CHAPTER 9

Network Management

Network management is an afterthought in many networks. This is a pity because the network designer can do many things to facilitate network management. In most large organizations, the job of network manager is considered "operations," while network design is done by a different implementation group. Frequently, these two groups report to different departments of the company.

If a network can be managed easily, then it is inherently more reliable. Thus, manageability is a fundamental design goal for a good network. Before I launch into a discussion of design implications for manageability, I need to spend some time talking about what I mean by network management.

Network-Management Components

The OSI has published an official definition of network management that includes five different components: configuration management, fault management, performance management, security management, and accounting management. I usually think of performance management as being composed of two separate subcomponents. The first is a tactical performance management, and the second is the more strategic long-term capacity planning component.

Configuration Management

Configuration management actually includes two different but related activities. The first keeps track of physical hardware, serial numbers, locations, patching information, and so forth. The second part of configuration management is the process of modifying, backing up, and restoring the software configuration of network equipment. This aspect of configuration management often becomes the focus of the whole activity. Many hardware vendors for routers and switches have excellent software for building and modifying software configurations. This software usually includes the ability to do scheduled backups of running configurations. This ability is an extremely important feature. If you have a recent configuration backup, then

replacing a failed router with a new one is a fast and easy operation. Without a backup, this replacement is time consuming and usually requires an experienced engineer to reconstruct the software configuration.

However, remember the physical tracking side of configuration management, especially if you deal with the configurations of Layer 2 devices such as hubs and switches. If network managers have accurate information about physical locations, MAC addresses, and cabling for end devices such as user workstations, then they can easily handle hardware moves, adds, and changes. In most organizations, business requirements force network administration to respond quickly and efficiently to requests for end-user moves and service changes. However, the cabling and hardware records are usually out-of-date, so every small move requires a technician to visit the site and carefully document the equipment and cabling. This process is expensive and slow.

Unfortunately, no software can solve this problem; it is primarily a procedural issue. Technicians making changes have to keep the records up-to-date, and the cabling and patch panels have to be periodically audited to ensure accuracy of the records. However, the network designer can do much to facilitate this process. If the patch panels are well designed and there is a clear correlation between physical floor location and cable numbers, then the technicians can at least get a running start at the job.

Fault Management

Fault management is what most people picture regarding network management. This management is the active monitoring of the various key network components to find problems and alert the appropriate people. But there is another side to fault management that is also important, particularly to the network designer—the troubleshooting process.

Troubleshooting occurs after the appropriate person knows of a problem. Usually, all that the fault-management software says is that a failure occurred somewhere near a particular device. It is usually not able to say what caused the problem, precisely which device needs attention, or even what the failure actually was. Upon receiving an alert, the network engineer must troubleshoot the problem, try to isolate the source, and look for a solution. For many problems there is a short-term solution to get the network back up immediately, as well as a long-term solution to make sure it doesn't happen again.

Performance Management

Performance management requires monitoring the network carefully and looking for bottlenecks and congestion issues. There is some overlap between performance management and fault management when performance problems become so severe that they interfere with the basic functioning of the network.

Capacity planning is the natural outcome of performance management. When network managers discover a systematic performance problem, such as a bandwidth shortage through performance management, they turn to capacity planning to resolve this problem. Capacity planning is fundamentally a network-design issue.

Security Management

Security management is the set of activities that ensure that the network's security measures work properly. Every firewall must be carefully monitored to see if it is in danger of compromise or if it is being abused in some way. Similarly, security management includes the maintenance of any filtering or encryption options.

Accounting Management

Security management leads directly into the concept of accounting management. Accounting partly deals with security. One of the main reasons for giving individual users different accounts is to ensure that they can only have access to the resources they require. This access is essentially a security issue. However, accounting management also includes the general problem of keeping track of who uses what on the network. In some cases, this information is used to bill for these services.

It should now be clear that all of the different activities of network management have network-design implications.

Designing a Manageable Network

A well-designed network has network management included in its basic requirements. At each stage of the design process, one of the key questions should be "how will this be managed"? The network designer should know from the outset where the network-management servers will be, both physically and logically, on the network. If special protocols are used for network management, then the design must ensure that this information can be delivered. If protocol analyzers or RMON probes are used to monitor network performance and assist in troubleshooting, then these devices should be placed in the design.

A probe is used to watch all of the traffic passively as it passes by. The issue of where to put probes is particularly difficult. In a switched or VLAN-based environment, probes are nearly useless if they are not deployed carefully.

Before switches, when Ethernet networks were made up of large bus configurations, it was possible to deploy a few probes and see everything. The probes would be placed near the Core of the network. From there, they could easily be switched to whatever LAN segment needed monitoring. However, in a switched Ethernet design, every end device is on its own LAN segment. This situation fundamentally changes how network monitoring has to be done.

One way to use a probe is to look at all traffic going to and coming from a particular end device by configuring the probe's switch port to mirror the port connecting to this device. This mirroring requires, of course, that a probe be available for use with this switch. In the ideal case where everything can be monitored centrally without having to leave the network operations center, this implies that there must be a separate probe on every switch. This prospect can be rather expensive. Thus, many organizations use either full or partial RMON probes built into their switches instead. The use of these probes allows good monitoring capabilities for every device in the network.

Another way to use a probe is on trunk links. In a hierarchical VLAN architecture, keeping a close watch on trunk utilization is important because this is where congestion problems usually first arise.

The discussion of hierarchical designs in Chapter 3 showed that trunks are used in four ways. They connect the Access Level to the Distribution Level and the Distribution to the Core. Internal trunks also exist within the Distribution and Core Levels. The important thing is that, while most of the switches wind up being at the Access Level, all trunks have at least one end in the Distribution or Core Levels. Thus, there is no need to deploy probes for monitoring trunk links to the Access Level. Not needing to deploy the probes at every switch should provide an important cost savings.

Chapter 3 also mentioned another important design issue for large-scale LAN environments in the discussion of hierarchical VLAN designs—the presence of a dedicated network-management VLAN. Obviously, the same VLAN cannot be present in different VLAN Distribution Areas, but every Distribution Area should have such a VLAN.

There are several reasons for having a separate network management VLAN that contains no user traffic:

- If you monitor traffic on a user VLAN, you don't want to see the probe traffic mixed in with user traffic.
- If you have to transfer configuration or software to or from the network devices, you don't want this traffic to interfere with production-application traffic.
- Separating the management VLAN from user traffic can be useful for security reasons. A router can then completely block SNMP and other management-specific traffic from passing between user and management VLANs. Blocking the traffic greatly reduces the chances of a successful attack.
- Perhaps most importantly, if there is a serious problem with a user VLAN, having a separate network-management VLAN allows the network engineer to get to the switch and hopefully fix the problem.

A network-management VLAN should always be part of the network design for every Distribution Area. This VLAN should contain the management addresses for all hubs and switches in the managed Distribution Area. It should also hold the

management interfaces for all probes and protocol analyzers in the area. If any devices, such as Inverse Terminal Servers, are used for out-of-band management, then these devices should also be connected through this management VLAN.

The management VLAN can suffer failures without affecting production traffic. Thus, it is not necessary to provide the same level of redundancy for this VLAN as for the rest of the network. However, if a large number of devices are to be managed through this VLAN, then it is wise to make it fairly robust. Economy of design may mean just building this VLAN according to the same specifications as the rest of the production network.

A network designer can do several different things at the physical layer to ensure that a network is designed to be managed effectively. These steps generally involve ease of access, clear labeling, and logical layout.

By ease of access, I mean that equipment that needs to be touched frequently should be in a prominent position. It should be safe from being bumped accidentally; there should also be no obstructions such as walls or poles to prevent a technician from seeing or handling the equipment. A good example would be cabling patch panels. Patch panels are almost always the network elements that need the most frequent physical access. Fiber patch panels tend to be used less frequently than the patch panels for user LAN drops. It is common for fiber patch panels to be mounted too high or too low to access easily.

Usually, the best way to handle LAN-cabling patch panels is to mount them in logical groups in equipment cabinets with locking doors. When the technician needs to make changes, the door can be easily unlocked and opened. The changes can then be documented and the door locked again to prevent tampering.

A clear, consistent, and simple labeling scheme is critical, particularly for patch panels and patch cords. Every patch-panel port should have a unique code number, and the formats of these codes should be consistent. These codes should clarify to what this port attaches. Suppose, for example, that you want to number the patch panels in a wiring closet that supports a large number of end users. In general, there should be a consistent floor plan in which every user work area is numbered.

