Chapter 3

Setting Up a Network

This chapter contains the following sections:

Configuring an IRIS System for a Network

The procedure in this section explains how to configure an IRIS station, with one interface, for an Ethernet network using its local */etc/hosts* file (without BIND or NIS). Configuring the station takes six steps:

1. Bring the station down
2. Attach the station to the network
3. Check the station's network configuration
4. Modify the */etc/hosts* database
5. Name the station
6. Test the connection

Each of these steps is explained in the sections that follow.

Attaching Your Station to an Ethernet Network

Attach your station to the network by connecting one end of the Attachment Unit Interface (AUI) cable to the Ethernet I/O port and the other end to the network transceiver or Medium Access Unit (MAU).

If your transceiver model has status lights, make sure that the power light on the transceiver comes on when you attach the AUI cable to your workstation and the transceiver. This light indicates that the Ethernet card or the integral Ethernet controller is alive. Your station must be powered on to activate the power light on the transceiver. You may also see another light, which indicates that your link to the network is activated.

If the power light on the transceiver is lit or if your transceiver or MAU has no lights, bring your station back up into multiuser mode now.

Checking Your Ethernet Connection

You can use the *ping* command to check your Ethernet connection. This command tests whether you can connect with another system on the Ethernet network. Perform the following steps:

1. Obtain the hostname of at least one reliable station on the local area network to which your system is connected. If possible, get the fully qualified hostname and the IP address. (For example, a hostname might be *hancock*, and the fully qualified hostname might be *hancock.corp.gen.com*, while the IP address might be *192.70.3.56*.) It is important that the station you select has a reliable Ethernet connection and that it is up and running.
2. Once you have obtained a hostname and IP address, give the command

   `ping -r` *hostname*

   You should see a series of records indicating the returned packets from the remote host. For example (using our example system):

   ```
   PING hancock (192.70.3.56): 56 data bytes
   ```

```
64 bytes from 192.70.3.56: icmp_seq=0 ttl=255 time=2 ms
64 bytes from 192.70.3.56: icmp_seq=1 ttl=255 time=2 ms
64 bytes from 192.70.3.56: icmp_seq=2 ttl=255 time=2 ms
64 bytes from 192.70.3.56: icmp_seq=3 ttl=255 time=2 ms
64 bytes from 192.70.3.56: icmp_seq=4 ttl=255 time=2 ms
```

3. Press **<Ctrl>-C** or your **Delete** key to stop the *ping* command. You see the tallied results of the *ping* command. For example:

```
----hancock PING Statistics----

5 packets transmitted, 5 packets received, 0% packet loss

round-trip min/avg/max = 2/2/2 ms
```

4. If your network connection is working, you should see results comparable to those above. Your *ping* results should show 0% packet loss and an equal number of packets transmitted and received. If some packets are being lost, the first thing you should check is the tightness and quality of the cable connections. Loose cables are frequently the cause of lost packets. The round-trip time factors are a function of the size and general load of your network, and not necessarily an indication of a problem with your Ethernet connection.

If your ping command is not successful, there are several things you should check. Perform these steps:

1. Try to ping the station by its IP address. For example, using our sample host *hancock*, use the command

   ping -r 192.70.3.56

2. Try to ping a different station on your local network.
3. Check the network configuration on your system with the *netstat -in* command. You should see information similar to this:

```
Name Mtu  Network  Address    Ipkts Ierrs Opkts Oerrs

ec0  1500 192.70.3 192.70.3.9 18    0     18    0
```

   The `ec0` entry indicates your primary Ethernet connection. The `Ipkts` and `Opkts` fields indicate the number of inbound and outbound packets the network interface has processed. The `Ierrs` and `Oerrs` fields indicate the number of errors in input and output, respectively.

For the purposes of this troubleshooting session, though, check that the portion of the IP address shown under the `Network` heading match the IP address of the hostname that you attempted to *ping*. If the network addresses are not the same, the station is on a different network and the *ping* likely failed for that reason. Find a system to *ping* that is on your immediate local network.

4. Check the */var/adm/SYSLOG* file for Ethernet error messages.
5. Check to ensure that other stations are operating normally on the local network.
6. Check to ensure that the correct software (*eoe2.sw.tcp*) package has been installed on your system.
7. Check the physical connections to the Ethernet cables and transceivers for tightness and connection. A good indicator is the status light display on your transceiver (if your transceiver has these lights). If your Ethernet hardware is loose or disconnected, the */var/adm/SYSLOG* file and your system console should both show messages such as

   ```
   ec0: no carrier: check Ethernet cable
   ```
8. If all connections are tight and you still receive errors, replace the pieces of the Ethernet connection outside your system (the cable and transceiver). Different transceivers require different wire connections, and sometimes the wrong cables are used.
9. If you receive a message indicating that another host has the same IP address, find out which host has the same address, and determine which host has set their address incorrectly. (It is more likely that the same address was accidentally assigned to a second system, or that the new system being tested incorrectly set the address.)

Checking the Network Software Configuration

When the station comes up, verify the station's software configuration. Log into the station as *root*.

Issue the *chkconfig*(1M) command to check that your station's standard networking software is correctly configured:

```
/etc/chkconfig
```

You see information similar to this:

```
named      off
network    on
rwhod      off
timed      on
timeslave  off
gated      off
mrouted    off
rtnetd     off
snmpd      on
routed     on
```

Note: Your output will vary from the output above depending upon installed software and configuration flag settings.

If you are familiar with the network-related daemons, you can customize your configuration flags to suit your network needs. If you are not familiar with the network-related daemons, set your network-related flags as shown above. In particular, make sure that the *network* variable is configured *on*.

Modifying the Hosts Database

Before you modify the hosts database, you should have a list of station names and valid Internet addresses of all stations in your network, as discussed in “Allocating IP Addresses” on page 27. If the network has routers (stations with multiple network interfaces), there must be a valid Internet address and name for each interface. See “Internet Protocol Addresses” on page 13 for a description of IP addresses.

The *hosts* file is the hostname database. It contains information regarding known stations, from the perspective of the local station. This example assumes that you are not using NIS or BIND. If you are using NIS, refer to the *NIS Administration Guide* for more information. If you are using BIND, refer to Chapter 6, “BIND Name Server,” for more information.

The */etc/hosts* database is an ASCII file that you can modify with any text editor. The file contains lines of text that specify a station’s address, its “official” name, and any aliases. The “official” name should be the fully qualified domain name. The address and name(s) are separated by blanks, tabs, or both. Comments begin with a pound sign (#) and continue to the end of the line.

