, whether it is through a single port or through multiple ports. An end device should have a single connection to the network unless there is a compelling reason to do something more complicated.

VLAN Distribution Areas

One of the key concepts in building a VLAN-based network is the VLAN Distribution Area. Many networks have only one VLAN Distribution Area, but having only one in extremely large networks is not practical. It may be useful to break up the Distribution Areas of a network to improve efficiency. Figure 3-23 shows what I mean by a Distribution Area. This example is unrealistically symmetrical but the symmetry is not relevant to the concept.

In this diagram, four Access switches are connected to each pair of Distribution switches; Access switches A1, A2, A3 and A4 all connect to Distribution switches D1 and D2. Similarly, the next four Access switches connect to the next two Distribution switches, and so on. The central routing Core of the network allows the VLANs that appear on these various switches to connect to one another.

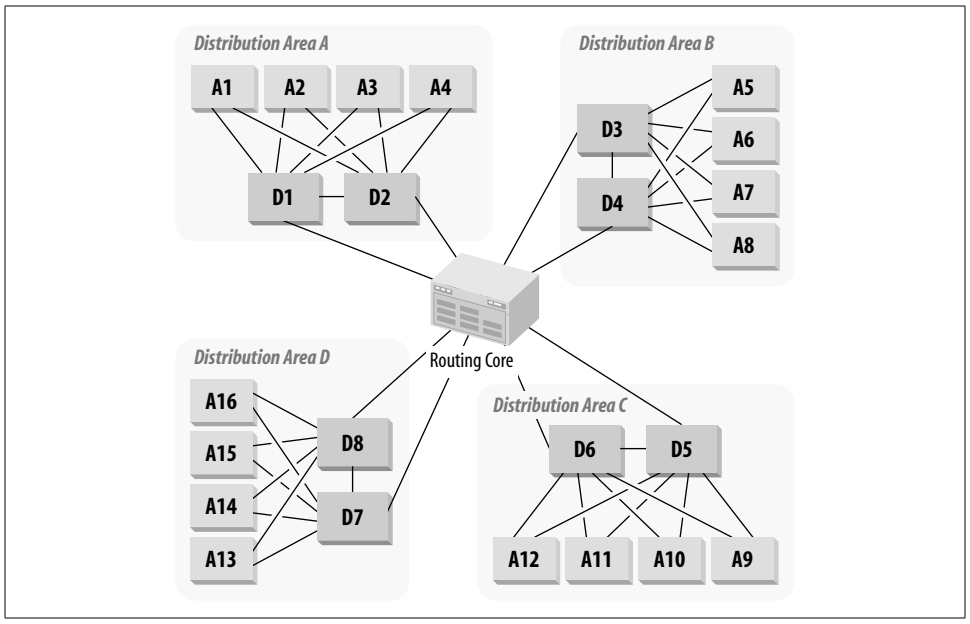

Figure 3-23. Distribution Areas

The four VLAN Distribution Areas in this picture are arbitrarily named A, B, C, and D. There is really no need to name your Distribution Areas, but it might help to rationalize the scheme if you do so. The essential idea is that the VLAN scheme is broken up so that there is no connection between the VLANs of different areas.

Why would you want to break up the scheme this way? Well, there are two main advantages to this approach. First, you may need to reuse certain VLAN numbers. This might happen because certain VLAN numbers such as VLAN 1, which is often reserved for network management purposes, are special. Or, it may happen simply because of limitations on VLAN numbering schemes on some hardware. For example, some types of switches only allow VLAN numbers up to 1000 or 1005, despite the theoretical limit of 4094 in 802.1Q.

The second and more compelling reason for breaking up your VLAN Distribution Areas is to simplify your Spanning Tree configuration. The network shown in Figure 3-23 has four different Root Bridges. All traffic has to pass through the Root Bridge in Spanning Tree networks. This situation can result in wildly inefficient traffic patterns. Breaking up your hierarchical design, as in this example, allows you to control your traffic patterns so that all packets between Core and Access Levels take the most direct path.

The disadvantage to building a network this way is that it makes it harder to share VLANs throughout the larger network. For example, since no trunks exist between Distribution Areas A and B, sharing VLANs between these areas is not possible. It is

critically important that you thoroughly understand what VLANs need to go where when constructing a VLAN Distribution system.

In most cases, it is best to build these Distribution Areas geographically. It is quite rare to find an organization that does not physically group employees performing related tasks. If there is a need for easy information sharing over the network, then chances are that this need exists for physical proximity as well. This is not true universally, of course, but most organizations attempt to group themselves this way. A logical way to build Distribution Areas would be to build on a campus LAN, or by groups of floors in a large building.

The other nice feature about using Distribution Areas in this way is that it tends to prevent propagation of the VLAN Spaghetti problem. It tends to force the network to use both a reasonable number of VLANs in an area as well as prevent too much geographical spreading of VLANs.

