CHAPTER
1

Defining Network Performance

Before you dive into detailed discussions of network performance tools, it is a good idea to first understand what network performance is, and how it can be measured. This chapter defines network performance, describes the elements involved in measuring it, and shows techniques many network performance tools use to measure it

The words that network administrators hate to hear are "The network seems slow today." What exactly is a slow network, and how can you tell? Who determines when the network is slow, and how do they do it? There are usually more questions than answers when you're dealing with network performance in a production network environment.

It would be great if there were standard answers to these questions, along with a standard way to solve slow network performance. The open source network performance tools presented in this book can help the network administrator determine the status of the network, and identify the areas of the network that could be improved to increase performance. Often, network bottlenecks can be found, and simply reallocating the resources on a network can greatly improve performance, without the addition of expensive new network equipment.

Knowing the elements of network performance will help you better understand how the network performance tools work, and how to interpret the vast amount of information the tools provide. The first section of this chapter describes the elements involved in determining network performance.

The Elements of Network Performance

Much work has been devoted to the attempt to define network performance exactly. It is not the intention of this book to bore you with numerous equations that describe theoretical network philosophy about how packets traverse networks. Network performance is a complex issue, with lots of independent variables that affect how clients access servers across a network. However, most of the elements involved in the performance of networks can be boiled down to a few simple network principles that can be measured, monitored, and controlled by the network administrator with simple—often free—software.

Most network performance tools use a combination of five separate elements to measure network performance:

- Availability
- Response time
- Network utilization
- Network throughput
- Network bandwidth capacity

This section describes each of these elements, and explains how network performance tools use each element to measure network performance.

Availability

The first step in measuring network performance is to determine if the network is even working. If traffic cannot traverse the network, you have bigger problems than just network performance issues. The simplest test for network availability is the ping program. By attempting to ping remote servers from a client device on the network, you can easily determine the state of your network.

Just about all Unix implementations include the ping program to query remote hosts for availability. The ping program sends an Internet Control Message Protocol (ICMP) echo request packet to the destination host. When the echo request packet is received, the remote host immediately returns an echo reply packet to the sending device.

While most network administrators know what the ping program is, few know that there are lots of fancy options that can be used to perform advanced testing using the ping program. The format of the ping command is:

```
ping [-dfnqrvR] [-c count] [-i wait] [-l preload] [-p pattern] [-s
packetsize]
```

You can use different combinations of options and parameters to create the ping test that best suits your network environment. Often, just using the default options and parameters provides enough information about a network link to satisfy availability questions.

Receiving an echo reply packet from the remote host means that there is an available network path between the client and server devices. If no echo reply packet is received, there is a problem with either a network device or a link along the path (assuming the remote server is available and answering pings).

By selecting different remote hosts on the network, you can determine if all of the segments on your network are available for traffic. If multiple hosts do not respond to a ping request, a common network device is most likely down. Determining the faulty network device takes some detective work on your part.

While sending a single ping packet to a remote host can determine the availability of a network path, performing a single ping by itself is not a good indicator of network performance. You often need to gather more information to determine the performance of any connections between the client and the server. A better way to determine basic network performance is to send a string of multiple ping request packets.

Using Availability Statistics

When multiple ping packets are sent to a remote host, the ping program tracks how many responses are received. The result is displayed as the percentage of the packets that were not received. A network performance tool can use the ping statistics to obtain basic information regarding the status of the network between the two endpoints.

By default the Unix ping program continually sends ping requests to the designated remote host until the operator stops the operation by pressing a Ctrl-C key combination. Alternately, you can use the -c option in the ping command to specify a specific number of ping requests to send. Each ping request is tracked separately using the ICMP sequence field.

