



# Internet Protocol Network Multipathing (Updated)

---

*Mark Garner, Enterprise Engineering*

*Sun BluePrints™ OnLine - November 2002*



<http://www.sun.com/blueprints>

**Sun Microsystems, Inc.**  
4150 Network Circle  
Santa Clara, CA 95045 U.S.A.  
650 960-1300

Part No.: 806-7230-12  
Revision 3.0 9/22/03  
Edition: November 2002

Copyright 2002 Sun Microsystems, Inc. 4150 Network Circle, Santa Clara, California 95045 U.S.A. All rights reserved.

Sun Microsystems, Inc. has intellectual property rights relating to technology embodied in the product that is described in this document. In particular, and without limitation, these intellectual property rights may include one or more of the U.S. patents listed at <http://www.sun.com/patents> and one or more additional patents or pending patent applications in the U.S. and in other countries.

This product or document is protected by copyright and distributed under licenses restricting its use, copying, distribution, and decompilation. No part of this product or document may be reproduced in any form by any means without prior written authorization of Sun and its licensors, if any. Third-party software, including font technology, is copyrighted and licensed from Sun suppliers.

Parts of the product may be derived from Berkeley BSD systems, licensed from the University of California. UNIX is a registered trademark in the United States and other countries, exclusively licensed through X/Open Company, Ltd.

Sun, Sun Microsystems, the Sun logo, Sun BluePrints, SunATM, OpenBoot, JumpStart, and Solaris are trademarks or registered trademarks of Sun Microsystems, Inc. in the United States and other countries.

The OPEN LOOK and Sun™ Graphical User Interface was developed by Sun Microsystems, Inc. for its users and licensees. Sun acknowledges the pioneering efforts of Xerox in researching and developing the concept of visual or graphical user interfaces for the computer industry. Sun holds a non-exclusive license from Xerox to the Xerox Graphical User Interface, which license also covers Sun's licensees who implement OPEN LOOK GUIs and otherwise comply with Sun's written license agreements.

U.S. Government Rights—Commercial use. Government users are subject to the Sun Microsystems, Inc. standard license agreement and applicable provisions of the FAR and its supplements.

DOCUMENTATION IS PROVIDED "AS IS" AND ALL EXPRESS OR IMPLIED CONDITIONS, REPRESENTATIONS AND WARRANTIES, INCLUDING ANY IMPLIED WARRANTY OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE OR NON-INFRINGEMENT, ARE DISCLAIMED, EXCEPT TO THE EXTENT THAT SUCH DISCLAIMERS ARE HELD TO BE LEGALLY INVALID.

Copyright 2002 Sun Microsystems, Inc., 4150 Network Circle, Santa Clara, Californie 95045 Etats-Unis. Tous droits réservés.

Sun Microsystems, Inc. a les droits de propriété intellectuels relatants à la technologie incorporée dans le produit qui est décrit dans ce document. En particulier, et sans la limitation, ces droits de propriété intellectuels peuvent inclure un ou plus des brevets américains énumérés à <http://www.sun.com/patents> et un ou les brevets plus supplémentaires ou les applications de brevet en attente dans les Etats-Unis et dans les autres pays.

Ce produit ou document est protégé par un copyright et distribué avec des licences qui en restreignent l'utilisation, la copie, la distribution, et la décompilation. Aucune partie de ce produit ou document ne peut être reproduite sous aucune forme, par quelque moyen que ce soit, sans l'autorisation préalable et écrite de Sun et de ses bailleurs de licence, s'il y en a. Le logiciel détenu par des tiers, et qui comprend la technologie relative aux polices de caractères, est protégé par un copyright et licencié par des fournisseurs de Sun.

Des parties de ce produit pourront être dérivées des systèmes Berkeley BSD licenciés par l'Université de Californie. UNIX est une marque enregistrée aux Etats-Unis et dans d'autres pays et licenciée exclusivement par X/Open Company Ltd.

Sun, Sun Microsystems, le logo Sun, Sun BluePrints, SunATM, OpenBoot, JumpStart, et Solaris sont des marques de fabrique ou des marques déposées, ou marques de service, de Sun Microsystems, Inc. aux Etats-Unis et dans d'autres pays.

L'interface d'utilisation graphique OPEN LOOK et Sun™ a été développée par Sun Microsystems, Inc. pour ses utilisateurs et licenciés. Sun reconnaît les efforts de pionniers de Xerox pour la recherche et le développement du concept des interfaces d'utilisation visuelle ou graphique pour l'industrie de l'informatique. Sun détient une licence non exclusive de Xerox sur l'interface d'utilisation graphique Xerox, cette licence couvrant également les licenciés de Sun qui mettent en place l'interface d'utilisation graphique OPEN LOOK et qui en outre se conforment aux licences écrites de Sun.

