Managing Network Datalinks in Oracle® Solaris 11.2

Part No: E37516-02
September 2014

ORACLE

Contents

Using This Documentation

- **Overview** – Provides an overview of the advanced features that are used to manage network datalinks to improve network performance. Describes how to combine links into aggregations by using trunk or DLMP aggregation, divide your network into subnetworks by using virtual local area networks, connect separate network segments by using bridges, exchange network connectivity information by using Link Layer Discovery Protocol, and manage converged networks by using data center bridging.
- **Audience** – System administrators.
- **Required knowledge** – Basic and some advanced network administration skills.

Product Documentation Library

Late-breaking information and known issues for this product are included in the documentation library at http://www.oracle.com/pls/topic/lookup?ctx=E36784.

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.

Feedback

Provide feedback about this documentation at http://www.oracle.com/goto/docfeedback.

♦ ♦ ♦ CHAPTER 1

Introduction to Managing Network Datalinks

This chapter introduces the advanced features that are used to manage network datalinks, which are described in the remaining chapters of this book. 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. Therefore, you do not need to complete all the configuration procedures in this book. Instead, select and deploy the technologies that address your network requirements.

Before you perform any of the configurations that are described in this book, you must have completed basic network configuration and have an understanding about the fundamental datalink configuration. For information about basic network configuration, see "Configuring and Administering Network Components in Oracle Solaris 11.2 ". For information about elements of managing datalink configuration, see Chapter 2, "Administering Datalink Configuration in Oracle Solaris," in "Configuring and Administering Network Components in Oracle Solaris 11.2 ".

For a summary of network configuration features in Oracle Solaris, see Chapter 1, "Summary of Oracle Solaris Network Administration," in "Strategies for Network Administration in Oracle Solaris 11.2 ".

This chapter contains the following topics:

What's New in Managing Network Datalinks in Oracle Solaris 11.2

For existing customers, this section highlights the following key changes in this release:

- **Probe-based failure detection in datalink multipathing (DLMP) aggregation** – Detects the loss of connectivity between DLMP aggregated links and configured targets. This type of failure detection addresses the limitations of the link-based failure detection mechanism, which can only detect failures caused by the loss of direct connection between

the datalink and the first-hop switch. For more information, see “Failure Detection in DLMP Aggregation” on page 22.

- **Transmission of the enhanced transmission selection (ETS) recommended values to the peer** – ETS feature in data center bridging (DCB) is enhanced to enable the Oracle Solaris host to recommend bandwidth shares to the next hop switch in the DCB network. For more information, see “Recommending ETS Configuration to the Peer” on page 110, “Setting the ETS-Related Datalink Properties” on page 108, and Example 6-8.
- **Displaying the effective value of the datalink properties** – The `dladm show-linkprop` subcommand is enhanced to display the `EFFECTIVE` field for some datalink properties. The values for the `EFFECTIVE` field is determined by the system based on the availability of the resource, capability of the underlying device, or negotiations with the peer. The effective value need not be the same as the configured values. A datalink property can have an effective value, even though the property is not configured with a value. For an example about how to display the `EFFECTIVE` field, see “Displaying Datalink Properties” on page 104 and Example 6-7.
- **Displaying bridge statistics** – The `dlstat show-bridge` subcommand displays the statistics of the bridges and the statistics of the links connected to each bridge and also shows the nested view of the bridge statistics. For more information, see “Displaying Information About Configured Bridges” on page 71.

Features and Components That Are Used to Manage Network Datalinks

Managing network datalinks refers to the use of features and technologies to fine tune the way your systems process the 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. Although these features address different areas of network operations, they provide common benefits such as network connectivity, network administration, and efficiency.

Link Aggregations

Link aggregations enable you to pool multiple datalink resources that you administer as a single unit. You can improve the bandwidth and provide high availability for applications by combining multiple physical NICs together. Link aggregation of network datalinks ensures that a system has continuous access to the network. Trunk aggregation and DLMP aggregation are the two types of link aggregation.

Trunk aggregation provides consolidated bandwidth of the underlying datalinks for the clients configured over the aggregation. DLMP aggregation provides high availability across multiple

switches for the clients configured over the aggregation. DLMP aggregation also supports link-based failure detection and probe-based failure detection to ensure continuous availability of the network to send and receive traffic. For more information about different types of link aggregation and procedures for configuring and administering link aggregations, see Chapter 2, “Configuring High Availability by Using Link Aggregations”.

Virtual Local Area Networks

Virtual local area networks (VLANs) enable you to divide your network into subnetworks without having to add resources to the physical network environment. Therefore, the subnetworks are virtual and you use the same physical network resources. VLANs provide applications with isolated subnetworks so that only the applications in the same VLAN can communicate with each other. You can configure multiple virtual networks within a single network unit, for example, a switch by combining VLANs and Oracle Solaris zones. For more information about VLANs and procedures to configure and administer VLANs, see Chapter 3, “Configuring Virtual Networks by Using Virtual Local Area Networks”.

Bridged Networks

Bridges 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. Bridges use a packet-forwarding mechanism to connect subnetworks together and enable a system to transmit packets to their destinations by using the shortest connection routes. For more information about bridged networks and procedures to administer bridges, see Chapter 4, “Administering Bridging Features”.