A sample ping session that uses multiple ping packets looks like this:

```
$ ping 192.168.1.100
PING 192.168.1.100 (192.168.1.100): 56 data bytes
64 bytes from 192.168.1.100: icmp_seq=0 ttl=255 time=0.712 ms
64 bytes from 192.168.1.100: icmp_seq=1 ttl=255 time=0.620 ms
64 bytes from 192.168.1.100: icmp_seq=2 ttl=255 time=0.698 ms
64 bytes from 192.168.1.100: icmp_seq=3 ttl=255 time=0.662 ms
64 bytes from 192.168.1.100: icmp_seq=4 ttl=255 time=0.649 ms
^C
--- 192.168.1.100 ping statistics ---
5 packets transmitted, 5 packets received, 0% packet loss
round-trip min/avg/max/stddev = 0.620/0.668/0.712/0.033 ms
$
```

In this example, a response was received for all of the packets that were sent, indicating no problems on the network. If any of the ping packets do not solicit a response, it can be assumed that either the echo request packet did not make it to the remote server, or the remote server's echo response packet did not make it back to the pinging client. In either case, something on the network caused a packet to be lost.

Once you establish that there are lost packets in the ping sequence, you must determine what caused the packet losses. The two biggest causes of lost packets are:

- Collisions on a network segment
- Packets dropped by a network device

Within an Ethernet segment, only one station is allowed to transmit at a time. When more than one station attempts to transmit at the same time, a collision occurs. Having collisions is normal for an Ethernet network, and not something that should cause panic for the network administrator.

However, as an Ethernet segment gets overloaded, excessive collisions will begin to take over the network. As more traffic is generated on the network, more collisions occur. For each collision, the affected senders must retransmit the packets that caused the collision. As more packets are retransmitted, more network traffic is generated, and more collisions can occur. This event is called a *collision storm*, and can severely affect the performance of a network segment.

Dropped packets can also result in packet losses. All network devices contain packet buffers. As packets are received from the network, they are placed in a packet buffer, waiting for their turn to be transmitted. This is demonstrated in Figure 1.1.

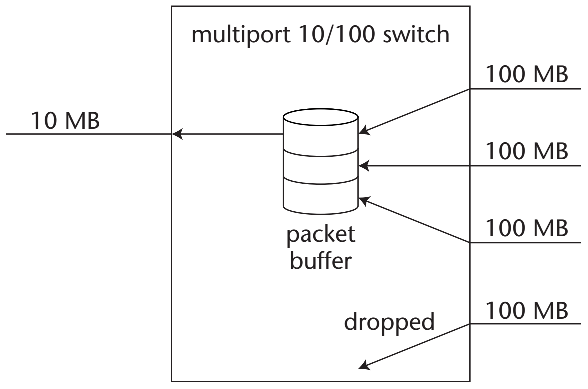

Figure 1.1 Dropping packets in a network device.

Each port on a router or switch device contains an individual buffer area that accepts packets destined to go out the interface. If excessive network traffic occurs, preventing the timely emptying of the buffer, or if more packets arrive than the port can transmit, the buffer will fill up.

If a network device's packet buffer gets filled up, it has no choice but to drop incoming packets. This scenario happens frequently on network devices that connect to networks running at different speeds, such as a 10/100 switch or router. If lots of traffic arrives on a high-speed 100-MB connection destined for a lower-speed 10-MB connection, packets will be backed up in the buffers, and often overflow, causing dropped packets and retransmissions from the sending devices.

To minimize this effect, most network devices are configured to allocate ample memory space for handling packet buffers. However, it is impossible to predict all network conditions, and dropped packets still may occur.

Using Large Ping Packets

Another problem with measuring availability is the size of the packets used in the ping request. Many network devices handle packets with multiple packet buffers, based on average packet sizes. Different buffer pools handle different-sized packets. Too many of one particular size of packet can cause dropped packets for that size category, while packets of other sizes are passed without a problem.

For example, switches often have three classes of packet buffers—one for small packets, one for medium-sized packets, and one for large packets. To accurately test these network devices, you must be able to send different-sized packets to test the different packet buffers.

To accommodate this, most network performance tools allow you to alter the size of the packets used in the testing. When testing networks that utilize routers or switches, you must ensure that a wide variety of packet sizes are used to traverse the network.

TIP **There have been many instances of security problems with large ping packets. As a result, most Unix systems only allow the root account to send large ping packets. You should be careful when sending larger packets to remote servers, so as to not adversely affect the remote server.**

By default, the packet size used in the ping utility is 64 bytes (56 bytes of data and the 8-byte ICMP header). You can use the -s option to change the packet size, up to the maximum that is allowed on the network segment (1,500 for Ethernet networks).

