CHAPTER 9

Performance Management with NNM

Introduction

There is an often-heard philosophy out there about performance management in the real world:

> "To do network performance planning properly you have to measure utilization at all points of the network and model the topology in detail. You don't have the resources to measure anything, you don't really know what the network looks like, and it would take forever to simulate it all. Therefore, it's impractical to do capacity planning, and we should continue to provision bandwidth to deal with performance problems."

Regardless of the planning methodology used, it's important to measure network performance in order to manage it. Network troubleshooters need real-time utilization and error data. The help desk needs to view performance data in relation to a user complaint. The network engineering staff needs performance data for capacity planning. The IT group needs data to present at the monthly service level agreement (SLA) meetings.

Providing data for the monthly meetings means configuring NNM to measure agreed-upon performance metrics. Given the difficulty of gathering end-to-end transaction response time data, more robust metrics such as line utilization and active user counts are more practical.

Determining how long to keep NNM performance data online involves a trade-off between performance, convenience, and cost. Troubleshooters need about an hour's worth of real-time data, while capacity planners require up to a year's worth of historical data. Storing more data online can reduce performance and increase system administration overhead unless a more powerful and costly NNM platform is configured.

What is an appropriate SNMP data sample rate? Sampling too quickly may cause some network devices to overload and will certainly

increase network management traffic. But long sample intervals miss all the useful variation in the performance metrics. A five-minute sample interval is suggested.

The Heisenberg uncertainty principle of quantum physics can be stretched to explain why excessive SNMP prodding of the network can limit how accurately it can be measured.

How much traffic does an NNM system actually create? You can attempt to quantify this with a simple polling example. Note that configuration checking, status polling, HTTP and Java, X-Windows, ITO, and Measureware contribute traffic as well.

Deciding what SNMP data to collect from the hundreds of possible MIB values is best done using the KISS (keep it simple, stupid) principle. A few system and network utilization and error statistics often suffice. MIB expressions are appropriate because percentages are more useful than raw numeric information.

NNM allows you to configure performance thresholds to generate alarms. Thresholds can be established using baselining, or analytical and rule-of-thumb methods. Set threshold events to low priority unless you have a process for dealing with them.

MIB expressions allow you to configure NNM to sample several SNMP variables, evaluate a formula containing them, and return the result. Typically, you want to calculate a percentage. For example, an error counter is meaningless unless it's normalized by the packet counter and converted to a percentage. NNM provides many predefined MIB expressions which you use out of the box or as a template.

Viewing historical performance data online can be done graphically with *xnmgraph,* or by using the SNMP historical data configuration GUI. Data can be viewed and saved textually using *xnmgrap*h or *snmpColDump*.

Presenting data offline means taking a screenshot or exporting textual data to a presentation or spreadsheet tool such as Star Office, Wingz, or one of the Windows or Macintosh equivalents.

SNMPv2C supports 64-bit counters. These are essential for managing links operating at or faster than 100 megabits per second (Mbps). NNM automatically detects devices with SNMPv2C support, and the ifXentry section in the interface portion of MIB2 defines several 64-bit counter variables.

Collecting RMON data is best done with HP NetMetrix. Remote shared-medium Ethernet segments can also be monitored (in a limited way) with NNM directly using the Etherstats group and a few good MIB expressions.

Crossing over to HP NetMetrix means having complete end-to-end traffic flow by application at your fingertips. Network probes or switches and hubs with built-in RMON2 properly situated on the network can collect enterprise-wide performance data that NetMetrix can massage, present, and report.

After you've collected NetMetrix network-wide performance data and a meaningful NNM baseline topology, capacity planning follows. HP Service Simulator can import the performance data and topology. Armed with what-if questions, you can use the simulator to verify that the network can meet performance objectives under various conditions you specify.

Who Needs Performance Data?

Network performance data is an essential component of network management because:

> "you can't manage what you don't measure."

Performance information is needed at every stage in a network's evolution. Those stages include requirements, architecture, design,

implementation, routine maintenance, and upgrades. Therefore, it's common to depend on SNMP agents in the network to provide performance data and to use an SNMP-based network management system like NNM to collect it. Consumers of network performance data include troubleshooters, the help desk, network engineering, SLA administrators, financial planners, and system administrators.

Troubleshooters need both real time and historic data to diagnose the root cause of user complaints. They will usually locate the path traffic takes between the user's workstation and the server (see Figure 9-1) and check on media utilization and error percentage rates. If the problem symptoms are active, then a real-time view is necessary. If the problem symptoms have passed, then the historical data at the time the problem occurred is what's needed.

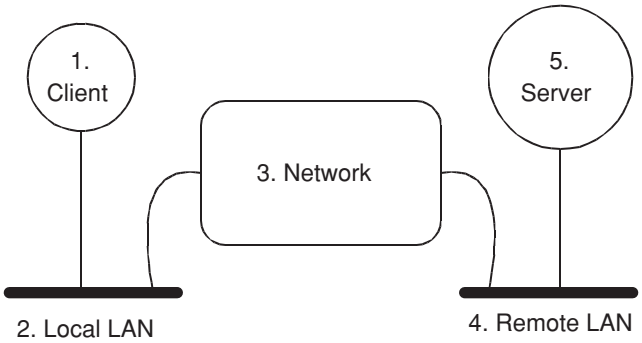

Figure 9-1 Potential performance problem locations. Troubleshooting network problems between a client system and the corresponding server requires that five locations be instrumented. A performance problem might be caused by a lack of resources, excessive utilization, CRC errors, or packet loss. SNMP agents located at locations 1-5 can help isolate the problem and correct the direction of fingerpointing.

The help desk staff needs to view capacity, error, and packet loss data as they collect basic information about a user complaint. If the user's workstation has a working SNMP agent, it's possible to measure performance data for its LAN adapter; for advanced agents, resource usage such as CPU, RAM, and disk is available. Server systems almost always support SNMP. Client, network, and server systems are all involved in most user performance complaints, so it's important to measure SNMP data at all of them to isolate the true cause. Performance problems are not always caused by the network. For example, it is entirely possible that poor client-server response time is caused by the client having too many applications open, resulting in excessive virtual memory (VM) activity.

The network engineering staff needs operational metrics to validate network changes and bandwidth upgrades. Knowing link utilization is important, but so is knowing the source-destination metrics for various applications. For example, some web traffic will be directed to local servers and the rest may be destined for the Internet via the firewall. The same is true for e-mail. File and print traffic is usually sent to a local file server and a local LAN-attached printer. An understanding of application-specific source-destination pairs provides valuable insight into how a network redesign should be done. While simple link-based utilization data can be used to correct a bandwidth allocation problem, RMON2-based data is needed to collect application-specific source-destination pairs. Display tools such as HP NetMetrix can display this data graphically and create tabular reports as well. This performance data is often input to a more comprehensive network capacity planning process. This validates that a given network design, which clearly provides the necessary connectivity, also provides the necessary performance.

The IT organization needs metrics to comply with service level agreements (SLA) and their user communities. Each user community is a special interest group with specific network needs, and an SLA is an agreement between them and IT about what level of service is expected. The metrics agreed upon must be realizable and measurable. Application response time is very difficult to measure when

developers have not instrumented the code. Code that is instrumented using the application resource monitor (ARM) APIs is very easy to monitor with the standard HP PerfView application. Without ARM-enabled application code, simpler SNMP measurable metrics such as utilization are typically the basis for the SLA.

Financial planners use historical performance data to demonstrate to management why money is needed to upgrade the network. A properly presented line chart depicting historical traffic growth plus a credible forecasting technique can ethically and accurately depict the urgency for a network upgrade. See Figure 9-2.