Sizing VLAN Distribution Areas

Although technical and theoretical limitations on how many VLANs one can define in a VLAN Distribution Area exist, practical limitations are considerably lower. The Distribution switches have to see all of these VLANs, as do the routers that allow VLAN-to-VLAN connections. If the connection to the router is done by means of trunk connections, then the router has to have a logical interface for every VLAN.

Every additional VLAN in a Distribution Area requires additional CPU and memory resources in the Distribution (and possibly also the Core) Level of the network. Since every vendor implements these features differently, establishing solid rules for the maximum number of VLANs in a VLAN Distribution Area is not possible. A dozen VLANs are not likely to cause any problems, but a thousand is probably a bad idea. The two places you need to be concerned about are the routers that handle VLAN-to-VLAN connections and the Distribution switches (particularly the Root Bridge) that have to handle all the individual VLANs.

On Cisco routers, the usual rule for a safe upper limit to the number of logical interfaces is somewhere between 50 and 200, depending on the type of router and the amount of processing required. If the router (or Layer 3 switch) has to do a lot of filtering or has to look at more than just the destination address of each packet, then the number of VLANs should be reduced radically.

Remember that these numbers, while just general orders of magnitude, are for the entire router. If the router is used to interconnect several different Distribution Areas, then the number of VLANs in each area should be kept low to allow the router to function effectively.

The same arguments apply to the switches themselves. If the Distribution switches act strictly as switches, without needing to do any filtering, prioritization or other CPU intensive activities, they should be able to handle more VLANs. The more additional work the switch needs to do, the fewer VLANs it should have to carry.

In many cases, the governing factor for how many VLANs to allow in a Distribution Area is actually the backplane bandwidth of the Root Bridge (which should be the primary Distribution switch for the area) and the aggregate downstream bandwidth used by the trunks to the Access switches. There is a single Root Bridge through which all off-segment packets for a VLAN must pass.

Earlier in this chapter, I said that a good rule for trunk aggregation is to assume that 5% of the devices on the network will burst simultaneously. If you apply this limit to the backplane of the Root Bridge, then you should get an extreme upper limit to how many devices should be supported by a single Distribution Area, independent of the number of VLANs used.

Typical modern switch backplane speeds are between 10 and 50Gbps. If all workstations are connected to Fast Ethernet ports, then this switch can support somewhere between 10,000 (for the 10Gbps backplane) and 50,000 (for the 50Gbps backplane) workstations. Because the aggregate backplane speed includes all possible directions, I have included a factor of 2 to account for both sending and receiving by the bursting workstations.

Clearly, these numbers are vast overestimates for several reasons. First, these nominal aggregate backplane speeds are measured under optimal conditions and ideal traffic flow patterns that are almost certainly not realized in a live network. Second, this switch may have to do a lot of work filtering, tagging, and prioritizing traffic, as well as its primary switching functions. So it probably doesn't have the CPU capacity to handle this much traffic, even if its backplane does. Third, you should always keep a little bit of power in reserve for those rare moments when the network is abnormally busy. Fourth, related to the third point, you should always allow room for growth.

A reasonably safe hands-waving estimate for the maximum number of workstations that should go into a Distribution Area is somewhere on the order of 1000. If every VLAN supports 50 workstations, it would probably be a good idea to keep the number of VLANs in each Distribution Area at around 20.

As the backplane speeds of these switches increases, generally so do the attachments speeds of devices. The reader may have access to switches with backplane speeds of several hundred Gbps that were not available when this book was written. If the reader also has a number of devices connected using Gigabit (or the emerging 10Gbps Ethernet Standard), then the factors still come out about the same.

Implementing Reliability

Reliability in a network comes primarily from careful design work—the result of the right mixture of simplicity and redundancy. Too much redundancy in either equipment or connections results in complexity, which makes a network harder to maintain and more likely to break in strange, unexpected ways, having too many links

also makes it hard for the dynamic routing protocols to find the best paths through the network, which results in instability as well. Of course, you need some redundancy to eliminate your key single points of failure. However, you should never sacrifice the simplicity in your overall concept of the network.

Coupled with this concept is the issue of scaling. The concept of the network should be clear enough that adding new parts or eliminating old ones should not change it fundamentally. Scaling becomes a reliability issue because every network grows and changes over time. You should ensure that something that once worked will continue to work.

Throughout this chapter, I show example networks that have every Distribution Area connected through two Distribution switches, with every Access switch connected to both. Every Core or Distribution router has a backup. Every trunk link has a secondary link. These backup connections are never *ad hoc*; they are part of the global plan of the network. If a particular structure is used in the Distribution Level of the network, then it is used similarly in every Distribution Area. This modular construction scheme makes the network much easier to manage and easier to grow, migrate, and change.

Wherever you use backup links and backup devices, you must have automated fault recovery systems. There is little point in implementing a secondary device that does not automatically take over when the primary fails. Once again, simplicity of concept is the rule in the fault recovery system.