LA DOCUMENTATION EST FOURNIE "EN L'ÉTAT" ET TOUTES AUTRES CONDITIONS, DECLARATIONS ET GARANTIES EXPRESSES OU TACITES SONT FORMELLEMENT EXCLUES, DANS LA MESURE AUTORISEE PAR LA LOI APPLICABLE, Y COMPRIS NOTAMMENT TOUTE GARANTIE IMPLICITE RELATIVE A LA QUALITE MARCHANDE, A L'APTITUDE A UNE UTILISATION PARTICULIERE OU A L'ABSENCE DE CONTREFAÇON.



Please  
Recycle



Adobe PostScript

# Internet Protocol Network Multipathing

---

This article describes the features and configuration of Internet Protocol network multipathing (IPMP). IPMP can provide network adapter resilience and increased data throughput by using multiple network adapters connected to the same subnet.

The intended audience for this article is systems designers or administrators. This article is a concise overview of the Solaris™ 8 Operating Environment (Solaris OE) *IP Network Multipathing Administration Guide* documentation. The following elements are addressed:

- Design Considerations
- Configuration
- Basic Management

This article also discusses two essential considerations for IP network multipathing that are not covered in the administration guide. See sections, *Data Address Definition* within the *Design Considerations* section and the addendum, *The Problem With ping*.

This article covers the IPv4 network protocols—for information on IPv6, see the *IP Network Multipathing Administration Guide* section of the Solaris 8 OE documentation. Go to: <http://docs.sun.com/>

---

**Note:** The commands of IP network multipathing are not listed in the man pages of the Solaris OE version 8 update 2 (10/00), but are in later releases.

---

---

## Functional Overview

IP network multipathing was introduced in the Solaris 8 OE update 2 (10/00) software release. This feature enables a server to have multiple network ports connected to the same subnet.

Network adapters function in the following modes:

- Active-active
- Active-standby

IP network multipathing coupled with multiple network connections per subnet provide a server with one or both of the following advantages:

- Resilience from network adapter failure
- Increased data throughput for outbound traffic

To provide resilience, IP network multipathing detects the failure or repair of a network adapter, and switches the network address to or from an alternative adapter. If more than one network adapter is active, outbound packets are spread across adapters, thereby increasing data throughput.

---

## Design Considerations

IP network multipathing requires hardware and Solaris OE configuration. The following sections cover the design considerations for the hardware and the Solaris OE configuration.

IP network multipathing has the following requirements (discussed in more detail later in this section):

- Solaris OE version 8 update 2 (10/00) or later.
- Unique MAC addresses on each network interface.
- Multiple network adapter interfaces (of similar type) on each subnet for a resilient configuration.
- A network adapter group name.
- Test addresses on every network interface.
- Data addresses on every network interface.
- Network interfaces should run in active mode, not standby.

In addition, there is a problem with `ping` in both: Solaris 8 OE update 2 (10/00), and Solaris 8 OE update 4 (04/01). This problem is fixed in Solaris 8 OE update 5 (07/01). The addendum in this article, *The problem with ping*, describes the problem in detail and offers three possible workarounds.

The terms *test address* and *data address* are used throughout this article. The purpose of the test addresses is to detect failure and recovery of an interface only. The test addresses are tied to each interface for this purpose. Test addresses should not be

used for server-client communication. In addition, the data addresses migrate between interfaces in the event of an interface failure, and should be used exclusively for host-client communication.

## Solaris OE Version

In this article, all testing, observations, and examples use the Solaris 8 OE update 2 (10/00) version except where otherwise stated.

## Unique Ethernet MAC Addresses

The default configuration for Sun hardware is that all network interfaces (on a specific server) have the same Ethernet MAC address. If more than one network interface is to be connected to the same subnet, the default configuration must be changed to avoid a MAC address conflict. That is, each interface connected to the same subnet must have a different MAC address.

## Network Adapter Selection

A resilient network configuration requires that two or more network interfaces be connected to the same subnet.

---

**Note:** IP network multipathing can be configured with a single network interface; however, only failure detection will be operative.

---

To increase resilience (where the hardware configuration permits), network interfaces should be located on different I/O boards. Create a symmetrical configuration (pairing ports on I/O cards in a mirrored fashion) to avoid confusion. Therefore, ports with the same numbers will be connected to the same subnets and belong to the same IP network multipathing group.