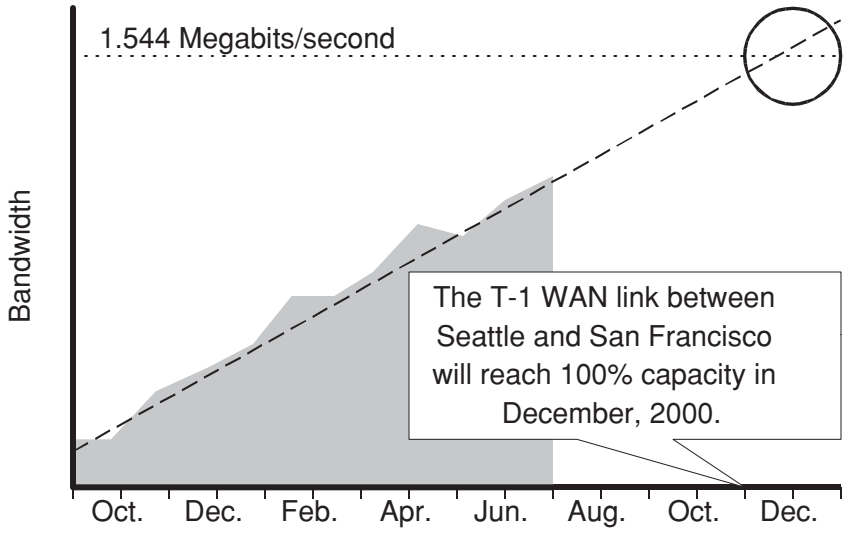

Figure 9-2 Projecting capacity using SNMP data.
NNM can collect historical SNMP utilization data from a router interface. Repurposing the data into a professional presentation gives impact to the message that a network upgrade is critical.

Even system administrators benefit from knowing traffic patterns when they decide where a new server is best placed, much like

network designers who benefit from knowing the source-destination traffic a new server will generate.

Providing Data for an SLA

Let's make a list of what users really care about when it comes to their network applications.

What Users Care About:

response time for interactive transactions
throughput for file transfers and print jobs
high availability
ease of use
convenience

Now let's make another list of things users don't care about as they use their day-to-day network applications:

What Users Don't Care About:

network backbone utilization
percent error rate
percent packet loss
ping round trip time

Now isn't that interesting? The very things that network managers routinely measure are of no concern to the users. SNMP provides dozens of performance metrics, none of which relate directly to the user experience. This is because SNMP was designed to manage networks, not applications.

Network managers may take the position that they provide error-free network bandwidth and that application response time is the business of server administrators. Application response time can be measured at the application server if the code is instrumented with ARM. But the transaction response time seen by the user is the sum of client-side

response time, server-side response time, and network latency at each network location between client and server. No wonder there is so much fingerpointing when a user complains about performance.

Network and system administrators alike know that capacity headroom is critical for good performance. That's why they measure utilization and that's why an SLA should contain an agreement about this metric. Remember that an SLA is a tool used by the end-user community and their IT service provider to come to an agreement. It is not a monthly excuse to complain, point fingers, exercise political agendas, and take down names.

So you see that an SLA must be "as simple as possible and no simpler"[1] and be based on measurable quantities. For example, it may be agreed upon that the IT department will engineer the connection to the Internet such that utilization is below 50% for 90% of the time as long as there are fewer than 100 active users. Utilization data is easily obtained from the router via SNMP. The number of active users isn't directly measurable with SNMP. Indeed, the measurement is a difficult one to make properly, but it can be harvested from the firewall proxy server log or from the router via its IP accounting feature. Every month at the SLA meeting a simple graph is presented, such as in Figure 9-3.

Determining Performance Data Retention Intervals

The *snmpCollect* daemon will happily save SNMP historical data into its database until the hard drive capacity reaches zero. Unpredictable things happen to mission-critical systems when their disks fill up, so it's necessary that you address how long you want online SNMP data to be available.

1. This quote is attributed to Albert Einstein.

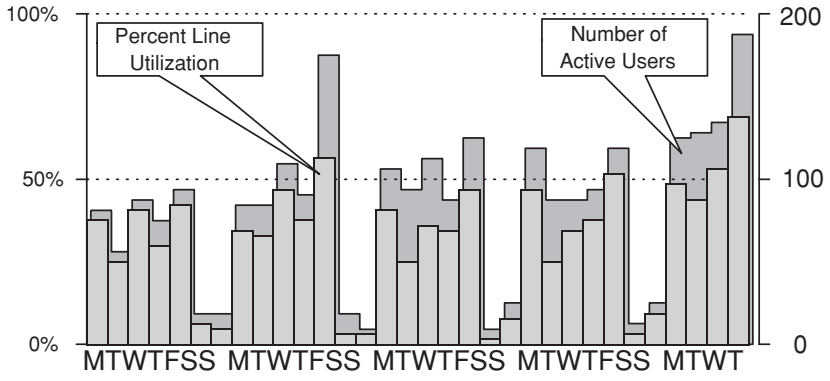

Figure 9-3 A sample chart used in an SLA.
The IT department and the user community have agreed that 90% of the time the line utilization to the Internet will not exceed 50% as long as there are less than 100 active users. Just by graphing this data, you see that Friday is the busiest day of the week and the last day of the month is also the busiest. Special business days often bring out more users when they use a different application mix.

If you want to support users performing ad-hoc SNMP data collections, then you need to provide the necessary disk space for them. The user is entrusted to behave responsibly. The golden rules of ad-hoc data collection are:

- massive data collections are generally inappropriate
- very long term data collections should be modest
- delete the data collection when the study is over
- limit rapid polling studies to the session interval

Many NNM systems are configured to collect basic SNMP data on all devices in the management domain as a general service to the user community. By default, an out of the box (OOTB) NNM system has all defined data collections suspended, so no SNMP data is available without intervention by the NNM system administrator. NNM offers

ovbackup and the data warehouse feature to backup or trim the historical SNMP data.

Trimming the amount of SNMP historical data is necessary because it will eventually fill the disk volume. A bulging database will also slow down the x*nmgraph* data display tool to a crawl as it searches for data to plot. Troubleshooters require only recent SNMP data for their tasks. Long-term performance data used by the network engineering staff can be culled from the NNM backup data just as easily.

To perform SNMP data trimming, HP provides a sample UNIX script in the manpage for the *snmpColDump* application (see Figure 9-4). Most NNM administrators will modify this script to suit their local needs and create a UNIX *cron* job to run it periodically, say, hourly.

```
#To keep only the last 2000 entries in file 1MinLoadAvg

lineno='snmpColDump 1MinLoadAvg | wc -l'
if [ $lineno -gt 2000 ]; then
lineno='expr $lineno - 2001'
else
lineno=1
if
snmpColDump -tTI 1MinLoadAvg | sed -n $lineno',$p' |
awk -F\t '{printf("%d\t%d\t%s\t%lg\n", $4, $5, $6, $3)}' |
snmpColDump -r - 1MinLoadAvg
```

Figure 9-4 A sample SNMP data trimmer script.
This small shell script from the *snmpColDump* manpage trims data in the *1MinLoadAvg* file to 2000 entries. This is a UNIX SNMP variable that measures the average number of processes in the run queue over a one-minute interval. Customize the script by running it against every SNMP data file on the system. Assuming that the NNM system administrator understands the Nyquist Sampling Theorem, the samples in *1MinLoadAvg* are probably taken every 30 seconds. Retaining 2000 data samples corresponds to 1000 minutes (about 16.7 hours) of data.

To speed up the data trimming process on a multiprocessor system, you can launch parallel trimming scripts, with each one assigned to an independent portion of the SNMP database. You'll notice a dramatic speed improvement. An alternative to using *cron* to launch the data trimmer is to configure ITO to monitor the size of the database. ITO can automatically execute the trimmer script when necessary.