Link Layer Discovery Protocol

Link Layer Discovery Protocol (LLDP) enables exchanging of connectivity and management information between the systems on the network for the purpose of topology discovery. The information can include system capabilities, management addresses, and other information relevant to network operations. The network diagnostics service uses LLDP to detect problems that might lead to limited or degraded network connectivity. For more information about LLDP and procedures to configure LLDP, see Chapter 5, “Exchanging Network Connectivity Information With Link Layer Discovery Protocol”.

Data Center Bridging

Data center bridging (DCB) is used to manage the bandwidth, relative priority, and flow control of multiple traffic types when sharing the same network link, for example, when sharing a datalink between networking and storage protocols. DCB enables information exchange with the peers about the features that support converged network by using LLDP. The information is related to the configurations affecting the integrity of network packets especially in heavy traffic environments, such as data centers. DCB enables efficient network infrastructure by consolidating storage area network (SAN) and local area network (LAN) and thereby reducing operational and management costs in data centers.

You can configure DCB features such as priority-based flow control (PFC) for the prevention of packet loss and enhanced transmission selection (ETS) for bandwidth sharing among packets based on class of service (CoS) priorities. For more information, see Chapter 6, “Managing Converged Networks by Using Data Center Bridging”.

♦ ♦ ♦ CHAPTER 2

Configuring High Availability by Using Link Aggregations

This chapter provides an overview of link aggregations and describes procedures to configure and administer link aggregations.

This chapter contains the following topics:

Overview of Link Aggregations

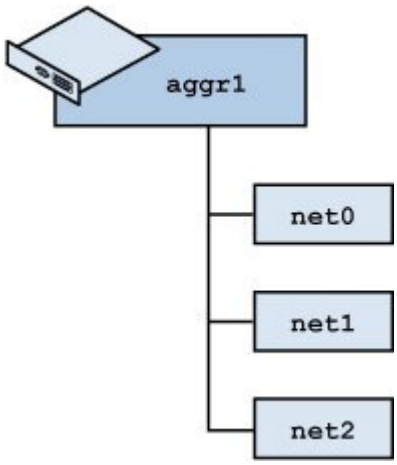
Link aggregation is a method to combine several physical datalinks on a system to improve the bandwidth for applications. These physical datalinks are configured together as a single logical unit to increase the throughput of network traffic and to achieve high availability at the datalink layer.

You can include any link aggregation that is listed in the available NICs as an individual interface by using Oracle Enterprise Manager Ops Center. You can also display the details of

both trunk and DLMP aggregation by using Oracle Enterprise Manager Ops Center. For more information about the Oracle Enterprise Manager Ops Center, see http://www.oracle.com/pls/topic/lookup?ctx=oc122&id=OPCCM.

The following figure shows an example of a simple link aggregation configured on a system.

FIGURE 2-1 Link Aggregation Configuration



The illustration shows an aggregation `aggr1` that consists of three underlying datalinks, `net0`, `net1`, and `net2`. These datalinks are dedicated to serve the traffic that traverses the system through the aggregation. The underlying links are hidden from external applications. Instead, the logical datalink `aggr1` is accessible.

Benefits of Link Aggregation

Link aggregation has the following benefits:

- **Increased bandwidth** – The capacity of multiple links is combined into one logical link.
- **Automatic failover and failback** – The traffic from a failed link is automatically switched over to other working links in the aggregation, thereby achieving high availability.
- **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.

Because link aggregation groups multiple links into a single logical datalink, features of datalinks such as link protection and resource management work well with link aggregations.

For information about link protection, see Chapter 1, “Using Link Protection in Virtualized Environments,” in “Securing the Network in Oracle Solaris 11.2 ”. For information about resource management, see Chapter 7, “Managing Network Resources,” in “Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ”.

Link aggregation overcomes some of the problems that are encountered when using high availability features with network virtualization, for example IPMP. With link aggregation, you create the aggregation in a global zone first, then specify it as the underlying link when you configure the non-global zone. When the zone boots, it is assigned a virtual network interface card (VNIC) on the link aggregation. Unlike IPMP, which needs to be configured in every non-global zone, link aggregations are configured in global zones only and provide highly available VNICs to non-global zones. The global zone can also configure properties such as bandwidth on the VNICs assigned to the zone.

Note - Link aggregations perform similar functions to IP multipathing (IPMP) to improve network performance and availability at the datalink layer. For a comparison of these two technologies, see Appendix A, “Link Aggregations and IPMP: Feature Comparison”.

The following types of link aggregations are supported:

- Trunk aggregations
- Datalink multipathing (DLMP) aggregations

For the differences between these two types of link aggregations, see “Comparing Trunk and DLMP Aggregation” on page 40.

Trunk Aggregations

Trunk aggregations are based on the IEEE 802.3ad standard and work by enabling multiple flows of traffic to be spread across a set of aggregated ports. IEEE 802.3ad requires switch configuration, as well as switch-vendor proprietary extensions in order to work across multiple switches. In trunk aggregations, the clients configured over the aggregations get a consolidated bandwidth of the underlying links, because each network port is associated with every configured datalink over the aggregation. When you create a link aggregation, the aggregation is by default created in the trunk mode. You might use a trunk aggregation in the following situations:

- For systems in the network that run applications with distributed heavy traffic, you can dedicate a trunk aggregation to that application's traffic to take advantage of the increased bandwidth.
- For sites with limited IP address space that require large amounts of bandwidth, you need only one IP address for the trunk aggregation of datalinks.

