Managing Oracle® Solaris 11.1 Network Performance

Part No: E28993–02
September 2013

ORACLE

130903@25097

Contents

Preface

Welcome to *Managing Oracle Solaris 11.1 Network Performance*. This book is part of the series *Establishing An Oracle Solaris 11.1 Network* that cover basic topics and procedures to configure Oracle Solaris networks. This book assumes that you have already installed Oracle Solaris. You should be ready to configure your network or ready to configure any networking software that is required on your network.

Who Should Use This Book

This book is intended for anyone responsible for administering systems that run Oracle Solaris, which are configured in a network. To use this book, you should have at least two years of UNIX system administration experience. Attending UNIX system administration training courses might be helpful.

Access to Oracle Support

Oracle customers have access to electronic support through My Oracle Support. For information, visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info` or visit `http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs` if you are hearing impaired.

Typographic Conventions

The following table describes the typographic conventions that are used in this book.

TABLE P–1 Typographic Conventions

Typeface	Description	Example
`AaBbCc123`	The names of commands, files, and directories, and onscreen computer output	Edit your `.login` file. Use `ls -a` to list all files. `machine_name% you have mail.`
`AaBbCc123`	What you type, contrasted with onscreen computer output	`machine_name%` **`su`** `Password:`

TABLE P–1 Typographic Conventions *(Continued)*

Typeface	Description	Example
aabbcc123	Placeholder: replace with a real name or value	The command to remove a file is `rm` *filename*.
AaBbCc123	Book titles, new terms, and terms to be emphasized	Read Chapter 6 in the *User's Guide*.
		A *cache* is a copy that is stored locally.
		Do *not* save the file.
		Note: Some emphasized items appear bold online.

Shell Prompts in Command Examples

The following table shows UNIX system prompts and superuser prompts for shells that are included in the Oracle Solaris OS. In command examples, the shell prompt indicates whether the command should be executed by a regular user or a user with privileges.

TABLE P–2 Shell Prompts

Shell	Prompt
Bash shell, Korn shell, and Bourne shell	`$`
Bash shell, Korn shell, and Bourne shell for superuser	`#`
C shell	`machine_name%`
C shell for superuser	`machine_name#`

◆ ◆ ◆ CHAPTER 1

Introduction to Network Performance Management

This introductory chapter presents an overview of managing the network's performance. This chapter also introduces the features described in the rest of the book that enable you to improve network performance.

Before you perform any of the configurations described in this book, you must have completed basic network configuration to connect the systems to the network. For information about basic network configuration, see *Connecting Systems Using Reactive Network Configuration in Oracle Solaris 11.1* and *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

For an overview of networking in general on Oracle Solaris 11 systems, see *Introduction to Oracle Solaris 11 Networking*

Features for Improving Network Performance

Configuring the network interface connects your systems to the network. However, other features in Oracle Solaris 11 improve the network's overall performance. Managing network performance refers to the use of these features and technologies to fine tune the way your systems process network traffic. Systems that are configured with these technologies can manage network traffic better, which contributes to the improvement of the network's total performance. These features address different areas of network operations. However, they might provide common benefits, such as the following:

- Network connectivity – Certain features such as link aggregations and IPMP ensure that a system has continuous access to the network.
- Efficiency – Some features such as load balancing enable a system to spread the load of network processing throughout available resources for greater efficiency.
- Network administration – Features such as virtual LANs (VLANs) facilitate administration. Using the other features that improve the network simplifies network administration as well.

- Cost – All of these Oracle Solaris 11 technologies enhance network performance without requiring additional expensive hardware to be purchased.

This book describes the features that improve how the system hosts and processes network traffic. Consider the following examples:

- You can configure datalinks and interfaces into a single unit. The pooled resources can be dedicated to network processing and consequently increase network throughput. See Chapter 2, "Using Link Aggregations," or Chapter 5, "Introduction to IPMP," for details.
- You can subdivide the local area network into virtual subnetworks to simplify management. See Chapter 3, "Working With VLANs," for details.
- You can enable a system to transmit packets to their destinations by using the shortest connection routes. See Chapter 4, "Administering Bridged Networks (Tasks)," for details.
- On a network with different systems and configurations, you can enable a mechanism where all peers on the network are informed of each other's configurations. Thus, network packets can be exchanged based on a negotiated set of configurations that both peers support. Peer negotiations are automatic. Therefore, manual configurations are not required. See Chapter 7, "Exchanging Network Connectivity Information With LLDP," for details.

Use of the different technologies described in this book depends on specific circumstances. Also, some hardware configurations might require you to use a specific type of feature. For example, if your system has multiple interfaces that are configured to belong to the same subnet, then you must use the IP multipathing (IPMP) technology. Thus, you do not need to complete all the configuration procedures in this book. Instead, select and deploy the technologies that address your network requirements.

Refer also to the Oracle Solaris 11.1 documentation library for other network-related system configurations that contribute to the improvement of network. For example, using virtual networks, zones, and resource management enables you to deploy multiple networks-in-a-box performance while maximizing the use of available network resources. See *Using Virtual Networks in Oracle Solaris 11.1* and *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management* for details.

◆ ◆ ◆ CHAPTER 2

Using Link Aggregations

Link aggregations enable you to pool multiple datalink resources that you administer as a single unit. By combining the resources of the multiple datalinks and dedicating them to serving the system's network operations, the system's performance is greatly improved. This chapter describes procedures that you use to configure and maintain link aggregations.

This chapter covers the following topics:

Overview of Link Aggregations

Link aggregation, also referred to as *trunking*, consists of several interfaces on a system that are configured together as a single, logical unit to increase throughput of network traffic. The following figure shows an example of a link aggregation configured on a system.

FIGURE 2–1 Link Aggregation Configuration

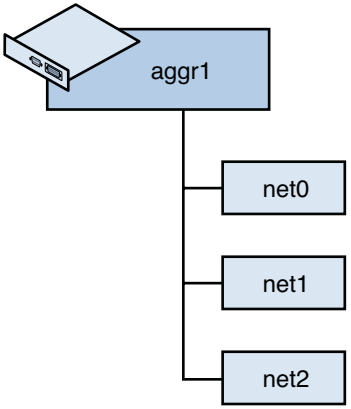


Figure 2–1 shows an aggregation `aggr1` that consists of three underlying datalinks,`net0` through `net2`. These datalinks are dedicated to serving the traffic that traverses the system through the aggregation. The underlying links are hidden from external applications. Instead, the logical datalink `aggr1` is accessible.

Link aggregation has the following features:

- *Increased bandwidth* – The capacity of multiple links is combined into one logical link.
- *Automatic failover and failback* – By supporting link-based failure detection, traffic from a failed link is failed over to other working links in the aggregation.
- *Improved administration* – All underlying links are administered as a single unit.
- *Less drain on the network address pool* – The entire aggregation can be assigned one IP address.
- *Link protection* – You can configure the datalink property that enables link protection for packets flowing through the aggregation.
- *Resource management* – Datalink properties for network resources as well as flow definitions enable you to regulate applications' use of network resources. For more information about resource management, see Chapter 3, "Managing Network Resources in Oracle Solaris," in *Using Virtual Networks in Oracle Solaris 11.1*.

Note – Link aggregations perform similar functions as IP multipathing (IPMP) to improve network performance and availability. For a comparison of these two technologies, see Appendix B, "Link Aggregations and IPMP: Feature Comparison."

Oracle Solaris supports two types of link aggregations:

- Trunk aggregations
- Datalink multipathing (DLMP) aggregations

For a quick view of the differences between these two types of link aggregations, see Appendix A, “Link Aggregation Types: Feature Comparison.”

The following sections describe each type of link aggregation in greater detail.

Trunk Aggregations

Trunk aggregation benefits a variety of networks with different traffic loads. For example, if a system in the network runs applications with distributed heavy traffic, you can dedicate a trunk aggregation to that application's traffic to avail of the increased bandwidth. For sites with limited IP address space that nevertheless require large amounts of bandwidth, you need only one IP address for a large aggregation of interfaces. For sites that need to hide the existence of internal interfaces, the IP address of the aggregation hides its interfaces from external applications.

In Oracle Solaris, trunk aggregations are configured by default when you create an aggregation. Typically, systems that are configured with link aggregations also use an external switch to connect to other systems. See the following figure.

FIGURE 2–2 Link Aggregation Using a Switch

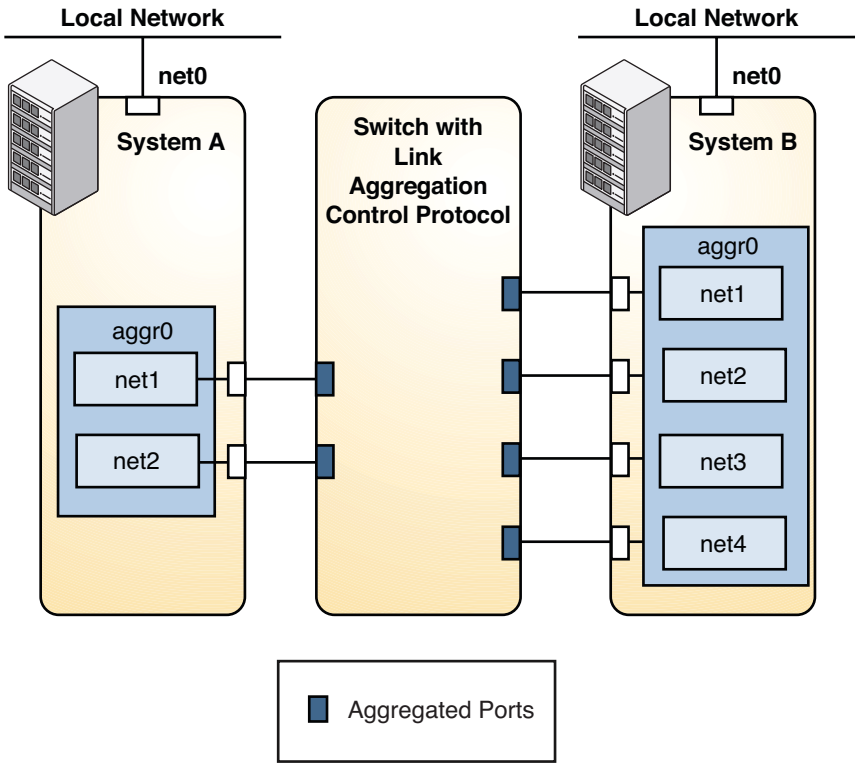


Figure 2–2 depicts a local network with two systems, and each system has an aggregation configured. The two systems are connected by a switch on which link aggregation control protocol (LACP) is configured.

System A has an aggregation that consists of two interfaces, `net1` and `net2`. These interfaces are connected to the switch through aggregated ports. System B has an aggregation of four interfaces, `net1` through `net4`. These interfaces are also connected to aggregated ports on the switch.

In this link aggregation topology, the switch must support aggregation technology. Accordingly, its switch ports must be configured to manage the traffic from the systems.

Trunk aggregations also supports back-to-back configuration. Instead of using a switch, two systems are directly connected together to run parallel aggregations, as shown in the following figure.

FIGURE 2–3 Back-to-Back Link Aggregation Configuration

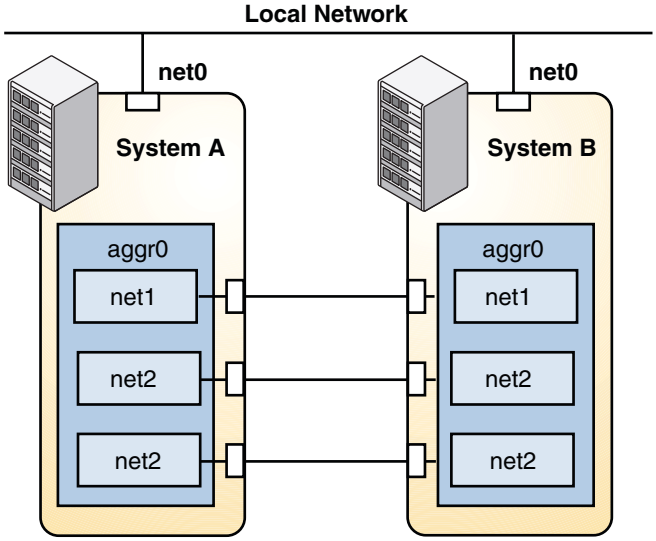


Figure 2–3 shows link aggregation `aggr0` on System A directly connected to link aggregation `aggr0` on System B by means of the corresponding links between their respective underlying datalinks. In this way, Systems A and B provide redundancy and high availability, as well as high-speed communications between both systems. Each system also has `net0` configured for traffic flow within the local network.

The most common application for back-to-back link aggregations is the configuration of mirrored database servers. Both servers must be updated together and therefore require significant bandwidth, high-speed traffic flow, and reliability. The most common use of back-to-back link aggregations is in data centers.

Note – Back-to-back configurations are not supported on DLMP aggregations.

The following sections describe other features that are unique to trunk aggregations. Do not configure these features when creating DLMP aggregations.

Policies and Load Balancing

If you plan to use a trunk aggregation, consider defining a policy for outgoing traffic. This policy can specify how you want packets to be distributed across the available links of an aggregation, thus establishing load balancing. The following are the possible layer specifiers and their significance for the aggregation policy:

- *L2* – Determines the outgoing link by hashing the MAC (L2) header of each packet
- *L3* – Determines the outgoing link by hashing the IP (L3) header of each packet

- *L4* – Determines the outgoing link by hashing the TCP, UDP, or other ULP (L4) header of each packet

Any combination of these policies is also valid. The default policy is L4.

Aggregation LACP Mode and Switches

If your setup of a trunk aggregation includes a switch, you must note whether the switch supports LACP. If the switch supports LACP, you must configure LACP for the switch and the aggregation. The aggregation's LACP can be set to one of three values:

- `off` – The default mode for aggregations. LACP packets, which are called *LACPDUs* are not generated.
- `active` – The system generates LACPDUs at regular intervals, which you can specify.
- `passive` – The system generates an LACPDU only when it receives an LACPDU from the switch. When both the aggregation and the switch are configured in passive mode, they cannot exchange LACPDUs.

Datalink Multipathing Aggregations

A trunk aggregation generally suffices for the requirements of a network setup. However, a trunk aggregation is limited to work with only one switch. Thus, the switch becomes a single point of failure for the system's aggregation. Past solutions to enable aggregations to span multiple switches present their own disadvantages:

- Solutions that are implemented on the switches are vendor-specific and not standardized. If multiple switches from different vendors are used, one vendor's solution might not apply to the products of the other vendors.
- Combining link aggregations with IP multipathing (IPMP) is very complex, especially in the context of network virtualization that involves global zones and non-global zones. The complexity increases as you scale the configurations, for example, in a scenario that includes large numbers of systems, zones, NICs, virtual NICs (VNICs), and IPMP groups. This solution would also require you to perform configurations on both the global zone and every non-global zone on every system.
- Even if a combination of link aggregation and IPMP is implemented, that configuration does not benefit from other advantages of working on the link layer alone, such as link protection, user-defined flows, and the ability to customize link properties such as bandwidth.

DLMP aggregations overcome these disadvantages. The following figure shows how a DLMP aggregation works.

FIGURE 2–4 DLMP Aggregation

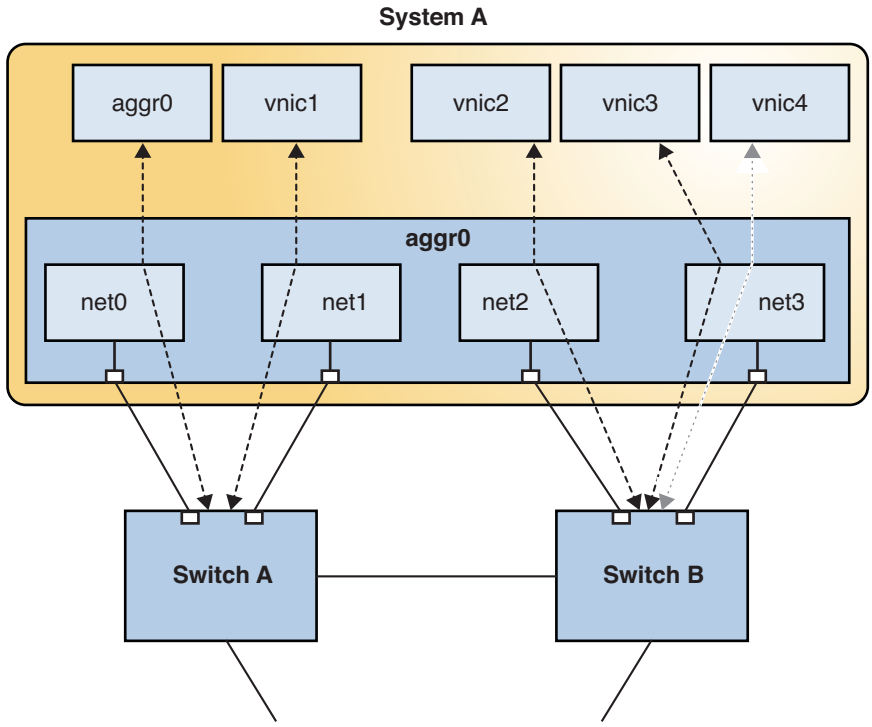


Figure 2–4 shows System A with link aggregation `aggr0`. The aggregation consists of four underlying links, from `net0` to `net3`. In addition to `aggr0`, the primary interface, VNICs are also configured over the aggregation: `vnic1` through `vnic4`. The aggregation is connected to Switch A and Switch B which, in turn, connect to other destination systems in the wider network.

In a trunk aggregation, every port is associated with every configured datalink over the aggregation. In a DLMP aggregation, a port is associated with any of the aggregation's configured datalinks as well as with the primary the interface and VNICs over that aggregation.

If the number of VNICs exceeds the number of underlying links, then an individual port is associated with multiple datalinks. As an example, Figure 2–4 shows that `vnic4` shares a port with `vnic3`.

Similarly, if an aggregation's port fails, then all the datalinks that use that port are distributed among the other ports. For example, if `net0` fails, then `aggr0` will share a port with one of the other datalinks. The distribution among the aggregation ports occurs transparently to the user and independently of the external switches connected to the aggregation.

If a switch fails, the aggregation continues to provide connectivity to its datalinks by using the other switches. A DLMP aggregation can therefore use multiple switches.

In summary, DLMP aggregations support the following features:

- The aggregation can span multiple switches.
- No switch configuration is required nor must be performed on the switches.
- You can switch between a trunk aggregation and a DLMP aggregation by using the `dladm modify-aggr` command, provided that you use only the options supported by the specific type.

Note – If you switch from a trunk aggregation to a DLMP aggregation, you must remove the switch configuration that was previously created for the trunk aggregation.

Requirements for Link Aggregations

Your link aggregation configuration is bound by the following requirements:

- No IP interface must be configured over the datalinks that will be configured into an aggregation.
- All the datalinks in the aggregation must run at the same speed and in full-duplex mode.
- For DLMP aggregations, you must have at least one switch to connect the aggregation to the ports in other systems. You cannot use a back-to-back setup when configuring DLMP aggregations.
- On SPARC based systems, each datalink must have its own unique MAC address. For instructions, refer to "How to Ensure That the MAC Address of Each Interface Is Unique" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Devices must support link state notification as defined in the IEEE 802.3ad Link Aggregation Standard in order for a port to attach to an aggregation or to detach from an aggregation. Devices that do not support link state notification can be aggregated only by using the `-f` option of the `dladm create-aggr` command. For such devices, the link state is always reported as `UP`. For information about the use of the `-f` option, see "How to Create a Link Aggregation" on page 23.

Administering Link Aggregations

This section covers different procedures for configuring and administering link aggregations. Note that some steps in the procedures are common between configuring trunk aggregations and DLMP aggregations. Steps that are unique to either type are pointed out.

- "How to Add a Link to an Aggregation" on page 27
- "How to Remove a Link From an Aggregation" on page 28
- "How to Delete a Link Aggregation" on page 28

How to Create a Link Aggregation

Before You Begin

Note – Link aggregation only works on full-duplex, point-to-point links that operate at identical speeds. Make sure that the interfaces in your aggregation conform to this requirement.

If you are using a switch in your aggregation topology and you are creating a trunk aggregation, make sure that you have done the following on the switch:

- Configured the ports to be used as an aggregation
- If the switch supports LACP, configured LACP in either active mode or passive mode

These prerequisites do not apply to DLMP aggregations.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **To determine which datalinks to aggregate, display datalink information.**

   ```
   # dladm show-link
   ```

3. **Make sure that the datalink which you are configuring into an aggregation is not opened by any application.**

   For example, if an IP interface is created over the datalink, remove the IP interface first.

   a. **To determine whether a link is being used by any application, examine the output of the `ipadm show-if` command.**

      ```
      # ipadm show-if
      IFNAME       CLASS        STATE     ACTIVE     OVER
      lo0          loopback     ok        yes        --
      net0         ip           ok        no         --
      ```

      The output indicates that an IP interface exists over the datalink `net0`.

   b. **To remove the IP interface, type the following command:**

      ```
      # ipadm delete-ip interface
      ```

      where *interface* specifies the IP interface over the link.

4. **Create a link aggregation by issuing one of the following commands:**

- To create a trunk aggregation, issue the following command:

  ```
  # dladm create-aggr [-f] [-P policy]  [-L lacpmode] \
   [-T time] [-u address] -l link1 -l link2 [...] aggr
  ```

`-f`	Forces the creation of the aggregation. Use this option when you are attempting to aggregate devices that do not support link state notification.
`-P` *policy*	Specifies the load balancing policy for the aggregation.
`-L` *lacpmode*	Specifies the mode of LACP if it is used, which can be `off`, `active`, or `passive`. See "Aggregation LACP Mode and Switches" on page 20.
`-T` *time*	Specifies the time for the LACP.
`-u` *address*	Specifies fixed unicast address for the aggregation.
`-l` *linkn*	Specifies the datalinks that you want to aggregate.
aggr	Specifies the name of the aggregation, which can be any customized name. For rules for assigning names, see "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*.

- To create a DLMP aggregation, issue the following command:

  ```
  # dladm create-aggr -m haonly -l link1 -l link2 [...] aggr
  ```

`-l` *linkn*	Specifies the datalinks that you want to aggregate.
aggr	Specifies the name of the aggregation.

5. **(Optional) Check the status of the aggregation you just created.**

   ```
   # dladm show-aggr
   ```

   The aggregation's state should be `UP`.

6. **Perform further configuration of the aggregation, such as creating IP interfaces, VNICs, and so on.**

Example 2–1 Creating a Trunk Aggregation

This example shows the commands how to create a link aggregation with two underlying datalinks, net0 and net1. The aggregation is also configured to transmit LACP packets. The example begins with the removal of existing IP interfaces over the underlying datalinks.

```
# ipadm show-if
IFNAME       CLASS        STATE     ACTIVE     OVER
lo0          loopback     ok        yes        --
net0         ip           ok        no         --
net1         ip           ok        no         --

# ipadm delete-ip net0
# ipadm delete-ip net1
```

```
# dladm create-aggr -L active -l net0 -l net1 aggr0

# dladm show-aggr
LINK    MODE      POLICY   ADDRPOLICY     LACPACTIVITY   LACPTIMER
aggr0   standard  L4       auto           on             short
```

Example 2–2 Creating a DLMP Aggregation

This example shows how to create a DLMP aggregation. The aggregation has three underlying datalinks.

```
# dladm create-aggr -m haonly -l net0 -l net1 -l net2 aggr0
# dladm show-link
LINK      CLASS     MTU     STATE   BRIDGE     OVER
net0      phys      1500    up      --         ----
net1      phys      1500    up      --         ----
net2      phys      1500    up      --         ----
aggr0     aggr      1500    up      --         net0, net1, net2

# dladm show-aggr
LINK    MODE       POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER
aggr0   haonly     --      ----            ---            ----
```

How to Switch Between Link Aggregation Types

To switch aggregation types between a trunk aggregation and a DLMP aggregation, you use the `dladm modify-aggr` command to modify the aggregation's mode. Note that switching the type of an aggregation changes the entire configuration. Thus, this procedure affects the aggregation in a more comprehensive way than simply modifying other link aggregation properties.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Determine the current type of link aggregation.**

   ```
   # dladm show-aggr
   ```

   The output's `MODE` field indicates the current type of the aggregation. The value of `MODE` is either `standard` if for a trunk aggregation, or `haonly` for a DLMP aggregation.

3. **Switch the aggregation to a DLMP aggregation.**

   ```
   # dladm modify-aggr -m mode aggr
   ```

   where *mode* is `standard` if you are switching to a trunk aggregation, or `haonly` if you are switching to a DLMP aggregation.

4 **Perform adjustments to the switch according to the requirements of the new type of link aggregation.**

5 **(Optional) Display the link aggregation configuration.**

```
# dladm show-aggr
```

Example 2–3 Switching From a Trunk Aggregation to a DLMP Aggregation

This example shows how to change an aggregation from a trunk aggregation to a DLMP aggregation.

```
# dladm show-aggr
LINK    MODE       POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER
aggr0   standard   L2      auto            active         short

# dladm modify-aggr -m haonly aggr0
# dladm show-aggr
LINK    MODE       POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER
aggr0   haonly     --      ----            -----          ----
```

Next, on the switch side, the previous switch configuration that previously applied to a trunk aggregation would be removed.

▼ How to Modify a Trunk Aggregation

This procedure shows how to modify selected attributes of a trunk aggregation only. These attributes are not supported in DLMP aggregations.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Modify the policy of the aggregation.**

```
# dladm modify-aggr -P policy aggr
```

policy Represents one or more of the policies L2, L3, and L4, as explained in "Policies and Load Balancing" on page 19.

aggr Specifies the aggregation whose policy you want to modify.

3 **Modify the LACP mode of the aggregation.**

```
# dladm modify-aggr -L lacpmode -T time aggr
```

`-L` *lacpmode* Indicates the LACP mode in which the aggregation is to run. The values are `active`, `passive`, and `off`.

`-T` *time*	Indicates the LACP timer value, either `short` or `long`.
aggr	Specifies the aggregation whose policy you want to modify.

Example 2–4 Modifying a Trunk Aggregation

This example shows how to modify the policy of link aggregation `aggr0` to L2 and then turn on active LACP mode.

```
# dladm modify-aggr -P L2 aggr0
# dladm modify-aggr -L active -T short aggr0
# dladm show-aggr
LINK    MODE       POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER
aggr0   standard   L2      auto            active         short
```

How to Add a Link to an Aggregation

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Ensure that the link you want to add has no IP interface that is plumbed over the link.**

   `# ipadm delete-ip` *interface*

   where *interface* is the IP interface configured over the datalink.

3. **Add the link to the aggregation.**

   `# dladm add-aggr -l` *link* `[-l` *link*`] [...]` *aggr*

   where *link* represents a datalink that you are adding to the aggregation and *aggr* is the name of the aggregation.

4. **(For trunk aggregations only) If the link aggregation does not have LACP configured, then reconfigure the switch to accommodate the additional datalinks.**

   Refer to the switch documentation to perform any reconfiguration tasks on the switch.

Example 2–5 Adding a Link to an Aggregation

This example shows how to add a link to the aggregation `aggr0`.

```
# dladm show-link
LINK     CLASS     MTU     STATE     BRODGE    OVER
net0     phys      1500    up        --        ----
net1     phys      1500    up        --        ----
aggr0    aggr      1500    up        --        net0, net1
```

```
net3     phys      1500    up        --       ----

# ipadm delete-ip net3
# dladm add-aggr -l net3 aggr0
# dladm show-link
LINK     CLASS     MTU     STATE     BRIDGE   OVER
net0     phys      1500    up        --       ----
net1     phys      1500    up        --       ----
aggr0    aggr      1500    up        --       net0, net1, net3
net3     phys      1500    up        --       ----
```

How to Remove a Link From an Aggregation

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Remove a link from an aggregation.

```
# dladm remove-aggr -l link aggr
```

Example 2–6 Removing a Link From an Aggregation

This example shows how to remove a link from the aggregation `aggr0`.

```
dladm show-link
LINK     CLASS     MTU     STATE     OVER
net0     phys      1500    up        --       ----
net1     phys      1500    up        --       ----
aggr0    aggr      1500    up        --       net0, net1, net3
net3     phys      1500    up        --       ----

# dladm remove-aggr -l net3 aggr0
# dladm show-link
LINK     CLASS     MTU     STATE     BRIDGE   OVER
net0     phys      1500    up        --       ----
net1     phys      1500    up        --       ----
aggr0    aggr      1500    up        --       net0, net1
net3     phys      1500    unknown   --       ----
```

How to Delete a Link Aggregation

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Delete the IP interface that is configured over the link aggregation.**

   ```
   # ipadm delete-ip IP-aggr
   ```

   where *IP-aggr* is the IP interface over the link aggregation.

3. **Delete the link aggregation.**

   ```
   # dladm delete-aggr aggr
   ```

Example 2–7 Deleting a Link Aggregation

This example shows how to delete the aggregation `aggr0`. The deletion is persistent.

```
# ipadm delete-ip aggr0
# dladm delete-aggr aggr0
```

◆ ◆ ◆ CHAPTER 3

Working With VLANs

Virtual LANs enable you to divide your network into subnetworks without having to add to the physical network environment. Thus, the subnetworks are virtual and you use the same physical network resources. VLANs facilitate network administration because the smaller groups are easier to maintain. The features and advantages of VLANs are discussed more in detail in the following sections.

This chapter describes procedures to configure and maintain virtual local area networks (VLANs). It covers the following topics:

Deploying VLANs: An Overview

A *virtual local area network (VLAN)* is a subdivision of a local area network at the datalink layer of the protocol stack. You can create VLANs for local area networks that use switch technology. By assigning groups of users to VLANs, you can improve network administration and security for the entire local network. You can also assign interfaces on the same system to different VLANs.

The following sections provide a brief overview of VLANs.

When to Use VLANs

Consider dividing your local network into VLANs if you need to do the following:

- Create a logical division of workgroups.

  For example, suppose all hosts on a floor of a building are connected on one switched-based local network. You could create a separate VLAN for each workgroup on the floor.

- Enforce differing security policies for the workgroups.

  For example, the security requirements of a Finance department and an Information Technology department are quite different. If systems for both departments share the same local network, you could create a separate VLAN for each department. Then, you could enforce the appropriate security policy on a per-VLAN basis.
- Split workgroups into manageable broadcast domains.

  The use of VLANs reduces the size of broadcast domains and improves network efficiency.

VLANs and Customized Names

VLANs demonstrate the advantage of using generic or customized names. In previous releases, the VLAN was identified by the physical point of attachment (PPA) that required combining the hardware-based name of the datalink and the VLAN ID. In Oracle Solaris 11, you can select a more meaningful name to identify the VLAN. The name must conform to the rules for naming datalinks that are provided in "Rules for Valid Link Names" in *Introduction to Oracle Solaris 11 Networking*. Thus, a VLAN can be assigned the name `sales0` or `marketing1`, for example.

VLAN names work in conjunction with VLAN IDs. Each VLAN in a local area network is identified by a VLAN ID, also known as a VLAN tag. The VLAN ID is assigned during VLAN configuration. When you configure switches to support VLANs, you need to assign a VLAN ID to each port. The VLAN ID on the port must be the same as the VLAN ID assigned to the interface that connects to the port.

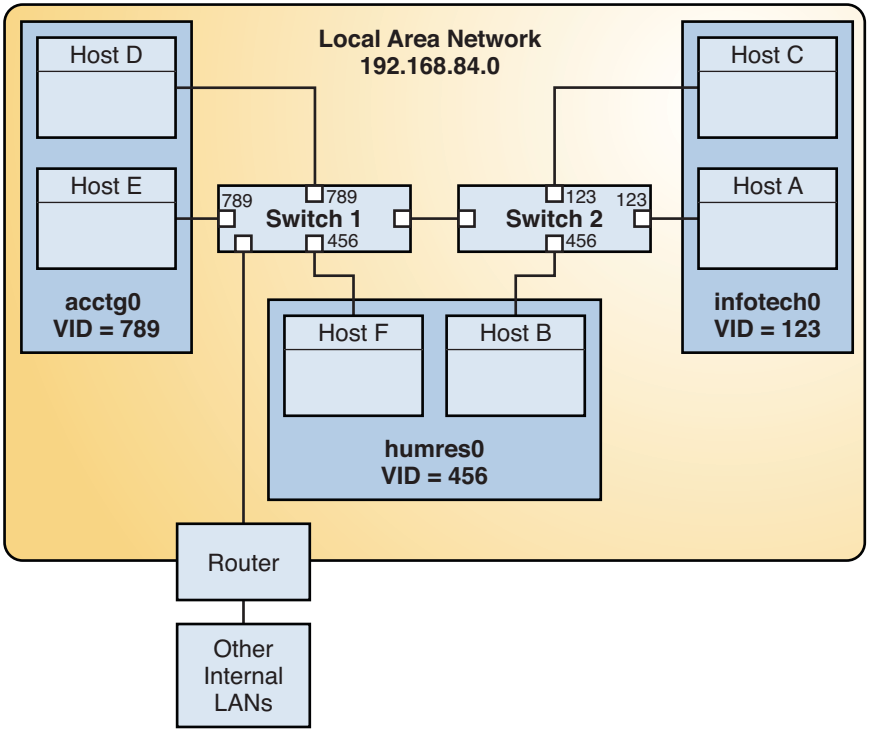
The use of customized names and VLAN IDs is shown in the following section.

VLAN Topology

Switched LAN technology enables you to organize the systems on a local network into VLANs. Before you can divide a local network into VLANs, you must obtain switches that support VLAN technology. You can configure all ports on a switch to serve a single VLAN or multiple VLANs, depending on the VLAN topology design. Each switch manufacturer has different procedures for configuring the ports on a switch.

The following figure shows a local area network that has been divided into three VLANs.

FIGURE 3–1 Local Area Network With Three VLANs

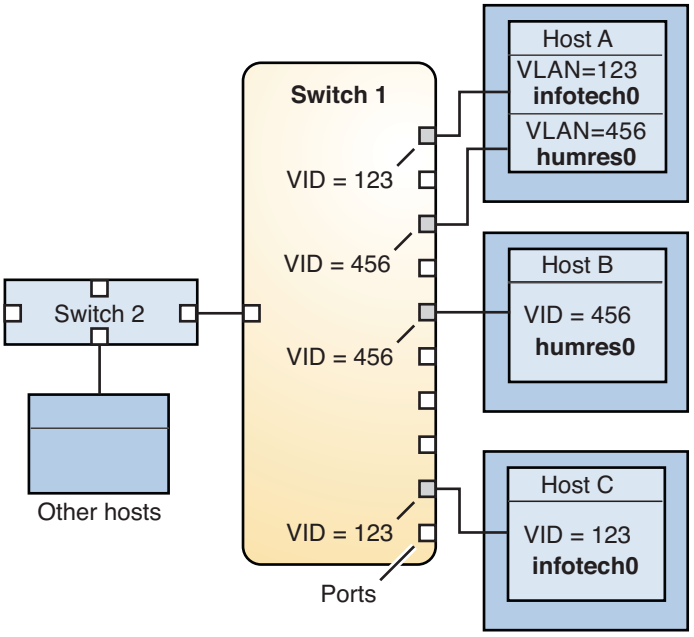


In Figure 3–1, the LAN has the subnet address `192.168.84.0`. This LAN is subdivided into three VLANs to correspond with three workgroups:

- `acctg0` with VLAN ID `789` – Accounting group. This group owns Host D and Host E.
- `humres0` with VLAN ID `456` – Human Resources group. This group owns Host B and Host F.
- `infotech0` with VLAN ID `123` – Information Technology group. This group owns Host A and Host C.

A variation of Figure 3–1 is shown in Figure 3–2 where only one switch is used, and multiple hosts belonging to different VLANs connect to that single switch.

