

Network Considerations

NFS is an acronym for “Network File System,” so it should come as no surprise that NFS performance is heavily affected by the latency and bandwidth of the underlying network. Before embarking on a detailed investigation into a specific area of NFS, it is a good idea to first verify that the underlying network is performing as expected.

This chapter focuses on three main areas: analyzing the physical layout of the network that separates your NFS clients and servers, measuring the throughput capabilities of the network, and network troubleshooting concepts.

This chapter describes a recommended methodology and set of tools available for understanding the physical layout of your network, measuring its throughput, and performing routine network troubleshooting tasks. This chapter does not discuss the myriad of networking topologies and interface cards that are currently available for HP-UX systems. NFS runs on most any networking link supporting Internet Protocol (IP), and it typically performs better on faster links.¹

1.1 Analyze Network Layout

An important early step in troubleshooting any NFS performance issue is to learn as much as possible about the physical layout of the underlying network topology. Some of the questions you should be trying to answer at this stage are:

1. There is a wealth of information about the latest and greatest networking technologies, such as Gigabit Ethernet, Auto Port Aggregation (APA), etc., available from HP’s IT Resource Center web site: <http://itrc.hp.com>, and HP’s online documentation repository: <http://docs.hp.com>.

- How many network hops (i.e. bridges, hubs, routers, switches, etc.) do network packets traverse between the client and the server systems?
- What is the speed of each link separating these systems?
- Does your network equipment use auto-negotiation to set speed and duplex settings?
- Are your network interfaces configured for half-duplex or full-duplex mode?
- Do your switch port settings match the speed and duplex settings of your host interfaces?
- What is the maximum transmission unit (MTU) size of the links between these systems?
- If the links are using different MTU sizes, how are the packets being translated? For example, if the NFS client resides in an FDDI ring and uses an MTU size of 4352 and the NFS server uses a 100BT interface with an MTU size of 1500, how are the 4352 byte packets from the client being fragmented into 1500 byte packets for the server?
- Do packets sent from the client to the server take the same route through the network as the packets sent from the server to the client?
- Are your NFS client and server members of a simple Local Area Network (LAN), such as the example shown in Figure 1.1, where the systems are connected to the same switch?

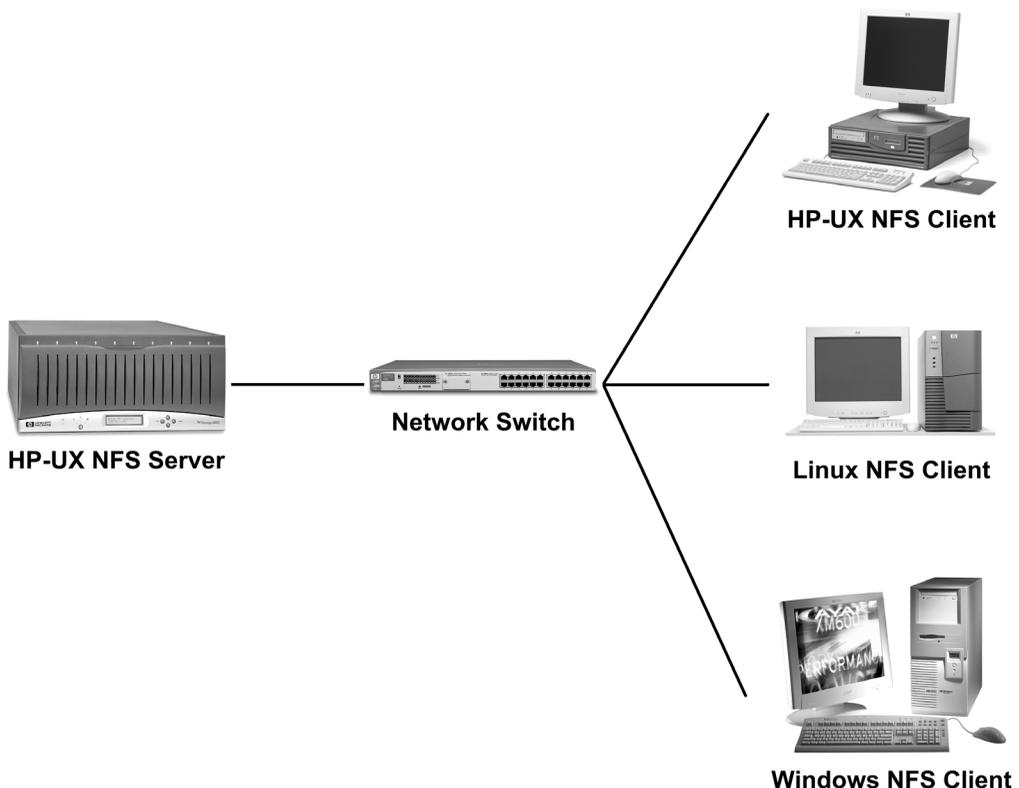


Figure 1.1 NFS Clients and Servers on Same Physical LAN

- Are the systems geographically separated by a Wide Area Network (WAN) such as the example shown in Figure 1.2, where NFS requests and replies must traverse many network switches, routers, and firewalls?