After altering the packet size of the ping packets, you can see how this affects the ping statistics by observing the output from the ping command:

```
# ping -s 1000 192.168.1.100
PING 192.168.1.100 (192.168.1.100):1000 data bytes
1008 bytes from 192.168.1.100: icmp_seq=0 ttl=127 time=2.994 ms
1008 bytes from 192.168.1.100: icmp_seq=1 ttl=127 time=2.952 ms
1008 bytes from 192.168.1.100: icmp_seq=2 ttl=127 time=2.975 ms
1008 bytes from 192.168.1.100: icmp_seq=3 ttl=127 time=2.940 ms
1008 bytes from 192.168.1.100: icmp_seq=4 ttl=127 time=3.133 ms
1008 bytes from 192.168.1.100: icmp_seq=5 ttl=127 time=2.960 ms
1008 bytes from 192.168.1.100: icmp_seq=6 ttl=127 time=2.988 ms
^C
--- 192.168.1.100 ping statistics ---
7 packets transmitted, 7 packets received, 0% packet loss
round-trip min/avg/max/stddev = 2.940/2.992/3.133/0.060 ms
#
```

In this example, all of the large ping packets were still successful, indicating that all of the segments between the host and the client were processing the larger packets without any problems. If you experience packet loss with larger packets, but not with smaller packets, this often indicates a problem with a router or switch buffer somewhere in the network. Most router and switch devices allow the administrator to change the packet buffer allocations to allot more buffers for a particular packet-size range.

Response Time

As seen in the ping example, while network availability is one element of network performance, it cannot accurately reflect the overall performance of the network. The network customers' perception of the network is not limited to whether or not they can get to an individual server. It also includes how long it takes to process data with the server.

To obtain a more accurate picture of the network performance, you must observe how long it takes packets to traverse the network. The time that it takes a packet to travel between two points on the network is called the *response time*.

The response time affects how quickly network applications appear to be working. Slow response times are often magnified by network applications that need to send and receive lots of information across the network, or applications that produce immediate results from a customer entry. Applications such as TELNET, which require the customer to wait for a keystroke to be echoed from the remote host, are extremely vulnerable to slow network response times.

While network response time is often obvious to customers, trying to measure the response time between two separate hosts can be a difficult thing to do. Determining the time it takes for a packet to leave one network device and arrive at a remote network device is not easy. There must be some mechanism to time the leaving and arriving events, independent of the two systems on the network.

When using network performance tools that utilize round-trip response times, it is always wise to incorporate the remote system's CPU utilization in the data taken, to ensure that you are comparing response times run at similar system loads, eliminating the system-loading factor.

Response-Time Factors

In large networks, there are many factors that can affect response times between a client and a server. As the network administrator, you can control some of these factors, but others are completely out of your control. These factors can include:

- Overloaded network segments
- Network errors
- Faulty network wiring
- Broadcast storms
- Faulty network devices
- Overloaded network hosts

Any one or combination of these factors can contribute to slow network response time. Measuring the individual factors can be difficult, but the network performance tools presented in this book can measure the overall effect each factor has on network response times by sending known network traffic samples and determining how the data traverses the network

Determining Response Time from Ping Packets

As seen in the sample outputs for the ping program, the round-trip response time values for each ping packet sent are shown in the ping packet statistics:

```
64 bytes from 192.168.1.100: icmp_seq=0 ttl=255 time=0.712 ms
```

The response time is shown in milliseconds. For internal LAN connections, the response times should be well within 1 or 2 milliseconds. For WAN connections, the response times can often be over 200 or 300 milliseconds, depending on WAN connectivity speeds.

WARNING Remember that the ping response time values are round-trip response times. The current load on the remote system affects these values.

When multiple ping packets are sent, an average of their response times is calculated and displayed:

```
round-trip min/avg/max/stddev = 2.940/2.992/3.133/0.060 ms
```

The response time for a connection can depend on many different factors within the network connection. As the packets traverse the network, each network device can play a role in the total response time. The network performance tool must be able to take into account the response-time factors for each network connection.

The best use for ping response times is to establish a baseline value, or the values seen when the network is performing at normal speeds. When customers complain about slow network performance, the ping response time values taken can then be compared against response times taken during normal network performance. Any drastic deviations in these times can represent a problem with a network device.