It is best to use as few automated fault recovery systems as possible. Spanning Tree is able to swing traffic to backup trunk links when the primary trunks fail, but the same configuration can also bring a backup switch on line if the primary switch fails. There is no need in this case to implement more complex strategies that might treat these two problems separately.

Multiple Connections

Not every device can have a backup. In most cases, it is neither cost effective nor technically practical to back up the Access Level of the network. Most end devices can only effectively use one network connection. Any system of redundancy at the Access Level ultimately reaches a single point of failure somewhere. Since one of the primary design goals is simplicity, it is best to acknowledge that one cannot readily implement redundancy in the Access Level and should instead work on ensuring the reliability of the rest of the network.

Looking back at Figure 3-23, each Access switch has two trunks, one to each of two redundant Distribution switches. With this configuration, you can lose any trunk connection, or even one of the Distribution switches, without affecting user traffic through the network.

In this picture, both Distribution switches also have redundant connections to the network Core, but the Core itself is not shown in detail. It is not shown because, up to this point, the network design is fairly generic. Later in this chapter, I will discuss the different options for locations of routers in a large-scale LAN. They can be in the Core, in the Distribution Level, or in both. The appropriate types of connections for these different design types are slightly different.

The key to working with backup links, switches, and routers is in the automated fault recovery system used. Since a Distribution Area is essentially a set of parallel broadcast domains, the best way to implement redundancy is to use Spanning Tree.

I mentioned earlier in this chapter that to use Spanning Tree effectively, the two Distribution switches must have a trunk connection between them. Another way of looking at this is by our simplicity requirement. The Spanning Tree Protocol needs to have a Root Bridge, which is the main switch for a Distribution Area through which all traffic must pass. Simplicity tells you that you should have every other switch in the Distribution Area connected as directly as possible to this Root Bridge. If possible, the Root Bridge should have a single trunk connection to every other switch in the area. Then, the backup Root Bridge switch also must have a direct trunk connection to the primary. Similarly, every other switch in the area needs a direct trunk to the backup Root Bridge in case the primary fails.

There are inherent scaling problems with directly connecting every switch to both of the two Distribution switches, which will always limit the number of switches in a Distribution Area, as I have already discussed. Keeping your Distribution Areas relatively small and modular will be good for overall network performance anyway.

Router-to-router redundancy has different requirements for multiple connections than the switch-to-switch case I was just discussing. The dynamic routing protocols for IP operate completely differently from the Spanning Tree Protocol. Instead of shutting down redundant links, IP routing protocols seek only to rate the different path options and select the most appropriate at the time. If one path goes away, another is selected from the list of possibilities.

Again, simplicity is the watchword for IP dynamic routing protocols. Every router in an OSPF area must maintain information about all of its neighboring routers (the ones with which it shares a direct link), and routing table information about every other device in the area. It is important to keep the topology of an area as simple as possible. The simplest schemes connect everything redundantly, but with as few connections as possible.

Large-Scale LAN Topologies

There are three main options for large-scale topology. If you want to use VLANs, and their benefits should be clear by now, then you need to have routers to interconnect them. Your options basically come down to where to put these routers. You can put

them in the Core or in the Distribution Level, or you put them in both. It is usually best to avoid putting routers at the Access Level of a LAN, but for very large networks it is easy to see that you get much better scaling properties if you include routers in the Distribution Level.

Routers in the Core Level

Perhaps the simplest and most obvious way to build a large-scale hierarchical network is to use a model like that shown in Figure 3-24. In this diagram, several different Distribution Areas are connected via a central Routing Core consisting of two routers. All Distribution Areas are redundantly connected to both Core Routers from both Distribution switches.