Then, if each work area has three or four drops, you usually label the patch-panel ports with the work-area number followed by an alphabetic character to indicate to which cable drop the port connects. Each patch-panel port has a unique number that is easily associated with a particular cable drop in a particular work area. Thus, for example, desk number 99 may have 3 jacks beside it. If two of these jacks are for data and one is for voice, then you might number them 99-A, 99-B, and 99-V, respectively. This way, which ports are for what purposes is completely clear both at the desk and in the wiring closet.

Documenting Patch-Panel Changes

There are two good methods for documenting patch-panel changes. One method is to have every change accompanied by a work order. The technician notes the changes on the work order. Then, when the work order is closed, the changes can be input into a database or spreadsheet of cabling information. The other method, which is probably more effective in most organizations, is to simply have a printed spreadsheet of the patching information taped to the inside of the cabinet door. When a technician makes a change, he immediately notes it on this sheet of paper. Then somebody needs to gather up the paper sheets periodically, input the changes, and print out new paper sheets to tape back inside the cabinets. The principle advantage to this method is that not all changes are accompanied by work orders. In particular, emergency changes usually have to be done quickly by whoever is on call. This person may not have time to access the work-order system and update databases with the small changes that they had to make to fix the problem.

If you later found that you had to run an additional data cable to this desk, you could number it 99-C. An additional voice line, perhaps for a fax machine, could be numbered 99-W.

These designations are merely intended as examples. Every network is different, so the network designer has to come up with a locally appropriate scheme.

Giving consistent labels to the patch cords that connect to these patch panels is also important. There are many different ways of doing this. Some organizations like to label the patch cord with a tag that indicates what is on the other end. For example, suppose that a cord connects panel 1, port 2 to panel 2, port 3. Then the first end plugs into panel 1, port 2, and it has a label saying "panel 2, port 3." This method is quite common, and it is generally not very good. The problem is that, at some point, somebody will need to move that patch cord. If they fail to change the label in the heat of the moment, then they will have a situation that is worse than having no labels at all because the labels cannot be trusted.

The simplest and most effective method for labeling patch cords that I have seen is simply to give every patch cord a unique number. These cables can be prenumbered and left in convenient locations in each wiring closet. Whenever a patch cable is connected between two ports, the technician writes the information on the sheet inside the cabinet. Patch panel 1, port 2 connects to patch cord number 445, which connects to panel 2, port 3. This system greatly facilitates the process of auditing patch panels. All that you need to do is go through the patch panels port by port and write

down what patch-cord number connects to that port. Then you can put all of this information into a spreadsheet and sort it by patch-cord number to see all of the cross-connections immediately.

If the spreadsheets get badly outdated, and there is an emergency problem involving a particular port, then the technician will have to trace the cable manually regardless. An effective labeling scheme will be of no help if the spreadsheets are outdated.

Having universal rules for what constitutes a logical patch-panel layout is difficult. This is because what is logical depends on how the cabling is used. For example, suppose every user workstation has two LAN drops, labeled A and B. The first drop is for data and is connected to a computer. The second drop is for an IP telephone. In this case, it makes sense to separate the patch panels to put all drop As together in one group of panels and all drop Bs together in another group. Alternatively, if all drops are intended for user workstations and many users simply have two workstations, then grouping the A and B drops together may be simpler. In this case, the pattern might even alternate A and B on the same patch panel.

What is universal is that the patch-panel layout should make finding devices easy. Usually, workstations are numbered logically through the user work area. Consecutive numbers should indicate roughly adjacent workstations. Then the ports on the patch panel should be arranged in numerical order. In this way, it becomes easy for the technician who usually deals with cabling in this wiring closet to look at the patch panel and know at least approximately where the corresponding workstations are.

However, even with the best of initial intentions, the pattern can be badly broken over time. This is because you frequently have to deal with changes to the work area. Sometimes a cubicle pattern may change on the floor, and sometimes you need to run extra cabling to support extra devices. The network designer and manager have to tread a very careful line between forcing these changes into the logical flow of the entire area and wanting to minimize changes to unaffected areas. This situation usually means that any new drops are taken as exceptions and put at the end of the existing group of patch panels.

One of the most important considerations in designing a network to be manageable is deciding how and where to connect the network-management equipment. Is there a separate network-management center to accommodate? Do nonoperational staff members like the network designer sit in a different area? Do they require access to the network-management center's equipment through the network?

In general, the design should include a separate VLAN just for network-management equipment. This VLAN is not necessarily the same one mentioned earlier. That management VLAN was used to access management functions on remote network equipment. This network management–equipment VLAN houses servers and workstations used to manage the network.

This VLAN is usually as close to the Core of the network as possible. However, it is not always close to the Core. Many organizations are opting to outsource their network-management functions. This outsourcing permits highly trained staff to be available at all hours. It also means that network management must be done from offsite, usually from behind a firewall.

SNMP

No discussion of network management in IP networks would be complete without including the Simple Network Management Protocol (SNMP). I want to stress that SNMP is primarily used for fault management and, to a lesser extent, for configuration and performance management. It is definitely not the only tool required for a complete network-management system, but it is an important one.

SNMP is a UDP-based network protocol. It has been adapted to run over IPX, as well as IP. However, IP is by far the most common network protocol for SNMP.

SNMP has three general functions. It can request information from a remote device using a get command. It can be used to configure the remote device with a set command. Or, the remote device can send information to the network-management server without having been prompted, which is called a trap. A trap is usually sent when there has been a failure of some kind. In general, a trap can be sent for any reason deemed useful or appropriate for this particular device. However, the main application alerts the network-management server of a failure of some kind.

In general, two types of devices speak SNMP. The remote device that is managed has a relatively small engine called an SNMP *agent*. The agent is a piece of software that responds to get and set packets. It also monitors the functioning of the device it runs on and sends out trap packets whenever certain conditions are met.

The other general type of device is the SNMP *server*. This server is typically a relatively powerful computer whose only function is to monitor the network. The server polls remote devices using get and set commands. The IP address of the server is configured in the remote agents so that they will know where to send trap messages.

Many network engineers prefer not to use SNMP for configuration. This is because they believe there are too many serious security problems with the model, making it relatively easy to attack and reconfigure key pieces of network equipment. These problems can make configuration much more difficult. However, if there is a security concern, then turning off SNMP write access on your network devices is worthwhile.

There are several commercial SNMP server systems. They usually come with a number of complex features such as the ability to discover and map the network and display it graphically. Almost all modern network equipment includes an SNMP agent, at least as an optional feature.

The amount of information that can be exchanged with SNMP is enormous. Every device that has an SNMP agent keeps track of a few basic variables that the server can query with get commands. Thousands of other optional variables are appropriate for different types of devices. For example, a router with a Token Ring interface allows the server to poll for special parameters that are relevant to Token Ring. If this router doesn't have any Ethernet ports, then it doesn't make sense for it to keep track of collisions, since there will never be any. However, it does need to keep track of beacon events, for example.

This same router also has a number of special-purpose variables that are unique to this type of equipment and this particular vendor. All of these different variables are accessed by a large tree structure called the Management Information Base (MIB). People talk about "the MIB" and different vendor-specific "MIBs." However, it is all one large database. The only difference is that some parts of it are used on some types of devices, some parts of it are defined by particular hardware vendors, and others are globally relevant. Thus, I prefer to talk about vendor- or technology-specific "MIB extensions."

Every network-hardware vendor has its own set of MIB extensions. These extensions allow different vendors to implement special customizations that express how they handle different interface types, for example. They also allow the different vendors to give information on things such as CPU load and memory utilization in a way that is meaningful to their particular hardware configuration.

Three different revisions of SNMP are currently in popular use—SNMP-1, 2, and 3. The differences between these revisions are relatively subtle. They primarily concern factors such as security. The important thing is to ensure that your SNMP server understands to which version of SNMP the agent on each device expects to speak. Most networks wind up being a hybrid of these different SNMP flavors.

How to Monitor

In general, a network is monitored with a combination of polling and trapping. Devices are polled on a schedule—every few minutes, for example. But you need a way to determine if something bad has happened in between polls. This requires the device to send trap packets whenever important events occur. On the other hand, traps alone are not sufficient because some failures prevent the remote device from sending a trap. If the failure you are concerned about loses the only network path from the remote device to the network-management server, then there is no way to deliver the trap. Thus, failures of this type can only be seen by polling, so any successful network-management system always uses a combination of polling and trapping.

