

# ◆ ◆ ◆ CHAPTER 7

## Introducing IPMP

---

IP network multipathing (IPMP) provides physical interface failure detection, transparent network access failover, and packet load spreading for systems with multiple interfaces that are connected to a particular local area network or LAN.

This chapter contains the following information:

- “What's New With IPMP” on page 101
- “Deploying IPMP” on page 102
- “Solaris IPMP Components” on page 111
- “Types of IPMP Interface Configurations” on page 112
- “IPMP Addressing” on page 113
- “Failure and Repair Detection in IPMP” on page 114
- “IPMP and Dynamic Reconfiguration” on page 118
- “IPMP Terminology and Concepts” on page 120

---

**Note** – Throughout the description of IPMP in this chapter and in [Chapter 8, “Administering IPMP”](#), all references to the term *interface* specifically mean *IP interface*. Unless a qualification explicitly indicates a different use of the term, such as a network interface card (NIC), the term always refers to the interface that is configured on the IP layer.

---

## What's New With IPMP

The following features differentiate the current IPMP implementation from the previous implementation:

- An IPMP group is represented as an IPMP IP interface. This interface is treated just like any other interface on the IP layer of the networking stack. All IP administrative tasks, routing tables, Address Resolution Protocol (ARP) tables, firewall rules, and other IP-related procedures work with an IPMP group by referring to the IPMP interface.

- The system becomes responsible for the distribution of data addresses among underlying active interfaces. In the previous IPMP implementation, the administrator initially determines the binding of data addresses to corresponding interfaces when the IPMP group is created. In the current implementation, when the IPMP group is created, data addresses belong to the IPMP interface as an address pool. The kernel then automatically and randomly binds the data addresses to the underlying active interfaces of the group.
- The `ipmpstat` tool is introduced as the principal tool to obtain information about IPMP groups. This command provides information about all aspects of the IPMP configuration, such as the underlying IP interfaces of the group, test and data addresses, types of failure detection being used, and which interfaces have failed. The `ipmpstat` functions, the options you can use, and the output each option generates are all described in [“Monitoring IPMP Information” on page 149](#).
- The IPMP interface can be assigned a customized name to identify the IPMP group more easily within your network setup. For the procedures to configure IPMP groups with customized names, see any procedure that describes the creation of an IPMP group in [“Configuring IPMP Groups” on page 130](#).

## Deploying IPMP

This section describes various topics about the use of IPMP groups.

### Why You Should Use IPMP

Different factors can cause an interface to become unusable. Commonly, an IP interface can fail. Or, an interface might be switched offline for hardware maintenance. In such cases, without an IPMP group, the system can no longer be contacted by using any of the IP addresses that are associated with that unusable interface. Additionally, existing connections that use those IP addresses are disrupted.

With IPMP, one or more IP interfaces can be configured into an *IPMP group*. The group functions like an IP interface with data addresses to send or receive network traffic. If an underlying interface in the group fails, the data addresses are redistributed among the remaining underlying active interfaces in the group. Thus, the group maintains network connectivity despite an interface failure. With IPMP, network connectivity is always available, provided that a minimum of one interface is usable for the group.

Additionally, IPMP improves overall network performance by automatically spreading out outbound network traffic across the set of interfaces in the IPMP group. This process is called outbound *load spreading*. The system also indirectly controls inbound load spreading by performing source address selection for packets whose IP source address was not specified by the application. However, if an application has explicitly chosen an IP source address, then the system does not vary that source address.

## When You Must Use IPMP

The configuration of an IPMP group is determined by your system configurations. Observe the following rules:

- Multiple IP interfaces on the same local area network or LAN must be configured into an IPMP group. LAN broadly refers to a variety of local network configurations including VLANs and both wired and wireless local networks whose nodes belong to *the same link-layer broadcast domain*.
- Underlying IP interfaces of an IPMP group must not span different LANs.

For example, suppose that a system with three interfaces is connected to two separate LANs. Two IP interfaces link to one LAN while a single IP interface connects to the other. In this case, the two IP interfaces connecting to the first LAN must be configured as an IPMP group, as required by the first rule. In compliance with the second rule, the single IP interface that connects to the second LAN cannot become a member of that IPMP group. No IPMP configuration is required of the single IP interface. However, you can configure the single interface into an IPMP group to monitor the interface's availability. The single-interface IPMP configuration is discussed further in [“Types of IPMP Interface Configurations” on page 112](#).

Consider another case where the link to the first LAN consists of three IP interfaces while the other link consists of two interfaces. This setup requires the configuration of two IPMP groups: a three-interface group that links to the first LAN, and a two-interface group to connect to the second.

## Comparing IPMP and Link Aggregation

IPMP and link aggregation are different technologies to achieve improved network performance as well as maintain network availability. In general, you deploy link aggregation to obtain better network performance, while you use IPMP to ensure high availability.

The following table presents a general comparison between link aggregation and IPMP.

	IPMP	Link Aggregation
Network technology type	Layer 3 (IP layer)	Layer 2 (link layer)
Configuration tool	<code>ifconfig</code>	<code>dladm</code>
Link-based failure detection	Supported.	Supported.

	IPMP	Link Aggregation
Probe-based failure detection	ICMP-based, targeting any defined system in the same IP subnet as test addresses, across multiple levels of intervening layer-2 switches.	Based on Link Aggregation Control Protocol (LACP), targeting immediate peer host or switch.
Use of standby interfaces	Supported	Not supported
Span multiple switches	Supported	Generally not supported; some vendors provide proprietary and non-interoperable solutions to span multiple switches.
Hardware support	Not required	Required. For example, a link aggregation in the system that is running the Solaris OS requires that corresponding ports on the switches be also aggregated.
Link layer requirements	Broadcast-capable	Ethernet-specific
Driver framework requirements	None	Must use GLDv3 framework
Load spreading support	Present, controlled by kernel. Inbound load spreading is indirectly affected by source address selection.	Finer grain control of the administrator over load spreading of outbound traffic by using <code>dladm</code> command. Inbound load spreading supported.

In link aggregations, incoming traffic is spread over the multiple links that comprise the aggregation. Thus, networking performance is enhanced as more NICs are installed to add links to the aggregation. IPMP's traffic uses the IPMP interface's data addresses as they are bound to the available active interfaces. If, for example, all the data traffic is flowing between only two IP addresses but not necessarily over the same connection, then adding more NICs will not improve performance with IPMP because only two IP addresses remain usable.