- For sites that need to hide any internal datalinks, the IP address of the trunk aggregation hides these datalinks from external applications.
- For applications that need reliable network connection, trunk aggregation protects network connections against link failure.

Trunk aggregation supports the following features:

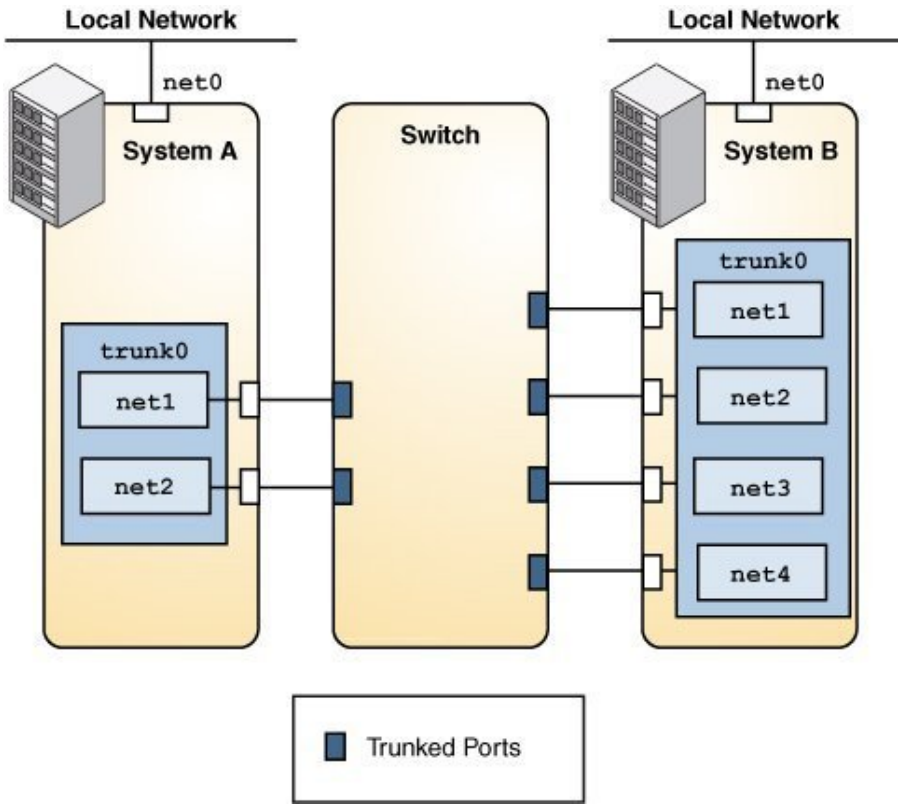
- Using a switch
- Using a switch with the Link Aggregation Control Protocol (LACP)
- Back-to-back trunk aggregation configuration
- Aggregation policies and load balancing

The following sections describe the features of trunk aggregations.

Using a Switch

Systems that are configured with trunk aggregations might use an external switch to connect to other systems. The following figure depicts a local network with two systems where each system has a trunk aggregation configured.

FIGURE 2-2 Trunk Aggregation Using a Switch

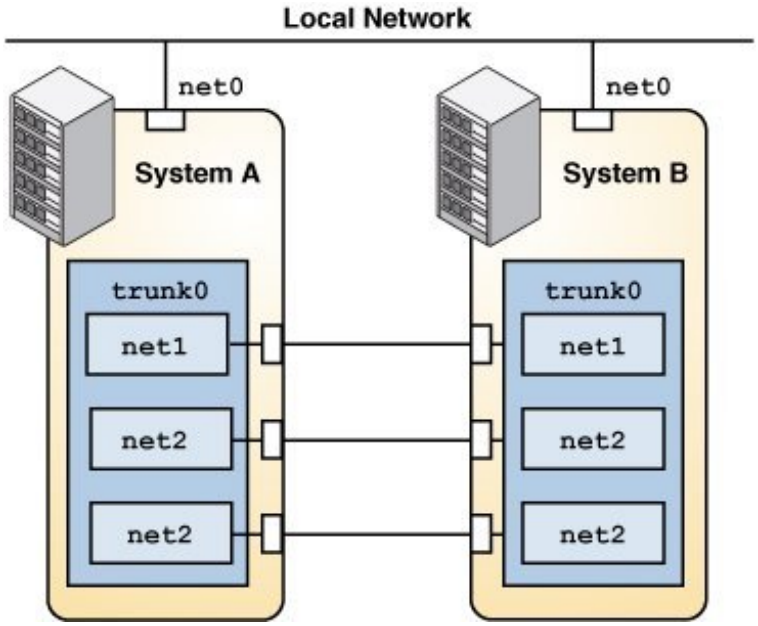


The two systems are connected by a switch. System A has a trunk aggregation that consists of two datalinks, `net1` and `net2`. These datalinks are connected to the switch through aggregated ports. System B has a trunk aggregation of four datalinks, `net1` through `net4`. These datalinks are also connected to the aggregated ports on the switch. In this trunk aggregation topology, the switch must support the IEEE 802.3ad standard and the switch ports must be configured for aggregation. See the switch manufacturer's documentation to configure the switch.