FIGURE 3–2 Switch Configuration for a Network with VLANs



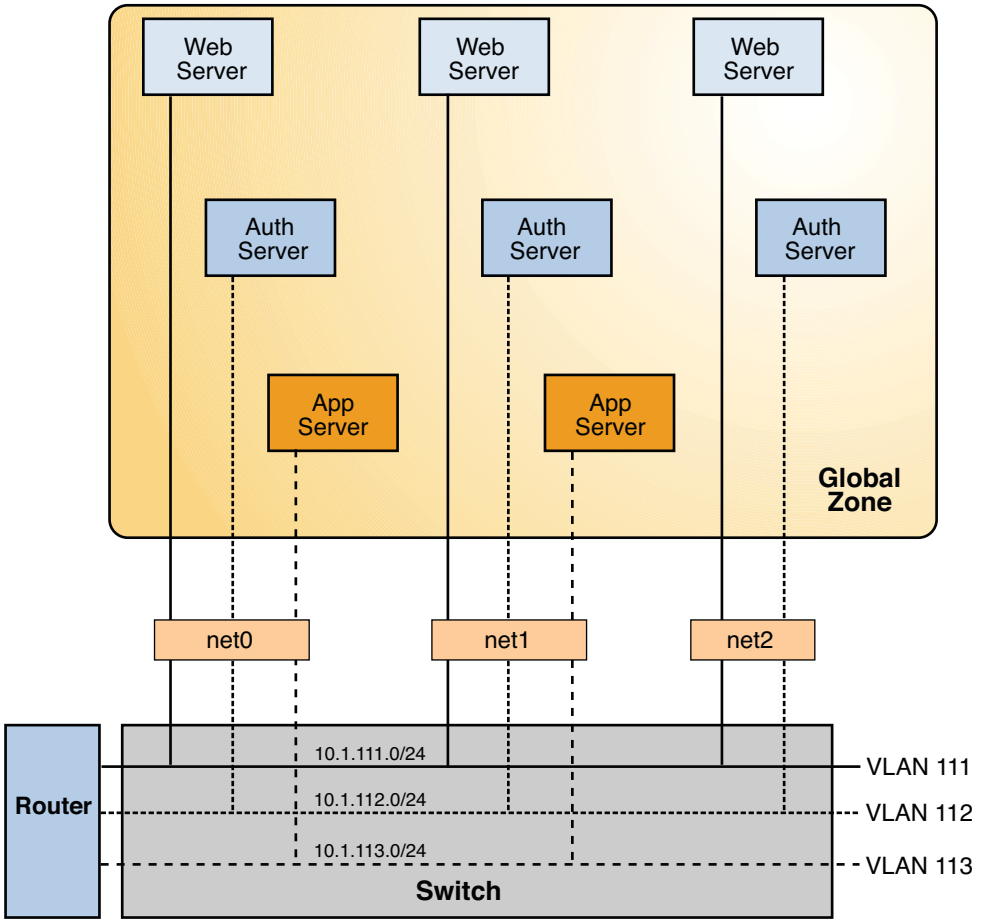
In Figure 3–2, Host A and Host C belong to the Information Technology VLAN with the VLAN ID 123. Thus, one of Host A's interface is configured with the VLAN ID 123. This interface connects to Port 1 on Switch 1, which is also configured with the VLAN ID 123. Host B is a member of the Human Resources VLAN with the VLAN ID 456. Host B's interface connects to Port 5 on Switch 1, which is configured with the VLAN ID 456. Finally, the interface of Host C is configured with the VLAN ID 123. The interface connects to Port 9 on Switch 1. Port 9 is also configured with the VLAN ID 123.

Figure 3–2 also shows that a single host can belong to multiple VLANs. For example, Host A has two VLANs configured over the host's interface. The second VLAN is configured with the VLAN ID 456 and is connected to Port 3 which is also configured with the VLAN ID 456. Thus, Host A is a member of both the `infotech0` and the `humres0` VLANs.

Using VLANs and Zones

You can configure multiple virtual networks within a single network unit such as a switch by combining VLANs and Oracle Solaris Zones. Consider the following illustration of a system with three physical network cards `net0`, `net1`, and `net2`.

FIGURE 3–3 System With Multiple VLANs



Without VLANs, you would configure different systems to perform specific functions and connect these systems to separate networks. For example, web servers would be connected to one LAN, authentication servers to another LAN, and application servers to a third LAN. With VLANs and zones, you can collapse all eight systems and configure them as zones in a single system. Then you use VLAN IDs to assign a VLAN to each set of zones that performs the same functions. The information provided in the figure can be tabulated as follows.

Function	Zone Name	VLAN Name	VLAN ID	IP Address	NIC
Web server	`webzone1`	`web1`	`111`	`10.1.111.0`	`net0`
Authentication server	`authzone1`	`auth1`	`112`	`10.1.112.0`	`net0`

Function	Zone Name	VLAN Name	VLAN ID	IP Address	NIC
Application server	appzone1	app1	113	10.1.113.0	net0
Web server	webzone2	web2	111	10.1.111.0	net1
Authentication server	authzone2	auth2	112	10.1.112.0	net1
Application server	appzone2	app2	113	10.1.113.0	net1
Web server	webzone3	web3	111	10.1.111.0	net2
Authentication server	authzone3	auth3	112	10.1.112.0	net2

To create the configuration shown in the figure, refer to Example 3–1.

Administering VLANs

This section contains procedures for configuring and administering VLANs.

▼ How to Plan a VLAN Configuration

1 **Examine the LAN topology and determine where subdivision into VLANs is appropriate.**

For a basic example of such a topology, refer to Figure 3–1.

2 **Create a numbering scheme for the VLAN IDs, and assign a VLAN ID to each VLAN.**

Note – A VLAN numbering scheme might already exist on the network. If so, you must create VLAN IDs within the existing VLAN numbering scheme.

---

3. **On each system, determine which interfaces will be components of a particular VLAN.**

   a. **Determine which interfaces are configured on the system.**

      ```
      # dladm show-link
      ```

   b. **Identify which VLAN ID will be associated with each datalink on the system.**

   c. **Create the VLAN.**

4. **Check the connections of the interfaces to the network's switches.**

   Note the VLAN ID of each interface and the switch port where each interface is connected.

5. **Configure each port on the switch with the same VLAN ID as the interface to which it is connected.**

   Refer to the switch manufacturer's documentation for configuration instructions.

▼ How to Configure a VLAN

Before You Begin This procedure assumes that the zones are already created on the system. The steps to create zones and to assign interfaces to the zones are not covered in this procedure. For more information about zone configuration, refer to Chapter 17, "Planning and Configuring Non-Global Zones (Tasks)," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Determine the types of links that are in use on the system.**

   ```
   # dladm show-link
   ```

3. **Create a VLAN link over a datalink.**

   ```
   # dladm create-vlan -l link -v vid vlan-link
   ```

   link Specifies the link on which the VLAN interface is being created.

   vid Indicates the VLAN ID number.

   vlan-link Specifies the name of the VLAN, which can also be an administratively-chosen name.

4. **Verify the VLAN configuration.**

   ```
   # dladm show-vlan
   ```

5 **Create an IP interface over the VLAN.**

```
# ipadm create-ip interface
```

where *interface* uses the VLAN name.

6 **Configure the IP interface with an IP address.**

```
# ipadm create-addr -a address interface
```

Example 3–1 Configuring a VLAN

This example shows how to create the VLAN configuration that is illustrated in Figure 3–3. This example assumes that you have already configured the different zones in the system. For more information about configuring zones, see Part II, "Oracle Solaris Zones," in *Oracle Solaris 11.1 Administration: Oracle Solaris Zones, Oracle Solaris 10 Zones, and Resource Management*.

The administrator begins by checking the available links that can be used for configuring VLANs, and then creates the VLANs over the specific links.

```
global# dladm show-link
LINK     CLASS     MTU     STATE     BRIDGE     OVER
net0     phys      1500    up        --         --
net1     phys      1500    up        --         --
net2     phys      1500    up        --         --

global# dladm create-vlan -l net0 -v 111 web1
global# dladm create-vlan -l net0 -v 112 auth1
global# dladm create-vlan -l net0 -v 113 app1
global# dladm create-vlan -l net1 -v 111 web2
global# dladm create-vlan -l net1 -v 112 auth2
global# dladm create-vlan -l net1 -v 113 app2
global# dladm create-vlan -l net2 -v 111 web3
global# dladm create-vlan -l net2 -v 112 auth3

global# dladm show-vlan
LINK      VID     OVER        FLAGS
web1      111     net0        ----
auth1     112     net0        ----
app1      113     net0        ----
web2      111     net1        ----
auth2     112     net1        ----
app2      113     net1        ----
web3      111     net2        ----
auth3     113     net2        ----
```

When link information is displayed, the VLANs are included in the list.

```
global# dladm show-link
LINK      CLASS    MTU      STATE     BRIDGE     OVER
net0      phys     1500     up        --         --
net1      phys     1500     up        --         --
net2      phys     1500     up        --         --
```

```
web1      vlan     1500     up         --         net0
auth1     vlan     1500     up         --         net0
app1      vlan     1500     up         --         net0
web2      vlan     1500     up         --         net1
auth2     vlan     1500     up         --         net1
app2      vlan     1500     up         --         net1
web3      vlan     1500     up         --         net2
auth3     vlan     1500     up         --         net2
```

Next, the administrator assigns the VLANs to their respective zones. After the VLANs have been assigned, information similar to the following would be displayed for each zone:

```
global# zonecfg -z webzone1 info net
net:
     address not specified
     physical: web1

global# zonecfg -z authzone1 info net
net:
     address not specified
     physical: auth1

global# zonecfg -z appzone2 info net
net:
     address not specified
     physical: app2
```

The value of the property `physical` indicates the VLAN that is set for the given zone.

Next, the administrator logs in to each non-global zone to configure the VLAN with an IP address.

In `webzone1`:

```
webzone1# ipadm create-ip web1
webzone1# ipadm create-addr -a 10.1.111.0/24 web1
ipadm: web1/v4
```

In `webzone2`:

```
webzone2# ipadm create-ip web2
webzone2# ipadm create-addr -a 10.1.111.0/24 web2
ipadm: web2/v4
```

In `webzone3`:

```
webzone3# ipadm create-ip web3
webzone3# ipadm create-addr -a 10.1.111.0/24 web3
ipadm: web3/v4
```

In `authzone1`:

```
authzone1# ipadm create-ip auth1
authzone1# ipadm create-addr -a 10.1.112.0/24 auth1
ipadm: auth1/v4
```

In authzone2:

```
authzone2# ipadm create-ip auth2
autzone2# ipadm create-addr -a 10.1.112.0/24 auth2
ipadm: auth2/v4
```

In authzone3:

```
authzone3# ipadm create-ip auth3
authzone3# ipadm create-addr -a 10.1.112.0/24 auth3
ipadm: auth3/v4
```

In appzone1:

```
appzone1# ipadm create-ip app1
appzone1# ipadm create-addr -a 10.1.113.0/24 app1
ipadm: app1/v4
```

In appzone2:

```
appzone2# ipadm create-ip app2
appzone2# ipadm create-addr -a 10.1.113.0/24 app2
ipadm: app2/v4
```

After all the VLANs have been configured with IP addresses, configuration is complete. The three VLANs are operative and can host traffic for their respective zones.

▼ How to Configure VLANs Over a Link Aggregation

In the same manner as configuring VLANs over an interface, you can also create VLANs on a link aggregation. Link aggregations are described in Chapter 2, "Using Link Aggregations." This section combines configuring VLANs and link aggregations.

1 List the link aggregations that are configured on the system.

```
# dladm show-link
```

2 For every VLAN that you want to create over the link aggregation you selected, issue the following command:

```
# dladm create-vlan -l link -v vid vlan-link
```

link Specifies the link on which the VLAN interface is being created. In this procedure, the link refers to the link aggregation.

vid Indicates the VLAN ID number

vlan-link Specifies the name of the VLAN, which can also be an administratively-chosen name.

3 For every VLAN that you created in the previous step, create an IP interface over the VLAN.

```
# ipadm create-ip interface
```

where *interface* uses the VLAN name.

4 For each IP interface on a VLAN, configure a valid IP address.

```
# ipadm create-addr -a address interface
```

Example 3–2 Configuring Multiple VLANs Over a Link Aggregation

In this example, two VLANs are configured on a link aggregation. The VLANs are assigned VLAN IDs 193 and 194, respectively.

```
# dladm show-link
LINK          CLASS     MTU     STATE     BRIDGE     OVER
net0      phys      1500    up        --         ----
net1      phys      1500    up        --         ----
aggr0     aggr      1500    up        --         net0, net1

# dladm create-vlan -l aggr0 -v 193 acctg0
# dladm create-vlan -l aggr0 -v 194 humres0

# ipadm create-ip acctg0
# ipadm create-ip humres0

# ipadm create-addr -a 192.168.10.0/24 acctg0
ipadm: acctg0/v4
# ipadm create-addr -a 192.168.20.0/24 humres0
ipadm: humres0/v4
```

▼ How to Configure VLANs on a Legacy Device

Certain legacy devices handle only packets whose maximum transmission unit (MTU) size, also known as frame size, is 1514 bytes. Packets whose frame sizes exceed the maximum limit are dropped. For such cases, follow the same procedure listed in "How to Configure a VLAN" on page 37. However, when creating the VLAN, use the `-f` option to force the creation of the VLAN.

1 Create the VLAN with the `-f` option.

```
# dladm create-vlan -f -l link -v vid vlan-link
```

link Specifies the link on which the VLAN interface is being created. In this procedure, the link refers to the legacy device.

vid Indicates the VLAN ID number

vlan-link Specifies the name of the VLAN, which can also be an administratively-chosen name.

2 Set a lower size for the maximum transmission unit (`MTU`), such as 1496 bytes.

```
# dladm set-linkprop -p default_mtu=1496 vlan-link
```

The lower MTU\ value allows space for the link layer to insert the VLAN header prior to transmission.

3 Perform the same step as Step 2 to set the same lower value for the MTU size of each node in the VLAN.

For more information about changing link property values, refer to "Basic dladm Commands" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Displaying VLAN Information

Because VLANs are datalinks, you can use the `dladm show-link` command to view information about VLANs. However, for information that is specific to VLANs, use the `dladm show-vlan` command.

The following example compares the type of information you obtain with either command. The first output that uses the `dladm show-link` command displays all the datalinks on the system, including those that are not VLANs. The second output that uses the `dladm show-vlan` command displays a subset of datalink information that is relevant only to VLANs.

```
# dladm show-link
LINK     CLASS    MTU      STATE     BRIDGE     OVER
net0     phys     1500     up        --         --
net1     phys     1500     up        --         --
net2     phys     1500     up        --         --
web1     vlan     1500     up        --         net0
auth1    vlan     1500     up        --         net0
app1     vlan     1500     up        --         net0
web2     vlan     1500     up        --         net1
auth2    vlan     1500     up        --         net1
app2     vlan     1500     up        --         net1
web3     vlan     1500     up        --         net2
auth3    vlan     1500     up        --         net2

# dladm show-vlan
LINK      VID     OVER     FLAGS
web1      111     net0     ----
auth1     112     net0     ----
app1      113     net0     ----
web2      111     net1     ----
auth2     112     net1     ----
app2      113     net1     ----
web3      111     net2     ----
auth3     113     net2     ----
```

Modifying VLANs

By using the `dladm modify-vlan` command, you can modify a VLAN in the following ways:

- Change a VLAN's VLAN ID
- Migrate a VLAN to another underlying link

To change the VLAN ID of a VLAN, use one of the following commands:

- `dladm modify-vlan -v` *vid* `-L` *datalink*

  In this command, *vid* specifies the new VLAN ID that you are assigning to the VLAN. *Datalink* refers to the underlying link over which the VLAN is configured. You can use this command syntax provided that only a single VLAN exists on the datalink. The command fails if you use it on a datalink that has multiple configured VLANs because VLANs on a datalink must have unique VLAN IDs.

- `dladm modify-vlan -v` *vid vlan*

  Use this command to change the unique VLAN IDs of multiple VLANs over a single datalink. Each VLAN on the datalink has a unique VLAN ID. Therefore you must change the VLAN IDs one at a time. From Figure 3–3, suppose you want to change the VLAN IDs of `web1`, `auth1`, and `app1` configured over `net0`. To change their VLAN IDs, you would proceed as follows:

  ```
  # dladm modify-vlan -v 123 web1
  # dladm modify-vlan -v 456 app1
  # dladm modify-vlan -v 789 auth1
  ```

You can migrate a VLAN from one underlying datalink to another underlying datalink without deleting and reconfiguring the VLAN. The underlying link can be a physical link, a link aggregation, or an etherstub. For more information about etherstubs, see "Components of Network Virtualization" in *Using Virtual Networks in Oracle Solaris 11.1*.

To successfully migrate a VLAN, the underlying datalink to which the VLAN is moved must be able to accommodate the datalink properties of the VLAN. If those properties are not supported, then migration fails and the user is notified. After a successful migration, all the applications that use that VLAN continue to operate normally, provided that the VLAN remains connected to the network.

Certain hardware-dependent properties might change after a VLAN migration. For example, a VLAN always shares the same MAC address as its underlying datalink. Thus, when you migrate a VLAN, the VLAN's MAC address changes to the primary MAC address of the target datalink. Other properties that might be affected are the datalink state, link speed, MTU size, and so on. However, applications continue to operate without interruption.

Note – A migrated VLAN does not retain any of its hardware lane statistics from the original datalink. Available hardware lanes for the VLAN on the target datalink become the new source of statistics information. However, software statistics that are displayed by default by the `dlstat` command are preserved.

You can perform a VLAN migration either globally or selectively. Global migration means that you migrate all the VLANs over a datalink to another datalink. To perform a global migration, you only need to specify the source datalink and the target datalink. The following example moves all the VLANs on `ether0` to `net1`:

```
# dladm modify-vlan -l net1 -L ether0
```

where

- `-L` refers to the original datalink over which the VLANs are configured.
- `-l` refers to the target datalink to which the VLANs are migrated.

Note – You must specify the target datalink before the source datalink.

To perform selective VLAN migration, you specify the VLANs that you want to move. In the following example based on Figure 3–3, VLANs are moved from `net0` to `net3`.

```
# dladm modify-vlan -l net3 web1,auth1,app1
```

Note – When migrating VLANs selectively, omit the `-L` option, which applies only to global migration.

You can change the VLAN IDs of VLANs while performing a migration. Using Figure 3–3 as the basis, the following example shows how you would migrate multiple VLANs and change their VLAN IDs at the same time.

```
# dladm show-vlan
LINK    VID      OVER     FLAGS
web1    111      net0     -----
auth1   112      net0     -----
app1    113      net0     -----

# dladm modify vlan -l net3 -v 123 web1
# dladm modify vlan -l net3 -v 456 auth1
# dladm modify vlan -l net3 -v 789 app1
# dladm show-vlan
LINK    VID      OVER     FLAGS
web1    123      net3     -----
```

```
auth1   456      net3     -----
app1    789      net3     -----
```

Note – A parallel subcommand, `dladm modify-vnic` migrates VNICs that are configured as VLANs. You must use the correct subcommand depending on whether you are migrating VLANs or VNICs that are configured as VLANs. Use the `modify-vlan` subcommand on VLANs that are displayed by the `dladm show-vlan` subcommand. Use the `modify-vnic` subcommand on VNICs, including those with VLAN IDs, that are displayed by the `dladm show-vnic` subcommand. To modify VNICs, see "Components of Network Virtualization" in *Using Virtual Networks in Oracle Solaris 11.1*.

Deleting a VLAN

Use the `dladm delete-vlan` command to delete VLAN configurations on your system.

Note – You must first delete any existing IP configurations on the VLAN that you intend to delete before you can delete the VLAN. Deleting a VLAN will fail if IP interfaces exist over the VLAN.

EXAMPLE 3–3 Deleting a VLAN Configuration

To delete a VLAN configuration, you would perform steps similar to the following example:

```
# dladm show-vlan
LINK      VID     OVER        FLAGS
web1      111     net0     ----
auth1     112     net0     ----
app1      113     net0     ----
web2      111     net1     ----
auth2     112     net1     ----
app2      113     net1     ----
web3      111     net2     ----
auth3     113     net2     ----

# ipadm delete-ip web1
# dladm delete-vlan web1
```

Use Case: Combining Link Aggregations and VLAN Configurations

This section provides an example that shows how to create a combination of network configurations that uses link aggregations and VLANs over which IP interfaces are created. Articles that present other networking scenarios can be found at http://www.oracle.com/us/sun/index.htm.

In the following example, a system that uses four NICs must be configured to be a router for eight separate subnets. To attain this objective, eight links will be configured, one for each subnet. First, a link aggregation is created on all four NICs. This untagged link becomes the default untagged subnet for the network to which the default route points.

Then VLAN interfaces are configured over the link aggregation for the other subnets. The subnets are named based on a color-coded scheme. Accordingly, the VLAN names are likewise named to correspond to their respective subnets. The final configuration consists of eight links for the eight subnets: one untagged link, and seven tagged VLAN links. The example begins with verifying whether IP interfaces already exist on the datalinks. These interfaces must be deleted before the datalinks can be combined into an aggregation.

The administrator begins by removing any IP interfaces that have been configured over the datalinks.

```
# ipadm show-if
IFNAME     CLASS      STATE   ACTIVE   OVER
lo0        loopback   ok      yes      --
net0       ip         ok      yes      --
net1       ip         ok      yes      --
net2       ip         ok      yes      --
net3       ip         ok      yes      --

# ipadm delete-ip net0
# ipadm delete-ip net1
# ipadm delete-ip net2
# ipadm delete-ip net3
```

Then the administrator creates the link aggregation `default0`.

```
# dladm create-aggr -P L2,L3 -l net0 -l net1 -l net2 -l net3 default0

# dladm show-link
LINK        CLASS      MTU  STATE    BRIDGE     OVER
net0        phys      1500  up       --         --
net1        phys      1500  up       --         --
net2        phys      1500  up       --         --
net3        phys      1500  up       --         --
default0    aggr      1500  up       --         net0 net1 net2 net3
```

Next, the administrator creates the VLANs over `default0`.

```
# dladm create-vlan -v 2 -l default0 orange0
# dladm create-vlan -v 3 -l default0 green0
# dladm create-vlan -v 4 -l default0 blue0
# dladm create-vlan -v 5 -l default0 white0
# dladm create-vlan -v 6 -l default0 yellow0
# dladm create-vlan -v 7 -l default0 red0
# dladm create-vlan -v 8 -l default0 cyan0

# dladm show-link
LINK        CLASS      MTU  STATE    BRIDGE     OVER
net0        phys      1500  up       --         --
net1        phys      1500  up       --         --
net2        phys      1500  up       --         --
net3        phys      1500  up       --         --
default0    aggr      1500  up       --         net0 net1 net2 net3
orange0     vlan      1500  up       --         default0
green0      vlan      1500  up       --         default0
blue0       vlan      1500  up       --         default0
white0      vlan      1500  up       --         default0
yellow0     vlan      1500  up       --         default0
red0        vlan      1500  up       --         default0
cyan0       vlan      1500  up       --         default0

# dladm show-vlan
LINK          VID   OVER        FLAGS
orange0         2   default0    -----
green0          3   default0    -----
blue0           4   default0    -----
white0          5   default0    -----
yellow0         6   default0    -----
red0            7   default0    -----
cyan0           8   default0    -----
```

Finally, the administrator creates IP interfaces over the VLAN links and assigns IP addresses to the interfaces.

```
# ipadm create-ip orange0
# ipadm create-ip green0
# ipadm create-ip blue0
# ipadm create-ip white0
# ipadm create-ip yellow0
# ipadm create-ip red0
# ipadm create-ip cyan0

# ipadm create-addr -a address orange0
# ipadm create-addr -a address green0
# ipadm create-addr -a address blue0
# ipadm create-addr -a address white0
# ipadm create-addr -a address yellow0
# ipadm create-addr -a address red0
# ipadm create-addr -a address cyan0
```

◆ ◆ ◆ CHAPTER 4

Administering Bridged Networks (Tasks)

This chapter describes bridged networks and how to administer them. The following topics are discussed:

Bridging Overview

Bridges are used to connect separate network segments, which are paths between two nodes. When connected by a bridge, the attached network segments communicate as if they were a single network segment. Bridging is implemented at the datalink layer (L2) of the networking stack. Bridges use a packet-forwarding mechanism to connect subnetworks together.

Although both bridging and routing can be used to distribute information about the locations of resources on the network, they differ in several ways. Routing is implemented at the IP layer (L3) and uses routing protocols. No routing protocols are used on the datalink layer. Instead, the destinations of forwarded packets are determined by examining the network traffic that is received on the links that are attached to the bridge.

When a packet is received, its source address is examined. The packet's source address associates the node from which the packet was sent to the link on which it is received. Thereafter, when a received packet uses that same address as the destination address, the bridge forwards the packet over the link to that address.

The link associated with a source address might be an intermediate link that is connected to another bridge in the bridged network. Over time, all of the bridges within the bridged network "learn" which of the links sends a packet toward a given node. Thus, the packet's destination address is used to direct the packet to its final destination by means of hop-by-hop bridging.

A local "link-down" notification indicates that all nodes on a given link are no longer reachable. In this situation, packet forwarding to the link is halted, and all forwarding entries over the link are flushed. Older forwarding entries are also flushed over time. When a link is restored, packets

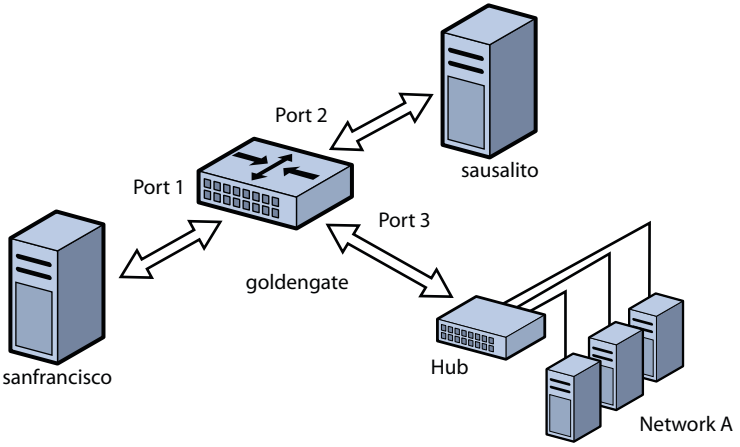
that are received over the link are treated as new. The “learning” process based on a packet's source address begins again. This process enables the bridge to properly forward packets over that link when the address is used as the destination address.

To forward packets to their destinations, bridges must listen in promiscuous mode on every link that is attached to the bridge. Listening in promiscuous mode causes bridges to become vulnerable to the occurrences of forwarding loops, in which packets circle forever at full-line rate. Thus, bridging uses the Spanning Tree Protocol (STP) mechanism to prevent network loops that would render the subnetworks unusable.

In addition to using STP and the Rapid Spanning Tree Protocol (RSTP) for bridges, Oracle Solaris supports the Transparent Interconnection of Lots of Links (TRILL) protection enhancement. STP is used by default, but you can use TRILL by specifying the `-P trill` option for the bridging commands.

Using a bridge configuration simplifies the administration of the various nodes in the network by connecting them into a single network. By connecting these segments through a bridge, all the nodes share a single broadcast network. Thus, each node can reach the other nodes by using network protocols such as IP rather than by using routers to forward traffic across network segments. If you do not use a bridge, you must configure IP routing to permit the forwarding of IP traffic between nodes.

FIGURE 4–1 Simple Bridged Network

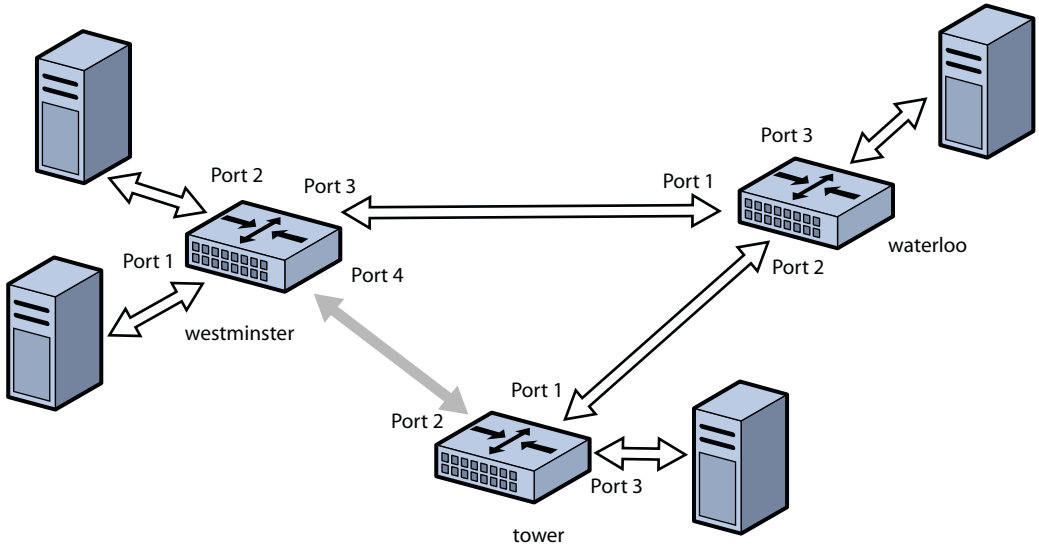


The previous figure shows a simple bridged network configuration. The bridge, `goldengate`, is an Oracle Solaris system that has bridging configured. The systems `sanfrancisco` and `sausalito` are physically connected to the bridge. Network A uses a hub that is physically connected to the bridge on one side and to computer systems on the other side. The bridge ports are links, such as `bge0`, `bge1`, and `bge2`.

Bridged networks can be formed into rings that physically connect several bridges together. Such configurations are common in networks. This type of configuration could cause problems with old packets saturating the network links by endlessly looping around the ring. To protect against such looping conditions, Oracle Solaris bridges implement both the STP and TRILL protocols. Note that most hardware bridges also implement STP loop protection.

The following figure shows a bridged network that is configured in a ring. The configuration shows three bridges. Two systems are physically connected to the `westminster` bridge. One system is physically connected to the `waterloo` bridge. And one system is physically connected to the `tower` bridge. The bridges are physically connected to each other through the bridge ports.

FIGURE 4–2 Bridged Network Ring



When STP or RSTP is used for loop protection, the physical loop is mitigated by preventing one of the connections in the loop from forwarding packets. Figure 4–2 shows that the physical link between the `westminster` and `tower` bridges is not used to forward packets.

Note that by shutting down usable physical links to perform loop protection, STP and RSTP cost you bandwidth.

Unlike STP and RSTP, TRILL does not shut down physical links to prevent loops. Instead, TRILL computes the shortest-path information for each TRILL node in the network and uses that information to forward packets to individual destinations.

As a result, TRILL enables the system to keep *all* links in use at all times. Loops are not a problem as they are handled similarly to the way that IP handles loops. Namely, TRILL creates routes as needed and uses forwarding hop limits to avoid problems that are caused by momentary loop states.

Caution – Do *not* set `local-mac-address?=false` on SPARC platforms. If you do, the systems will errantly use the same MAC address on multiple ports and on the same network.

Caution – *Do not* configure a link into a bridge when the highest possible levels of performance are required. Bridging *requires* that the underlying interfaces are in promiscuous mode, which disables a number of important optimizations that are in the hardware, driver, and other layers of the system. The disabling of these performance enhancements is an unavoidable consequence of the bridging mechanism.

You can use a bridge on a system where *some* of the system's links are not bridged and are thus not subject to those constraints. These performance issues only affect links that are configured to be part of a bridge.

For information about STP, see IEEE 802.1D-1998. For information about RSTP, see IEEE 820.1Q-2004. For information about TRILL, see the Internet Engineering Task Force (IETF) TRILL draft documents (`http://tools.ietf.org/wg/trill`).

Link Properties

The following link properties can be viewed by the `dladm show-linkprop` command, and modified by the `dladm set-linkprop` and `reset-linkprop` commands:

`default_tag` Defines the default virtual local area network (VLAN) ID for untagged packets that are sent to and from a link. Valid values are from `0` to `4094`. The default value is `1`. Only non-VLAN and non-virtual network interface card (VNIC) type links have this property. Setting this value to `0` disables the forwarding of untagged packets to and from the port. (This is a MAC property.)

Note – This property is also used outside the scope of bridging to specify the IEEE port VLAN identifier (PVID) for the link. When `default_tag` is non-zero, you cannot create a VLAN that has that same ID on the link because the base link itself automatically represents the PVID.

For example, if PVID is set to `5` on `net0`, you cannot create a VLAN with ID `5` on `net0`. To specify VLAN `5` in this situation, use `net1`.

You cannot set `default_tag` to be equal to the ID of any existing VLAN that is created on that link. For example, the following command creates VLAN `22` on `net0`:

```
# dladm create-vlan -l net0 -v 22 myvlan0
```

In this situation, you cannot set `default_tag` to `22`, which would make both `net0` and `myvlan0` represent the same VLAN.

By setting `default_tag` to `0`, you enable untagged packets on `net0` to be unassociated with any VLAN. This situation prevents such packets from being forwarded by a configured bridge.

`forward` Enables and disables traffic forwarding through the bridge. This property exists on all links except for VNIC links. Valid values are `1` (true) and `0` (false). The default value is `1`. When traffic forwarding is disabled, a VLAN that is associated with a link instance will not forward traffic through the bridge. Disabling forwarding is equivalent to removing the VLAN from the "allowed set" for a traditional bridge. This means that VLAN-based I/O to the underlying link from local clients continues, but no bridge-based forwarding is performed.

`stp` Enables and disables STP and RSTP. Valid values are `1` (true) and `0` (false). The default value is `1`, which enables STP and RSTP. When this property is set to `0`, the link does not use STP or RSTP and is placed into forwarding mode at all times. The forwarding mode uses bridge protocol data unit (BPDU) guarding. Disable STP and RSTP when you want to configure point-to-point links that are connected to end nodes. Only non-VLAN and non-VNIC type links have this property.

`stp_cost` Represents STP and RSTP cost values for using the link. Valid values are from `1` to `65535`. The default value is `0`, which is used to signal that cost is automatically computed by link type. The following values represent the cost for several link types: `100` for 10 Mbit/sec, `19` for 100 Mbit/sec, `4` for 1 Gbit/sec, and `2` for 10 Gbit/sec.

`stp_edge`	Specifies whether the port is connected to other bridges. Valid values are `1` (true) and `0` (false). The default value is `1`. If this property is set to `0`, the daemon assumes that the port is connected to other bridges even if no BPDUs of any type are detected.
`stp_p2p`	Specifies the connection mode type. Valid values are `true`, `false`, and `auto`. The default value is `auto`, which automatically discovers point-to-point connections. Specify `true` to force to point-to-point mode. Specify `false` to force normal multipoint mode.
`stp_priority`	Sets the STP and RSTP port priority value. Valid values are from `0` to `255`. The default value is `128`. The STP and RSTP port priority value is used to determine the preferred root port of a bridge by prepending the value to the PVID. The lower the numerical value is, the higher the priority.

STP Daemon

Each bridge that you create by using the `dladm create-bridge` command is represented as an identically named Service Management Facility (SMF) instance of `svc:/network/bridge`. Each instance runs a copy of the `/usr/lib/bridged` daemon, which implements the STP.

For example, the following command creates a bridge called `pontevecchio`:

```
# dladm create-bridge pontevecchio
```

The system creates an SMF service instance called `svc:/network/bridge:pontevecchio` and an observability node called `/dev/net/pontevecchio0`.

For safety purposes, all ports run standard STP by default. A bridge that does not run some form of bridging protocol, such as STP, can form long-lasting forwarding loops in the network. Because Ethernet has no hop-count or transistor-to-transistor logic (TTL) on packets, any such loops are fatal to the network.

When a particular port is not connected to another bridge (for example, because the port has a direct point-to-point connection to a host system), you can administratively disable STP for that port. Even if all ports on a bridge have STP disabled, the STP daemon still runs. The daemon continues to run for the following reasons:

- To handle any new ports that are added
- To implement BPDU guarding
- To enable or disable forwarding on the ports, as necessary

When a port has STP disabled, the `bridged` daemon continues to listen for BPDUs (BPDU guarding). The daemon uses `syslog` to flag any errors and disables forwarding on the port to

indicate a serious network misconfiguration. The link is re-enabled when the link goes down and comes up again, or when you manually remove the link and re-add it.

If you disable the SMF service instance for a bridge, bridge forwarding stops on those ports as the STP daemon is stopped. If the instance is restarted, STP starts from its initial state.

TRILL Daemon

Each bridge that you create by using the `dladm create-bridge -P trill` command is represented as an identically named SMF instance of `svc:/network/bridge` and `svc:/network/routing/trill`. Each instance of `svc:/network/routing/trill` runs a copy of the `/usr/lib/trilld` daemon, which implements the TRILL protocol.

For example, the following command creates a bridge called `bridgeofsighs`:

```
# dladm create-bridge -P trill bridgeofsighs
```

The system creates two SMF services called `svc:/network/bridge:bridgeofsighs` and `svc:/network/routing/trill:bridgeofsighs`. In addition, the system creates an observability node called `/dev/net/bridgeofsighs0`.

Debugging Bridges

Each bridge instance is assigned an *observability node*, which appears in the `/dev/net/` directory and is named with the bridge name plus a trailing `0`.

The observability node is intended for use with the `snoop` and `wireshark` utilities. This node operates like a standard Ethernet interface, except for the transmission of packets, which are silently dropped. You cannot plumb IP on top of an observability node, and you cannot perform bind requests (`DL_BIND_REQ`) unless you use the passive option.