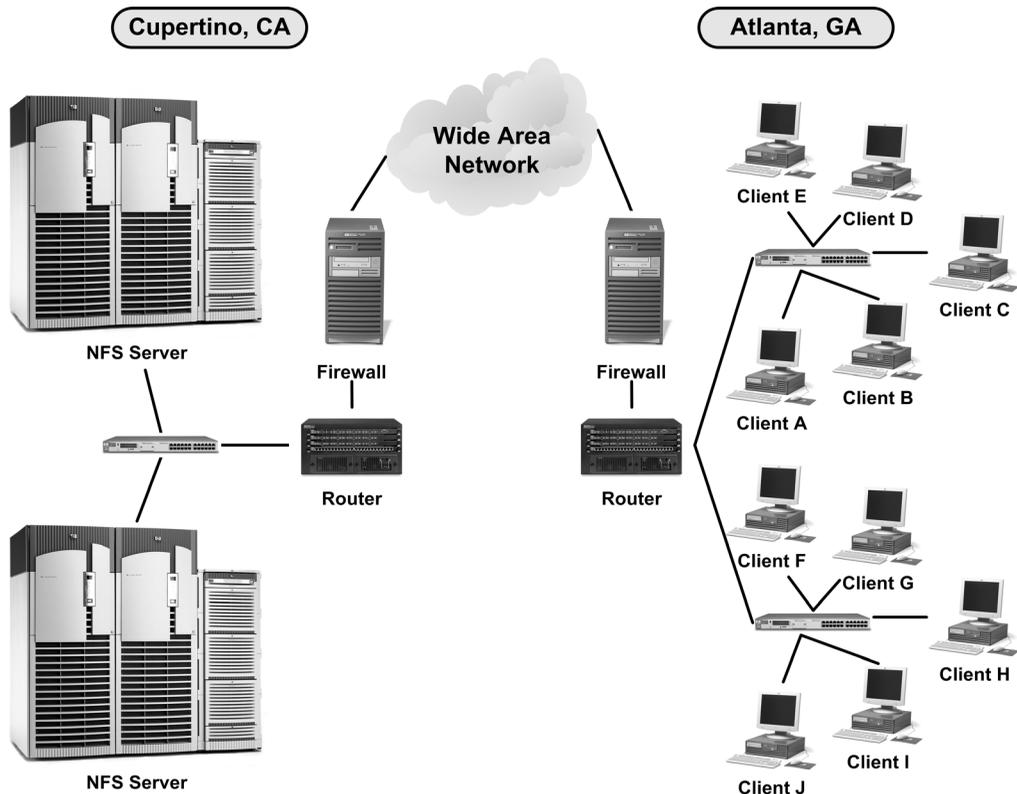


Figure 1.2 NFS Clients and Servers Separated by a WAN

While network administrators should be the best source of knowledge about the layout and capabilities of the underlying network, even they are not always up-to-date on the current state of the network. In many large corporate environments, the physical network is constantly evolving as new equipment replaces old, new backbone technologies replace antiquated technologies, new systems are added to existing networks, new subnets are created, etc. Whenever there is any uncertainty as to the physical layout of the network separating the NFS clients and servers, a network layout analysis should be performed.

Among the best tools available for analyzing network capabilities is the HP OpenView suite of products, such as Network Node Manager.² Even without using a full-blown network

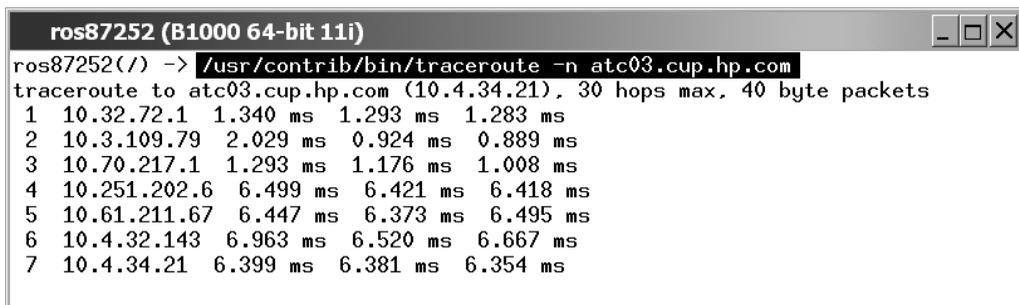
2. For more information about the OpenView product family, visit <http://openview.hp.com>.

management tool such as OpenView, you can still collect a good deal of information about your network topology using tools that ship with HP-UX.

1.1.1 traceroute (1M)

The `traceroute (1M)` tool provides a simple means of determining the path through the network taken by packets sent from one system to another. HP's version of `traceroute` is based on the source code originally developed by Van Jacobson. Since `traceroute` is considered “contributed” software, it resides in the `/usr/contrib/bin` directory.

Figure 1.3 shows a sample screen shot of `traceroute` output displaying the physical path taken between a system located in Roseville, CA and a system located in Cupertino, CA. In this example, and the sample shown in Figure 1.4, `traceroute` is run with the “-n” option, instructing `traceroute` to display IP address information for each network hop without performing address-to-hostname translation.³



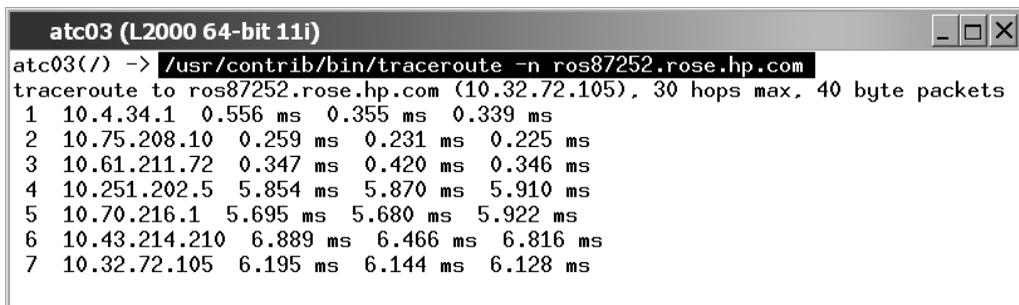
```

ros87252 (B1000 64-bit 11i)
ros87252(/) -> /usr/contrib/bin/traceroute -n atc03.cup.hp.com
traceroute to atc03.cup.hp.com (10.4.34.21), 30 hops max, 40 byte packets
 1 10.32.72.1  1.340 ms  1.293 ms  1.283 ms
 2 10.3.109.79 2.029 ms  0.924 ms  0.889 ms
 3 10.70.217.1 1.293 ms  1.176 ms  1.008 ms
 4 10.251.202.6 6.499 ms  6.421 ms  6.418 ms
 5 10.61.211.67 6.447 ms  6.373 ms  6.495 ms
 6 10.4.32.143 6.963 ms  6.520 ms  6.667 ms
 7 10.4.34.21  6.399 ms  6.381 ms  6.354 ms

```

Figure 1.3 `traceroute` Output from Roseville to Cupertino

Figure 1.4 reveals the network path taken by `traceroute` packets sent from the Cupertino system to the Roseville system.



```

atc03 (L2000 64-bit 11i)
atc03(/) -> /usr/contrib/bin/traceroute -n ros87252.rose.hp.com
traceroute to ros87252.rose.hp.com (10.32.72.105), 30 hops max, 40 byte packets
 1 10.4.34.1  0.556 ms  0.355 ms  0.339 ms
 2 10.75.208.10 0.259 ms  0.231 ms  0.225 ms
 3 10.61.211.72 0.347 ms  0.420 ms  0.346 ms
 4 10.251.202.5 5.854 ms  5.870 ms  5.910 ms
 5 10.70.216.1 5.695 ms  5.680 ms  5.922 ms
 6 10.43.214.210 6.889 ms  6.466 ms  6.816 ms
 7 10.32.72.105 6.195 ms  6.144 ms  6.128 ms

```

Figure 1.4 `traceroute` Output from Cupertino to Roseville

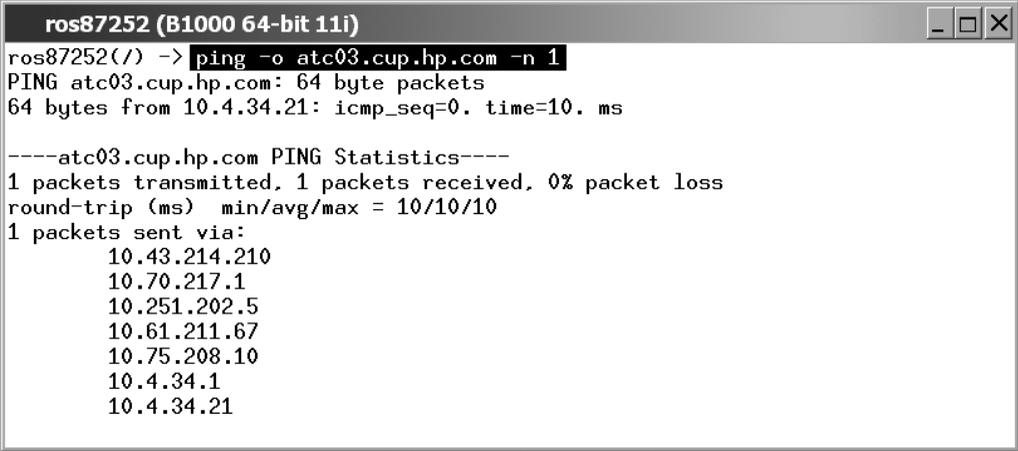
3. For more information about available `traceroute` command-line options, refer to the `traceroute(1M)` man page.

