Chapter 4

Managing a Network

This chapter provides information on how to manage a network after installation. Management includes maintenance, monitoring, and problem isolation. This chapter provides brief descriptions of the various network management tools, help in interpreting network statistics, a discussion of the factors that can affect network performance, and some of the network kernel configuration options. The following topics are covered:

- "Network Startup and Shutdown" on page 67.
- "Network Management Tools" on page 69
- "Interpreting Network Statistics" on page 72
- "Troubleshooting Poor Network Performance" on page 74

Network Startup and Shutdown

The main network script is */etc/init.d/network*. Other scripts for other network applications (UUCP, mail, and so on) also reside in this directory, but are covered in their appropriate chapter in this guide. A brief description of the *network* script is provided.

The *network* master script is called during system startup and shutdown. It defines the system name and host ID, ensures that the system has a valid Internet address, starts networking daemons, and initializes the network interfaces. Site-dependent configuration commands to start and stop local daemons, add static routes, and publish ARP entries should be put in a separate shell script called */etc/init.d/network.local*. Make symbolic links from */etc/rc0.d* and */etc/rc2.d* to */etc/init.d/network.local* so the *network.local* file is called at system startup and shutdown (see "Creating a Local Network Script" on page 53 for setup procedure).

The *network* master script is linked to */etc/rc0.d/K40network*, which is invoked from */etc/rc0* during shutdown, and to */etc/rc2.d/S30network*, which is invoked from */etc/rc2* during startup. The script understands two arguments: **start** and **stop**. It can be run manually for

testing and troubleshooting network-related problems without having to reboot the system.

Network Initialization Process

During system initialization, the shell script */etc/init.d/network* is called. These are the actions performed by the script at startup:

1. Checks hostname and Internet address to determine if system should be configured as standalone or networked. Checks *sys_id* and *hosts* files. If the **network** configuration flag is off, the system is configured for *standalone operation*.
2. Determines names and addresses or primary and router interfaces for typical configurations.
3. Obtains any site-dependent information for interfaces from the *netif.options* file.
4. If system is not diskless, the shell script flushes all old routes.
5. Configures all interfaces, including loopback, using the *ifconfig* command.
6. If configured for IP packet filtering, the shell script starts the IP packet filtering daemon (*/usr/etc/ipfilterd*). The *ipfilterd* daemon must be started before gateway interface initialization.
7. Initializes gateway interface.
8. Initializes additional interfaces specified in the *netif.options* file.
9. If specified, initializes the Hypernet interface according to the *ifconfig-hy.options* file.
10. Initializes the loopback interface.
11. Using the *chkconfig* command, determines daemon configuration and reads relevant daemon configuration files (**.options*).
12. Sets default route for all IP multicast packets to the primary interface.
13. If NIS software is configured, defines and sets NIS domain name.
14. If NIS software is configured, starts appropriate NIS daemons.
15. If NFS software is configured, starts appropriate NFS daemons and mounts any NFS filesystems listed in the */etc/fstab*.

16. If configured on with *chkconfig*, it starts standard daemons (*inetd*, *timed*, *timeslave*, *rarpd*, *rwhod*, *snmpd*, and so on).

Network Shutdown Process

During system shutdown, */etc/init.d/network* stops the daemons and disables the network devices. These are the actions the script performs at system shutdown:

1. Kills all network services that may be associated with a shell (*rlogind, rexecd, rshd, ftpd, telnetd,* and so on).
2. Kills some network daemons immediately (*inetd, bootp, tftpd, snmpd,* and so on).
3. If NFS is running, unmounts remote filesystems.
4. Kills all remote daemons.
5. If NFS is running, unexports exported filesystems. See the *ONC3/NFS Administrator's Guide* and the *NIS Administration Guide* for complete information about the optional NFS software.
6. Kills daemons that must be kept alive until the last minute (*portmap, slip, ipfilterd*).
7. Gracefully takes the system off the FDDI ring, if it is on the ring.
8. Stops the *ypbind* process of NIS.

Network Management Tools

This section describes a number of standard and some optional networking tools at your disposal for day-to-day management of your network. Except as noted, the standard networking tools reside in the */usr/etc* directory. See the online reference pages for additional information.

ifconfig(1M) Configures the network interface devices. *ifconfig* is performed at boot time by the master network configuration script */etc/init.d/network*.