An */etc/hosts* database is shown in this sample */etc/hosts* file:

```
# This is a comment
127.0.0.1   localhost
119.0.3.20  tuna.salad.com tuna  # tuna is an alias
119.0.3.21  chicken.salad.com  salad
119.0.3.22  walrus.salad.com  walrus
```

Each IRIS system must have a copy of */etc/hosts* that contains entries for **localhost** and all of its network interfaces. As shipped, the */etc/hosts* database contains two entries. The first entry is a name you can use to test the local network software:

```
127.0.0.1 localhost
```

When you reference **localhost**, the message is looped back internally; it is never transmitted across the network.

Caution: Many important programs depend on the **localhost** entry—*do NOT remove or modify it.* If the master copy of */etc/hosts* is not maintained on an IRIS station or if you are using BIND or NIS, make sure that the host database contains the **localhost** entry.

To enable the IRIS system to access the network, add an entry containing the newly assigned IP address and the name in */etc/sys_id.* The entry *must* contain the *sys_id* name, either as the official hostname or as an alias.

Using the example */etc/hosts* file above, the */etc/sys_id* file for the host walrus should contain either "walrus" or "walrus.salad.com".

If you change the IRIS system's name in */etc/sys_id*, make sure to update the entry in */etc/hosts;* otherwise the network software will not initialize properly. If the following message appears during station startup, then the */etc/hosts* and /*etc/sys_id* files are inconsistent and must be fixed:

```
*** Can't find hostname's Internet address in /etc/hosts
```

If your IRIS system is a gateway, each network interface must be assigned an Internet address and have an entry in */etc/hosts,* as described in "Setting Up a Router" on page 38.

It is important that each station have a consistent version of the *host* database. The proper method for maintaining the consistency depends on the size of your network and whether the network is connected to the Internet. You can use the *rcp* or *rdist* programs by means of a *cron* job to ensure that the *hosts* files stay in sync.

Edit the */etc/hosts* file and add the hostnames and Internet addresses for all stations on your network. Each station on the network must have all station names in the local

/etc/hosts file. If you have a large */etc/hosts* file, the easiest way to install it on a new system is to set up a minimal hosts file with entries for the new system and for another system that has an authoritative copy of the hosts file. This allows you to get the new system on the net and copy the more complete hosts file from the other system (using *rcp* or *ftp*). Another option is to transfer the hosts file on tape or disk.

Naming Your Station

Once you have determined your station's name, edit the */etc/sys_id* file to give your station its new identity.

1. Remove the default station name (IRIS) and replace it with the new station name (*setup1* for this example). You can use this command:

   `echo setup1 > /etc/sys_id`

2. For the change to take affect, reboot your station with this command:

   `reboot`

When your station comes back up, it should have the new station name as the login prompt.

Testing Your Network Connectivity

Two network management tools, *rup* and *ping*, provide quick information about network connectivity. *rup* indicates if there is a physical problem with the network, such as your station being unable to contact the other stations. Since *rup* uses broadcasts as a default, it does not go through routers. If your station can see the other stations on the network, use *ping* to test communication abilities. *ping* uses the Internet Control Message Protocol (ICMP), which requests an echo from the designated station. It lets you know if you can transmit and receive packets to and from specific stations.

1. Issue the *rup* command to determine if your station can contact the other stations on the network:

   `/usr/bin/rup`

   You should get output on each of the stations on your network. The other stations on your network must be up for your station to get a user-friendly response. If the other stations are powered on and attached to the network but not up in user mode, the information comes back in hexadecimal.

2. Issue the *ping* command to see if your station can communicate with the other stations on the network:

   /usr/etc/ping *station_name*

   Let the output run a few seconds, then use **<Ctrl-C>** to break it. Pay particular attention to the ping statistics. *ping* gives you the number of packets transmitted, number of packets received, percentage packet loss, and round trip time (minimum, maximum, and average). These are all good indicators as to the general condition of your network. Obviously, you want 0% packet loss and a fast round-trip time.

Setting Up a Router

A router is a station with multiple network connections that forwards packets between networks. This section provides the configuration procedure for a router with two interfaces and a router with more than two interfaces. A station can have multiple interfaces and not act as a router. The procedure for turning forwarding off on a station with multiple interfaces is also provided in this section.

Configuring a Router With Two Interfaces

The */etc/init.d/network* script is designed to automatically detect and configure a router with two interfaces *if* the default naming scheme for the interfaces is used. By default, the Internet addresses of the primary and secondary interfaces are derived from the */etc/sys_id* file. The primary interface uses the name in the *sys_id* file. The secondary interface prefixes *gate-* to the name specified in the *sys_id* file.

To set up a router with two interfaces using the default naming scheme, follow this procedure:

1. Log in as **root**.
2. Assign valid Internet names and addresses to both interfaces in the */etc/hosts* file. For example, the */etc/hosts* file entries for the primary and secondary interfaces on the station *biway* might look like this:

   ```
   192.26.75.2     biway
   192.26.80.3     gate-biway
   ```

3. Ensure that the router has the appropriate name in its */etc/sys_id* file. Following this example, the */etc/sys_id* file should look like this:

```
biway
```

4. Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems prompt you for permission, as in the following example. Others simply return a shell prompt. In either case, enter the *reboot* command when the kernel has been reconfigured:

   ```
   /etc/autoconfig
   ```

   ```
   Automatically reconfigure the operating system? (y/n)y
   ```

   ```
   reboot
   ```

Note: If you do not want to use the standard router naming scheme, you must modify the */etc/config/netif.options* file. "Modifying the Network Interface Configuration" on page 46 details the procedure for changing an interface name.

Configuring a Router With More Than Two Interfaces

If the router contains more than two interfaces, you must modify the */etc/config/netif.options* file in addition to the */etc/hosts* and */etc/sys_id* files. In the *netif.options* file, you must define the interface type (enp1, ipg0, and so on). By default, the names for the third and fourth interfaces are *gate2-$HOSTNAME* and *gate3-$HOSTNAME*, where $HOSTNAME is the value returned when you issue the *hostname* command. If you want to modify the interface names, see "Modifying the Network Interface Configuration" on page 46 for the detailed procedure.