In this example there are six network links separating these two systems (the last line in the `traceroute` output is the actual destination system). The output shows the cumulative latency experienced as the packets transition through the various network links on their way to the final destination. By comparing these two `traceroute` outputs you can see the number of network hops taken in both directions is the same, but the physical path through the network is different in each direction. For example, going to Cupertino the packets went through “10.4.32.143” while on the way to Roseville the packets went through “10.75.208.10.” The latencies experienced in both directions appear comparable.

1.1.2 ping(1M)

Another tool shipping with HP-UX that can simplify the process of collecting network topology information is `ping(1M)`. Michael Muuss originally developed the `ping` program at the US Army Ballistics Research Lab (BRL) in the early 1980’s. It has since become one of the most widely used UNIX networking commands. Most every UNIX administrator has used `ping` at one time or another to quickly verify connectivity between two network nodes. When used with the “-o” option, `ping` inserts an IP Record Route⁴ option into its outgoing packets, allowing `ping` to track and display the physical network path taken between two nodes.

Figure 1.5 shows the “`ping -o`” command displaying the network path taken by packets sent between the same systems used in the earlier `traceroute` example. The “-n 1” option was also used to instruct `ping` to only send a single packet to the remote system.⁵



```
ros87252 (B1000 64-bit 11i)
ros87252(/) -> ping -o atc03.cup.hp.com -n 1
PING atc03.cup.hp.com: 64 byte packets
64 bytes from 10.4.34.21: icmp_seq=0. time=10. ms

----atc03.cup.hp.com PING Statistics----
1 packets transmitted, 1 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 10/10/10
1 packets sent via:
    10.43.214.210
    10.70.217.1
    10.251.202.5
    10.61.211.67
    10.75.208.10
    10.4.34.1
    10.4.34.21
```

Figure 1.5 `ping -o` between Roseville and Cupertino Systems

4. Many network hardware vendors allow the IP Record Route feature to be disabled, effectively making these hops *invisible* to the “`ping -o`” command.
5. For more information about available `ping` command-line options, refer to the `ping(1M)` man page.

Comparing this output against the `traceroute` screen shot in Figure 1.3, you can see that the `ping` packet took the same number of hops to get from Roseville to Cupertino as the `traceroute` packet. However, the `ping` traffic took a different route through the HP network than the `traceroute` traffic. These differences in the network path are most likely the result of dynamic routing tables kept in the routers and bridges used in corporate network backbone.

Unlike `traceroute`, `ping` does not report the individual latencies experienced at each location in the topology. While this information may not be as useful as the `traceroute` output, it provides an easy way of determining the network topology separating two systems.

Once you've determined the physical layout of the network separating your NFS clients and servers, there are several other questions to ask yourself:

- Does the layout make sense to you? In other words, is the network laid out as you expected or were you surprised by what `traceroute` and `ping` revealed?
- Did the physical layout change recently? If so, why?
- Are packets taking the most efficient path through the network? If not, can you force them to take the most efficient path?
- Can any of the network hops separating the clients and servers be eliminated?

If there is any way to optimize the physical layout of the network (i.e. minimize the number of network hops and use the most efficient route between the systems) it can help NFS performance tremendously.

1.2 Measure Network Throughput Capabilities

Once the layout of the physical network is understood, the next step in validating your network is to measure the throughput of the connection separating the client and server. Generally speaking, the faster your network throughput, the better your NFS performance will be.

When testing the performance of the network for NFS purposes, it is essential to eliminate the NFS layer from consideration by simply testing the network transport layer using non-NFS protocols. If a network throughput problem exists between an NFS client and server, the problem would likely affect any network-based application, not just NFS.⁶ It is also important to measure the throughput going in both directions (i.e. client to server, server to client) to make sure the performance is comparable. Similarly, when attempting to characterize network throughput, it is important to eliminate any influence of the local filesystems. It is therefore necessary to select measurement tools that are not dependent upon NFS or filesystem resources.

Several tools exist to help system and network administrators measure the throughput of their network connections. Two of the more prominent tools are `tcp(1)` and `netperf`.

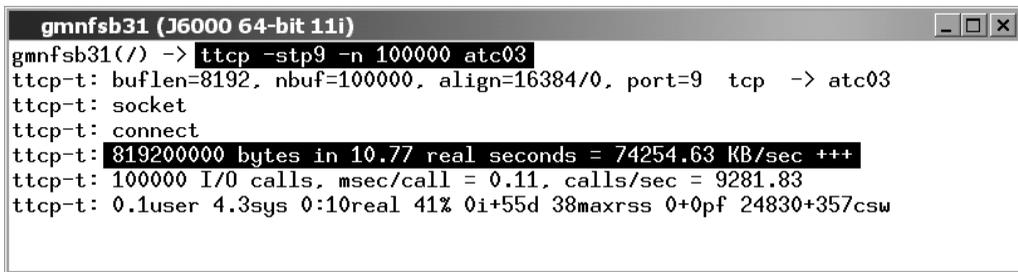
6. A minor packet loss problem may not affect TCP/IP applications such as FTP or NFS/TCP filesystems, but may cause problems for NFS/UDP filesystems. For a detailed explanation of how packet loss affects NFS/UDP and NFS/TCP filesystems differently, refer to Section 10.3 “Managing Retransmissions and Timeouts.”

1.2.1 `ttcp(1)`

`ttcp(1)` is a simple, lightweight program that measures the throughput of any network connection without relying on the filesystem layer. It can generate either UDP or TCP traffic and the packet size is adjustable, allowing it to simulate the behavior of different applications.

Mike Muuss (the author of the `ping` command) and Terry Slattery, of the US Army Ballistics Research Lab (BRL), developed the `ttcp` (Test TCP Connection) program in the early 1980's. The program now resides in the public domain and is freely available to download from <http://ftp.arl.mil/ftp/pub/ttcp>. The `ttcp` man page is also available from this site, which documents the many available command-line arguments.

A sample `ttcp` session is shown in Figure 1.6.