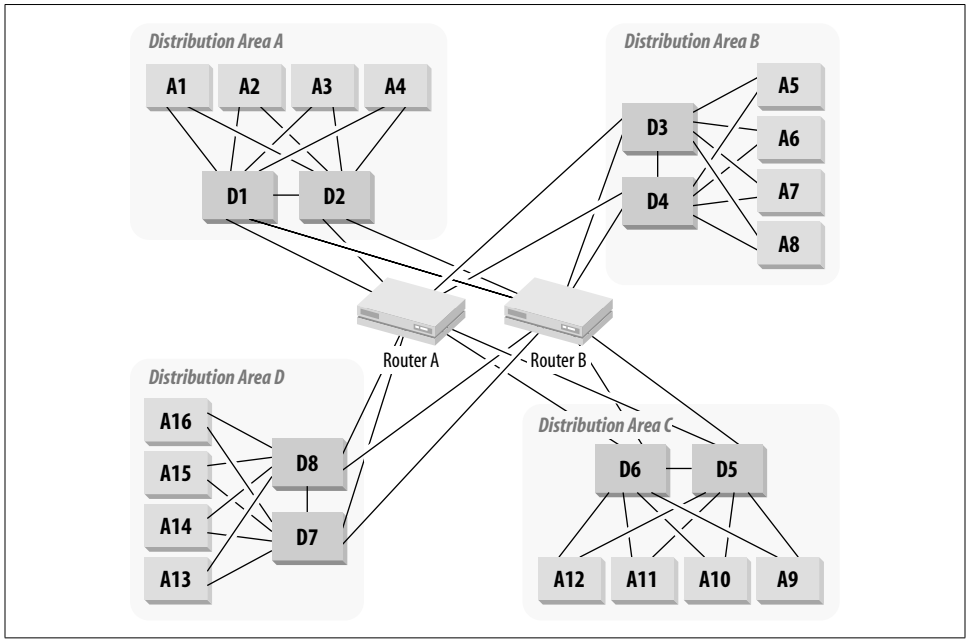

Figure 3-24. A hierarchical network with central routing

The result of all these redundant connections is that any device in the Core or Distribution Levels can fail without affecting network operation. Each Distribution Area has redundant Distribution switches, either of which can act as a Root Bridge for this area. Both Distribution switches have connections to both of the two Core routers. If either Core Router fails, you have complete redundancy.

The best part of the redundancy in this network is its simplicity. There are only two central routers (there may be additional routers connecting to remote sites, as I will discuss shortly), and either can quickly take over all central routing functions in case

the other fails. Because of the way that these routers are connected to one another and to all Distribution switches, they can both be used simultaneously. However, the extent to which these routers share the load depends on how the dynamic routing protocols are configured.

The limitation to this design is the capacity of one of the Core Routers. You must configure these two routers so that either is able to support the entire network load in case the other fails. So Figure 3-25 shows a simple way of overcoming this limitation. It still has a central routing Core, but now there are four routers in the Core. Each pair of routers is responsible only for a small part of the network.

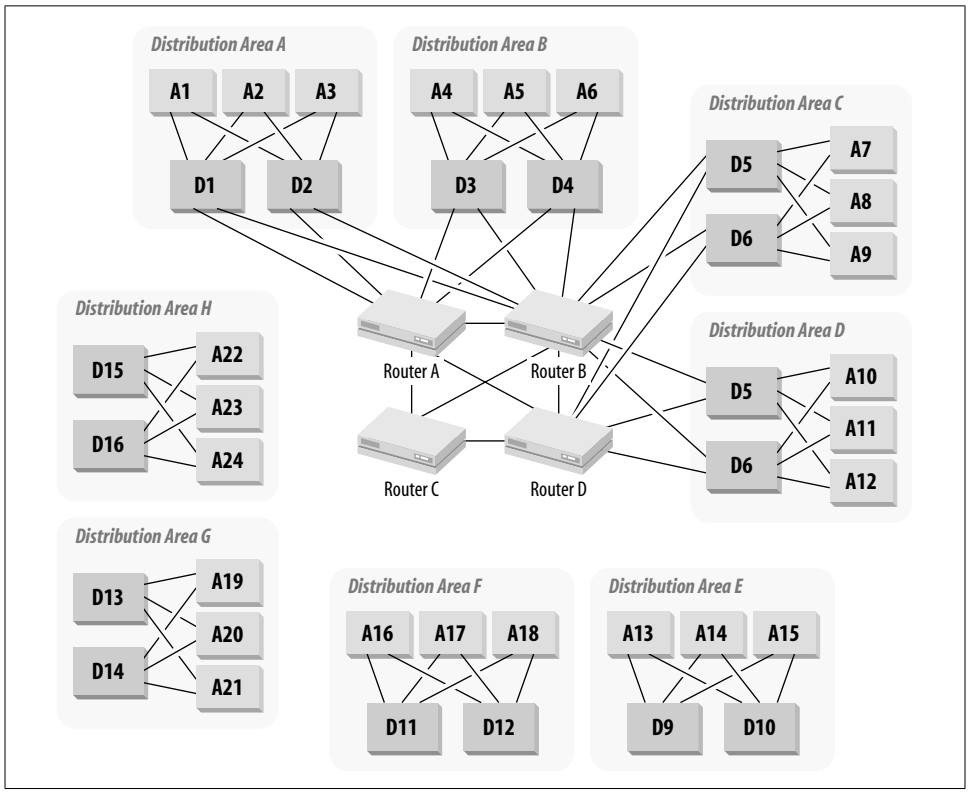

Figure 3-25. Central routing model for increased capacity

There are many different ways to connect such a Core. Figure 3-25 shows a Core with four routers that are interconnected with a full mesh. I have already indicated that a full mesh does not scale well, so if the network will need further expansion, full mesh would not be a good option. Figure 3-26 shows a similar network but with six central routers connected to one another by a pair of central switches.

Note that there need be no VLANs defined on these two switches. Both switches S1 and S2 have connections to all six routers. A natural way to define the IP segments on these switches is to have one switch carry one subnet and the other carry a different subnet. Then if either switch fails, the dynamic routing protocol takes care of moving all traffic to the second switch.