Setting an appropriate polling interval is one of the most important network-management decisions. You want to poll as often as possible so that you will know as soon as something has failed. Polling a device too frequently can have two bad side effects.

First, polling too often, particularly on slow WAN links, has the potential to cause serious bandwidth problems. For example, suppose each poll and each response is a 1500 byte packet. Then, each time you poll, you send a total of 3000 bytes through the network. If you poll each of 100 different remote devices all through the same WAN serial interface (a common configuration in Frame Relay networks), then each poll cycle generates 300 kilobytes of traffic. Therefore, if you poll each of these devices once every 30 seconds, then this generates an average of 10kbps on the link just because of polling traffic.

These numbers are relatively small, but in a large network they can become large very quickly. If instead of polling 100 devices, you have a network with 100,000 devices, then that 10kbps becomes 10Mbps. This increase will cause a noticeable load on even a Fast Ethernet segment.

The second problem with a short polling interval, however, is much more dangerous. Consider the example of 100 remote devices again. Suppose one of these devices is not available. The usual prescription is that the server will try three to five times, waiting a default timeout period for a response. The default timeout is usually between 1 and 5 seconds, so the server will have to wait between 3 and 25 seconds for this device before it can move on to the next one in the list. As a result, if there are several simultaneous problems, or a single problem affects several downstream devices, the management server can get stuck in its polling cycle. When this happens, it spends so much time trying to contact the devices that are not available that it loses the ability to monitor the ones that are still up effectively.

A number of different SNMP server vendors have come up with different ways of getting around this polling-interval problem. Some vendors allow the server to know about downstream dependencies—if a router fails, then the server stops trying to contact the devices behind it.

Another clever method for dealing with the same problem is to break up the queue of devices to be polled into a number of shorter queues. These shorter queues are then balanced so that they can poll every device within one polling cycle even if most devices in the list are unreachable. The most extreme example of this is when the queues contain only one poll each. This means that all polling is completely asynchronous, so no failure on one device can delay the polling for another device. This situation loses some of the efficiencies of using queues, however, and may consume significantly more memory and CPU resources on the server. Some servers can use some variation of both methods simultaneously for maximum efficiency.

Whether the server discovers a problem by means of a poll or a trap, it then has to do something with this information. Most commercial network-management systems

include a graphical-display feature that allows the network manager to see at a glance when there is a problem anywhere on the network. This idea sounds great, but in practice, it is less useful than it appears. The problem is that, in a very large network, a few devices are always in trouble. So the network manager just gets used to see a certain amount of red flashing trouble indicators. To tell when a new failure has really occurred, it is necessary to watch the screen for changes constantly. Constantly watching the screen can strain one's eyes, which tend to get sore from such activities, so network managers have different methods for dealing with this problem.

Some people don't look at the map, but look at a carefully filtered text-based list of problems. This list can be filtered and sorted by problem type. It is even possible to have these text messages sent automatically to the alphanumeric pagers of the appropriate network engineers.

Another popular system is to use the network-management software to open trouble tickets automatically. These tickets must be manually verified by staff on a help desk. If they see no real problem, they close the ticket. If they do see a real problem, then they escalate appropriately.

Any combination of solutions like this should work well, but beware of network-management solutions that are purely graphical because they are only useful in very small networks.

SNMP monitoring has many uses. Until now I have focused on fault management. But it can also generate useful performance-management data. For example, one of the simplest things you can do is set up the server to simply send a get message to find out the number of bytes that were sent by a particular interface. If this poll is done periodically—say, every five minutes—the data can be graphed to show the outbound utilization on the port. In this way, you can readily obtain large historical databases of trunk utilization for every trunk in the network. Usually, the only limitation on this sort of monitoring is the amount of physical storage on the server.

Besides port utilization, of course, you can use this method to monitor anything for which there is a MIB variable. You can monitor router CPU utilization, dropped packets, and even physical temperature with some device types.

Another interesting, underused application of network-management information is to have automated processes that sift through the trap logs looking for interesting but noncritical events. For example, you might choose to ignore interface resets for switch ports that connect to end-user workstations. End users reboot their workstations frequently, so seeing such an event in the middle of the day is usually considered an extremely low priority. The network manager generally just ignores these events completely. But what if one port resets itself a thousand times a day? If you ignore all port-reset events, you will never know this information. This problem is actually fairly common.

It is a good idea to have automated scripts that pass through the event logs every day looking for interesting anomalies like this. Some organizations have a set of scripts that analyze the logs every night and send a brief report to a network engineer. This sort of data can provide an excellent early warning of serious hardware or cabling problems. In fact, these reports highlight one of the most interesting and troubling aspects to network management. The problem is almost never that the information is not there. Rather, there is usually so much information that the server has to ignore almost all of it.

In a modestly sized network of a few thousand nodes, it is relatively common to receive at least one new event every second. A human being cannot even read all of the events as they come in, much less to figure out what problems they might be describing. Instead, you have to come up with clever methods for filtering the events. Some events are important. These events are passed immediately to a human for support. Other events are interesting, but not pressing, and are written to a log for future analysis. Other events are best considered mere noise and ignored.

The most sophisticated network-management servers are able to correlate these events to try to determine what is actually going on. For example, if the server sees that a thousand devices have suddenly gone down, one of which is the gateway to all others, then it is probably the gateway that has failed.

The server can in principle do even more clever event correlation by examining the noncritical events. For example, it might see that a router drops packets on a particular interface. There are many reasons for dropping packets. Perhaps there is a serious contention problem on the link. If, at the same time, the free memory on the same router is low and the CPU utilization is high, then this router is probably not powerful enough to handle the load. Perhaps this router has been configured to do too much processing of packets in the CPU instead of in the interface hardware. If the dropped packets occur when the router receives a large number of broadcast packets, then it may be a broadcast storm and not a router problem at all.

Setting up this sort of sophisticated event correlation can be extremely difficult and time consuming. Some relatively recent software systems are able to do much of this correlation out of the box. They tend to be rather expensive, but they are certainly more reliable than homemade systems.

What to Monitor

In general, the network manager needs to monitor key devices such as switches and routers to see if they work properly. The simplest and most common sort of polling is the standard ping utility. Since every device that implements the IP protocol has to respond to ping, this is a good way to see if the device is currently up. In fact, a few devices, particularly firewalls, deliberately violate this rule for security reasons. However, if a device has disabled ping responses for security reasons, it will probably have SNMP disabled as well, so it has to be managed out-of-band anyway.

Ping is really not a great way to see what is going on with the device. If the device supports SNMP at all, it is better to ask it how long it has been up rather than simply ask whether it is there. This way, you can compare the response with the previous value. If the last poll showed that the device was up for several days and the current poll says that it was up for only a few minutes, then you know that it has restarted in the meantime. This may indicate a serious problem that you would otherwise have missed. The SNMP MIB variable for up time is called *sysUpTime*. It is conventional to call the difference between the current value and the previous value for the same parameter on the same device delta.

Table 9-1 shows several different standard tests that are done by a network-management system. Some of these tests, such as the *coldStart*, *linkUp*, and *linkDown* events, are traps. Note that it is important to look even for good events such as *linkUp* because the device may have an interface that flaps. In this case, the traps saying that the interface has failed may be undelivered because of that failure.

Table 9-1. Standard set of items to monitor

Parameter	MIB variable	Test	Comments
Reachability	ICMP (not SNMP)	Time > N, % not responded	All devices, including those that don't support SNMP
Reboot	coldStart	Trap	Indicates that the SNMP agent has restarted
Uptime	sysUptime	delta < 0	Number of seconds since the SNMP agent started running
Interface Status (for every active Interface on the device)	ifOperStatus	delta ! = 0	Shows that the status of the interface has changed
	ifInOctets	Record	The number of bytes received
	ifInDiscards	delta > N	Incoming packets that had to be dropped
	ifInErrors	delta > N	Incoming packets with Layer 2 errors
	ifOutOctets	Record	The number of bytes sent
	ifOutDiscards	delta > N	Outgoing packets that had to be dropped
	ifOutErrors	delta > N	Outgoing packets sent with errors (should always be zero)
	ifInNUcastPkts	delta > N	Incoming multicast and broadcast packets
	ifOutNUcastPkts	delta > N	Outgoing multicast and broadcast packets
	linkDown	Trap	Indicates that an interface has gone down
	linkUp	Trap	Indicates that an interface has come up

This set of variables tells the network manager just about everything she needs to know for most types of devices. Many other important MIB variables are specific to certain technologies, however. For example, parameters such as CPU and memory utilization are important, but these parameters are different for each different hardware vendor. Consult the hardware documentation for the appropriate names and values of these parameters.