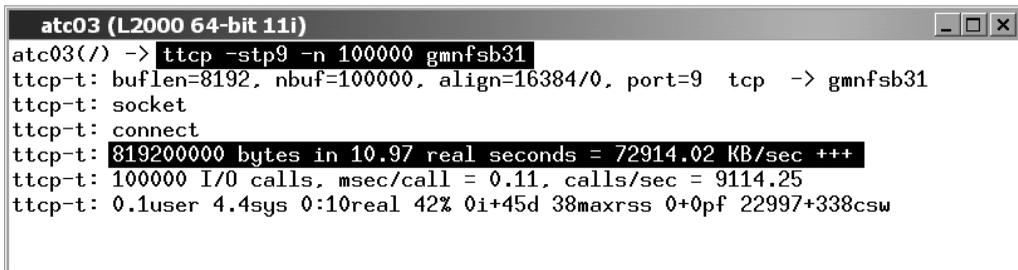
```

gmfnfsb31 (J6000 64-bit 11i)
gmfnfsb31(/) -> ttcp -stp9 -n 100000 atc03
ttcp-t: buflen=8192, nbuf=100000, align=16384/0, port=9 tcp -> atc03
ttcp-t: socket
ttcp-t: connect
ttcp-t: 819200000 bytes in 10.77 real seconds = 74254.63 KB/sec +++
ttcp-t: 100000 I/O calls, msec/call = 0.11, calls/sec = 9281.83
ttcp-t: 0.1user 4.3sys 0:10real 41% 0i+55d 38maxrss 0+0pf 24830+357csw

```

Figure 1.6 `ttcp` Sending TCP Traffic from Client to Server

In this example, `ttcp` is run on the NFS client system and sends TCP/IP traffic to the NFS server's discard port (9) across a Gigabit Ethernet connection. This output shows the client can send over 72MB/sec. on this link. Figure 1.7 shows the server using `ttcp` to send TCP/IP traffic back to the client across the same link and getting roughly the same throughput.



```

atc03 (L2000 64-bit 11i)
atc03(/) -> ttcp -stp9 -n 100000 gmfnfsb31
ttcp-t: buflen=8192, nbuf=100000, align=16384/0, port=9 tcp -> gmfnfsb31
ttcp-t: socket
ttcp-t: connect
ttcp-t: 819200000 bytes in 10.97 real seconds = 72914.02 KB/sec +++
ttcp-t: 100000 I/O calls, msec/call = 0.11, calls/sec = 9114.25
ttcp-t: 0.1user 4.4sys 0:10real 42% 0i+45d 38maxrss 0+0pf 22997+338csw

```

Figure 1.7 `ttcp` Sending TCP Traffic from Server to Client

The system or network administrator now has a fairly good idea what the capabilities are of the network link between these two systems. Of course, this does not imply that the NFS throughput should be 72-73MB/sec. across this link, since the NFS protocol relies much more on CPU, filesystem, and memory resources than `ttcp` does.

1.2.2 netperf

netperf is a benchmark utility that can measure the performance of many different types of networks. It provides tests for both unidirectional throughput and end-to-end latency. Like `ttcp`, netperf measures throughput without relying on any filesystem resources. However, it is a far more sophisticated network-measuring tool, compared to `ttcp`.

Rick Jones, of HP's Infrastructure Solutions and Partners group, developed netperf in 1991 and he has continued to add new features and capabilities to the program as new networking technologies became available. The best source of information is the official netperf web site: <http://netperf.org>. The latest version of the source code is available at the following location: <ftp://ftp.cup.hp.com/dist/networking/benchmarks/netperf>.

Figure 1.8 shows the screen output of netperf sending TCP and UDP data from system "atc03.cup.hp.com" (Cupertino, CA) to three different network destinations:

- Gigabit Ethernet connection to "atc01.cup.hp.com" (Cupertino, CA)
- 100Mb Ethernet connection to "atc01.cup.hp.com" (Cupertino, CA)
- WAN connection to "ros87252.rose.hp.com" (Roseville, CA)

```

atc03 (L2000 64-bit 11i)
atc03(/opt/netperf) -> netperf -fM -Hatc01-ge -tUDP_STREAM -- -m1472
UDP UNIDIRECTIONAL SEND TEST to atc01-ge
Socket Message Elapsed      Messages
Size   Size   Time      Okay Errors   Throughput
bytes  bytes  secs      #      #      MBytes/sec

65535   1472   10.01     527813    0      74.05
65535   1472   10.01     527719    0      74.04

atc03(/opt/netperf) -> netperf -fM -Hatc01 -tTCP_STREAM -- -s65536 -S65536
TCP STREAM TEST to atc01
Recv  Send  Send
Socket Socket Message Elapsed
Size  Size  Size   Time   Throughput
bytes bytes bytes  secs.  MBytes/sec

65536 65536 65536  10.02  11.31

atc03(/opt/netperf) -> netperf -fM -Hros87252.rose.hp.com -tTCP_STREAM
TCP STREAM TEST to ros87252.rose.hp.com
Recv  Send  Send
Socket Socket Message Elapsed
Size  Size  Size   Time   Throughput
bytes bytes bytes  secs.  MBytes/sec

32768 32768 32768  10.03  1.98

```

Figure 1.8 netperf Throughput from Different Network Connections

In this example, the UDP traffic sent over a Gigabit Ethernet connection to a neighbor system in Cupertino yielded a throughput of over 74 MB/sec. Sending TCP traffic to the same Cupertino-based system over a 100BT link yielded over 11MB/sec. Finally, sending TCP traffic across HP's network from Cupertino to Roseville (via the six hops identified earlier with `ping` and `traceroute`) yielded just under 2 MB/sec.

Also evident from this example is the fact that `netperf` has a myriad of command-line options, both for the `netperf` command itself and for the specific type of test you are performing. Be sure to read the `netperf` manual, available at <ftp://ftp.cup.hp.com/dist/networking/benchmarks/netperf>, very carefully to understand which options are available and their proper usage, as the throughput numbers can vary greatly depending upon how the tests are run.

Just as with the `ttcp` example earlier, `netperf` numbers do not directly translate into the expected NFS throughput values. However, they do provide a good means of estimating the upper bounds of a given network connection's bandwidth.

1.3 Network Troubleshooting Tools

At this point, you should be very familiar with the network topology separating the NFS client and server, and have verified that IP packets are taking the appropriate route through the network in both directions. If you have not yet performed these critical steps, refer to the earlier Section 1.1 “Analyze Network Layout” for instructions on how to collect this information. After running network throughput tests using the tools described in Section 1.2 “Measure Network Throughput Capabilities,” if you believe your network is experiencing a performance throughput issue then it is time to troubleshoot the network itself.