In this sort of configuration, it is generally useful to make the routers act in tandem. Assuming that Distribution Areas consist of two Distribution switches and several Access switches, you would connect both switches to both routers in this pair, and you can connect several Distribution Areas to each pair of routers. The actual numbers depend on the capacity of the routers. All connections will be fully redundant. Then only the Distribution switches that are part of this group of Distribution Areas will connect to this pair of routers. The next group of Distribution Areas will connect to the next pair of Core Routers.

Routers in the Distribution Level

There are two ways to bring the routers into the Distribution Level. One is to simply extend the concept shown in Figure 3-26 and arbitrarily proclaim that the two central switches S1 and S2 are now the Core and the routers are all in the Distribution Level. The distinction between "Core" and "Distribution" Levels is somewhat vague and depends partially on where you draw the lines. One problem with this way of drawing the lines is that these routers interconnect different Distribution Areas, so it is a little tenuous to claim that they are part of the Distribution Level.

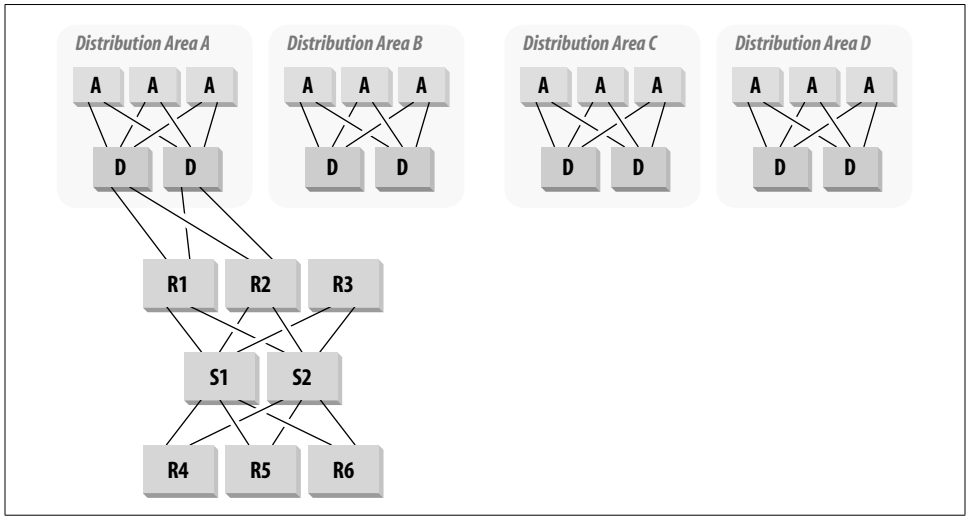

Figure 3-26. Central routing and switching

The second way of bringing routers into the Distribution Level is to have one (or preferably two, for redundancy) router for each Distribution Area. This option is shown in Figure 3-27.

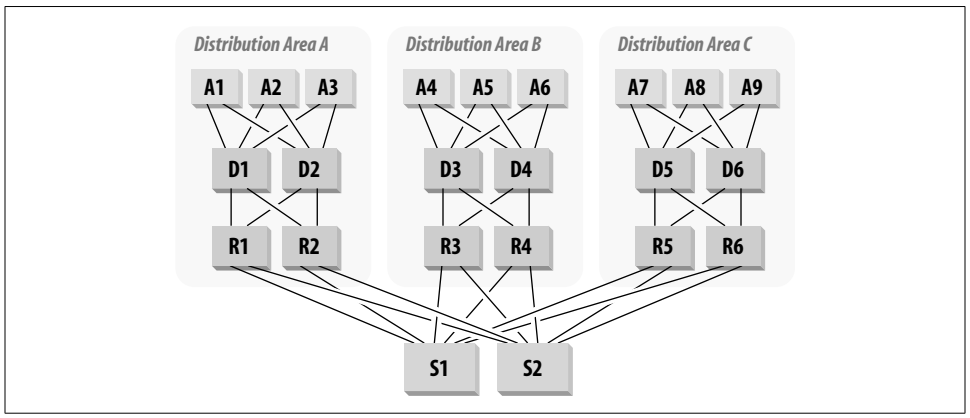

Figure 3-27. Distributed routing and central switching

One advantage to this approach is that it provides a very natural application of Layer 3 switching. Each Distribution switch could contain a Layer 3 switching module. This way, you can provide efficient VLAN-to-VLAN communication within each Distribution Area. You would then construct two additional VLANs on each Distribution switch that would connect to the two central switches.

In this sort of model, where routing functions are downloaded to the Distribution Level, another sort of efficiency can be used. Since how you decide which VLANs comprise a VLAN Distribution Area is somewhat arbitrary, you can deliberately choose your areas to limit traffic through the Core. This may not always be practical, particularly if the Distribution Areas are selected for geographical reasons. If it can be done, though, it may radically improve the network performance through the Core.