When used, the observability node makes a single unmodified copy of every packet handled by the bridge available to the user to monitor and debug. This behavior is similar to a monitoring port on a traditional bridge and is subject to the usual data link provider interface (DLPI) promiscuous mode rules. You can use the `pfmod` command or features in the `snoop` and `wireshark` utilities to filter packets based on VLAN ID.

The delivered packets represent the data that is received by the bridge.

Note – In the cases where the bridging process adds, removes, or modifies a VLAN tag, the data shown by the `dlstat` command describes the state prior to this process taking place. This rare situation might be confusing if distinct `default_tag` values are used on different links.

To see the packets that are transmitted and received on a particular link (after the bridging process is complete), run `snoop` on the individual links rather than on the bridge observability node.

You can also obtain statistics on how network packets use network resources on links by using the `dlstat` command. For information, see Chapter 4, "Monitoring Network Traffic and Resource Usage in Oracle Solaris," in *Using Virtual Networks in Oracle Solaris 11.1*.

How Link Behavior Changes When Bridges Are Used

The following sections describe how link behavior changes when bridges are used in a network configuration.

For information about standard link behavior, see "Deploying VLANs: An Overview" on page 31.

DLPI Behavior

The following describes the differences in link behavior when a bridge is enabled:

- Link up (`DL_NOTE_LINK_UP`) and link down (`DL_NOTE_LINK_DOWN`) notifications are delivered in the aggregate. This means that when all external links are showing the link-down status, the upper-level clients that are using the MAC layers also see link-down events. When any external link on the bridge shows the link-up status, all upper-level clients see the link-up events.

  This aggregate link-up and link-down reporting is performed for the following reasons:

  - When the link-down status is seen, nodes on the link are no longer reachable. This is not true when the bridging code can still send and receive packets through another link. Administrative applications that require the actual status of links can use the existing MAC-layer kernel statistics to reveal the status. These applications are unlike ordinary clients, such as IP, in that they report hardware status information and do not get involved in forwarding packets.
  - When all external links are down, the status appears as though the bridge itself were shut down. In this special case, the system recognizes that no nodes could possibly be reachable. The trade-off is that bridges cannot be used to enable local-only communication in the case where all interfaces are "real" (not virtual), and all links are disconnected.
- All link-specific features are made generic. Links that support special hardware acceleration features are unable to use those features because actual output link determination is not made entirely by the client. The bridge forwarding function must choose an output link based on the destination MAC address, and this output link can be any link on the bridge.

Administering VLANs on Bridged Networks

By default, VLANs that are configured on the system forward packets among all the ports on a bridge instance. When you invoke the `dladm create-vlan` or `dladm create-vnic -v` command, and the underlying link is part of a bridge, the command also enables packet forwarding of the specified VLAN on that bridge link.

To configure a VLAN on a link and disable packet forwarding to or from other links on the bridge, you must disable forwarding by setting the `forward` property for the VLAN with the `dladm set-linkprop` command.

Use the `dladm create-vlan` command to automatically enable the VLAN for bridging when the underlying link is configured as part of a bridge.

VLANs are ignored by the standards-compliant STP. The bridging protocol computes just one loop-free topology by using tag-free BPDU messages, and uses this tree topology to enable and disable links. You must configure any duplicate links that are provisioned in your networks such that when those links are automatically disabled by STP, the configured VLANs are not disconnected. This means that you must either run all VLANs everywhere on your bridged backbone or carefully examine all redundant links.

The TRILL protocol does not follow the complex STP rules. Instead, TRILL automatically encapsulates packets that have the VLAN tag intact and passes them through the network. This means that TRILL binds isolated VLANs where the same VLAN ID has been reused within a single bridged network.

This important difference from STP means that you can reuse VLAN tags in isolated sections of the network to manage sets of VLANs that are larger than the 4094 limit. Although you cannot use TRILL to manage networks in this way, you might be able to implement other solutions, such as provider-based VLANs.

In an STP network with VLANs, it might be difficult to configure the failover characteristics to prevent VLAN partitioning when STP disables the “wrong” link. The relatively small loss of functionality in isolated VLANs is minor compared to the robustness of the TRILL model.

VLAN Behavior

The bridge performs packet forwarding by examining the allowed set of VLANs and the `default_tag` property for each link. The general process is as follows:

1. **Input VLAN determination.** This process begins when an incoming packet is received by the system on a link. When a packet is received, it is checked for a VLAN tag. If that tag is not present or if the tag is priority only (set to zero), the `default_tag` property value configured on that link (if not set to zero) is taken as the internal VLAN tag. If the tag is not present or zero and `default_tag` is zero, the packet is ignored. No untagged forwarding is

performed. If the tag is present and is the same as the `default_tag` value, the packet is also ignored. Otherwise, the tag is taken to be the incoming VLAN packet.

2. **Link membership check.** If the input VLAN is not configured as an allowed VLAN on this link, the packet is ignored. Forwarding is then computed, and the same check is made for the output link.
3. **Tag update.** If the VLAN tag (nonzero at this point) is the same as the `default_tag` value on the output link, the tag on the packet (if any) is removed, regardless of priority. If the VLAN tag is not the same as the `default_tag` value on the output link, a tag is added if not currently present, and the tag is set for the outgoing packet with the current priority copied into the packet.

Note – In the case where forwarding sends packets to multiple interfaces (for broadcast, multicast, and unknown destinations), the output link check and tag update must be performed independently for each output link. Some transmissions might be tagged while others are untagged.

Viewing Bridge Configurations

The following examples show how to view information about bridge configurations and bridging services:

- You can view information about bridges by running the following command:

```
# dladm show-bridge
BRIDGE      PROTECT ADDRESS            PRIORITY DESROOT
tonowhere   trill   32768/66:ca:b0:39:31:5d 32768 32768/66:ca:b0:39:31:5d
sanluisrey  stp     32768/ee:2:63:ed:41:94 32768 32768/ee:2:63:ed:41:94
pontoon     trill   32768/56:db:46:be:b9:62 32768 32768/56:db:46:be:b9:62
```

- You can view TRILL nickname information for a bridge by running the following command:

```
# dladm show-bridge -t tonowhere
NICK FLAGS LINK        NEXTHOP
38628 --   simblue2    56:db:46:be:b9:62
58753 L    --          --
```

Administering Bridges

In Oracle Solaris, you use the `dladm` command and the SMF feature to administer bridges. You can use SMF commands to enable, disable, and monitor bridge instances by using the fault-managed resource identifier (FMRI) of the instance, `svc:/network/bridge`. You can use the `dladm` command to create or destroy bridges, as well as to assign links to bridges or to remove links from them.

Administering Bridges (Task Map)

The following table points to the tasks that you can use to administer bridges.

Task	Description	For Instructions
View information about configured bridges.	Use the `dladm show-bridge` command to view information about configured bridges on the system. You can view information about configured bridges, links, statistics, and kernel forwarding entries.	"How to View Information About Configured Bridges" on page 60
View configuration information about links that are attached to a bridge.	Use the `dladm show-link` command to view information about configured links on the system. If the link is associated with a bridge, see the output in the `BRIDGE` field.	"How to View Configuration Information About Bridge Links" on page 61
Create a bridge.	Use the `dladm create-bridge` command to create a bridge and add optional links. By default, bridges are created by using STP. To instead use TRILL to create a bridge, add `-P trill` to the `dladm create-bridge` command line. Alternatively, use the `dladm modify-bridge` command to enable TRILL.	"How to Create a Bridge" on page 62
Modify the protection type for a bridge.	Use the `dladm modify-bridge` command to modify the protection type for a bridge.	"How to Modify the Protection Type for a Bridge" on page 63
Add a link to a bridge.	Use the `dladm add-bridge` command to add one or more links to an existing bridge.	"How to Add One or More Links to an Existing Bridge" on page 63
Remove links from a bridge.	Use the `dladm remove-bridge` command to remove links from a bridge. You cannot delete a bridge until all of its links are removed.	"How to Remove Links From a Bridge" on page 64
Delete a bridge from the system.	Use the `dladm delete-bridge` command to delete a bridge from the system.	"How to Delete a Bridge From the System" on page 64

How to View Information About Configured Bridges

This procedure explains how to use the `dladm show-bridge` command with various options to show different kinds of information about configured bridges.

For more information about the `dladm show-bridge` command options, see the `dladm(1M)` man page.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 View information about a bridge or all configured bridges.

- View the list of bridges.

  ```
  # dladm show-bridge
  ```

- Show the link-related status for the bridge.

  ```
  # dladm show-bridge -l bridge-name
  ```

- Show statistics for the bridge.

  ```
  # dladm show-bridge -s bridge-name
  ```

  Note – The names and definitions of the bridge statistics reported are subject to change.

- Show link-related statistics for the bridge.

  ```
  # dladm show-bridge -ls bridge-name
  ```

- Show kernel forwarding entries for the bridge.

  ```
  # dladm show-bridge -f bridge-name
  ```

- Show TRILL information about the bridge.

  ```
  # dladm show-bridge -t bridge-name
  ```

Example 4–1 Viewing Bridge Information

The following are examples of using the `dladm show-bridge` command with various options.

- The following command shows information about all bridges that are configured on the system:

  ```
  # dladm show-bridge
  BRIDGE       PROTECT ADDRESS           PRIORITY DESROOT
  goldengate   stp     32768/8:0:20:bf:f 32768    8192/0:d0:0:76:14:38
  baybridge    stp     32768/8:0:20:e5:8 32768    8192/0:d0:0:76:14:38
  ```

- The following command shows link-related status information for a single bridge instance, `tower`. To view configured properties, use the `dladm show-linkprop` command.

```
# dladm show-bridge -l tower
LINK         STATE        UPTIME   DESROOT
net0         forwarding   117      8192/0:d0:0:76:14:38
net1         forwarding   117      8192/0:d0:0:76:14:38
```

- The following command shows statistics for the specified bridge, `terabithia`:

```
# dladm show-bridge -s terabithia
BRIDGE       DROPS        FORWARDS
terabithia   0            302
```

- The following command shows statistics for all of the links on the specified bridge, `london`:

```
# dladm show-bridge -ls london
LINK         DROPS     RECV      XMIT
net0         0         360832    31797
net1         0         322311    356852
```

- The following command shows kernel forwarding entries for the specified bridge, `avignon`:

```
# dladm show-bridge -f avignon
DEST              AGE     FLAGS  OUTPUT
8:0:20:bc:a7:dc   10.860  --     net0
8:0:20:bf:f9:69   --      L      net0
8:0:20:c0:20:26   17.420  --     net0
8:0:20:e5:86:11   --      L      net1
```

- The following command shows TRILL information about the specified bridge, `key`:

```
# dladm show-bridge -t key
NICK FLAGS LINK        NEXTHOP
38628 --   london      56:db:46:be:b9:62
58753 L    --          --
```

How to View Configuration Information About Bridge Links

The `dladm show-link` output includes the `BRIDGE` field. If a link is a member of a bridge, this field identifies the name of the bridge of which it is a member. This field is shown by default. For links that are not part of a bridge, the field is blank if the `-p` option is used. Otherwise, the field shows `--`.

The bridge's observability node also appears in the `dladm show-link` output as a separate link. For this node, the existing `OVER` field lists the links that are members of the bridge.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 View configuration information about any link that is a member of a bridge.

```
# dladm show-link [-p]
```

The `-p` option produces output in a parseable format.

How to Create a Bridge

This procedure explains how to use STP to create a bridge, which is the default protocol. For more information about bridge creation options, see the description of the `dladm create-bridge` command in the `dladm(1M)` man page.

Note – To use TRILL to create a bridge, add `-P trill` to the `dladm create-bridge` command. Alternatively, use the `dladm modify-bridge` command to enable TRILL.

The `dladm create-bridge` command creates a bridge instance and optionally assigns one or more network links to the new bridge. Because no bridge instances are present on the system by default, Oracle Solaris does not create bridges between network links by default.

To create a bridge between links, you must create at least one bridge instance. Each bridge instance is separate. Bridges do not include a forwarding connection between them, and a link is a member of at most one bridge.

bridge-name is an arbitrary string that must be a legal SMF service instance name. This name is an FMRI component that has no escape sequences, which means that white space, ASCII control characters, and the following characters cannot be present:

```
; / ? : @ & = + $ , % < > # "
```

The name `default` is reserved, as are all names beginning with the `SUNW` string. Names that have trailing digits are reserved for the creation of observability devices, which are used for debugging. Because of the use of observability devices, the names of legal bridge instances are further constrained to be a legal `dlpi` name. The name must begin and end with an alphabetic character or an underscore character. The rest of the name can contain alphanumeric and underscore characters.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Create the bridge.

```
# dladm create-bridge [-l link]... bridge-name
```

The `-l` *link* option adds a link to the bridge. If any of the specified links cannot be added, the command fails and the bridge is not created.

The following example shows how to create the `brooklyn` bridge by connecting the `net0` and `net1` links:

```
# dladm create-bridge -l net0 -l net1 brooklyn
```

How to Modify the Protection Type for a Bridge

This procedure explains how to use the `dladm modify-bridge` command to modify the protection type from STP to TRILL or from TRILL to STP.

- **Modify the protection type for a bridge.**

  `# dladm modify-bridge -P` *protection-type bridge-name*

  The `-P` *protection-type* option specifies which protection type to use. By default, the protection type is STP (`-P stp`). To use the TRILL protection type, use the `-P trill` option.

  The following example shows how to modify the protection type for the `brooklyn` bridge from the default STP to TRILL:

```
# dladm modify-bridge -P trill brooklyn
```

How to Add One or More Links to an Existing Bridge

This procedure explains how to add one or more links to a bridge instance.

A link can be a member of at most one bridge. So, if you want to move a link from one bridge instance to another bridge instance, you must first remove the link from the current bridge before adding it to another bridge.

The links that are assigned to a bridge cannot be VLANs, VNICs, or tunnels. Only links that are acceptable as part of an aggregation, or links that are aggregations themselves can be assigned to a bridge.

Links that are assigned to the same bridge must have the same MTU value. Note that Oracle Solaris allows you to change the MTU value on an existing link. However, the bridge instance goes into maintenance state until you remove or change the assigned links so that the MTU values match before you restart the bridge.

The links that are assigned to the bridge must be an Ethernet type, which includes 802.3 and 802.11 media.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Add a new link to an existing bridge.**

   `# dladm add-bridge -l` *new-link bridge-name*

   The following example shows how to add the `net2` link to the existing bridge `rialto`:

```
# dladm add-bridge -l net2 rialto
```

How to Remove Links From a Bridge

This procedure explains how to remove one or more links from a bridge instance. Use this procedure if you intend to delete a bridge. Before the bridge can be deleted, all of its links must first be removed.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Remove the links from the bridge.

```
# dladm remove-bridge [-l link]... bridge-name
```

The following example shows how to remove the `net0`, `net1`, and `net2` links from the bridge `charles`:

```
# dladm remove-bridge -l net0 -l net1 -l net2 charles
```

How to Delete a Bridge From the System

This procedure explains how to delete a bridge instance. Before you can delete a bridge, you must first deactivate any attached links by running the `dladm remove-bridge` command. See "How to Remove Links From a Bridge" on page 64.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Delete the bridge from the system.

```
# dladm delete-bridge bridge-name
```

The following example shows how to remove the `net0`, `net1`, and `net2` links from the `coronado` bridge, and then delete the bridge itself from the system:

```
# dladm remove-bridge -l net0 -l net1 -l net2 coronado
# dladm delete-bridge coronado
```

◆ ◆ ◆ CHAPTER 5

Introduction to IPMP

IP network multipathing (IPMP) is a Layer 3 technology that enables you to group multiple IP interfaces into a single logical interface. With features such as failure detection, transparent access failover, and packet load spreading, IPMP improves network performance by ensuring that the network is always available to the system.

This chapter covers the following topics:

Note – Throughout the description of IPMP in this chapter and in Chapter 6, "Administering IPMP (Tasks)," 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.

IPMP in Oracle Solaris

In Oracle Solaris, IPMP contains the following features:

- IPMP enables you to configure multiple IP interfaces into a single group called an IPMP group. As a whole, the IPMP group with its multiple underlying IP interfaces is represented as a single *IPMP interface*. This interface is treated just like any other interface on the IP layer of the network 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 handles the distribution of data addresses among underlying active interfaces. 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 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 102.
- The IPMP interface can be assigned a customized name to identify the IPMP group more easily. See "Configuring IPMP Groups" on page 85.

Benefits of Using IPMP

Different factors can cause an interface to become unusable, such as an interface failure or interfaces being taken offline for maintenance. Without IPMP, 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, multiple 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.

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.

Note – Link aggregations perform similar functions as IPMP to improve network performance and availability. For a comparison of these two technologies, see Appendix B, "Link Aggregations and IPMP: Feature Comparison."

Rules for Using IPMP

The configuration of an IPMP group is determined by your system configuration.

When using IPMP, observe the following rules:

- Multiple IP interfaces on the same 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.

  Note – Multiple IPMP groups on the same link layer (L2) broadcast domain are unsupported. A L2 broadcast domain typically maps to a specific subnet. Therefore, you must configure only one IPMP group per subnet.

- 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 connect to one LAN while a single IP interface connects to the other LAN. 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 for 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 68.

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 connects to the first LAN, and a two-interface group that connects to the second LAN.

IPMP Components

The following are the IPMP software components:

- **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 used, 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 interfaces that are not configured to belong to an IPMP group. For a description of failure detection, see "Failure Detection in IPMP" on page 76.

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 how the `in.mpathd` daemon manages the activation of underlying interfaces, refer to "How IPMP Works" on page 69. For more information about the daemon, refer to the `in.mpathd(1M)` man page.

- **IP kernel module** - Manages outbound load spreading by distributing the set of available IP data addresses in the IPMP 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 module improve network traffic performance.
- **IPMP configuration file** (`/etc/default/mpathd`) - Defines the daemon's behavior.

  You customize the file to set the following parameters:

  - The target interfaces to probe when running probe-based failure detection
  - The time duration to probe a target to detect failure
  - The status with which to flag a failed interface after that interface is repaired
  - The scope of IP interfaces to monitor, whether to also include IP interfaces in the system that are not 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 101.
- `ipmpstat` **command** - Provides different types of information about the status of IPMP as a whole. The tool also displays other information about the underlying IP interfaces for each IPMP group, as well as data and test addresses that have been configured for the group. For more information about this command, see "Monitoring IPMP Information" on page 102 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.

  Note – By default, an underlying interface becomes active when you configure the interface to become part of an IPMP group.

- **Active-standby configuration** – An IPMP group in which at least one interface is administratively configured as a *standby interface*. Although idle, the standby 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-interface IPMP group is to monitor the availability of the interface by using failure detection. By configuring a test address on the interface, the multipathing daemon can track the interface by using probe-based failure detection.

Typically, a single-interface IPMP group configuration is used with other technologies that have broader failover capabilities, such as the Oracle Solaris Cluster software. The system can continue to monitor the status of the underlying interface, but the Oracle Solaris Cluster software provides the functionality to ensure availability of the network when a failure occurs. For more information about the Oracle Solaris Cluster software, see *Oracle Solaris Cluster Concepts Guide*.

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 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.

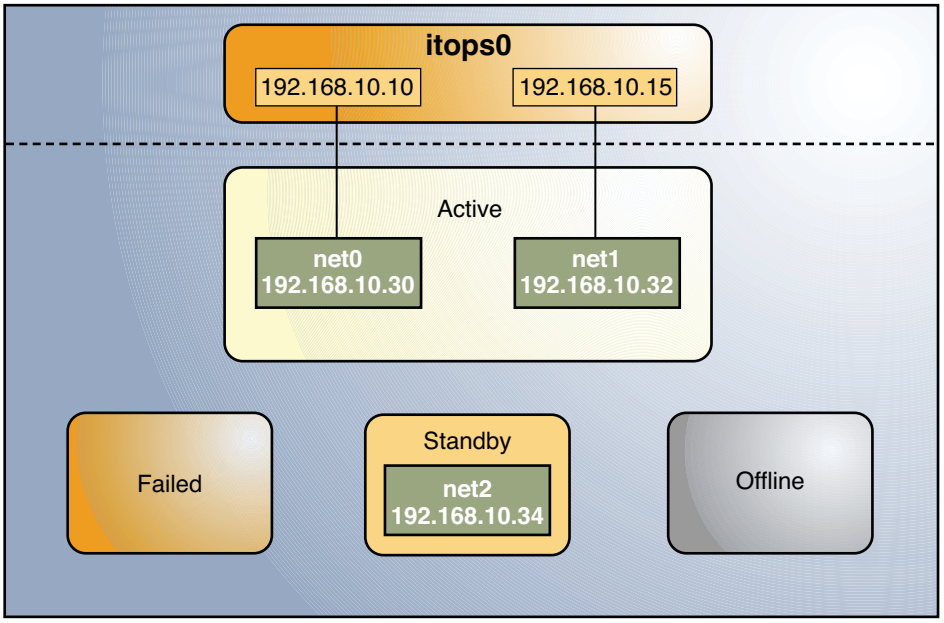
How IPMP Works

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

IPMP failure detection can be link-based, 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 the following figure..

FIGURE 5–1 IPMP Active-Standby Configuration



The IPMP 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: `net0` and `net1`.
- The group has one standby interface, also with a flexible link name: `net2`.
- Probe-based failure detection is used, and thus the active and standby interfaces are configured with test addresses, as follows:
  - `net0`: `192.168.10.30`
  - `net1`: `192.168.10.32`
  - `net2`: `192.168.10.34`

Note – The Active, Offline, Standby, and Failed areas in Figure 5–1, Figure 5–2, Figure 5–3, and Figure 5–4 indicate only the status of underlying interfaces, and not physical locations. No physical movement of interfaces or addresses, or any transfer of IP interfaces, occurs 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 102.

The following `ipmpstat` command displays information about the IPMP configuration in Figure 5–1:

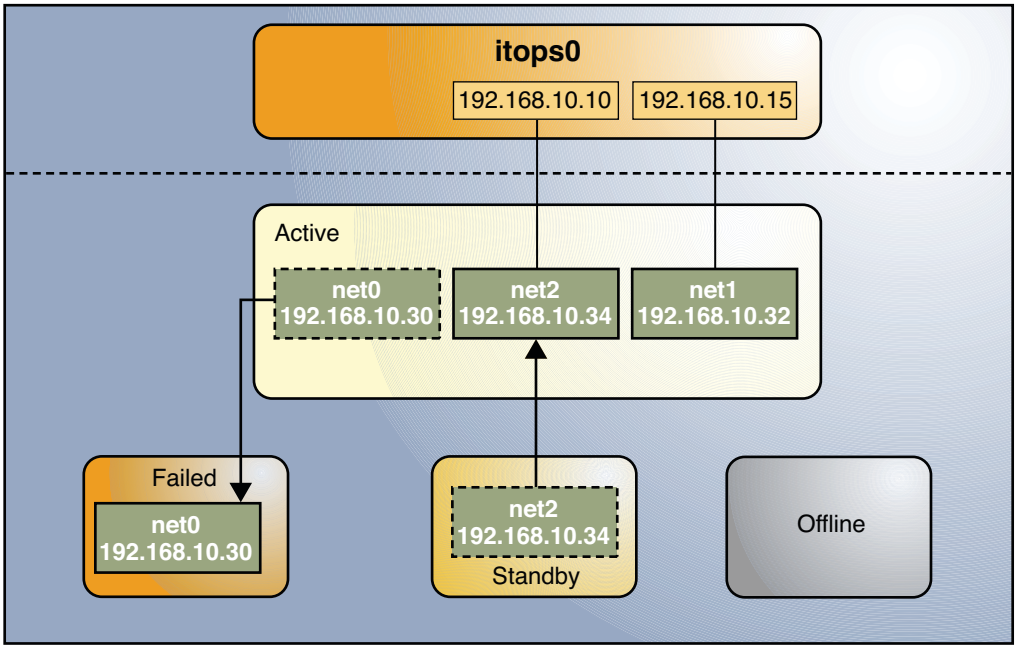
```
# ipmpstat -g
GROUP     GROUPNAME     STATE     FDT       INTERFACES
itops0    itops0        ok        10.00s    net1 net0 (net2)
```

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

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP     FLAGS      LINK        PROBE     STATE
net0        yes        itops0    -------    up          ok        ok
net1        yes        itops0    --mb---    up          ok        ok
net2        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 `net0` fails, then `net2` is deployed to ensure that the IPMP group continues to have two active interfaces. The activation of the `net2` is shown in the following figure.

FIGURE 5–2 Interface Failure in IPMP



Note – The one to one mapping of data addresses to active interfaces in Figure 5–2 serves only to simplify the illustration. The IP kernel module can randomly assign data addresses without necessarily adhering to a one to one relationship between data addresses and interfaces.

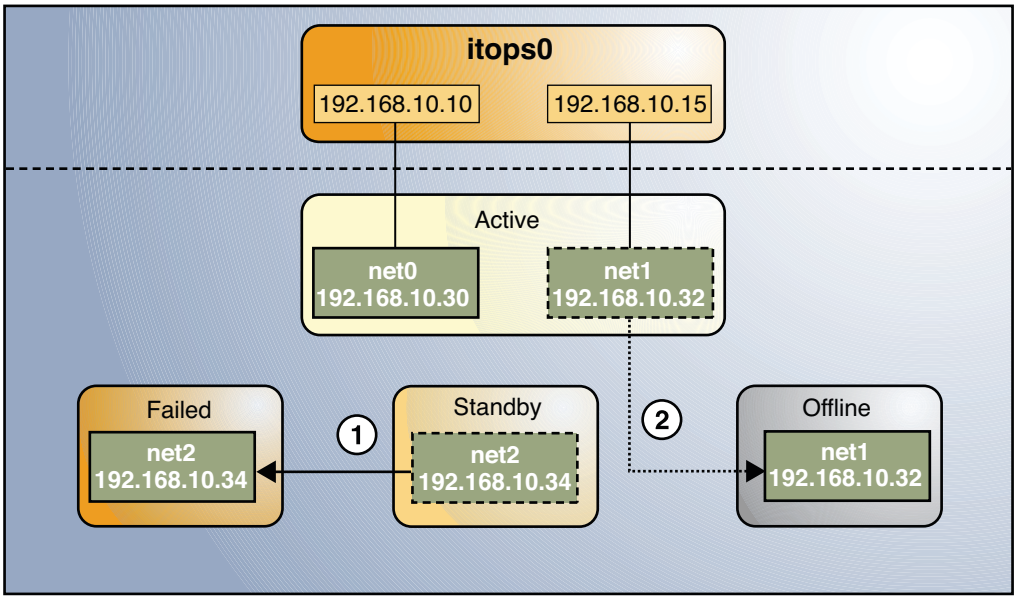
The `ipmpstat` command displays the information in Figure 5–2 as follows:

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP      FLAGS      LINK       PROBE      STATE
net0        no         itops0     -------    up         failed     failed
net1        yes        itops0     --mb---    up         ok         ok
net2        yes        itops0     -s-----    up         ok         ok
```

After `net0` is repaired, it reverts to its status as an active interface. In turn, `net2` is returned to its original standby status.

A different failure scenario is shown in Figure 5–3, where the standby interface `net2` fails (1). Later, one active interface, `net1`, is taken offline by the administrator (2). The result is that the IPMP group is left with a single functioning interface, `net0`.

FIGURE 5–3 Standby Interface Failure in IPMP

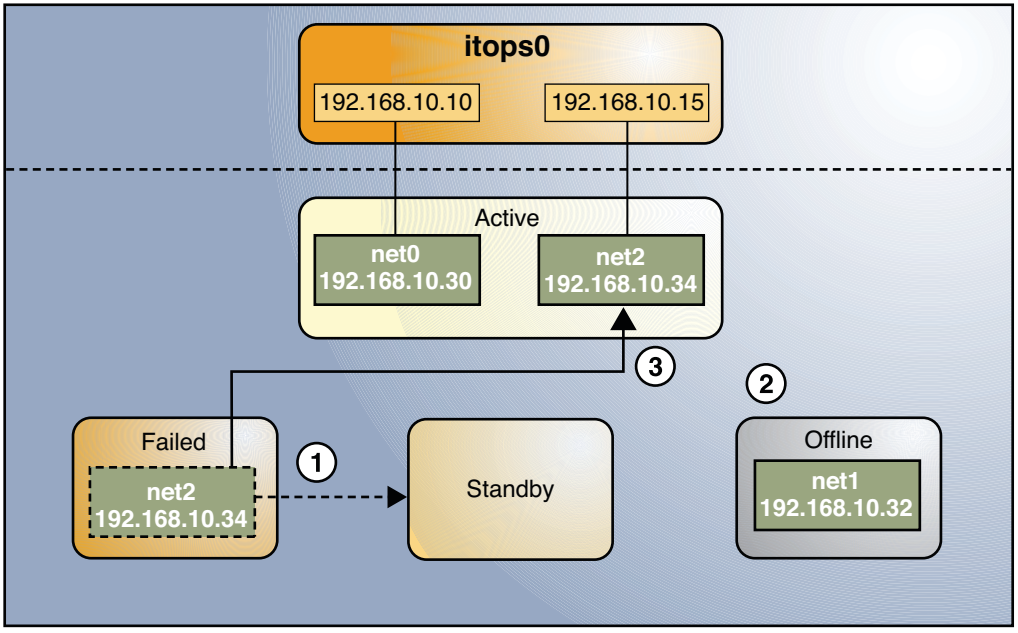


The `ipmpstat` command displays the information in Figure 5–3 as follows:

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP     FLAGS       LINK        PROBE     STATE
net0        yes        itops0    -------     up          ok        ok
net1        no         itops0    --mb-d-     up          ok        offline
net2        no         itops0    is-----     up          failed    failed
```

For this particular failure, the recovery after an interface is repaired operates differently. The recovery process 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 the following figure:

FIGURE 5–4 IPMP Recovery Process



In Figure 5–4, when `net2` is repaired, it would normally revert to its original status as a standby interface (1). However, the IPMP group would still not reflect the original number of two active interfaces because `net1` continues to remain offline (2). Thus, IPMP instead deploys `net2` as an active interface (3).

The `ipmpstat` command displays the post-repair IPMP scenario as follows:

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP     FLAGS       LINK        PROBE     STATE
net0        yes        itops0    -------     up          ok        ok
net1        no         itops0    --mb-d-     up          ok        offline
net2        yes        itops0    -s-----     up          ok        ok
```

A similar recovery process 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 that `net0` in Figure 5–2 is configured in `FAILBACK=no` mode. In that mode, a repaired `net0` is becomes a standby interface, even though it was originally an active interface. The interface `net2` remains active to maintain the IPMP group's original number of two active interfaces.

The `ipmpstat` command displays the recovery information as follows:

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP     FLAGS       LINK        PROBE     STATE
net0        no         itops0    i------     up          ok        ok
```

```
net1        yes         itops0    --mb---    up          ok        ok
net2        yes         itops0    -s-----    up          ok        ok
```

For more information about this type of configuration, see "FAILBACK=no Mode" on page 80.

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, which are described in the following sections. Starting with Oracle Solaris 11, IP addresses reside on the IPMP interface *only* and are specified as data addresses, while test addresses reside on the underlying interfaces.

Data Addresses

Data addresses are the conventional IPv4 and IPv6 addresses that are dynamically assigned to an IP interface at boot time by the DHCP server or manually by using the `ipadm` 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 uses the data addresses that are hosted on the IPMP interface and flow through the active interfaces of that IPMP interface or group.

Test Addresses

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 `ipadm` command. Only test addresses are assigned to the underlying interfaces of the IPMP group. 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 want to use probe-based failure detection. Otherwise, you can enable transitive probing to detect failure without using test addresses. For more information about probe-based failure detection with or without using test addresses, refer to "Probe-Based Failure Detection" on page 76.

In previous IPMP implementations, test addresses had 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 are not considered a possible source for data packets, the system automatically marks any addresses with the `NOFAILOVER` flag as `DEPRECATED`.

You can use any IPv4 address on your subnet as a test address. 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`).

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 the `ipadm` command.

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 as test addresses.

Failure 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 they are repaired. Remaining active interfaces continue to function while any existing standby interfaces are deployed as needed.

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

- Probe-based failure detection, of two types:
  - No test addresses are configured (transitive probing).
  - Test addresses are configured.
- Link-based failure detection, if supported by the NIC driver

Probe-Based Failure Detection

Probe-based failure detection consists of using ICMP probes to check whether an interface has failed. The implementation of this failure detection method depends on whether test addresses are used.

Probe-Based Failure Detection Using Test Addresses

This failure detection method involves sending and receiving ICMP probe messages that use test addresses. These messages, also called *probe traffic* or *test traffic*, are sent over the interface

to one or more target systems on the same local network. The `in.mpathd` 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 FDT in the IPMP configuration file. For instructions, go to "How to Configure the Behavior of the IPMP Daemon" on page 101.

To optimize probe-based failure detection, you must set multiple target systems to receive the probes from the `in.mpathd` 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.

The `in.mpathd` daemon determines which target systems to probe dynamically. First, the daemon searches the routing table for target systems 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 the daemon 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 the daemon cannot find routers or hosts that responded to the multicast probes, then the daemon cannot detect probe-based failures. In this case, the `ipmpstat -i` command reports the probe state as `unknown`.

You can use host routes to explicitly configure a list of target systems to be used by the `in.mpathd` daemon. For instructions, refer to "Configuring Probe-Based Failure Detection" on page 98.

Probe-Based Failure Detection Without Using Test Addresses

With no test addresses, this method is implemented by using two types of probes:

- **ICMP probes**

  ICMP probes are sent by the active interfaces in the IPMP group to probe targets that are defined in the routing table. An *active* interface is an underlying interface that can receive inbound IP packets that are addressed to the interface's link layer (L2) address. The ICMP probe uses the data address as the probe's source address. If the ICMP probe reaches its target and gets a response from the target, then the active interface is operational.

- **Transitive probes**

  Transitive probes are sent by the alternate interfaces in the IPMP group to probe the active interface. An alternate interface is an underlying interface that does not actively receive any inbound IP packets.

For example, consider an IPMP group that consists of four underlying interfaces. The group is configured with one data address but no test addresses. In this configuration, outbound packets can use all the underlying interfaces. However, inbound packets can only be received by the interface to which the data address is bound. The remaining three underlying interfaces that cannot receive inbound packets are the *alternate* interfaces.

If an alternate interface can successfully send a probe to an active interface and receive a response, then the active interface is functional, and by inference, so is the alternate interface that sent the probe.

Note – In Oracle Solaris, probe-based failure detection operates with test addresses. To select probe-based failure detection without test addresses, you must manually enable transitive probing. For the procedures, see "How to Select Which Failure Detection Method to Use" on page 99.

Group Failure

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.

In an IPMP group that has no test addresses, a single interface that can probe the active interface is designated as a prober. This designated interface has both the `FAILED` flag and `PROBER` flag set. The data address is bound to this interface, which enables the interface to continue probing the target to detect recovery.

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 in 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.

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.

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 `ipmpstat -g` command, the anonymous group is displayed as double-dashes (`--`). In anonymous groups, the interfaces have their data addresses also function as test addresses. Because these interfaces do not belong to a named IPMP group, then these addresses are visible to applications. To enable the tracking of interfaces that are not part of an IPMP group, see "How to Configure the Behavior of the IPMP Daemon" on page 101.

Detecting Physical Interface Repairs

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 marked with the `RUNNING` flag again and the failure detection method has detected the interface as repaired, the `in.mpathd` daemon clears the interface's `FAILED` flag. The repaired interface is redeployed depending on the number of active interfaces that the administrator originally set.

When an underlying interface fails and probe-based failure detection is used, the `in.mpathd` daemon continues probing, either by means of the designated prober when no test addresses are configured or by using the interface's test address. During an interface repair, the recovery process proceeds depending on the original configuration of the failed interface as follows:

- If the 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 IPMP group as defined by the system administrator.

  Note – An exception is when the repaired active interface is also configured with the `FAILBACK=no` mode. For more information, see "`FAILBACK=no` Mode" on page 80

- If the 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 becomes an active interface.

To see a graphical presentation of how IPMP operates during interface failure and repair, see "How IPMP Works" on page 69.

FAILBACK=no Mode

By default, active interfaces that have failed and then been repaired automatically return to become active interfaces in the IPMP group. This behavior is controlled by the value of the `FAILBACK` parameter in the `in.mpathd` 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. These administrators might prefer to enable an activated standby interface to continue as an active interface. IPMP allows administrators to override the default behavior to prevent an interface from automatically becoming 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 101.

When an active interface in `FAILBACK=no` mode fails and is subsequently repaired, the `in.mpathd` 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

The dynamic reconfiguration (DR) feature of Oracle Solaris enables you to reconfigure system hardware, such as interfaces, while the system is running. DR can be used only on systems that support this feature. On systems that support DR, 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. For example, you can attach, plumb, and then add new interfaces to existing IPMP groups. After these interfaces have been configured, they are immediately available for use by IPMP.