To set up a router with more than two interfaces and using the default naming scheme, follow this procedure:

1. Log in as **root**.
2. Assign valid Internet names and addresses to all interfaces in the */etc/hosts* file. For example, the */etc/hosts* file entries for the router *freeway*, with four interfaces, would look like this:

   ```
   192.26.30.1     freeway
   192.26.32.4     gate-freeway
   192.26.41.5     gate2-freeway
   192.26.59.6     gate3-freeway
   ```

3. Ensure that the router has the appropriate name in its */etc/sys_id* file. Following this example, the */etc/sys_id* file should look like this:

   ```
   freeway
   ```

4. Modify the *netif.options* file to define your interface types. For this example, the third and fourth interfaces are FDDI (ipg*). Change the *if3name* and *if4name* variables from

   ```
   if3name=
   if4name=
   ```

   to

   ```
   if3name=ipg0
   if4name=ipg1
   ```

5. Save your changes to the *netif.options* file.

6. Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems prompt you for permission, as in the following example. Others simply return a shell prompt. In either case, enter the *reboot* command when the kernel has been reconfigured:

   /etc/autoconfig

   ```
   Automatically reconfigure the operating system? (y/n)y
   ```

   reboot

Configuring Routing Behavior

By default, when a station has two or more network interfaces, it automatically forwards (routes) packets between the two interfaces. If you do not want the station to serve as a router on its network, disable its route advertising.

1. Modify the */etc/config/routed.options* file so the router will not supply routing information about general network routes (**–q**) or local interface routes (**–h**). Enter:

   echo –qh > /etc/config/routed.options

2. Using the */etc/init.d/network* script, turn the network off momentarily, then start it again. Enter:

   /etc/init.d/network stop

   /etc/init.d/network start

3. When the network comes up, verify that it is not forwarding packets with the *netstat* tool. Enter:

   netstat –s –p ip |grep forward

To change forwarding dynamics, create or edit the */etc/config/routed.options* file to contain the desired options. Some of the behaviors that can be altered by means of the */etc/config/routed.options* file are listed below:

- Suppress or force publicizing of routing information
- Enable tracing of all received packets
- Filter packets destined for specific networks

See the routed(1M) online reference page for more details.

Turning On Multicast Routing

If you're using Silicon Graphics workstations as routers, you can turn on multicast routing by doing the following:

1. Identify the routers on each network that need to support multicasting. Make sure that the routers you select are running IRIX Version 5.2 or later.
2. If you haven't already done so, install on each router the *eoe2.sw.ipgate* subsystem from your IRIX Version 5.2 distribution source. Run *autoconfig(1M)* if necessary.
3. As **root**, enter the command

   ```
   chkconfig mrouted on
   ```
4. Restart the system with the *reboot* command.

Understanding Where Multicast Packets are Forwarded

Figure 3-1 shows an example network with three routers.

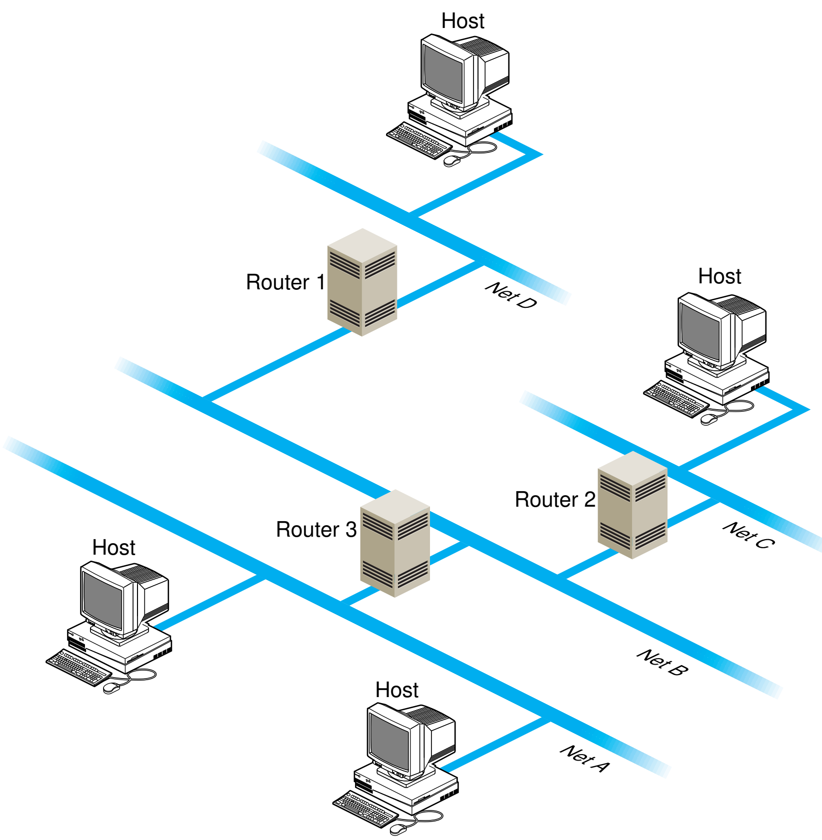

Figure 3-1 A Network With Multicast Routers

- If people on networks A, C, and D are listening for packets, as shown in the figure, all four networks receive the multicast packets.
- If people on networks A and C are listening for packets, networks A, B, and C receive the multicast packets.
- If three or more people on network A are listening for packets, networks A and B receive the multicast packets. The multicast routing protocol prevents packets from being sent to “leafs” of the network, such as networks C and D; however, it doesn’t prevent packets from being sent to interior networks, such as network B.

Setting Up Tunnels to Support Multicast Packets

If your routers do not support multicast routing, you can support multicast packets by creating *tunnels* between the networks. Any Silicon Graphics workstation running IRIX Version 5.2 or later can be configured as the endpoint of a tunnel. With tunneling, the multicast packets are encapsulated inside unicast packets, and then are sent to the other end of the tunnel. They are converted back into multicast packets when they are received.

Figure 3-2 shows an example of a setup with a tunnel between networks A and C.

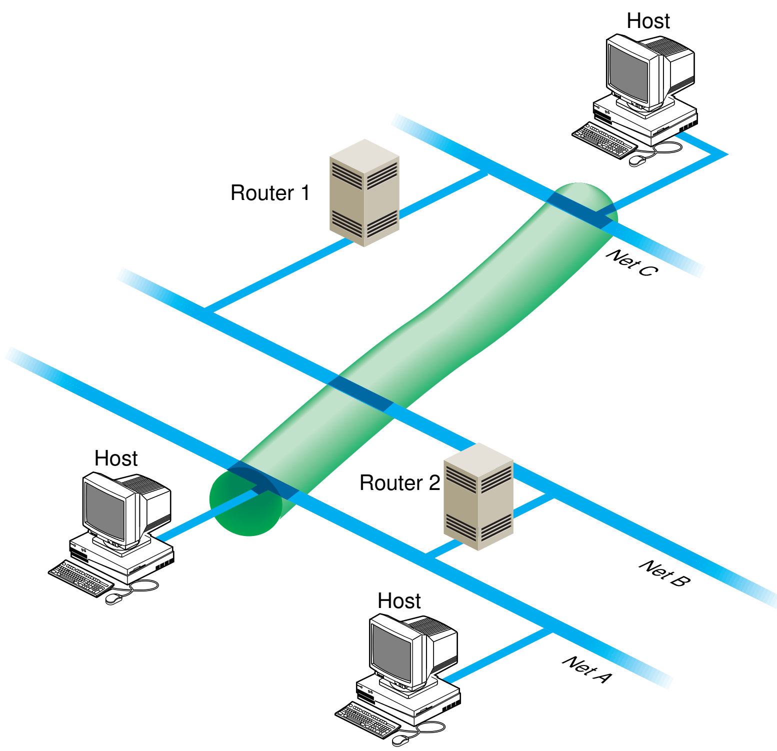

Figure 3-2 A Tunnel Between Networks A and C

To create the tunnel, edit the file */etc/mrouted.conf*. Step-by-step instructions follow:

1. Select the systems on each network that you will use for the sending and receiving end of the tunnel.

   Choose a system that is running IRIX Version 5.2 or later, is fast, and is not used extensively. The audio and video data may be intermittent if the system you select is slow or overloaded.

2. If you haven't already done so, install the *eoe2.sw.ipgate* subsystem from your IRIX distribution source.

3. As *root*, edit the file */etc/mrouted.conf* on the sending and receiving end of the tunnel. Note that these endpoints can be separated by many routers; you can use any machine on the network that is running IRIX Version 5.2 or later.

   Add the following line for each network to which you want to establish a tunnel.

   tunnel *<local IP address> <remote IP address>*

   In the above example, the system on network D would have the following entry:

   tunnel 192.26.58.1 192.48.170.2

   The system on network A would have the following entry:

   tunnel 192.48.170.2 192.26.58.1

4. You can specify other optional settings for the tunnel. For details, see the mrouted(1M) reference page.

5. Restart the system.

Note: One copy of the multicast packets is sent for each tunnel entry in *mrouted.conf*. This results in extra network traffic. For example, suppose you have a workstation with one network interface and you set up tunnels to workstations on three other networks. The packets are replicated three times.

Updating /etc/rpc for NIS Users

The IRIX copy of the */etc/rpc* file contains additions that are essential for using certain multicast utilities if you're running the Network Information Service (NIS). If the NIS master is not running IRIX Version 5.2 or later, or is not a Silicon Graphics workstation, verify that the */etc/rpc* file on the NIS master includes these additions:

```
sgi_iphone 391010
sgi_videod 391011
```

Subnetting a Network

Implementing the subnet address scheme is a very simple procedure. The following steps are necessary to implement subnetting:

1. Set the netmask.
2. Reboot the station.

Note: If you have not done so already, read Chapter 2, "Planning a Network." It contains information about partitioning Internet addresses for subnetting.

Setting the Netmask

The **netmask** option of the *ifconfig* command is used to interpret and define the network portion (subnets included) of the Internet address. A *network mask* defines the portion of the Internet address that is to be considered the network part. This mask normally contains the bits corresponding to the standard network part as well as the portion of the host part that has been assigned to subnetworks. If no network mask is specified when the address is set, it will be set according to the class of the network.

To configure a station's primary interface to recognize a subnetted Class B address that is using 8 bits of the host ID for defining the subnets, create or modify the */etc/config/ifconfig-1.options* file and insert the following line:

```
netmask 0xffffff00
```

This netmask value indicates that for the primary interface, the upper 24 bits of the Internet address represent the network portion of the address, and the remaining 8 bits represent the host ID. The nonsubnetted netmask for a Class B address (in hexadecimal) would be 0xffff0000. The netmask value can be set using hexadecimal, dot-notation Internet address, or pseudo-network name formats. This entry must be made on all stations on the subnet.

Note: For stations with multiple interfaces, the network mask should be set for each interface in the appropriate *ifconfig-*.options* file.

Rebooting the Station

When the **netmask** value is set, the station must be reconfigured and rebooted to incorporate the new network address into the stations routing tables. Always reboot routers before regular stations, because they provide routing information to other stations and networks.

Reconfigure the kernel and reboot the station to initialize your changes and interfaces. Some systems prompt for permission before reconfiguring the kernel, while others simply return a shell prompt. In either case, enter the *reboot* command explicitly:

```
/etc/autoconfig
Automatically reconfigure the operating system? (y/n)y
reboot
```

Modifying the Network Interface Configuration

You do not always need to modify (configure) a station's network interface; in most situations, the default configuration suits the site's needs. Modifying the network interface configuration requires that you modify the */etc/config/netif.options* file. You modify this file if

- the station has more than two interfaces
- you don't like or use the default naming conventions
- the default order is incorrect

There are two configurable variables in the *netif.options* file: **interface name** and **interface address**. The interface name variable designates the order (first, second, third, or fourth) and type of interface involved. The interface address variable assigns a valid Internet address to each interface. There must be a valid Internet address for each interface in the */etc/hosts* file. Table 3-1 summarizes the interface name and interface address variables.

Table 3-1 Variables for the netif.options File

Variable Name	Variable	Examples
Interface Name	if*x**name*=	if1name=enp0
	where:	if2name=ipg0
	x = 1, 2, 3, or 4	if3name=enp1
	name=ec0, et0, enp0, enp1, fxp1, ipg0, ipg1, etc.	if4name=enp2
Interface Address	if*x**address*=	if1address=$HOSTNAME
	where:	if2address=fddi-$HOSTNAME
	x = 1, 2, 3, or 4	if3address=gate-goofy
	address=$HOSTNAME, station name, or Internet address	if4address=192.30.28.2

You can modify either or both variables. Instructions for modifying both variables are provided below.

Modifying the Interface Name

When a station has more than two network interfaces, you must modify the name entries in the */etc/config/netif.options* file to assign the interface order. Default order is assigned only to the first two interfaces. Additionally, if you want to change the order (first, second, and so on) or interface type assigned to a network interface, you must modify the */etc/config/netif.options* file. This example makes the first FDDI interface secondary.

1. Using the *netstat* command, verify the network interface's name:

   `/usr/etc/netstat –ina`

2. Using vi or any editor, open the *netif.options* file for editing:

   `vi /etc/config/netif.options`

3. Locate and modify the appropriate interface name variable. For this example, search for the secondary interface name variable (*if2name*) and change it from the default configuration to a configuration that supports the first FDDI interface as secondary:

Change from this:

```
: if2name =
```

to this:

```
if2name=ipg0
```

Caution: Note that all *default* variables (primary and secondary) start with a leading colon (:). You must remove it *and* enter the interface type to change the default interface name.

4. Save and exit the file.

If you have no other changes, autoconfigure and reboot the station. Otherwise, repeat this procedure for each interface name change.