---

**Note:** IP network multipathing does not work with dissimilar network interfaces—for example, Token Ring with Ethernet, or SunATM™ interface with Token Ring, etc.

---

## Network Adapter Group Name

Configuration of multiple network interfaces within Solaris OE is performed by grouping the interfaces into an IP network multipathing group. A name should be chosen for each group that describes the network function—for example, production, backup, administration, etc.

## Test Addresses Definition

Test addresses must be defined for each interface. The `in.mpathd` daemon uses these addresses in the detection of network interface failures and repairs. A test address must be a valid *routable* address. For the purposes of fault detection and isolation, include the name of the device in the name of the test address. This way, a test address on `qfe0` will have a “-qfe0” suffix as its name. These addresses should not be used for inbound connections, as they will not failover in the event of an interface failure. For this reason, do not advertise them in a naming service or by other means. It is, in fact, the use on the `-failover` flag during configuration that causes `in.mpathd` to choose these addresses as test addresses because they will not failover.

## Data Address Definition

The IP network multipathing group requires addresses be defined for data communication between server and client. It is these data addresses that migrate between interfaces during a failure.

We recommend that you define data addresses for every interface in an IPMP group to allow IPMP to load spread outbound connections between all interfaces in the group. This means that connections will emanate from different IP addresses on the server. In a few cases, secure shell (SSH) is one, this will require additional consideration, for example all the IP addresses will need to be registered for SSH.

The second reason for defining data on every interface was to prevent a potential problem on Solaris 8 OE systems. This problem is now fixed by applying patch 108528-15, or greater, available from <http://sunsolve.sun.com>. This problem does not occur with Solaris 9 OE. It occurs where data addresses are defined on only one interface in an IPMP group. Should the interface where the data addresses are defined fail during system boot, an alternative working interface is potentially never recognized—the data addresses do not failover. This situation does not occur if an alternative interface within IPMP group has a data address configured or if patch 108528-15, or greater, has been applied.

Therefore, it is recommended that you define data addresses for every interface in an IPMP group to take advantage of outbound load spreading. If this is not possible with Solaris 8 OE, or if the standby feature has to be used with Solaris 8 OE, you must apply patch 108528-15, or greater.

## Design Parameters

The parameters defined in TABLE 1 are used in the commands and examples throughout this article for configuration of IP network multipathing:

**TABLE 1** IP Network Multipathing Configuration Parameters

<b>Parameter</b>	<b>Value</b>
Network adapter interfaces (active or standby) (I/O slot, port number)	qfe0 (active) (slot 0, Port 0) qfe4 (active) (slot 1, Port 0)
Group name	Production
IP Address (name) interface	192.168.49.42 (camelot) qfe0 192.168.49.7 (camelot-1) qfe4
Test address (name)	qfe0 = 192.168.49.105 (camelot-qfe0) qfe4 = 192.168.49.106 (camelot-qfe4)
Netmask	255.255.255.0
Is the node to perform network routing?	No

---

# Configuration

The following sections describe the steps required to configure an IP network multipathing group with two active interfaces. These sections use the specifications defined in TABLE 1 and cover the following topics:

- Enabling unique Ethernet MAC addresses.
- Defining TCP/IP addresses.
- Disabling routing (if applicable).
- Configuring the network interfaces.

## Enabling Unique Ethernet MAC Addresses

To support the setting of a unique MAC address automatically, the adapter must have a MAC address stored in its `Fcode` PROM. Not all network adapters (particularly on-board adapters) have this feature. In this situation, the MAC addresses must be set manually.

The definitive way to determine if a MAC address will be set automatically is to try it and check the outcome. Follow the procedure below:

1. Set the EEPROM variable `local-mac-address?` to `true` as follows:

```
# eeprom local-mac-address?=true
```

---

**Note:** With the preceding command, the server must be rebooted for the change to take effect.

---

2. If working at the OpenBoot™ Prompt (OBP), enter the following command:

```
ok setenv local-mac-address? true
```

3. Boot the server and `plumb` each of the interfaces in the IP network multipathing group by issuing the following command for each interface:

```
# ifconfig <interface> plumb
```

- Execute the following command to determine the MAC address for each interface. The following codebox also includes an example of the output:

```
# ifconfig -a
hme0: flags=1000843<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index
2
    inet 0.0.0.0 netmask 0
    ether 8:0:20:f7:c3:f
hme1: flags=1000842<BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500 index
8
    inet 0.0.0.0 netmask 0
    ether 8:0:20:f7:c3:f
```

- The `ether` line shows the MAC address. If any interfaces are showing identical MAC addresses and are connected to the same subnet, then set the MAC addresses manually with the following command:

```
# ifconfig <interface> ether <MAC address>
```

It is suggested that the last three octets of the MAC address be made the same as the last three octets of the test address. This address convention helps make the MAC address unique within a subnet—for example, 192.168.49.106 – 08:00:20:a8:31:6a. Although the chance of a MAC address conflict is extremely small, it can be checked for by using the `snoop` command to search for the chosen MAC address, while using the `ping` command to verify connection to the broadcast address of the subnet.