All requests to detach NICs are first checked to ensure that connectivity can be preserved. For example, 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 the `cfgadm` command, as explained in the `cfgadm(1M)` man page.

If the checks are successful, the `in.mpathd` 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 is displayed. Otherwise, the detach request completes successfully. You can remove the component from the system. No existing connections are disrupted.

Note – When replacing NICs, make sure that the cards are of the same type, such as Ethernet. After the NIC is replaced, then the persistent IP interface configurations are applied to that NIC.

◆ ◆ ◆ CHAPTER 6

Administering IPMP (Tasks)

This chapter provides tasks for administering interface groups with IP network multipathing (IPMP) in the Oracle Solaris 11 release.

This chapter covers the following topics:

Note – The tasks in this chapter describe how to configure IPMP using the `ipadm` command, which replaces the `ifconfig` command that is used in Oracle Solaris 10 and previous releases. To find out more about how these two commands map to each other, see Appendix A, "Comparison Map: ifconfig and ipadm Commands," in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Oracle Solaris 11 also introduces a new conceptual model for IPMP. For a detailed explanation, see "How IPMP Works" on page 69.

Maintaining Routing While Deploying IPMP

When you configure an IPMP group, the IPMP interface inherits the IP addresses of its underlying interfaces to use them as data addresses. The underlying interfaces then receive the IP address `0.0.0.0`. Consequently, routes that are defined using specific IP interfaces become lost if these interfaces are subsequently added to an IPMP group.

Loss of routing when configuring IPMP commonly involves the default route and occurs in association with an Oracle Solaris installation. During installation, you are required to define a default route, for which you use an interface in the system, such as the primary interface.

Subsequently, you configure an IPMP group by using the same interface on which you defined the default route. After the IPMP configuration, the system can no longer route network packets because the interface's address has been transferred to the IPMP interface.

To ensure that the default route is preserved while you use IPMP, the route must be defined without specifying the interface. In this manner, any interface, including the IPMP interface, can be used for routing. Thus the system can continue to route traffic.

Note – This section uses the primary interface as an example on which the default route is defined. However, the routing loss case applies to any interface that is used for routing and which later becomes part of an IPMP group.

▼ How to Define Routes While Using IPMP

The following procedure explains how to preserve the default route when you configure IPMP.

1. **Log in to the system by using a console.**

   You must use the console to perform this procedure. If you use the `ssh` or `telnet` command to log in, the connection will be lost when you perform the subsequent steps.

2. **(Optional) Display the routes that are defined in the routing table.**

   ```
   # netstat -nr
   ```

3. **Delete the route that is bound to the specific interface.**

   ```
   # route -p delete default gateway-address -ifp interface
   ```

4. **Add the route without specifying an interface.**

   ```
   # route -p add default gateway-address
   ```

5. **(Optional) Display the redefined routes.**

   ```
   # netstat -nr
   ```

6. **(Optional) If the information has not changed in the routing table, restart the routing service, then recheck the information in the routing table to make sure the routes have been correctly redefined.**

   ```
   # svcadm restart routing-setup
   ```

Example 6–1 Defining Routes for IPMP

This example assumes that the default route was defined for `net0` during the installation.

```
# netstat -nr
Routing Table: IPv4
 Destination     Gateway       Flags    Ref     Use       Interface
```

```
-------------  ------------  --------  -----  -----------  --------
default        10.153.125.1     UG      107    176682262   net0
10.153.125.0   10.153.125.222   U        22    137738792   net0

# route -p delete default 10.153.125.1 -ifp net0
# route -p add default 10.153.125.1

# netstat -nr
Routing Table: IPv4
 Destination     Gateway       Flags    Ref     Use       Interface
-------------  ------------  --------  -----  -----------  --------
default        10.153.125.1     UG      107    176682262
10.153.125.0   10.153.125.222   U        22    137738792   net0
```

Configuring IPMP Groups

This section provides procedures for planning and configuring IPMP groups. The overview in Chapter 5, "Introduction to IPMP," describes the implementation of an IPMP group as an interface. Thus, in this chapter, the terms *IPMP group* and *IPMP interface* are used interchangeably.

▼ How to Plan an IPMP Group

The following procedure includes the required planning tasks and information to be gathered prior to configuring an IPMP group. The tasks do not have to be performed in sequence.

Note – You must configure only one IPMP group for each subnet or L2 broadcast domain. For more information, see "Rules for Using IPMP" on page 67.

---

1 Determine the general IPMP configuration that would suit your needs.

Your IPMP configuration depends on the network requirements to handle the type of traffic that is hosted on your system. IPMP spreads outbound network packets across the IPMP group's interfaces and thus improves network throughput. However, for a given TCP connection, inbound traffic normally follows only one physical path to minimize the risk of processing out-of-order packets.

Thus, if your network handles a huge volume of outbound traffic, configuring many interfaces into an IPMP group can improve network performance. If instead the system hosts heavy inbound traffic, then having a large number of interfaces in the group does not necessarily improve performance by load-spreading traffic. However, having more underlying interfaces helps to guarantee network availability during interface failure.

2 **Verify that each interface in the group has a unique MAC address.**

To configure a unique MAC address for each interface on the system, see "How to Ensure That the MAC Address of Each Interface Is Unique" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

3 **Ensure that the same set of STREAMS modules is configured and pushed on all interfaces in the IPMP group.**

All interfaces in the same group must have the same STREAMS modules configured in the same order.

a. **Check the order of STREAMS modules on all interfaces in the prospective IPMP group.**

You can print a list of STREAMS modules by using the `ifconfig` *interface* `modlist` command. For example, here is the `ifconfig` output for a `net0` interface:

```
# ifconfig net0 modlist
    0 arp
    1 ip
    2 e1000g
```

As the output shows, interfaces normally exist as network drivers directly below the IP module. These interfaces do not require additional configuration.

However, certain technologies are pushed as STREAMS modules between the IP module and the network driver. If a STREAMS module is stateful, then unexpected behavior can occur on failover, even if you push the same module to all of the interfaces in a group. However, you can use stateless STREAMS modules, provided that you push them in the same order on all interfaces in the IPMP group.

b. **Push the modules of each interface in the standard order for the IPMP group.**

For example:

```
# ifconfig net0 modinsert vpnmod@3
```

4 **Use the same IP addressing format on all interfaces in the IPMP group.**

If one interface is configured for IPv4, then all interfaces in the IPMP group must be configured for IPv4. For example, if you add IPv6 addressing to one interface, then all interfaces in the IPMP group must be configured for IPv6 support.

5 **Determine the type of failure detection that you want to implement.**

For example, if you want to implement probe-based failure detection, then you must configure test addresses on the underlying interfaces. For related information, see "Failure Detection in IPMP" on page 76.

6 **Ensure that all interfaces in the IPMP group are connected to the same local network.**

For example, you can configure Ethernet switches on the same IP subnet into an IPMP group. You can configure any number of interfaces into an IPMP group.

Note – You can also configure a single-interface IPMP group, for example, if your system has only one physical interface. For related information, see "Types of IPMP Interface Configurations" on page 68.

7 **Ensure that the IPMP group does not contain interfaces with different network media types.**

The interfaces that are grouped together must be of the same interface type. For example, you cannot combine Ethernet and Token Ring interfaces in an IPMP group. As another example, you cannot combine a Token bus interface with asynchronous transfer mode (ATM) interfaces in the same IPMP group.

8 **For IPMP with ATM interfaces, configure the ATM interfaces in LAN emulation mode.**

IPMP is not supported for interfaces using Classical IP over ATM technology as defined in IETF RFC 1577 (`http://datatracker.ietf.org/doc/rfc1577/`)and IETF RFC 2225 (`http://datatracker.ietf.org/doc/rfc2225/`).

▼ How to Configure an IPMP Group That Uses DHCP

A multiple-interfaced IPMP group can be configured with active-active interfaces or active-standby interfaces. For related information, see "Types of IPMP Interface Configurations" on page 68. The following procedure describes how to configure an active-standby IPMP group by using DHCP.

Before You Begin Make sure that IP interfaces that will be in the prospective IPMP group have been correctly configured over the system's network datalinks. For procedures to configure links and IP interfaces, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*. You can create an IPMP interface even if underlying IP interfaces have not yet been created. However, without created underlying IP interfaces, subsequent configurations on the IPMP interface will fail.

Additionally, if you are using a SPARC based system, configure a unique MAC address for each interface. For the procedures, see "How to Ensure That the MAC Address of Each Interface Is Unique" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

Finally, if you are using DHCP, make sure that the underlying interfaces have infinite leases. Otherwise, if an IPMP group failure occurs, the test addresses will expire and the `in.mpathd` daemon will then disable probe-based failure detection and link-based failure detection will be used. If link-based failure detection discovers that the interface is functioning, the daemon might erroneously report that the interface has been repaired. For more information about configuring DHCP, refer to *Working With DHCP in Oracle Solaris 11.1*.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Create an IPMP interface.**

   ```
   # ipadm create-ipmp ipmp-interface
   ```

   where *ipmp-interface* specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, for example, `ipmp0`.

3. **Create the underlying IP interfaces if they do not yet exist.**

   ```
   # ipadm create-ip under-interface
   ```

   where *under-interface* refers to the IP interface that you will add to the IPMP group.

4. **Add underlying IP interfaces that will contain test addresses to the IPMP group.**

   ```
   # ipadm add-ipmp -i under-interface1 [-i under-interface2 ...] ipmp-interface
   ```

   You can add as many IP interfaces to the IPMP group as are available on the system.

5. **Have DHCP configure and manage the data addresses on the IPMP interface.**

   ```
   # ipadm create-addr -T dhcp ipmp-interface
   ```

   Step 5 associates the address provided by the DHCP server with an address object. The address object uniquely identifies the IP address by using the format *interface*/*address-type*, for example, `ipmp0/v4`. For more information about the address object, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*

6. **If you use probe-based failure detection with test addresses, have DHCP manage the test addresses on the underlying interfaces.**

   Issue the following command for each underlying interface of the IPMP group.

   ```
   # ipadm create-addr -T dhcp under-interface
   ```

   The address object that is automatically created by Step 6 uses the format *under-interface*/*address-type*, for example, `net0/v4`.

Example 6–2 Configuring an IPMP Group With DHCP

This example shows how to configure an active-standby IPMP group with DHCP and is based on the following scenario:

- Three underlying interfaces `net0`, `net1`, and `net2` are configured into an IPMP group.
- The IPMP interface `ipmp0` shares the same name with the IPMP group.
- `net2` is the designated standby interface.
- All the underlying interfaces are assigned test addresses.

First, the administrator creates the IPMP interface.

```
# ipadm create-ipmp ipmp0
```

Next, the administrator creates the underlying IP interfaces and adds them to the IPMP interface.

```
# ipadm create-ip net0
# ipadm create-ip net1
# ipadm create-ip net2

# ipadm add-ipmp -i net0 -i net1 -i net2 ipmp0
```

Next, the administrator assigns DHCP-managed IP addresses to the IPMP interface. IP addresses assigned to the IPMP interface are data addresses. In this example, the IPMP interface has two data addresses.

```
# ipadm create-addr -T dhcp ipmp0
ipadm: ipmp0/v4
# ipadm create-addr -T dhcp ipmp0
ipadm: ipmp0/v4a
```

Next, the administrator assigns DHCP-managed IP addresses to the underlying IP interfaces of the IPMP group. IP addresses assigned to the underlying interfaces are test addresses to be used for probe-based failure detection.

```
# ipadm create-addr -T dhcp net0
ipadm: net0/v4
# ipadm create-addr -T dhcp net1
ipadm: net1/v4
# ipadm create-addr -T dhcp net2
ipadm net2/v4
```

Finally, the administrator configures `net2` to become a standby interface.

```
# ipadm set-ifprop -p standby=on net2
```

▼ How to Manually Configure an Active-Active IPMP Group

The following procedure describes how to manually configure an active-active IPMP group. In this procedure, Steps 1-4 describe how to configure a link-based active-active IPMP group. Step 5 describes how to make the link-based configuration probe-based.

Before You Begin Make sure that IP interfaces that will be in the prospective IPMP group have been correctly configured over the system's network datalinks. For the procedures to configure links and IP interfaces, see "How to Configure an IP Interface" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*. You can create an IPMP interface even if underlying IP interfaces do not yet exist. However, subsequent configurations on this IPMP interface will fail.

Additionally, if you are using a SPARC based system, configure a unique MAC address for each interface. For the procedures, see "How to Ensure That the MAC Address of Each Interface Is Unique" in *Connecting Systems Using Fixed Network Configuration in Oracle Solaris 11.1*.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Create an IPMP interface.

```
# ipadm create-ipmp ipmp-interface
```

where *ipmp-interface* specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, for example, `ipmp0`.

3 Add underlying IP interfaces to the group.

```
# ipadm add-ipmp -i under-interface1 [-i underinterface2 ...] ipmp-interface
```

where *under-interface* refers to the underlying interface of the IPMP group. You can add as many IP interfaces as are available on the system.

Note – In a dual-stack environment, placing the IPv4 instance of an interface under a particular group automatically places the IPv6 instance under the same group.

---

4 Add data addresses to the IPMP interface.

```
# ipadm create-addr -a address ipmp-interface
```

where *address* can be in CIDR notation.

Note – Only the DNS address of the IPMP group name or IP address is required.

---

5 If you use probe-based failure detection with test addresses, add test addresses on the underlying interfaces.

```
# ipadm create-addr -a address under-interface
```

where *address* can be in CIDR notation. All test IP addresses in an IPMP group must belong to a single IP subnet and therefore using same network prefix.

How to Manually Configure an Active-Standby IPMP Group

For information about standby interfaces, see "Types of IPMP Interface Configurations" on page 68. The following procedure explains how to configures an IPMP group in which one interface is kept as a standby. This interface is deployed only when an active interface in the group fails.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Create an IPMP interface.**

```
# ipadm create-ipmp ipmp-interface
```

where *ipmp-interface* specifies the name of the IPMP interface. You can assign any meaningful name to the IPMP interface. As with any IP interface, the name consists of a string and a number, for example, `ipmp0`.

3 **Add underlying IP interfaces to the group.**

```
# ipadm add-ipmp -i under-interface1 [-i underinterface2 ...] ipmp-interface
```

where *under-interface* refers to the underlying interface of the IPMP group. You can add as many IP interfaces as are available on the system.

Note – In a dual-stack environment, placing the IPv4 instance of an interface under a particular IPMP group automatically places the IPv6 instance under the same group.

---

4 **Add data addresses to the IPMP interface.**

```
# ipadm create-addr -a address ipmp-interface
```

where *address* can be in CIDR notation.

5 **If you use probe-based failure detection with test addresses, add test addresses on the underlying interfaces.**

```
# ipadm create-addr -a address under-interface
```

where *address* can be in CIDR notation. All test IP addresses in an IPMP group must belong to a single IP subnet and therefore using same network prefix.

6 **Configure one of the underlying interfaces as a standby interface.**

```
# ipadm set-ifprop -p standby=on under-interface
```

Example 6–3 Configuring an Active-Standby IPMP Group

This example shows how to manually create an active-standby IPMP configuration.

First, the administrator creates the IPMP interface.

```
# ipadm create-ipmp ipmp0
```

Next, the administrator creates the underlying IP interfaces and adds them to the IPMP interface.

```
# ipadm create-ip net0
# ipadm create-ip net1
# ipadm create-ip net2

# ipadm add-ipmp -i net0 -i net1 -i net2 ipmp0
```

Next, the administrator assigns IP addresses to the IPMP interface. IP addresses assigned to the IPMP interface are data addresses. In this example, the IPMP interface has two data addresses.

```
# ipadm create-addr -a 192.168.10.10/24 ipmp0
ipadm: ipmp0/v4
# ipadm create-addr -a 192.168.10.15/24 ipmp0
ipadm: ipmp0/v4a
```

The IP address in this example includes a `prefixlen` property, which is expressed as a decimal number. The `prefixlen` portion of the IP address specifies the number of left-most contiguous bits of the address that comprise the IPv4 netmask or the IPv6 prefix of the address. The remaining low-order bits define the host part of the address. When the `prefixlen` property is converted to a text representation of the address, the address contains 1's for the bit positions that are to be used for the network part and 0's for the host part. This property is not supported on the `dhcp` address object type. For more information, see the `ipadm(1M)` man page.

Next, the administrator assigns IP addresses to the underlying IP interfaces of the IPMP group. IP addresses assigned to the underlying interfaces are test addresses to be used for probe-based failure detection.

```
# ipadm create-addr -a 192.168.10.30/24 net0
ipadm: net0/v4
# ipadm create-addr -a 192.168.10.32/24 net1
ipadm: net1/v4
# ipadm create-addr -a 192.168.10.34/24 net2
ipadm: net2/v4
```

Finally, the administrator configures `net2` to become a standby interface.

```
# ipadm set-ifprop -p standby=on net2
```

The administrator can view the IPMP configuration by using the `ipmpstat` command.

```
# ipmpstat -g
GROUP     GROUPNAME   STATE      FDT        INTERFACES
ipmp0     ipmp0       ok         10.00s     net0 net1 (net2)

# ipmpstat -t
INTERFACE  MODE     TESTADDR         TARGETS
net0       routes   192.168.10.30    192.168.10.1
net1       routes   192.168.10.32    192.168.10.1
net2       routes   192.168.10.34    192.168.10.5
```

Maintaining IPMP

This section contains procedures for maintaining the IPMP group that you have created on the system.

▼ How to Add an Interface to an IPMP Group

Before You Begin Make sure that the interface that you add to the group meets all of the requirements. For a list of requirements, see "How to Plan an IPMP Group" on page 85.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 If the underlying IP interface does not yet exist, create the interface.

```
# ipadm create-ip under-interface
```

3 Add the IP interface to the IPMP group.

```
# ipadm add-ipmp -i under-interface ipmp-interface
```

where *ipmp-interface* refers to the IPMP group to which you want to add the underlying interface.

Example 6–4 Adding an Interface to an IPMP Group

The following example adds the interface `net4` to the IPMP group `ipmp0`:

```
# ipadm create-ip net4
# ipadm add-ipmp -i net4 ipmp0
# ipmpstat -g
```

```
GROUP   GROUPNAME   STATE      FDT       INTERFACES
ipmp0   ipmp0      ok         10.00s    net0 net1 net4
```

How to Remove an Interface From an IPMP Group

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Remove one or more interfaces from the IPMP group.**

   # ipadm remove-ipmp -i *under-interface*[**-i** *under-interface* **...]** *ipmp-interface*

   where *under-interface* refers to an IP interface that you are removing from the IPMP group and *ipmp-interface* refers to the IPMP group from which you are removing underlying interfaces.

   You can remove as many underlying interfaces in a single command as required. Removing all underlying interfaces does not delete the IPMP interface. Instead, it exists as an empty IPMP interface or group.

Example 6–5 Removing an Interface From an IPMP Group

The following example removes the interface `net4` from the IPMP group `ipmp0`:

```
# ipadm remove-ipmp net4 ipmp0
# ipmpstat -g
GROUP   GROUPNAME   STATE      FDT       INTERFACES
ipmp0   ipmp0      ok         10.00s    net0 net1
```

How to Add IP Addresses

To add IP addresses, you use the `ipadm create-addr` subcommand. Note that in IPMP, an IP address can be either a data address or a test address. A data address is added to an IPMP interface. A test address is added to an underlying interface of the IPMP interface. The following procedure describes how to add IP addresses that are either test addresses or data addresses.

1. **Become an administrator.**

   For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **To add data addresses to an IPMP group, type the following command:**

   # ipadm create-addr -a *address ipmp-interface*

An address object is automatically assigned to the IP address that you just created. An address object is a unique identifier of the IP address. The address object's name uses the naming convention *interface*/*random-string*. Thus, address objects of data addresses would include the IPMP interface in their names.

3 **To add test addresses to an underlying interface of an IPMP group, type the following command:**

```
# ipadm create-addr -a address  under-interface
```

An address object is automatically assigned to the IP address that you just created. An address object is a unique identifier of the IP address. The address object's name uses the naming convention *interface*/*random-string*. Thus, address objects of test addresses would include the underlying interface in their names.

How to Delete IP Addresses

To remove IP addresses, you use the `ipadm delete-addr` subcommand. Note that in IPMP, data addresses are hosted on the IPMP interface and test addresses are hosted on underlying interfaces. The following procedure shows how to remove IP addresses that are either data addresses or test addresses.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Determine the addresses that you want to remove.**

- To display a list of data addresses, type the following command:

  ```
  # ipadm show-addr ipmp-interface
  ```

- To display a list of test addresses, type the following command:

  ```
  # ipadm show-addr
  ```

  Test addresses are identified by address objects whose names include the underlying interfaces where the addresses are configured.

3 **To remove data addresses from an IPMP group, type the following command:**

```
# ipadm delete-addr addrobj
```

where *addrobj* must include the name of the IPMP interface. If the address object that you type does not include the IPMP interface name, then the address that will be deleted is not a data address.

4 **To remove test addresses from an IPMP group, type the following command:**

```
# ipadm delete-addr addrobj
```

where *addrobj* must include the name of the correct underlying interface to delete the correct test address.

Example 6–6 Removing a Test Address From an Interface

The following example uses the configuration of the active-standby IPMP group `ipmp0` in Example 6–3. The example removes the test address from the underlying interface `net1`.

```
# ipadm show-addr net1
ADDROBJ           TYPE     STATE     ADDR
net1/v4           static   ok        192.168.10.30

# ipadm delete-addr net1/v4
```

▼ How to Move an Interface From One IPMP Group to Another IPMP Group

You can place an interface in a new IPMP group when the interface belongs to an existing IPMP group. You do not need to remove the interface from the current IPMP group. When you place the interface in a new group, the interface is automatically removed from any existing IPMP group.

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Move the interface to a new IPMP group.

`# ipadm add-ipmp -i` *under-interface ipmp-interface*

where *under-interface* refers to the underlying interface that you want to move and *ipmp-interface* refers to the IPMP interface to which you want to move the underlying interface.

Example 6–7 Moving an Interface to a Different IPMP Group

In this example, the underlying interfaces of the IPMP group are `net0`, `net1`, and `net2`. The following command removes the `net0` interface from IPMP group `ipmp0` and then places `net0` in the IPMP group `cs-link1`:

```
# ipadm add-ipmp -i net0 ca-link1
```

How to Delete an IPMP Group

Use this procedure if you no longer need a specific IPMP group.

1 **Become an administrator.**

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 **Identify the IPMP group and the underlying IP interfaces to be deleted.**

```
# ipmpstat -g
```

3 **Remove all IP interfaces that currently belong to the IPMP group.**

```
# ipadm remove-ipmp -i under-interface[, -i under-interface, ...] ipmp-interface
```

where *under-interface* refers to the underlying interface that you want to remove and *ipmp-interface* refers to the IPMP interface from which you want to remove the underlying interface.

Note – To successfully delete an IPMP interface, no IP interface must exist as part of the IPMP group.

---

4 **Delete the IPMP interface.**

```
# ipadm delete-ipmp ipmp-interface
```

After you delete the IPMP interface, any IP address that is associated with the interface is deleted from the system.

Example 6–8 Deleting an IPMP Interface

The following example deletes the interface `ipmp0` with the underlying IP interface `net0` and `net1`:

```
# ipmpstat -g
GROUP   GROUPNAME   STATE      FDT        INTERFACES
ipmp0   ipmp0       ok         10.00s     net0 net1

# ipadm remove-ipmp -i net0 -i net1 ipmp0

# ipadm delete-ipmp ipmp0
```

Configuring Probe-Based Failure Detection

Probe-based failure detection involves the use of target systems, as explained in "Probe-Based Failure Detection" on page 76. In identifying targets for probe-based failure detection, the `in.mpathd` daemon operates in two modes: router target mode or multicast target mode. In router target mode, the daemon probes targets that are defined in the routing table. If no targets are defined, then the daemon operates in multicast target mode, where multicast packets are sent out to probe neighbor hosts on the LAN.

Preferably, you should set up target systems for the `in.mpathd` daemon to probe. For some IPMP groups, the default router is sufficient as a target. However, for some IPMP groups, you might want to configure specific targets for probe-based failure detection. To specify the targets, set up host routes in the routing table as probe targets. Any host routes that are configured in the routing table are listed before the default router. IPMP uses the explicitly defined host routes for target selection. Thus, you should set up host routes to configure specific probe targets rather than use the default router.

To set up host routes in the routing table, you use the `route` command. You can use the `-p` option with this command to add persistent routes. For example, `route -p add` adds a route that will remain in the routing table even after you reboot the system. The `-p` option thus enables you to add persistent routes without needing any special scripts to re-create these routes with every system startup. To optimally use probe-based failure detection, make sure that you set up multiple targets to receive probes.

The procedure "How to Manually Specify Target Systems for Probe-Based Failure Detection" on page 100 shows the exact syntax to add persistent routes to targets for probe-based failure detection. For more information about the options for the `route` command, refer to the `route`(1M) man page.

The following topics are covered in this section:

Requirements for Choosing Targets for Probe-based Failure Detection

Follow this list of criteria when evaluating which hosts on your network might serve as good targets:

- Make sure that the prospective targets are available and running. Make a list of their IP addresses.
- Make sure that the target interfaces are on the same network as the IPMP group that you are configuring.
- The netmask and broadcast addresses of the target systems must be the same as the addresses in the IPMP group.

- The target host must be able to answer ICMP requests from the interface that is using probe-based failure detection.

Configuring Probe-Based Failure Detection (Task Map)

The following task map lists the procedures to configure probe-based failure detection for an IPMP group.

Task	Description	For Instruction
Choose how probe-based failure detection operates.	Determine whether probe-based failure detection should use transitive probes or test addresses.	"How to Select Which Failure Detection Method to Use" on page 99
Select target systems to be used for probe-based failure detection.	Specify the IP addresses of the systems that probe-based failure detection would use to test the status of underlying interfaces of the IPMP group.	"How to Manually Specify Target Systems for Probe-Based Failure Detection" on page 100
Choose how repaired interfaces are redeployed in the IPMP group.	Determine whether repaired IP interfaces should automatically be reactivated in the IPMP group or remain deactivated.	"How to Configure the Behavior of the IPMP Daemon" on page 101

▼ How to Select Which Failure Detection Method to Use

By default, probe-based failure detection operates by using test addresses. If the NIC driver supports it, link-based failure detection is also enabled automatically.

You cannot disable link-based failure detection if this method is supported by the NIC driver. However, you can select which type of probe-based failure detection to implement.

Before You Begin Make sure that your probe targets meet the requirements listed in "Requirements for Choosing Targets for Probe-based Failure Detection" on page 98.

1 To use only transitive probing, perform the following steps:

a. Use the appropriate SMF commands to switch on the IPMP property `transitive-probing`.

```
# svccfg -s svc:/network/ipmp setprop config/transitive-probing=true
# svcadm refresh  svc:/network/ipmp:default
```

For more information about setting this property, see the `in.mpathd(1M)` man page.

b. **Remove any existing test addresses that have been configured for the IPMP group.**

```
# ipadm delete-addr address addrobj
```

where *addrobj* must refer to an underlying interface that hosts a test address.

2 **To use test addresses to probe for failure, perform the following steps:**

a. **If necessary, turn off transitive probing.**

```
# svccfg -s svc:/network/ipmp setprop config/transitive-probing=false
# svcadm refresh  svc:/network/ipmp:default
```

b. **Assign test addresses to the underlying interfaces of the IPMP group.**

```
# ipadm create-addr -a address under-interface
```

where *address* can be in CIDR notation and *under-interface* is an underlying interface of the IPMP group.

▼ How to Manually Specify Target Systems for Probe-Based Failure Detection

This procedure describes how to add specific targets if you are using test addresses to implement probe-based failure detection.

Before You Begin Make sure that your probe targets meet the requirements listed in "Requirements for Choosing Targets for Probe-based Failure Detection" on page 98.

1 **Log in with your user account to the system on which you are configuring probe-based failure detection.**

2 **Add a route to a particular host to be used as a target in probe-based failure detection.**

```
$ route -p add -host destination-IP gateway-IP -static
```

where *destination-IP* and *gateway-IP* are IPv4 addresses of the host to be used as a target. For example, you would type the following to specify the target system `192.168.10.137`, which is on the same subnet as the interfaces in IPMP group `ipmp0`:

```
$ route -p add -host 192.168.10.137 192.168.10.137 -static
```

This new route will be automatically configured every time the system is restarted. If you only want to define a temporary route to a target system for probe-based failure detection, then do not use the `-p` option.

3 **Add routes to additional hosts on the network to be used as target systems.**

How to Configure the Behavior of the IPMP Daemon

Use the IPMP configuration file `/etc/default/mpathd` to configure the following system-wide parameters for IPMP groups.

- `FAILURE_DETECTION_TIME`
- `FAILBACK`
- `TRACK_INTERFACES_ONLY_WITH_GROUPS`

1 Become an administrator.

For more information, see "How to Use Your Assigned Administrative Rights" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Edit the `/etc/default/mpathd` file.

Change the default value of one or more of the three parameters.

a. Type the new value for the `FAILURE_DETECTION_TIME` parameter.

```
FAILURE_DETECTION_TIME=n
```

where *n* is the amount of time in seconds for ICMP probes to detect whether an interface failure has occurred. The default is 10 seconds.

b. Type the new value for the `FAILBACK` parameter.

```
FAILBACK=[yes | no]
```

- `yes`– The `yes` value is the default for the failback behavior of IPMP. When the repair of a failed interface is detected, network access fails back to the repaired interface, as described in "Detecting Physical Interface Repairs" on page 79.
- `no` – The `no` value indicates that data traffic does not return to a repaired interface. When a failed interfaces is detected as repaired, the `INACTIVE` flag is set for that interface. This flag indicates that the interface is currently not to be used for data traffic. The interface can still be used for probe traffic.

  For example, assume that the IPMP group `ipmp0` consists of two interfaces, `net0` and `net1`. In the `/etc/default/mpathd` file, the `FAILBACK=no` parameter is set. If `net0` fails, then it is flagged as `FAILED` and becomes unusable. After repair, the interface is flagged as `INACTIVE` and remains unusable because of the `FAILBACK=no` value.

  If `net1` fails and only `net0` is in the `INACTIVE` state, then the `INACTIVE` flag for `net0` is cleared and the interface becomes usable. If the IPMP group has other interfaces that are also in the `INACTIVE` state, then any one of these `INACTIVE` interfaces, and not necessarily `net0`, can be cleared and become usable when `net1` fails.

c. Type the new value for the `TRACK_INTERFACES_ONLY_WITH_GROUPS` parameter.

```
TRACK_INTERFACES_ONLY_WITH_GROUPS=[yes | no]
```

Note – For information about this parameter and the anonymous group feature, see "Failure Detection and the Anonymous Group Feature" on page 79.

- `yes`– The `yes` value is the default for the behavior of IPMP. This value causes IPMP to ignore network interfaces that are not configured into an IPMP group.
- `no` – The `no` value sets failure and repair detection for *all* network interfaces, regardless of whether they are configured into an IPMP group. However, when a failure or repair is detected on an interface that is not configured into an IPMP group, no action is triggered in IPMP to maintain the networking functions of that interface. Therefore, the `no` value is only useful for reporting failures and does not directly improve network availability.

3 Restart the `in.mpathd` daemon.

```
# pkill -HUP in.mpathd
```

The daemon restarts with the new parameter values in effect.

Monitoring IPMP Information

The examples in this section use the `ipmpstat` command to enable you to monitor different aspects of IPMP groups on the system. You can observe the status of an IPMP group as a whole or its underlying IP interfaces. You can also verify the configuration of data and test addresses for an IPMP group. Information about failure detection is also obtained by using the `ipmpstat` command. For more details about the `ipmpstat` command and its options, see the `ipmpstat(1M)` man page.

When you use the `ipmpstat` command, by default, the most meaningful fields that fit in 80 columns are displayed. In the output, all the fields that are specific to the option that you use with the `ipmpstat` command are displayed, except in the case of the `ipmpstat -p` syntax.

By default, host names are displayed in the output instead of numeric IP addresses, provided that the host names exist. To list the numeric IP addresses in the output, use the `-n` option with other options to display specific IPMP group information.

Note – In the following examples, use of the `ipmpstat` command does not require system administrator privileges, unless stated otherwise.

You use the following options with the `ipmpstat` command to determine the information to be displayed:

- `-g` displays information about the IPMP groups on the system. See Example 6–9.
- `-a` displays the data addresses that are configured for the IPMP groups. See Example 6–10.

- `-i` displays information about IP interfaces that are related to IPMP configuration. See Example 6–11.
- `-t` displays information about target systems that are used for detecting failure. This option also displays the test addresses that are used by the IPMP group. See Example 6–12.
- `-p` displays information about the probes that are being used for failure detection. See Example 6–13.

The following examples show information about your system's IPMP configuration that you can obtain with the `ipmpstat` command.

EXAMPLE 6–9 Obtaining IPMP Group Information

The `-g` option lists the status of the various IPMP groups on the system, including the status of their underlying interfaces. If probe-based failure detection is enabled for a specific group, the command also includes the failure detection time for that group.

```
$ ipmpstat -g
GROUP   GROUPNAME   STATE      FDT        INTERFACES
ipmp0   ipmp0       ok         10.00s     net0 net1
acctg1  acctg1      failed     --         [net3 net4]
field2  field2      degraded   20.00s     net2 net5 (net7) [net6]
```

The output fields provide the following information:

`GROUP` Specifies the IPMP interface name. For an anonymous group, this field is empty. For more information about anonymous groups, see the `in.mpathd(1M)` man page.

`GROUPNAME` Specifies the name of the IPMP group. In the case of an anonymous group, this field will be empty.

`STATE` Indicates an IPMP group's current status, which can be one of the following:

- `ok` indicates that all underlying interfaces of the IPMP group are usable.
- `degraded` indicates that some of the underlying interfaces in the group are unusable.
- `failed` indicates that all of the group's interfaces are unusable.

`FDT` Specifies the failure detection time, if failure detection is enabled. If failure detection is disabled, this field is empty.

`INTERFACES` Specifies the underlying interfaces that belong to the IPMP group. In this field, active interfaces are listed first, then inactive interfaces, and finally unusable interfaces. The status of an interface is indicated by the manner in which it is listed:

- *interface* (without parentheses or square brackets) indicates an active interface. Active interfaces are being used by the system to send or receive data traffic.

EXAMPLE 6–9 Obtaining IPMP Group Information *(Continued)*

- *(interface)* (with parentheses) indicates a functioning but inactive interface. The interface is not in use as defined by administrative policy.
- *[interface]* (with square brackets) indicates that the interface is unusable because it has either failed or been taken offline.

EXAMPLE 6–10 Obtaining IPMP Data Address Information

The `-a` option displays data addresses and the IPMP group to which each address belongs. The displayed information also includes which addresses are available for use, depending on whether the addresses have been toggled by the `ipadm [up-addr/down-addr]` command. You can also determine on which inbound or outbound interface an address can be used.

```
$ ipmpstat -an
ADDRESS         STATE    GROUP      INBOUND     OUTBOUND
192.168.10.10   up       ipmp0         net0    net0 net1
192.168.10.15   up       ipmp0         net1    net0 net1
192.0.0.100     up       acctg1       --          --
192.0.0.101     up       acctg1       --          --
128.0.0.100     up       field2        net2    net2 net7
128.0.0.101     up       field2        net7    net2 net7
128.0.0.102     down     field2       --          --
```

The output fields provide the following information:

`ADDRESS` Specifies the host name or the data address, if the `-n` option is used with the `-a` option.

`STATE` Indicates whether the address on the IPMP interface is `up`, and therefore usable, or `down`, and therefore unusable.

`GROUP` Specifies the IPMP interface that hosts a specific data address. Typically, in Oracle Solaris, the name of the IPMP group is the IPMP interface.

`INBOUND` Identifies the interface that receives packets for a given address. The field information might change depending on external events. For example, if a data address is down, or if no active IP interfaces remain in the IPMP group, this field is empty. The empty field indicates that the system is not accepting IP packets that are destined for the given address.

`OUTBOUND` Identifies the interface that sends packets that are using a given address as a source address. As with the `INBOUND` field, the `OUTBOUND` information might also change depending on external events. An empty field indicates that the system is not sending packets with the given source address. The field might be empty either because the address is down or because no active IP interfaces remain in the group.