For routers, one is usually interested in buffering and queuing statistics. Again, this information is in the vendor-specific MIB extensions. There are also certain technology-specific MIB extensions. For example, an 802.3 MIB extension includes a number of useful parameters for Ethernet statistics. Similarly, there are useful MIB variables for Token Ring, ATM, T1 circuits, Frame Relay, and so forth. In all of these cases, it is extremely useful to sit down and read through the MIB, looking at descriptions of each variable. In this way, one can usually find out if there is a convenient way to measure particular performance issues or to look for particular fault problems that may be unique to the network.

Ad Hoc SNMP

All of the SNMP polling I have discussed so far has been periodic and scheduled. However, the same SNMP server software can also do ad hoc queries. This means that the network manager can use the system to generate a single poll manually. This can be an extremely useful tool for fault isolation and troubleshooting. For example, this facility can quickly query a set of different devices to see which ones have high CPU loads, errors, or whatever you happen to be looking for. Using this facility is usually much faster than logging into all of these devices manually and poking around on the command line. In fact, the network-management software for many types of hardware makes it possible to do a large list of standard ad hoc queries on a device automatically.

Many hardware vendors make SNMP software called *instance managers*. This software gives a relatively detailed, graphical view of the complete state of one device all at once. Usually, these instance managers also provide the ability to make configuration changes via SNMP as well.

For Ethernet switches, it is often true that the instance-manager software is the fastest and most efficient way to do basic configuration changes such as manipulating VLAN memberships.

This topic actually brings up one of the most serious issues with SNMP. With SNMP Version 1 and, to a lesser extent, with Versions 2 and 3, it is remarkably easy to subvert the security. It is not difficult to load publicly available SNMP server software onto a PC. This software can then be used to reconfigure key pieces of network equipment.

Even nonmalicious users and applications can cause problems. For example, some ill-behaved server-management software automatically attempts to discover the network path to the remote managed-server devices. In doing so, this software generally does detailed SNMP polling of key network devices. Once the path is discovered, this software then periodically polls these network devices as if it were managing the network instead of just the servers.

This situation is not in itself a problem because the server-management software is only polling and not actually changing anything. Remember that the SNMP agent running on a router or a switch is a CPU-bound software process. It uses memory from the main memory pool. If a server program repeatedly polls this device, requesting large parts of its MIB, it can overload the device's CPU. Many such programs all requesting this data at the same time can cause network problems.

As I have stressed repeatedly throughout this book, there is no reason for any end device to ever have to know the topology of a well-built network. It may be necessary in these cases to implement access controls on the SNMP agents of key devices. These access controls have the effect of preventing the agent from speaking SNMP with any devices other than the officially sanctioned SNMP servers.

Some network engineers go further and actually block SNMP from passing through the network if it does not originate with the correct server. However, this measure is extreme. There may be well-behaved applications that happen to use SNMP to communicate with their well-behaved clients. In this case, the network should not prevent legitimate communication.

Automated Activities

SNMP allows much flexibility in how network managers deal with the network. They can set up the network-management server to automatically poll a large list of devices on a schedule looking for well-defined measures of the network's health. If the results are within the expected range of results, then the server concludes that this part of the network works properly. If the result is outside of the expected range, then the server treats it as a problem and somehow prompts an engineer to investigate further. As a rule, it is best if these noninvasive monitoring and polling activities are done automatically without requiring any user intervention. It is a routine repetitive task—exactly the kind of thing that computers are good at. The server can do many different types of things automatically. It is particularly useful to download a copy of the configuration information for every device in the network. This downloading is usually scheduled to execute once per night or once per week in networks that seldom change.

Sometimes a network device fails completely and needs to be replaced. When this happens, it is necessary to configure the new device to look like the one replaces. If the network-management server maintains an archive of recent software configurations for all network devices, then this task is relatively easy.

Another good reason to maintain automatic configuration backups is to note changes. For example, many organizations automatically download the configurations from every router and every switch each night. They then run a script that compares each new image to the previous night's backup. This information is encapsulated into a

report that is sent to a network engineer. Usually, the only changes are the ones the engineer remembers making. But a report like this can be an extremely useful and efficient way to discover if somebody has tampered with the network.

All of these fully automatic processes are completely noninvasive. Some organizations also use a noninvasive suite of test scripts. These test scripts are executed automatically if the network-management software sees a potentially serious problem. The result of these automated tests can be helpful in isolating the problem quickly.

Sometimes network managers want to partially automate invasive procedures as well. For example, it is relatively common in a large network to have a script automatically change login passwords on routers. This way, every router can be changed in a single night. With any sort of invasive procedure, it is usually wise only to partially automate it. A user should start the script and monitor its progress. That person should then verify that the change is correct.

Some network managers go further and allow full automation of invasive procedures such as scheduled VLAN topology or filtering changes. In some cases, invasive scripts are run to reconfigure network devices automatically in response to certain failures. I do not recommend this type of automation; it is simply too dangerous in a complex network. Usually, the automated change assumes a well-defined starting point. However, it is possible to have an obscure problem in the network create a different initial configuration than what the automated procedure expects. This configuration could cause the scripted changes to give unexpected, perhaps disastrous, results. It is also possible to have an unexpected event while the reconfiguration is in progress. In this case, the network might be left in some strange state that requires extensive manual work to repair.

A large network is generally such a complex system that it is not wise to assume that it will behave in a completely predictable way. There are simply too many different things that can happen to predict every scenario reliably. If weird things happen, you should be in control, rather than allow a naïve program to continue reconfiguring the network and probably make the problem worse.

Management Problems

A number of design decisions can make network management more difficult. This doesn't necessarily mean that you should avoid these features, but it does mean that you need to be aware of their implications. It usually also means that you need to devise ways of working around the management problems that you create.

For example, sometimes parts of the network are hidden from a protocol, as in a tunnel, for example. If an IP tunnel passes through a number of devices, then it becomes impossible to see the intermediate devices in-band. If there is a problem in an intermediate device, and if there is no external way to observe that device, then it is impossible to tell which intermediate device has a problem, much less what the problem is.

In the example of tunnel-hiding intermediate devices, the most obvious workaround is to provide out-of-band access. This may mean something as simple as IP addressing in a different range. Or, it may require something as complex as putting modems on serial ports for the inaccessible devices.

Besides architectural features, certain network applications and protocols can create management problems. Again, I don't necessarily advise avoiding them, but the designer should be aware of the problems and provide alternatives.

DHCP

Dynamic Host Configuration Protocol (DHCP) is a system that allows end devices to learn network information automatically. In its minimal form, DHCP allows end devices to acquire IP addresses dynamically, while learning the correct netmask and default gateway. However, other important pieces of network information can also be conveyed by DHCP. For example, DHCP can tell the end device about its time zone, as well as the addresses for time servers (NTP), name servers (DNS), log servers, printers, and cookie servers. It can specify various network parameters such as timer values and MTU values. Literally dozens of different kinds of information can be conveyed by DHCP. It even has some open fields that convey special vendor-specific application parameters. For all of these reasons, DHCP can greatly assist in the management of end devices. The device can be set up anywhere in the network, and it will automatically discover the correct DHCP server and learn everything it needs to know to use the network.

One problem with DHCP, however, comes from its ability to assign addresses out of a pool. The first device to be turned on in the morning gets the first address, the second device gets the second address, and so on. This is by no means the only way to configure a DHCP server. It can also be set up to look for the end device's MAC address and give out a unique predetermined set of parameters that will always be associated with this device. But a simple dynamic assignment from a pool of addresses is frequently used because it is easy to implement. The problem with doing this is that there is often no easy way to determine which device has a particular IP address. This situation can be corrected by linking the DHCP server to the DNS server. When the DHCP server gives out a particular IP address, it informs the DNS server to which device it assigned this address. Then there is a simple association between the device's name and address.

However, even with a linking between DNS and DHCP, it can be difficult to do some types of fault isolation when IP addresses are assigned from a dynamic pool. In particular, when looking at historical records correlating IP addresses with actual devices can be difficult. This correlation becomes a problem when, for example, a server records in its logs that it has had network problems associated with a particular IP address. It can be extremely difficult to reconstruct which actual device this was. The only solution is to ensure that the DHCP server keeps a reliable record of

historical data. It must be possible to determine which end device had a particular IP address at a particular point in time. This data has to be reliable at least as far back in history as any other logging information.