KEY IDEA — Common Causes of Dropped Network Packets

In many cases, network throughput problems are caused by packets being dropped somewhere on the network, either by the NFS client or server system itself or at some intermediate point in the network separating the two systems. Some of the more common reasons network packets are dropped include:

- Defective hardware (i.e. network interface cards, cables, switch ports, etc.)
- Mismatching configuration settings between interface cards and switch equipment. The most common configuration issue is where one side of a connection is set to half-duplex and the other side to full-duplex, causing “late” collisions to be logged on the half-duplex side and FCS or CRC errors logged on the full-duplex side.^a
- Network interconnect device buffer memory exhaustion (described in Section 10.4)
- UDP socket overflows occurring on the NFS server, indicating that not enough daemons are running to handle the inbound requests for a specific port

a. The `lanadmin(1M)` tool, described in Section 1.3.5, displays duplex settings and reports any link-level errors.

The goal of this phase of the investigation is to determine if the network throughput problem is affecting all IP traffic or only NFS. In some cases, the only tools that can detect these types of problems are external analyzers and reporting tools specific to your network hardware. HP does provide a number of software-based tools to help detect and analyze network problems. Two frequently used network troubleshooting tools are `netstat(1)` and `lanadmin(1M)`.

1.3.1 netstat -s

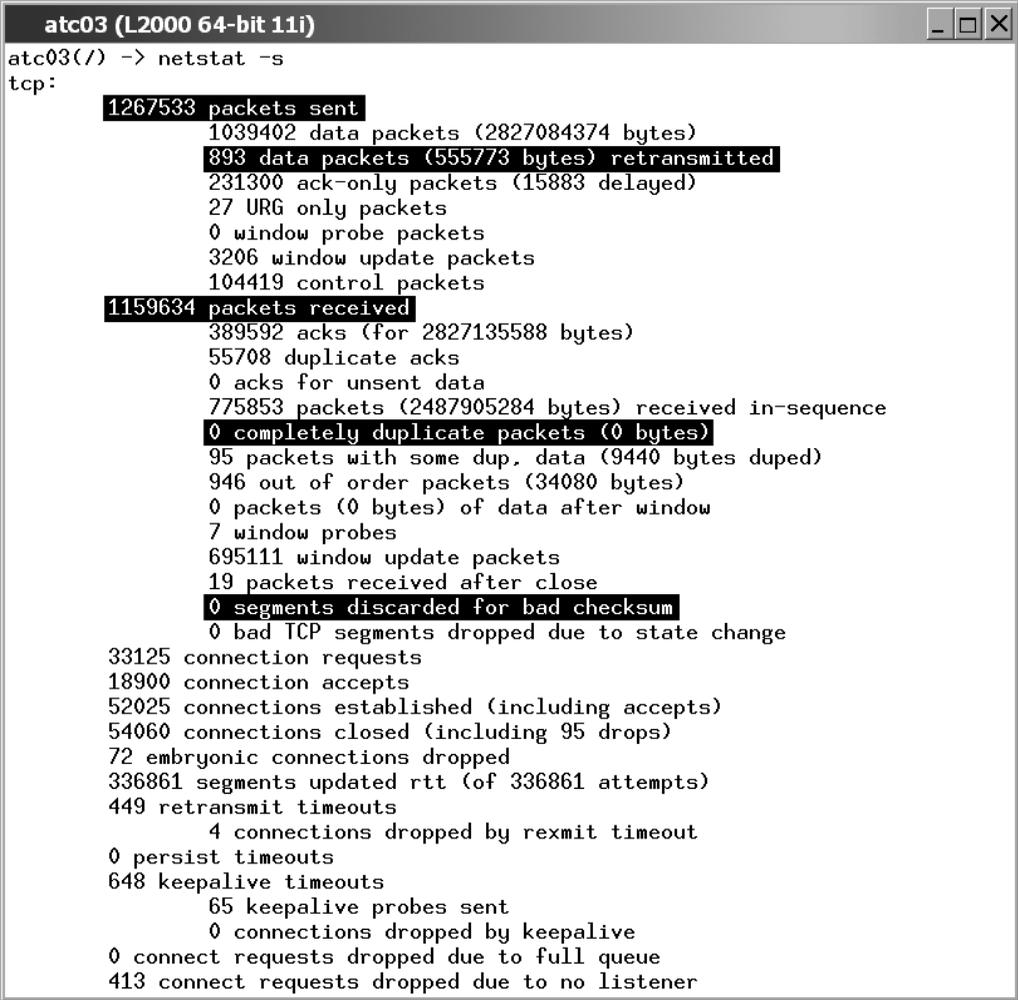
The `netstat(1)` command can be used to display statistics for network interfaces and protocols, it can list active network connections, print routing tables, etc. When executed with the “-s” option, `netstat` returns a complete list of all network transport statistics (TCP, UDP, IP, ICMP, and IGMP) arranged by protocol.

A portion of “`netstat -s`” output is shown in Figure 1.9 and Figure 1.10. These two screen shots illustrate how this single command returns an enormous amount of information about the underlying network protocols, including the number of TCP packets sent and received, the number of UDP socket overflows and checksum failures that occurred, etc. Also readily available are statistics such as the total number of IP packets received, the ICMP port unreachable and source quench messages generated, etc. All of this information can be extremely useful when troubleshooting a network or protocol layer problem.

Using netstat and diff to troubleshoot a network problem Listed below is an example of how to use the “`netstat -s`” command to help determine if packets are being lost somewhere in your network. The underlined steps are the commands you type.

1. Initialize the “before” file with the current date and time.
`# date > netstat.before`
2. Collect a baseline set of `netstat -s` statistics on both the NFS client and server and append the output to the “before” file created in step 1.
`# netstat -s >> netstat.before`
3. Perform a test that exhibits the performance problem using the TCP protocol (such as `ttcp` or `netperf`).
`# ttcp -stp9 -n 100000 server`
4. Initialize the “after” file with the current date and time.
`# date > netstat.after`
5. Collect a second set of `netstat -s` statistics on both the NFS client and server and append the output to the “after” file created in step 4.
`# netstat -s >> netstat.after`
6. Locate any differences between the “before” and “after” `netstat` outputs to identify which statistics were incrementing during the test.
`# diff netstat.before netstat.after`