EXAMPLE 6–11 Obtaining Information About Underlying IP Interfaces of an IPMP Group

The `-i` option displays information about an IPMP group's underlying IP interfaces.

```
$ ipmpstat -i
INTERFACE   ACTIVE   GROUP      FLAGS      LINK       PROBE      STATE
net0        yes      ipmp0      --mb---    up         ok         ok
net1        yes      ipmp0      -------    up         disabled   ok
net3        no       acctg1     -------    unknown    disabled   offline
net4        no       acctg1     is-----    down       unknown    failed
net2        yes      field2     --mb---    unknown    ok         ok
net6        no       field2     -i-----    up         ok         ok
net5        no       filed2     -------    up         failed     failed
net7        yes      field2     --mb---    up         ok         ok
```

The output fields provide the following information:

`INTERFACE` Specifies each underlying interface in each IPMP group.

`ACTIVE` Indicates whether the interface is functioning and in use (`yes`) or not (`no`).

`GROUP` Specifies the IPMP interface name. For anonymous groups, this field is empty. For more information about anonymous groups, see the `in.mpathd(1M)` man page.

`FLAGS` Indicates the status of each underlying interface, which can be one or any combination of the following:

- `i` indicates that the `INACTIVE` flag is set for the interface. Therefore, the interface is not used to send or receive data traffic.
- `s` indicates that the interface is configured to be a standby interface.
- `m` indicates that the interface is designated by the system to send and receive IPv4 multicast traffic for the IPMP group.
- `b` indicates that the interface is designated by the system to receive broadcast traffic for the IPMP group.
- `M` indicates that the interface is designated by the system to send and receive IPv6 multicast traffic for the IPMP group.
- `d` indicates that the interface is down and therefore unusable.
- `h` indicates that the interface shares a duplicate physical hardware address with another interface and has been taken offline. The `h` flag indicates that the interface is unusable.

`LINK` Indicates the status of link-based failure detection, which is one of the following:

- `up` or `down` indicates the availability or unavailability of a link.
- `unknown` indicates that the driver does not support notification of whether a link is `up` or `down` and therefore does not detect changes in the state of the link.

EXAMPLE 6–11 Obtaining Information About Underlying IP Interfaces of an IPMP Group *(Continued)*

PROBE — Specifies the state of probe-based failure detection for interfaces that have been configured with a test address, as follows:

- `ok` indicates that the probe is functional and active.
- `failed` indicates that probe-based failure detection has detected that the interface is not working.
- `unknown` indicates that no suitable probe targets could be found, Therefore, probes cannot be sent.
- `disabled` indicates that no IPMP test address is configured on the interface. Therefore, probe-based failure detection is disabled.

STATE — Specifies the overall state of the interface, as follows:

- `ok` indicates that the interface is online and working normally based on the configuration of failure detection methods.
- `failed` indicates that the interface is not working either because the interface's link is down or because the probe detection has determined that the interface cannot send or receive traffic.
- `offline` indicates that the interface is not available for use. Typically, the interface is taken offline under the following circumstances:
  - The interface is being tested.
  - Dynamic reconfiguration is being performed.
  - The interface shares a duplicate hardware address with another interface.
- `unknown` indicates that the IPMP interface's state cannot be determined because no probe targets were found for probe-based failure detection.

EXAMPLE 6–12 Obtaining IPMP Probe Target Information

The `-t` option identifies the probe targets that are associated with each IP interface in an IPMP group. The first output is an example of an IPMP configuration that uses test addresses for probe-based failure detection.

```
$ ipmpstat -nt
INTERFACE   MODE          TESTADDR        TARGETS
net0        routes        192.168.85.30   192.168.85.1 192.168.85.3
net1        disabled      --              --
net3        disabled      --              --
net4        routes        192.1.2.200     192.1.2.1
net2        multicast     128.9.0.200     128.0.0.1 128.0.0.2
net6        multicast     128.9.0.201     128.0.0.2 128.0.0.1
net5        multicast     128.9.0.202     128.0.0.1 128.0.0.2
net7        multicast     128.9.0.203     128.0.0.1 128.0.0.2
```

EXAMPLE 6–12 Obtaining IPMP Probe Target Information *(Continued)*

The second output is an example of an IPMP configuration that uses transitive probing or probe-based failure detection without test addresses.

```
$ ipmpstat -nt
INTERFACE   MODE         TESTADDR         TARGETS
net3        transitive   <net1>           <net1> <net2> <net3>
net2        transitive   <net1>           <net1> <net2> <net3>
net1        routes       172.16.30.100    172.16.30.1
```

The output fields provide the following information:

`INTERFACE` Specifies each underlying interface of an IPMP group.

`MODE` Specifies the method for obtaining the probe targets.

- `routes` indicates that the system routing table is used to find probe targets.
- `mcast` indicates that multicast ICMP probes are used to find targets.
- `disabled` indicates that probe-based failure detection has been disabled for the interface.
- `transitive` indicates that transitive probing is used for failure detection, as shown in the second example. Note that you cannot implement probe-based failure detection while simultaneously using transitive probes and test addresses. If you do not want to use test addresses, then you must enable transitive probing. If you do not want to use transitive probing, then you must configure test addresses. For an overview, see "Probe-Based Failure Detection" on page 76.

`TESTADDR` Specifies the host name or, if the `-n` option is used with the `-t` option, the IP address that is assigned to the interface to send and receive probes.

If transitive probing is used, then the interface names refer to the underlying IP interfaces that are not actively used to receive data. The names also indicate that the transitive test probes are being sent with the source address of these specified interfaces. For active underlying IP interfaces that receive data, an IP address that is displayed indicates the source address of outgoing ICMP probes.

Note – If an IP interface is configured with both IPv4 and IPv6 test addresses, the probe target information is displayed separately for each test address.

`TARGETS` Lists the current probe targets in a space-separated list. The probe targets are displayed either as host names or IP addresses. If the `-n` option is used with the `-t` option, the IP addresses are displayed.

EXAMPLE 6–13 Observing IPMP Probes

The `-p` option enables you to observe ongoing probes. When you use this option with the `ipmpstat` command, information about probe activity on the system is continuously displayed until you terminate the command by pressing Control-C. You must have Primary Administrator privileges to run this command.

The first output is an example of an IPMP configuration that uses test addresses for probe-based failure detection.

```
# ipmpstat -pn
TIME    INTERFACE   PROBE     NETRTT   RTT      RTTAVG     TARGET
0.11s   net0        589       0.51ms   0.76ms   0.76ms     192.168.85.1
0.17s   net4        612       --       --       --         192.1.2.1
0.25s   net2        602       0.61ms   1.10ms   1.10ms     128.0.0.1
0.26s   net6        602       --       --       --         128.0.0.2
0.25s   net5        601       0.62ms   1.20ms   1.00ms     128.0.0.1
0.26s   net7        603       0.79ms   1.11ms   1.10ms     128.0.0.1
1.66s   net4        613       --       --       --         192.1.2.1
1.70s   net0        603       0.63ms   1.10ms   1.10ms     192.168.85.3
^C
```

The second output is an example of an IPMP configuration that uses transitive probing or probe-based failure detection without test addresses.

```
# ipmpstat -pn
TIME    INTERFACE   PROBE      NETRTT   RTT      RTTAVG     TARGET
1.39S   net4        t28        1.05ms   1.06ms   1.15ms     <net1>
1.39s   net1        i29        1.00ms   1.42ms   1.48ms     172.16.30.1
^C
```

The output fields provide the following information:

`TIME` Specifies the time a probe was sent relative to when the `ipmpstat` command was issued. If a probe was initiated prior to `ipmpstat` being started, then the time is displayed with a negative value, relative to when the command was issued.

`INTERFACE` Specifies the interface on which the probe is sent.

`PROBE` Specifies the identifier that represents the probe. If transitive probing is used for failure detection, the identifier is prefixed with either `t` for transitive probes or `i` for ICMP probes.

`NETRTT` Specifies the total network round-trip time of the probe, measured in milliseconds. `NETRTT` covers the time between the moment when the IP module sends the probe and the moment the IP module receives the `ack` packets from the target. If the `in.mpathd` daemon has determined that the probe is lost, then the field is empty.

`RTT` Specifies the total round-trip time for the probe, measured in milliseconds. `RTT` covers the time between the moment the `in.mpathd` daemon executes the code

EXAMPLE 6–13 Observing IPMP Probes *(Continued)*

to send the probe and the moment the daemon completes processing of the `ack` packets from the target. If the daemon has determined that the probe is lost, then the field is empty. Spikes that occur in the `RTT` that are not present in the `NETRTT` might indicate that the local system is overloaded.

`RTTAVG` Specifies the probe's average round-trip time over the interface between the local system and the target. The average round-trip time helps identify slow targets. If data is insufficient to calculate the average, this field is empty.

`TARGET` Specifies the host name or, if the `-n` option is used with the `-p` option, the target address to which the probe is sent.

Customizing the Output of the `ipmpstat` Command

The `ipmpstat` command's `-o` option enables you to customize the output. You use this option with the other previously `ipmpstat` options to select specific fields to be displayed out of the total fields that the main option normally displays.

For example, the `-g` option provides the following information:

- IPMP group
- IPMP group name
- Status of the group
- Failure detection time
- Underlying interfaces of the IPMP group

Suppose that you want to display only the status of the IPMP groups on the system. You would combine the `-o` and `-g` options and specify the fields `groupname` and `state`, as shown in the following example:

```
$ ipmpstat -g -o groupname,state
GROUPNAME  STATE
ipmp0      ok
accgt1     failed
field2     degraded
```

To display all the fields of the `ipmpstat` command for a specific type of information, include the `-o all` in the syntax. For example, to list all the fields that provide group information, type `ipmpstat -g -o all`.

Using the `ipmpstat` Command in Scripts

The `-o` option is useful when you issue the command from a script or by using a command alias, particularly if you also want to generate machine-parseable output.

To generate machine-parseable information, you combine the `-P` and `-o` options with one of the other main `ipmpstat` options, together with specific fields you want to display. A machine-parseable output differs from normal output in the following ways:

- Column headers are omitted.
- Fields are separated by colons (`:`).
- Fields with empty values are empty rather than filled with the double dash (`--`).
- When multiple fields are requested, if a field contains a literal colon (`:`) or backslash (`\`), these characters can be escaped or excluded by prefixing them with a backslash (`\`).

To correctly use the `ipmpstat -P` syntax, observe the following rules:

- Use the `-o` *option field* with the `-P` option. Separate multiple option fields with commas.
- Never use `-o all` with the `-P` option.

Ignoring either one of these rules causes `ipmpstat -P` to fail.

The following example shows the format of the information when you use the `-P` option.

```
$ ipmpstat -P -o -g groupname,fdt,interfaces
ipmp0:10.00s:net0 net1
acctg1::[net3 net4]
field2:20.00s:net2 net7 (net5) [net6]
```

The group name, failure detection time, and underlying interfaces are group information fields. Thus, you use the `-o -g` options with the `-P` option.

The `-P` option is intended to be used in scripts. The following example shows how the ipmpstat command is issued from a script. The script displays the failure detection time for an IPMP group.

```
getfdt() {
         ipmpstat -gP -o group,fdt | while IFS=: read group fdt; do
             [[ "$group" = "$1" ]] && { echo "$fdt"; return; }
         done
     }
```

◆ ◆ ◆ CHAPTER 7

Exchanging Network Connectivity Information With LLDP

On any local area network, individual components such as systems and switches are not configured in isolation. To host network traffic efficiently, the configuration of systems on the network must be coordinated with each other. Thus, packet exchange occurs based on the components' capabilities, link property configuration, bandwidth limits, and so on. You can manually configure each system, switch, and other components to ensure compatibility among these components. However, this method is risky and can easily cause misconfigurations, particularly if different administrators work independently on different systems. A better alternative is to use a technology that enables systems to transmit their individual configuration information to peer systems. With this technology, risk is diminished and network administration becomes easier. Oracle Solaris supports Link Layer Discovery Protocol (LLDP) for this specific purpose.

This chapter describes how to enable systems to exchange system and network connectivity information throughout the local network by using LLDP. The following topics are covered:

Overview of LLDP in Oracle Solaris

LLDP advertises information throughout a LAN for purposes of topology discovery. With this protocol, a system can advertise connectivity and management information to other systems on the network. This information can include system capabilities, management addresses, and other information relevant to network operations. This protocol also enables that same system to receive similar information about other systems that are on the same local network.

In Oracle Solaris, LLDP is also used to exchange data center bridging exchange protocol (DCBX) TLV units. DCBX provides configuration information about DCB features such as

priority-based flow control (PFC) and enhanced transmission selection (ETS). For more information about DCB, see Chapter 8, "Working With Data Center Bridging Features in Oracle Solaris."

With LLDP, the system administrator can easily detect faulty system configurations, particularly in complex networks that include virtual local area networks (VLANs), link aggregations, and so on. Information about the topology can be obtained readily without requiring to trace physical connections between servers, switches, and other devices that comprise the network.

Components of an LLDP Implementation

LLDP is implemented with the following components:

- The LLDP package must be installed to enable the LLDP feature. This package delivers the LLDP daemon, command-line utilities, the service manifest and scripts, and other components that are required for LLDP to operate.
- The `lldp` service is enabled by the `svcadm` command. This service manages the LLDP daemon and is responsible for starting, stopping, restarting, or refreshing the daemon. This service is automatically enabled after you install the LLDP package.
- The `lldpadm` command administers LLDP on individual links and is used, for example, to configure the operating mode of LLDP, to specify Type-Length-Value (TLV) units that will be transmitted, and to configure DCBX TLV units. Specifically, the command is used to set per-agent LLDP properties as well as global LLDP properties. The general subcommands of the `lldpadm` command parallel the subcommands of the `dladm` and `ipadm` commands.
  - `lldpadm set-*` specifies the action to be performed in which one or more values are set for a given LLDP property.
  - `lldpadm show-*` displays the values that are set for a specified LLDP property.
  - `lldpadm reset-*` resets the configuration of a specified LLDP property to its default values.

  Use of these subcommands is illustrated in subsequent sections. For more information about the `lldpadm` command, refer to the `lldpadm(1M)` man page.
- The LLDP library (`liblldp.so`) provides APIs that can be used to retrieve LLDP information on a link, to parse LLDP packets, and to perform other functions.
- LLDP agents are LLDP instances that are associated with the network interface card where LLDP is enabled. An LLDP agent controls LLDP behavior on an associated NIC. LLDP agents can be configured only on NICs or physical links and uses the ports of these links to advertise information. Thus, in this LLDP documentation, the name of the LLDP agent, the physical link on which it is enabled, and the port are identical.

- The LLDP daemon (`lldpd`) manages the LLDP agents on the system. It also interacts with `snmpd`, the daemon for the Simple Network Management Protocol (SNMP), to retrieve LLDP information that is received on the system through SNMP. In addition, the daemon posts LLDP `sysevents` information as well as responds to queries from the LLDP library.

Information Sources of the LLDP Agent

The LLDP agent transmits as well as receives LLDP data units (LLDPDUs). The agent manages and stores the information contained in these LLDPDUs in two types of data stores:

- **Local management information base, or local MIB** - This data store contains network information that pertains to a system's specific link on which the LLDP agent is enabled. A local MIB contains both common and unique information. For example, the chassis ID is common information that is shared among all the LLDP agents on the system. However, port IDs for the system's datalinks are different. Thus, each agent manages its own local MIB.
- **Remote MIB** - Information in this data store is received from LLDP agents of peer hosts.

LLDP Agent Modes of Operation

The LLDP agent operates in the following modes:

- Transmit only (`txonly`): In this mode, the LLDP agent does not process incoming LLDPDUs. Therefore, the remote MIB is empty.
- Receive only (`rxonly`): In this mode, the agent processes only incoming LLDPDUs and stores the information in remote MIBs. However, no information from the local MIB is transmitted.
- Transmit and receive (`both`): In this mode, the agent transmits local information and processes incoming LLDPDUs and thus, maintains both local and remote MIBs. If the underlying link supports DCB features, DCBX TLV units are automatically enabled for the supported DCB features.
- Disabled (`disable`): In this mode, the agent does not exist.

SMF Property for LLDP

The service management facility (SMFS) property `auto-enable-agents` controls how LLDP is enabled on the system. With this property, you can choose to enable LLDP globally across all the physical links or only one physical link at a time. The property can have one of three possible values:

- `yes` is the default value of the SMF property. With this value, LLDP is enabled globally on all ports in both Rx and Tx modes, provided that no previous LLDP configuration exists on a port. If a configuration exists on a port, then that port's configuration is retained. For example, if a port has been previously configured with LLDP in Rx mode only, then the LLDP service will not switch the agent to run in Rx and Tx modes. LLDP on that port continues to be in Rx mode.
- `force` enables LLDP in both Rx and Tx modes on all ports and overrides any existing LLDP configurations on any port. For example, if a previous LLDP configuration on a port runs in Rx mode only, the LLDP agent is switched to run in both Rx and Tx modes, which is the default LLDP mode.
- `no` disables automatic enabling of LLDP on all ports except those with existing LLDP configurations. On these ports, existing LLDP configuration is retained.

Note that each time you customize the `auto-enable-agents` property, you must restart the LLDP SMF service for the new value to become effective.

Information the LLDP Agent Advertises

The LLDP agent transmits system and connectivity information in LLDP packets or LLDPDUs. These packets contain information units that are individually formatted in Type-Length-Value (TLV) format. Thus, the information units are also called *TLV units*. Certain TLV units are mandatory and are included in LLDP packets by default when LLDP is enabled. You cannot use the `lldpadm` command to exclude any of these units. The mandatory TLV units are as follows:

- Chassis ID
- Port ID
- TTL (time to live)
- End of PDU

The Chassis ID is the same information that is generated by the `hostid` command. The Port ID is the MAC address of the physical NIC. Multiple LLDP agents can be enabled on a single system depending on the number of links. The Chassis ID and Port ID combination uniquely identifies an agent and distinguishes it from other agents on the system.

Optional TLV units can be added to an LLDP packet. These optional TLV units are means for vendors to insert vendor-specific TLV units to be advertised. The TLV units are identified by individual organization unique identifiers (OUIs) and are typed according to whether these OUIs follow IEEE 802.1 specifications or IEEE 802.3 specifications. LLDP agent properties can be configured to enable or disable the transmission of these optional TLV units.

The following table lists each TLV type or group, its corresponding property name, as well as the TLV units for each property, and their descriptions. You configure any one of these properties to specify the TLV units to be included in the packets when LLDP is enabled.

TABLE 7–1 TLV Units That Can Be Enabled for an LLDP Agent

TLV Type	Property Name	TLV units	Description
Basic management	`basic-tlv`	`sysname`, `portdesc`, `syscapab`, `sysdesc`, `mgmtaddr`	Specifies the system name, port description, system capability, system description, and management address to be advertised.
802.1 OUI	`dot1-tlv`	`vlanname`, `pvid`, `linkaggr`, `pfc`, `appln`, `evb`, `etscfg`	Specifies the following to be advertised: VLAN name, port VLAN ID, link aggregation, TLV units for priority-based flow control, application, enhanced transmission selection, and edge virtual bridging.
802.3 OUI	`dot3-tlv`	`max-framesize`	Specifies the maximum frame size to be advertised.
Oracle-specific OUI (which is defined as `0x0003BA`)	`virt-tlv`	`vnic`	Specifies the VNIC to be advertised if a virtual network is configured.

TLV Units and Their Properties

Each TLV unit has properties that you can further configure with specific values. When the TLV unit is enabled as an LLDP agent's property, then that TLV unit is advertised in the network only with the specified values. For example, consider the TLV value `syscapab`, which advertises a system's capabilities. These capabilities can potentially include support for routers, bridges, repeaters, telephones, and other devices. However, you can set `syscapab` so that only those capabilities that are actually supported on your specific system, such as routers and bridges, are advertised.

The procedure for configuring TLV units depends on whether you are configuring *global TLV units* or *per-agent TLV units*.

Global TLV units apply to all the LLDP agents on the system. The following table displays the global TLV values and their corresponding possible configurations.

TABLE 7–2 Global TLV units and Their Properties

TLV Name	TLV Property Name	Possible Property Values	Value Description
`syscapab`	`supported`	`other`, `repeater`, `bridge`, `wlan-ap`, `router`, `telephone`, `docsis-cd`, `station`, `cvlan`, `sylvan`, `tpmr`	Represent the primary supported functions of the system. Default values are `router`, `station`, and `bridge`.
	`enabled`	Subset of the values listed for `supported`	Represents the enabled functions of the system.
`mgmtaddr`	`ipaddr`	`ipv4` or `ipv6`	Specifies the type of IP addresses that will be associated with the local LLDP agent. The addresses will be used to reach higher layer entities and will assist in discovery by network management. Only one type can be specified.

TLV units that cannot have global values are managed at the LLDP agent level. With per-agent TLV units, the values that you provide are used when the TLV unit is enabled for transmission by a specific LLDP agent.

The following table displays the TLV values and their corresponding possible configurations for an LLDP agent.

TABLE 7–3 Per-Agent TLV Units and Their Properties

TLV Name	TLV Property Name	Possible Property Values	Value Description
`pfc`	`willing`	`on`, `off`	Sets an LLDP agent to accept or reject configuration information from a remote machine that pertains to priority-based flow control.
`appln`	`apt`	Values are taken from the information that is defined in the Application Priority Table.	Configures the Application Priority Table. This table contains the list of application TLV units and their corresponding priorities. The application is identified by the `id/selector` pair. The contents of the table use the following format: `id/selector/priority`

TABLE 7–3 Per-Agent TLV Units and Their Properties *(Continued)*

TLV Name	TLV Property Name	Possible Property Values	Value Description
etscfg	willing	on, off	Sets an LLDP agent to accept or reject configuration information from a remote machine that pertains to enhanced transmission selection.

For a discussion about per-agent TLV units, see Chapter 8, "Working With Data Center Bridging Features in Oracle Solaris."

Enabling LLDP on the System

The following procedures describe how to configure LLDP to exchange system information with other hosts or peers on the network.

▼ How to Deploy LLDP

The following procedure describes how to use LLDP on your system to begin advertising system capabilities. By default, LLDP is enabled and ready to be used after you have completed installing the LLDP package. If you are satisfied with the default LLDP configurations, then most of the steps would be optional.

Before You Begin You must install the LLDP package to use LLDP. To install the package, type the following command:

```
# pkg install lldp
```

1 Ensure that the LLDP service has started.

```
# svcs lldp
STATE          STIME    FMRI
online         Jul_10   svc:/network/lldp:default
```

If the LLDP service is disabled, start the service with the following command:

```
# svcadm enable svc:/network/lldp:default
```

2 Perform one of the following steps.

- If you want the LLDP service to be enabled globally on the system, specify the TLV units that the LLDP agents would advertise.

  ```
  # lldpadm set-agentprop -p property=value agent
  ```

  where *agent* is the LLDP agent and is identified by the physical link on which the agent is enabled. Thus, if LLDP is enabled on `net0`, then the agent is `net0`.

  Note – You can use abbreviated forms when issuing `lldpadm` subcommands. For example, for `lldpadm set-agentprop` , you can instead type `lldpadm set-ap`. Refer to the `lldpadm`(1M) man page for the subcommands and their abbreviated forms.

  For an explanation of the properties of the LLDP agent, see "Information the LLDP Agent Advertises" on page 114.

  For a list of properties of the LLDP agent, type `lldpadm show-agentprop`. Or, refer to Table 7–1.

  For instructions, see "How to Specify TLV Units for an Agent's LLDP Packet" on page 120.

- If you want the LLDP service to be enabled only on selective ports, perform the following steps.

  a. Change the SMF `auto-enable-agents` property to `no`.

     ```
     # svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
     ```

     This SMF property determines how LLDP is enabled on the system. It has three possible values: `yes`, `force`, and `no`. By default, the property is set to `yes`. For an explanation of these values and the consequent behavior of the LLDP agent from these values, see "SMF Property for LLDP" on page 113.

  b. Restart the LLDP service.

     ```
     # svcadm restart svc:/network/lldp:default
     ```

  c. Enable LLDP agents on selected ports or links.

     ```
     # lldpadm set-agentprop -p mode=value agent
     ```

     where *agent* is the LLDP agent and is identified by the physical link on which the agent is enabled. Thus, if you enable LLDP on `net0`, the agent is `net0`.

     The property `mode` can be set to one of four possible values that represent the LLDP agent's modes of operation: `tx`, `rx`, `both`, and `disable`. For an explanation of these values, see "LLDP Agent Modes of Operation" on page 113.

  d. Specify the TLV units that the LLDP agent would advertise.

     ```
     # lldpadm set-agentprop -p property=value agent
     ```

     For an explanation of the properties of the LLDP agent, see "Information the LLDP Agent Advertises" on page 114.

For a list of the other properties of the LLDP agent in addition to the `mode` property, type `lldpadm show-agentprop`. Or, refer to Table 7–1.

For instructions, see "How to Specify TLV Units for an Agent's LLDP Packet" on page 120.

3 If necessary, customize global TLV units.

```
# lldpadm set-tlvprop -p property=value global-tlv
```

where *property* refers to the property of the global TLV unit.

For an explanation of global TLV units, see "TLV Units and Their Properties" on page 115.

For a list of global TLVs, type `lldpadm show-tlvprop`. Or, refer to Table 7–2.

For instructions, see "How to Define TLV Values" on page 121.

4 If necessary, customize per-agent TLV units.

```
# lldpadm set-agenttlvprop -p property=value -a agent per-agent-tlv
```

where *property* refers to the property of the per-agent TLV unit.

For an explanation of per-agent TLV units, see "TLV Units and Their Properties" on page 115.

For a list of per-agent TLVs, type `lldpadm show-tlvprop`. Or, refer to Table 7–2.

For instructions, see "How to Define TLV Values" on page 121.

Example 7–1 Customizing the auto-enable-agents SMF Property

The following example shows the different manner LLDP is enabled if you change the value of the SMF property. Suppose that on a system with four ports, LLDP is configured on two ports as follows:

- `net0`: Rx and Tx modes
- `net1`: Rx only
- `net2` and `net3`: none

With the SMF property set to the default value `yes`, LLDP is automatically enabled on `net2` and `net3`. The LLDP configuration is displayed as follows:

```
# lldpadm show-agentprop -p mode
AGENT   PROPERTY   PERM   VALUE   DEFAULT   POSSIBLE
net0    mode       rw     both    disable   txonly,rxonly,both,
                                            disable
net1    mode       rw     rxonly  disable   txonly,rxonly,both,
                                            disable
net2    mode       rw     both    disable   txonly,rxonly,both,
                                            disable
net3    mode       rw     both    disable   txonly,rxonly,both,
                                            disable
```

If you switch the SMF property to `no`, the configuration changes when you restart the service.

```
# svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
# svcadm restart svc:/network/lldp:default
# lldpadm show-agentprop -p mode
AGENT   PROPERTY   PERM   VALUE    DEFAULT   POSSIBLE
net0    mode       rw     both     disable   txonly,rxonly,both,
                                             disable
net1    mode       rw     rxonly   disable   txonly,rxonly,both,
                                             disable
net2    mode       rw     disable  disable   txonly,rxonly,both,
                                             disable
net3    mode       rw     disable  disable   txonly,rxonly,both,
                                             disable
```

In the sample output, `net2` and `net3`, whose LLDP modes were previously automatically enabled, are now flagged as disabled. However, no change occurs on `net0` and `net1` whose LLDP agents were previously configured.

Example 7–2 Enabling LLDP on Multiple Datalinks

This example shows you how to enable LLDP selectively. A system has two datalinks, `net0` and `net1`. On `net0`, you want the agent to transmit and receive LLDP packets. On `net1`, you want the agent to transmit LLDP packets only. You would type the following commands:

```
# svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
# svcadm restart svc:/network/lldp:default
# lldpadm set-agentprop -p mode=both net0
# lldpadm set-agentprop -p mode=txonly net1
```

▼ How to Specify TLV Units for an Agent's LLDP Packet

This procedure explains how to specify TLV units to be advertised in an LLDP packet that an agent transmits. To specify TLV units, you use the `lldpadm set-agentprop` subcommand.

1 If necessary, identify the LLDP agent property that can contain the TLV unit that you want to add.

This subcommand also displays the TLV units that are already set for each property.

```
# lldpadm show-agentprop agent
```

Without specifying the property, this subcommand displays all the LLDP agent properties and their TLV values.

2 Add the TLV unit to the property.

```
# lldpadm set-agentprop -p property[+|-]=value[,...] agent
```

The `+|-` qualifiers are used for properties that accept multiple values. These qualifiers enable you to add (`+`) or remove (`-`) values from the list. If you do not use the qualifiers, then the value that you set replaces all the values that were previously defined for the property.

3 (Optional) Display the new values for the property.

```
# lldpadm show-agentprop -p property agent
```

Example 7–3 Adding Optional TLV Units to an LLDP Packet

In this example, the LLDP agent `net0` is already configured to advertise VLAN information in its LLDP packet. You want to include system capabilities, link aggregation, and network virtualization information to be advertised as well. However, you want to remove the VLAN description from the packet.

```
# lldpadm show-agentprop net0
AGENT   PROPERTY   PERM   VALUE          DEFAULT   POSSIBLE
net0    mode       rw     both           disable   txonly,rxonly,both,
                                                   disable
net0    basic-tlv  rw     sysname,        none     none,portdesc,
                          sysdesc                  sysname,sysdesc,
                                                   syscapab,mgmtaddr,
                                                   all
net0    dot1-tlv   rw     vlanname,       none     none,vlanname,pvid,
                          pvid,pfc                 linkaggr,pfc,appln,
                                                   evb,etscfg,all
net0    dot3-tlv   rw     max-framesize   none     none, max-framesize,
                                                   all
net0    virt-tlv   rw     none            none     none,vnic,all


# lldpadm set-agentprop -p basic-tlv+=syscapab,dot1-tlv+=linkaggr,virt-tlv=vnic net0
# lldpadm set-agentprop -p dot1-tlv-=vlanname net0
# lldpadm show-agentprop -p net0
AGENT   PROPERTY   PERM   VALUE          DEFAULT   POSSIBLE
net0    mode       rw     both           disable   txonly,rxonly,both,
                                                   disable
net0    basic-tlv  rw     sysname,        none     none,portdesc,
                          sysdesc,                 sysname,sysdesc,
                          syscapab                 syscapab,mgmtaddr,
                                                   all
net0    dot1-tlv   rw     pvid,           none     none,vlanname,pvid,
                          linkaggr                 linkaggr,pfc,appln,
                                                   evb,etscfg,all
net0    dot3-tlv   rw     max-framesize   none     none, max-framesize,
                                                   all
net0    virt-tlv   rw     vnic            none     none,vnic,all
```

▼ How to Define TLV Values

This procedure explains how to provide values for specific TLV units. Use either of the following subcommands:

- `lldpadm set-tlvprop` to configure global TLV units.
- `lldpadm set-agenttlvprop` to configure per-agent TLV units.

1 Perform one of the following steps depending on whether you are configuring a global TLV unit or a per-agent unit.

- To configure a global TLV unit, set the appropriate TLV property to contain the values that you want to advertise.

  ```
  # lldpadm set-tlvprop -p tlv-property=value[,value,value,...] tlv-name
  ```

  where *tlv-name* is the name of the global TLV unit and *tlv-property* is a property of that TLV unit. You can assign multiple values to the property. For reference, see Table 7–2.
- To configure a per-agent TLV unit, configure the appropriate TLV property of the LLDP agent to contain the values that you want the agent to advertise,

  ```
  # lldpadm set-agenttlvprop -p tlv-property[+|-]=value[,value,value,...] -a agent tlv-name
  ```

  where *tlv-name* is the name of the agent TLV unit and *tlv-property* is a property of that TLV unit. You can assign multiple values to the property. For reference, see Table 7–3.

2 (Optional) Display the values of the TLV property that you have just configured by performing one of the following steps:

- To display the global TLV property values, use the following command:

  ```
  # lldpadm show-tlvprop
  ```
- To display the values of an agent's TLV property values, use the following command:

  ```
  # lldpadm show-agenttlvprop
  ```

Example 7–4 Specifying the System's Capabilities and the Management IP Address

This example accomplishes two objectives:

- Provides specific information about the system's capabilities to be advertised in the LLDP packet. To achieve this objective, both `supported` and `enabled` properties of the `syscapab` TLV unit must be configured.
- Provides the management IP address that is used in the advertisement.

```
# llpdadm set-tlvprop -p supported=bridge,router,repeater syscapab
# llpdadm set-tlvprop -p enabled=router syscapab
# llpdadm set-tlvprop -p ipaddr=192.168.1.2 mgmtaddr
# llpdadm show-tlvprop
TLVNAME    PROPERTY   PERM  VALUE        DEFAULT         POSSIBLE
syscapab   supported  rw    bridge,      bridge,router,  other,router,
                            router,      station         repeater,bridge,
                            repeater                     wlan-ap,telephone,
                                                         docis-cd,station,
                                                         cvlan,svlan,tpmr
syscapab   enabled    rw    router       none            bridge,router,
                                                         repeater
mgmtaddr   ipaddr     rw    192.162.1.2  none            --
```

Disabling LLDP

To disable LLDP selectively on individual ports, use one of the following commands:

- `lldpadm set-agentprop -p mode=disable` *agent*

  where *agent* is the LLDP agent and is identified by the physical link on which the agent is enabled. Thus, if you enable LLDP on `net0`, the agent is `net0`. This command disables LLDP by changing the mode of the agent.

- `lladpadm reset-agentprop`

  In this command, you do not set a value for the `mode` property. This command disables LLDP by removing the LLDP configuration from the port.

Caution – The subcommand `lldpadm reset–agentprop` entirely removes LLDP configuration from the port, If `auto-enable-agents` that is set to `no` is switched back to `yes`, LLDP behaves differently than if the agent's mode on that port were simply disabled.

To disable LLDP globally across all of the system's interfaces, perform the following steps.

1. Change the SMF LLDP property to `no`.

   ```
   # svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
   ```

2. Restart the LLDP service.

   ```
   # svcadm restart svc:/network/lldp:default
   ```

3. Disable LLDP on each port whose previous LLDP configuration is retained.

   ```
   # lldpadm set-agentprop -p mode=disable agent
   ```

Monitoring LLDP Agents

The `lldpadm show-agent` subcommand displays the complete information that is advertised by an LLDP agent. Relative to a given system, the advertisement can be information about the local system that is transmitted to the rest of the network. Or, the advertisement can be information that is received by the system from other systems on the same network.

This section describes two procedures:

- "How to Display Advertisements" on page 124
- "How to Display LLDP Statistics" on page 125

How to Display Advertisements

This procedure shows how to display the information that is being advertised by an LLDP agent. The information can be either local or remote. *Local* information comes from the local system. *Remote* information comes from other systems on the network, which is received by the local system.

- **Use the `lldpadm show-agent` subcommand with the appropriate option to display the information what you want.**
  - **To display local information that is advertised by the LLDP agent, type the following command:**

    `# lldpadm show-agent -l` *agent*
  - **To display remote information that is received by the LLDP agent, type the following command:**

    `# lldpadm show-agent -r` *agent*
  - **To display either the local or the remote information in detail, type the following command:**

    `# lldpadm show-agent -[l|r]v` *agent*

Example 7–5 Obtaining LLDP Agent Information That Is Advertised

The following example shows how to display the information that is being advertised locally or remotely by an LLDP agent. By default, the information is displayed in short form. By using the `-v` option, you can obtain verbose or detailed information.

```
# lldpadm show-agent -l net0
AGENT   CHASSISID   PORTID
net0    004bb87f    00:14:4f:01:77:5d

# lldpadm show-agent -lv net0
                      Agent: net0
         Chassis ID Subtype: Local(7)
            Port ID Subtype: MacAddress(3)
                    Port ID: 00:14:4f:01:77:5d
           Port Description: net0
               Time to Live: 81 (seconds)
                System Name: hosta.example.com
         System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
     Supported Capabilities: bridge,router
       Enabled Capabilities: router
         Management Address: 192.168.1.2
         Maximum Frame Size: 3000
               Port VLAN ID: --
               VLAN Name/ID: vlan25/25
        VNIC PortID/VLAN ID: 02:08:20:72:71:31
    Aggregation Information: Capable, Not Aggregated
                PFC Willing: --
```

```
                  PFC Cap: --
                  PFC MBC: --
               PFC Enable: --
              PFC Pending: --
Application(s) (ID/Sel/Pri): --
    Information Valid Until: 117 (seconds)

# lldpdadm show-agent -r net0
AGENT   SYSNAME   CHASSISID   PORTID
net0    hostb     0083b390    00:14:4f:01:59:ab

# llpdadm show-agent -rv net0
                    Agent: net0
         Chassis ID Subtype: Local(7)
            Port ID Subtype: MacAddress(3)
                    Port ID: 00:14:4f:01:59:ab
           Port Description: net0
               Time to Live: 121 (seconds)
                System Name: hostb.example.com
         System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
     Supported Capabilities: bridge,router
       Enabled Capabilities: router
         Management Address: 192.168.1.3
         Maximum Frame Size: 3000
               Port VLAN ID: --
               VLAN Name/ID: vlan25/25
        VNIC PortID/VLAN ID: 02:08:20:72:71:31
    Aggregation Information: Capable, Not Aggregated
                PFC Willing: --
                    PFC Cap: --
                    PFC MBC: --
                 PFC Enable: --
Application(s) (ID/Sel/Pri): --
    Information Valid Until: 117 (seconds)
```

How to Display LLDP Statistics

You can display LLDP statistics to obtain information about LLDP packets that are being advertised by the local system or by remote systems. The statistics refer to significant events that involve LLDP packet transmission and reception.