Note: When you alter the order of one network interface, the other interfaces in the station remain in their default relative ordering. For example, on a three-interface station (a=first, b=second, and c=third), if you were to make the default second the first (b=first), the remaining interfaces would be configured a=second and c=third.

Modifying the Interface Address

To change the default interface address, modify the */etc/config/netif.options* file. All interface names require valid Internet addresses as found in the */etc/hosts* file. The $HOSTNAME variable pulls the station name from the */etc/sys_id* file. This example changes the second, third, and fourth interface addresses as follows:

- second interface address: fddi-$HOSTNAME
- third interface address: gate-goofy
- fourth interface address: 192.30.28.26

Follow these instructions to modify your network interface address according to the above example:

1. Verify or assign a valid entry for each interface in the */etc/hosts* file. Write down the name and address for each interface.
2. Using *vi* or any editor, open the *netif.options* file for editing:

```
vi /etc/config/netif.options
```

3. Locate and modify the appropriate interface address variable. For this example, we want to modify the second, third, and fourth interface address variables. Find and modify each variable as follows:

   Change from this:

   ```
   : if2addr=gate-$HOSTNAME
   if3addr=gate2-$HOSTNAME
   if4addr=gate3-$HOSTNAME
   ```

   to this:

   ```
   if2addr=fddi-$HOSTNAME
   if3addr=gate-goofy
   if4addr=192.30.28.26
   ```

   Caution: Note that all *default* variables (primary and secondary) start with a leading colon (:). You must remove it *and* enter the interface address to change the default interface address.

4. Save and exit the file.

Repeat this procedure for each interface address change. If you have no other changes, reconfigure and reboot the station.

Changing Network Parameters

Network parameters are those settings that determine how an interface processes or supplies certain network information. Modifying network parameters requires that you create and modify the appropriate */etc/config/ifconfig-#.options* file (where "#" can be 1, 2, 3, or 4, based on the interface name).

The default parameter settings are known to function well and suit most site needs. Changing the settings can cause a network to become dysfunctional. It is recommended that these parameters be changed only by experienced or trained network managers.

Modifying the ifconfig-#.options File

There are four nondefault network parameters that can be set in the *ifconfig-#.options* file:

- netmask
- broadcast address

- Address Resolution Protocol (ARP)
- route metric

These steps show how to set the nondefault network parameters:

1. Using the *netstat* command, determine the order assigned to the network interface at configuration time:

   `/usr/etc/netstat -i`

2. Using vi or any editor, open or create the file */etc/config/ifconfig-#.options*, where the pound sign (#) represents the network interface's name. For example, to configure a primary interface, open or create the file */etc/config/ifconfig-1.options*.

3. To change the network mask value for a network interface, enter a line with the word `netmask` followed by a space and either the 32-bit hexadecimal value, Internet address dot-notation, or the pseudo-network name:

   ```
   netmask 0xffffff00
   ```

   See "Turning On Multicast Routing" on page 41 for more details regarding netmasks.

4. To change the broadcast address for a network interface, enter a line with the word `broadcast` followed by a space and the dotted decimal IP broadcast address:

   ```
   broadcast 189.92.6.0
   ```

   To enable or disable the Address Resolution Protocol (ARP), enter a line with *arp* (to enable) or *-arp* (to disable):

   ```
   arp
   ```

   The ARP tables are used by some of the network management tools (*netstat*, *ifconfig*, and so on) and provide the administrator with invaluable network information.

5. To change the routing metric count for a network interface, enter a line with the word `metric` followed by a space and the count:

   ```
   metric 7
   ```

   The default metric count, also called *hops*, is 0. The *routed* daemon monitors the number of hops required to route data from one network to another. If you want to reduce network traffic on a particular route, increase the metric count on the appropriate router.

The interface configuration file for the secondary interface might look like the following:

cat /etc/config/ifconfig-2.options

```
netmask 255.255.255.0
broadcast 129.38.50.0
-arp
metric 4
```

The interface configuration file above indicates that the Class B network is subnetted using 8 bits of the host ID for the subnet. It uses 0 as the broadcast address instead of the default 1. ARP has been disabled and the metric (hop) count has been set to 4 to discourage router traffic.

Dynamic Host Configuration With Proclaim

This section describes the IRIX *proclaim* facility, which is based on the dynamic host configuration protocol (DHCP) described in RFC 1541. Proclaim and DHCP allow you to allocate hostnames and network addresses automatically.

DHCP permits site administrators to set up one or more server systems that dynamically distribute network IP addresses and site configuration parameters to new or requesting client systems. In this way, a site with only a few available addresses can serve a large number of hosts that connect to the network only occasionally, or a large site can manage the permanent assignment of addresses with a minimum of administrative attention.

The *proclaim* facility consists of a daemon that runs on a master server, optional relay daemons for servers on subnets, GUI configuration programs, and a client application that communicates with the DHCP servers. All these programs are bundled with the IRIX operating system.

The sections below provide information about configuring *proclaim* servers and clients. If you are not comfortable with the basic concepts of networking, IP addresses, and netmasks, please read portions of the *IRIX Advanced Site and Server Administration Guide* before you continue reading this section.

Configuring the DHCP Server

The *dhcp_bootp* program is a server that communicates with DHCP and *proclaim* clients to provide host configuration information, including an IP address at minimum. If your site uses DNS to maintain the hosts map, you need some external mechanism (such as a script) to update the DNS map from the DHCP server. If your site uses NIS to maintain the hosts and ethers maps, *dhcp_bootp* must be run on the same machine as the NIS master server.

The DHCP server generally uses configuration parameters based upon the subnet number of the originating client request. The configuration files are all placed in the directory */var/dhcp/config* and are named in the form *config*.<*netnumber*>. For example, the configuration file for clients on the 192.26.61 network should be named *config.192.26.61.0*. If the subnet configuration file does not exist, then the *config.Default* file applies instead. If the default configuration file is missing or unreadable, clients are configured to match the DHCP server itself.

To configure the DHCP server, you can run the *ProclaimServerMgr* graphical interface, as described by the ProclaimServerMgr(1M) reference page. If you prefer, you can configure server options using your favorite text editor, altering the keywords documented on the dhcp_bootp(1M) reference page.

Configuring the DHCP Relay Agent

The *dhcp_relay* program is a subnet agent that communicates with the main DHCP server to provide DHCP and *proclaim* clients with host configuration information. If your site uses DNS to maintain the hosts map, you need some external mechanism (such as a script) to update the DNS map from the DHCP server. If your site uses NIS to maintain the hosts and ethers maps, your site's main DHCP server must be on the same machine as the NIS master. In any case, you should run *dhcp_relay* on all subnets besides the one with the main DHCP server.