Interface configuration includes enabling and disabling the interface and any interface options that should be set when the interface is configured. Options include support for the Address Resolution Protocol (ARP), routing metrics, netmask, broadcast addresses, and so on. The *ifconfig* tool used with the station's interface name displays the current interface configuration.

netstat(1)	Displays various network-related data structures that are useful for monitoring and troubleshooting a network. It has many options to show information about all or specific interfaces (**-i** or **-I**), routing tables (**-r** and **-M**), socket information (**-a** or **-A**), queue information (**-iq**), network memory (**-m**), and protocols (**-p**).
arp(1M)	Displays and manipulates *arp* table entries located in cache. *arp* options include **-a** for all entries, **-d** to delete an entry, **-s** to publish an entry and act as a server for this entry, and **-f** to pull information from a specified file versus */dev/kmem. arp* is a good tool for troubleshooting physical address resolution on a network. *arp* does not display the local station's Ethernet address. To get a local station's Ethernet address, use the *netstat* command with the **-ia** options.
rpcinfo(1M)	Provides information about Remote Procedure Call (RPC) based programs on local and remote stations. This is an excellent tool for isolating network problems related to the RPC service. The information provided by *rpcinfo* includes a list of *rpc*-based applications (*portmapper*, NIS, *rstatd*, and so on), and the program number, version number, protocol (TCP/UDP), and associated port number. If you are running an RPC-based network application and cannot get a response from the remote station, use the *rpcinfo* tool to ensure that the remote station supports the desired application.
ping(1M)	Tests and measures general network performance. It is based on the Internet Control Message Protocol (ICMP) and sends an ECHO_REQUEST soliciting an ECHO_RESPONSE, thereby creating a two-way stream. It provides general information about packet loss and round-trip time. *ping* increases network load; this factor should be considered when testing a network with *ping*.
spray(1M)	Sends a one-way stream of packets to a station using remote procedure calls. It reports information about the transfer rate and the number of packets received by the remote station. It provides very limited information about general network performance.
rtquery(1M)	Sends a request to a designated station for information on the station's network routing tables (*routed* or *gated*). This tool is especially useful for troubleshooting routing problems.
traceroute(1M)	Tracks packets as they journey through the network. This tool is very useful for isolating network and router faults in a large heterogeneous network. It displays the names and addresses of all the intermediary

routers that support the Internet Protocol "time-to-live" (TTL) field. It also displays the amount of time the packet spends traveling to the router, on the router, and leaving the router. *traceroute* increases network load; this factor should be considered when testing a network with *traceroute*.

route(1M) Manipulates the network routing tables. Typically, the routing tables are handled automatically by the *routed* or *gated* daemon. However, *route* can be used to create, maintain, and delete static routing tables, to flush routing tables, and to show metric information about routes. To have static routes incorporated at startup, modify the file */etc/gateways* and */etc/config/routed.options*.

rup(1C) Displays status information, including uptime and load average, about remote stations using Sun RPC broadcasts. If no specific station is specified, it uses broadcasting and returns information about stations on the local network; broadcasting does not go through routers. This tool is useful for isolating physical problems with a station or the network.

ttcp(1) Used to test Transmission Control Protocol (TCP) and User Datagram Protocol (UDP) performance. This tool provides a more realistic measurement of performance than the standard tests (*spray*, *rup*, *ping*). It allows measurements to be taken at both the local and remote end of the transmission.

These network management tools are available as options for use on any Silicon Graphics system:

NetVisualyzer
A passive network management product. It offers a set of graphical traffic monitoring, diagnostics, planning, and performance analysis tools that provide network information and statistics for Ethernet or FDDI networks in a visually intuitive form. NetVisualyzer comprises six tools: NetLook, NetGraph, NetCPA, Analyzer, RouteQuery, and TraceRoute. NetVisualyzer allows you to view and monitor your network, collect network statistics and generate reports based on those statistics, and decode heterogeneous packets layer by layer.

SPECTRUM A UNIX based network management tool for large-scale, multi-LAN, multi-vendor networks. It provides you with graphical, real-time views of your network. These views represent network location layouts,

network topology maps, and device front panels. The information provided by these views allows you to intelligently manage, monitor, and configure your network.

IRIS NetWorker™
: An automatic backup and recovery service for networked stations that performs backups of client systems to a centralized server. Backup schedules are specified by the network administrator. Stations in the network may be heterogeneous.

Interpreting Network Statistics

The network management tools provide the network administrator with valuable information about the network. However, the presentation of these statistics can be overwhelming. This section illustrates how to use three of the most common management tools and how to interpret the network statistics generated by these tools.