The two technologies complement each other and can be deployed together to provide the combined benefits of network performance and availability. For example, except where proprietary solutions are provided by certain vendors, link aggregations currently cannot span multiple switches. Thus, a switch becomes a single point of failure for a link aggregation between the switch and a host. If the switch fails, the link aggregation is likewise lost, and network performance declines. IPMP groups do not face this switch limitation. Thus, in the scenario of a LAN using multiple switches, link aggregations that connect to their respective switches can be combined into an IPMP group on the host. With this configuration, both enhanced network performance as well as high availability are obtained. If a switch fails, the data addresses of the link aggregation to that failed switch are redistributed among the remaining link aggregations in the group.

For other information about link aggregations, see [Chapter 5, “Administering Link Aggregations.”](#)

## Using Flexible Link Names on IPMP Configuration

With support for customized link names, link configuration is no longer bound to the physical NIC to which the link is associated. Using customized link names allows you to have greater flexibility in administering IP interfaces. This flexibility extends to IPMP administration as well. If an underlying interface of an IPMP group fails and requires a replacement, the procedures to replace the interface is greatly facilitated. The replacement NIC, provided it is the same type as the failed NIC, can be renamed to inherit the configuration of the failed NIC. You do not have to create new configurations before you can add a new interface to the IPMP group. After you assign the link name of the failed NIC to the new NIC, the new NIC is configured with the same settings as the failed interface. The multipathing daemon then deploys the interface according to the IPMP configuration of active and standby interfaces.

Therefore, to optimize your networking configuration and facilitate IPMP administration, you must employ flexible link names for your interfaces by assigning them generic names. In the following section “[How IPMP Works](#)” on page 105, all the examples use flexible link names for the IPMP group and its underlying interfaces. For details about the processes behind NIC replacements in a networking environment that uses customized link names, refer to “[IPMP and Dynamic Reconfiguration](#)” on page 118. For an overview of the networking stack and the use of customized link names, refer to “[Overview of the Networking Stack](#)” on page 15.

## How IPMP Works

IPMP maintains network availability by attempting to preserve the original number of active and standby interfaces when the group was created.

IPMP failure detection can be link-based or probe-based or both to determine the availability of a specific underlying IP interface in the group. If IPMP determines that an underlying interface has failed, then that interface is flagged as failed and is no longer usable. The data IP address that was associated with the failed interface is then redistributed to another functioning interface in the group. If available, a standby interface is also deployed to maintain the original number of active interfaces.

Consider a three-interface IPMP group `itops0` with an active-standby configuration, as illustrated in [Figure 7-1](#).

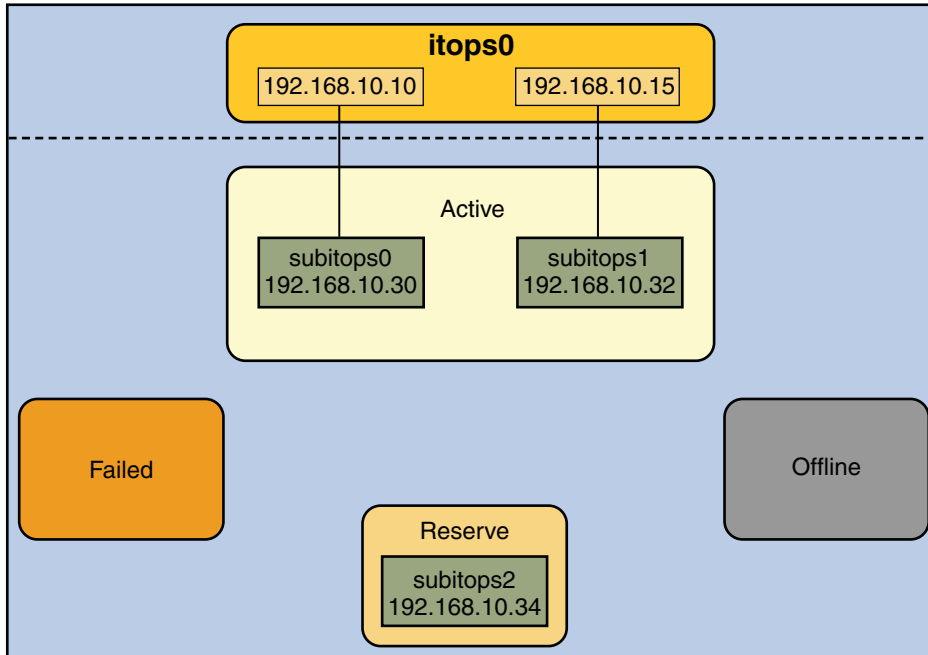


FIGURE 7-1 IPMP Active-Standby Configuration

The group `itops0` is configured as follows:

- Two data addresses are assigned to the group: `192.168.10.10` and `192.168.10.15`.
- Two underlying interfaces are configured as active interfaces and are assigned flexible link names: `subitops0` and `subitops1`.
- The group has one standby interface, also with a flexible link name: `subitops2`.
- Probe-based failure detection is used, and thus the active and standby interfaces are configured with test addresses, as follows:
  - `subitops0: 192.168.10.30`
  - `subitops1: 192.168.10.32`
  - `subitops2: 192.168.10.34`

**Note** – The Active, Offline, Reserve, and Failed areas in the figures indicate only the status of underlying interfaces, and not physical locations. No physical movement of interfaces or addresses nor transfer of IP interfaces occur within this IPMP implementation. The areas only serve to show how an underlying interface changes status as a result of either failure or repair.

You can use the `ipmpstat` command with different options to display specific types of information about existing IPMP groups. For additional examples, see [“Monitoring IPMP Information” on page 149](#).

The IPMP configuration in [Figure 7-1](#) can be displayed by using the following `ipmpstat` command:

```
# ipmpstat -g
GROUP      GROUPNAME  STATE   FDT      INTERFACES
itops0     itops0     ok      10.00s   subitops1 subitops0 (subitops2)
```

To display information about the group's underlying interfaces, you would type the following:

```
# ipmpstat -i
INTERFACE  ACTIVE  GROUP  FLAGS  LINK  PROBE  STATE
subitops0  yes    itops0  -----  up    ok     ok
subitops1  yes    itops0  --mb---  up    ok     ok
subitops2  no     itops0  is-----  up    ok     ok
```

IPMP maintains network availability by managing the underlying interfaces to preserve the original number of active interfaces. Thus, if `subitops0` fails, then `subitops2` is deployed to ensure that the group continues to have two active interfaces. The activation of the `subitops2` is shown in [Figure 7-2](#).