Let's review some reasons for retaining SNMP historical data online. Assume the above issues can be mitigated. Perhaps you add more RAID disk stripes, a second SCSI controller, and another CPU to enhance performance. Perhaps you modify the script in Figure 9-4 to resample the data by averaging older five-minute samples into one-hour samples, thus reducing the data volume by a factor of 12. You then benefit from having enough online SNMP historical data that covers the following important periods of any business:

busiest hour of the day
busiest day of the week
busiest day of the month
busiest day of the quarter
busiest day of the year
busiest day of a special event

Troubleshooters can check back to see if the utilization they see now is comparable with that seen at a similar time in the past. For example, historical performance data shows that high network utilization at the end of the month in a sales office is actually normal, as it is at the end of a fiscal quarter.

A final note about long-term SNMP data retention deals with the cost issue of disk drives. At this writing, the cost of an 18-gigabyte internal SCSI disk drive is in the $600 range. Therefore, an 18-gigabyte dual-mirrored triple-striped disk array can be built for $3600 plus change. Obviously, you need to choose a computer platform that can accommodate these disks internally or externally, and this increases the price accordingly. But these figures are not outlandish; in fact, for a mission-critical NNM system they are more than acceptable.

Estimating SNMP Data Sample Rates

When you ask five network managers “what is a sensible sampling rate for SNMP data?” you will get six different answers. There are many conflicting issues responsible for this. Let’s review some of them.

NNM itself allows SNMP sampling intervals as small as one second. SNMP agents running on 8-bit hardware as low-priority processes may be unable to respond to an SNMP request for multiple objects in such a short interval. They will often time-out and become unresponsive when pressed too hard. Recall that NNM is configured by default to try an SNMP request three extra times with a 0.8-second exponential time-out (0.8, 1.6, 3.2, and 6.4 seconds for four time-outs totalling 12 seconds). The retries will just serve to overload slow SNMP agents. Therefore, one-second polling intervals are generally avoided because that’s smaller than the time-out interval. NNM administrators don’t want to spend the extra time to configure specific SNMP timing parameters for specific network devices.

Polling a device on a remote network may incur a one-second round trip latency, especially if there are congested serial links to traverse. The default short SNMP time-outs will just add to the network traffic. For entire subnets it *is* practical to define one set of SNMP timeouts because it involves just one wild card setting in the configuration GUI. Still, one-second polling is too small for this case because the latency will disturb the sampling times by up to several seconds.

High-speed and high-volume SNMP polling can generate considerable network traffic. Given that many NNM systems use a fast Ethernet adapter, it is conceivable that a serial line can be swamped by SNMP traffic to the detriment of mission-critical traffic. A rule of thumb is that network management traffic should not use more than 10% of any link’s capacity. Assuming a 200-byte SNMP packet size, you can multiply by the number of devices and divide by the polling interval to calculate the traffic SNMP polling adds to the

network. Note that *snmpCollect* attempts to reduce the number of SNMP requests by testing each device's SNMP agent for the number of values it can return in one request. This reduces the overhead of sending multiple, single-valued requests, which is a good thing. It also increases the average packet size well above 200 bytes assumed above. Traffic overhead on the network is another reason for using larger polling intervals.

Short polling intervals on many SNMP objects will cause the *snmpCollect* daemon to consume more CPU time. This can negatively impact a small NNM system. Ideally, you want to keep *snmpCollect* below 10% CPU utilization. On the other hand, if too many concurrent data collections are attempted, *snmpCollect* can fall behind. Help *snmpCollect* keep up by configuring the *-n numberconcurrentsnmp* option in *snmpCollect.lrf*. Monitor the *$OV_LOG*/*snmpCol.trace* file for problems, because this option can be set too high. Keep this value well below *maxfiles*, the operating system's maximum number of open files parameter. If *maxfiles* is 64, then *-n 35* is found empirically to work well (but check *$OV_LOG*/*snmpCol.trace* to verify *snmpCollect's* health*)*. Once again, you have a reason to keep polling intervals on the high side.

We've seen that excessive polling can negatively impact network devices, the network itself, and the NNM system. A one-second polling interval isn't a good idea. But if you poll hourly, all the interesting variations in the network's statistics are averaged into one fairly unrepresentative and virtually useless statistic. Review Figure 9-5 to appreciate how sampling rates effect the quality of the resulting data. When in doubt, choose a five-minute sampling interval. Experience shows this captures enough of the changing statistics of network metrics and yet doesn't stress the network, NNM system, or sensitive network devices.

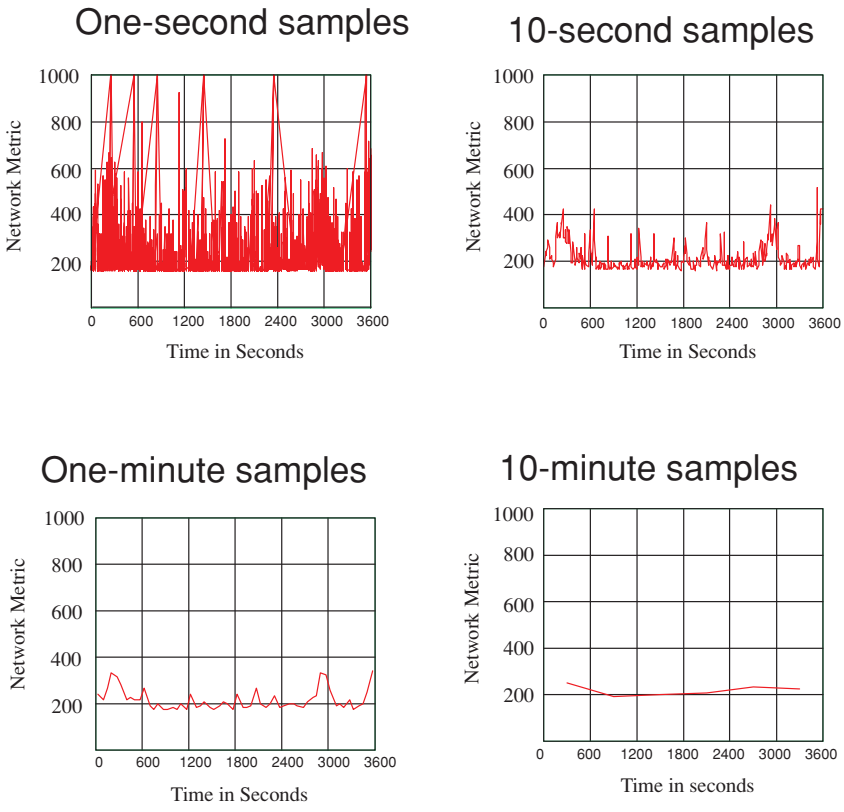

Figure 9-5 How sample rates effect data quality.
High sampling intervals like one second, ten seconds, and one minute capture all of the interesting variations in the network metric. These sample rates are too high. The 10-minute samples have removed all of the interesting information in the data. This is why a common sampling interval is five minutes.

The Heisenberg Uncertainty Principle of SNMP Polling

Practitioners in the field of quantum mechanics appreciate the famous Heisenberg uncertainty principle which states that it is not possible to determine simultaneously and exactly the position and momentum coordinates of a particle. The product of the two uncertainties is always greater than a minimum value approximately the size of Planck's constant (6.6256×10^{-34}). Quantum physicists know this is because the act of measurement disturbs the process being measured.