Using traceroute for Redundant Paths

In a network that has redundant paths, it is often desirable to determine which path packets are taking at any given time. If you determine that packets are not being routed in the most efficient manner, you can often make simple configuration changes to routers to increase response times.

The Unix traceroute program allows the network administrator to determine exactly which route packets are taking to get between two points on the network. The traceroute utility uses the IP Time to Live (TTL) value to purposely force a packet to expire along the path to the destination.

The TTL value specifies how many hops an individual packet can make before expiring. When a router sees a packet with an expired TTL value, it should report the problem back to the sending network device. By starting the TTL value at 1 and incrementing it at each ping attempt, the traceroute utility forces remote routers along the network path to expire the ping packet and return an ICMP destination unreachable packet to the client. Since the router itself must return the packet, each router traversed along the network path is identified.

The format for the traceroute command is:

traceroute [-dFInrvx] [-f firsttl] [-g gateway] [-i iface] [-m maxttl] [-p port] [q nqueries] [-s srcaddr] [-t tos] [-w waittime] host [packetlength]

As can be seen from the command-line format, the ping program, like the traceroute program, has many options that can be used to fine-tune the testing.

The default values for all of the options can be used to send a simple traceroute probe to a remote host. The output from a sample traceroute across the Internet to the www.cisco.com host looks like this:

```
$ traceroute www.cisco.com
traceroute to www.cisco.com (198.133.219.25), 30 hops max, 40 byte
packets
 1 209.244.188.162 (209.244.188.162)  175 ms  170 ms  171 ms
 2 gige7-0-2.hsipacces1.Cincinnati1.Level3.net (63.212.221.2) 154 ms
150 ms  150 ms
 3 ge-6-0-0.mp1.Cincinnati1.Level3.net (64.159.0.173)  152 ms  150 ms
149 ms
 4 so-3-0-0.mp2.Chicago1.Level3.net (64.159.1.34)  150 ms  149 ms  150
ms
 5 pos9-0.core1.Chicago1.level3.net (209.247.10.170) 150 ms 150 ms 151
ms
 6 144.232.26.185 (144.232.8.185) 151 ms 152 ms 151 ms
 7 sl-bb20-chi-13-0.sprintlink.net (144.242.26.50) 151 ms  150 ms  150
ms
 8 sl-bb20-sj-6-0.sprintlink.net (144.232.8.117) 200 ms 201 ms 203 ms
 9 sl-gw11-sj-9-0.sprintlink.net (133.232.3.138) 210 ms 211 ms 210 ms
10 sl-ciscopsn2-11-0-0.sprintlink.net (144.228.44.14) 211 ms 210 ms 210
ms
11 sjce-dirty-gw1.cisco.com (128.107.239.89)  210 ms  210 ms  210 ms
12 sjck-sdf-ciod-gw2.cisco.com (128.107.239.12) 209 ms 209 ms 210 ms
13 www.cisco.com (198.133.219.25) 211 ms 210 ms 211 ms
$
```

The output from the traceroute program shows every router that responds to the expired test packet along the path to the destination host. Along with that information, the round-trip times that it took for the packet to reach each router are displayed (three separate test packets are sent with the same TTL value for each test). Remember that these values are round-trip response times, and can change with different loading on the individual routers.

Networks that use load balancing will show inconsistent route paths between two points on the network, depending on the network load at the time of the test. As with other response-time techniques, the best thing to do in these scenarios is to take baseline tests under various network loads to see how and when each network path is utilized.

Network Utilization

A major factor in network performance is the utilization of each network segment along the path between two endpoints. The *network utilization* represents the percent of time that the network is in use over a given period. By definition, individual Ethernet segments can only carry one packet at a time. For any given moment, the Ethernet segment is either at 100-percent utilization (carrying a packet), or at 0-percent utilization (idle). The network utilization percentage shows the percentage of time the network is in use over a set period.

Calculating the network utilization requires you to find out how many bytes of network traffic are being handled by the network over a set period. This value depends on the type of network interface that is being monitored.