The DHCP relay agent reads the configuration file */var/dhcp/config/dhcp_relay.servers* to determine the location of the main DHCP server. This configuration file should contain the IP address and hostname of the DHCP server, on two separate lines.

To configure the relay agent, you can run the *ProclaimRelayMgr* graphical interface, as described by the ProclaimRelayMgr(1M) reference page. If you prefer, you can configure relay agent options using your favorite text editor, following the steps documented on the dhcp_relay(1M) reference page.

The Proclaim Client

The *proclaim* client communicates with a DHCP server to obtain host configuration parameters, including an IP address and IP address lease at minimum. You can use *proclaim* to set up and configure new systems automatically, and also to move systems from one network to another without administrative intervention. The *proclaim* client can also verify configurations at reboot time.

For more information about setting up *proclaim* on client systems, see the proclaim(1M) reference page.

Limitations and Restrictions

The following restrictions and limitations are present in this release:

- The DHCP server must be the NIS master if your site uses NIS for hostname, IP address, or network hardware address validation and mapping. Note that NIS is an optional software product, and not all systems and networks use it.
- If your site uses DNS to maintain the hosts map, you need an external mechanism (manual or automated) to update the DNS map from DHCP. This limitation should be lifted in the next release.
- The server cannot respond simultaneously to both DHCP and *bootp* clients requests, although it can act as the *bootp* server for normal *bootp* requests.

Creating a Local Network Script

To start and terminate locally developed network daemons, or to publish ARP entries and set routes, create the separate shell script */etc/init.d/network.local.* Make symbolic links in */etc/rc0.d* and */etc/rc2.d* to this file to have it called during station startup and shutdown:

```
ln -s /etc/init.d/network.local /etc/rc0.d/K39network
ln -s /etc/init.d/network.local /etc/rc2.d/S31network
```

See */etc/init.d/network* for the basic format of the script. Also refer to *network*(1M), *rc2*(1M), and *rc0*(1M) for more information.

Turning On Remote Access Logging

Several network daemons have an option that lets you log remote accesses to the station log file */var/adm/SYSLOG* by using *syslogd*. Sites connected to the Internet should use this feature. To enable logging for *ftpd*, *tftpd*, and *rshd*, edit */etc/inetd.conf* and add "-l" after the right-most instance of *ftpd* and *tftpd*. Add "-L" after the right-most instance of *rshd.* For additional *ftp* logging, add "-ll" to the *ftpd* entry. Signal *inetd* to reread its file after you have added your changes:

```
/etc/killall -HUP inetd
```

Remote logins by means of *rlogin*, *telnet*, and the 4DDN *sethost* programs can be logged by *login*. Edit */etc/default/login* and add the keywords "SYSLOG=ALL" or "SYSLOG=FAIL" to it. For example, this command in the *login* file logs successful and failed local and remote login attempts to *syslogd*:

```
syslog=all
```

See the login(1) reference page for details.

Setting Up Network-Wide Services

In addition to the required software that you must set up on each system on the network, you may want to designate certain systems to provide network-wide services. For example, you can set up a system as an InSight server, so that users can access a complete set of InSight books without installing them all on their local disks. The following sections discuss setting up an anonymous *ftp* server, and setting up an InSight server.

How to Set Up a Proper Anonymous FTP Account

An anonymous FTP account is a way for you to make information on your system available to anybody, while still restricting access to your system. An anonymous FTP account lets anybody log in to your system as user "anonymous," or "ftp." Any such login causes the system to *chroot* to the FTP home directory (*~ftp*). This effectively limits the anonymous FTP user from accessing any part of your directory structure that is not a subdirectory of *~ftp* (see chroot(1M)).

The following procedure is designed to help you set up a network-accessible anonymous FTP account. As usual, understanding the various steps and continually monitoring how the account is used are necessary to protect your system security.

1. Create the anonymous FTP user entry in */etc/passwd*. The user name should be "ftp." Put an asterisk (*) in the password field, and assign user and group IDs, a home directory, and a login shell. The following is an example of a typical entry in */etc/passwd* for an anonymous FTP account:

   ```
   ftp:*:997:999:anonymous FTP account:/usr/people/ftp:/dev/null
   ```

   The login shell */dev/null* is recommended but not required, and the home directory can be anywhere, with reservations as explained in the next step.

2. Create an anonymous FTP directory. This may be wherever you like but, especially if you are going to allow writes to it, it should probably be on a separate partition from / or *usr*. That way, if the partition fills up, it will not disable basic system operations.

   In this example procedure, */usr/people/ftp* is the name of the anonymous FTP directory. First, make the directory:

   ```
   # mkdir /usr/people/ftp
   ```

   and then, if it is a separate disk or disk partition, you can mount the device on it (see mount(1M)). The anonymous FTP home directory you make must be the same one you specify in the */etc/passwd* file.

3. Set global read and access permission for the anonymous FTP directory, and change the owner to "ftp" and the group to "other":

   ```
   # chmod 555 /usr/people/ftp
   # chown ftp.other /usr/people/ftp
   ```

4. Change directory to the *ftp* home directory and create the subdirectories used for FTP access:

   ```
   # cd /usr/people/ftp
   # mkdir bin etc pub private
   ```

   In addition to the standard *bin*, *etc*, and *pub* directories, you may wish to make a *private* directory for private transmissions, as explained below.

5. For the *bin* and *etc* directories, set the owner to "root," group to "sys," and global read and access permissions:

   ```
   # chmod 555 bin etc
   # chown root.sys bin etc
   ```

6. For the *pub* directory, set the owner to "ftp," the group to "other," and global read, write, and access permission:

   `# `**`chown ftp.other pub`**

   `# `**`chmod 777 pub`**

   Caution: By allowing write permission, you make it possible for anonymous FTP users to fill the disk partition.

7. If you created a *private* directory, set the permissions to allow anybody to write to it but not to read its contents:

   `# `**`chown ftp.guest private`**

   `# `**`chown 773 private`**

   Anybody logging in can now place or retrieve files in the *private* directory, but they must be told the name of the file beforehand, because they cannot list the directory contents.

   Caution: By allowing write permission, you make it possible for anonymous FTP users to fill the disk partition.