Network managers don't have a formal-sounding principle to explain it, but you know that using SNMP management software on a network disturbs it. Polling the network at a rate that lets us measure its real behavior disturbs it so badly that you can use a fairly timid polling interval of five minutes. And your rule of thumb is to limit SNMP traffic to 10% of the line speed. What do you disturb when you use NNM to poll devices? Consider the following list:

- adds to traffic, which only makes high utilization higher
- burdens the network equipment and server systems
- the data itself is subject to jitter due to latencies
- many devices give their SNMP agents low priority

Most of these problems can be avoided. Consider using RMON2 agents to sample network statistics. RMON2 probes don't need to poll the network equipment and HP NetMetrix can upload the statistics periodically with minimal impact on the network.

How Much Traffic Does NNM Create?

A network management system will generate many different kinds of traffic in various amounts. A reasonably complete list of traffic types follows:

- DNS forward and reverse lookups plus zone transfers
- status polling (ICMP echo request and reply)
- SNMP configuration polling by *netmon*
- collection station to management station traffic
- SNMP traps received from the network
- HTTP and Java traffic to remote browser users
- X-Window traffic from remote *ovw* users
- performance data collection by *snmpCollect*
- RMON SNMP traffic collected by NetMetrix
- ITO agent traffic to the ITO manager
- MeasureWare statistic to PerfView

Given the bursty nature of this traffic, a dedicated 100BASE-T full duplex switch port should be used to connect the NMS system to the network. The full duplex feature eliminates the possibility of collisions and the 100-megabit/second line speed ensures high throughput and reduces LAN latency to a minimum. It also allows the remote disk backup process to complete in a timely manner.

Despite the conservative musing above, it's the author's experience that a standard Ethernet connection is generally sufficient. Still, it's

preferable to know how much traffic the NNM system is generating. This can be done using one of these methods:

- let the NNM system poll its own LAN interface
- let the NNM system poll the switch port it's connected to
- use the NNM system's MeasureWare agent and PerfView
- use NetMetrix and an RMON probe to monitor NNM traffic
- attach a LAN analyzer to the NNM system Ethernet port

The NNM system is supposed to be situated in the heart of its management domain to minimize the impact of network management traffic by diversifying it. All traffic in and out of the NNM system does not impact just one WAN circuit. Yet if a large number of managed devices and subnets exist at the far end of a relatively slow serial line, you should estimate the impact of NNM traffic to avoid breaking our 10% rule. RMON2 probes with HP NetMetrix can accurately measure the traffic.

Note that the network management staff and the NNM administrators need to work together. It's already problematic that router SNMP agents are assigned the lowest task priorities in Cisco routers. It's less helpful if SNMP traffic is also assigned to a low priority queue by a router's manager, or if the NNM system's IP address is excluded from the router's access control list (ACL). Such antics are symptomatic of dysfunctional teams.

Prior to deploying NNM, the question of NNM's traffic impact can't be answered by measurement, so you need to attempt a quick analysis. Assume that your management domain looks like Figure 9-6. Given the bandwidth of the WAN links and the number of managed devices in each campus area, can you calculate the SNMP traffic impact due to performance polling? (You don't have RMON2 probes or NetMetrix in this discussion). Assume that traffic between sites A and D passes through sites B and C. There is no NNM traffic on the line between sites F and C since both have a direct line to site D. A simple spreadsheet is used to tabulate the data and calculate the results. You

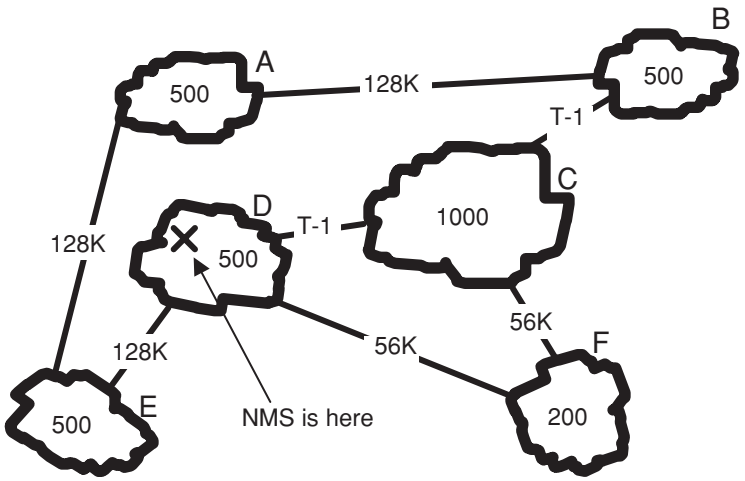

Figure 9-6 Calculating the impact of performance polling. This network consists of six sites that look like small clouds connected with WAN links. The number of nodes at each site is given along with the line speed. You assume in the arithmetic that the line speed equals the actual data rate available. The arithmetic in Table 9-1 calculates the line utilization due to the NNM performance polling.

can assume five-minute polling and 200-byte packets. Each device polled is assumed to have one interface in the sample arithmetic. You should adjust the arithmetic to reflect your device average interface count. You calculate that the line between sites E and D suffers a mere 2.1% utilization, the worst of all the lines you calculated. This is well below your 10% maximum rule of thumb. Table 9-1 organizes the arithmetic. Polling traffic in bytes-per-second is calculated by multiplying the packet size in bytes by eight times the number of nodes and dividing by the polling interval of five minutes times 60 seconds per minute:

$$(\text{packet\_size\_in\_bytes} \times 8 \times \text{Number\_of\_nodes})/(300 \text{ seconds})$$

Now calculate the percent line utilization by dividing the polling traffic in bits-per-second by the line speed in bits-per-second and multiplying by 100:

(polling_traffic/line_speed)×100

Table 9-1 Estimating Line Utilization for SNMP Polling

Link	Traffic From	Nodes Polled	Traffic Bits/ sec.	Line Speed Bits/sec.	Line % Utilization
A-B	A	500	2667	128,000	2.1
B-C	A,B	1000	5333	1,544,000	0.3
C-D	A,B,C	2000	10667	1,544,000	0.7
F-D	F	200	1067	56,000	1.9
E-D	E	500	2667	128,000	2.1
C-F	n/a	n/a	n/a	n/a	n/a
A-E	n/a	n/a	n/a	n/a	n/a

SNMP Performance Data in MIB2 and Private MIBs

There are hundreds of useful variables in the industry standard MIB-2. Most are located in the interfaces section, and a few more can be found in the IP section. It is prudent to limit the amount of data you collect to the bare essentials. This avoids overloading SNMP agents, taxing the network, and storing a lot of unnecessary performance data on the hard drive.

Often, a single variable does not tell the whole story by itself. For example, the number of input errors on an interface is meaningless by itself. You have to divide it by the number of received packets and multiply by 100 before it's possible to judge if the error rate is too high.

NNM lets you form mathematical formulas comprised of MIB values. These formulas are called MIB expressions, which are generally much more useful than raw SNMP values. Table 9-2 lists some recommendations plus some threshold settings to get you started.

Table 9-2 Threshold Value Rules of Thumb

NNM MIB Expression	Suggested Threshold Settings	Explanatory Notes
avgBusy5	Alarm 90% for 4 samples. Rearm 50% for 2 samples.	CPU router utilization five-minute average. High router CPU utilization may not be a bad thing.
If%util	Alarm 90% for 4 samples. Rearm 60% for 2 samples.	The interface is treated as a full duplex. High utilization by itself is not a bad thing.
If%inErrors	Alarm 1% for 4 samples. Rearm 0% for 2 samples.	Small but consistent errors are a bad thing.

Table 9-2 Threshold Value Rules of Thumb

NNM MIB Expression	Suggested Threshold Settings	Explanatory Notes
If%outErrors	Alarm 1% at 4 samples. Rearm 0% for 2 samples.	Small but consistent errors are a bad thing.
IP%PacketLoss	Alarm 1% for 4 samples. Rearm 0% for 2 sample.	Packet loss in the router due to buffer overflow and other maladies reduce application performance. For Cisco routers, it counts the number of packets the router could not forward because no ARP reply was received for a hop.