When DHCP is configured to give addresses from a pool as they are required, it often creates confusion on network-management servers, even if there is a good link between DNS and DHCP systems. These servers tend to maintain large databases of every device in the network. This information is usually discovered automatically by periodically polling ranges of IP addresses. If a particular IP address is associated with a particular DNS name when the device is discovered, the network-management software records that association permanently in its database. Then, at some later time, it may record an error associated with that IP address. However, it will often report this error as being associated with the previous DNS name, which is no longer accurate.

Some network-management software provides methods to work around this problem. It is possible simply to provide a mechanism to look up the names dynamically each time they are required, for example. However, remember that this is not usually the default configuration and that there is potential for confusion.

This first problem can be mitigated somewhat by setting the DHCP lease time to be extremely long. This setting allows each device to receive the same IP address each time it reboots. If the lease time is sufficiently long, the addresses become effectively static.

Another problem with using DHCP is its actual operation in a large network. In many networks, it is common practice to tie a particular end device's configuration information to its MAC address. This method is useful, but it means that this information must be maintained carefully. If the Network Interface Card (NIC) in the end device is changed because of a hardware problem, then the DHCP database must be updated. Similarly, if this end device is moved to another location, the DHCP database has to reflect this new information as well.

These situations are not really problems, but rather facts of life in this sort of implementation. However, they do represent a significant amount of work that is required each time maintenance work is done on an end device.

Architectural Problems

Some types of architecture can result in network-management challenges. By architectural problems I do not necessarily mean that these architectural features are bad. In fact, some of these features, such as firewalls and VLAN trunks, are extremely useful. We would not want to do without them. When we use them, though, we have to ensure that there are ways around the management difficulties. This section discusses some of these problems and suggests some solutions.

VLAN structures

Most modern LANs use VLANs and trunks. There are too many advantages to these features to avoid them. However, you should be careful about how you monitor trunks. A trunk link that contains many different VLANs treats all of these VLANs as a single stream of traffic. Consequently, if there is a physical failure, it takes out everything. However, there are two basic ways to implement the Spanning Tree protocol in a VLAN trunk. In the most common configuration, the whole trunk is replaced by a redundant trunk in case of a failure. But some vendors have features that allow Spanning Tree to operate on each VLAN separately. The principal advantage to this approach is that the backup link is configured to take some of the load during normal operation. However, determining which VLANs are using which trunks can be very difficult. Thus, if a problem involves a few VLANs is discovered, it might take a long time to determine that all affected VLANs happen to traverse the same trunk at one point in the network.

Conversely, the design could employ a system in which each trunk has a backup that is unused except when the primary fails. In this case there is the danger of suffering a secondary trunk failure and not noticing the failure because it has not affected any production traffic.

The best way around both of these problems is simply to provide the network-management system with a detailed view of the VLAN and trunk status for every switch. Furthermore, since most problems that occur will be physical problems of some sort, it is important to maintain physical monitoring of all trunk ports on the switch. This monitoring is particularly critical for trunk backup ports because they do not pass traffic. Thus, you have to rely on the switch to tell you when there is a problem.

For all higher-layer problems, it is useful to have protocol analyzers available to monitor the flow of traffic through the trunks. These devices are usually too expensive to deploy on every trunk. It is often possible to set up a system to allow probes to be patched manually into the required location quickly.

In general, there are several issues to consider when managing VLAN structures. Some hardware vendors provide useful software that allow the manipulation and configuration of VLANs. Individual ports can be readily moved from one VLAN to another. This movement is useful, but configuration management is only part of what the network managers need to do. They also need to do fault and performance management on all switches and trunks.

This management requires a system that allows you to readily determine where a given end device MAC address is connected. If you look in the MAC address tables of the switches, every switch that supports the right VLAN knows about the device. But if you have to locate it by following trunks from one switch to the next, it can be extremely time consuming. Some software can make this easy, but it shouldn't be asssumed.

There also needs to be a method for monitoring trunk traffic. This means both the gross trunk utilization and the per-VLAN portions of that overall utilization. The total trunk utilization is important because it indicates when it is time to upgrade the trunks. It also shows where trunk congestion occurs in the network. The network manager also needs to know exactly how much of each trunk's capacity is consumed by each VLAN. Knowing this shows which groups of users are actually causing the congestion problems. Then you can decide if, for example, they should be moved onto a new trunk of their own to prevent their traffic from interfering with other user groups.

This per-VLAN utilization is somewhat harder to determine. A good protocol analyzer can do it, and some switches include sufficiently powerful probes to do this sort of analysis.

LAN extension

LAN extension is a general term for providing a Layer 2 LAN protocol over a larger distance. This provision might be handled with dark fiber and a few transceivers and repeaters. Or, it could be implemented using a LAN bridge through some wide-area technology such as an ATM network.

The reason why LAN extension represents a management problem is that the actual inner workings are usually hidden from view. For example, one particularly common implementation of a LAN extension is to use RFC 1483 bridging. This simple protocol allows encapsulation of all Layer 2 information in ATM. The customer of this sort of service sees only a LAN port on either end of an ATM PVC link, which makes it possible to deliver what looks like a Fast Ethernet connection between two different cities, for example. The problem is that there is no easy way to determine if a problem exists in this link. All internal workings of the ATM network are hidden from view. All the customer's network-management software can see is an Ethernet port on either end of the link.

Ethernet link always remains up because the Ethernet signaling is provided by a port on an ATM/Ethernet switch that is physically located on the customer premises. Thus, there is no way to receive a physical indication of a failure.

The only way to work around this management problem is to configure the network management software to poll through the LAN extension links periodically. Doing this configuration requires a detailed understanding of the PVC structure within the ATM network.

Figure 9-1 shows an example ATM LAN-extension configuration. In this example, one central site talks to each of three different branch sites. To the device shown in the main site, all three remote devices appear to be simply on the same Ethernet segment.

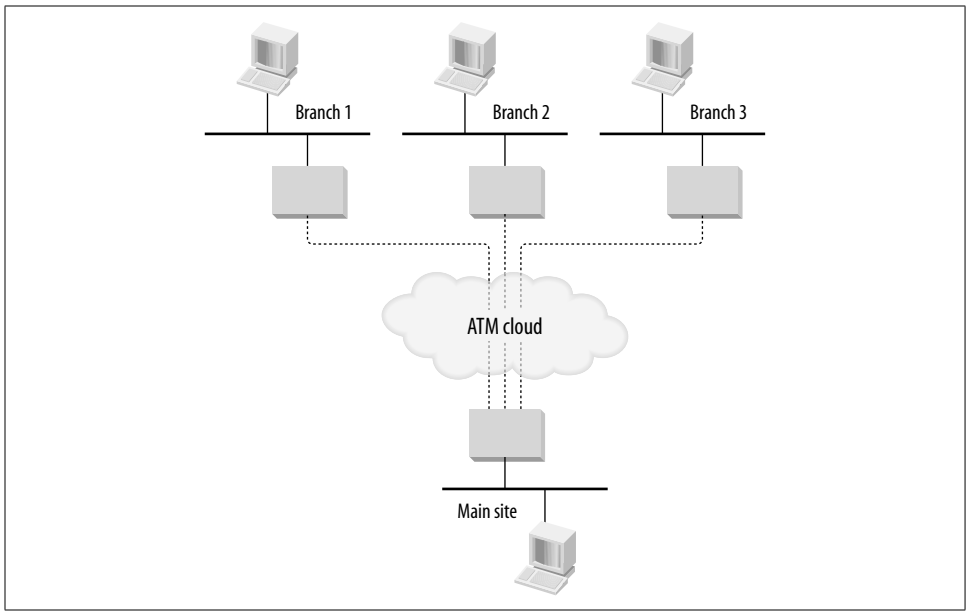

Figure 9-1. Managing a LAN-extension network

Now suppose that you suffer a failure in the ATM cloud that knocks out the PVC to Branch 1. The other two devices look fine, but you have lost contact with the first device. Most critically for network management, however, the server has not received a trap of any kind for this failure. In fact, it is almost impossible to issue a trap on this sort of failure. The only way to verify that the link is still available is to continuously poll through that link to the other side. This polling can be either SNMP or ping. Of course, even with this sort of active polling through the links, the only indication of trouble is a complete loss of the remote site. Many things could cause such a failure; power or cabling problems within the remote site can cause the same symptoms.