1. **To display all the statistics about LLDP packet transmission and reception, use the following command:**

   ```
   # lldpadm show-agent -s agent
   ```

2. **To display selected statistics information, use the `-o` option.**

   ```
   # lldpadm show-agent -s -o field[,field,...]agent
   ```

   where *field* refers to any field name in the output of the `show-agent -s` command.

Example 7–6 Displaying LLDP Packet Statistics

This example shows how to display information about LLDP packet advertisement.

```
# lldpadm show-agent -s net0
AGENT IFRAMES IEER IDISCARD OFRAMES OLENERR TLVDISCARD TLVUNRECOG AGEOUT
net0        9    0        0      14       0          4          5      0
```

The command output provides the following information:

- AGENT specifies the name of the LLDP agent, which is identical to the datalink on which the LLDP agent is enabled.
- IFRAMES, IEER, and IDISCARD display information about packets being received, incoming packets with errors, and incoming packets that are dropped.
- OFRAMES and OLENERR refer to outgoing packets as well as packets that have length errors.
- TLVDISCARD and TLVUNRECOG display information about TLV units that are discarded as well as TLV units that are not recognized.
- AGEOUT refers to packets that have timed out.

The example indicates that out of 9 frames received into the system, 5 TLV units are unrecognized, possibly because of noncompliance with standards. The example also shows that 14 frames were transmitted by the local system to the network.

◆ ◆ ◆ CHAPTER 8

Working With Data Center Bridging Features in Oracle Solaris

Like LLDP that is described in Chapter 7, "Exchanging Network Connectivity Information With LLDP," data center bridging (DCB) also involves information exchange with peers on the network. The information refers to configurations affecting the integrity of network packets especially in heavy traffic environments such as data centers. DCB enables efficient network traffic exchange by coordinating component configuration that relates to high speed traffic in these centers.

This chapter covers the following topics:

Overview of Data Center Bridging (DCB)

Data center bridging is a set of features that enhances traditional Ethernet networks' abilities to manage traffic especially in environments where network traffic volume and transmission rates are high. Fiber channel can be dedicated to host this type of traffic. However, using dedicated links to service only fiber channel traffic can be costly. Thus, fiber channel traffic over Ethernet (FCoE) is more commonly used. DCB features address fiber channel's sensitivity to packet loss while traversing the Ethernet network.

DCB enables peers to distinguish traffic based on priorities. By distinguishing priorities, hosts can ensure that for traffic with higher priorities, packet integrity is preserved in cases of congestion between hosts. With the DCB exchange protocol (DCBX), communicating hosts can exchange configuration information that affect high speed network traffic. The peers can then negotiate on a common configuration that ensures continuous traffic flow while preventing packet loss for those packets with high priority.

In Oracle Solaris, LLDP is used to exchange DCBX TLV units. Provided that the underlying NIC supports DCB, DCB features such as priority-based flow control (PFC) and enhanced transmission selection (ETS) can be shared with peer hosts on the network.

- Priority-based flow control (PFC) prevents packet loss by implementing a mechanism that pauses traffic flow for packets with defined class of service (CoS) priority. See "Priority-Based Flow Control" on page 129. For more information about CoS, refer to the description of for the `cos` link property in the `dladm(1M)` man page.
- Enhanced transmission selection (ETS) enables bandwidth sharing among packets based on defined CoS priority. See "Enhanced Transmission Selection" on page 134.

As with all system information that is exchanged using LLDP, two types of DCB information exists on the host: local DCB information and remote DCB information. For PFC features to be effective, these two types of DCB information for PFC on the host must be symmetric. Typically, the local host must be able to match the DCB information it receives from the peer. In Oracle Solaris systems where DCB is enabled, this ability to synchronize DCB information with the peer is also enabled.

Note – You can use DCB capabilities on your Oracle Solaris 11 system only if the physical NIC supports DCB. Further, that card must be configured to run in DCB mode.

How to Enable DCBX

Support for DCBX is automatically enabled when you enable LLDP. This procedure provides alternative manual steps in case certain automatic processes fail. In the procedure, assume that the steps are implemented on `net0`.

1. **Install the LLDP package.**

   ```
   # pkg install lldp
   ```

2. **Verify that the LLDP service is running.**

   ```
   # svcs lldp
   ```

   If the LLDP service is disabled, start the service with the following command:

   ```
   # svcadm enable svc:/network/lldp:default
   ```

3. **Ensure that the LLDP agent is running on Rx and Tx modes.**

   ```
   # lldpadm show-agentprop -p mode net0
   ```

   If the LLDP agent is not enabled on both modes, type the following:

   ```
   # lldpadm set-agentprop -p mode=both net0
   ```

   For more information, see "SMF Property for LLDP" on page 113.

For other possible configurations of the LLDP agents, see "Enabling LLDP on the System" on page 117.

4 **Ensure that the underlying NIC supports DCB.**

```
# dladm show-linkprop -p ntcs net0
```

A property value that is greater than zero (`0`) indicates that the NIC supports DCB.

Priority-Based Flow Control

PFC extends the standard PAUSE frame to include IEEE 802.1p CoS values. With PFC, instead of halting all traffic on the link when a PAUSE frame is sent, traffic is paused only for those CoS values that are enabled in the PFC frame. A PFC frame is sent for the enabled priority for which traffic needs to be paused. The sending host stops traffic for that priority while traffic for other disabled priorities are unaffected. After a time interval that is specified in the PFC frame, or after the sending host receives another PFC frame, transmission resumes for those packets. Pausing based on priority ensures that packets are not dropped for that priority. For packets without any defined priority, no PAUSE frames are sent. Thus, traffic continues to flow, and packets might be dropped during traffic congestion.

The priorities are represented by an 8–bit mask (0–7) on the `pfcmap` datalink property. The lowest bit represents priority 0 while the highest bit represents priority 7. Each bit in this mask signifies whether PFC is enabled for a corresponding priority. By default, `pfcmap` is set to `1111111`, which means that PFC is enabled on all the priorities. For any packet transmitted over a link, a PFC frame would be sent to the sending host if congestion builds up on the receiving host.

PFC-Related Datalink Properties

Aside from the `pfcmap` property, the following properties provide information about priority definitions and mappings:

- `pfcmap-lcl-effective` refers to the operative PFC mapping on the local host. This property has read-only permissions. The property can reflect either the value of the `pfcmap` property or the value of the `pfcmap-rmt-effective` property.
- `pfcmap-rmt-effective` refers to the operative PFC mapping on the remote peer. This property also has read-only permissions.

For PFC frames to be properly sent, communicating hosts must have symmetric DCB configuration information. An Oracle Solaris 11 system can automatically adjust its PFC configurations to match the PFC configurations on the remote peer.

The two listed properties indirectly indicate whether PFC information between peers are synchronized. On a datalink with matching PFC information between the local and remote

peers, the values of `pfcmap-lcl-effective` and `pfcmap-rmt-effective` are identical regardless of the value set for `pfcmap`. If the ability to synchronize is disabled on the local host, then `pfcmap-lcl-effective` would reflect the value of the local host's `pfcmap` property.

See "Obtaining PFC Configuration Information" on page 131 for examples of PFC information that these property configurations provide..

Priority-based Flow Control TLV Units

The PFC TLV unit controls the host's behavior with regards to the information that is received from the peer host. This TLV unit has only one configurable property, `willing`. By default, this property is set to `on` and enables the local host to synchronize its PFC priority definitions with the PFC definitions on the remote peer. You can prevent automatic synchronization of information for a specific agent by switching the property to `off`, as follows:

```
# lldpadm set-agenttlvprop -p willling=off -a agent pfc
```

where *agent* is identified by the datalink on which the agent is enabled.

▼ How to Customize Priority-based Flow Control for DCB

In most cases, the default configuration for PFC on is sufficient. This configuration is automatically set up when LLDP is enabled. However, to show the different options that you can use when configuring PFC, this procedure lists the manual steps for PFC configurations. The steps assume that no automatic configuration exists. To facilitate understanding of the steps, all configurations are performed on `net0`.

1 Ensure that DCBX is enabled.

See "How to Enable DCBX" on page 128.

2 (Optional) Customize which DCB feature that you want to enable.

By default, PFC, ETS, and edge virtual bridging (EVB) are enabled. Suppose that you prefer to use only PFC. Then you must remove the other two values from the `dot1–tlv` property of the LLDP agent. For a list of possible values for `dot1–tlv`, refer to Table 7–3.

```
# lldpadm set-agenttlvprop -p dot1-tlv-=etscfg,evb net0
```

3 Ensure that the datalink's `flowctrl` property is set to `pfc`.

```
# dladm show-linkprop -p flowctrl net0
```

If the property does not include `pfc` on the list of values, issue the following command:

```
# dladm set-linkprop -p flowctrl=pfc net0
```

4. **Set the `pfcmap` property as appropriate if you do not want to use the default value `11111111`.**

   For example, to enable priority only on CoS priority 6, type the following command:

   ```
   # dladm set-linkprop -p pfcmap=01000000 net0
   ```

5. **Ensure that the host can synchronize its PFC information with the PFC information on the remote peer.**

   ```
   # lldpadm show-agenttlvprop -p willing -a net0 pfc
   ```

   If the PFC TLV property `willing` is set to `off`, issue the following command:

   ```
   # lldpadm set-agenttlvprop -p willing=on -a net0 pfc
   ```

Obtaining PFC Configuration Information

This section contains several examples of information related to PFC after LLDP and DCB are configured.

The following commands display information that is related to PFC:

- `dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective` *datalink*

  This command displays the priority definitions as well as the effective PFC mappings on the datalink.

- `dladm show-phys -D pfc` *datalink*

  This command displays PFC information on the physical link with regards to enabled priorities on the NIC.

- `lldpadm show-agenttlvprop -a` *agent* `pfc`

  where *agent* is identified by the datalink on which LLDP is enabled. Thus, the LLDP agent's name is identical to the name of the datalink. This command displays the PFC TLV property that controls a host's capability to synchronize its PFC mapping with a peer.

- `lldpadm show-agent -lv -o "PFC Pending"` *agent*

  This command alerts you to a mismatch of PFC mapping information between the local host and the peer.

The following examples show the types of information that is displayed by the previously listed commands.

EXAMPLE 8–1 Displaying PFC-Related Datalink Properties

This example shows how to display the status of datalink properties that are related to priority-based flow control.

```
# dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective net0
LINK   PROPERTY              PERM   VALUE      DEFAULT    POSSIBLE
net0   pfcmap                rw     11111111   11111111   00000000-11111111
```

EXAMPLE 8–1 Displaying PFC-Related Datalink Properties *(Continued)*

```
net0   pfcmap-lcl-effective   r-     11111111   --         --
net0   pfcmap-rmt-effective   r-     01000000   --         --
```

The output indicates that the PFC mapping on the local host has the default value where all the 8 priorities are enabled. The mismatched values for `pfcmap-lcl-effective` and `pfcmap-rmt-effective` indicate that the local host has not synchronized its PFC information with the remote peer. The mismatch can be due to the property that enables synchronization being switched off. Or, the peer is not sending PFC TLV units to the network. You can confirm this configuration by typing the following command:

EXAMPLE 8–2 Displaying the Capability of the Local Host to Synchronize PFC Information

This example shows how to display the current status of the host's ability to adjust to the peer's PFC configurations.

```
# lldpadm show-agenttlvprop -a net0 pfc
AGENT   TLVNAME   PROPERTY   PERM   VALUE   DEFAULT   POSSIBLE
net0    pfc       willing    rw     off     on        on,off
```

To enable synchronization, issue the following command:

```
# lldpadm set-agenttlvprop -p willing=on -a net0 pfc

# dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective net0
LINK   PROPERTY               PERM   VALUE      DEFAULT    POSSIBLE
net0   pfcmap                 rw     11111111   11111111   00000000-11111111
net0   pfcmap-lcl-effective   r-     01000000   --         --
net0   pfcmap-rmt-effective   r-     01000000   --         --
```

In the second output, the local host has dropped its own PFC mapping (`11111111`). Instead, the host has synchronized with the peer and its effective PFC mapping is now identical to the peer's PFC mapping. With this convergence of values, hosts can successfully exchange PFC PAUSE frames.

EXAMPLE 8–3 Verifying Symmetry of PFC Information Between Host and Peer

This example show how to verify in actual running time whether PFC information is synchronized between host and peer, or whether a mismatch occurs.

```
# lldpadm show-agent -lv -o "PFC Pending" net0
PFC Pending: True
```

`PFC Pending` returns a `True` status if PFC information between the host and the peer does not converge. After the mismatch is resolved, the status of `PFC Pending` returns to `False`.

To display all the information an agent advertises, use the `verbose` option of the `lldpadm show-agent` command:

```
# lldpadm show-agent -v agent
```

EXAMPLE 8–4 Displaying CoS Priority Definitions

This example shows how to display the current CoS priority definitions on a specific datalink based on the value of the `pfcmap` property. For example, assume that `pfcmap` is configured as `01000000`. To display the corresponding priority mappings on the physical link, you would proceed as follows:

```
# dladm show-phys -D pfc net0
LINK        COS   PFC   PFC_EFFECT   CLIENTS
ixgbe0      0     YES           NO   net0,vnic1
            1     YES          YES   vnic2
            2     YES           NO   vnic3
            3     YES           NO   vnic4
            4     YES           NO   vnic5
            5     YES           NO   vnic6
            6     YES           NO   vnic7
            7     YES           NO   vnic8
```

For the physical link `net0`, priority is enabled for all the VNIC clients configured over the datalink. However, the local host adjusts its PFC mapping to the PFC mapping on the peer, as shown by the values of the `PFC_EFFECT` field, where priority is disabled on CoS `0` and `2-7`. Thus, no PFC frames would be exchanged for traffic on all VNICs except `vnic2` regardless of the availability of resources. With this configuration, packet drops are allowed on traffic that flows on all VNICs except `vnic2`. For traffic on `vnic2`, PFC PAUSE frames are sent when traffic congestion occurs to prevent packet loss on this client.

Application TLV Units

Application TLV units contain information about the priority to be used for an application on the host. The priority is defined in the Application Priority Table. Each entry on the table contains the application's name and the priority assigned to the application. The application TLV uses the table for transmitting application priority information with other hosts.

An entry on the table uses the following format:

protocol-id/*selector*/*priority*

The pair *protocol-id*/*selector* identifies the application. *Priority* contains a value from 0 to 7 that identifies the priority for a corresponding application.

To exchange this information about an applications' priority with other hosts, you set an application TLV as follows:

```
# lldpadm set-agenttlvprop -p property=value -a agent appln
```

For example, for FCoE traffic, the protocol ID is `0x8906` and the selector ID is `1`. Suppose that the priority `4` is assigned to this application. Based on Table 7–3 that lists the parameters for setting an application TLV, type, the following command:

```
# lldpadm set-agenttlvprop -p apt=8906/1/4 -a net0 appln
# lldpadm show-agenttlvprop -a net0 appln
AGENT   TLVNAME   PROPERTY   PERM   VALUE      DEFAULT   POSSIBLE
net0    appln     apt        rw     8906/1/4   --        --
```

Enhanced Transmission Selection

ETS is a DCB feature that allows allocation of bandwidth on a NIC to applications based on their DCB priority. The DCB priority is a VLAN header with a 3 bit priority field. The priority field's value differentiates Ethernet packets in the network. DCB uses the priority value, also called the 802.1p priority, to associate traffic with other DCB properties such as PFC configuration and link bandwidth. You configure DCB to set specific bandwidth to be allocated to packets depending on their priority values.

To use ETS, the NIC must support DCB and run in DCB mode.

ETS-Related Datalink Properties

The properties of datalinks that refer to PFC information applies to the prevention of packet loss based on the CoS priorities defined for the packets. Properties that refer to ETS information applies to the allocation of shared bandwidth for packets based on the same CoS priorities You configure ETS on the following datalink properties:

- `cos` specifies the class of service for a datalink. The property represents the Ethernet priority. The property's value, which ranges from 0 to 7, is applied to outbound packets on the datalink. The value is set on the VLAN tag of the outbound packets. If this property is set on the physical link itself, then the priority applies only to the traffic on that link's primary client. The priority is not set on other secondary clients such as VNICs. By default, `cos` is set to `0` if the NIC is running on DCB mode, or if the link is a VLAN.
- `etsbw-lcl` indicates the ETS bandwidth that is allocated on the TX side for the datalink. This property is configurable only if the underlying physical NIC has DCB capabilities and supports ETS. You set the value by specifying the percentage of total bandwidth of the underlying NIC that you want to allocate to a secondary datalink or client. You can set this property provided that the link's `cos` is not set to zero (`0`).

Note – ETS currently is not supported on physical links on DCB mode that are configured into an aggregation.

The bandwidth percentage that is defined on `etsbw-lcl` is not a reserved amount only for that secondary client. If the allocated bandwidth is not used, then it can be used by other clients that are similarly configured. Further, the bandwidth allocation is enforced only on the transmission side of the host's traffic.

In addition to the properties in the previous list, the following read-only properties provide information about the bandwidth data that is exchanged between the local host and its peer:

- `etsbw-lcl-advice` specifies the recommended bandwidth share. This recommended bandwidth for a datalink is sent by the remote peer to the local host.
- `etsbw-lcl-effective` refers to the actual bandwidth share that is implemented on the local host's datalink. The property can reflect either the value of the `etsbw-lcl` property or the value of the `etsbw-lcl-advice` property.
- `estbw-rmt-effective` refers to the bandwidth share that is configured on the remote peer.

For the appropriate bandwidth to be used for packets with specific priorities, symmetric or synchronized ETS information between the communicating hosts is preferable. Specifically, the local system's ability to adjust its bandwidth share to the value of `etsbw-lcl-advice` is desirable. An Oracle Solaris 11 system can automatically adjust its ETS configurations to match the ETS configurations on the remote peer.

The `estbw-lcl-effective` property indirectly indicate if the ability of the local host to match ETS information with the peer is enabled or not. If the property's value matches the value of `etsbw-lcl-advice`, then the ability is enabled. Otherwise, the values of the `etsbw-lcl-effective` and `etsbw-lcl` properties would be identical.

Enhanced Transmission Selection TLV Units

The ETS TLV unit `etscfg` controls the host's behavior with regards to the information that is received from the peer host. This TLV unit has only one configurable property, `willing`. By default, this property is set to `on` and enables the local host to synchronize its ETS configuration with the ETS configuration of the remote peer. If you need to prevent synchronization of information for a specific agent, set the `willing` property to `off` as follows:

```
# lldpadm set-agenttlvprop -p willing=off -a agent etscfg
```

where *agent* is identified by the datalink on which the agent is enabled.

▼ How to Customize Enhanced Transmission Selection for DCB

In most cases, the default configuration for ETS on the system is sufficient. This configuration is automatically set up when LLDP is enabled, DCB is supported by the underlying link, and the underlying link is running on DCB mode. However, to show the different options that you can use when configuring ETS, this procedure lists the manual steps for ETS configuration. The steps assume that no automatic configuration exists and that the configurations are performed on the virtual client `vnic1`. The virtual client is configured over `net0`, which is the LLDP agent.

1. **Ensure that DCBX is enabled.**

   See "How to Enable DCBX" on page 128.

2. **(Optional) Customize which DCB feature that you want to enable.**

   By default, PFC, ETS, and edge virtual bridging (EVB) are enabled. Suppose that you prefer to disable EVB. Then remove the other two from the `dot1–tlv` property of the LLDP agent.

   ```
   # lldpadm set-agenttlvprop -p dot1-tlv-=evb net0
   ```

3. **Set a CoS priority definition to the VNIC.**

   ```
   # dladm set-linkprop -p cos=value vnic1
   ```

4. **Set the VNIC's bandwidth that is shared with the total bandwidth of the physical link.**

   ```
   # dladm set-linkprop -p etsbw-lcl=value vnic1
   ```

   The value that you assign to the `etsbw-lcl` property represents a percentage of the total bandwidth capacity of the underlying link. The sum of all the allocated bandwidth values that you assign to the clients must not exceed 100 percent.

5. **Verify that the host can synchronize its ETS information with the ETS information of the remote peer.**

   ```
   # lldpadm show-agenttlvprop -p willing -a net0 etscfg
   ```

   If the `willing` property is set to `off`, issue the following command.

   ```
   # lldpadm set-agenttlvprop -p willing=on -a net0 etscfg
   ```

Obtaining ETS Configuration Information

This section contains several examples of information related to ETS configuration after LLDP and DCB are configured.

The following commands display information about ETS configuration:

- `dladm show-linkprop -p etsbw-lcl,etsbw-advise,etsbw-lcl-effective,etsbw-rmt-effective` *datalink*

  This command displays the bandwidth allocation definitions as well as the effective allocation that is implemented on the datalink.

- `dladm show-phys -D ets` *datalink*

  This command displays ETS configuration on the physical link with regard to bandwidth allocation and distribution across the link.

- `lldpadm show-agenttlvprop -a` *agent* `etscfg`

  where *agent* is identified by the datalink on which LLDP is enabled. This command displays the ETS TLV property that controls a host's capability to synchronize ETS information with a peer.

The following examples show the types of information that is displayed by the listed commands.

EXAMPLE 8–5 Displaying ETS-Related Datalink Properties

This example shows how to display the status of datalink properties that are related to enhanced transmission selection.

```
# dladm show-linkprop -p cos,etsbw-lcl,etsbw-lcl-advise, \
etsbw-lcl-effective,etsbw-rmt-effective vnic1
LINK    PROPERTY              PERM  VALUE      DEFAULT    POSSIBLE
vnic1   cos                   rw    2          0          0-7
vnic1   etsbw-lcl             rw    20         0          --
vnic1   etsbw-lcl-advise      r-    --         --         --
vnic1   etsbw-lcl-effective   r-    --         --         --
vnic1   etsbw-rmt-effective   r-    --         --         --
```

The output shows that `vnic1` is configured to have a bandwidth share of 20% of the total bandwidth available for the physical link. The VNIC's 802.1p priority, indicated by the `cos` property, is set to two.

EXAMPLE 8–6 Displaying the Capability of the Local Host to Synchronize ETS Information

This example shows how to display the current status of the local host's ability to adjust to the peer's ETS configurations.

```
# lldpadm show-agenttlvprop -a net0 etscfg
AGENT   TLVNAME   PROPERTY   PERM   VALUE   DEFAULT   POSSIBLE
net0    etscfg    willing    rw     off     on        on,off
```

To enable synchronization, issue the following command:

```
# lldpadm set-agenttlvprop -p willing=on -a net0 etscfg

# dladm show-linkprop -p etsbw-lcl,etsbw-lcl-advise, \
etsbw-lcl-effective,etsbw-rmt-effective vnic0
LINK    PROPERTY              PERM  VALUE      DEFAULT    POSSIBLE
vnic1   cos                   rw    2          0          0-7
vnic1   etsbw-lcl             rw    20         0          --
vnic1   etsbw-lcl-advise      r-    15         --         --
vnic1   etsbw-lcl-effective   r-    15         --         --
vnic1   etsbw-rmt-effective   r-    25         --         --
```

Although `estbw-lcl` was set to 20% for `vnic1`, the VNIC's effective bandwidth share is 15% to match the advised bandwidth received from the peer. The adjustment occurs as a consequence of switching the willing property of the `etscfg` TLV unit to on.

The following example shows priority mappings on the physical link:

```
# dladm show-phys -D ets net0
LINK       COS   ETSBW   ETSBW_EFFECT   CLIENTS
ixgbe0     0     20      20             <default,mcast>,net0
```

```
1     15      15            vnic2
2     20      20            vnic1
3     30      30            vnic5
4     15      15            vnic3
5     0       0             vnic4
6     0       0             vnic6
7     0       0             vnic7
```

In this example, different VNICs are set with their own corresponding `cos` values. Based on the previous output, `vnic1`'s `cos` property is set to two, Under the `ETSBW` field, the client `vnic1` has an effective bandwidth share of 15% to match the advised value that is received from the peer, shown under the `ETSBW_EFFECT` field, The example also shows that the biggest share of bandwidth is allocated to `vnic5`. Note that an allocation of 0% to `vnic4`, `vnic6`, and `vnic7` does not indicate that the clients have no share of the bandwidth at all. Rather, these clients do not receive bandwidth if the other clients are using their allocated bandwidth.

◆ ◆ ◆ CHAPTER 9

Edge Virtual Bridging in Oracle Solaris

This chapter describes edge virtual bridging features (EVB). EVB further extends the information exchange features that were described in the following chapters:

- Chapter 7, "Exchanging Network Connectivity Information With LLDP"
- Chapter 8, "Working With Data Center Bridging Features in Oracle Solaris"

EVB enables the exchange of information relative to virtual links on the system. The following topics are covered:

Overview of Edge Virtual Bridging

Edge virtual bridging is an evolving IEEE standard for a host to exchange virtual link information with an external switch. With EVB, more information about virtual link configurations can be advertised on the network beyond, for example, bandwidth share or priority definitions for physical links that DCB features provide.

In general, EVB can be used for the following operations:

- Enable reflective relay on the external bridge port. See "Reflective Relay Capability" on page 140.
- Automate virtual port configuration on the bridge. See "Automatic Virtual Port Configuration on the Bridge" on page 140.

To understand the mechanism of EVB, note the following terminology used with EVB.

- **Station** refers to the system or host.
- **Bridge** refers to the external switch that is connected to the station.
- **Virtual station instance (VSI)** refers to a VNIC that is configured on the station.

- **Virtual machine (VM)** is a general term to refer to zones, Oracle VM VirtualBox, and other software-implemented machines on the system.

The following sections describe in detail the functions of EVB.

Reflective Relay Capability

With network virtualization, a station can be configured with multiple virtual network interface cards (VNICs) over a single physical NIC. The VNICs are assigned to virtual machines on the station. In this setup, communication among the virtual machines can occur without packets having to leave the station. Instead, packets are routed from one VM to another VM by the virtual switch of the physical link. For an illustration of a system configuration that includes, VNICs, a virtual switch, and virtual machines, see "Components of Network Virtualization" in *Using Virtual Networks in Oracle Solaris 11.1*.

In certain cases, internal communication between VMs might require the use of an external bridge. For example, internal communication might need to be subjected to access control lists (ACLs) that are configured on the external bridge. The packets from the originating VM would therefore exit the station through a port to the external bridge. Those packets are then sent from the bridge back to the station to the receiving VM.

By default, a bridge cannot send packets on the same port where the packets are received. Thus, for communications between VMs that uses an external bridge, the bridge must have reflective relay capability. This capability enables the bridge to relay packets from the sending VM back to the receiving VM on the same link as it received the packets.

EVB defines a new organization-specific LLDP TLV unit to inform network peers about reflective relay capability, while the EVB TLV unit serves as the vehicle for the information The information exchange consists of a station first requesting the bridge to enable reflective relay if it is supported on the bridge. The bridge enables the capability, if it is supported, and informs the requesting host of that capability. If the bridge does not support reflective relay, then a disabled status is sent back to the station. In that case, communications between virtual machines simply use the physical link's virtual switch.

Automatic Virtual Port Configuration on the Bridge

LLDP and DCBX enable a station to exchange configuration information with the nearest bridge. With this exchange, the bridge can detect, for example, priorities that have been defined for traffic classes on the station. Based on this information, the bridge is automatically configured to process packets based on their 802.1p priority value.

Without the information exchange and automatic configuration, the bridge would need to be manually configured separately from the station to mirror the station configuration on traffic class priorities. Manual configuration risks bridge misconfiguration and might cause inconsistencies between the station and the bridge.

With EVB, the exchange mechanism is extended to include information about the station's VSIs to the bridge. In this manner, the configuration of the VNICs can be extended to the bridge. For example, suppose that a VNIC has been configured with a specific bandwidth limit. With EVB, the bridge can enforce the bandwidth limit on packets that are destined for that VNIC.

EVB Components for VSI Information Exchange

The following EVB components enable the station to advertise VSI information to the bridge:

- The **VSI profile** consists of link properties that have been configured for the specific VNIC. Thus, a station can have as many VSI profiles as there are configured VNICs.
- A VSI identifier consists of a **VSI Type ID** and **VSI Version ID** pair that uniquely identifies a VSI profile.
- The **VSI Manager** manages the multiple VSI profiles on the station by mapping the VSI Type ID-VSI Version ID identifier with a specific set of VNIC properties.
- **VSI Manager ID** identifies the VSI Manager that is relevant to a specific VSI Type ID - VSI Version pair. The VSI Manager ID is represented as an IPv6 address.

The combined VSI Manager ID, VSI Type ID, and VSI Version constitute a tuple that identifies a set of properties of a specific VNIC..

VSI information is exchanged by using the VSI discovery and configuration protocol (VDP), while the VDP TLV unit serves as the vehicle for the information. The bridge receives the VDP TLV unit from the station. The bridge then uses the tuple contained in the TLV unit to obtain the set of properties that are associated with the VSI. After the bridge obtains the properties for the VSI profile or type, then the bridge can apply the property configurations on packets for that VSI.

Before VSI information can be advertised to the bridge, the following requirements must be met:

- The station must know what VSI Manager ID to use when sending VSI discovery protocol requests.
- The bridge must recognize and support the VSI Manager ID that is sent by the station.

EVB Support in Oracle Solaris

Currently, no defined standards exist for defining VSI profiles, for example, the specific properties that should be included in a profile. Furthermore, the definition of VSI types is closely linked to a VSI Manager ID, which is typically vendor-specific.

Oracle Solaris defines a VSI Manager by using a 3-byte encoding, `oracle_v1`. This VSI Manager supports the following datalink properties:

- Bandwidth limits

- Link speed of the underlying link
- Traffic Class
- Maximum transmission unit (MTU) of the VNIC

The `oracle_v1` encoding is defined as follows:

Bits	Properties
0-4	**Link Bandwidth Limit** 00000-10100 : 0-100% of link speed in increments of 5%. rest: reserved
5–7	**Link Speed** 000 - Unknown 001 - 10 Mbps 010 - 100 Mbps 011 - 1 Gbps 100 - 10 Gbps 101 - 40 Gbps 110 - 100 Gbps 111 - Reserved
8–12	**Reserved**
13-15	**Traffic Class (0-7)**
16-17	**Link MTU** 00 - 1500 bytes 01 - 9000 bytes 10 - Custom 11 - Reserved

The 3–byte encoding is directly used as the VSI Type ID in Oracle Solaris

Thus, in Oracle Solaris, the tuple that is advertised to the bridge is the Oracle VSI Manager and the combined VSI Type ID-VSI Version ID pair. The mechanism for the exchange of VSI information follows the same process described in "EVB Components for VSI Information Exchange" on page 141. The bridge is configured to recognize the Oracle VSI Manager. The bridge then uses the Oracle VSI Manager ID and the combined VSI Type ID-VSI Version ID to

obtain the set of properties that are associated with the VSI profile. After the bridge obtains the property information, the bridge can apply the property configurations on packets for that VNIC.

An Oracle organization-specific OUI TLV unit is sent after the transmission of the VSI Manager ID TLV. The OUI TLV indicates any encoding if any that is used for the VSI Manager ID that it follows. If a bridge recognizes the Oracle-defined VSI Manager ID, the bridge includes that TLV unit when replying to the requesting station. The absence of the Oracle-specific TLV unit in the bridge's response indicates that the Oracle VSI Manager is not recognized nor supported on the switch.

EVB-Related Datalink Properties

The following is a list of configurable datalink properties that are related to EVB:

- `vsi-mgrid` specifies the VSI Manager ID that is set either for the physical link or the VNIC. In Oracle Solaris, this property is associated with `ORACLE_VSIMGR_V1`, the default VSI Manager ID.

  If you prefer to use an IPv6 address, then you must also define the VSI Type ID and VSI Version ID. Otherwise, the tuple will not be recognized by Oracle Solaris. Further, you must also manually configure the appropriate datalink properties that correspond to a VSI Type ID-VSI Version ID/VSI Manager ID tuple.

  Preferably, you should use the default Oracle VSI Manager ID when using EVB. In this manner, the Oracle VSI Manager can automatically generate VSI Type IDs and VSI Version IDs for the station's VSI profiles.
- `vsi-mgrid-enc` indicates the encoding that is associated with the VSI Manager ID. By default, this property is set to `oracle_v1`. If you do not want to associate the `oracle_v1` with the VSI Manager ID, set this property to the value `none`.
- `vsi-typeid` specifies a VSI Type ID. A VSI Type ID pairs with a VSI Version ID to be associated with a VSI profile. This 3-byte value is automatically generated if you use the default values for `vsi-mgrid` and `vsi-mgrid-enc`. Otherwise, you must explicitly specify a value for this property.
- `vsi-vers` specifies a VSI Version ID. The VSI Version ID pairs with a VSI Type ID to be associated with a VSI profile. This 1-byte value is automatically generated if you use the default values for `vsi-mgrid` and `vsi-mgrid-enc`. Otherwise, you must explicitly specify a value for this property.

Note – All of these properties can be configured manually on all VNICs, However, only the `vsi-mgrid` and `vsi-mgrid-enc` properties can be configured on the physical link.

In addition to the properties in the previous list, the following read-only properties provide information about the actual EVB configuration that is operative on the system:

- `vsi-mgrid-effective` specifies the VSI Manager ID on a virtual link or VNIC.
- `vsi-mgrid-enc-effective` refers to the VSI Manager ID encoding that is used for a virtual link or VNIC, and which is the basis of the VSI Manager ID.
- `vsi-typeid-effective` specifies the effective VSI Type ID on a virtual link or VNIC.
- `vsi-vers-effective` specifies the effective VSI Version on a link.

Using EVB On the Station

To use EVB on your station, you must install the EVB package. Type the following command:

```
# pkg install evb
```

Preferably, you should accept the default EVB configuration that is automatically enabled after package installation. The EVB configuration is based on using the Oracle VSI Manager for enabling EVB. By accepting the default EVB configuration, the station can immediately exchange VSI information with the bridge about any VNIC that you have configured on the station.

The following example displays EVB-related properties on the physical link:

```
# dladm show-linkprop -p vsi-mgrid,vsi-mgrid-enc
LINK     PROPERTY            PERM VALUE        DEFAULT      POSSIBLE
net4     vsi-mgrid           rw   --           ::           --
net4     vsi-mgrid-enc       rw   --           oracle_v1    none,oracle_v1
```

The output displays the default configuration of EVB in Oracle Solaris 11. By using the `oracle_v1` encoding, the Oracle VSI Manager manages VSIs and their datalink properties that are recognized and supported by the Oracle VSI Manager.

If you do not want to use the default configuration, change the encoding to `none`.

```
# dladm set-linkprop -p vsi-mgrid-enc=none net4
```

Thereafter, you must manually provide the IPv6 address to be used as the VSI Manager ID, define VSI Type IDs, all the other EVB-related components and their properties.

The following example displays EVB-related properties on a VSI or VNIC.

```
# dladm show-linkprop vnic0
LINK     PROPERTY                 PERM VALUE      DEFAULT  POSSIBLE
...
vnic0    vsi-typeid               rw   --         --       --
vnic0    vsi-typeid-effective     r-   65684      --       --
vnic0    vsi-vers                 rw   --         --       --
vnic0    vsi-vers-effective       r-   0          --       --
vnic0    vsi-mgrid                rw   --         --       --
vnic0    vsi-mgrid-effective      r-   ::         --       --
vnic0    vsi-mgrid-enc-effective  r-   oracle_v1  --       --
...
```

The output displays the values based on using the Oracle VSI Manager. The VSI's effective encoding of its VSI Manager ID is `oracle_v1`. In turn, the Type ID `65684` is automatically generated and effective for `vnic0`.

The following example shows information about the VDP state for physical Ethernet links if EVB is enabled on the station. To display information only for a single link, specify that link in the command. Otherwise, VDP information for all Ethernet links is displayed.

```
# dladm show-ether -P vdb
VSI     LINK   VSIID             VSI-TYPEID VSI-STATE   CMD-PENDING
vnic0   net4   2:8:20:2c:ed:f3   65684/0    TIMEDOUT    NONE
vnic1   net4   2:8:20:df:73:77   65684/0    TIMEDOUT    NONE
```

The output shows that two VSIs are configured over the link `net4`. Their specific VSI IDs refer to their respective MAC addresses. Based on the default values of `vsi-mgrid`, both VSIs have the same VSI Type ID, which is 65684.