Note that the first entry in this table is not a MIB expression, it is part of the Cisco enterprise MIB. For all table values the recommended sample interval is five minutes. The thresholds given are rules of thumb. The alarm sample duration of four samples ensures the condition is persistent while the two-sample rearm ensures that you detect return to normal situations quickly.

Strategies for Setting Threshold Values

Should you define threshold values for performance polling? What is the staff going to do about a threshold event? What is the local

process? The usual answer is that nothing can be done or should be done. After all, if a user is downloading a large file or if a backup is in progress, is it appropriate to hunt down the offending system and shut down the switch port it's connected to? This kind of draconian network policing is certain to cause vehement user reaction.

A threshold event should be classified as a mere warning and not acted upon directly by the staff. There is also an opportunity for an ECS-trained network manager to create a custom circuit that intelligently correlates threshold events into a more useful one. For example, if a significant portion of the network is experiencing threshold events, the custom event correlator should detect this and generate a major event.

Assuming you want to generate threshold events, how should the threshold values be set? Three approaches can be considered.

The easiest answer is to use the published rules of thumb such as those in Table 9-2 and in textbooks on capacity planning[2].

A more difficult approach is to adjust the threshold levels upward until the event rate is acceptably low. This approach is clearly labor-intense since each and every interface has to be monitored and individually tweaked. Another name for this technique is baselining. You monitor the performance metrics for a few weeks for each device and then change the threshold levels accordingly. Note that these threshold levels can be manually added in the *$OV_CONF/snmpCol.conf* file to avoid a lot of tedious work with the SNMP data collector configuration GUI. The format of this file changed at NNM 6.0, so be cautious when you upgrade your NNM system to migrate this file format.

A final approach to setting threshold values is based on analytic considerations. For example, suppose you model a serial circuit as a simple M/M/1 queue. Recall that if this queue is 90% utilized, then the average wait time is 10x the norm. That's a good reason to generate an alarm. For error rates and packet loss, consider the graph

2. John Blommers, Prentice Hall, 1996, Practical Planning for Network Growth, ISBN 0-13-206111-2

in Figure 9-7. It shows how the TCP throughput of an application performing a continuous TCP data transfer slows as packet loss increases. Even a one percent packet loss effectively reduces throughput to negligible levels.

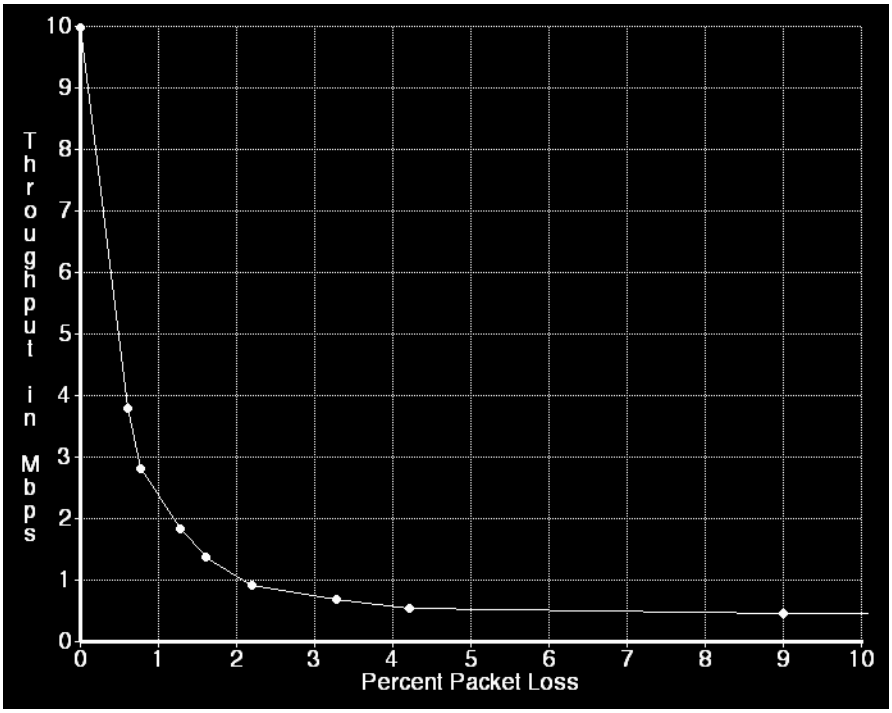

Figure 9-7 TCP throughput vs. packet loss.
An analytic approach to setting error thresholds involves examining empirical data as depicted here. This graph depicts a single streaming TCP data connection between workstations on the same LAN segment which is subject to packet loss. Clearly, the slightest loss percentage effectively negates the streaming advantages of TCP. An error threshold of one percent is justified by this data.

To avoid a random single error burst from generating an alarm, you can take advantage of NNM's clever threshold features. By specifying

that a given metric must exceed the threshold value for four consecutive samples, you ensure that only a sustained error condition will generate an alarm. At five-minute samples this means a 20-minute duration is required to trigger an alarm. To ensure the restoration of service is detected promptly, you can set the duration of the rearm interval to only two samples, or 10 minutes. The principles are explained in Figure 9-8.

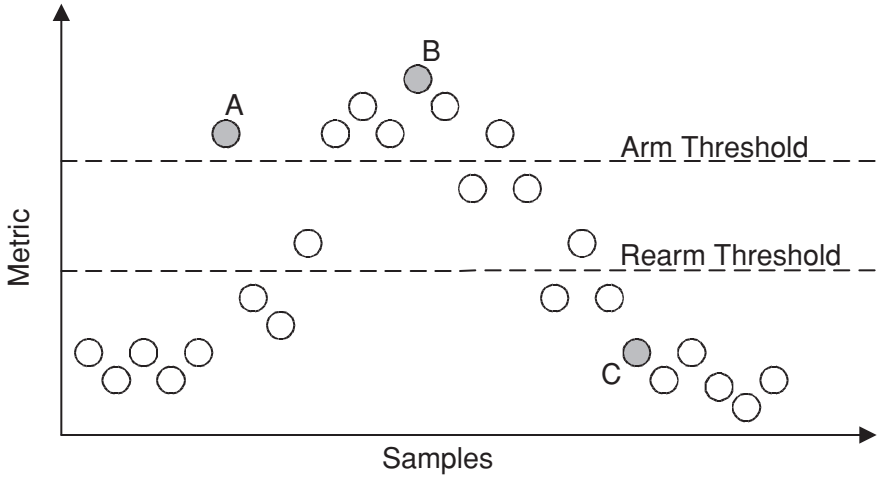

Figure 9-8 NNM threshold parameters.
In the examples in the text you require four consecutive samples to exceed the arm threshold to generate a threshold event. The sample labeled B will generate an event, but sample A will not because only a single consecutive sample has exceeded the arm threshold. Two consecutive samples must be lower than the rearm threshold, so sample C is the only one that qualifies.

How to Create MIB Expressions

Standard MIB2 and enterprise MIBs alike support counter variables for bytes, octets, packets, and errors. Among the really useful metrics you need to manage network performance are percent utilization and percent errors. NNM's historical SNMP data collector predefines quite a few useful MIB expressions. See Chapter 11 of the manual *Managing Your Network with HP OpenView Network Node Manager* for a detailed list of them. A MIB expression is a reverse polish notation (RPN) arithmetic formula made up of standard MIB object identifiers. The expressions are stored in the *$OV_CONF/mibExpr.conf* file, which you maintain manually using your favorite text editor.