An example of the type of output returned by `diff (1)` from such an exercise is shown in Figure 1.11. Some of the TCP statistics to monitor are “data packets retransmitted,” “completely duplicate packets,” and “segments discarded for bad checksum.” If these statistics are steadily increasing over time it would indicate that packet loss is occurring somewhere in the network or that a possible network hardware problem is causing TCP checksum failures.



```

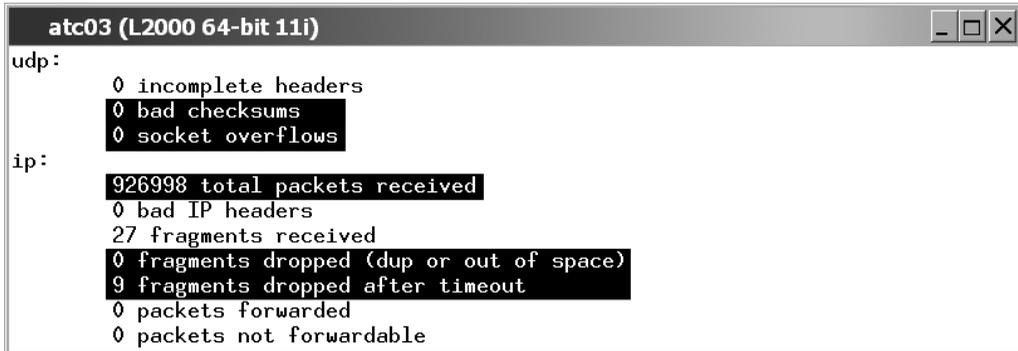
atc03 (L2000 64-bit 11i)
atc03(/) -> netstat -s
tcp:
  1267533 packets sent
    1039402 data packets (2827084374 bytes)
    893 data packets (555773 bytes) retransmitted
    231300 ack-only packets (15883 delayed)
    27 URG only packets
    0 window probe packets
    3206 window update packets
    104419 control packets
  1159634 packets received
    389592 acks (for 2827135588 bytes)
    55708 duplicate acks
    0 acks for unsent data
    775853 packets (2487905284 bytes) received in-sequence
    0 completely duplicate packets (0 bytes)
    95 packets with some dup. data (9440 bytes duped)
    946 out of order packets (34080 bytes)
    0 packets (0 bytes) of data after window
    7 window probes
    695111 window update packets
    19 packets received after close
    0 segments discarded for bad checksum
    0 bad TCP segments dropped due to state change
  33125 connection requests
  18900 connection accepts
  52025 connections established (including accepts)
  54060 connections closed (including 95 drops)
  72 embryonic connections dropped
  336861 segments updated rtt (of 336861 attempts)
  449 retransmit timeouts
    4 connections dropped by rexmit timeout
  0 persist timeouts
  648 keepalive timeouts
    65 keepalive probes sent
    0 connections dropped by keepalive
  0 connect requests dropped due to full queue
  413 connect requests dropped due to no listener

```

Figure 1.9 netstat -s Output Showing TCP Statistics

Also of concern would be an increasing number of UDP “bad checksums,” as this could indicate that an IP level device in the network (perhaps a router performing IP fragmentation) is not correctly fragmenting UDP datagrams as it forwards them. These datagrams would be

discarded by the receiving system, forcing the sending system to re-send this data. UDP “socket overflows” usually indicates that an application, such as NFS, is receiving requests on a particular UDP socket faster than it can process them, and consequently discarding requests.⁷

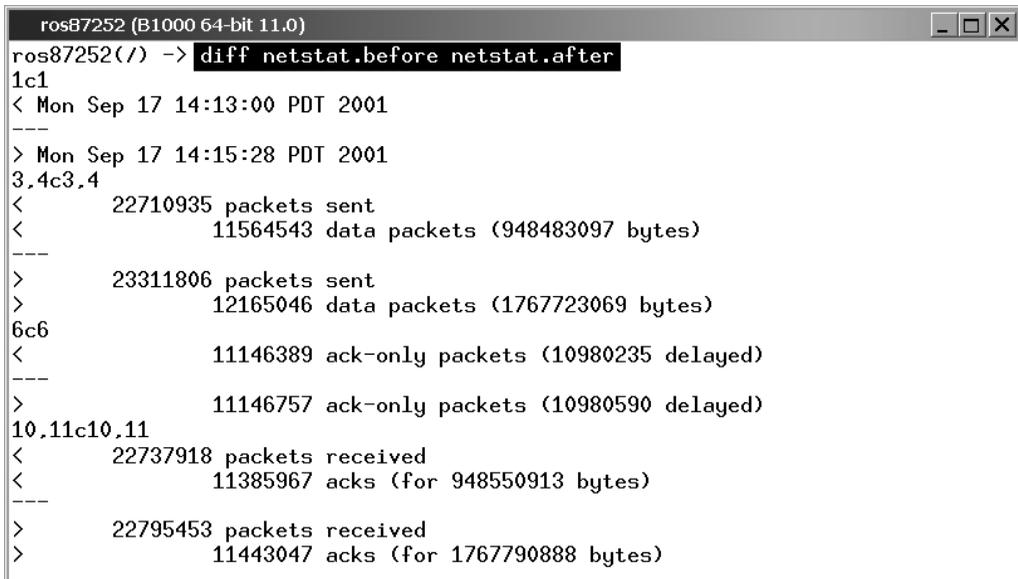


```

atc03 (L2000 64-bit 11i)
udp:
    0 incomplete headers
    0 bad checksums
    0 socket overflows
ip:
    926998 total packets received
    0 bad IP headers
    27 fragments received
    0 fragments dropped (dup or out of space)
    9 fragments dropped after timeout
    0 packets forwarded
    0 packets not forwardable
  
```

Figure 1.10 netstat -s Output Including UDP and IP Statistics

Of the IP statistics reported, “fragments dropped” and “fragments dropped after timeout” would indicate packet loss is occurring in the network.



```

ros87252 (B1000 64-bit 11.0)
ros87252(/) -> diff netstat.before netstat.after
1c1
< Mon Sep 17 14:13:00 PDT 2001
---
> Mon Sep 17 14:15:28 PDT 2001
3,4c3,4
<      22710935 packets sent
<          11564543 data packets (948483097 bytes)
---
>      23311806 packets sent
>          12165046 data packets (1767723069 bytes)
6c6
<          11146389 ack-only packets (10980235 delayed)
---
>          11146757 ack-only packets (10980590 delayed)
10,11c10,11
<      22737918 packets received
<          11385967 acks (for 948550913 bytes)
---
>      22795453 packets received
>          11443047 acks (for 1767790888 bytes)
  