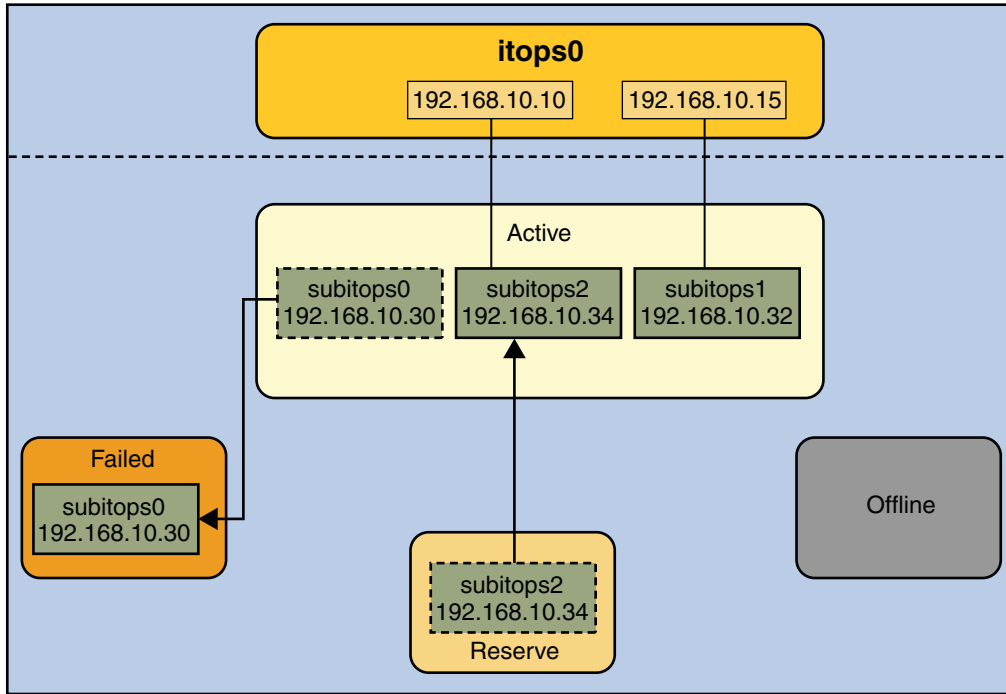


FIGURE 7-2 Interface Failure in IPMP

**Note** – The one-to-one mapping of data addresses to active interfaces in Figure 7-2 serves only to simplify the illustration. The IP kernel module can assign data addresses randomly without necessarily adhering to a one-to-one relationship between data addresses and interfaces.

The `ipmpstat` utility displays the information in Figure 7-2 as follows:

```
# ipmpstat -i
INTERFACE      ACTIVE   GROUP    FLAGS    LINK    PROBE    STATE
subitops0      no      itops0   - - - - - up      failed   failed
subitops1      yes     itops0   - - mb - - up      ok       ok
subitops2      yes     itops0   - s - - - up      ok       ok
```

After `subitops0` is repaired, then it reverts to its status as an active interface. In turn, `subitops2` is returned to its original standby status.

A different failure scenario is shown in Figure 7-3, where the standby interface `subitops2` fails (1), and later, one active interface, `subitops1`, is switched offline by the administrator (2). The result is that the IPMP group is left with a single functioning interface, `subitops0`.

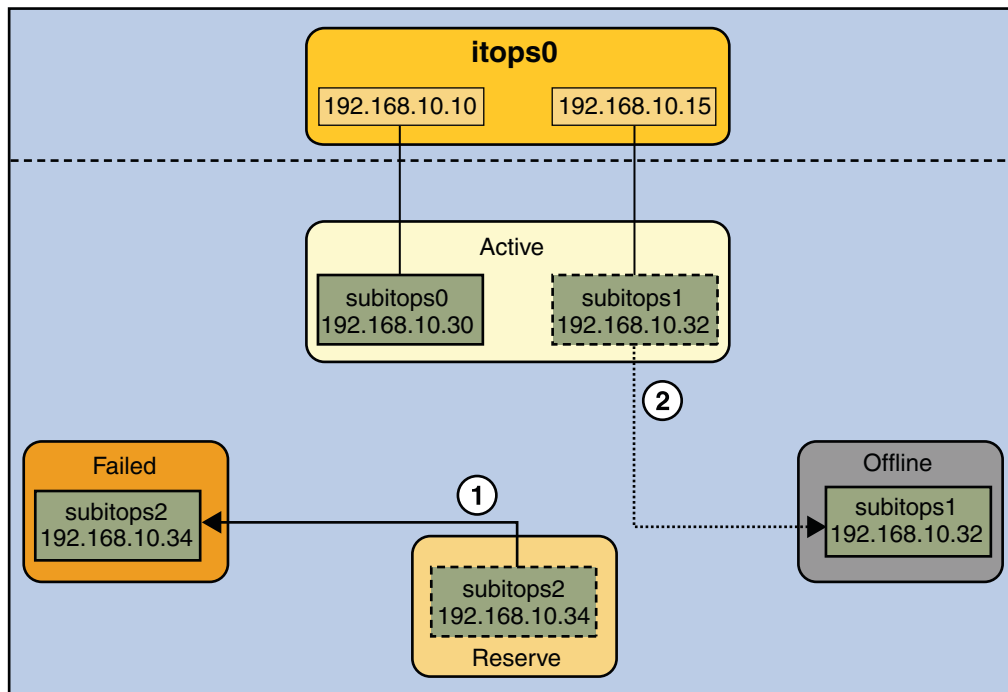


FIGURE 7-3 Standby Interface Failure in IPMP

The `impstat` utility would display the information illustrated by Figure 7-3 as follows:

```
# impstat -i
INTERFACE      ACTIVE   GROUP   FLAGS    LINK    PROBE    STATE
subitops0     yes     itops0  - - - - -  up      ok       ok
subitops1     no      itops0  - - m b - d -  up      ok       offline
subitops2     no      itops0  i s - - - -  up      failed  failed
```

For this particular failure, the recovery after an interface is repaired behaves differently. The restoration depends on the IPMP group's original number of active interfaces compared with the configuration after the repair. The recovery process is represented graphically in Figure 7-4.