---

**Note:** If a MAC address is to be set manually, insert the `ether` parameter when the interface is configured in the following sections.

---

Additionally, it is possible to determine whether the `Fcode` PROM of some Ethernet adapters has a local MAC address with the following command:

```
# prtconf -vp | grep local-mac-address
```

Therefore, if local-mac-addresses are displayed for all interfaces, except the on-board adapter, then the setting of local-mac-address provides unique addresses. Because this command has not been tested against all adapters, it is not a recommended method, however, if used diligently it may be helpful.

## Defining TCP/IP Addresses

The interface data addresses and IPMP test addresses should be added to `/etc/hosts` file. It is assumed that no existing network configuration information exists on the server. For example (using the example addresses defined in TABLE 1), the following lines should be included in `/etc/hosts` file:

```
#
#
# IP Network Multipathing Group - Production
#
192.168.49.42    camelot          # Data Address
192.168.49.7   camelot-1       # Additional Data Address
192.168.49.105 camelot-qfe0    # Test Address for qfe0
192.168.49.106 camelot-qfe4    # Test Address for qfe4
```

The netmask setting for the subnet should be set in the `/etc/netmasks` file. Using the example addresses defined in TABLE 1, the following line should be included in the `/etc/netmasks` file:

```
192.168.49.0      255.255.255.0
```

## Disabling Routing

If the node is not intended to perform network routing, enter the following command:

```
# touch /etc/notrouter
```

---

**Note:** The server must be rebooted for this change to take effect unless the IP driver parameter, `ip_forwarding` is set to zero using the `ndd /dev/ip` command.

---

## Configuring the Network Interfaces

After the network address information has been added to the `/etc/hosts` and `/etc/netmasks` files, the network interfaces can be configured. The following commands perform the configuration dynamically. In addition, the following paragraphs describe how to create a permanent configuration. In the following examples, the addresses defined in TABLE 1 are used.

Create the network interfaces:

```
# ifconfig qfe0 plumb
# ifconfig qfe4 plumb
```

Create an IP network multipathing group named *production*, which consists of network interfaces `qfe0` and `qfe4`:

```
# ifconfig qfe0 group production
# ifconfig qfe4 group production
```

After executing the commands above, the following `syslog` messages may be issued. The messages simply warn that failures cannot be detected, until test addresses are established on the interfaces.

```
May 21 14:14:15 camelot in.mpathd[430]: Failures cannot be detected on
qfe0 as no IFF_NOFAILOVER address is available
May 21 14:14:15 camelot in.mpathd[430]: Failures cannot be detected on
qfe4 as no IFF_NOFAILOVER address is available
```

Use the following commands to create an address on each interface for data transmission and add the `failover` flag. The `failover` flag allows the address to migrate if an interface failure is detected.

```
# ifconfig qfe0 camelot netmask + broadcast + failover up
# ifconfig qfe4 camelot-1 netmask + broadcast + failover up
```

The following commands configure a test address on each network interface; these addresses are used by `mpathd` to detect interface failures. Test addresses should not be used by host applications for data communication; hence, they should be marked with the `deprecated` flag. In addition, test addresses must not failover and should also be marked with the `-failover` flag. It is the presence of this `-failover` flag that causes `in.mpathd` to use the address as a test address, because it is tied to the interface. If IPMP was used to detect failures and not to provide failover, these test addresses could be omitted and the `-failover` flag could be used to define the data addresses.

```
# ifconfig qfe0 addif camelot-qfe0 netmask + broadcast + \
-failover deprecated up
# ifconfig qfe4 addif camelot-qfe4 netmask + broadcast + \
-failover deprecated up
```

To enable the interface configuration to persist after a reboot, the files `hostname.qfe0` and `hostname.qfe4` must be created in the `/etc` directory. The files should look as follows:

For `hostname.qfe0`:

```
camelot netmask + broadcast + failover up \
group production \
addif camelot-qfe0 netmask + broadcast + \
deprecated -failover up
```

For `hostname.qfe4`:

```
camelot-1 netmask + broadcast + failover up \
group production \
addif camelot-qfe4 netmask + broadcast + \
deprecated -failover up
```

---

**Note:** If the MAC address of an interface must be set manually do this as a separate RC script, the ether parameter cannot be included in the above files.