Half-duplex devices can only carry data in one direction at a time, and therefore calculating their network utilization involves totaling the input and output byte counts for a set period, and dividing by the total capacity of the device interface for that period. To determine the total number of bits received on the interfaces, each of the packet byte rates is multiplied by 8. This value is divided by the total interface capacity multiplied by the time interval of the sample (in seconds):

```
%utilization = ((datasent + datarecv) * 8) / (intspeed * sampletime) *
100
```

For example, a 10-MB half-duplex network interface that over a 5-second period sends 700,000 bytes of data and receives 175,000 bytes would have a network utilization of:

```
%utilization = (((700,000 + 175,000) * 8) / (10,000,000 * 5) * 100 = 14%
```

The 14-percent utilization represents the network utilization only for that 5-second period. It is not uncommon to see high network utilization for a short period of time, given that Ethernet traffic is often bursty in nature. You have a problem when you take the same calculation for a longer period of time, such as a 5- or 30-minute interval, and still get high network utilization.

Although calculating network utilization on an individual network segment can be easy, determining the network utilization between two separate endpoints on the network can be complex. You must calculate the network utilization for each segment traversed along the network path, and determine how each segment's utilization affects the overall response time of the packet.

Due to the complexity of this, most network performance tools utilize different elements—the network throughput and the network bandwidth capacity—to determine network performance between two remote network endpoints.

Network Throughput

Network throughput is similar in concept to network utilization. The throughput of a network represents the amount of network bandwidth available for a network application at any given moment, across the network links. As network applications use network bandwidth, the amount of bandwidth left over for other applications is decreased. The amount of bandwidth left over is considered the network throughput.

Determining network throughput allows the network administrator to find network bottlenecks that slow down performance over a given network link between clients and servers. Often a novice network administrator places a group of clients on a high-speed network device, and the application server on another high-speed network device, to increase application performance. However, what the administrator forgets is that the two high-speed devices may be connected via a slow-speed link. Figure 1.2 demonstrates an example of this.

While the networks that contain the client and server devices are high-speed and have good network performance, it is the interconnecting network device that is causing performance problems. First off, the intermediate network link is limiting the overall speed of the end-to-end link to only 10 MB, no matter how fast the clients and server are connected to the network. Second, since the intermediate network device is a shared hub, it may contain other clients and application servers, which puts additional traffic load on the slow-speed link.

Usually, finding the network bottleneck is not this simple. On complex networks, there can be several network devices within the path of clients and servers. The hardest part of determining the network throughput is calculating the effect that each intermediate link has on the overall end-to-end network connection.

Calculating network throughput is a mathematical process that is best left to the mathematical geniuses. It involves sending periodic streams of packets, and determining the rate at which the server receives the streams. Each stream sample produces data elements used to determine the amount of bandwidth remaining on the network link. The streams are increased until the maximum bandwidth is observed, then quickly backed off so as not to affect the network performance.

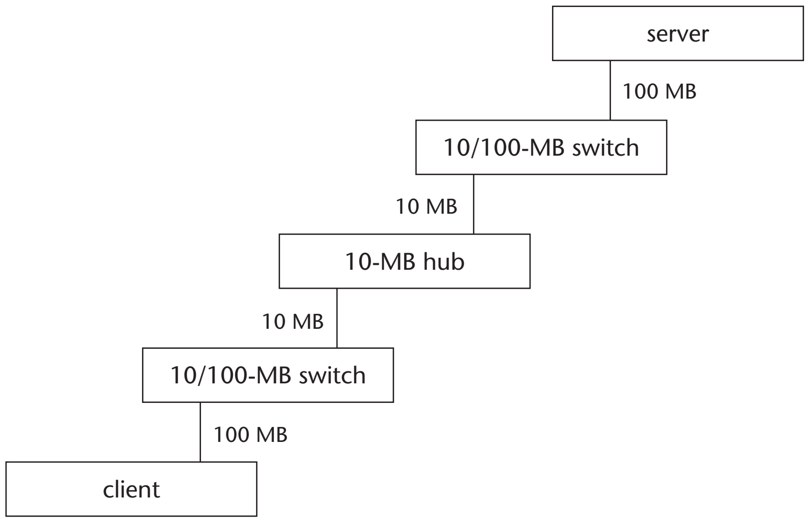

Figure 1.2 Finding the throughput bottleneck.