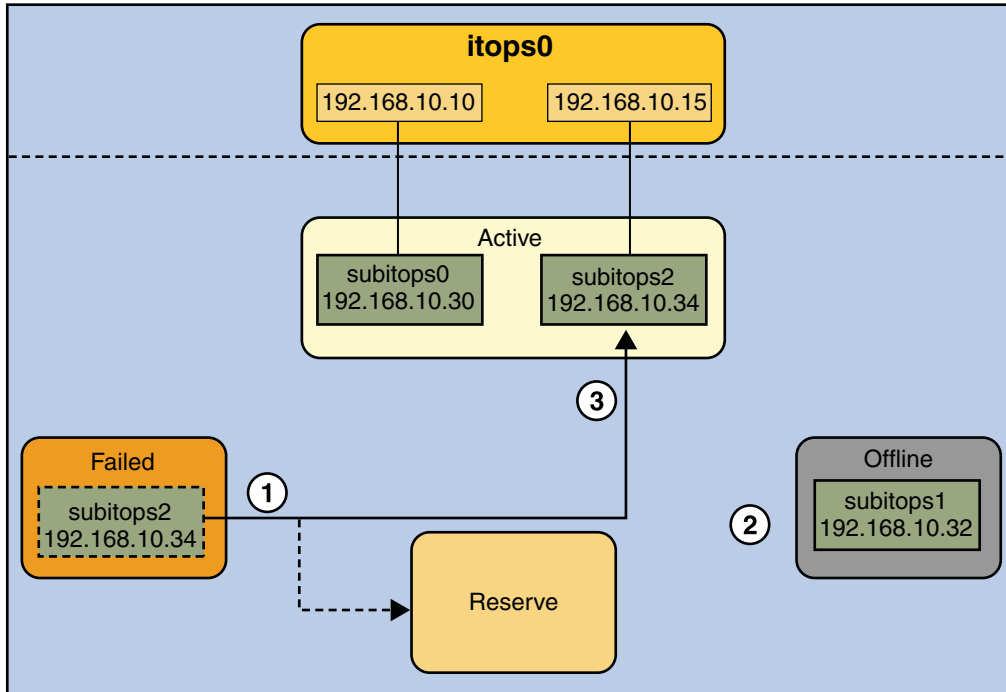


FIGURE 7-4 IPMP Recovery Process

In Figure 7-4, when subitops2 is repaired, it would normally revert to its original status as a standby interface (1). However, the IPMP group still would not reflect the original number of two active interfaces, because subitops1 continues to remain offline (2). Thus, IPMP deploys subitops2 as an active interface instead (3).

The `ipmpstat` utility would display the post-repair IPMP scenario as follows:

```
# ipmpstat -i
INTERFACE      ACTIVE  GROUP   FLAGS   LINK    PROBE   STATE
subitops0     yes    itops0  - - - - -  up      ok      ok
subitops1     no     itops0  - - m b - d -  up      ok      offline
subitops2     yes    itops0  - s - - - - -  up      ok      ok
```

A similar restore sequence occurs if the failure involves an active interface that is also configured in `FAILBACK=no` mode, where a failed active interface does not automatically revert to active status upon repair. Suppose subitops0 in Figure 7-2 is configured in `FAILBACK=no` mode. In that mode, a repaired subitops0 is switched to a reserve status as a standby interface, even though it was originally an active interface. The interface subitops2 would remain active to maintain the IPMP group's original number of two active interfaces. The `ipmpstat` utility would display the recovery information as follows:

```
# ipmpstat -i
INTERFACE      ACTIVE    GROUP    FLAGS    LINK    PROBE    STATE
subitops0      no       itops0   i----- up      ok       ok
subitops1      yes      itops0   --mb--- up      ok       ok
subitops2      yes      itops0   -s----- up      ok       ok
```

For more information about this type of configuration, see [“The FAILBACK=no Mode” on page 117](#).

## Solaris IPMP Components

Solaris IPMP involves the following software:

The *multipathing daemon* `in.mpathd` detects interface failures and repairs. The daemon performs both link-based failure detection and probe-based failure detection if test addresses are configured for the underlying interfaces. Depending on the type of failure detection method that is employed, the daemon sets or clears the appropriate flags on the interface to indicate whether the interface failed or has been repaired. As an option, the daemon can also be configured to monitor the availability of all interfaces, including those that are not configured to belong to an IPMP group. For a description of failure detection, see [“Failure and Repair Detection in IPMP” on page 114](#).

The `in.mpathd` daemon also controls the designation of active interfaces in the IPMP group. The daemon attempts to maintain the same number of active interfaces that was originally configured when the IPMP group was created. Thus `in.mpathd` activates or deactivates underlying interfaces as needed to be consistent with the administrator's configured policy. For more information about the manner by which the `in.mpathd` daemon manages activation of underlying interfaces, refer to [“How IPMP Works” on page 105](#). For more information about the daemon, refer to the `in.mpathd(1M)` man page.

The *IP kernel module* manages outbound load-spreading by distributing the set of available IP data addresses in the group across the set of available underlying IP interfaces in the group. The module also performs source address selection to manage inbound load-spreading. Both roles of the IP module improve network traffic performance.

The *IPMP configuration file* `/etc/default/mpathd` is used to configure the daemon's behavior. For example, you can specify how the daemon performs probe-based failure detection by setting the time duration to probe a target to detect failure, or which interfaces to probe. You can also specify what the status of a failed interface should be after that interface is repaired. You also set the parameters in this file to specify whether the daemon should monitor all IP interfaces in the system, not only those that are configured to belong to IPMP groups. For procedures to modify the configuration file, refer to [“How to Configure the Behavior of the IPMP Daemon” on page 145](#).

The *ipmpstat utility* provides different types of information about the status of IPMP as a whole. The tool also displays other specific information about the underlying IP interfaces for each

group, as well as data and test addresses that have been configured for the group. For more information about the use of this command, see [“Monitoring IPMP Information” on page 149](#) and the `ipmpstat(1M)` man page.

## Types of IPMP Interface Configurations

An IPMP configuration typically consists of two or more physical interfaces on the same system that are attached to the same LAN. These interfaces can belong to an IPMP group in either of the following configurations:

- active-active configuration – an IPMP group in which all underlying interfaces are active. An *active interface* is an IP interface that is currently available for use by the IPMP group. By default, an underlying interface becomes active when you configure the interface to become part of an IPMP group. For additional information about active interfaces and other IPMP terms, see also [“IPMP Terminology and Concepts” on page 120](#).
- active-standby configuration – an IPMP group in which at least one interface is administratively configured as a reserve. The reserve interface is called the *standby interface*. Although idle, the standby IP interface is monitored by the multipathing daemon to track the interface's availability, depending on how the interface is configured. If link-failure notification is supported by the interface, link-based failure detection is used. If the interface is configured with a test address, probe-based failure detection is also used. If an active interface fails, the standby interface is automatically deployed as needed. You can configure as many standby interfaces as you want for an IPMP group.

A single interface can also be configured in its own IPMP group. The single interface IPMP group has the same behavior as an IPMP group with multiple interfaces. However, this IPMP configuration does not provide high availability for network traffic. If the underlying interface fails, then the system loses all capability to send or receive traffic. The purpose of configuring a single-interfaced IPMP group is to monitor the availability of the interface by using failure detection. By configuring a test address on the interface, you can set the daemon to track the interface by using probe-based failure detection. Typically, a single-interfaced IPMP group configuration is used in conjunction with other technologies that have broader failover capabilities, such as Sun Cluster software. The system can continue to monitor the status of the underlying interface. But the Sun Cluster software provides the functionalities to ensure availability of the network when failure occurs. For more information about the Sun Cluster software, see [Sun Cluster Overview for Solaris OS](#).