Back-to-Back Trunk Aggregation Configuration

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

FIGURE 2-3 Back-to-Back Trunk Aggregation Configuration



The illustration shows trunk aggregation `trunk0` on System A directly connected to trunk aggregation `trunk0` on System B by means of the corresponding links between their respective underlying datalinks. This setup enables Systems A and B to provide redundancy, high availability, and high-speed communication between both systems. Each system also has `net0` configured for traffic flow within the local network.

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

Using a Switch With the Link Aggregation Control Protocol

If your setup of a trunk aggregation has a switch and the switch supports LACP, you can enable LACP for the switch and the system. See the switch manufacturer's documentation to configure the switch.

LACP enables a more reliable detection method of datalink failures. Without LACP, a link aggregation relies only on the link state reported by the device driver to detect the failure of an aggregated datalink. With LACP, LACPDUs are exchanged at regular intervals to ensure that the aggregated datalinks can send and receive traffic. LACP also detects some misconfiguration cases, for example, when the grouping of datalinks does not match between the two peers.

LACP exchanges special frames called Link Aggregation Control Protocol Data Units (LACPDUs) between the aggregation and the switch if LACP is enabled on the system. LACP uses these LACPDUs to maintain the state of the aggregated datalinks.

Use the `dladm create-aggr` command to configure an aggregation's LACP to one of the following three modes:

- `off` – The default mode for aggregations. The system does not generate LACPDUs.
- `active` – The system generates LACPDUs at specified intervals.
- `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 do not exchange LACPDUs.

  For information about how to configure LACP, see "How to Create a Link Aggregation" on page 26.

Defining Aggregation Policies for Load Balancing

You can define a policy for the outgoing traffic that specifies how to distribute load across the available links of an aggregation, thus establishing load balancing. You can use the following load specifiers to enforce various load balancing policies:

- L2 – Determines the outgoing link by using the MAC (L2) header of each packet
- L3 – Determines the outgoing link by using the IP (L3) header of each packet
- L4 – Determines the outgoing link by using the TCP, UDP, or other ULP (L4) header of each packet

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

Datalink Multipathing Aggregations

Datalink multipathing (DLMP) aggregation is a type of link aggregation that provides high availability across multiple switches without requiring switch configuration. DLMP aggregation supports link-based failure detection and probe-based failure detection to ensure continuous availability of the network to send and receive traffic.

Advantages of DLMP Aggregation

DLMP aggregation provides the following advantages:

- The IEEE 802.3ad standard implemented by trunk aggregations does not have provisions to span multiple switches. Allowing failover between multiple switches in trunk mode requires vendor-proprietary extensions on the switches that are not compatible between vendors. DLMP aggregations allow failover between multiple switches without requiring any vendor-proprietary extensions.
- Using IPMP for high availability in the context of network virtualization is very complex. An IPMP group cannot be assigned directly to a zone. When network interface cards (NICs) have to be shared between multiple zones, you have to configure VNICs so that each zone gets one VNIC from each of the physical NICs. Each zone must group its VNICs into an IPMP group to achieve high availability. The complexity increases as you scale configurations, for example, in a scenario that includes large numbers of systems, zones, NICs, virtual NICs (VNICs), and IPMP groups. With DLMP aggregations, you create a VNIC or configure a zone's `anet` resource on top of the aggregation, and the zone sees a highly available VNIC.
- DLMP aggregation enables you to use the features of link layer such as link protection, user-defined flows, and the ability to customize link properties, such as bandwidth.

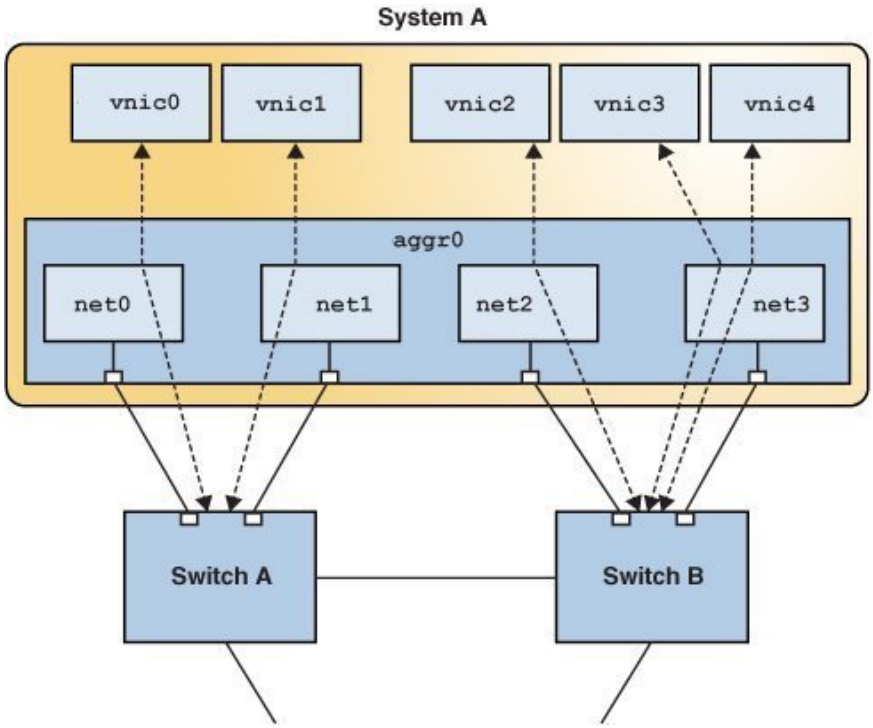
Note - You must not configure an IPMP group over a DLMP aggregation. However, you can configure an IPMP group over a trunk aggregation.