To obtain statistics about outgoing or incoming VDP packets, use the following command:

```
# dlstat show-ether -P vdb
```

◆ ◆ ◆ CHAPTER 10

Integrated Load Balancer (Overview)

This chapter describes the Integrated Load Balancer (ILB), a feature of Oracle Solaris. ILB provides Layer 3 and Layer 4 load-balancing capabilities for Oracle Solaris installed on SPARC and x86 based systems. ILB intercepts incoming requests from clients, decides which back-end server should handle the request based on load-balancing rules, and then forwards the request to the selected server. ILB performs optional health checks and provides the data for the load-balancing algorithms to verify if the selected server can handle the incoming request. By performing the above functionalities, ILB spreads work load directed to the server across multiple servers. This can improve reliability, minimize response time and in general improve performance of the server.

The following topics are discussed:

ILB Features

The key features of ILB include:

- Supports stateless Direct Server Return (DSR) and Network Address Translation (NAT) modes of operation for IPv4 and IPv6
- Enables ILB administration through a command-line interface (CLI)
- Provides server monitoring capabilities through health checks

The following list describes the additional features of ILB:

- **Enables clients to ping virtual IP (VIP) addresses** – ILB can respond to Internet Control Message Protocol (ICMP) echo requests to IP address of virtual service from clients. ILB provides this capability for DSR and NAT modes of operation.
- **Enables you to add and remove servers from a server group without interrupting service** – You can dynamically add and remove servers from a server group, without interrupting existing connections established with the back-end servers. ILB provides this capability for the NAT mode of operation.
- **Enables you to configure session persistence (stickiness)** – For many applications, it is important that a series of connections, packets, or both, from the same client are sent to the same back-end server. You can configure session persistence (that is, source address persistence) for a virtual service by using the `-p` option and specifying the `pmask` in the subcommand `ilbadm create-rule`. For more information, see "How to Create an ILB Rule" on page 177. After a persistent mapping is created, subsequent requests for connections, packets, or both, to a virtual service with a matching source IP address of the client are forwarded to the same back-end server. The prefix length in Classless Inter-Domain Routing (CIDR) notation is a value between `0-32` for IPv4 and `0-128` for IPv6. The support for session persistence is available for DSR and NAT modes of operation.
- **Enables you to perform connection draining** – ILB provides support for this capability only for servers of NAT-based virtual services. This capability prevents new connections from being sent to a server that is disabled. This feature is useful for shutting down the servers without disrupting the active connections or sessions. Existing connections to the server continue to function. After all the connections to that server terminate, the server can then be shut down for maintenance. After the server is ready to handle requests, the server is enabled so that the load balancer can forward new connections to it. This feature enables you to shut down servers for maintenance without disrupting active connections or sessions.
- **Enables load-balancing of TCP and UDP ports** – ILB can load balance all ports on a given IP address across different sets of servers without requiring you to set up explicit rules for each port. ILB provides this capability for DSR and NAT modes of operation.
- **Enables you to specify independent ports for virtual services within the same server group** – With this feature, you can specify different destination ports for different servers in the same server group for the NAT mode of operation.
- **Enables you to load balance a simple port range** – ILB can load balance a range of ports on the VIP to a given server group. For convenience, you can conserve IP addresses by load-balancing different port ranges on the same VIP to different sets of back-end servers. Also, when session persistence is enabled for NAT mode, ILB sends requests from the same client IP address for different ports in the range to the same back-end server.
- **Enables port range shifting and collapsing** – Port range shifting and collapsing depend on the port range of a server in a load-balancing rule. So, if the port range of a server is different from the VIP port range, port shifting is automatically implemented. If the server port range is a single port, then port collapsing is implemented. These features are provided for the NAT mode of operation.

ILB Components

ILB has three major components:

- `ilbadm` CLI – You can use the command-line interface to configure load-balancing rules, perform optional health checks, and view statistics.
- `libilb` configuration library – `ilbadm` and third-party applications can use the functionality implemented in `libilb` for ILB administration.
- `ilbd` daemon – This daemon performs the following tasks:
  - Manages persistent configuration across reboots and package updates.
  - Provides serial access to the ILB kernel module by processing the configuration information and sending it to the ILB kernel module for execution
  - Performs health checks and sends the results to the ILB kernel module so that the load distribution is properly adjusted

ILB Operation Modes

ILB supports stateless Direct Server Return (DSR) and Network Address Translator (NAT) modes of operation for IPv4 and IPv6, in single-legged and dual-legged topologies.

- Stateless DSR topology
- NAT mode (full-NAT and half-NAT) topology

Direct Server Return Topology

In DSR mode, ILB balances the incoming requests to the back-end servers, but lets the return traffic from the servers to the clients bypass it. However, you can also set up ILB to be used as a router for a back-end server. In this case, the response from the back-end server to the client is routed through the system that is running ILB. ILB's current implementation of DSR does not provide TCP connection tracking (meaning that it is stateless). With stateless DSR, ILB does not save any state information of the processed packets, except for basic statistics. Because ILB does not save any state in this mode, the performance is comparable to the normal IP forwarding performance. This mode is best suited for connectionless protocols.

Advantages:

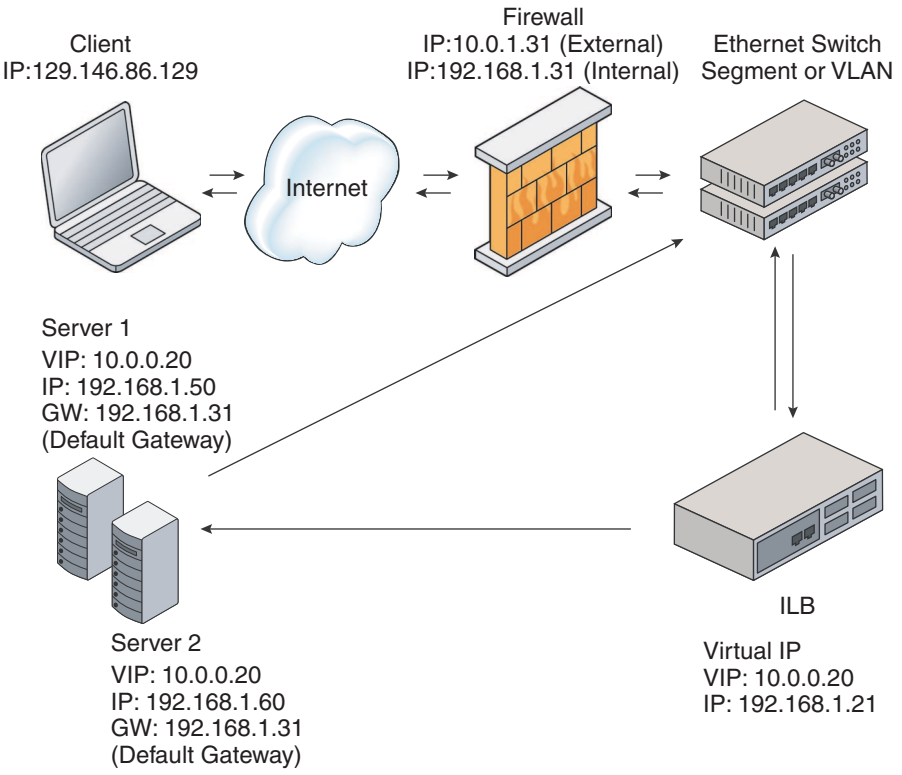
- Better performance than NAT because only the destination MAC address of packets is changed and servers respond directly to clients.
- There is full transparency between the server and the client. The servers see a connection directly from the client IP address and reply to the client through the default gateway.

Disadvantages:

- The back-end server must respond to both its own IP address (for health checks) and the virtual IP address (for load-balanced traffic).
- Because the load balancer maintains no connection state (meaning that it is stateless), adding or removing servers will cause connection disruption.

The following figure shows the implementation of ILB using the DSR topology.

FIGURE 10–1 Direct Server Return Topology



In this figure, both back-end servers are in the same subnet (192.168.1.0/24) as the ILB box. The servers are also connected to the router so that they can reply directly back to clients after getting a request forwarded by the ILB box.

Network Address Translator Topology

ILB uses NAT in stand-alone mode strictly for load-balancing functionality. In this mode, ILB rewrites the header information and handles the incoming as well as the outgoing traffic. ILB

operates in both the half-NAT and full-NAT modes. However, full-NAT also rewrites the source IP address, making it appear to the server that all connections are originating from the load balancer. NAT does provide TCP connection tracking (meaning that it is stateful). NAT mode provides additional security and is best suited for Hypertext Transfer Protocol (HTTP) (or Secure Sockets Layer (SSL)) traffic.

Advantages:

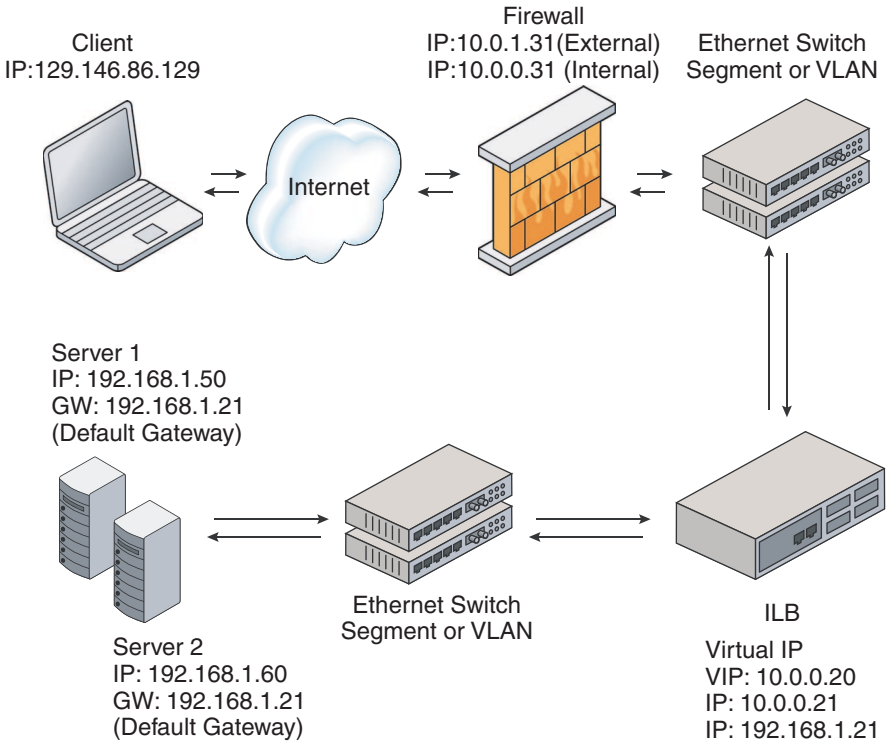
- Works with all back-end servers by changing the default gateway to point to the load balancer.
- Because the load balancer maintains the connection state, adding or removing servers without connection disruption is possible.

Disadvantages:

- Slower performance than DSR because processing involves manipulation of the IP header and servers send responses to the load balancer.
- All the back-end servers must use the load balancer as a default gateway.

The general implementation of the NAT topology as shown in the following figure.

FIGURE 10–2 Network Address Translation Topology



In this case, all requests to the VIP go through the ILB box and are forwarded to the back-end servers. All the replies from the back-end servers pass through the ILB box for NAT.

Caution – The NAT code path that is implemented in ILB differs from the code path that is implemented in the IP Filter feature of Oracle Solaris. Do *not* use both of these code paths simultaneously.

Half-NAT Load-Balancing Topology

In the half-NAT mode of ILB operation, ILB rewrites only the destination IP address in the header of the packets. If you are using the half-NAT implementation, you cannot connect to a virtual IP (VIP) address of the service from the same subnet on which the server resides. The following table shows the IP addresses of the packets flowing between client and ILB, and between ILB and back-end servers.

TABLE 10–1 Request Flow and Response Flow for the Half-NAT Implementation When the Server and Client are on Different Networks

	Request Flow	Source IP Address	Destination IP Address
1.	Client –> ILB	Client	VIP of ILB
2.	ILB –> Server	Client	Server
	Response Flow		
3.	Server –> ILB	Server	Client
4.	ILB –> Client	VIP of ILB	Client

If you connect the client system to the same network as that of the servers, the intended server responds directly to the client. The fourth step does not occur. Hence, the source IP address for the server response to the client is invalid. When the client sends a connection request to the load balancer, the response occurs from the intended server. Henceforth, the client's IP stack correctly drops all the responses.

In that case, the request flow and response flow proceed as shown in the following table.

TABLE 10–2 Request Flow and Response Flow for the Half-NAT Implementation When the Server and Client are on the Same Network

	Request Flow	Source IP Address	Destination IP Address
1.	Client –> ILB	Client	VIP of ILB
2.	ILB –> Server	Client	Server
	Response Flow		
3.	Server –> Client	Server	Client

Full-NAT Load-Balancing Topology

In the full-NAT implementation, the source and destination IP addresses are rewritten to ensure that the traffic goes through the load balancer in both directions. The full-NAT topology makes it possible to connect to the VIP from the same subnet that the servers are on.

The following table depicts the IP addresses of the packets flowing between a client and ILB, and between ILB and a back-end server using the full-NAT topology. No special default route using the ILB box is required in the servers. But note that the full-NAT topology requires the administrator to set aside one or a range of IP addresses to be used by ILB as source addresses to communicate with the back-end servers. Assume that the addresses used belong to subnet C. In this scenario, the ILB behaves as a proxy.

TABLE 10–3 Request Flow and Response Flow for the Full-NAT Implementation

	Request Flow	Source IP Address	Destination IP Address
1.	Client –> ILB	Client	VIP of ILB
2.	ILB –> Server	Interface address of the load balancer (subnet C)	Server
	Response Flow		
3.	Server –> ILB	Server	Interface address of the ILB (subnet C)
4.	ILB –> Client	VIP of ILB	Client

How ILB Works?

This section describes the working of ILB, how it processes a request from a client to the VIP, how it forwards the request to a back-end server and how it processes the response.

Client-to-server packet processing:

1. ILB receives an incoming request that is sent by the client to a VIP address and matches the request to a load-balancing rule.
2. If ILB finds a matching load-balancing rule, it uses a load-balancing algorithm to forward the request to a back-end server depending on the mode of operation.
   - In DSR mode, ILB replaces the MAC header of the incoming request with the MAC header of the selected back-end server.
   - In half-NAT mode, ILB replaces the destination IP address and the transport protocol port number of the incoming request with that of the selected back-end server.
   - In full-NAT mode, ILB replaces the source IP address and the transport protocol port number of the incoming request with the load-balancing rule's NAT source address. ILB also replaces the destination IP address and the transport protocol port number of the incoming request with that of the selected back-end server.
3. ILB forwards the modified incoming request to the selected back-end server.

Server-to-client packet processing:

1. The back-end server sends a reply to ILB in response to the incoming request from the client.
2. ILB's action after receiving the response from the back-end server is based on the mode of operation.

- In DSR mode, the response from the back-end server bypasses ILB and goes directly to the client. However, if ILB is also used as a router for the back-end server, then the response from the back-end server to the client is routed through the system running ILB.
- In half-NAT mode and full-NAT mode, ILB matches the response from the back-end server to the incoming request, and replaces the changed IP address and the transport protocol port number with that of the original incoming request. ILB then forwards the response to the client.

ILB Algorithms

ILB algorithms control traffic distribution and provide various characteristics for load distribution and server selection. ILB provides the following algorithms for the two modes of operation:

- Round-robin – In a round-robin algorithm, the load balancer assigns the requests to a server group on a rotating basis. After a server is assigned a request, the server is moved to the end of the list.
- *src IP* hash – In the source IP hash method, the load balancer selects a server based on the hash value of the source IP address of the incoming request.
- *src-IP, port* hash – In the source IP, port hash method, the load balancer selects a server based on the hash value of the source IP address and the source port of the incoming request.
- *src-IP, VIP* hash – In the source IP, VIP hash method, the load balancer selects a server based on the hash value of the source IP address and the destination IP address of the incoming request.

Service Management Facility

ILB is managed by the Service Management Facility (SMF) service `svc:/network/loadbalancer/ilb:default`. For an overview of SMF, see Chapter 1, "Managing Services (Overview)," in *Managing Services and Faults in Oracle Solaris 11.1*. For step-by-step procedures that are associated with SMF, see Chapter 2, "Managing Services (Tasks)," in *Managing Services and Faults in Oracle Solaris 11.1*.

ILB Command-Line Interface

The ILB command–line interface is located in the `/usr/sbin/ilbadm` directory. The CLI includes subcommands to configure load-balancing rules, server groups, and health checks. It also includes subcommands to display statistics as well as view configuration details. The subcommands can be divided into two categories:

- **Configuration subcommands** – These subcommands enable you to perform the following tasks:
  - Create and delete load-balancing rules
  - Enable and disable load-balancing rules
  - Create and delete server groups
  - Add and remove servers from a server group
  - Enable and disable back-end servers
  - Create and delete server health checks for a server group within a load-balancing rule

  Note – To administer the configuration subcommands, you require privileges. The privileges are obtained through the role-based access control (RBAC) of Oracle Solaris. To create the appropriate role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

- **View subcommands** – These subcommands enable you to perform the following tasks:
  - View configured load-balancing rules, server groups, and health checks
  - View packet forwarding statistics
  - View the NAT connection table
  - View health check results
  - View the session persistence mapping table

  Note – You do not need privileges to administer the view subcommands.

For a list of `ilbadm` subcommands, see "ILB Command and Subcommands" on page 156. For more detailed information about `ilbadm` subcommands, refer to the `ilbadm(1M)` man page.

ILB Command and Subcommands

You can use `ilbadm` and its subcommands to manipulate the load-balancing rules. For more detailed information about `ilbadm` subcommands, refer to the `ilbadm(1M)` man page.

TABLE 10–4 ILB Subcommands Used to Manipulate the Load-balancing Rules

ILB Subcommand	Description
`ilbadm create-rule`	Creates a `rule name` with the given characteristics.
`ilbadm show-rule`	Displays the characteristics of specified rules or displays all the rules if no rules are specified.
`ilbadm delete-rule`	Removes all information pertaining to a `rule name`. If `rule name` does not exist, this subcommand fails.
`ilbadm enable-rule`	Enables a named rule or all the rules if no names are specified.
`ilbadm disable-rule`	Disables a named rule or all the rules if no names are specified.
`ilbadm show-statistics`	Shows ILB statistics. For example, `-t` with this subcommand includes a time stamp with every header.
`ilbadm show-hc-result`	Shows the health check results for the servers that are associated with the specified name of the rule `rule-name`. If `rule-name` is not specified, the health check results of servers for all the rules are displayed.
`ilbadm show-nat`	Displays NAT table information.
`ilbadm create-servergroup`	Creates a server group with one or more servers. Additional servers can be added by using `ilbadm add-server`.
`ilbadm delete-servergroup`	Deletes a server group.
`ilbadm show-servergroup`	Lists a server group or lists all the server groups if no server group is specified.
`ilbadm enable-server`	Enables a disabled server.
`ilbadm disable-server`	Disables the specified servers.
`ilbadm add-server`	Adds the specified servers to server groups.
`ilbadm show-server`	Displays servers associated with the named rules or displays all the servers if a rule name is not specified.
`ilbadm remove-server`	Removes one or more servers from a server group.
`ilbadm create-healthcheck`	Sets up health check information that can be used to set up rules.
`ilbadm show-healthcheck`	Displays detailed information about the configured the health check.
`ilbadm delete-healthcheck`	Deletes the health check information.
`ilbadm show-persist`	Displays the session persistence mapping table.
`ilbadm export-config` *filename*	Exports the existing ILB configuration file in a format suitable for importing by using `ilbadm import`. If *filename* is not specified, then `ilbadm export` writes to `stdout`.

TABLE 10–4 ILB Subcommands Used to Manipulate the Load-balancing Rules *(Continued)*

ILB Subcommand	Description
`ilbadm import-config -p` *filename*	Imports a file and replaces the existing ILB configuration with the contents of this imported file. If *filename* is not specified, then `ilbadm import` reads from `stdin`.

◆ ◆ ◆ CHAPTER 11

Configuring Integrated Load Balancer

This chapter describes the installation of Integrated Load Balancer (ILB) and gives examples on setting up simple ILB configurations. The following topics are discussed:

Installing ILB

ILB has two portions, the kernel and the userland. The kernel portion is automatically installed as a part of the Oracle Solaris 11 installation. To obtain the userland portion of ILB, you must manually install the `ilb` package by using the `pkg install ilb` command.

Enabling ILB

This section describes the procedures you use to enable ILB.

▼ How to Enable ILB

Before You Begin Make sure that the system's role-based access control (RBAC) attribute files have the following entries. If the entries are not present, add them manually.

- File name: `/etc/security/auth_attr`
  - `solaris.network.ilb.config:::Network ILB Configuration::help=NetworkILBconf.html`

    - `solaris.network.ilb.enable:::Network ILB Enable Configuration::help=NetworkILBenable.html`
    - `solaris.smf.manage.ilb:::Manage Integrated Load Balancer Service States::help=SmfILBStates.html`
- File name: `/etc/security/prof_attr`
    - `Network ILB:::Manage ILB configuration via ilbadm:auths=solaris.network.ilb.config,solaris.network.ilb.enable;help=RtNetILB.htm`
    - Network Management entry in the file must include `solaris.smf.manage.ilb`.
- File name: `/etc/user_attr`
    - `daemon::::auths=solaris.smf.manage.ilb,solaris.smf.modify.application`

You must set up user authorization for ILB configuration subcommands. You must have the `solaris.network.ilb.config` RBAC authorization to execute the ILB configuration subcommands listed in "ILB Command and Subcommands" on page 156.

- To assign the authorization to an existing user, see Chapter 9, "Using Role-Based Access Control (Tasks)," in *Oracle Solaris 11.1 Administration: Security Services*.
- You can also provide the authorization when creating a new user account on the system.

  The following example creates a user `ilbadm` with group ID `10`, user ID `1210` and with the authorization to administer ILB in the system.

  ```
  # useradd -g 10 -u 1210 -A solaris.network.ilb.config ilbadmin
  ```

  The `useradd` command adds a new user to the `/etc/passwd`, `/etc/shadow`, and `/etc/user_attr` files. The `-A` option assigns the authorization to the user.

1 Assume a role that includes the ILB Management rights profile, or become superuser.

You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Enable the appropriate forwarding service either IPv4 or IPv6 or both of them.

This command produces no output when successful.

```
# ipadm set-prop -p forwarding=on ipv4
# ipadm set-prop -p forwarding=on ipv6
```

3 Enable the ILB service.

```
# svcadm enable ilb
```

4 Verify that the ILB service is enabled.

```
# svcs ilb
```

Configuring ILB

This section describes the steps for setting up ILB to use a half-NAT topology to load balance traffic among two servers. See the NAT topology implementation in "ILB Operation Modes" on page 149.

How to Configure ILB

1. **Assume a role that includes the ILB Management rights profile, or become superuser.**

   You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Set up the back-end servers.**

   The back-end servers are set up to use ILB as the default router in this scenario. This can be done by running the following commands on both servers.

   ```
   # route add -p default 192.168.1.21
   ```

   After executing this command, start the server applications on both servers. Assume that it is a TCP application listening on port 5000.

3. **Set up the server group in ILB.**

   There are 2 servers, `192.168.1.50` and `192.169.1.60`. A server group, `srvgrp1`, consisting of these two servers can be created by typing the following command.

   ```
   # ilbadm create-sg -s servers=192.168.1.50,192.168.1.60 srvgrp1
   ```

4. **Set up a simple health check called `hc-srvgrp1`, can be created by typing the following command.**

   A simple TCP level health check is used to detect if the server application is reachable. This check is done every 60 seconds. It will try at most 3 times and wait for at most 3 seconds between trials to see if a server is healthy. If all 3 trials fail, it will mark the server as `dead`.

   ```
   # ilbadm create-hc -h hc-test=tcp,hc-timeout=3, \
   hc-count=3,hc-inerval=60 hc-srvgrp1
   ```

5. **Set up an ILB rule by typing the following command.**

   Persistence (with 32 bits mask) is used in this rule. And the load balance algorithm is `round robin`. The server group `srvgrp1` is used and the health check mechanism used is `hc-srvgrp1`. The rule can be created by typing the following command.

   ```
   # ilbadm create-rule -e -p -i vip=10.0.2.20,port=5000 -m \
    lbalg=rr,type=half-nat,pmask=32 \
   -h hc-name=hc-srvgrp1 -o servergroup=srvgrp1 rule1_rr
   ```

Disabling ILB

The following section describes the procedure to disable ILB.

▼ How to Disable ILB

1. **Assume a role that includes the ILB Management rights profile, or become superuser.**

   You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Disable the ILB service.**

   ```
   # svcadm disable ilb
   ```

3. **Verify that the ILB service is disabled.**

   ```
   # svcs ilb
   ```

Importing and Exporting Configurations

The `ilbadm export` subcommand exports the current ILB configuration to a user-specified file. This information can then be used as input for the `ilbadm import` subcommand.

The `ilbadm import` subcommand deletes the existing configuration before importing unless specifically instructed to retain it. Omission of a file name instructs the command to read from `stdin` or write to `stdout`.

To export an ILB configuration, use the `export-config` command. The following example exports the current configuration into the file, `/var/tmp/ilb_config`, in a format suitable for importing by using the `import` subcommand:

```
# ilbadm export-config /var/tmp/ilb_config
```

To import an ILB configuration, use the `import-config` command. The following example reads the contents of the file, `/var/tmp/ilb_config`, and overrides the existing configuration:

```
# ilbadm import-config /var/tmp/ilb_config
```

Configuring ILB for High-Availability (Active-Passive Mode Only)

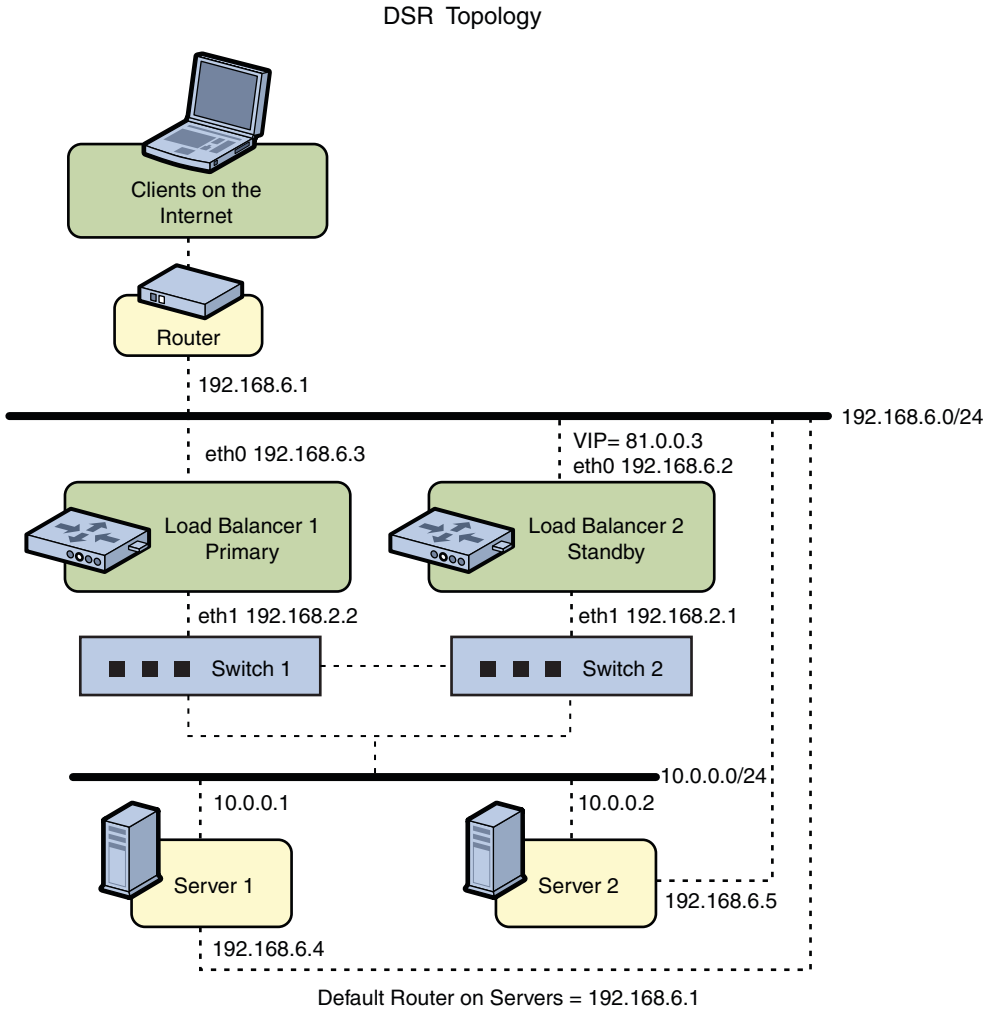
This section describes the high-availability (HA) configuration of ILB using the DSR, half-NAT topologies.

Configuring ILB for High-Availability Using the DSR Topology

This section describes how to set up the ILB connections to achieve high availability (HA) by using the DSR topology. You need to set up two load balancers, one as the primary load balancer and the other as the standby load balancer. If the primary load balancer fails, the standby load balancer assumes the role of the primary load balancer.

The following figure shows the DSR topology for configuring the ILB connections to achieve HA.

FIGURE 11–1 ILB for HA Configuration Using DSR Topology



All VIPs on Load Balancers are configured on interfaces facing subnet 192.168.6.0/24.

▼ How to Configure ILB to Achieve High-Availability by Using the DSR Topology

1 Assume a role that includes the ILB Management rights profile, or become superuser.

You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

2. **Configure both the primary and standby load balancers.**

```
# ilbadm create-servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -i vip=81.0.0.3,port=9001 \
-m lbalg=hash-ip-port,type=DSR -o servergroup=sg1 rule1
```

3. **Make sure that all servers have VIP configured on their `lo0` interface.**

```
Server1# ipadm create-addr -d -a 81.0.0.3/24 lo0
Server2# ipadm create-addr -d -a 81.0.0.3/24 lo0
```

4. **Configure Load Balancer 1 to serve as the primary load balancer.**

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# ipadm create-addr -d -a 81.0.0.3/24 vnic1
```

5. **Configure Load Balancer 2 to serve as the standby load balancer.**

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# ipadm create-addr -d -a 81.0.0.3/24 vincl
```

The preceding configuration provides protection against the following failure scenarios:

- If Load Balancer 1 fails, Load Balancer 2 becomes the primary load balancer. Load balancer 2 then takes over address resolution for the VIP `81.0.0.3` and handles all the packets from clients with the destination IP address `81.0.0.3`.

  When Load Balancer 1 recovers, Load Balancer 2 returns to standby mode.
- If one or both of Load Balancer 1's interfaces fails, Load Balancer 2 takes over as the primary load balancer. Load Balancer 2 then takes over address resolution for VIP `81.0.0.3` and handles all the packets from clients with the destination IP address `81.0.0.3`.

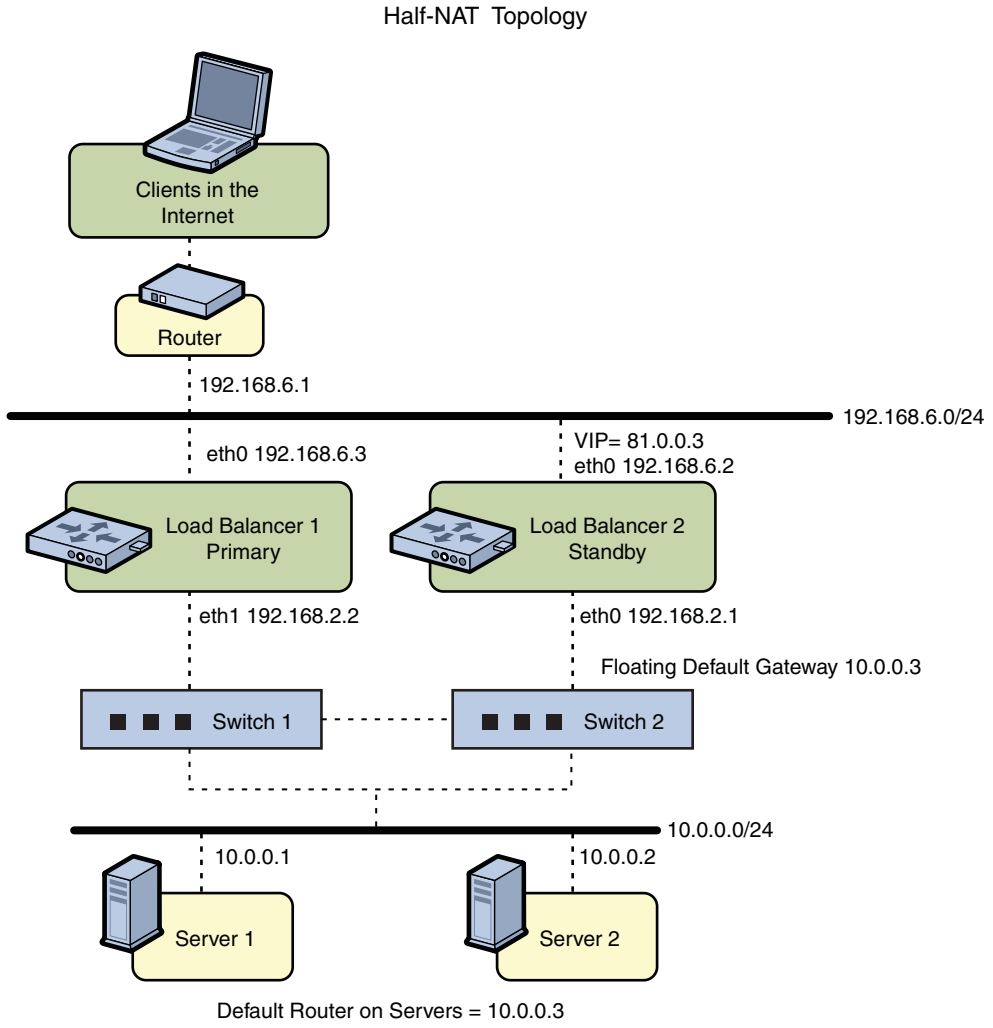
  When both of Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to standby mode.

Configuring ILB for High-Availability Using the Half-NAT Topology

This section describes how to set up the ILB connections to achieve high availability (HA) by using the half-NAT topology. You need to set up two load balancers, one as the primary and the other as the standby. If the primary load balancer fails, the standby load balancer assumes the role of the primary load balancer.

The following figure shows the half-NAT topology for configuring the ILB connections to achieve HA.

FIGURE 11–2 ILB for HA Configuration Using Half-NAT Topology



All VIPs on Load Balancers are configured on interfaces facing subnet 192.168.6.0/24.

How to Configure ILB to Achieve High-Availability by Using the Half-NAT Topology

1 Assume a role that includes the ILB Management rights profile, or become superuser.

You can assign the ILB Management rights profile to a role that you create. To create the role and assign the role to a user, see "Initially Configuring RBAC (Task Map)" in *Oracle Solaris 11.1 Administration: Security Services*.

2 Configure both the primary and standby load balancers.

```
# ilbadm create servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -ep -i vip=81.0.0.3,port=9001-9006,protocol=udp \
-m lbalg=roundrobin,type=HALF-NAT,pmask=24 \
-h hc-name=hc1,hc-port=9006 \
-t conn-drain=70,nat-timeout=70,persist-timeout=70 -o servergroup=sg1 rule1
```

3 Configure Load Balancer 1 to serve as the primary load balancer.

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# ipadm create-addr -d -a 81.0.0.3/24 vnic1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB1# ipadm create-addr -d -a 10.0.0.3/24 vnic2
LB1# vrrpadm create-router -V 2 -A inet -l eth1 -p 255 vrrp2
```

4 Configure the Load Balancer 2 to serve as the standby load balancer.

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# ipadm create-addr -d -a 81.0.0.3/24 vnic1
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB2# ipadm create-addr -d -a 10.0.0.3/24 vnic2
LB2# vrrpadm create-router -V 2 -A inet -l eth1 -p 100 vrrp2
```

5 Add the IP address for the floating default gateway to both servers.

```
# route add net 192.168.6.0/24 10.0.0.3
```

The preceding configuration provides protection against the following failure scenarios:

- If Load Balancer 1 fails, Load Balancer 2 becomes the primary load balancer. Load balancer 2 then takes over address resolution for the VIP `81.0.0.3` and handles all the packets from clients with the destination IP address `81.0.0.3`. Load balancer 2 also handles all the packets that are sent to the floating gateway address `10.0.0.3`.

  When Load Balancer 1 recovers, Load Balancer 2 returns to the standby mode.