```

Figure 1.11 Comparing “before” and “after” netstat Outputs

7. The effect of UDP socket overflows on NFS performance is discussed in greater detail in Chapter 4.

Using `netstat` and `beforeafter` to troubleshoot a network problem

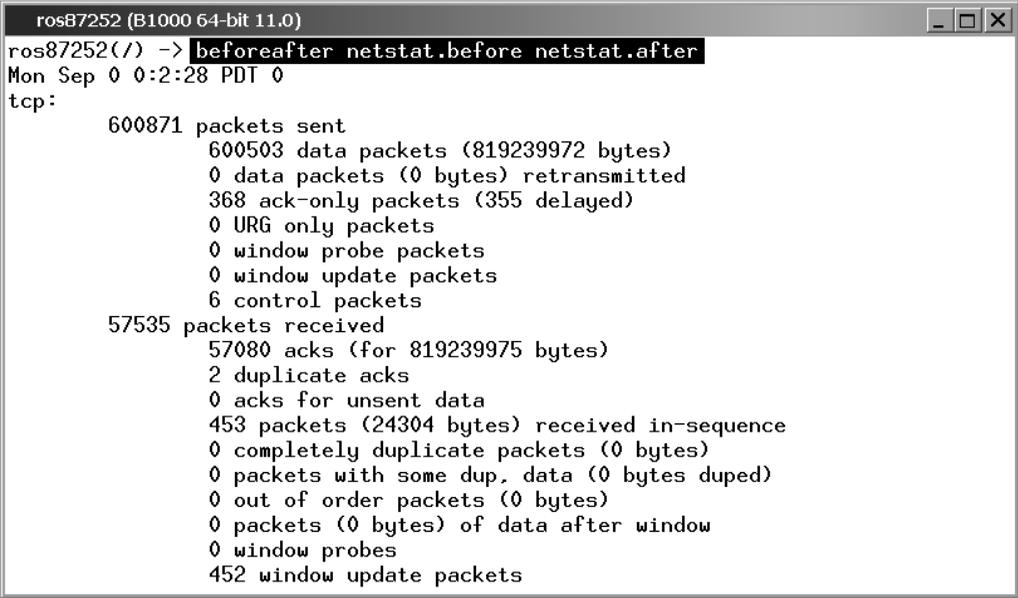
Although the `diff(1)` procedure does work for locating differences in `netstat` outputs, interpreting `diff` output can be a bit cumbersome. Not only do you have to manually subtract the “after” numbers from the “before” numbers, since the `diff` output removes the subsystem header information lines, you need to carefully confirm which statistics you are interpreting. For example, `netstat` returns “# packets received” under the TCP heading and “# total packets received” under the IP heading. Confusing these two values could lead to incorrect conclusions about the health of your network.

To simplify this procedure of interpreting multiple sets of `netstat` output, HP developed a tool called `beforeafter`. This program takes two `netstat -s` output files and compares them against each other. The output from `beforeafter` looks identical to `netstat -s` output except that the statistics represent only the differences between the “before” and “after” files. The tool is available at: <ftp://ftp.cup.hp.com/dist/networking/tools/beforeafter.tar.gz>.

The procedure for using the `beforeafter` tool is the same as the `diff` method outlined earlier, with the exception of step 6. Instead of using the `diff` command to compare the files, the `beforeafter` tool should be used as follows:

```
# beforeafter netstat.before netstat.after
```

Figure 1.12 contains an example of the `beforeafter` output. Notice the output looks identical to that of “`netstat -s`”; however the statistics reported are not cumulative totals but instead represent the differences between the “before” and “after” files.



```
ros87252 (B1000 64-bit 11.0)
ros87252(/) -> beforeafter netstat.before netstat.after
Mon Sep 0 0:2:28 PDT 0
tcp:
    600871 packets sent
        600503 data packets (819239972 bytes)
        0 data packets (0 bytes) retransmitted
        368 ack-only packets (355 delayed)
        0 URG only packets
        0 window probe packets
        0 window update packets
        6 control packets
    57535 packets received
        57080 acks (for 819239975 bytes)
        2 duplicate acks
        0 acks for unsent data
        453 packets (24304 bytes) received in-sequence
        0 completely duplicate packets (0 bytes)
        0 packets with some dup. data (0 bytes duped)
        0 out of order packets (0 bytes)
        0 packets (0 bytes) of data after window
        0 window probes
        452 window update packets
```

Figure 1.12 `beforeafter` Comparing `netstat -s` Statistics

The `beforeafter` tool even calculates the difference in wall-clock times between the date contained in the “before” and “after” files. Looking at the date line in Figure 1.12 you can see that the “after” file was collected 2 minutes and 28 seconds after the “before” file. The test caused this system to send 600503 packets containing 819239972 bytes of data, and receive 57535 packets containing acknowledgement packets for 819239975 bytes.

Although the `diff` output and the `beforeafter` output reveal the same information, locating and quantifying the key statistics is much easier using the `beforeafter` tool.

1.3.2 `netstat -p <protocol>`

Once you have identified a subset of the `netstat -s` output that is of particular interest, you can limit the statistical output to a single protocol by using the “`-p <protocol>`” syntax. An example is shown in Figure 1.13. Confining the output to a single protocol can greatly simplify the process of identifying specific protocol-related problems compared to analyzing screens full of “`netstat -s`” output.

A screenshot of a terminal window titled "atc03 (L2000 64-bit 11i)". The prompt is "atc03(/) ->" and the command entered is "netstat -p udp". The output shows statistics for UDP: "udp: 11 incomplete headers, 0 bad checksums, 11 socket overflows".

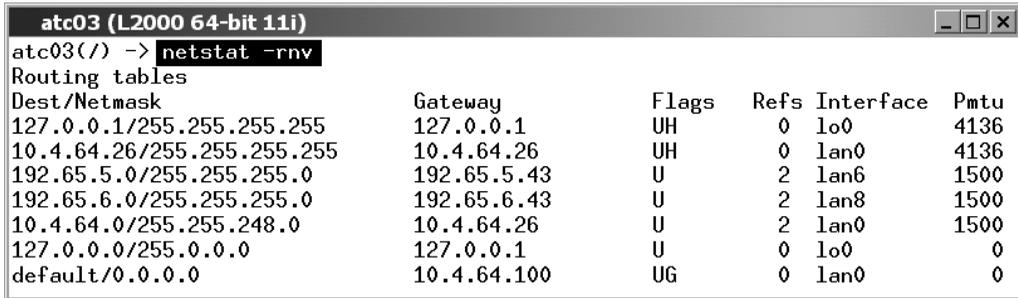
```
atc03 (L2000 64-bit 11i)
atc03(/) -> netstat -p udp
udp:
    11 incomplete headers
     0 bad checksums
    11 socket overflows
```

Figure 1.13 `netstat -p` Output

1.3.3 `netstat -r`

As discussed earlier in the “Analyze Network Layout” section, it is critical to understand the path NFS packets take through the network as they move between the client and server. We saw how utilities such as `traceroute` and “`ping -o`” display the various hops taken by packets going between two network nodes. If the `traceroute` or “`ping -o`” output reveals that packets are not taking the route you expect them to, you should verify the routing tables on both the client and server to make sure they are correct. Ensuring the accuracy of the routing tables is especially important on systems with multiple network interfaces, where outbound packets potentially have several paths to their final destination.