Routers in Both the Core and Distribution Levels

It's pretty clear that the network shown in Figure 3-27 has good scaling properties, but there are limits to even this model. In Chapter 6, I will discuss the IP dynamic routing protocol called OSPF. This protocol allows IP routers to keep one another informed about how best to reach the networks they are responsible for. There are other dynamic routing protocols but OSPF is an open standard and an industry norm. The comments that follow turn out to be applicable to most of the alternatives as well.

In Figure 3-27, all of the routers talk directly to one another through the Core switches. In any dynamic routing protocol, every router must know about all of its neighboring routers. It maintains a large table of these neighbor relationships and has to keep it continuously up to date. The more neighbors it has, the harder this job becomes, with similar scaling properties to a fully meshed network. The usual rule is that you never want more than 50 routers in one OSPF area. There are exceptions to this rule, as I will discuss in the section on OSPF, but it is never wise to push it too far.

If you want no more than 50 routers in your Core, then you can have no more than 25 VLAN Distribution Areas, since there are two routers in each area. With a capacity of over a thousand users in each Distribution Area, this is a limit that only large organizations will hit. However, it turns out that it isn't terribly difficult to overcome.

All you need to do is create a hybrid of the two solutions, with routers in the Core and Distribution Layers. Each Core router will handle several Distribution routers to allow excellent scaling properties. Figure 3-28 shows an example of how this hybrid might work. In this figure, the two Core routers that serve the Distribution Areas shown are the OSPF Area Border Routers (ABR) for these Distribution Areas.

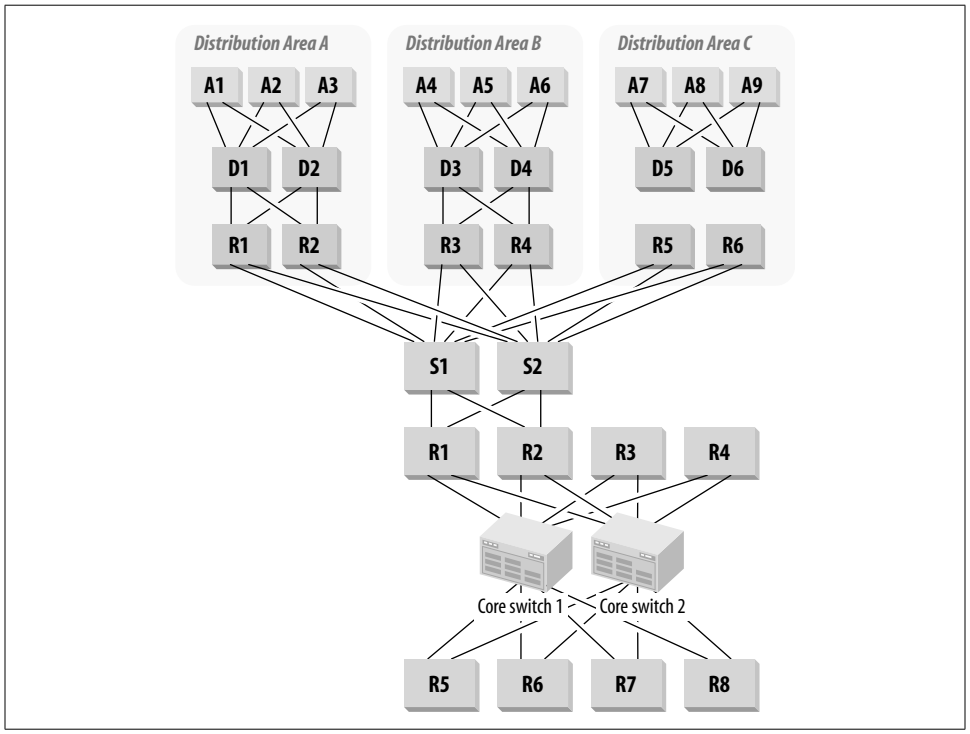

Figure 3-28. Distributed routing coupled with central routing

There are two other key advantages to this sort of design. First, it makes it extremely easy to spread the Distribution Areas geographically. In fact, you could even make your Core spread physically throughout a campus area, or even across several cities. However, doing so is generally not a good plan. The Core in this case represents our OSPF area 0 (a concept that I will explain in Chapter 6). There can be performance and reliability problems in a network that has its area 0 dispersed over wide area links. These problems can be overcome with careful tuning of OSPF parameters, but it leads to a network Core that has to be monitored very closely. A broken link in the Core could have disastrous consequences.

It is actually simpler to have the Core in a single geographical location and to bring the links to the various Distribution Areas via WAN links.

That point leads to the second advantage. It is very easy to integrate a large WAN into this sort of design, as I will show in the next section.

Connecting Remote Sites

In all but rare exceptions, if you want to get data to remote locations, it is best to route it. Bridging over WAN links should never be the first choice. Thus, the question becomes, where do you connect the routers for these remote sites into your LAN?

There are three possible types of WAN links that you might be interested in connecting. There might be a few geographically remote sites connecting into your LAN, or you might want to attach a more elaborate branch network. The third option involves connecting to external networks such as the public Internet.