Testing Network Connectivity With ping

The *ping* tool tests and measures general network performance. It tells you when there is a problem with your network. The most important piece of information provided by *ping* is the *percentage packet loss*. Ideally, you want to see 0% packet loss; however, anything under .01% is acceptable. This low packet-loss threshold is required because many network applications transmit large packets. If 0.1% of a packet is lost, the entire packet must be retransmitted. This can cause a network to be saturated by retransmissions.

The following example uses *ping* in its simplest form, but the information obtained is very useful. The *ping* tool is testing and measuring traffic between the local station and the station *testcase*. See the ping(1M) reference page for more details about the many *ping* options.

```
/usr/etc/ping testcase
PING testcase (192.55.43.4): 56 data bytes
64 bytes from 192.55.43.4: icmp_seq=0 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=1 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=2 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=3 ttl=255 time=0 ms
64 bytes from 192.55.43.4: icmp_seq=4 ttl=255 time=0 ms
----testcase PING Statistics----
```

```
5 packets transmitted, 5 packets received, 0% packet loss
round-trip (ms)  min/avg/max = 0/0/0
```

The percentage packet loss is highlighted in bold. Again, 0% packet loss is the goal. Anything over 0.1% should be investigated further.

Measuring Network Throughput With ttcp

The *ttcp* tool measures network throughput. It provides a realistic measurement of network performance between two stations because it allows measurements to be taken at both the local and remote ends of the transmission. As with all network management tools, the statistics must be interpreted with the network configuration and applications in mind. For example, the statistics generated from a *ttcp* probe between two stations with routers in between results in lower throughput than if the stations were located on the same network. On the same note, users running applications that transmit large data structures see slower throughput than users running applications that transmit smaller data structures.

In any case, on a relatively quiet network, you should expect to see throughput in the 700 KBps or greater range. Throughput of 500 KBps or less is questionable, and 400 KBps or less may indicate a definite network problem.

The following example illustrates the statistics you might see if you ran a simple *ttcp* test between the stations *sheridan* and *longstreet* (two workstations) on a clean network. See the ttcp(1) reference page for details about the many *ttcp* options.

On *sheridan*, give the command

```
ttcp -r -s
```

You see the following output:

```
ttcp-r: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp
ttcp-r: socket ttcp-r: accept from 192.102.108.4
ttcp-r: 16777216 bytes in 19.99 real seconds = 819.64 KB/sec +++
ttcp-r: 10288 I/O calls, msec/call = 1.99, calls/sec = 514.67
ttcp-r: 0.1user 3.4sys 0:19real 17%
```

On *longstreet*, enter the command

```
ttcp -t -s sheridan
```

You see the following output:

```
ttcp-t: buflen=8192, nbuf=2048, align=16384/0, port=5001 tcp  ->
sheridan
ttcp-t: socket
ttcp-t: connect
ttcp-t: 16777216 bytes in 19.98 real seconds = 820.02 KB/sec +++
ttcp-t: 2048 I/O calls, msec/call = 9.99, calls/sec = 102.50
ttcp-t: 0.0user 2.3sys 0:19real 12%
```

The throughput statistics are highlighted in bold and are in units of KBps. The throughput on the station *sheridan* is 819.64 KBps and the throughput on the station *longstreet* is 820.02 KBps. Both throughput values indicate good network performance between the stations.

Collecting Network Statistics With *netstat*

The *netstat* tool displays various network-related data structures that are useful for monitoring and troubleshooting a network. Detailed statistics about network collisions can be captured with the *netstat* tool. Collision rates above 5% could indicate a problem with the network. The problem could be physical (bad tap, transceiver, loose terminator, and so on) or there might be too much traffic on the network.

This example illustrates the statistics you might see on a station using *netstat*. See the netstat(1) reference page for details about the many *netstat* options:

```
netstat -i
Name Mtu  Network  Address  Ipkts   Ierrs  Opkts   Oerrs Coll
enp0 1500 b1-channels thumper 498690  937   1066135   3  4858
lo0  32880 loopback localhost 1678915 0     1678915   0   0
```

The collision rate is approximately 0.45%, well within the acceptable range.

Troubleshooting Poor Network Performance

A variety of factors can affect network performance, including hardware problems, network configuration, network applications, and packet size.