An IPMP group without underlying interfaces can also exist, such as a group whose underlying interfaces have been removed. The IPMP group is not destroyed, but the group cannot be used to send and receive traffic. As underlying IP interfaces are brought online for the group, then the data addresses of the IPMP interface are allocated to these interfaces and the system resumes hosting network traffic.

## IPMP Addressing

You can configure IPMP failure detection on both IPv4 networks and dual-stack, IPv4 and IPv6 networks. Interfaces that are configured with IPMP support two types of addresses:

- *Data Addresses* are the conventional IPv4 and IPv6 addresses that are assigned to an IP interface dynamically at boot time by the DHCP server, or manually by using the `ifconfig` command. Data addresses are assigned to the IPMP interface. The standard IPv4 packet traffic and, if applicable, IPv6 packet traffic are considered *data traffic*. Data traffic flow use the data addresses that are hosted on the IPMP interface and flow through the active interfaces of that group.
- *Test Addresses* are IPMP-specific addresses that are used by the `in.mpathd` daemon to perform probe-based failure and repair detection. Test addresses can also be assigned dynamically by the DHCP server, or manually by using the `ifconfig` command. These addresses are configured with the `NOFAILOVER` flag that identifies them as test addresses. While data addresses are assigned to the IPMP interface, only test addresses are assigned to the underlying interfaces of the group. For an underlying interface on a dual-stack network, you can configure an IPv4 test address or an IPv6 test address or both. When an underlying interface fails, the interface's test address continues to be used by the `in.mpathd` daemon for probe-based failure detection to check for the interface's subsequent repair.

---

**Note** – You need to configure test addresses only if you specifically want to use probe-based failure detection. For more information about probe-based failure detection and the use of test addresses, refer to “[Probe-Based Failure Detection](#)” on page 115.

---

In previous IPMP implementations, test addresses needed to be marked as `DEPRECATED` to avoid being used by applications especially during interface failures. In the current implementation, test addresses reside in the underlying interfaces. Thus, these addresses can no longer be accidentally used by applications that are unaware of IPMP. However, to ensure that these addresses will not be considered as a possible source for data packets, the system automatically marks any addresses with the `NOFAILOVER` flag as also `DEPRECATED`.

### IPv4 Test Addresses

In general, you can use any IPv4 address on your subnet as a test address. IPv4 test addresses do not need to be routeable. Because IPv4 addresses are a limited resource for many sites, you might want to use non-routeable RFC 1918 private addresses as test addresses. Note that the `in.mpathd` daemon exchanges only ICMP probes with other hosts on the same subnet as the test address. If you do use RFC 1918-style test addresses, be sure to configure other systems, preferably routers, on the network with addresses on the appropriate RFC 1918 subnet. The `in.mpathd` daemon can then successfully exchange probes with target systems. For more information about RFC 1918 private addresses, refer to [RFC 1918, Address Allocation for Private Internets](#) (<http://www.ietf.org/rfc/rfc1918.txt?number=1918>).

## IPv6 Test Addresses

The only valid IPv6 test address is the link-local address of a physical interface. You do not need a separate IPv6 address to serve as an IPMP test address. The IPv6 link-local address is based on the Media Access Control (MAC) address of the interface. Link-local addresses are automatically configured when the interface becomes IPv6-enabled at boot time or when the interface is manually configured through `ifconfig`. Just like IPv4 test addresses, IPv6 test addresses must be configured with the `NOFAILOVER` flag.

For more information on link-local addresses, refer to “[Link-Local Unicast Address](#)” in *System Administration Guide: IP Services*.

When an IPMP group has both IPv4 and IPv6 plumbed on all the group's interfaces, you do not need to configure separate IPv4 test addresses. The `in.mpathd` daemon can use the IPv6 link-local addresses with the `NOFAILOVER` flag as test addresses.

## Failure and Repair Detection in IPMP

To ensure continuous availability of the network to send or receive traffic, IPMP performs failure detection on the IPMP group's underlying IP interfaces. Failed interfaces remain unusable until these are repaired. Remaining active interfaces continue to function while any existing standby interfaces are deployed as needed.

A *group failure* occurs when all interfaces in an IPMP group appear to fail at the same time. In this case, no underlying interface is usable. Also, when all the target systems fail at the same time and probe-based failure detection is enabled, the `in.mpathd` daemon flushes all of its current target systems and probes for new target systems.

## Types of Failure Detection in IPMP

The `in.mpathd` daemon handles the following types of failure detection:

- Link-based failure detection, if supported by the NIC driver
- Probe-based failure detection, when test addresses are configured
- Detection of interfaces that were missing at boot time

### Link-Based Failure Detection

Link-based failure detection is always enabled, provided that the interface supports this type of failure detection.

To determine whether a third-party interface supports link-based failure detection, use the `ipmpstat -i` command. If the output for a given interface includes an unknown status for its LINK column, then that interface does not support link-based failure detection. Refer to the manufacturer's documentation for more specific information about the device.

These network drivers that support link-based failure detection monitor the interface's link state and notify the networking subsystem when that link state changes. When notified of a change, the networking subsystem either sets or clears the `RUNNING` flag for that interface, as appropriate. If the `in.mpathd` daemon detects that the interface's `RUNNING` flag has been cleared, the daemon immediately fails the interface.

## Probe-Based Failure Detection

The multipathing daemon performs probe-based failure detection on each interface in the IPMP group that has a test address. Probe-based failure detection involves sending and receiving ICMP probe messages that use test addresses. These messages, also called *probe traffic* or test traffic, go out over the interface to one or more target systems on the same local network. The daemon probes all the targets separately through all the interfaces that have been configured for probe-based failure detection. If no replies are made in response to five consecutive probes on a given interface, `in.mpathd` considers the interface to have failed. The probing rate depends on the *failure detection time* (FDT). The default value for failure detection time is 10 seconds. However, you can tune the failure detection time in the IPMP configuration file. For instructions, go to [“How to Configure the Behavior of the IPMP Daemon” on page 145](#). To optimize probe-based failure detection, you must set multiple target systems to receive the probes from the multipathing daemon. By having multiple target systems, you can better determine the nature of a reported failure. For example, the absence of a response from the only defined target system can indicate a failure either in the target system or in one of the IPMP group's interfaces. By contrast, if only one system among several target systems does not respond to a probe, then the failure is likely in the target system rather than in the IPMP group itself.

*Repair detection time* is twice the failure detection time. The default time for failure detection is 10 seconds. Accordingly, the default time for repair detection is 20 seconds. After a failed interface has been repaired and the interface's `RUNNING` flag is once more detected, `in.mpathd` clears the interface's `FAILED` flag. The repaired interface is redeployed depending on the number of active interfaces that the administrator has originally set.