Here is an example of a MIB expression:

```
If%deferred \ "packets deferred/packets transmitted" \
.1.3.6.1.4.1.11.2.4.1.1.1.4. .1.3.6.1.4.1.11.2.4.1.1.1.2. / 100 *
```

The label *If%deferred* in the first field refers to the "interface percent deferred" which is intended to calculate the percent of transmitted packets that experienced a deferral before being transmitted. The backward slash is a field delimiter. The text between quotation marks in the second field is a comment which usually contains information about the MIB expression. In this case, it shows us the formula for calculating percent deferred packets. The third field contains the actual formula in an RPN format (a.k.a postfix format). You see five elements in the example above. The first two are the object identifiers from the HP vendor-specific MIB for packets deferred and packets transmitted; they are placed on the stack. The third element is a forward slash which is the divide operator. This places the quotient on the stack. The fourth element is the integer 100 which is placed on the stack. The fifth element is an asterisk which is the multiply operator, and the result is a percentage left on the stack representing the percent deferred packets.

When the *snmpCollect* daemon encounters this MIB expression in a collection, it requests values for all the objects contained in the expression from the device, evaluates the formula, and stores the result in the database.

Here is an additional example of a MIB expression which measures the packet loss in a router:

```
IP%PacketLoss \ "Percent of packets a router discarded\n\
Computed by:\n\
(100*(ipInDiscards+ipOutDiscards/(ipInDiscards+ipOutDiscards+ipForwDatagrams)\n\
which is a percentage." \
100 \
.1.3.6.1.2.1.4.8.0  \
.1.3.6.1.2.1.4.11.0 \
+ * \
.1.3.6.1.2.1.4.8.0  \
.1.3.6.1.2.1.4.11.0 \
.1.3.6.1.2.1.4.6.0  \
+ + /
```

As a practical matter, Cisco routers include those packets in the *ipOutDiscards* metric that could not be forwarded because no ARP cache entry existed for the target IP address in the dropped packet, despite attempts to find the device's MAC address using ARP. In other words, if a packet cannot be delivered to its destination, the router will increment the counter, even though this has nothing to do with buffer overflow, which is what you are hoping to measure. The lesson here is that despite your best efforts at applying common sense in fabricating a MIB expression, you have to be very clear about what the vendor's SNMP implementation really measures.

See Chapter 11 of the *Managing Your Network with HP OpenView Network Node Manager* manual for detailed information on writing and using MIB expression.

Viewing Performance Data Online

The simplest way to look at historical SNMP data is to select the device you're interested in and choose the *Performance:Display SNMP data:For Selected Nodes* menu. This brings up the *xnmgraph* GUI which displays the device performance data found in the database, if any.

To view live SNMP data from several devices on specific interfaces, you can launch the *xnmgraph* application from a command line. For example, to plot *ifInOctets* and *ifOutOctets* for MIB instances 2-5 for both *node1* and *node2*, use this command:

```
xnmgraph -mib \
"interfaces.ifTable.ifEntry.ifInOctets:In Bytes:[2-5]::::::,\
interfaces.ifTable.ifEntry.ifOutOctets:Out Bytes:[2-5]::::::" \
node1 node2
```

You can use the *snmpColDump* application along with some scripting to print out collected SNMP data. Assume that data has been collected using *snmpCollect* under the label *macDeferred.1*. To print out the average value of *macDeferred.1* for *node1,* use this command:

```
snmpColDump $OV_DB/snmpCollect/macDeferred.1 |
awk -F\t '/node1/{num++; sum+=$3} END{print sum/num}'
```

The file *macDeferred.1* contains the historic SNMP data for all nodes and the *awk* script filters out the data for *node1*, calculates the totals, computes the average value, and displays it.

When troubleshooting a performance problem, suppose that you just received a threshold event in the event browser. Double-click the event, and the offending node is displayed in its parent map. Select the node and use the menus to display the performance data. If the graph is cluttered with other data which isn't related to this collection, you can customize the graph by turning off the other lines.

Users wishing to view SNMP historical data in a web browser may use the contributed Java viewer.

Presenting Performance Data to the Consumer

A network manager spends some of the time making presentations about network performance and isn't always in a position to present the data online. NNM performance data needs to be captured and repurposed for offline meetings.

Assuming that NNM historical performance data is available, a few simple screenshots could suffice. (See "Taking Screenshots of Maps" on page 88.) Note that the standard *xnmgraph* window background is black, which lets the other line colors stand out clearly. The background color is controlled by an X-Window resource located in the *$APP_DEFS/XNmgraph* resource file. The resulting file should be transferred from the UNIX NNM system to the authoring workstation in binary format.

Should textually formatted performance data be preferred, *xnmgraph* may be used to save the displayed data in a text file. While UNIX does not require or recognize file name extensions, it may be appropriate to append the *.txt* extension for the benefit of challenged authoring workstations that do need them. Transfer the text file in ASCII format from the NNM system to the authoring workstation. The columnar textual data is usually imported into a spreadsheet or word processor to reformat it for presentation purposes.

The s*nmpColDump* utility may also be used to save specific SNMP performance data into a text file.

SNMPv2c and 64-bit Counters

The standard integer in SNMP MIB2 is 32 bits long. SNMPv2C MIB defines a new type of integer with the attribute name *unsigned64* that is 64 bits long. Its maximum value is:

18,446,744,073,709,551,615

The reason for specifying such a large counter is found in rfc2233:

"As the speed of network media increase, the minimum time in which a 32 bit counter will wrap decreases. For example, a 10Mbs stream of back-to-back, full-size packets causes ifInOctets to wrap in just over 57 minutes; at 100Mbs, the minimum wrap time is 5.7 minutes, and at 1Gbs, the minimum is 34 seconds. Requiring that interfaces be polled frequently enough not to miss a counter wrap is increasingly problematic."

Note that NNM handles a single counter wrap properly, but if two counter wraps occur between SNMP samples, the result is bad data. Recall that you suggested using a five-minute sample interval for SNMP data collections (See "Estimating SNMP Data Sample Rates" on page 149). Here you see that ordinary fast Ethernet interfaces must be sampled at almost exactly this speed to avoid counter wrap in highly utilized situations. Since it is impossible to know in advance how utilized a circuit is, all of them have to be sampled at five-minute intervals to avoid a wrap. If you use the SNMPv2C 64-bit counters, you can tolerate a larger sample interval for SNMP historical data collections.

The new SNMPv2C 64-bit counters are found in the *IfXEntry* branch in the interfaces section of MIB2, as shown in Figure 9-9. You can find detailed definitions for them in *rfc2233*.

Note that NNM makes a standard configuration check for every SNMP-capable device to determine if it supports SNMPv2C. This

```
IfXEntry ::=
       SEQUENCE {
           ifName                      DisplayString,
           ifInMulticastPkts           Counter32,
           ifInBroadcastPkts           Counter32,
           ifOutMulticastPkts          Counter32,
           ifOutBroadcastPkts          Counter32,
           ifHCInOctets                Counter64,
           ifHCInUcastPkts             Counter64,
           ifHCInMulticastPkts         Counter64,
           ifHCInBroadcastPkts         Counter64,
           ifHCOutOctets               Counter64,
           ifHCOutUcastPkts            Counter64,
           ifHCOutMulticastPkts        Counter64,
           ifHCOutBroadcastPkts        Counter64,
           ifLinkUpDownTrapEnable      INTEGER,
           ifHighSpeed                 Gauge32,
           ifPromiscuousMode           TruthValue,
           ifConnectorPresent          TruthValue,
           ifAlias DisplayString,
           ifCounterDiscontinuityTime TimeStamp
       }
```

Figure 9-9 64-bit counters in SNMPv2C.
This is the extended interface section defined by SNMPv2C. Note the Counter64 variables such as *ifHCInOctets* and *ifHCOutOctets* are sufficient such that a 1-terabit/second (1,000 Gbps.) link will cause the counter to wrap in just under five years. The old *ifSpeed* variable is limited to about 2.2Gbps, while the new *ifHighSpeed* variable measures in units of 1,000,000 bits per second.

check can be disabled by specifying the *-2* option in *$OV_CONF/netmon.lrf*. Some devices and systems may misbehave, log a message, or display a warning when they receive an SNMPv2C request. This *netmon* option avoids the problem. It's better in the long term to update the offending SNMP agents while temporarily unmanaging these devices or putting their IP addresses in *netmon.noDiscover*.