In both of the internal cases, it is best to put these connections on routers, and in both cases you should put these routers as close to the Core as possible. Exactly where you connect them depends on where your other routers are. In the external case, the connection should almost certainly be behind a firewall. The question you need to answer here is where to put the firewall.

For internal networks, including both WANs of minor and major proportions, you have to share dynamic routing information with the existing routers. I assume throughout this discussion that this dynamic routing protocol is OSPF, but again, the comments apply generally to most dynamic routing protocols.

The main difference between the case of the small and large WAN is just one of numbers. A WAN of any size should never be part of the network's OSPF area 0. For a single external site, you might be able to get away with it, so I will briefly discuss the single site case.

The easiest way to treat a single external site is to think of it as a VLAN Distribution Area of its own. If it is a sufficiently important external site, then you might want to

allow multiple routers and multiple WAN links. A smaller site might be connected with only a single link, probably with dial backup.

There will be a router on the remote site with some sort of WAN circuit connecting it to a router on the main site. One simple way of connecting this router on the main site is to treat it as just another VLAN router. For the case where routers are connected only in the Core, the easiest method is to connect this WAN router to the Core routers as if it were one of them.

It is generally not a good idea to use the LAN Core router as a WAN router. The requirements for LAN Core routers are different from the requirements for a WAN router. The LAN Core router has to handle a lot of VLANs and has to move packets between similar media as quickly as possible. The WAN router has to buffer data and act as a point of junction between LAN and WAN. It is likely that this role will force you to use different router models, perhaps even from different vendors for these two functions.

In the cases in which the VLAN routers are moved into the Distribution Level, it becomes easier to connect the WAN routers. Then, for either a single-site or a multiple-site WAN, you would connect them as shown in Figure 3-29.

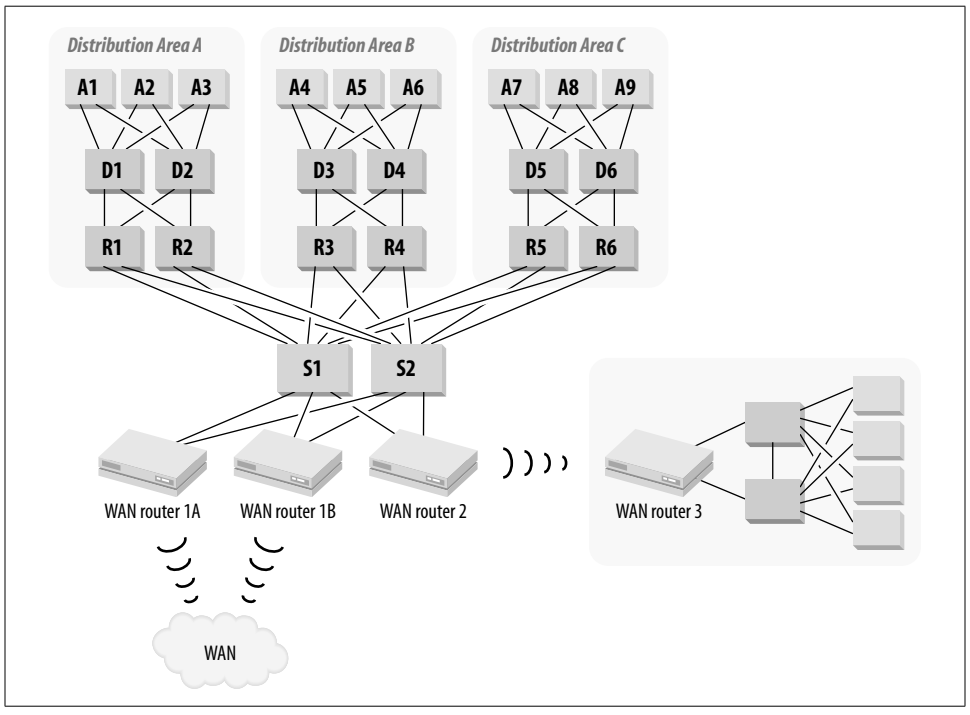

Figure 3-29. Connecting to remote networks in the Distributed Routing model

This diagram shows both a single-site and a multiple-site WAN. For the single-site WAN, I assume that the remote site is sufficiently complex to require its own hierarchical network. If it were smaller, then a single switch might be sufficient.

For the multiple-site WAN, the entire cloud is connected to two routers for redundancy. Both routers connect to the LAN Core switches. These new WAN Access routers become members of the OSPF area 0. In the multiple-site case, there will be too many downstream routers to put them all into area 0. This means that the router at the main site must be an OSPF area Border Router. Again, I will explain this concept in more detail in Chapter 6.

The other common way of connecting remote sites uses a WAN Touchdown segment. This segment is simply a separate router-to-router LAN segment or VLAN that only connects WAN routers, as shown in Figure 3-30. In this picture, the Touchdown Segment is created by two redundant routers that connect into the network Core. These routers may be part of the Core, or they may be Distribution Level routers; it all depends on the type of large-scale topology being used.