---

## Bug ID: 4710499, “Synopsis: rpc.bootparamd cannot send out reply with physical interface flag DEPRECATED

The advice in the original article was to place the test addresses on the physical interfaces. That is, on `qfe0` and `qfe4` and not on the logical interfaces `qfe0:1` and `qfe4:1` as is now described. The reasoning behind tying the test addresses to the physical interface was as follows.

It simplifies management; the test addresses will not failover, and therefore, they can always be guaranteed to be the first address on an interface. An interface may have more than one data address and this allows for a company standard. For example, test addresses are always found on the physical interface. The logical interfaces can be unplumbed (for example, `ifconfig qfe0:1 unplumb`), whereas unplumbing the physical interface also removes all logical addresses. While data addresses may need to be unplumbed, the test addresses should remain for as long as an interface is wired to a particular subnet.

### ping

If for any reason, a message similar to the following is generated from using the `ping` command to verify connection to one of the data addresses, consult “Addendum - The Problem With `ping`” on page 16.

```
ICMP Protocol Unreachable from gateway camelot (192.168.49.42) for icmp
from clusterclient00 (192.168.49.4) to camelot (192.168.49.42)
```

---

## Basic Management

The configuration of all network adapters can be viewed using the following command:

```
# ifconfig -a
```

Using the example addresses defined in TABLE 1, the `ifconfig -a` command yields the following output:

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232
      index 1
      inet 127.0.0.1 netmask ff000000
qfe0: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
      index 2
      inet 192.168.49.42 netmask ffffffff broadcast 192.168.49.255
      groupname production
      ether 8:0:20:c7:6e:bc
qfe0:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,
NOFAILOVER> mtu 1500 index 2
      inet 192.168.49.105 netmask ffffffff broadcast 192.168.49.255
qfe4: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
      index 3
      inet 192.168.49.7 netmask ffffffff broadcast 192.168.49.255
      groupname production
      ether 8:0:20:b3:e6:f7
qfe4:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4,
NOFAILOVER> mtu 1500 index 3
      inet 192.168.49.106 netmask ffffffff broadcast 192.168.49.255
```

The output in the preceding codebox shows that four addresses have been defined. The two IP network multipathing test addresses, `qfe0:1` and `qfe4:1`, are tied to each interface; that is, they are marked `NOFAILOVER`, and will not migrate to the surviving interface during a failure. The purpose of the addresses being marked `NOFAILOVER` is to detect failure and recovery of an interface.

## Interface Failure

To test that IP network multipathing is functioning correctly (using the example configuration defined in TABLE 1), unplug the network cable attached to `qfe0`. This causes the following error messages to be displayed on the console:

```
Dec 11 16:32:49 camelot qfe: NOTICE: SUNW,qfe0: No response from
Ethernet network: Link Down - cable problem?
Dec 11 16:32:57 camelot in.mpathd[36]: NIC failure detected on qfe0
Dec 11 16:32:57 camelot in.mpathd[36]: Successfully failed over from
NIC qfe0 to NIC qfe4
Dec 11 16:33:01 camelot qfe: NOTICE: SUNW,qfe0: No response from
Ethernet network: Link Down - cable problem?
```

---

**Note:** It takes approximately 10 seconds to detect and recover from a failure with the default configuration. The configuration of the IP Networking Multipathing daemon is set in the `/etc/default/mpathd` file.

---

Executing the `ifconfig -a` command produces the following output:

```
lo0: flags=1000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4> mtu 8232
    index 1
    inet 127.0.0.1 netmask ff000000
qfe0: flags=19000842<BROADCAST,RUNNING,MULTICAST,IPv4,NOFAILOVER,
FAILED> mtu 1500 index 3
    inet 0.0.0.0 netmask 0
    groupname production
    ether 8:0:20:b3:e6:f7
qfe0:1 flags=19040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,
IPv4,NOFAILOVER,FAILED> mtu 1500 index 2
    inet 192.168.49.105 netmask ffffffff broadcast 192.168.49.255
    groupname production
    ether 8:0:20:c7:6e:bc
qfe4: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
index 3
    inet 192.168.49.7 netmask ffffffff broadcast 192.168.49.255
    groupname production
    ether 8:0:20:b3:e6:f7
qfe4:1: flags=9040843<UP,BROADCAST,RUNNING,MULTICAST,DEPRECATED,IPv4
NOFAILOVER> mtu 1500 index 3
    inet 192.168.49.106 netmask ffffffff broadcast 192.168.49.255
qfe4:2: flags=1000843<UP,BROADCAST,RUNNING,MULTICAST,IPv4> mtu 1500
index 3
    inet 192.168.49.42 netmask ffffffff broadcast 192.168.49.255
```

Notice in the preceding codebox the output, `qfe0` has been marked as `FAILED` and the IP address `192.168.49.42` has been moved from `qfe0` to `qfe4:2`; thus, clients can still reach the host at this address.

---

**Note:** To detect errors on a single network interface, use IP network multipathing configured with a group containing one interface and a data address marked `NOFAILOVER`. However, be aware that there is no resilience in such a configuration. Further information on this feature can be found in the *IP Network Multipathing Administration Guide* section of the Solaris 8 OE documentation. Go to: <http://docs.sun.com/>

---

---

## Summary

This article described the necessary steps to configure IP network multipathing, and focused on obtaining maximum resilience with the following configurations:

- Create more than one physical connection to each subnet.
- Connect each subnet to different network adapters on different I/O boards.
- Connect the same subnet to the same port number on different network adapters.
- Create a test network address for each network adapter.
- Create a data address for each adapter.

---

## Addendum - snoop of Failover

The network cable connected to the `qfe0` interface was unplugged to cause the failover and then reconnected to cause the failback. A trace was taken of the network traffic during these events.

The following console messages, show the times of failure detection and recovery. The times can be cross-referenced against the network trace so that the traffic caused by the failure and recovery can be identified.

```
camelot#
camelot#
camelot#
camelot# May  9 16:18:28 camelot qfe: SUNW,qfe0 : No response from Ethernet network : Link down --
cable problem?
May  9 16:18:35 camelot in.mpathd[33]: NIC failure detected on qfe0 of group production
May  9 16:18:35 camelot in.mpathd[33]: Successfully failed over from NIC qfe0 to NIC qfe4
May  9 16:18:40 camelot qfe: SUNW,qfe0 : No response from Ethernet network : Link down -- cable problem?
May  9 16:18:51 camelot last message repeated 1 time
May  9 16:19:01 camelot qfe: SUNW,qfe0 : External Transceiver Selected.
May  9 16:19:01 camelot qfe: SUNW,qfe0 : Auto-Negotiated 100 Mbps Half-Duplex Link Up
May  9 16:19:17 camelot in.mpathd[33]: NIC repair detected on qfe0 of group production
May  9 16:19:17 camelot in.mpathd[33]: Successfully failed back to NIC qfe0
```

The following is the network trace taken

```
16:18:24.43283  camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10541)
16:18:24.43349  defrtr -> camelot-qfe0  ICMP Echo reply (ID: 8450 Sequence number: 10541)
16:18:25.57280  camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10508)
16:18:25.57361  defrtr -> camelot-qfe4  ICMP Echo reply (ID: 8451 Sequence number: 10508)
```

```

16:18:25.78279 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10542)
16:18:25.78338 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10542)
16:18:27.04279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10509)
16:18:27.04348 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10509)
16:18:27.07279 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10543)
16:18:27.07340 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10543)
16:18:28.41295 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10510)
16:18:28.41367 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10510)
16:18:29.68282 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10511)
16:18:29.68344 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10511)
16:18:31.08279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10512)
16:18:31.08379 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10512)
16:18:32.32288 camelot-1 -> 129.152.10.1 DNS C 57.47.153.129.in-addr.arpa. Internet PTR ?
16:18:32.39289 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10513)
16:18:32.39357 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10513)
16:18:33.93279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10514)
16:18:33.93361 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10514)
16:18:35.49433 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:18:35.49473 camelot-1 -> (broadcast) ARP C Who is defrtr, defrtr ?
16:18:35.49610 defrtr -> camelot-1 ARP R defrtr, defrtr is 0:50:bd:bb:a4:0
16:18:35.52294 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10515)
16:18:35.52357 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10515)
16:18:36.56277 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10516)
16:18:36.56337 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10516)
16:18:37.49274 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:18:38.46287 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10517)
16:18:38.46363 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10517)
16:18:39.49265 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:18:39.98279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10518)
16:18:39.98352 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10518)

```

The repeating ICMP Echo requests and replys have been deleted.