The `in.mpathd` daemon determines which target systems to probe dynamically. First the daemon searches the routing table for target systems that are on the same subnet as the test addresses that are associated with the IPMP group's interfaces. If such targets are found, then the daemon uses them as targets for probing. If no target systems are found on the same subnet, then `in.mpathd` sends multicast packets to probe neighbor hosts on the link. The multicast packet is sent to the all hosts multicast address, `224.0.0.1` in IPv4 and `ff02::1` in IPv6, to determine which hosts to use as target systems. The first five hosts that respond to the echo packets are chosen as targets for probing. If `in.mpathd` cannot find routers or hosts that responded to the multicast probes, then ICMP echo packets, `in.mpathd` cannot detect probe-based failures. In this case, the `ipmpstat -i` utility will report the probe state as unknown.

You can use host routes to explicitly configure a list of target systems to be used by `in.mpathd`. For instructions, refer to [“Configuring for Probe-Based Failure Detection” on page 144](#).

## NICs That Are Missing at Boot

NICs that are not present at system boot represent a special instance of failure detection. At boot time, the startup scripts track any interfaces with `/etc/hostname.interface` files. Any data addresses in such an interface's `/etc/hostname.interface` file are automatically configured on the corresponding IPMP interface for the group. However, if the interfaces themselves cannot be plumbed because they are missing, then error messages similar to the following are displayed:

```
moving addresses from missing IPv4 interfaces: hme0 (moved to ipmp0)
moving addresses from missing IPv6 interfaces: hme0 (moved to ipmp0)
```

---

**Note** – In this instance of failure detection, only data addresses that are explicitly specified in the missing interface's `/etc/hostname.interface` file are moved to the IPMP interface.

---

If an interface with the same name as another interface that was missing at system boot is reattached using DR, the Reconfiguration Coordination Manager (RCM) automatically plumbs the interface. Then, RCM configures the interface according to the contents of the interface's `/etc/hostname.interface` file. However, data addresses, which are addresses without the `NOFAILOVER` flag, that are in the `/etc/hostname.interface` file are ignored. This mechanism adheres to the rule that data addresses should be in the `/etc/hostname.ipmp-interface` file, and only test addresses should be in the underlying interface's `/etc/hostname.interface` file. Issuing the `ifconfig` group command causes that interface to again become part of the group. Thus, the final network configuration is identical to the configuration that would have been made if the system had been booted with the interface present.

For more information about missing interfaces, see [“About Missing Interfaces at System Boot” on page 149](#).

## Failure Detection and the Anonymous Group Feature

IPMP supports failure detection in an anonymous group. By default, IPMP monitors the status only of interfaces that belong to IPMP groups. However, the IPMP daemon can be configured to also track the status of interfaces that do not belong to any IPMP group. Thus, these interfaces are considered to be part of an “anonymous group.” When you issue the command `ipmpstat -g`, the anonymous group will be displayed as double-dashes (- -). In anonymous groups, the interfaces would have their data addresses function also as test addresses. Because these interfaces do not belong to a named IPMP group, then these addresses are visible to applications. To enable tracking of interfaces that are not part of an IPMP group, see [“How to Configure the Behavior of the IPMP Daemon” on page 145](#).

## Detecting Physical Interface Repairs

When an underlying interface fails and probe-based failure detection is used, the `in.mpathd` daemon continues to use the interface's test address to continue probing target systems. During an interface repair, the restoration proceeds depending on the original configuration of the failed interface:

- Failed interface was originally an active interface – the repaired interface reverts to its original active status. The standby interface that functioned as a replacement during the failure is switched back to standby status if enough interfaces are active for the group as defined by the system administrator.

---

**Note** – An exception to this step are cases when the repaired active interface is also configured with the `FAILBACK=no` mode. For more information, see [“The FAILBACK=no Mode” on page 117](#)

---

- Failed interface was originally a standby interface – the repaired interface reverts to its original standby status, provided that the IPMP group reflects the original number of active interfaces. Otherwise, the standby interface is switched to become an active interface.

To see a graphical presentation of how IPMP behaves during interface failure and repair, see [“How IPMP Works” on page 105](#).

### The FAILBACK=no Mode

By default, active interfaces that have failed and then repaired automatically return to become active interfaces in the group. This behavior is controlled by the setting of the `FAILBACK` parameter in the daemon's configuration file. However, even the insignificant disruption that occurs as data addresses are remapped to repaired interfaces might not be acceptable to some administrators. The administrators might prefer to allow an activated standby interface to continue as an active interface. IPMP allows administrators to override the default behavior to prevent an interface to automatically become active upon repair. These interfaces must be configured in the `FAILBACK=no` mode. For related procedures, see [“How to Configure the Behavior of the IPMP Daemon” on page 145](#).

When an active interface in `FAILBACK=no` mode fails and is subsequently repaired, the IPMP daemon restores the IPMP configuration as follows:

- The daemon retains the interface's `INACTIVE` status, provided that the IPMP group reflects the original configuration of active interfaces.
- If the IPMP configuration at the moment of repair does not reflect the group's original configuration of active interfaces, then the repaired interface is redeployed as an active interface, notwithstanding the `FAILBACK=no` status.

---

**Note** – The FAILBACK=NO mode is set for the whole IPMP group. It is not a per-interface tunable parameter.

---

## IPMP and Dynamic Reconfiguration

Dynamic reconfiguration (DR) feature allows you to reconfigure system hardware, such as interfaces, while the system is running. DR can be used only on systems that support this feature.

You typically use the `cfgadm` command to perform DR operations. However, some platforms provide other methods. Make sure to consult your platform's documentation for details to perform DR. For systems that use the Solaris OS, you can find specific documentation about DR in the resources that are listed in [Table 7-1](#). Current information about DR is also available at <http://docs.sun.com> and can be obtained by searching for the topic “dynamic reconfiguration.”

**TABLE 7-1** Documentation Resources for Dynamic Reconfiguration

Description	For Information
Detailed information on the <code>cfgadm</code> command	<a href="#">cfgadm(1M)</a> man page
Specific information about DR in the Sun Cluster environment	<i>Sun Cluster 3.1 System Administration Guide</i>
Specific information about DR in the Sun Fire environment	<i>Sun Fire 880 Dynamic Reconfiguration Guide</i>
Introductory information about DR and the <code>cfgadm</code> command	Chapter 6, “Dynamically Configuring Devices (Tasks),” in <i>System Administration Guide: Devices and File Systems</i>
Tasks for administering IPMP groups on a system that supports DR	“ <a href="#">Recovering an IPMP Configuration With Dynamic Reconfiguration</a> ” on page 147

The sections that follow explain how DR interoperates with IPMP.

On a system that supports DR of NICs, IPMP can be used to preserve connectivity and prevent disruption of existing connections. IPMP is integrated into the Reconfiguration Coordination Manager (RCM) framework. Thus, you can safely attach, detach, or reattach NICs and RCM manages the dynamic reconfiguration of system components.