The best you can do is to detect that there has been some sort of problem. More troubleshooting is needed before you can conclude that there was a PVC failure in the ATM cloud.

Filtering

Another common feature in networks—one that I have advocated earlier in this book—is filtering. You can filter traffic, or you can filter routing information. In IPX, you can also filter SAP information.

Filtering represents serious potential problems for network management. In particular, if there are traffic filters, you should be sure that the network-management traffic is still allowed to pass.

There are cases in which stringent filters have been implemented for security reasons. For example, an application vendor might need to place a semitrusted server on the inside of a customer network to deliver the service. The customer might react to this situation by placing the server behind a router with strict filtering to allow only the desired application to pass through. The problem is that, at some point, there will be a problem and somebody will need to troubleshoot. If the router filters out all traffic except application traffic and the application is not working, then the network engineer is left with no way of testing. Is the server hung? Is the application broken? Or, is there perhaps a problem with its network connection? There has to be a way to verify network connectivity to the server, and this usually means ping.

For this reason, wherever traffic filters are employed, simple ICMP (ping) packets should be permitted along with the application. This way, the network-management system can at least determine if there is basic connectivity. What you lose in security, you more than make up for in the reliability that comes from swift problem analysis.

In IPX networks, SAP information is frequently filtered separately from route information. This filtering can cause a relatively common problem. It is possible for the route and SAP filters to be different in a subtle way. Either the SAP or the route is visible, but not both. When both are not visible, the network-management staff must be able to track the flow of routing and SAP information through the network. Remember to follow the whole round trip. SAP information flows from the server out to the end devices. If the end device can see the server in its server list, then the SAP must have arrived safely. There is generally no need in general for workstations to send SAP information back to the server.

Then you have to follow the routing information. Routes must exist on the workstation end that points to the server; routes must exist on the server end that point to the workstation.

Firewalls

The most extreme form of filtering is a firewall. A firewall is always called for in any location where devices on one network must communicate with devices on another untrusted network. In general, no routing information flows through firewalls. They are usually configured only with static routes. If any dynamic routing capabilities are available within a firewall, they should be restricted to BGP-4.

It can be extremely difficult to manage devices on the other side of a firewall. The typical configuration involves putting a network-management server inside of the firewall and the device to be managed on the outside of the firewall.

Firewalls are generally set up to allow just about anything to pass from the inside to the outside, but they strictly block inbound traffic. If you ping the device on the inside from the outside, you generally get no response. If you instead ping something on the outside from the inside, it usually works because the firewall knows to expect the returning ping response.

Let's look at this process in a little more detail. The network-management server sends out some kind of poll packet. For the time being, suppose that it is a ping request packet. This packet is received by the firewall. Most firewalls translate the IP source address of outbound packets. Instead of having the real network-management server's IP address, the packet has the firewall's external address when it is sent along. The external device receives this poll packet and responds. It creates a ping response packet and sends it to the firewall's IP address. The firewall has been waiting for this particular device to respond. It remembers that it passed through a ping request for this device that originated with the network-management server. Thus, it changes the destination address in the packet to the network management server's IP address and delivers the packet. If the external device had sent this packet without being prompted, the firewall would not know how to forward it internally, so it would simply drop it.

Now suppose the network manager needs something more sophisticated than ping. The firewall can be configured to pass SNMP packets, so the same pattern follows. The network-management server sends a packet. The source address is translated and the packet is delivered to the external device. The external device responds and sends the packet back to the firewall, which forwards it back to the server. Everything works well. But what about traps? SNMP traps are a critical part of the whole network-management system, but these traps are never prompted by a request. So how does the firewall know where to forward the inbound packets?

Many firewalls have the ability to define special inbound addresses. In effect, the outside of the firewall appears to have several different IP addresses. One of these addresses is configured to correspond to the network-management server. As long as the external device forwards its trap packets to this special address on the outside of the firewall, it is possible to deliver the addresses.

Alternatively, it is possible on many firewalls to deliver unexpected inbound packets based on their port number. SNMP has a well-known UDP port number of 161 for polls and poll responses, and 162 for traps. It is easy to ensure that all inbound SNMP traffic is forwarded to the network-management server.

Another interesting problem occurs when managing networks through firewalls. Sometimes the address translation works in the other direction. That is, the network-management server sees translated addresses for everything inside the managed cloud. Some network address–translation devices are capable of doing this kind of wholesale address translation, giving every device a new unique address.

This configuration especially appears in cases in which an external network-management service provider is used. The internal network may contain sensitive information and therefore require protection from the service provider by means of a firewall. The firewall may be configured to pass only SNMP and ICMP packets (and perhaps telnet, FTP, and TFTP for configuration-management purposes) and to translate all IP addresses in the internal network.

This address translation may be used for a good reason. If the network-management service provider manages two large networks, there is a good chance that both of them use the common unregistered `10.0.0.0` address range. If the service provider wants to see both networks properly, they have to do some address translation.

This configuration leads to serious complications, however. Many types of SNMP packets include IP addresses in their data segments. An IP address is just one of many pieces of management information that could be sent. However, this means that the address in the payload of the packet is different from the address in the header of the packet because of address translation. This difference causes serious confusion in many cases. There is no clean workaround. The best way to handle this situation is simply to avoid it. The network-management service provider is advised to maintain a separate, disconnected management server for each client.

In some cases, such as when managing sections of the public Internet, there may be security concerns about allowing SNMP through the firewall. In fact, there are security concerns about using SNMP at all in such hostile environments. Most frequently, the devices that are connected directly to the public Internet have SNMP disabled.

Disabling SNMP presents a serious management problem, however. How can the network manager monitor a device that doesn't use SNMP? As it turns out, a lot of devices, particularly in legacy networks, do not use SNMP. In all of these cases, it is necessary to use out-of-band management techniques. Some of these techniques are discussed later in this chapter.

Redundancy features

Redundancy is one of the most important features of a robust network design. It is also one of the most dangerous because it makes it possible to get away with extremely poor network-management procedures. Suppose, for example, that you have a large network in which every trunk is redundant. If you have a trunk failure anywhere in the network, you suffer no noticeable application failure. This is a good thing, but that broken trunk now needs to be fixed. If you have another failure in the same area, you could have a severe outage. However, if you do not manage the network carefully, you might have missed the failure. After all, the phones didn't ring.

There have been many cases of networks running for years on a backup link because nobody noticed that the primary had failed.

Just as serious, and even less likely to be noticed, is a failure of a redundant backup when the primary was still working properly. Some network managers rely on interface up and down traps that indicate that the backup link or device was activated. This is certainly a good way of telling that the primary has failed, but there is no change of state if the redundant backup systems fail first.

Both of these scenarios reinforce the same point. All systems and links, even redundant backups, should be monitored constantly.

Constant monitoring can be particularly difficult in the case of links that are protected by Spanning Tree. Spanning Tree disables links that are in the backup state. It isn't possible to just ping through these links to see if they are operational. The Spanning Tree protocol does keep track of the status of its disabled links, however. There is an SNMP MIB extension specifically for monitoring Spanning Tree.

The MIB extension is called the *dot1dStp* (for 802.1d Spanning Tree Protocol) defined if RFC 1286. It contains specific entries describing the state of every port, *dot1dStpPortState*. The values that each port can have correspond to the various allowed states: disabled(1), blocking(2), listening(3), learning(4), forwarding(5), and broken(6). Using this SNMP MIB extension should provide all of the necessary information about the health of all redundant Spanning Tree links.

The ideal management technique for these links is to configure automated periodic polling for the states of all Spanning Tree ports using this special `dot1dStpPortState` variable. This information, combined with the traps generated by Link State changes, give a good picture of all primary and backup links.

Tunnels

There are several tunneling protocols. Some, like DLSw, are used to tunnel foreign protocols through IP networks. There are also several ways of tunneling IP in IP.

There are many reasons for tunneling IP in IP. Usually, they have to do with needing to pass transparently through sections of the network that are either externally controlled or lacking in some important feature. A common example of the missing feature problem is a legacy IP network that does not support the required dynamic routing protocol. Similarly, a device might need to take part in multicast applications. If it is located behind network devices that do not support multicasting, then it might be necessary to pass a tunnel through these devices to reach the multicast-enabled portion of the network.

It is also relatively common to use tunnels to hide the network structure of a foreign network that traffic must pass through. For example, it may be necessary to interconnect two buildings by means of a network operated by a telephone company. If the telephone company's network is essentially an IP network, this company might deliver the service as a tunnel to hide their internal network structure.