```

16:19:14.24296 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10542)
16:19:14.24359 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10542)
16:19:14.52282 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10577)
16:19:14.52343 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10577)
16:19:15.51279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10543)
16:19:15.51347 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10543)
16:19:16.13297 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10578)
16:19:16.13360 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10578)
16:19:17.13298 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10544)
16:19:17.13355 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10544)
16:19:17.53401 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:19:17.53464 camelot -> (broadcast) ARP C Who is defrtr, defrtr ?
16:19:17.53543 defrtr -> camelot ARP R defrtr, defrtr is 0:50:bd:bb:a4:0
16:19:17.53545 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10579)
16:19:17.53601 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10579)
16:19:17.84286 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10580)
16:19:17.84353 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10580)
16:19:18.07316 camelot-1 -> (broadcast) ARP C Who is defrtr, defrtr ?
16:19:18.07376 defrtr -> camelot-1 ARP R defrtr, defrtr is 0:50:bd:bb:a4:0
16:19:18.07380 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10545)
16:19:18.07441 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10545)
16:19:19.10283 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10581)
16:19:19.10341 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10581)
16:19:19.41279 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10546)

```

```

16:19:19.41339 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10546)
16:19:19.53279 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:19:20.24284 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10582)
16:19:20.24342 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10582)
16:19:20.73276 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10547)
16:19:20.73331 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10547)
16:19:21.53278 camelot -> (broadcast) ARP C Who is 192.168.49.42, camelot ?
16:19:21.69284 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10583)
16:19:21.69343 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10583)
16:19:22.79281 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10548)
16:19:22.79350 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10548)
16:19:23.04303 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10584)
16:19:23.04356 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10584)
16:19:23.97280 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10549)
16:19:23.97337 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10549)
16:19:24.31281 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10585)
16:19:24.31352 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10585)
16:19:25.69282 camelot-qfe4 -> defrtr ICMP Echo request (ID: 8451 Sequence number: 10550)
16:19:25.69340 defrtr -> camelot-qfe4 ICMP Echo reply (ID: 8451 Sequence number: 10550)
16:19:25.89281 camelot-qfe0 -> defrtr ICMP Echo request (ID: 8450 Sequence number: 10586)
16:19:25.89352 defrtr -> camelot-qfe0 ICMP Echo reply (ID: 8450 Sequence number: 10586)
16:19:26.84279 camelot-qfe4 -> defrtr ICMP Echo request (ID:8451 Sequence number: 10551)
16:19:26.84354 defrtr -> camelot-qfe4 ICMP Echo reply (ID:8451 Sequence number: 10551)
16:19:27.39283 camelot-qfe0 -> defrtr ICMP Echo request (ID:8450 Sequence number: 10587)
16:19:27.39342 defrtr -> camelot-qfe0 ICMP Echo reply (ID:8450 Sequence number: 10587)
16:19:28.29302 camelot-qfe4 -> defrtr ICMP Echo request (ID:8451 Sequence number: 10552)
16:19:28.29370 defrtr -> camelot-qfe4 ICMP Echo reply (ID:8451 Sequence number: 10552)

```

## Addendum – The Problem With ping

There is a problem with ping and IP network multipathing between Solaris OE version 8 update 2 (10/00) and Solaris OE version 8 update 4 (04/01). The problem has been fixed in Solaris OE version 8 update 5 (07/01) and later. If the router discovery daemon (`in.rdisc`) is running, this problem does not present itself. However, if `in.rdisc` is not running—for example, if an `/etc/defaultrouter` file was created—the following ICMP messages appear if the data address in an IP Mutlipathing group is sent a ping request (not necessarily on the first attempt):