It is not advisable to manage gigabit Ethernet without using 64-bit counters.

Collecting RMON Data

The remote monitoring MIB (RMON) provides very detailed performance data. In this section we'll focus on the Ethernet portion of RMON, which is in the original standard. Extensions for token ring and FDDI won't be covered here. Recall that RMON agents are built into special probe devices, such as the HP LANProbe, which are attached to remote shared-medium Ethernet hubs to monitor them, as shown in Figure 9-10. These agents may be found in network

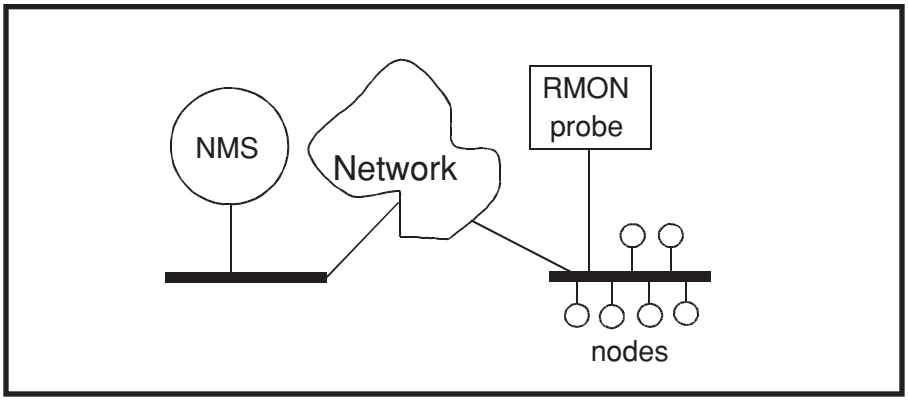

Figure 9-10 A shared medium RMON probe.
The network management station (NMS) uses standard SNMP requests to gather segment statistics from the remote RMON probe. The nodes indicated share the LAN medium and the RMON probe operates in a promiscuous[a] mode to collect statistics such as total bytes, packets, errors, broadcasts, and multicasts. NNM can collect the simpler *Etherstats* group statistics while HP NetMetrix can control the probe to deliver sophisticated, end-to-end performance data across the enterprise network.

a. When it is operating in a promiscuous mode, a LAN adapter is able to capture all LAN frames present on the shared medium, even those not specifically addressed to it. This is a necessary feature for RMON probes and LAN analyzers in general. A LAN adapter is normally able to receive only frames addressed to its MAC address, the all-ones broadcast address, and possibly one or more multicast addresses.

equipment such as switches, where they can monitor all ports. While special software such as HP NetMetrix is best suited to controlling and managing RMON agents, simple Ethernet statistics can be collected using simple SNMP methods.

The Etherstats group of the RMON MIB provides information about the LAN segment the probe is connected to. A shared medium LAN segment behaves exactly as does the traditional Ethernet cable and the Ethernet hub, repeater, or concentrator. All devices share the same medium, so a single RMON probe can monitor all the devices connected to it. A list of the performance variables in the RMON Etherstats group is given in Table 9-3.

Table 9-3 The RMON Etherstats MIB Variables

Variable	Description
etherStatsOctets	The total number of bytes counted.
etherStatsPkts	The total number of packets counted.
etherStatsBroadcastPkts	The number of MAC broadcast packets counted.
etherStatsMulticastPkts	The number of multicast packets counted.
etherstatsCRCAlignErrors	The number of frames with extra bits.
etherstatsUndersizePkts	The number of packets shorter than 64 bytes.
eterstats OversizePkts	The number of packets exceeding 1518 bytes.
etherStatsFragments	The number of collision fragments counted.
etherStatsJabbers	The number of jabber frames (> 8000 bits) seen.
etherStatsCollisions	The total number of collisions counted.
etherStatsPkts64Octets	The number of packets with exactly 64 bytes.

Table 9-3 The RMON Etherstats MIB Variables

Variable	Description
etherStatsPkts65to127Octets	The number of packets between 65 and 127 bytes in size.
etherStatsPkts128to255Octets	The number of packets between 128 and 255 bytes.
etherStatsPkts256to511Octets	The number of packets between 256 and 511 bytes.
etherStatsPkts512to1023Octets	The number of packets between 512 and 1023 bytes.
etherStatsPkts1024to1518Octets	The number of packets between 1024 and 1518 bytes.

Be sure to load the RMON MIB on your NNM system using the *xnmloadmib* GUI before attempting to collect any data from the network. Note that none of these statistics actually measures LAN utilization. A MIB expression is needed that implements the formula:

%utilization = (etherStatsOctets×800)/ifSpeed

For example, for a standard Ethernet, *ifSpeed* is 10,000,000 bits per second, and let's say you measure 600,000 octets per second. (As a practical matter, *snmpCollect* knows that *etherStatsOctets* is a *Counter32* variable, so it actually differences successive samples and then divides by the number of seconds between samples, returning a rate measured in octets per second.) The formula returns 48% utilization. The variable *etherStatsOctets* is in the *rmon* group under the *mib2* branch, while *ifSpeed* is under the *interfaces* group of *mib2*.

Note that standard RMON statistics measure only MAC layer statistics and are network-layer agnostic. To collect network-layer and application-layer statistics requires RMON2 and an application such as HP NetMetrix that can decode, present, and report the data

efficiently. However, for simple shared-medium Ethernet, token ring, and FDDI LANs, standard NNM SNMP data collection suffices for basic segment-level performance management.

Crossing Over into HP NetMetrix

Network Node Manager makes a great launching platform for NetMetrix. NNM discovers all known RMON probes on the network, which NetMetrix can then use. NetMetrix can collect statistics from all these probes in the background. You can then view this historical data. For probes with RMON2 or extended RMON, you can also display traffic between source and destination subnets by application and by system, as shown in Figure 9-11.

Capacity Planning

NNM provides the as-built network topology needed for capacity planning. HP NetMetrix can provide the existing performance metrics. HP Service Simulator can import both topology (see Figure 9-12) and performance data to create a baseline simulation model of your network. What's next, then?

Before you can use the simulator and write a capacity plan, you must have a question to answer. The kinds of questions you might consider are as follows:

- Can the network handle another 1000 users?
- What happens when I consolidate 100 mail hubs into 10?
- What upgrades are needed to support our new ERP application?
- Will adding a second T-1 line meet response time requirements?
- Can the network handle the new print-on-demand technology?