## Attaching New NICs

With DR support, you can attach, plumb, and then add new interfaces to existing IPMP groups. Or, if appropriate, you can configure the newly added interfaces into their own IPMP group. For procedures to configure IPMP groups, refer to [“Configuring IPMP Groups” on page 130](#). After these interfaces have been configured, they are immediately available for use by IPMP. However, to benefit from the advantages of using customized link names, you must assign generic link names to replace the interface's hardware-based link names. Then you create corresponding configuration files by using the generic name that you just assigned. For procedures to configure a single interface by using customized link names, refer to [“How to Configure an IP Interface After System Installation” on page 42](#). After you assign a generic link name to interface, make sure that you always refer to the generic name when you perform any additional configuration on the interface such as using the interface for IPMP.

## Detaching NICs

All requests to detach system components that contain NICs are first checked to ensure that connectivity can be preserved. For instance, by default you cannot detach a NIC that is not in an IPMP group. You also cannot detach a NIC that contains the only functioning interfaces in an IPMP group. However, if you must remove the system component, you can override this behavior by using the `-f` option of `cfgadm`, as explained in the `cfgadm(1M)` man page.

If the checks are successful, the daemon sets the `OFFLINE` flag for the interface. All test addresses on the interfaces are unconfigured. Then, the NIC is unplumbed from the system. If any of these steps fail, or if the DR of other hardware on the same system component fails, then the previous configuration is restored to its original state. A status message about this event will be displayed. Otherwise, the detach request completes successfully. You can remove the component from the system. No existing connections are disrupted.

## Replacing NICs

When an underlying interface of an IPMP group fails, a typical solution would be to replace the failed interface by attaching a new NIC. RCM records the configuration information associated with any NIC that is detached from a running system. If you replace a failed NIC with an *identical* NIC, then RCM automatically configures the interface according to the contents of the existing `/etc/hostname.interface` file.

For example, suppose you replace a failed `bge0` interface with another `bge0` interface. The failed `bge0` already has a corresponding `/etc/hostname.bge0` file. After you attach the replacement `bge` NIC, RCM plumbs and then configures the `bge0` interface by using the information in the `/etc/hostname.bge0` file. Thus the interface is properly configured with the test address and is added to the IPMP group according to the contents of the configuration file.

You can replace a failed NIC with a different NIC, provided that both are the same type, such as ethernet. In this case, RCM plumbs the new interface after it is attached. If you did not use customized link names when you first configured your interfaces, and no corresponding configuration file for the new interface exists, then you will have to perform additional configuration steps. You will need to create a new corresponding configuration file for the new NIC. Additionally, you will need to add correct information to the file before you can add the interface to the IPMP group.

However, if you used customized link names, the additional configuration steps are unnecessary. By reassigning the failed interface's link name to the new interface, then the new interface acquires the configuration specified in the removed interface's configuration file. RCM then configures the interface by using the information in that file. For procedures to recover your IPMP configuration by using DR when an interface fails, refer to [“Recovering an IPMP Configuration With Dynamic Reconfiguration” on page 147](#).

## IPMP Terminology and Concepts

This section introduces terms and concepts that are used throughout the IPMP chapters in this book.

active interface

Refers to an underlying interface that can be used by the system to send or receive data traffic. An interface is active if the following conditions are met:

- At least one IP address is UP in the interface. See UP address.
- The FAILED, INACTIVE, or OFFLINE flag is not set on the interface.
- The interface has not been flagged as having a duplicate hardware address.

Compare to unusable interface, INACTIVE interface.

data address

Refers to an IP address that can be used as the source or destination address for data. Data addresses are part of an IPMP group and can be used to send and receive traffic on any interface in the group. Moreover, the set of data addresses in an IPMP group can be used continuously, provided that one interface in the group is functioning. In previous IPMP implementations, data addresses were hosted on the underlying interfaces of an IPMP group. In the current implementation, data addresses are hosted on the IPMP interface.

DEPRECATED address

Refers to an IP address that cannot be used as the source address for data. Typically, IPMP test addresses, which

---

	<p>have the <code>NOFAILOVER</code> flag, are also automatically marked as <code>DEPRECATED</code> by the system. However, any address can be marked <code>DEPRECATED</code> to prevent the address from being used as a source address.</p>
dynamic reconfiguration	<p>Refers to a feature that allows you to reconfigure a system while the system is running, with little or no impact on ongoing operations. Not all Sun platforms support DR. Some Sun platforms might only support DR of certain types of hardware. On platforms that support DR of NICs, IPMP can be used for uninterrupted network access to the system during DR.</p> <p>For more information about how IPMP supports DR, refer to <a href="#">“IPMP and Dynamic Reconfiguration” on page 118</a>.</p>
explicit IPMP interface creation	<p>Applies only to the current IPMP implementation. The term refers to the method of creating an IPMP interface by using the <code>ifconfig ipmp</code> command. Explicit IPMP interface creation is the preferred method for creating IPMP groups. This method allows the IPMP interface name and IPMP group name to be set by the administrator.</p> <p>Compare to implicit IPMP interface creation.</p>
FAILBACK=no mode	<p>Refers to a setting of an underlying interface that minimizes rebinding of incoming addresses to interfaces by avoiding redistribution during interface repair. Specifically, when an interface repair is detected, the interface's <code>FAILED</code> flag is cleared. However, if the mode of the repaired interface is <code>FAILBACK=no</code>, then the <code>INACTIVE</code> flag is also set to prevent use of the interface, provided that a second functioning interface also exists. If the second interface in the IPMP group fails, then the <code>INACTIVE</code> interface is eligible to take over. While the concept of failback no longer applies in the current IPMP implementation, the name of this mode is preserved for administrative compatibility.</p>
FAILED interface	<p>Indicates an interface that the <code>in.mpathd</code> daemon has determined to be malfunctioning. The determination is achieved by either link-based or probe-based failure detection. The <code>FAILED</code> flag is set on any failed interface.</p>

failure detection	Refers to the process of detecting when a physical interface or the path from an interface to an Internet layer device no longer works. Two forms of failure detection are implemented: link-based failure detection, and probe-based failure detection.
implicit IPMP interface creation	Refers to the method of creating an IPMP interface by using the <code>ifconfig</code> command to place an underlying interface in a nonexistent IPMP group. Implicit IPMP interface creation is supported for backward compatibility with the previous IPMP implementation. Thus, this method does not provide the ability to set the IPMP interface name or IPMP group name.  Compare to explicit IPMP interface creation.
INACTIVE interface	Refers to an interface that is functioning but is not being used according to administrative policy. The <code>INACTIVE</code> flag is set on any <code>INACTIVE</code> interface.  Compare to active interface, unusable interface.
IPMP anonymous group support	Indicates an IPMP feature in which the IPMP daemon tracks the status of all network interfaces in the system, regardless of whether they belong to an IPMP group. However, if the interfaces are not actually in an IPMP group, then the addresses on these interfaces are not available in case of interface failure.
IPMP group	Refers to a set of network interfaces that are treated as interchangeable by the system in order to improve network availability and utilization. Each IPMP group has a set of data addresses that the system can associate with any set of active interfaces in the group. Use of this set of data addresses maintains network availability and improves network utilization. The administrator can select which interfaces to place into an IPMP group. However, all interfaces in the same group must share a common set of properties, such as being attached to the same link and configured with the same set of protocols (for example, IPv4 and IPv6).
IPMP group interface	See IPMP interface.
IPMP group name	Refers to the name of an IPMP group, which can be assigned with the <code>ifconfig group</code> subcommand. All underlying interfaces with the same IPMP group name are