Hardware Problems

Hardware problems can cause a network to be slow or inoperable. These problems are usually in the form of packet loss or corruption. Both of these problems can cause increased network traffic to the point of unmanageable congestion. Items to check at the physical level are:

Controller board
: Even if the network media bandwidth is capable of handling the network traffic load, the individual station may not be able to handle the traffic. This is evidenced by a high degree of traffic on the network interface for no apparent reason. This traffic can be seen using the *gr_osview* tool (see the gr_osview(1) online reference page for options to see network traffic statistics). If traffic is unusually heavy on the interface, then there may be a problem with the controller, or the controller may be too slow to handle the volume of traffic. You may need a high-speed controller like the Efast card.

Transmitter and controller
: Ensure that the Signal Quality Error (SQE), also called *heartbeat*, is disabled on both the transmitter and controller. SQE can cause unnecessary network traffic between the local station and the transceiver. See the installation guides for your network controller and transceiver for instructions on disabling SQE. By default, all Silicon Graphics network controller boards are shipped with SQE disabled.

Physical problems with the media
: Cables, taps, and other hardware will periodically break or malfunction. A time domain reflectometer (TDR) is essential for troubleshooting Ethernet cable problems. A good analyzer is also strongly recommended to assist in isolating network physical problems. Silicon Graphics' NetVisualyzer product supplies a visual network analyzer ideal for locating physical problems with the media.

Network Configuration

The network configuration or topology can also adversely affect network performance. Check for the following conditions:

- Check network configuration to ensure that it is within the official guidelines for the media and topology. Pay special attention to the number and location of repeaters and bridges.
- Consider work group affiliations when determining which network is best suited for a particular station. Planning a network based on work group affiliations can reduce the amount of intranetwork traffic. Put stations that routinely share resources on the same net (NFS mounting, same NIS domain, electronic mail, financial databases).
- If connecting large subnets, use a dedicated router (station that performs routing function only) as the primary router. Ensure that there are at least two ways in and out of a network, because one of the routers will eventually fail. The router should also use appropriate filters to filter out unnecessary traffic.

Network Daemons

Some network daemons (*rwhod*, *rtnetd*, and so on) can have an undesirable effect on the network or network interface. For example, if a workstation is a multi-processor or is running real-time processes, the *rtnetd* daemon may be running on the station. This daemon is responsible for preempting incoming network packets to provide better response time for real-time processes. This is perfectly acceptable if the user is aware of the trade-offs between network processing and real-time processing. Some network daemons should be evaluated individually.

Do not load *rtnetd* software on routers or other network intensive stations (mail servers, NIS and DNS servers, and so on).

Packet Size

The maximum transfer unit (MTU) for data on the Ethernet is 1500 bytes. Network performance and efficiency increase with packet size up to the MTU for the medium. Packets that are larger than the media's MTU must be broken into smaller packets (fragmented) to fit within the medium's MTU. Applications and services must be configured to transmit MTU size packets.

Kernel Configuration

You can change several parameters to customize network behavior for local configurations. The parameters listed below are in the */var/sysgen/master.d/bsd* configuration file.For details on reconfiguring the kernel after changing this file, see IRIX Admin: System Configuration and Operation.

Kernel Tunable Options

There are four kernel parameters which directly affect the network performance: *tcp_sendspace*, *tcp_recvspace*, *udp_sendspace*, and *udp_recvgrams*.

These parameters determine the default amount of buffer space used by TCP (SOCK_STREAM) and UDP (SOCK_DGRAM) sockets. The *tcp_sendspace* and *tcp_recvspace* parameters define the initial buffer space allocated to a socket. The *udp_sendspace* parameter defines the default maximum size of UDP datagrams that can be sent. The *udp_recvgrams* parameter determines the number of maximally sized UDP datagrams that can be buffered in a UDP socket. The total receive buffer size in bytes for each UDP socket is the product of *udp_sendspace* and *udp_recvgrams*. A program can increase or decrease the send buffer and receive buffer sizes for a socket with the SO_SNDBUF and SO_RCVBUF options to the *setsockopt*(2) system call. Many older TCP implementations have problems with large TCP *sendspace/recvspace* values. This should be decreased from 60 to 24 in environments where older stations have problems communicating.

For 4.2BSD compatibility, the IRIX system limits its initial TCP sequence numbers to positive numbers.

PC Connectivity

Many industry-standard personal computers with TCP/IP implementations experience difficulty connecting to Silicon Graphics workstations and servers. This is because of the increased size of the *tcp_sendspace* and *tcp_recvspace* variables in the IRIX file */var/sysgen/master.d/bsd*.

To allow your personal computers to connect successfully, change the values of the above variables from the default (60 * 1024) to (24 * 1024) and reconfigure the kernel with the *lboot* command. For more information on reconfiguring these values, see IRIX Admin: System Configuration and Operation.