How DLMP Aggregation Works

In a trunk aggregation, each 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.

The following figure shows how a DLMP aggregation works.

FIGURE 2-4 DLMP Aggregation



The figure shows System A with link aggregation `aggr0`. The aggregation consists of four underlying links, from `net0` to `net3`. VNICs `vnic0` through `vnic4` are also configured over the aggregation. The aggregation is connected to Switch A and Switch B, which in turn connect to other destination systems in the wider network.

VNICs are associated with aggregated ports through the underlying links. For example, in the figure, `vnic0` through `vnic3` are associated with the aggregated ports through the underlying links `net0` through `net3`. That is, if the number of VNICs and the number of underlying links are equal, then each port is associated with an underlying link.

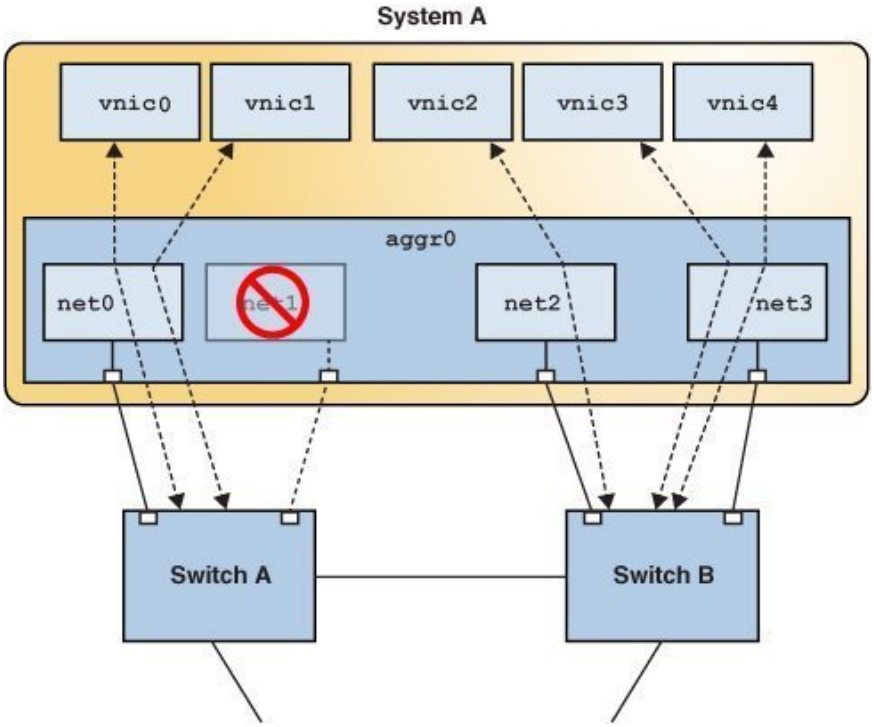
If the number of VNICs exceeds the number of underlying links, then one port is associated with multiple datalinks. For example, in the figure the total number of VNICs exceeds the number of underlying links. Hence, `vnic4` shares a port with `vnic3`.

When an aggregated port fails, all the datalinks that use that port are distributed among the other ports, thereby providing network high availability during failover. For example, if `net0` fails, then DLMP aggregation shares the remaining port `net1`, between VNICs. The distribution

among the aggregated ports occurs transparently to the user and independently of the external switches connected to the aggregation.

The following figure shows how DLMP aggregation works when a port fails. In the figure, `net1` has failed and the link between switch and `net1` is down. `vnic1` shares a port with `vnic0` through `net0`.

FIGURE 2-5 DLMP Aggregation When a Port Fails



Failure Detection in DLMP Aggregation

Failure detection in DLMP aggregation is a method to detect the failure of the aggregated ports. A port is considered to have failed when it cannot send or receive traffic. The port might fail because of the following reasons:

- Damage or cut in the cable

- Switch port goes down
- Failure in upstream network path

DLMP aggregation performs failure detection on the aggregated ports to ensure continuous availability of the network to send or receive traffic. When a port fails, the clients associated with that port are failed over to an active port. Failed aggregated ports remain unusable until they are repaired. The remaining active ports continue to function while any existing ports are deployed as needed. After the failed port recovers from the failure, clients from other active ports can be associated with it.

DLMP aggregation supports both link-based and probe-based failure detection.

Link-Based Failure Detection

Link-based failure detection detects failure when the cable is cut or when the switch port is down. It therefore can only detect failures caused by the loss of direct connection between the datalink and the first-hop switch. Link-based failure detection is enabled by default when a DLMP aggregation is created.

Probe-Based Failure Detection

Probe-based failure detection detects failures between an end host and the configured targets. This feature overcomes the known limitations of link-based failure detection. Probe-based failure detection is useful when a default router is down or when the network becomes unreachable. The DLMP aggregation detects failure by sending and receiving probe packets.