Another common type of tunnel is the ubiquitous VPN. In this case, an organization extends its private internal network to include a group of devices, or perhaps a single device, on the public Internet. VPN tunnels usually have the additional feature of being encrypted as they pass through the foreign network.

To the network manager, however, tunnels represent a difficult problem. If a failure or congestion occurs anywhere in the hidden region, the only symptoms are either interrupted or degraded service.

It is not possible to narrow down the problem any further than this unless there is another way to see the actual network devices that the tunnel passes through. If the tunnel passes through a foreign network that is managed by another organization, then you can simply pass the problem over to them. For tunnels that pass through internal pieces of network, it is necessary to have an out-of-band management system of some kind.

Out-of-Band Management Techniques

Out-of-band management means simply that user data and management data take different paths. There are many ways to accomplish this. Usually, when people use this term, they mean that the device is managed through a serial port. But it is useful to consider a much broader definition.

Devices are managed out-of-band for three main reasons:

- Many Layer 1 and 2 devices are incapable of seeing Layer 3 addressing.
- Some data streams contain untrusted data, so the devices should not be managed in-band for security reasons.
- Some networks contain tunnels that hide the intermediate devices. These devices must be managed from outside of the tunnel.

First, let's look at the simplest type of out-of-band management. Transceivers, modems, and CSU/DSU type devices are almost impossible to manage in band. This is because these devices function at the physical layer. They do not even see the Layer 3 signaling that would allow them to send and receive SNMP packets. They could be given this capability, but it would require that they look at all frames that pass through them. That generally means that a faster CPU is needed. It can also introduce serious latency problems.

However, many of these types of devices can be managed through a serial port. In fact, in many cases, there is full SNMP (and even RMON, in some cases) support by means of a SLIP or PPP connection through an RS-232 serial port.

Not all lower-layer devices must be managed out-of-band. Many Ethernet and Token Ring hubs are managed in-band, for example. These Layer 2 devices are typically managed by a special purpose management card. This card is connected to the network as if it were an external device, but it lives inside the hub's chassis. In this way, the card can monitor the functioning of the hub without interfering with it.

Security presents another common reason for out-of-band management. The classic example is a router that is connected directly to the Internet. It is dangerous to allow such devices to respond to SNMP gets and sets. The security within SNMP is too simplistic to prevent a dedicated saboteur from modifying the configuration.

Many small organizations with only one router on the public Internet can get away with SNMP management of the device. They can apply access-list restrictions that drop all SNMP packets coming from or going to the Internet. Many organizations also prevent ICMP packets, but these restrictions would not be applied to the internal network. These devices can then be safely managed in-band through the port that faces the private network.

However, this strategy does not work for any organization with several devices connected directly to the untrusted network. If there are several such devices, then it is possible that the topology has become complex, with multiple different paths to the Internet. This complexity makes a simple path-based restriction impractical. Also, if several different devices are all connected directly to the untrusted network, it is possible to compromise one device and then use it as a base. From this base, the other devices can be compromised more easily. Thus, for all but the simplest connections, security restrictions mean that devices directly connected to the Internet should be managed out-of-band.

It can be useful to think of a special variety of management that is only partly out-of-band. This is the case for any tunnel that contains IP traffic. The tunneled traffic does not see any of the intermediate devices. However, these hidden devices can be managed using IP and SNMP through the same physical ports that contain the tunneled data.

For management purposes, there are effectively two types of tunnels. The tunnel can pass through a section of network that has directly accessible IP addressing. Or, the tunnel might pass through devices that cannot be reached in-band from the network-management server.

In the first case, a tunnel might pass through a group of legacy devices. This could be necessary because the legacy devices do not support the preferred dynamic routing protocol, such as OSPF. Or, it may be necessary because they do not support some key part of the IP protocol that is required for the data stream. This might be the case if there are MTU restrictions on the application, or if there are special QoS or multicast requirements. Or, maybe the tunnel is there to trick the routing protocol into thinking that a device is in a different part of the network, such as a different OSPF area. However, in these cases, the tunnel passes through devices that are managed in-band. They are part of the same IP-address range and the same Autonomous System (AS). In effect, these devices are managed out-of-band from the tunnels, but through the same physical interfaces that tunnels use. In this case, the management is essentially in-band.

There are also times when a tunnel passes through a different AS. The devices in the other AS could even be part of a distinct group of IP addresses that the network cannot route to directly. This is where the line between in-band and out-of-band becomes rather blurred.

This construction is relatively common when a network vendor uses an IP-based network to carry the traffic of several different customers. They can get excellent fault tolerance through their networks by using dynamic IP-routing techniques, but they must prevent the different customer data streams from seeing one another. This prevention is easily done by simply passing tunnels through the vendor's network Core. The vendor places a router on each of the customer's premises and terminates the tunnels on these routers. Thus, no customer is able to see any of the Core devices directly, nor even the IP-address range used in it. In fact, the IP-address range in the vendor's Core can overlap with one or more different customer-address ranges without conflict. Everybody can use `10.0.0.0` internally, for example, without causing routing problems.

In this case, however, the customer would not do out-of-band management on the vendor's network. It should be managed by the vendor. I mention this configuration, though, because sometimes an organization must be this sort of vendor to itself. This happens in particular during mergers of large corporations.

In effect, all of these different options come down to management through either a LAN port or through a serial port. Management through a LAN port effectively becomes the same as regular in-band management. The only difference is that it might be necessary to engineer some sort of back-door path to the managed LAN port. However, management through a serial port always requires some sort of special engineering. Serial-port management is usually done in one of two ways. In some cases, a higher-layer protocol can run through the serial port using SLIP or PPP. In most cases, there is only a console application available through this port.

If SLIP or PPP options are available, they can make management of these devices much easier. I recommend using it wherever possible.

Figure 9-2 shows one common configuration for out-of-band management using a SLIP link.* The managed device is not specified, although it could be a CSU, a microwave transmitter, or any other lower-layer device that does not see Layer 3 packets directly.

In this case, a low-speed serial port on the router is configured for SLIP. A serial cable is then connected to the management port on the device. As anybody who has set up such a link will attest, there are many ways to misconfigure such a setup.

* The example indicates the use of SLIP on the serial link, but any other serial protocol, such as PPP or HDLC, would work in exactly the same way. I specifically mention SLIP because it is common for this type of application.

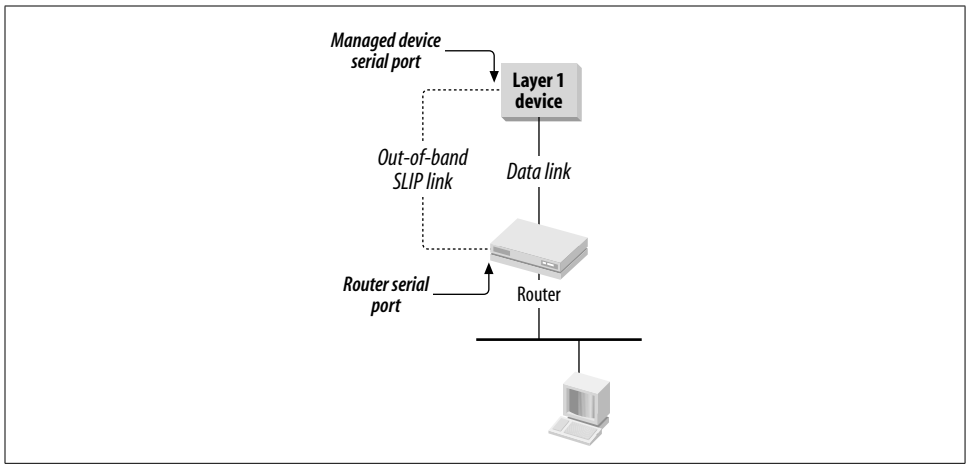

Figure 9-2. Out-of-band management using a SLIP link

Usually, SLIP links assume asynchronous serial connections. Conversely, most router serial ports are synchronous by default. Thus, the router's port must be configured to operate asynchronously.

SLIP creates a point-to-point IP link over the serial line. Therefore, both ends must have IP addresses and the managed device must have its default gateway configured to point to the router's address on this link. Usually, the managed device will not run nor need to run any dynamic routing protocols, so ensure that these are turned off on the router for this port. However, the router needs to distribute the route to this device into the dynamic routing protocol so that the management server knows how to reach it.