Of course, this calculation is extremely dependent on exiting network applications, and how they load the network at any given time. It is best to calculate network throughput at different times of the day, and on different days of the week. This enables you to gather all of the information on different applications as they are run on the network.

Many new network applications fail due to lack of available network throughput. If an application is tested in a development environment that does not include the other applications that will be running on the network, it is easy to forget about existing network throughput on the production network.

Bandwidth Capacity

Bandwidth capacity is another factor in the determination of network throughput. The total amount of bandwidth available between two network endpoints can greatly affect the performance of a network. Devices directly connected on a 100-MB network switch should have significantly better performance than devices that are remotely connected via a slower T1 circuit.

The ability to quantify this performance difference requires complex network theory to be built into the network performance tool. The network performance tool must be able to determine the possible end-to-end network bandwidth available on networks with varying link speeds. Each link that a packet must traverse must be included in the overall network performance of an application.

In an ideal network scenario, a constant data rate should be maintained between a client and a server as packets are sent back and forth. The constant data rate represents the network speed at which the two endpoints are linked together. By observing the time it takes for a packet to traverse the network, you can determine the maximum speed of the network link.

As we all know, there is no such thing as an ideal network scenario. In production networks, traffic is constantly traveling between network devices, affecting the perceived speed of a network link. In order to determine the maximum bandwidth capacity of a network link, the network performance tool must do some math tricks.

Two techniques, called *packet pairs* and *packet trains*, are used to determine the maximum network bandwidth capacity of an existing production network without affecting the normal traffic. First, a pair of packets is sent to a remote device at a known separation interval (the packet pair). As the packet pair traverses the network, the interval between the two will vary, depending on the existing traffic (the packet train). Figure 1.3 demonstrates this principle.

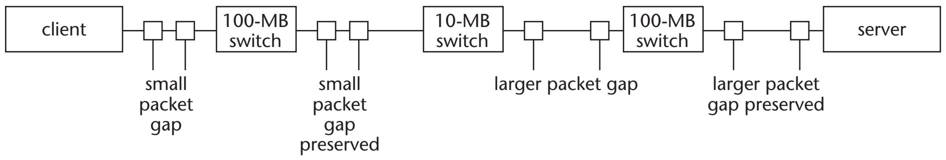

Figure 1.3 Packet separation in different speed segments.

The packets are sent from a device connected to a 100-MB network device. Along the path to the destination device, the network uses a 10-MB network link. The packets received by the switch device on the 100-MB network must be queued and wait their turn before being sent out on the 10-MB link. This affects the separation between the two packets. Again, on the other end of the network, the 10-MB link is connected to another switch, which in turn queues the packets for transmission on the 100-MB network.

When the packets arrive at the destination, the interval separating them is determined. The difference between the original interval and the calculated interval represents the loading on the network. Once the load value is calculated, a large burst of packets can be sent from the client to the server. The rate at which the packets arrive from the client represents the rate, or speed, at which the network was able to transport the data. Given the load factor and the data rate, the network performance tool can calculate the theoretical maximum speed at which the network link should be able to process data.

Methods of Collecting Performance Data

After determining which network performance elements to monitor, the network performance tool must be able to access data for the elements. Three different methods are used to obtain data from the network:

- Querying network devices for stored information
- Watching existing traffic on the network for signs of network performance issues
- Generating test traffic to send on the network to test network performance

Part of your job when evaluating a network performance tool is to determine how the tool extracts data about the network's performance, and if that method is appropriate for your particular network environment. This section describes the three different data-collecting methods, and shows how they can differ when collecting data on the network.

Querying Network Devices

One of the simplest methods used to collect network data is to go to the source of the network, the network devices. When purchasing network devices such as hubs and switches, you will find that two types of each network device are available:

- Managed network devices
- Unmanaged network devices

Managed network devices include management software that collects statistics about the network traffic traversing the device, whether it is a simple network hub or a complex network router. Unmanaged network devices do not collect any statistical information about the network traffic, and cannot be queried for information.