8. Copy the *ls* command from */bin* to *~ftp/bin*:

   `# `**`cp bin/ls bin`**

9. Copy */etc/passwd* and */etc/group* to *~ftp/etc* and edit the files to an acceptable minimum:

   `# `**`cp /etc/passwd etc`**

   `# `**`cp /etc/group etc`**

   A good choice for the contents of *passwd* might be

   ```
   root:*:0:0:super-user:/:/dev/null
   bin:*:2:2:system tools owner:/bin:/dev/null
   sys:*:4:0:system activity owner:/usr/adm:/dev/null
   ftp:*:997:999:anonymous FTP account:/usr/people/ftp:/dev/null
   ```

   A good choice for the contents of *group* might be

   ```
   other::995:
   guest:*:998:
   ftp:*:999:
   ```

10. Set restrictive permissions on *~ftp/etc/passwd* and *~ftp/etc/group*:

    `# `**`chmod 444 etc/*`**

11. Add appropriate device and library files for anonymous FTP as follows:

```
# mkdir dev
# /sbin/mknod /usr/people/ftp/dev/zero c 37 0
# mkdir lib
# cp /lib/libc.so.1 lib
# cp /lib/rld lib
```

12. Add the following entry to the file */etc/aliases* to cause mail sent to the user ftp to go to the postmaster:

    ```
    ftp: postmaster
    ```

    Run the command *newaliases* to make this take effect. (This assumes you have an alias of postmaster in */etc/aliases*. See aliases(4) and newaliases(1M).)

13. Enable FTP logging as described in “Limiting inetd Services” in Chapter 5 in IRIX Admin: Backup, Security, and Accounting . Note that if you use one **-l** argument with *ftpd*, you record only successful and unsuccessful FTP login attempts. If you use two “l”s, you also record actions on files and directories performed during ftp login sessions, and three “l”s cause the report to include the number of bytes transferred in *get* and *put* operations.

    For example, the following entry in */etc/inetd.conf* means all logging information but the byte count is sent to */var/adm/SYSLOG*:

    ```
    ftp     stream  tcp     nowait  root    /usr/etc/ftpd   ftpd -ll
    ```

14. Once you have edited */etc/inetd.conf*, restart *inetd* with the following command:

    ```
    # /etc/killall -HUP inetd
    ```

    Note: Although the FTP logging records in */var/adm/SYSLOG* now show any passwords entered by users logging in, no password checking is done for anonymous FTP. The convention is for anonymous users to enter their e-mail addresses for passwords, but they could just as easily enter another user’s address or anything at all.

If you place any text in the file */etc/issue*, it is displayed when a user connects to your system, before the login prompt is displayed. You might want to include information here about the kind of services your FTP site offers, and whom to contact in case of problems. In addition, any text in the file *~/ftp/README* is displayed after the anonymous FTP user logs in.

Refer to crontab(1), syslogd(1M), and the file */var/spool/cron/crontabs/root* for information on changing the frequency or nature of system log file maintenance—you may, for example, want to increase the length of time you keep log files. To help you keep track of the demands made on your public FTP server, see Chapter 6 of IRIX Admin: Backup, Security, and Accounting for information on auditing system resource usage, and Chapter 7 of IRIX Admin: Backup, Security, and Accounting for general system accounting information.

Setting Up an InSight File Server

The files and directories that make up the InSight system of online documentation take up a great deal of space. In your network, there is no reason for each system to maintain a separate copy of the InSight documents as long as all systems are at substantially the same software revision level. If the InSight software revision level is the same across several systems, you can designate one system to serve the others with the InSight files, thus freeing up disk space on the client systems.

Be sure to choose a server system that is not called upon to carry a heavy workload, and take your network performance into account before you begin. If your users use InSight a great deal and your network is already burdened, you may find that your users will not appreciate the decreased response time from both InSight and your network in general. Also, if a person is to use the InSight server as his or her workstation, that person must be prepared to accept a certain (possibly substantial) amount of disk, CPU, and network overhead as a result.

There are two convenient ways to set up an InSight server. These will be detailed in the following subsections. Both methods require that you have the NFS software option installed. If you do not understand the terms and concepts behind NFS, you should read the *ONC3/NFS Administrator's Guide* and the *NIS Administration Guide* before undertaking these projects. The second method described here also requires a dedicated CD-ROM drive to hold the InSight distribution media.

A Conventional InSight Server/Client System

To install the IRIS InSight Viewer and Document Library on a remote server and retrieve the information from a local client system, follow the steps below.

On the server system:

1. Log in as *root* (superuser).
2. Bring up *inst*(1M) and install the complete IRIS InSight product image (from your CD-ROM drive or distribution directory) with the commands

```
Inst> install insight insight_gloss *.books.*
Inst> go
```

The total size of the viewer and document library is a little less than 23 megabytes.

3. Export the */usr/share/Insight/library/SGI_bookshelves* directory using the System Manager or the *exportfs* command:

   ```
   exportfs -i /usr/share/Insight/library/SGI_bookshelves
   ```

   If the server does not have a graphical display, a warning is generated during the exit. This warning relates to updating the X server's font directory and can be ignored.

On the client system:

1. Log in as *root* (superuser).
2. Give the command

   ```
   versions remove *.books.*
   ```

   to remove the books on your bookshelves.
3. Run the *inst c*ommand on your client system and give the following commands to install the *insight.sw.client* subsystem (you may want to install the *insight.man.man* and the *insight.man.relnotes* subsystems as well):

   ```
   Inst> keep insight insight_gloss
   Inst> install insight.sw.client
   Inst> go
   ```

4. Mount the *SGI_bookshelves* directory from your server on your client system. In the example below, the name of the server machine is *capra*.

   ```
   mkdir /usr/share/Insight/library/SGI_bookshelves
   mount capra:/usr/share/Insight/library/SGI_bookshelves
   /usr/share/Insight/library/SGI_bookshelves
   ```

   Note that when you enter the *mount* command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

Note: If you have remote-mounted your InSight books, the Silicon Graphics *desktophelp* system will not operate correctly while the books are mounted. To view the *desktophelp*, unmount the books temporarily.

A CD-ROM InSight Server/Client System

With this method, you need not use up your disk space for the InSight files if you have a CD-ROM drive you can dedicate to InSight. You simply leave the CD with the InSight distribution in the drive and link the mounted distribution to the directory where InSight would have installed the files. The drawback of this system is that you must dedicate a CD-ROM drive to the purpose, but this method can be used with NFS to provide server access to your entire network.

Note: If you have a version of the IRIS InSight Document Library installed on your client disk but you want to mount the books from the CD-ROM, you must remove all the files in */usr/share/Insight/library* before creating the symbolic link described in step 2. Use the *versions remove* command to cleanly remove the books, then check the directory to be sure all files have been removed.

If you wish to use the IRIS InSight Viewer and Document Library from the CD-ROM and access the information from a local or remote system, follow the steps below.