On HP systems, the “`netstat -r`” command is used to display the routing tables. Figure 1.14 shows an example of this. In this example, the “`-n`” (do not resolve IP addresses to hostnames) and “`-v`” (verbose) options were used. Included in the output is the interface name associated with each IP address, as well as the PMTU (Path Maximum Transmission Unit) size for each interface. The MTU information can be very useful in environments where the NFS client and server are on different physical networks and packet fragmentation or translation needs to occur (for example FDDI to Ethernet).



```

atc03(L2000 64-bit 11i)
atc03(/) -> netstat -r
Routing tables
Dest/Netmask      Gateway          Flags   Refs  Interface  Pmtu
127.0.0.1/255.255.255.255  127.0.0.1      UH      0    lo0        4136
10.4.64.26/255.255.255.255  10.4.64.26     UH      0    lan0       4136
192.65.5.0/255.255.255.0    192.65.5.43    U       2    lan6       1500
192.65.6.0/255.255.255.0    192.65.6.43    U       2    lan8       1500
10.4.64.0/255.255.248.0     10.4.64.26     U       2    lan0       1500
127.0.0.0/255.0.0.0        127.0.0.1      U       0    lo0        0
default/0.0.0.0            10.4.64.100    UG      0    lan0       0

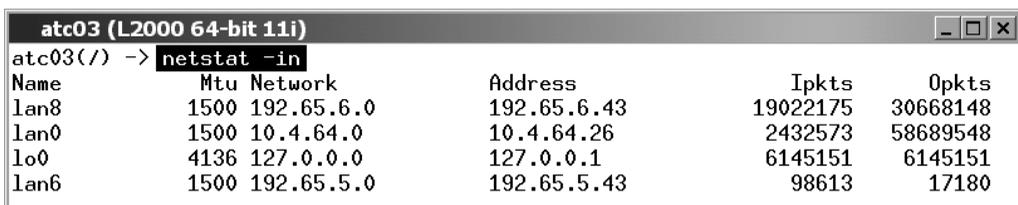
```

Figure 1.14 netstat -r Displaying Network Routing Tables

1.3.4 netstat -i

In large customer environments, particularly those where HP's MC/ServiceGuard⁸ product is used, it is not uncommon for NFS client and server systems to have multiple network paths to each other for redundancy reasons. In some cases the primary and backup interfaces are not equivalent in terms of their bandwidth capabilities. For example, the systems might use a Gigabit Ethernet interface as their primary connection and have a 100BT interface available as a backup connection. In these environments, sufficient care must be taken when configuring the NFS mount points to ensure that the traffic flows across the faster interface whenever possible. However, even with careful preparation, there is always a possibility that NFS traffic sent between the clients and servers will mistakenly use the slower interface.

A quick and easy way to verify which interface the majority of network traffic is using is to issue the "netstat -i" command and examine the inbound and outbound packet counts for all configured interfaces. Figure 1.15 provides an example of this output.



```

atc03(L2000 64-bit 11i)
atc03(/) -> netstat -in
Name      Mtu Network      Address      IpKts      OpKts
lan8      1500 192.65.6.0    192.65.6.43 19022175   30668148
lan0      1500 10.4.64.0     10.4.64.26  2432573    58689548
lo0       4136 127.0.0.0     127.0.0.1   6145151    6145151
lan6      1500 192.65.5.0    192.65.5.43 98613      17180

```

Figure 1.15 netstat -in Output

By monitoring the inbound and outbound packet rates of the interfaces, you can quickly determine if an unusually high amount of network traffic is using what should be an "idle" or "backup" interface. If this appears to be happening, a network trace can be taken to determine the hostnames of the remote systems that are sending requests to the slower interface.

8. For more information about HP's MC/ServiceGuard product or configuring your servers for Highly Available NFS access, visit the HP-UX High Availability web site: <http://hp.com/go/ha>.

1.3.5 lanadmin (1M)

As stated earlier, dropped packets on the network can occur if there are problems with the network interface cards, cables, or connectors. While many hardware-based problems can only be detected and identified with external analyzers, HP-UX provides several software-based tools to help monitor the health of the interfaces. The commands available for checking the state of any specific interface card will vary based on interface type (i.e. FDDI, Gigabit Ethernet, etc.). However, the `lanadmin(1M)` utility applies to all network links and it should be queried first.

The `lanadmin` command allows a system administrator to display many useful statistics kept by the LAN driver subsystem, regardless of the interface type. Figure 1.16 shows a sample screen output returned by `lanadmin`.

```

ros87252 (B1000 64-bit 11.0)
LAN INTERFACE STATUS DISPLAY
Sun, Mar 25, 2001 11:42:20

PPA Number                = 0
Description                = lan0 Hewlett-Packard 10/100 TX Full-Duplex TT = 1500
Type (value)              = ethernet-csmacd(6)
MTU Size                  = 1500
Speed                     = 100000000
Station Address           = 0x1083f937cf
Administration Status (value) = up(1)
Operation Status (value)  = up(1)
Last Change               = 16777321
Inbound Octets            = 1997507736
Inbound Unicast Packets   = 560074
Inbound Non-Unicast Packets = 14647962
Inbound Discards          = 0
Inbound Errors            = 0
Inbound Unknown Protocols = 12684844
Outbound Octets           = 114217800
Outbound Unicast Packets  = 248538
Outbound Non-Unicast Packets = 5175
Outbound Discards         = 0
Outbound Errors           = 0
Outbound Queue Length     = 0
Specific                  = 655367

Ethernet-like Statistics Group

Index                     = 1
Alignment Errors          = 0
FCS Errors                 = 0
Single Collision Frames   = 0
Multiple Collision Frames = 0
Deferred Transmissions     = 0
Late Collisions           = 0
Excessive Collisions      = 0
Internal MAC Transmit Errors = 0
Carrier Sense Errors      = 0
Frames Too Long           = 0
Internal MAC Receive Errors = 0

```

Figure 1.16 lanadmin Output

By reviewing this information you can learn a great deal about how the queried interface is configured and whether it has been logging any errors at the driver layer. For example, the output shown in Figure 1.16 indicates that this interface card is a 10/100BT card known to the system as device “lan0,” the card is enabled and active, it is running at a speed of 100 Mbits/second with an MTU size of 1500, and it is currently configured to run in full-duplex mode. In some cases, this information alone can be enough to determine the cause of a network performance problem (i.e. in the case where LAN interfaces and network switch configurations don’t match with regards to speed and duplex settings).

Also available in the lanadmin output are various error counts, collision rates, total inbound and outbound packet counts, etc. By monitoring these counters you can, with the assistance of an HP support representative, try to make a qualified determination as to whether a hardware problem exists somewhere in your network. When software-based analysis tools fail to identify the problem, external tools can be used to provide the definitive view of the traffic patterns on the network and to isolate a device that is losing packets.



KEY IDEA — The Importance of Patching LAN, Transport, and Network Drivers

HP continually strives to improve the quality of HP-UX by distributing software patches containing both defect fixes and functionality enhancements. Many of these fixes and enhancements can significantly improve the performance and behavior of critical system components, such as LAN Common, the Network Transports (TCP, UDP, IP), and the various Network Link Driver subsystems (100BT, 1000BT, FDDI, Token Ring, etc.).

Since NFS relies heavily upon the stability and performance of the network, it is *strongly* recommended that the latest LAN, Transport, and network link driver patches be installed on every HP-UX system in order to take advantage of these improvements. Contact HP support to obtain a current set of patches for your specific operating system. You can also generate a current patch list using the tools available at HP’s IT Resource Center: <http://itrc.hp.com>.

For a detailed discussion on the importance of keeping your HP-UX NFS client and server systems patched with current code, refer to Appendix B “Patching Considerations.”