- If one or both of Load Balancer 1's interfaces fails, Load Balancer 2 takes over as primary load balancer. Load Balancer 2 then takes over address resolution for VIP `81.0.0.3` and handles all packets from clients with the destination IP address `81.0.0.3`. Load balancer 2 also handles all the packets that are sent to the floating gateway address `10.0.0.3`.

  When both of Load Balancer 1's interfaces are healthy, Load Balancer 2 returns to standby mode.

Note – The current implementation of ILB does not synchronize primary and standby load balancers. When the primary load balancer fails and the standby load balancer takes over, the existing connections fail. However, HA without synchronization is still valuable under circumstances when the primary load balancer fails.

◆ ◆ ◆ CHAPTER 12

Managing Integrated Load Balancer

This chapter describes the procedures for administering ILB server groups like creating or removing a sever group, administering back-end servers like adding, deleting, re-enabling or disabling a server from the server group, creating and deleting rules and displaying statistics.

The following topics are discussed:

Administering ILB Server Groups

This section describes how you can use the `ilbadm` command to create, delete, and list ILB server groups.

▼ How to Create an ILB Server Group

1 **Select a name for the server group that you are about to create.**

2 **Select the servers that are to be included in the server group.**

Servers can be specified by their host name or IP address and optional port.

3 **Create the server group.**

```
# ilbadm create-servergroup -s servers= \
server1,server2,server3  servergroup
```

Example 12–1 Creating an ILB Server Group

The following example creates a server group called `webgroup` consisting of three servers:

```
# ilbadm create-servergroup -s servers=webserv1,webserv2,webserv3 webgroup
```

▼ How to Delete an ILB Server Group

1 In a terminal window, type the `show-servergroup` subcommand to obtain information about a specific server group or all server groups.

```
# ilbadm show-servergroup -o all
```

The following sample command lists detailed information about all the server groups:

sgname	serverID	minport	maxport	IP_address
specgroup	_specgroup.0	7001	7001	199.199.68.18
specgroup	_specgroup.1	7001	7001	199.199.68.19
test123	_test123.0	7002	7002	199.199.67.18
test123	_test123.1	7002	7002	199.199.67.19

The above table shows two server groups, `specgroup` and `test123`. The `specgroup` contains two servers, `199.199.68.18` and `199.199.68.19` and the server is using port `7001`. Similarly, `test123` also contains two servers, `199.199.67.18` and `199.199.67.19`. This server is using the port `7002`.

2 Select the server group that you want to delete.

The server group must not be in use by an active rule. Otherwise, the deletion will fail.

3 In a terminal window, use the following command to delete the server group.

```
# ilbadm delete-servergroup servergroup
```

Example 12–2 Deleting an ILB Server Group

The following example removes the server group called `webgroup`:

```
# ilbadm delete-servergroup webgroup
```

Administering Back-End Servers in ILB

This section describes how you can use the `ilbadm` command to add, remove, enable, and disable one or more back-end servers within a server group.

How to Add a Back-End Server to an ILB Server Group

- **Add a back-end server to a server group.**

  Server specifications must include a host name or IP address and can also include an optional port or a range of ports. Server entries with the same IP address are disallowed within a server group.

  ```
  # ilbadm add-server -s server=192.168.89.1,192.168.89.2 ftpgroup
  # ilbadm add-server -s server=[2001:7::feed:6]:8080 sgrp
  ```

  The `-e` option enables the servers that are added in the server groups.

Note – IPv6 addresses must be enclosed in square brackets.

Example 12–3 Adding a Back-End Server to an ILB Server Group

The following example adds back-end servers to server groups `ftpgroup` and `sgrp`, and enables the servers.

```
# ilbadm add-server -e -s \
server=192.168.89.1,192.168.89.2 ftpgroup
# ilbadm add-server -e -s server=[2001:7::feed:6]:8080 sgrp
```

How to Remove a Back-End Server From an ILB Server Group

1. **To remove a back-end server from a server group, follow these steps:**

   a. **Identify the server ID of the server that you want to remove from a server group.**

      The server ID is a unique name for the IP address that is assigned to a system when the server is added to a server group. This can be obtained from the output of `show-servergroup -o all` subcommand.

   b. **Remove the server.**

      ```
      # ilbadm remove-server -s server=serverID servergroup
      ```

2. **To remove a back-end server from all server groups, follow these steps:**

   a. **Identify the IP address and the host name of the server you want to remove.**

   b. **Use the output of the `ilbadm show-servergroup-o all` command to identify the server groups that include the server.**

   c. **For each server group, run the above subcommand to remove the server from the server group.**

Example 12–4 Removing a Back-End Server From an ILB Server Group

The following example removes the server with server ID `_sg1.2` from server group `sg1`:

```
# ilbadm remove-server -s server=_sg1.2 sg1
```

Note the following:

- If the server is being used by a NAT or half-NAT rule, disable the server by using the `disable-server` subcommand before removal. For more information, see "How to Re-enable or Disable a Back-End Server in an ILB Server Group" on page 172. When a server is disabled, it enters the connection-draining state. After all the connections are drained, the server can be removed by using the `remove-server` subcommand. After issuing the `disable-server` command, periodically check the NAT table (by using the `show-nat` command) to see if the server in question still has connections. After all of the connections are drained (the server does not get displayed in the `show-nat` command output), the server can then be removed by using the `remove-server` command.
- If the `conn-drain` timeout value is set, the connection-draining state will be completed upon conclusion of the timeout period. The default value of `conn-drain` timeout is `0`, meaning it will keep waiting until a connection is gracefully shut down.

▼ How to Re-enable or Disable a Back-End Server in an ILB Server Group

1 Identify the IP address, host name, or server ID of the back-end server you want to re-enable or disable.

If an IP address or host name is specified, the server will be re-enabled or disabled for the all rules associated with it. If a server ID is specified, the server will be re-enabled or disabled for the specific rules that are associated with the server ID.

Note – A server can have multiple server IDs, if it belongs to multiple server groups.

---

2 Re-enable or disable the back-end server.

```
# ilbadm enable-server webservergroup.1
# ilbadm disable-server webservergroup.1
```

Example 12–5 Re-enabling and Disabling a Back-End Server in an ILB Server Group

In the following example, a server with server ID `websg.1` is enabled and then disabled:

```
# ilbadm enable-server websg.1
# ilbadm disable-server websg.1
```

Administering Health Checks in ILB

ILB provides the following optional types of server health checks for you to select from:

- Built-in ping probes
- Built-in TCP probes
- Built-in UDP probes
- User-supplied custom tests that can run as health checks

By default, ILB does not perform any health checks. You can specify health checks for each server group when creating a load-balancing rule. You can configure only one health check per load-balancing rule. As long as a virtual service is enabled, the health checks on the server group that is associated with the enabled virtual service starts automatically and is repeated periodically. The health checks stop as soon as the virtual service is disabled. The previous health check states are not preserved when the virtual service is re-enabled.

When you specify a TCP, UDP, or custom test probe for running a health check, ILB sends a ping probe, by default, to determine if the server is reachable before it sends the specified TCP, UDP, or custom test probe to the server. The ping probe is a method of monitoring server health. If the ping probe fails, the corresponding server is disabled with the health check status of `unreachable`. If the ping probe succeeds, but the TCP, UDP, or custom test probe fails, the server is disabled with the health check status of `dead`.

Note –

- You can disable the default ping probe.
- The default ping probe cannot be disabled for the UDP probe. Thus, for the UDP health checks, the ping probe is always the default probe.

Creating a Health Check

In the following example, two health check objects, `hc1` and `hc-myscript`, are created. The first health check uses the built-in TCP probe. The second health check uses a custom test, `/var/tmp/my-script`.

```
# ilbadm create-healthcheck \
-h hc-timeout=3,hc-count=2,hc-interval=8,hc-test=tcp hc1
# ilbadm create-healthcheck -h hc-timeout=3, \
hc-count=2,hc-interval=8,hc-test=/var/tmp/my-script hc-myscript
```

A description of each argument is as follows:

`hc-timeout` Specifies the timeout when the health check is considered to have failed if it does not complete.

`hc-count` Specifies the number of attempts to run the `hc-test` health check.

`hc-interval`	Specifies the interval between consecutive health checks. To avoid synchronization, the actual interval is randomized between `0.5 * hc-interval` and `1.5 * hc-interval`.
`hc-test`	Specifies the type of health check.

Note – The port specification for `hc-test` is specified with the `hc-port` keyword in the `create-rule` subcommand. For details, refer to the `ilbadm(1M)` man page.

User-Supplied Test Details

The following criteria must be met by the user-supplied custom test:

- The test can be a binary or a script.
- The test can reside anywhere on the system, and you must specify the absolute path when using the `create-healthcheck` subcommand.

  When you specify the test (for example, `/var/tmp/my-script`) as part of the health check specification in the `create-rule` subcommand, the `ilbd` daemon forks a process and executes the test, as follows:

  ```
  /var/tmp/my-script $1 $2 $3 $4 $5
  ```

  A description of each argument is as follows:

`$1`	VIP (literal IPv4 or IPv6 address)
`$2`	Server IP (literal IPv4 or IPv6 address)
`$3`	Protocol (UDP, TCP as a string)
`$4`	Numeric port range (the user-specified value for `hc-port`)
`$5`	Maximum time (in seconds) that the test must wait before returning a failure. If the test runs beyond the specified time, it might be stopped, and the test is considered failed. This value is user-defined and specified in `hc-timeout`.

The user-supplied test, `my-script`, may or may not use all the arguments, but it *must* return one of the following :

- Round-trip time (RTT) in microseconds
- 0 if the test does not calculate RTT
- -1 for failure

By default, the health check test runs with the following privileges: `PRIV_PROC_FORK`, `RIV_PROC_EXEC`, and `RIV_NET_ICMPACCESS`.

If a broader privilege set is required, you must implement `setuid` in the test. For more details on the privileges, refer to the `privileges(5)` man page.

Displaying Health Checks

You can use the following `ilbadm list-healthcheck` subcommand to obtain detailed information about configured health checks:

```
# ilbadm list-healthcheck
```

The following sample output lists two configured health checks.

NAME	TIMEOUT	COUNT	INTERVAL	DEF_PING	TEST
hc1	3	2	8	Y	tcp
hc2	3	2	8	N	/var/usr-script

Displaying Health Check Results

You can use the `ilbadm list-hc-result` subcommand to obtain health check results. If a rule or a health check is not specified, the subcommand lists all the health checks.

The following example displays the health check results associated with a rule called `rule1`:

```
# ilbadm show-hc-result rule1
```

RULENAME	HCNAME	SERVERID	STATUS	FAIL	LAST	NEXT	RTT
rule1	hc1	_sg1:0	dead	10	11:01:19	11:01:27	941
rule1	hc1	_sg1:1	alive	0	11:01:20	11:01:34	1111

The LAST column in the table shows the time a health check was done on a server. The NEXT column shows the time the next health check will be done on a server.

Deleting a Health Check

The following example deletes a health check called *hc1*:

```
# ilbadm delete-healthcheck hc1
```

Administering ILB Rules

In ILB, a virtual service is represented by a load-balancing rule and is defined by the following parameters.

- Virtual IP address
- Transport protocol: TCP or UDP
- Port number (or a port range)
- Load-balancing algorithm
- Type of load-balancing mode (DSR, full-NAT, or half-NAT)
- Server group consisting of a set of back-end servers
- Optional server health checks that can be executed for each server in the server group
- Optional port to use for health checks

Note – You can specify health checks on a particular port or on any port that the `ilbd` daemon randomly selects from the port range for the server.

- Rule name to represent a virtual service

This section describes how you can use the `ilbadm` command to create, delete, and list the load-balancing rules.

Listing ILB Rules

To list the configuration details of a rule, use the `ilbadm show-rule` subcommand. If no rule name is specified, information is provided for all rules.

```
# ilbadm show-rule
```

The following is the sample command output.

RULENAME	STATUS	LBALG	TYPE	PROTOCOL	VIP	PORT
rule-http	E	hash-ip-port	HALF-NAT	TCP	10.0.0.1	80
rule-dns	D	hash-ip	DSR	UDP	10.0.0.1	53
rule-abc	D	roundrobin	NAT	TCP	2003::1	1024
rule-xyz	E	hash-ip-vip	NAT	TCP	2003::1	2048-2050

How to Create an ILB Rule

1. **Create a server group that includes the appropriate back-end servers.**

   ```
   # ilbadm create-servergroup -s server=server1:port-range1,server2:port-range2  sg1
   ```

2. **If you want to associate server health checks with a rule, create a health check.**

   ```
   # ilbadm create-healthcheck -h hc-test=protocol, \
   hc-timeout=value1,hc-count=value2 \
   ,hc-interval=value3 hc1
   ```

3. **Identify the VIP, port, and optional protocol that are to be associated with the rule.**

   These are specified using the `-i` option.

4. **Select the operation you want to use (DSR, half-NAT or full-NAT).**

   If NAT is selected, you must specify the IP address range that is to be used as the `proxy-src` address. The range is limited to 10 IP addresses for full-NAT topology.

5. **Select the load-balancing algorithm that is to be used.**

   The parameters in step 4 and step 5 can be specified in the `-m` option. For more information, see "ILB Algorithms" on page 155.

6. **Select other optional features.**

   For more information, see the `ilbadm(1M)` man page for details.

7. **Select a rule name.**

8. **Create and enable the rule.**

   For more information about each option, see the `ilbadm(1M)` man page.

   ```
   # ilbadm create-rule -e -i vip=ipaddr,port=port,protocol=protocol \
   -m lbalg=lb-algorithm,type=topology-type,proxy-src=ipaddr1-ipaddr2, \
   pmask=value4 -h hc-name=hc1 \
   -o servergroup=sg1  rule1
   ```

   The following example shows the steps to create a full-NAT rule with health check.

Example 12–6 Creating a Full-NAT Rule With Health Check Session Persistence

This example creates a health check called `hc1` and a server group called `sg1`. The server group consists of two servers, each with a range of ports. The last command creates and enables a rule called `rule1` and associates the rule to the server group and the health check. This rule implements the full-NAT mode of operation. Note that the creation of the server group and health check must precede the creation of the rule.

```
# ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2, \
hc-count=3,hc-interval=10 hc1
# ilbadm create-servergroup -s server=60.0.0.10:6000-6009,60.0.0.11:7000-7009 sg1
# ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009, \
protocol=tcp -m lbalg=rr,type=NAT, \
proxy-src=60.0.0.101-60.0.0.104,persist=24 \
-h hc-name=hc1 -o servergroup=sg1 rule1
```

When creating a half-NAT or a full-NAT rule, specify the value for the `connection-drain` timeout. The default value of `conn-drain` timeout is `0`, meaning it will keep waiting until a connection is gracefully shut down.

Deleting an ILB Rule

To delete a rule, use the `ilbadm delete-rule` subcommand. To delete all rules, use the `-a` option. The following example deletes the rule called `rule1`:

```
# ilbadm delete-rule rule1
```

Displaying ILB Statistics

This section describes how you can use the `ilbadm` command to obtain information such as the printing statistics for a server or statistics for a rule. You can also display NAT table information and the session persistence mapping table.

Obtaining Statistical Information

Use the `ilbadm show-statistics` subcommand to view load distribution details. The following example shows the usage of the `show-statistics` subcommand:

```
# ilbadm show-statistics
 PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
  9       636       0       0         0       0
```

`PKT_P` Packets processed

`BYTES_P` Bytes processed

`PKT_U` Unprocessed packets

`BYTES_U` Unprocessed bytes

`PKT_D` Packets dropped

`BYTES_D` Bytes dropped

Displaying the NAT Connection Table

Use the `ilbadm show-nat` subcommand to display the NAT connection table. No assumptions should be made about the relative positions of elements in consecutive runs of this command. For example, executing `{{ ilbadm show-nat 10}}` twice is not guaranteed to show the same 10 items twice, especially on a busy system. If a count value is not specified, the entire NAT connection table is displayed.

EXAMPLE 12–7 NAT Connection Table Entries

The following example displays five entries from the NAT connection table.

```
# ilbadm show-nat 5
UDP: 124.106.235.150.53688 > 85.0.0.1.1024 >>> 82.0.0.39.4127 > 82.0.0.56.1024
UDP: 71.159.95.31.61528 > 85.0.0.1.1024 >>> 82.0.0.39.4146 > 82.0.0.55.1024
UDP: 9.213.106.54.19787 > 85.0.0.1.1024 >>> 82.0.0.40.4114 > 82.0.0.55.1024
UDP: 118.148.25.17.26676 > 85.0.0.1.1024 >>> 82.0.0.40.4112 > 82.0.0.56.1024
UDP: 69.219.132.153.56132 > 85.0.0.1.1024 >>> 82.0.0.39.4134 > 82.0.0.55.1024
```

The format of entries is as follows:

```
T: IP1 > IP2 >>> IP3 > IP4
```

`T` Transport protocol used in this entry

`IP1` Client's IP address and port

`IP2` VIP and port

`IP3` If half-NAT mode, the client's IP address and port.

If full-NAT mode, the client's IP address and port.

`IP4` Back-end server's IP address and port.

Displaying the Session Persistence Mapping Table

Use the `ilbadm show-persist` subcommand to display the session persistence mapping table.

EXAMPLE 12–8 Session Persistence Mapping Table Entries

The following example displays five entries from the session persistence mapping table:

```
# ilbadm show-persist 5
rule2: 124.106.235.150 --> 82.0.0.56
rule3: 71.159.95.31 --> 82.0.0.55
rule3: 9.213.106.54 --> 82.0.0.55
rule1: 118.148.25.17 --> 82.0.0.56
rule2: 69.219.132.153 --> 82.0.0.55
```

EXAMPLE 12–8 Session Persistence Mapping Table Entries *(Continued)*

The format of entries is as follows:

```
R: IP1 --> IP2
```

`R` Rule that this persistence entry is tied to.

`IP1` Client's IP address.

`IP2` Back-end server's IP address.

◆ ◆ ◆ CHAPTER 13

Virtual Router Redundancy Protocol (Overview)

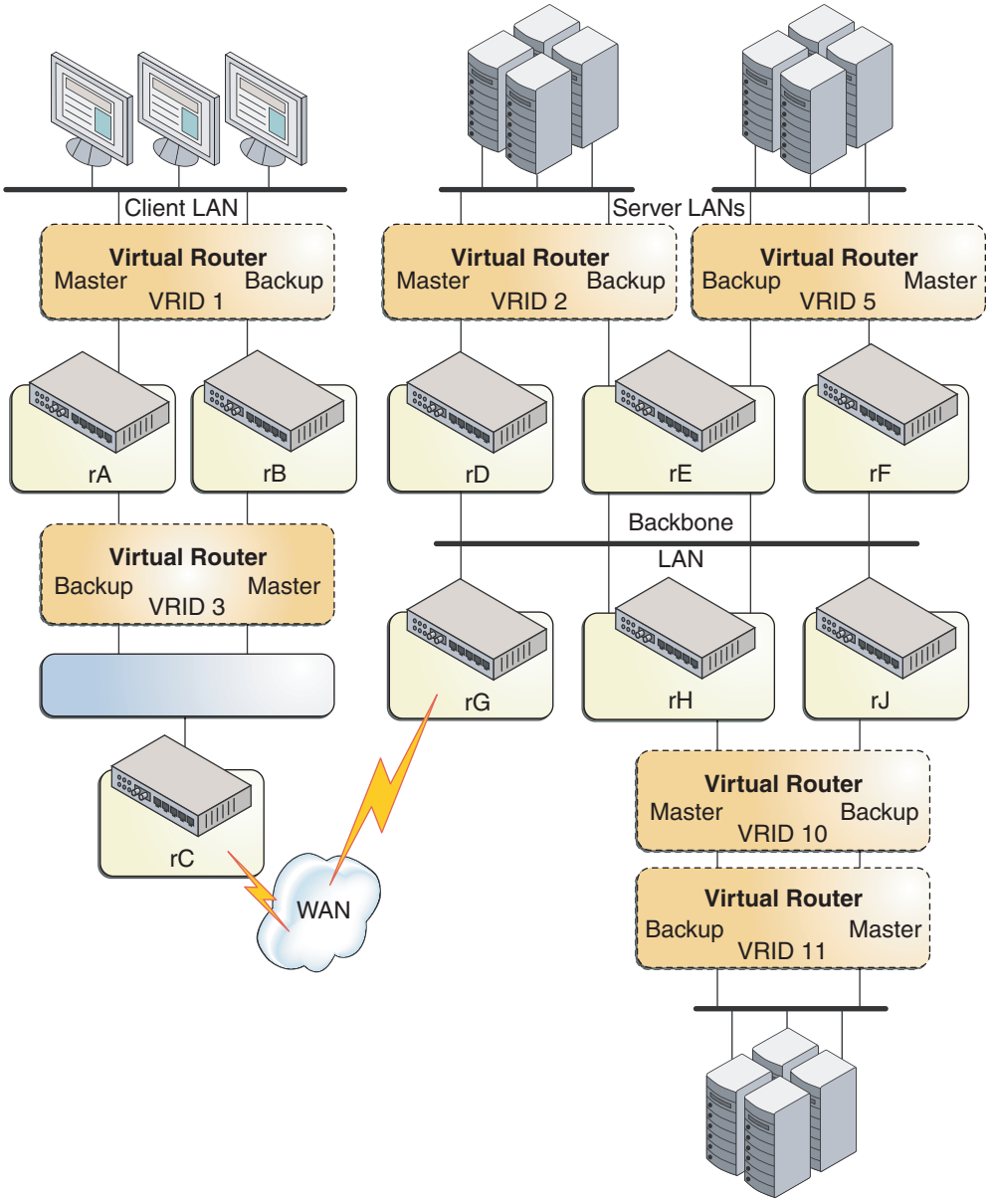
Virtual Router Redundancy Protocol (VRRP) is an Internet standard protocol specified in Virtual Router Redundancy Protocol Version 3 for IPv4 and IPv6. VRRP is supported in Oracle Solaris to provide high availability. Oracle Solaris provides an administrative tool that configures and manages the VRRP service.

The following topics are discussed:

How VRRP Works?

The following figure shows how VRRP works. The text that follows the figure explains the VRRP components that are used in the figure.

FIGURE 13–1 Working of VRRP



In the preceding figure, VRRP uses the following components:

- Router `rA` is the master router for virtual router VRID 1 and the backup router for VRID 3. Router `rA` handles the routing of packets that are addressed to the VIP for VRID 1 and is ready to assume the routing role for VRID 3.
- Router `rB` is the master router for virtual router VRID 3 and the backup router for VRID 1. Router `rB` handles the routing of packets that are addressed to the VIP for VRID 3 and is ready to assume the routing role for VRID 1.
- Router `rC` does not have VRRP functions, but it uses the VIP for VRID 3 to reach the client LAN subnet.
- Router `rD` is the master router for VRID 2. Router `rF` is the master router for VRID 5. Router `rE` is the backup router for both of these VRIDs. If `rD` or `rF` fails, `rE` becomes the master router for that VRID. Both `rD` and `rF` could fail at the same time. The fact that a VRRP router is a master router for one VRID does not preclude it from being a master router for another VRID.
- Router `rG` is the wide area network (WAN) gateway for the backbone LAN. All of the routers attached to the backbone are sharing routing information with the routers on the WAN by using a dynamic routing protocol such as Open Shortest Path First (OSPF). VRRP is not involved in this aspect, although router `rC` advertises that the path to the client LAN subnet is through the VIP of VRID 3.
- Router `rH` is the master router for VRID 10 and the backup router for VRID 11. Likewise, router `rJ` is the master router for VRID 11 and the backup router for VRID 10. This VRRP load-sharing configuration illustrates that multiple VRIDs can exist on a single router interface.

VRRP can be used as part of a network design that provides almost total routing redundancy for all systems on the network.

Using VRRP in Local Area Network

When you set up a network such as a local area network (LAN), it is very important to provide a high-availability service. One way to increase the reliability of the network is to provide backups of the critical components in the network. Adding components such as routers, switches, and links to the network ensures the continuity of the service across failures. Providing redundancy at the endpoints of a network is a crucial task that can be done easily with VRRP. Virtual routers can be introduced in the LAN by using VRRP to provide failure recovery for a router.

VRRP is an election protocol that dynamically assigns the responsibilities of a virtual router to one of the VRRP routers within the LAN. VRRP provides one or more backup routers for a statically configured router on the LAN.

A VRRP router called the *master router* controls the IPv4 or IPv6 address or addresses that are associated with the virtual router. The virtual router forwards the packets that are sent to the IP address of the master router.

The election process provides dynamic failover while forwarding packets that are sent to these IP addresses. VRRP eliminates the single point of failure that is inherent in the static default routed environment.

By using the VRRP feature in Oracle Solaris, you can have a more highly available default path for the routing process without having to configure the dynamic routing or router discovery protocols on every end-host.

VRRP Router

VRRP runs on each VRRP router and manages the state of the router. A host can have multiple VRRP routers configured, where each VRRP router belongs to a different virtual router.

A VRRP router has the following attributes:

- **Router name** – A system-wide unique identifier
- **Virtual Router ID (VRID)** – A unique number used to identify a virtual router on a given network segment. VRIDs identify the virtual router within a LAN
- **Primary IP address** – The source IP address of the VRRP advertisement
- **Virtual IP addresses (VRIP)** – An IP address associated with a VRID from which other hosts can obtain network service. The VRIP is managed by the VRRP instances belonging to a VRID.
- **VRRP parameters** – Includes priority, advertise interval, preempt mode, and accept mode
- **VRRP state information and statistics**

Administering VRRP Subcommands

The following sections summarize the `vrrpadm` subcommands. See the `vrrpadm(1M)` man page for details. The results of all the subcommands are persistent except for the `vrrpadm show-router` subcommand. For example, the VRRP router created by `vrrpadm create-router` will persist across reboot.

VRRP VNIC Creation

A pseudo network interface that is configured on top of a system's physical network adapter, also called a network interface (NIC) card. A physical interface can have more than one VNIC. VNICs are essential components of network virtualization. For more information, *Using Virtual Networks in Oracle Solaris 11.1*.

The existing `dladm create-vnic` subcommand has been extended to enable you create a VRRP VNIC. The syntax is as follows:

```
# dladm create-vnic [-t] [-R root-dir] [-l link] [-m vrrp -V VRID -A \
{inet | inet6}] [-v vlan-id] [-p prop=value[,...]] vnic-link
```

A new VNIC address type, `vrrp`, has been introduced. You must specify the VRID and address family with this new VNIC address type.

As a result, a VNIC with a well-known virtual router MAC address will be created.

Creating a Router

The `vrrpadm create-router` subcommand creates a VRRP router with the specified VRID and address family, along with other specified parameters. Each VRRP router requires a special VRRP VNIC to be created, and the VNIC can be created by using the `dladm create-vnic` command. For more information, see the `vrrpadm(1M)` man page. The syntax is as follows:

```
# vrrpadm create-router -V vrid -l link -A {inet | inet6} \
[-p priority] [-i adv-interval] [-o flags] router-name
```

The `-o` option is used to configure the preempt and accept modes of the VRRP router. The values can be: `preempt`, `un_preempt`, `accept`, `no_accept`. By default, both modes are set to `true`.

The *router-name* is used as the unique identifier of this VRRP router and is used in the other `vrrpadm` subcommands. The permitted characters in a router name are: alphanumeric (`a-z`, `A-Z`, `0-9`) and underscore (`'_'`). The maximum length of a router name is 31 characters.

Enabling a Router

A disabled VRRP router can be re-enabled by using the `enable-router` subcommand. The underlying datalink that the VRRP router is created over (specified with the `-l` option when the router is created with `vrrpadm create-router`) and the router's VRRP VNIC must exist when the router is enabled. Otherwise, the enable operation fails. The syntax is as follows:

```
# vrrpadm enable-router router-name
```

Modifying a Router

The `vrrpadm modify-router` subcommand changes the configuration of a specified VRRP router. The syntax is as follows:

```
# vrrpadm modify-router [-p priority] [-i adv-interval] \
[-o flags] router-name
```

Displaying the Configuration of a Router

The `vrrpadm show-router` subcommand shows the configuration and status of a specified VRRP router. See more details in the `vrrpadm(1M)` man page. The syntax is as follows:

```
# vrrpadm show-router [-P | -x] [-p] [-o field[,...]] [router-name]
```

The following are examples of the `vrrpadm show-router` output:

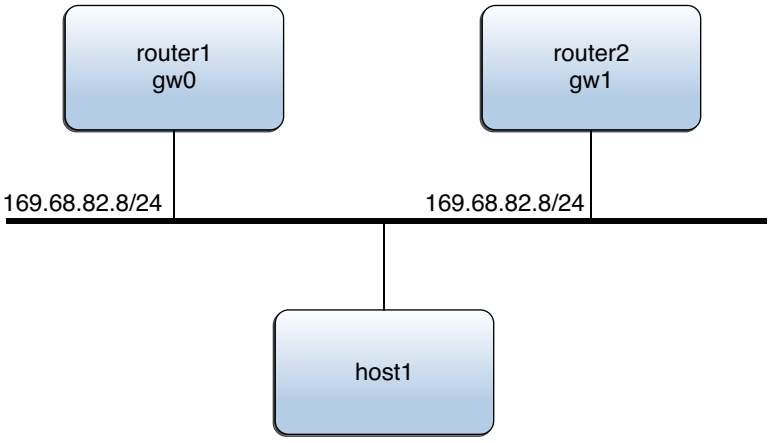
```
# vrrpadm show-router vrrp1
NAME VRID LINK AF PRIO ADV_INTV MODE STATE VNIC
vrrp1 1 bge1 IPv4 100 1000 e-pa- BACK vnic1

# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK MAST 1m17s vnic1 10.0.0.100 10.0.0.1

# vrrpadm show-router -P vrrp1
NAME PEER P_PRIO P_INTV P_ADV_LAST M_DOWN_INTV
vrrp1 10.0.0.123 120 1000 0.313s 3609
```

EXAMPLE 13–1 VRRP Configuration Example

The following figure shows a typical VRRP configuration.



In this example, the IP address `169.68.82.8` is configured as the default gateway for `host1`. This IP address is the virtual IP address that is protected by the virtual router that consists of two VRRP routers: `router1` and `router2`. At one time, only one of the two routers serves as the master router and assumes the responsibilities of the virtual router and forwards packets that come from `host1`.

EXAMPLE 13–1 VRRP Configuration Example *(Continued)*

Assume that the VRID of the virtual router is 12. The following examples shows the commands that are used to configure the preceding VRRP configuration on `router1` and `router2`. `router1` is the owner of the virtual IP address `169.68.82.8` and its priority is the default value (`255`). `router2` is the standby router whose priority is `100`.

```
router1:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw0 vnic1
# vrrpadm create-router -V 12 -A inet -l gw0 vrrp1
# ipadm create-addr -d -a 169.68.82.8/24 vnic1/router1
# ipadm create-addr -d -a 169.68.82.100/24 gw0/router1
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
router2:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw1 vnic1
# vrrpadm create-router -V 12 -A inet -l gw1 -p 100 vrrp1
# ipadm create-addr -d -a 169.68.82.8/24 vnic1/router2
# ipadm create-addr -d -a 169.68.82.101/24 gw1/router2
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK INIT 2m32s vnic1 169.68.82.101 169.68.82.8
```

Using the configuration of `router1` as an example, you must configure at least one IP address over `gw0`. In the following example, this IP address of `router1` is the primary IP address, which is used to send the VRRP advertisement packets:

```
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
```

Disabling a Router

A VRRP router does not function until it is enabled. By default, a VRRP router is enabled when it is first created. However, at times, it is useful to temporarily disable a VRRP router so that you can make configuration changes and then re-enable the router. The syntax is as follows:

```
# vrrpadm disable-router router-name
```

Deleting a Router

The `vrrpadm delete-router` subcommand deletes a specified VRRP router. The syntax is as follows:

```
# vrrpadm delete-router router-name
```

VRRP Security Considerations

The `solaris.network.vrrp` authorization is required to configure the VRRP service. Note that the read-only operation - `vrrpadm showrouter` does not require this authorization.

The `solaris.network.vrrp` authorization is part of the Network Management profile.

VRRP Limitations

This section describes the limitations of VRRP.

Exclusive-IP Zone Support

In each exclusive-IP zone, the VRRP service `svc:/network/vrrp/default` is enabled automatically when any VRRP router is created in the zone. The VRRP service manages the VRRP router for that specific zone.

However, the support for an exclusive-IP zone is limited because of the following reasons:

- A VNIC cannot be created inside a non-global zone. Therefore, the VRRP VNIC must be created in the global-zone first. Then the VNIC must be assigned to the non-global zone where the VRRP router resides. The VRRP router can then be created and started in the non-global zone by using the `vrrpadm` command.
- On a single Oracle Solaris system, it is not possible to create two VRRP routers in different zones to participate with the same virtual router. The reason is that Oracle Solaris does not allow you to create two VNICs with the same MAC address.

Inter-operations With Other Network Features

The VRRP service cannot work on an IP network multipathing (IPMP) interface. VRRP requires specific VRRP MAC addresses, but IPMP works completely in the IP layer.

Further, the VRRP virtual IP addresses can only be statically configured and cannot be auto-configured by the two existing auto-configuration tools for IP addresses: `in.ndpd` for IPv6 auto-configuration and `dhcpagent` for DHCP configuration. Because the master and the backup VRRP routers (VNICs) share the same MAC address, `in.ndpd` and `dhcpagent` can become confused. Eventually, unexpected results can occur. Therefore, IPv6 auto-configuration and DHCP configurations are not supported over VRRP VNICs. If you configure either IPv6 auto-configuration or DHCP over a VRRP VNIC, the attempt to bring up the auto-configured IP address fails, as will the auto-configuration operation.

◆ ◆ ◆ APPENDIX A

Link Aggregation Types: Feature Comparison

Both trunk aggregations and datalink multipathing (DLMP) aggregations support nearly the same features to improve network performance. However, differences do exist. The following table presents a general comparison of the two types of link aggregation in Oracle Solaris.

Feature	Trunk Aggregations	DLMP Aggregations
Link-based failure detection	Supported	Supported
Link Aggregation Control Protocol (LACP)	Supported	Not supported
Use of standby interfaces	Not supported	Supported
Span multiple switches	Not supported unless using vendor proprietary solution	Supported
Switch configuration	Required	Not required
Policies for load balancing	Supported	Not applicable
Load spreading across all the aggregation's ports	Supported	Limited[1]
User defined flows for resource management	Supported	Supported
Link protection	Supported	Supported
Back-to-back parallel configuration	Supported	Not supported[2]

[1]The aggregation spreads its VNICs across all ports. However, individual VNICs cannot spread the load on multiple ports.

[2]DLMP aggregations must always use an intermediary switch to send packets to other destination systems. However, when using DLMP aggregations, do not configure the switch for link aggregation.

◆ ◆ ◆ APPENDIX B

Link Aggregations and IPMP: Feature Comparison

IPMP and link aggregation are different technologies that achieve improved network performance as well as maintain network availability.

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

Feature	IPMP	Link Aggregation
Network technology type	Layer 3 (IP layer)	Layer 2 (link layer)
Configuration tool	`ipadm`	`dladm`
Link-based failure detection	Supported	Supported
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 LACP, targeting the immediate peer host or switch Not supported in DLMP aggregations.
Use of standby interfaces	Supported	Not supported in trunk aggregations Supported in DLMP aggregations
Span multiple switches	Supported	Not supported in trunk aggregations. Some vendors provide proprietary and non-interoperable solutions to span multiple switches. Supported in DLMP aggregations

Feature	IPMP	Link Aggregation
Hardware support	Not required	Required for trunk aggregations. For example, a trunk aggregation on an Oracle Solaris system requires that corresponding ports on the switches also be aggregated. Not required in DLMP aggregations
Link layer requirements	Broadcast-capable	Ethernet-specific
Driver framework requirements	None	Must use GLDv3 framework
Load-spreading support	Supported, controlled by the kernel. Inbound load spreading is indirectly affected by source address selection.	Supported in trunk aggregations only, and controlled by the administrator by using the `dladm` command. Inbound load spreading is supported. Load spreading by individual interfaces over the aggregation is not supported in DLMP aggregations.
Level of support when integrating with VNICs	Poor support	Excellent support
User defined flows for resource management	Not supported	Supported
Link protection	Not supported	Supported
Protocol requirements	None	None

In link aggregations, incoming traffic is spread over the multiple links that comprise the aggregation in standard mode. Thus, networking performance is enhanced as more NICs are installed to add links to the aggregation.

DLMP aggregations span multiple switches. As a Layer 2 technology, aggregations integrate well with other Oracle Solaris virtualization technologies.

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.

Trunk aggregations and IPMP can complement each other and can be deployed together to provide the combined benefits of network performance and availability.

Index

E

F

G

H

I

L