On the server system with the CD-ROM drive:

1. Log in as *root* (superuser).
2. Insert the IRIS InSight CD into the CD-ROM drive. If the drive is not mounted, use the System Manager or the *mount*(1M) command to mount the drive. The most common mount point is */CDROM*. If the drive is mounted correctly, you should be able to change directories to */CDROM* and see all the files in the IRIS InSight CD. To see the files, use the command

   ```
   cd /CDROM/insight
   ```
3. As *root*, create a symbolic link from */usr/Insight/library/SGI_bookshelves* to the CD *insight* directory. You may need to create the directory */usr/Insight/library* if it doesn't exist. Use the commands:

   ```
   mkdir -p /usr/share/Insight/library
   ```

   ```
   ln -s /CDROM/insight/SGI_bookshelves /usr/share/Insight/library
   ```

   Note that when you enter the link command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

   At this point, the InSight bookshelves are mounted on your server and available for use on that system. To allow users on other systems on your network to use the bookshelves, proceed to step 4.

4. If you want to share the bookshelves with other users on your network, export the */CDROM* directory using the System Manager or the *exportfs* command:

   ```
   exportfs –i /CDROM
   ```

On each client system, perform the following steps:

1. Log in as *root* (superuser).
2. Run the *inst* command and give the following commands to install the *insight.sw.client* subsystem (you may want to install the *insight.man.man* and the *insight.man.relnotes* subsystems as well). You can get the installation software from the CD in the */CDROM/dist* directory on the server.

   ```
   Inst> keep *
   Inst> install insight.sw.client
   Inst> go
   ```

3. Mount the bookshelves from the server. In the example below, the name of the server machine is *capra*. Use the following command:

   ```
   mount capra:/CDROM/insight/SGI_bookshelves
   /usr/share/Insight/library/SGI_bookshelves
   ```

   Note that when you enter the *mount* command on your client system, the entire command line goes on a single line. The command is broken across two lines in this example due to formatting limitations.

Using Remote InSight

After the software has been installed, you should force your shell to remake its list of available programs and commands with the command

```
rehash
```

You can now invoke the IRIS InSight Viewer. The command to invoke the viewer is *iiv* (an acronym for IRIS InSight Viewer):

```
iiv
```

Troubleshooting Your Ethernet Connection

This section addresses some of the common Ethernet-related problems. See also "Troubleshooting System Configuration Using System Error Messages" in IRIX Admin: System Configuration and Operation.

Cable Problems

Unplugged or loose cables are the most common problem that causes Ethernet-related error messages. For example:

```
unix: <ethernet_device>: no carrier: check Ethernet cable
```

This message means that the carrier was not detected when attempting to transmit. One recommendation is

- Check to see if the Ethernet cable is unplugged from the back of the machine. For detailed instructions on connecting the cable, see your owner's guide. You do not need to *shut down* the system to connect or disconnect the cable. If you re-connected the cable, test the network connection.

After determining that the transceiver cable is plugged into the back of the machine and also into the transceiver box, look for other possible reasons for failure.

- Check the machine's transceiver.
- Check the transceiver cable, and try swapping it out with one that is working on another machine.
- Check for a problem with a 10baseT hub.
- If you try all the troubleshooting techniques and you still cannot access the network, contact your service provider; the network itself may be temporarily out of order.

For more information on this error, see the reference page for ethernet(7).

Late Collisions

Another problem related to cabling is that of late collisions. These result in errors like the following:

```
unix: <ethernet_device>: no carrier: check Ethernet cable
```

The controller tried to transmit a packet but received a late collision signal from another machine. This usually indicates a problem in the Ethernet cable layout. The most frequent causes of this problem are excessively long cables and loose cable connections.

To isolate the problem, first look for violations of the Ethernet cable limits. For 10BASE5 cable (thicknet) the maximum segment length is 500m; for 10BASE2 cable (thinnet) the maximum segment length is 200m. The length of a transceiver cable (the cable that goes between a machine's Ethernet port and its transceiver) should not exceed 50meters. If all cables are within the specified limits and all connectors are firmly seated, this error could indicate a hardware problem in an Ethernet controller or transceiver. Some recommendations are:

- Check other machines on the network for bad Ethernet controllers.
- Check other machines on the network for bad transceivers. I f you suspect a bad transceiver, you can try swapping in a different one that works on another system.
- Check to see if removing a certain machine from the network makes the problem change or disappear.

Packet Size

A bad Ethernet controller somewhere on the network can cause problems by sending packets that are too large or too small, causing errors like this:

```
unix: <ethernet_device>: packet too small (length = <packet size>)
unix: <ethernet_device>: packet too large (length = <packet size>)
```

The Ethernet controller received a packet that was smaller than the minimum packet size or larger than the maximum packet size allowed by Ethernet. This problem is caused by another machine with a bad Ethernet controller or transceiver. Some recommendations are

- Check other machines on the network for bad Ethernet controllers.
- Check other machines on the network for bad transceivers. I f you suspect a bad transceiver, you can try swapping in a different one that works on another system.
- Check to see if removing a certain machine from the network makes the problem change or disappear.

For more information on this error, see the reference page for ethernet(7).

Unable to Contact Server System

When your system cannot contact a network system, an error message similar to this is displayed:

```
portmapper not responding: giving up
```

This problem occurs in one of these situations:

- The system is not running.

  Physically go to the system or call the system's Primary User or Administrator to check to see if the system is powered up and running.
- The network is not running.

  To check, try to access another system on the network.
- The network administrator changed some information about the system or about the system's logical location on the network.

  Check whether this is the case, and get the appropriate information to fix the problem.
- The system has too many users or systems using its resources. It cannot provide services to you at this time.

  Contact the system's Primary User or Administrator to check on this.

Checking Additional Network Interfaces

If your system has more than one network interface (additional interfaces are usually fiber-optic [FDDI] links or SLIP connections or other Ethernet boards) you can easily perform the above checks on each network interface.

To check your other network interfaces, give the *netstat -in* command. You see output similar to the following:

```
Name Mtu   Network    Address    Ipkts Ierrs Opkts Oerrs Coll
ec0  1500  192.70.0   192.70.0.9  15    0     15    0     24
ec1  1500  192.70.2   192.70.2.5  15    0     15    0     24
```

```
sl0  1006  (pt-to-pt) 192.70.0.9  0     0     0     0     0
lo0  8304  loopback   localhost   8101  0     8101  0     0
```

The second Ethernet connection is to the network 192.70.2, a different LAN from the first Ethernet connection. The address of the local station on the second LAN is 192.70.2.5. To check that connection, use the *ping* command to test the connection to another station on that network.

There is also a SLIP link running in this example. The SLIP link extends the same LAN as `ec0` to another system in a different location. To test this link, find the hostname or IP address of the station at the other end of the SLIP link and use the *ping* command to test connectivity.

The `lo0` interface is the loopback network interface on the local host.