In many cases, the managed device uses the same physical serial port for SLIP that it uses for regular console connections. It is important to ensure that the port is configured for SLIP rather than console mode before connecting. This configuration usually just involves connecting a terminal to the console and switching the mode, then disconnecting the terminal and connecting the router in its place.

With any serial connection, make sure that the DTE and DCE relationships are made correctly. This relationship is not just the physical pins on one end the cable being male and female on the other. It also specifies which pins are used for sending and receiving. The relationship becomes more involved for synchronous connections, in which you also have to worry about which device provides the clock signal. In most cases, the manufacturer assumes that the device is talking to a modem. Modems are always DCE, so the serial interface on the device is almost always DTE. Thus, the router must be configured to be DCE.

Once this has all been done properly, it should be possible to do SNMP management of the device. It will have an IP address, and it should respond to ping and SNMP polling. Some of these devices also support telnet to allow a remote-console connection.

For SNMP management, remember to set up the network-management station as the SNMP on the device. This usually does not restrict polling, but rather specifies where the device will send traps when it sees important error conditions. In general, it is a bad idea to specify a large number of trap recipients. One or two should be sufficient.

If too many trap recipients are specified, then each time the device encounters a serious problem, it has to send trap packets to all the devices. Over a slow serial line, sending these packets can take a relatively long time, perhaps as long as a second or more. In a disaster situation, this is a very long time, and it may mean that the device is unable to send the trap to every recipient. It may also mean that the device's CPU is temporarily overloaded by creating and sending traps, which could worsen disaster situation.

For devices that do not support SLIP or PPP, remote out-of-band management can become messy. Somehow, there has to be a console connection between the device's console serial port and the network-management server. If there are more than a few of these devices or if they are physically remote, it is not practical to use direct serial cables. Thus, you have to come up with other methods for making these connections. Once you have these physical connections, you need to have a way to use them automatically. As I said earlier in this chapter, all network monitoring should be automated, and it should have a way to report problems to humans for investigation.

Some network-management software provides the ability to build arbitrary scripts for managing nonstandard devices. If there is a way to connect physically to a device, the software has a way to ask that device about its health. In most cases, the network manager would then automate this script to run every few minutes. If there are no problems, the result is simply recorded in a log. If the device reports an error condition of some kind, it can trigger an alarm to allow a human to investigate.

This fact alone indicates why having hardware standards is so important. If you have a thousand identical devices that you have to manage this way, you can do it all with the same script. You can also afford to take the time to make this script robust and useful. However, if you have a thousand different devices from different hardware vendors, coming up with a thousand such scripts is impractical.

Physically, there are two main ways to create these out-of-band physical connections. For remote locations, the best method is simply to attach a modem to the console port. This attachment allows the device to be contacted for maintenance even if the primary network is down. It is also problematic because the server cannot do a frequent poll of the device's health.

Doing regular polling of serially managed devices requires a way to get this serial data into the network's data stream. A convenient device for doing this is called an inverse terminal server. An inverse terminal server is in many ways very similar to a normal terminal server. In fact, many commercial terminal servers are able to function as inverse terminal servers as well. Some low-speed multiport routers can also be used for this purpose.

A terminal server has a LAN port and one or more serial ports. They were once extremely common, as they provided a way to connect dumb terminals to the network. Each terminal would connect to a serial port on the terminal server. Then, from the terminal server, the user could use a text-communication protocol, such as telnet, to connect to the application server.

An inverse terminal server is similar, except that it makes connections from the network to the serially connected devices, rather than the other way around. Usually, this server works by making a telnet connection to the IP address of the terminal server, but on a special high-numbered TCP port that specifies a particular serial port uniquely.

As Figure 9-3 shows, you can use an inverse terminal server to manage a number of different devices through out-of-band serial connections.

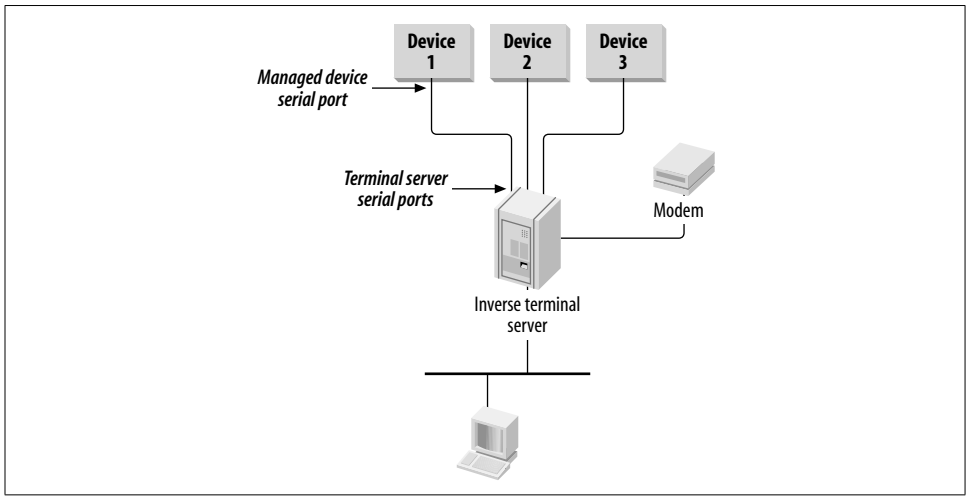

Figure 9-3. Out-of-band management using an inverse terminal server

The network-management server is configured to telnet to the appropriate set of IP address and TCP ports that represent a particular device. It then runs a script that queries the device about its health.

As noted previously, when connecting a serial console port to a router, you have to be careful to configure the DCE and DTE relationships properly. However, unlike the SLIP example, in the case of text-based console ports, there is no common standard for the gender of the console connection.

In some cases, the console port is DCE; in other cases, it is DTE. There are even pathological examples in which the gender of the cable does not match the Layer 1 signaling on the port. For example, it may be a female (DCE) connector physically, but with DTE pin arrangement. In this case, it is necessary to use a null-modem adapter with the cable to convert DCE to DTE instead of just swapping female to male connectors.

In other cases, you need to convert the gender, as well as the DCE/DTE relationship. There are no set rules to make this conversion. Therefore, it is usually a good idea to have a supply of gender-changer plugs and null-modem adapters on hand. To make matters worse, some devices require special cables because they use extra or nonstandard RS-232 signaling. Others do not use RS-232 standards. Consult the documentation of the device being connected.

With many inverse terminal server devices, it is possible to also run SLIP or PPP on the various serial ports. In this way, you can combine several console ports on one inverse terminal server. Since a network-management failure never affects service, it is not necessary to build redundancy. This permits the inverse terminal server to act as a router for several very slow links. The network-management server can then do direct SNMP polling to the out-of-band devices. Of course, if you can do SLIP or text-based console connections through an inverse terminal server, you can do a combination of the two. This configuration can provide a useful way of managing a group of these devices. For example, several CSU devices, firewalls, and other hard-to-manage devices may exist in the same computer equipment room. By running serial cables to a common inverse terminal server, it is possible to provide convenient secure management to all of them at once.

The advantages of remote modem access can be combined with the ability to do periodic polling out-of-band. Most inverse terminal servers provide the ability to connect a modem to one of the ports, as shown in Figure 9-3. This connection allows periodic polling of the console ports of the various serially attached devices through the network. If a serious network failure leaves this part of the network unreachable, the network manager can still get to it by dialing to the modem.

With SLIP-type connectison on an inverse terminal server, you will definitely have a DCE/DTE issue with the ports. As I said earlier, usually the SLIP port on the managed device expects to talk to a modem, so it is usually DTE for the modem's DCE. Those ports on the inverse terminal server connecting to SLIP managed devices will likely be configured as DCE. However, the port that connects to the modem have to be DTE. As always, have a handful of gender changers and null-modem adapters on hand whenever setting up this sort of configuration.

One final issue should be mentioned when connecting modems or terminal servers to a console port. This connection can represent a serious security problem on many devices. Most pieces of network equipment provide the ability to override the software configuration from the console during the boot sequence. This override is frequently called *password recovery*.

Password recovery means that an intruder can take advantage of a power failure to take control of a network device. From there, it might be possible to gain control of other network devices, perhaps even the Core of the network. Gaining control of these devices is a lot of work, requiring a skilled attacker, but it is possible. For this reason, some network-equipment vendors (particularly Cisco) actually include two serial ports, one called Console and the other Auxiliary. The Console port, but not the Auxiliary port, can be used for password recovery. In this case, it is safest to connect any out-of-band management equipment to the Auxiliary port.