While managed network devices are usually more expensive than unmanaged devices, they are well worth the extra price in large networks. Many network management software packages, as well as many freely available open source network performance tools, use the network device management interface to extract network information from the managed network devices. This information is invaluable for troubleshooting network problems.

How Tools Query Devices

All managed network devices implement the Simple Network Management Protocol (SNMP). SNMP provides both a mechanism for storing network data in a hierarchical database on the network device, and a protocol for querying the database from a remote device.

All SNMP devices include a common Management Information Base (MIB) that stores basic network information about the traffic on the device. This information includes the bytes transmitted and received per interface, the number and types of errors on the interfaces, and the network utilization on each interface. This information is invaluable for determining network performance on different segments of the network.

Besides the common MIB database objects, most SNMP devices also include proprietary MIB objects that track information specific to the network device. SNMP provides an area within the hierarchical database for companies to create their own entries to track information specific to their network devices. This area of the database is called the *enterprise* MIB. Access to the enterprise MIB is usually controlled by SNMP *community names*. A community name is a password that is used to grant access to specific parts of the MIB database.

Values to Query

When using SNMP to query network devices, you must know which values are pertinent for network performance information. There are plenty of data objects that can be found in the SNMP MIB tables on network devices, and dumping all of the tables would take an extremely large amount of time and network bandwidth.

The common MIB database provides much of the basic network performance data used to track performance of a network device. The second version of the common MIB database (called *MIB-2*) has been updated to include many error statistics for network devices.

The MIB-2 database objects provide many useful fields that can be used to determine the amount of network traffic and errors on a network device. Querying these values can give you lots of information regarding network traffic on the device. Table 1.1 shows some of the more useful MIB-2 objects that should be considered.

Table 1.1 SNMP MIB Network Performance Objects

OBJECT	DESCRIPTION
IfType	The physical type of interface
IfSpeed	The data rate capacity of the interface
IfMTU	The size of the largest data packet the interface can handle
IfAdminStatus	The status of the interface (active/inactive)
IfInOctets	The total number of octets received by the interface
IfOutOctets	The total number of octets sent by the interface
IfInUcasePkts	The total number of unicast packets received by the interface
IfOutUcastPkts	The total number of unicast packets sent by the interface
IfInNUcastPkts	The total number of non-unicast packets (broadcast/multicast) received by the interface
IfOutNUcastPkts	The total number of non-unicast packets (broadcast/multicast) sent by the interface
IfInErrors	The total number of packets received that contained errors
IfOutErrors	The total number of packets that could not be sent because they contained errors
IfInDiscards	The total number of packets received that could not be processed, even though they did not contain errors

Most of the MIB-2 values are continuous counters. For example, the ifInOctets object counts the number of bytes (octets) received on the interface since it was first powered on (or the MIB-2 database was reset). This value can reach a maximum value, and then roll over to zero and start over. To determine a data rate, most network performance tools query these values over a set period of time, and subtract the difference. Care must be taken when doing this, to ensure that the value has not rolled over to zero between the measuring times, affecting the data results.

Network devices that contain multiple ports (such as switches and hubs) maintain a separate MIB-2 table for each interface on the device, as well as a system-wide MIB-2 table. The separate port tables are accessed using special indexing within the MIB value. Chapter 3, "Network Device Utilization," describes how to access this information using SNMP network tools.

Watching Existing Traffic

Another common technique used for collecting network performance data is to watch existing traffic on the network. A lot of information can be gathered from the network just by watching the packets that are traveling between network devices.

In order to capture all of the traffic traveling on the network, a device's network interface must be placed in *promiscuous mode*. By default, a network interface only accepts packets that are destined for the device, or that are sent out on a multicast or broadcast address. Promiscuous mode allows the network interface to read all packets on the network, regardless of their destination. This feature allows the network device to inspect each packet on the network, no matter where it came from, and where it is sent.

TIP When attempting to capture traffic on a network, you must be careful of devices such as switches, bridges, and routers, which segment traffic. If your network device is on one of these types of devices, it will not see all of the traffic on the network.

After the network packets have been captured, they must be decoded and analyzed to see what trends and/or problems exist on the network. A few items that can be seen by analyzing network traffic are:

- Packet retransmissions
- Frozen TCP window sizes
- Broadcast storms
- Network advertisements
- Chatty applications
- Quality of service applications

03 433012 Ch01.qxd 6/16/03 9:09 AM Page 19

Each of these items can be a potential network performance problem, and should be watched in the network monitor.