```

ICMP Protocol Unreachable from gateway camelot (192.168.49.42) for
icmp from clusterclient00 (192.168.49.4) to camelot (192.168.49.42)
192.168.49.42 is alive

```

To a degree, this is a cosmetic problem. The ICMP Echo request (ping) has generated an ICMP Echo reply. However, in advance of the reply, it has also generated an ICMP Protocol Unreachable response. If the exit status of a ping command is queried, then a success would be determined. But this is not the case for all varieties of the ping

command. For example, using the `ping` command under Windows 98, it reports that the destination is unreachable. This would be somewhat confusing, because the destination is indeed reachable.

There are basically three approaches to dealing with this problem, which are listed below:

1. Ensure that `in.rdisc` is running either by not creating an `/etc/defaultrouter` file or by starting up `in.rdisc` by some other means. To ensure `in.rdisc` is running, an example startup script could be created. An example of such a startup script is provided at the end of this section.

It should be pointed out that `in.rdisc` is a potential security issue; hence, this solution may not be acceptable in all situations. The router discovery daemon is an implementation of *dynamic* routing that uses ICMP router discovery.

The following is extracted from a Sun BluePrints OnLine article by Keith Watson and Alex Noordergraaf entitled *Solaris Operating Environment Network Settings for Security, Updated for Solaris 8 Operating Environment (December 2000)* located at <http://www.sun.com/blueprints/1200/network-updt1.pdf>. It explains the potential security pit-falls of *dynamic* routing.

“There are several problems with dynamic routing that attackers can use to initiate denial of service attacks or view packet data from inaccessible systems. First, routing information can be forged. Routing information is typically sent through broadcast or multicast packets. An attacker can generate routing information packets claiming to be from a router and send them out to hosts or routers. These packets can direct hosts to send packets to a system that is not a router or to a busy router that cannot handle the increase in traffic. Misconfigured routers generate their own denial of service problems. A more sophisticated attack involves directing packets through a multihomed system to examine the packet data as it flows across this system, which now functions as a router. The attacker sends forged routing information packets to a router claiming a lower *hop count* metric to a destination network that the attacker cannot access. The target router then routes packets through the compromised system allowing the attacker to examine the traffic.”

2. Ignore the ICMP message that affects only the `ping` command, it could be viewed as superfluous. Beware of those versions of the `ping` command that report only the first message returned, this is an error message and may cause the `ping` command to report the destination as unreachable. Hence, this solution may not be applicable in all situations.
3. Avoid sending `ping` requests to the data addresses and send them to the test addresses instead.

## Starting `in.rdisc` By Other Means

The configuration of the network routing tables and default routes is handled automatically by the router discovery daemon (`in.rdisc`).

If routes are defined in the `/etc/defaultrouter` file, the `in.rdisc` daemon will not be started in the `/etc/rc2.d/S69inet` file and will lead to ICMP messages being generated as follows, when the node is sent a ping request (not necessarily on the first ping):

```
ICMP Protocol Unreachable from gateway camelot (192.168.49.42) for icmp
from clusterclient00 (192.168.49.4) to camelot (192.168.49.42)
```

Additionally, `in.rdisc` will fail to start if a working data address is not present at boot time (because it is started with the `-s` flag in `/etc/rc2.d/S69inet`).

To ensure that `in.rdisc` is started under all circumstances, create an additional *startup* script called `/etc/init.d/rdisc`. Create a hard link between this script and an appropriate file name in the startup directory. This determines at which point the script is run during the boot sequence. In this instance, the link would be created as follows:

```
# ln /etc/init.d/rdisc /etc/rc2.d/S70rdisc
```

The following is an example shell script that the file `/etc/init.d/rdisc` could contain to ensure the startup of the `in.rdisc` daemon:

```
#!/sbin/sh
#
#
# If parameter 1 is "start" then check if the router discovery
# daemon, in.rdisc, is running and if not, start it. If parameter 1
# is "stop" then stop in.rdisc
#

case "$1" in
'start')
    if [ -x /usr/bin/pgrep ]
    then
        /usr/bin/pgrep -x -u 0 in.rdisc >/dev/null 2>&1 || \
        /usr/sbin/in.rdisc -f >/dev/msglog 2>&1
    else
        logger Cannot execute /usr/bin/pgrep, in.rdisc not started.
    fi
    ;;

'stop')
    /usr/bin/pkill -x -u 0 in.rdisc
    ;;

*)
    echo "Usage: $0 { start | stop }"
    ;;

esac
exit 0
```

---

## About the Author

Mark Garner is a staff engineer with Sun's Enterprise Engineering organization where he focuses on a broad range of best practice procedures using Sun solutions. Mark joined Sun's professional Service organization in 1998 assisting numerous Global 1000 and Fortune 500 companies implement datacenter solutions. His experience spans financial services, science, government, transportation, retail, utilities, entertainment, and Internet service providers. Mark has over 16 years experience in the computer industry. Prior to joining Sun, he was a Systems Architect with IBM UK, Systems Integration Manager with NERC, and a developer for several leading software development companies.

---

## Ordering Sun Documents

The SunDocs<sup>SM</sup> program provides more than 250 manuals from Sun Microsystems, Inc. If you live in the United States, Canada, Europe, or Japan, you can purchase documentation sets or individual manuals through this program.

---

## Accessing Sun Documentation Online

The `docs.sun.com` web site enables you to access Sun technical documentation online. You can browse the `docs.sun.com` archive or search for a specific book title or subject. The URL is `http://docs.sun.com/`

To reference Sun BluePrints OnLine articles, visit the Sun BluePrints OnLine Web site at: `http://www.sun.com/blueprints/online.html`