---

	defined as part of the same IPMP group. In the current implementation, IPMP group names are de-emphasized in favor of IPMP interface names. Administrators are encouraged to use the same name for both the IPMP interface and the group.
IPMP interface	Applies only to the current IPMP implementation. The term refers to the IP interface that represents a given IPMP group, any or all of the interface's underlying interfaces, and all of the data addresses. In the current IPMP implementation, the IPMP interface is the core component for administering an IPMP group, and is used in routing tables, ARP tables, firewall rules, and so forth.
IPMP interface name	Indicates the name of an IPMP interface. This document uses the naming convention of <code>ipmpN</code> . The system also uses the same naming convention in implicit IPMP interface creation. However, the administrator can choose any name by using explicit IPMP interface creation.
IPMP singleton	Refers to an IPMP configuration that is used by Sun Cluster software that allows a data address to also act as a test address. This configuration applies, for instance, when only one interface belongs to an IPMP group.
link-based failure detection	Specifies a passive form of failure detection, in which the link status of the network card is monitored to determine an interface's status. Link-based failure detection only tests whether the link is up. This type of failure detection is not supported by all network card drivers. Link-based failure detection requires no explicit configuration and provides instantaneous detection of link failures.  Compare to probe-based failure detection.
load spreading	Refers to the process of distributing inbound or outbound traffic over a set of interfaces. Unlike load balancing, load spreading does not guarantee that the load is evenly distributed. With load spreading, higher throughput is achieved. Load spreading occurs only when the network traffic is flowing to multiple destinations that use multiple connections.  Inbound load spreading indicates the process of distributing inbound traffic across the set of interfaces in an IPMP group. Inbound load spreading cannot be

	controlled directly with IPMP. The process is indirectly manipulated by the source address selection algorithm.
	Outbound load spreading refers to the process of distributing outbound traffic across the set of interfaces in an IPMP group. Outbound load spreading is performed on a per-destination basis by the IP module, and is adjusted as necessary depending on the status and members of the interfaces in the IPMP group.
NOFAILOVER address	Applies only to the previous IPMP implementation. Refers to an address that is associated with an underlying interface and thus remains unavailable if the underlying interface fails. All NOFAILOVER addresses have the NOFAILOVER flag set. IPMP test addresses must be designated as NOFAILOVER, while IPMP data addresses must never be designated as NOFAILOVER. The concept of failover does not exist in the IPMP implementation. However, the term NOFAILOVER remains for administrative compatibility.
OFFLINE interface	Indicates an interface that has been administratively disabled from system use, usually in preparation for being removed from the system. Such interfaces have the OFFLINE flag set. The <code>if_mpadm</code> command can be used to switch an interface to an offline status.
physical interface	See: underlying interface
probe	Refers to an ICMP packet, similar to the packets that are used by the <code>ping</code> command. This probe is used to test the send and receive paths of a given interface. Probe packets are sent by the <code>in.mpathd</code> daemon, if probe-based failure detection is enabled. A probe packet uses an IPMP test address as its source address.
probe-based failure detection	Indicates an active form of failure detection, in which probes are exchanged with probe targets to determine an interface's status. When enabled, probe-based failure detection tests the entire send and receive path of each interface. However, this type of detection requires the administrator to explicitly configure each interface with a test address.  Compare to link-based failure detection.

---

probe target	Refers to a system on the same link as an interface in an IPMP group. The target is selected by the <code>in.mpathd</code> daemon to help check the status of a given interface by using probe-based failure detection. The probe target can be any host on the link that is capable of sending and receiving ICMP probes. Probe targets are usually routers. Several probe targets are usually used to insulate the failure detection logic from failures of the probe targets themselves.
source address selection	Refers to the process of selecting a data address in the IPMP group as the source address for a particular packet. Source address selection is performed by the system whenever an application has not specifically selected a source address to use. Because each data address is associated with only one hardware address, source address selection indirectly controls inbound load spreading.
STANDBY interface	Indicates an interface that has been administratively configured to be used only when another interface in the group has failed. All STANDBY interfaces will have the STANDBY flag set.
target systems	See probe target.
test address	Refers to an IP address that must be used as the source or destination address for probes, and must not be used as a source or destination address for data traffic. Test addresses are associated with an underlying interface. If an underlying interface is configured with an UP test address, the <code>in.mpathd</code> daemon monitors this address by using probe-based failure detection. All test addresses must be designated as NOFAILOVER. These addresses are also automatically marked DEPRECATED by the system to ensure that they will not be considered as a possible source address for data packets.
underlying interface	Specifies an IP interface that is part of an IPMP group and is directly associated with an actual network device. For example, if <code>ce0</code> and <code>ce1</code> are placed into IPMP group <code>ipmp0</code> , then <code>ce0</code> and <code>ce1</code> comprise the underlying interfaces of <code>ipmp0</code> . In the previous implementation, IPMP groups consist solely of underlying interfaces. However, in the current implementation, these interfaces underlie the IPMP interface (for example, <code>ipmp0</code> ) that represents the group, hence the name.

undo-offline operation	Refers to the act of administratively enabling a previously offlined interface to be used by the system. The <code>if_mpadm</code> command can be used to perform an undo-offline operation.
unusable interface	Refers to an underlying interface that cannot be used to send or receive data traffic at all in its current configuration. An unusable interface differs from an <code>INACTIVE</code> interface, that is not currently being used but can be used if an active interface in the group becomes unusable. An interface is unusable if one of the following conditions exists: <ul style="list-style-type: none"><li>▪ The interface has no UP address.</li><li>▪ The <code>FAILED</code> or <code>OFFLINE</code> flag has been set for the interface.</li><li>▪ The interface has been flagged as having the same hardware address as another interface in the group.</li></ul>
UP address	Refers to an address that has been made administratively available to the system by setting the UP flag. An address that is not UP is treated as not belonging to the system, and thus is never considered during source address selection.