Generating Test Traffic

Many network performance tools generate their own network traffic to determine the current performance of the network. This technique requires math skills, as well as a knowledge of network theory.

All network performance tools that analyze network performance by generating test traffic require two devices on the network. The network performance along the path between the two devices is determined by using the packet pair and packet train methods, described previously in the *Bandwidth Capacity* section. This is demonstrated in Figure 1.4.

In Figure 1.4, the network performance tool determines the performance only between devices A and B. No other paths in the network are tested. In order to test other paths on the network, the testing devices must be relocated to other points in the network. Of course the alternative is to have multiple test device pairs and locate them at different points in the network. The trick is to place the smallest number of testing points that can cover the largest area on the network.

As mentioned, calculating network performance requires you to send pairs and trains of packets across the network. The packet pairs do not take up much network bandwidth, but the packet trains can place a fair amount of data on the network. Care should be taken when using network performance tools that use packet trains, so as not to adversely affect production traffic on the network.

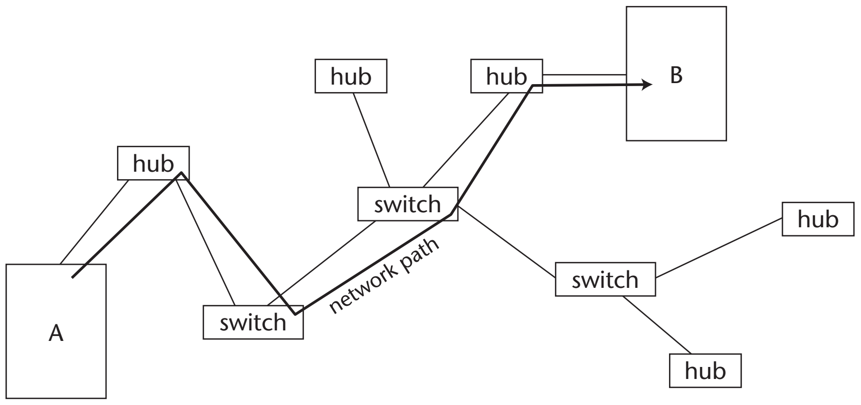

Figure 1.4 Generating test traffic on a network path.

Summary

This chapter describes what network performance is, and how a network performance tool can measure it. Network performance incorporates five separate elements: availability, response time, network utilization, network throughput, and bandwidth capacity.

The availability of the network is crucial for network applications. Testing the availability of the network is often done by using a simple ping test to determine which hosts on the network are reachable. After determining availability, you can measure the response time for various hosts on the network. Different response times can be found based on different network link types and different paths in the network.

Network utilization is measured to determine how much of the network is being used for applications, and the percentage of error transmissions. A network with high utilization will have an increased amount of errors in the network traffic. Similar to the network utilization are the network throughput and capacity. The capacity is the total amount of data that can theoretically pass between two points on the network. This can be affected by different link speeds across the network, and different types of cables used to connect the network devices. The network throughput represents the amount of network bandwidth currently available for applications.

The are three different methods of collecting network performance data from the network. The Simple Network Management Protocol (SNMP) is used to query managed network devices for network information. SNMP devices store network information in the Management Information Base (MIB) database. Information such as bytes received and sent, as well as errors received, is contained in the MIB database. A remote network management workstation can query the MIB database using SNMP to retrieve network information about the device.

Watching network traffic can also determine network performance. Telltale signs such as broadcast storms and packet retransmissions can be seen by capturing data as it flows through the network. The last method of collecting network performance data is to generate test traffic on the network. Some network performance tools generate test packets and send them across the network to determine the network capacity and performance. By using packet pairs and packet trains, network performance tools can calculate the network information based on packet separation (the spacing between packets) and throughput rates.

The next chapter describes one of the basic elements of network performance monitoring—watching network packets. By observing the actual network traffic, you can often identify the device (or devices) contributing the most to network load. There are several open source applications that are available to help you watch network traffic. Each one will be discussed and demonstrated.