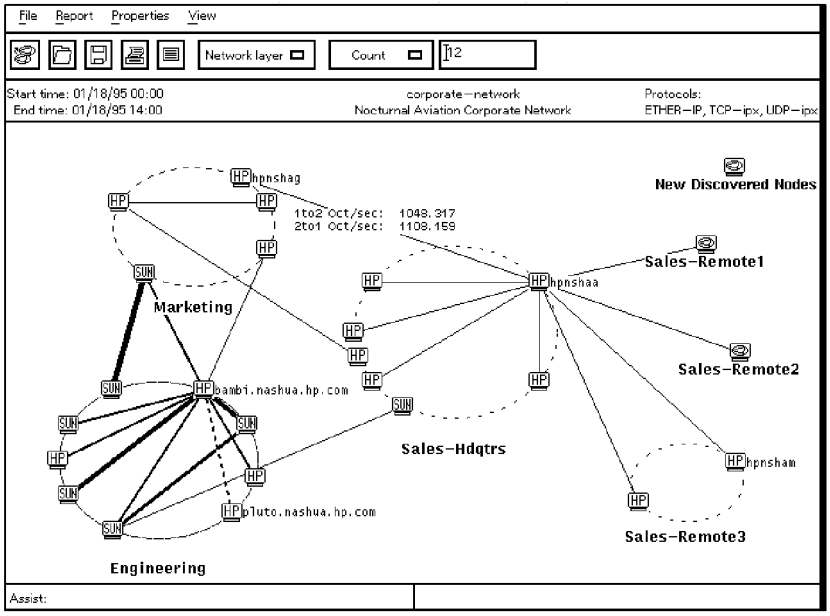

Figure 9-11 NetMetrix Internet view of point-to-point traffic. Properly located RMON2 probes can see traffic between subnets as well as local traffic on each subnet. This diagram shows subnets as ovals. Lines between traffic sources, such as those between the Marketing and Engineering subnets, show how much total traffic flows between devices on these subnets. A thick line represents more traffic than a thin line.

All of these are “what if?” questions. You typically take the baseline simulation model created with Service Simulator, make changes postulated in the question, and let the simulator calculate the impact on response time and circuit utilization (Figure 9-13).

If you are planning to add a new application to the network, you need data about its traffic requirements. You also need LAN analyzer traces of the most commonly used transactions. These can be imported into the simulator. Edit the network topology, move servers around, add new clients and servers, modify line speeds, and run the simulator to see what happens. The basic methodology is shown in Figure 9-14.

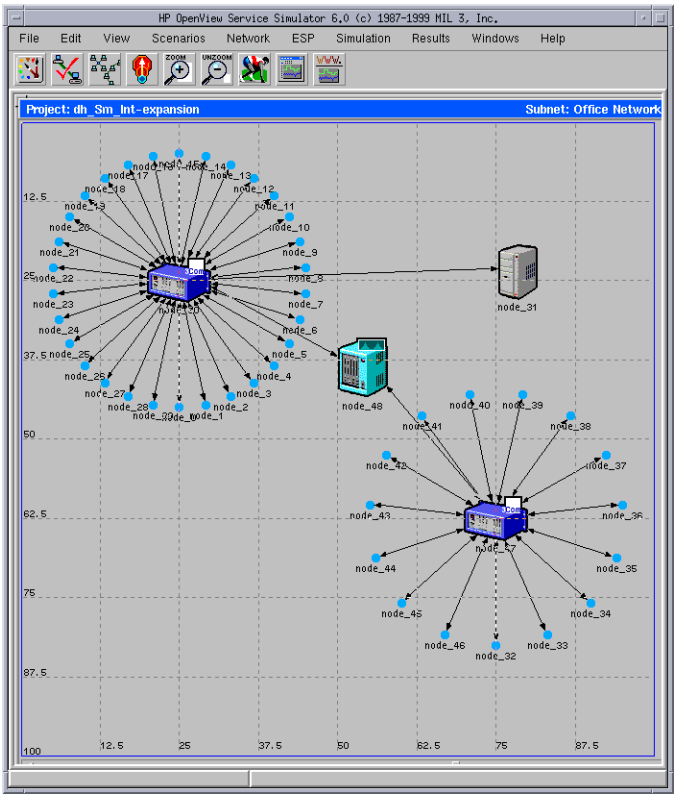

Figure 9-12 A simple network topology in service simulator. This is one of the tutorials provided with the Service Simulator product. Complex networks are dealt with using hierarchical container objects. In practice, the topology can be imported directly from NNM and then edited to define various what-if scenarios. After traffic profiles are created, the simulation is allowed to run until it reaches a steady state.

Do any circuits operate at capacity? Are transaction response times within acceptable limits? If not, then change some parameters and try again.

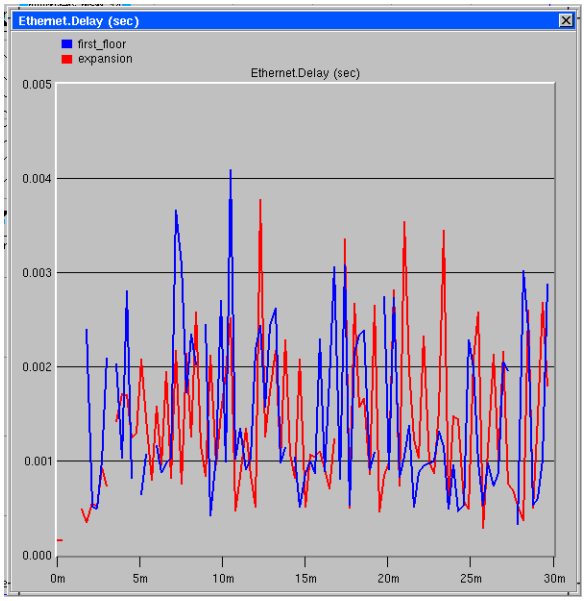

Figure 9-13 Simulation results for Ethernet delay.
One of the outputs of the simulator is the Ethernet delay versus time. You see that the steady state is reached early in the simulation and that the worst delay is only about 4 milliseconds.

Conclusion

Having read this chapter, you are now able to decide at which point between the extremes you want to practice capacity planning. Will you take the time to follow the full-blown approach in Figure 9-14, cross your fingers, and simply provision bandwidth to deal with capacity issues, or will you find a middle ground? Whatever you decide, may all your capacity plans be successful.

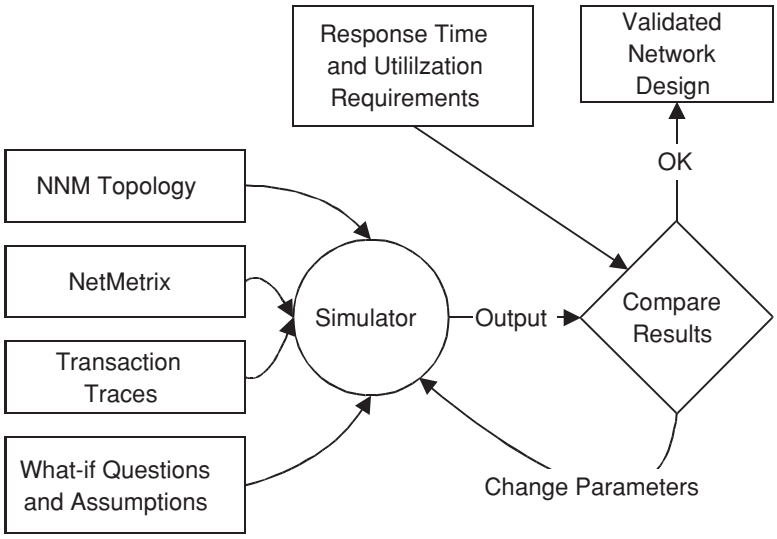

Figure 9-14 Capacity Planning Methodology.
The simulation tool is the core of the capacity planning methodology. A lot of time is saved by importing network topology from NNM, performance data from NetMetrix, and transaction traces from LAN analyzers. Armed with a question, you run the simulator to find out if the network can meet the response time and utilization requirements. If requirements aren't met, go back and move servers around, add bandwidth, change the topology, or reduce the traffic load until they are. When you validate that the network design can deliver the desired performance, you can produce the results to prove it with graphs and charts.