To enable probe-based failure detection in DLMP aggregation, you must configure the `probe-ip` property.

Note - In a DLMP aggregation, if no `probe-ip` is configured, then probe-based failure detection is disabled and only link-based failure detection is used.

When you create the first DLMP aggregation, the service `svc:/network/dlmp:default` is automatically enabled. This service starts the `in.dlmpd` daemon, which performs probe-based failure detection in DLMP aggregations. This service is disabled when no DLMP aggregation is in the system. For information, see “Configuring Probe-Based Failure Detection for DLMP Aggregation” on page 31.

Probe-based failure detection is performed by using the combination of two types of probes: Internet Control Message Protocol (ICMP (L3)) probes and transitive (L2) probes, which work together to determine the health of the aggregated physical datalinks.

- **ICMP Probing**

  You can configure a comma-separated list of source IP addresses and an optional target IP address or host name. The target IP address must be on the same subnet as the specified source IP address. You can specify the source IP address in four different forms. For more information, see "How to Configure Probe-Based Failure Detection for DLMP" on page 32.

  ICMP probing uses the configured source IP addresses of the `probe-ip` property only if the IP addresses are associated with clients such as VNICs. A port is associated with a client such as a VNIC only when the port receives inbound traffic and transmits outbound traffic for that client. At any particular time, the inbound or the outbound traffic for a client always goes through only one underlying port of the DLMP aggregation. The configured IP addresses of the `probe-ip` property are used to monitor the health of the ports only if the IP addresses are associated with that port.

  For each configured source IP address, the `in.dlmpd` daemon periodically sends out unicast ICMP packets directed at the configured targets. If target IP addresses are not configured, `in.dlmpd` uses the routing table for routes on the same subnet as the specified source IP address, and uses the specified next-hop as the target IP address.

  ICMP probe traffic is sent out only through the port associated with that IP client. The port is marked as *ICMP failed* if all the targets for that particular port become unreachable. The port is marked as *ICMP active* if at least one of the targets is reachable from that port through an ICMP probe.

- **Transitive Probing**

  Transitive probing is performed when the state of the health for all the network ports cannot be determined by ICMP probing. Hence, transitive probing is performed when any port is not associated with the source IP address that is configured for the `probe-ip` property. For example, transitive probing is performed when any port is not associated with an IP client or when the number of configured IP addresses of the `probe-ip` property is less than the total number of aggregated ports. Probe packets are sent periodically from the ports that are not associated with any IP client to the peer ports. If a port is able to reach any ICMP active port, then that port is considered *L2 active*.

  Oracle Solaris includes proprietary protocol packets for transitive probes that are transmitted over the network. For more information, see Appendix B, "Packet Format of Transitive Probes".

Probe-based failure detection is performed in the global zone when VNICs over an aggregation are created in the global zone and are assigned to non-global zones. However, probe traffic can be segregated from the non-global zone with the help of VLANs. For example, when the probe traffic runs on one VLAN in a global-zone, the non-global zone traffic can run on a different VLAN.

Requirements for Link Aggregations

The link aggregation configuration has the following requirements:

- The datalinks to be configured into an aggregation should not have any IP interface configured over them.
- You can create link aggregations only from the global zone. You cannot use datalinks to create a link aggregation from a non-global zone even if you do not have any IP interface configured over the datalinks. Link aggregation only combines multiple physical NICs but in a non-global zone all interfaces are virtual. Therefore, you cannot create a link aggregation from the non-global zone.
- 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 the 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 information, see "How to Ensure That the MAC Address of Each Interface Is Unique" in "Configuring and Administering Network Components in Oracle Solaris 11.2 ".
- (Trunk aggregation only) Devices must support link state notification as defined in the IEEE 802.3ad Link Aggregation Standard in order for a port to attach to a trunk aggregation or to detach from a trunk 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 26.

Creating a Link Aggregation

Link aggregation groups the underlying ports into a single logical group. The aggregation uses these underlying ports exclusively and you cannot perform any other operation such as configuring VNICs or assigning IP addresses on these ports. However, you can configure VNICs on top of the aggregation and not on the individual ports.

You must remove any existing IP interface on these ports before creating the link aggregation.

You can also configure a VLAN and create VNICs over the link aggregation that you have created. For information about how to create VLANs over link aggregations, see "How to Configure VLANs Over a Link Aggregation" on page 52.

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

How to Create a Link Aggregation

Before You Begin If you are creating a trunk aggregation and are using a switch in the aggregation, configure the ports to be used as an aggregation on the switch. If the switch supports LACP, configure LACP in either `active` or `passive` mode.

See the switch manufacturer's documentation to configure the switch.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Display the datalink information to identify the physical datalinks for the aggregation.**

   ```
   # dladm show-phys
   ```

3. **Ensure that the datalink that you intend to aggregate are not in use by any application.**

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

   a. **Determine the link state.**

      ```
      # 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. **Remove the IP interface.**

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

      where *interface* specifies the IP interface over the link. For more information, see the `ipadm`(1M) man page.