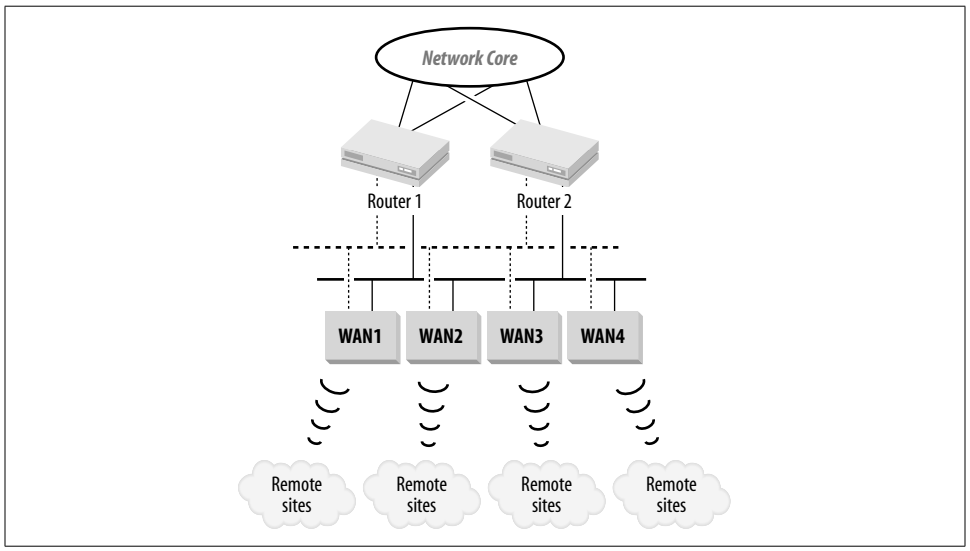

Figure 3-30. Connecting to remote networks with a WAN Touchdown Segment

The segment itself is a single VLAN that may be implemented on more than one switch or hub. A single segment like this has an inherent lack of redundancy. However, it can easily be improved by doubling the segment as indicated by the dotted lines.

The Touchdown Segment model for connecting WAN routers to a LAN is a fairly common technique. It has several advantages over connecting the WAN directly to the Core of the network due to the segment being separated from anything internal by means of routers.

First, if there is a requirement for security filtering, then this is a safer method for connecting the WAN. The remote sites may be less trustworthy than the internal network, or they may even be connections to an information vendor's site. In these cases, it is easy to offer basic security support by implementing filtering on the two routers that connect the Touchdown segment or segments to the main network.

Second, WAN links are inherently less reliable than LAN links. It may be desirable to protect the internal network from the effects of unstable links by using these routers as a sort of buffer zone. One of the problems with using a dynamic routing protocol is that flapping links cause all other routers in the area to repeatedly update their routing tables to reflect each change of state. One way to protect against this updating is by using the two routers that connect to the Core as a transition point in the routing protocol. You could run a different routing protocol on the Touchdown segments than you do in the Core, or you could use Border Gateway Protocol (BGP, another routing protocol) on these routers to separate the Touchdown segment's routing protocol from the internal network's routing protocol. BGP will be discussed in Chapter 6.

A third advantage to using Touchdown Segments this way is that it provides an easy expansion method for better scaling. If the Touchdown Segments become congested, building additional segments in the same pattern is relatively easy. If you were connecting each WAN router as a separate Distribution Area, then you would have to think very carefully about its connections to the Core each time. However, with Touchdown Segments, it is much more straightforward to expand the architectural model.

General Comments on Large-Scale Topology

Throughout all of these examples, I have assumed considerable symmetry in the large-scale topology. Although I haven't made a point of discussing this topic until now, it is actually an important feature of a good design. It's important to decide on a global strategy for the network and then follow it. Combining different types of designs doesn't work well.

For example, if your network is large enough to require using routers at the Distribution Level, then all Distribution Areas should have routers. It is certainly reasonable to have a migration plan to do this one area at a time, but this phase is transitional. In the target design, the network should follow consistent rules.

There will always be portions of the network that need to be treated as exceptions. It is generally a good idea to devise a standard method for dealing with exceptions, as I did with the remote sites considered in the previous section. If a few special VLANs require filtering, then they should all be treated with the same technique.

A theme that will repeat throughout this book is simplicity of concept. The benchmark for the appropriate level of simplicity is that an engineer familiar with the network in general should be able to troubleshoot problems on the network without documentation. This rule may sound arbitrary, but any engineer who has been awakened to diagnose network problems over the phone in the middle of the night will immediately recognize its value.

Another key advantage to this level of simplicity is that it allows new staff to learn the system quickly. Building a network that only one genius can understand is a terrible mistake. Sooner or later this genius will grow tired of taking trouble calls and will want to train a successor. Furthermore, a simple network design can also be handed over easily to relatively junior operations staff to manage. This feature has obvious advantages for maintainability, and maintainability is an important key to reliability.