4. **Create a link aggregation.**

   ```
   # dladm create-aggr [-f] [-m mode] [-P policy]  [-L LACP-mode] \
   [-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.

   `-m` *mode* Mode must be set to one of the following values. The default mode is `trunk`.

- `trunk` – IEEE 802.3ad compliant link aggregation mode
- `dlmp` – Datalink multipathing mode

`-P` *policy* — (Trunk aggregation only) Specifies the load balancing policy for the aggregation. Supported values are `L2`, `L3`, and `L4`. For more information, see "Defining Aggregation Policies for Load Balancing" on page 19.

`-L` *LACP-mode* — (Trunk aggregation only) Specifies the mode of LACP if it is used. Supported values are `off`, `active`, or `passive`. For information about the modes, see "Using a Switch" on page 16.

`-T` *time* — (Trunk aggregation only) Specifies the LACP timer value. The supported values are `short` or `long`.

`-u` *address* — Specifies the 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 information about the rules to assign names, see "Rules for Valid Link Names" in "Configuring and Administering Network Components in Oracle Solaris 11.2 ".

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

- Display the aggregations and links with the status information.

```
# dladm show-link
```

- Display the aggregations with the status and per port information.

```
# dladm show-aggr -x
```

The aggregation's state should be `up`.

Example 2-1 Creating a Trunk Aggregation

This example shows the commands 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         --
# ipadm delete-ip net0
# dladm create-aggr -L active -l net0 -l net1 trunk0
# dladm show-aggr -x
```

```
LINK        PORT           SPEED DUPLEX   STATE     ADDRESS            PORTSTATE
trunk0      --             1000Mb full    up        8:0:27:49:10:b8    --
            net0           1000Mb full    up        8:0:27:49:10:b8    attached
            net1           1000Mb full    up        8:0:27:e4:d9:46    attached
```

Example 2-2 Creating a DLMP Aggregation and Configuring an IP Interface on Top of the Aggregation

This example shows how to create a DLMP aggregation. The aggregation has three underlying datalinks `net0`, `net1`, and `net2`. An IP interface is created on top of the aggregation `aggr0` and a VNIC `vnic1` is created on top of it.

```
# dladm create-aggr -m dlmp -l net0 -l net1 -l net2 aggr0
# dladm show-link
LINK      CLASS     MTU     STATE   OVER
net0      phys      1500    up      --
net1      phys      1500    up      --
net2      phys      1500    up      --
aggr0     aggr      1500    up      net0 net1 net2
# dladm show-aggr -x
LINK        PORT           SPEED DUPLEX   STATE     ADDRESS            PORTSTATE
aggr0       --             1000Mb full    up        8:0:27:49:10:b8    --
            net0           1000Mb full    up        8:0:27:49:10:b8    attached
            net1           1000Mb full    up        8:0:27:e4:d9:46    attached
            net2           1000Mb full    up        8:0:27:38:7a:97    attached
# ipadm create-ip aggr0
# ipadm create-addr -T static -a local=10.10.10.1 aggr0/v4
# dladm create-vnic -l aggr0 vnic1
```

Next Steps You can perform further configuration of the aggregation such as creating IP interfaces and VNICs. You can use the created aggregation for configuring both non-global zones and kernel zones.

- For information about creating IP interfaces, see Chapter 3, “Configuring and Administering IP Interfaces and Addresses in Oracle Solaris,” in “Configuring and Administering Network Components in Oracle Solaris 11.2 ”.
- For information about configuring VLANs over a link aggregation, see “Configuring VLANs Over a Link Aggregation” on page 52.
- For information about configuring VNICs, see “How to Configure VNICs and Etherstubs” in “Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ”.
- For information about configuring zones, see “Creating and Using Oracle Solaris Zones ”.

Adding a Link to an Aggregation

You can include additional datalinks to an existing aggregation. If you are adding datalinks to a trunk aggregation, you might have to reconfigure the switch to accommodate the additional datalinks even though LACP is configured on the switch.

How to Add a Link to an Aggregation

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Ensure that no IP interfaces are configured over the link. If any IP interface is configured, delete that IP interface.**

   ```
   # ipadm show-if
   # 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. **(Trunk aggregations only) If necessary, reconfigure the switch.**

   You might need to reconfigure the switch to accommodate the additional datalinks based on how the switch is configured, even if LACP is enabled on the switch.

   See the switch manufacturer's documentation to perform any reconfiguration tasks on the switch.

Example 2-3 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     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     OVER
net0     phys      1500    up        --
net1     phys      1500    up        --
aggr0    aggr      1500    up        net0 net1 net3
net3     phys      1500    up        --
```

Removing a Link From an Aggregation

You can remove the individual datalinks associated with an aggregation by using the `dladm remove-aggr` command. You need to reconfigure the switch when you remove links from the aggregation.

Become an administrator and use the following command:

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

EXAMPLE 2-4 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     OVER
net0     phys      1500    up        --
net1     phys      1500    up        --
aggr0    aggr      1500    up        net0 net1
net3     phys      1500    unknown   --
```

Modifying a Trunk Aggregation

You can modify selected attributes such as `policy`, `lacpmode`, and `time` for the trunk aggregation. These attributes are not supported in DLMP aggregations.

- To modify the load balancing policy of the aggregation, become an administrator and issue the following command:

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

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

  aggr — Specifies the aggregation whose policy you want to modify.

- To modify the LACP mode of the aggregation, become an administrator and use the following command:

  ```
  # dladm modify-aggr -L LACP-mode -T time aggr
  ```

`-L` *LACP-mode*	Indicates the LACP mode in which the aggregation must operate. Possible values are `active`, `passive`, and `off`.
`-T` *time*	Indicates the LACP timer value, either `short` or `long`.
aggr	Specifies the aggregation whose LACP mode you want to modify.

EXAMPLE 2-5 Modifying a Trunk Aggregation

This example shows how to modify the load balancing policy of link aggregation `aggr0` to L2 and to change the LACP mode to `active`.

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

Configuring Probe-Based Failure Detection for DLMP Aggregation

You must configure the `probe-ip` property for a DLMP aggregation to enable probing. Otherwise, probing is disabled by default and only link-based failure detection is used. For more information, see "Failure Detection in DLMP Aggregation" on page 22.

The following datalink properties are used to configure probe-based failure detection:

- `probe-ip` – Specifies a comma separated list of source IP addresses and a target IP address or host name. The source IP addresses are used for ICMP probing. The list of source IP addresses is followed by an optional target address. The target IP address must be on the same subnet as the specified source IP address.

  You can use + to separate the source from the target. You can specify the targets as an IP address or the host name of the target. For information about how to specify the source address and the target address, see "How to Configure Probe-Based Failure Detection for DLMP" on page 32.
- `probe-fdt` – Specifies the failure detection time. You can configure the expected failure detection time value in seconds. The default value is 10 seconds.

How to Configure Probe-Based Failure Detection for DLMP

Before You Begin Create a DLMP aggregation. For more information, see "How to Create a Link Aggregation" on page 26.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **(Optional) Display all the existing aggregations to identify the aggregation for configuring probe-based failure detection.**

   ```
   # dladm show-aggr
   ```

3. **Set the probe targets for the aggregation for which you want to configure probe-based failure detection.**

   ```
   # dladm set-linkprop -p probe-ip=IP-address[/prefix-length]|hostname+[target]  aggr
   ```

   The source and destination addresses for the `probe-ip` property can be specified in the following ways:

   - `probe-ip=`*IP-address*[/*prefix-length*]|*hostname*+[*target*]

     IP address and its prefix length. For example, `10.130.10.1/24+`.
   - `probe-ip=`*addr-obj-name*+[*target*]

     Address object name. For example, `vnic1/addr1+192.168.0.1`.
   - `probe-ip=`*interface-name*+[*target*]

     Interface name, which can be either the name of the aggregation interface itself or any VNIC configured over the aggregation. For example, `[aggr1]+`.
   - `probe-ip=+`[*target*]

     No IP address is specified. When the source IP address is not specified, all the IP addresses configured on the aggregation and the VNICs are considered as the potential source IP addresses of ICMP probes. For example, `+10.130.10.1`.

4. **(Optional) Set the failure detection time.**

   ```
   # dladm set-linkprop -p probe-fdt=fdt  aggr
   ```

   where *fdt* is the failure detection time specified in seconds. The default value is 10 seconds.

5. **(Optional) Display the aggregation to see the probe-related information.**

   ```
   # dlstat show-aggr -n -P [[t],[i],[all]]
   ```

Example 2-6 Configuring Probe-Based Failure Detection

1. Display the existing aggregations.

```
# dladm show-aggr
LINK    MODE       POLICY  ADDRPOLICY      LACPACTIVITY   LACPTIMER
aggr0   dlmp       --      --              --             --
aggr1   dlmp       --      --              --             --
```

2. Set the probe targets for `aggr1`.

```
# dladm set-linkprop -p probe-ip=+ aggr1
```

   Since the source IP address is not specified, all the IP addresses configured on the aggregation `aggr1` and the VNICs become the source IP addresses of ICMP probes.

3. Set the failure detection time.

```
# dladm set-linkprop -p probe-fdt=15 aggr1
```

4. Display the properties that are set.

```
# dladm show-linkprop -p probe-ip,probe-fdt aggr1
LINK     PROPERTY         PERM   VALUE           EFFECTIVE    DEFAULT      POSSIBLE
aggr1    probe-ip         rw     192.168.85.137   --           --        --
aggr1    probe-fdt        rw     15               15           10          1-600
```

5. Display the statistics of the probes for the aggregation.

```
# dlstat show-aggr -n -P t,i aggr1
TIME   AGGR   PORT  LOCAL           TARGET          PROBE  NETRTT     RTT
0.45s  aggr1  net0  net0            net1            t16148  --         --
0.45s  aggr1  net0  net0            net1            t16148  0.63ms     0.81ms
1.08s  aggr1  net1  net1            net0            t16148  --         --
1.08s  aggr1  net1  net1            net0            t16148  0.72ms     0.99ms
2.07s  aggr1  net1  192.168.85.137  192.168.85.137  i15535  --         --
2.07s  aggr1  net1  192.168.85.137  192.168.85.137  i15535  0.18ms     0.54ms
```

Monitoring Probe-Based Failure Detection

You can monitor probe-based failure detection by using the `dladm show-aggr`, `dlstat show-aggr`, and `ipadm show-addr` commands.

EXAMPLE 2-7 Displaying Probe-Related Information

The following example displays the statistics of the probes for a DLMP aggregation. With the probe type `-P` option, you can provide a comma-separated list of arguments (`t` for transitive

probe, `i` for ICMP probe, and `all` for both ICMP and transitive probes) in the `dlstat show-aggr` command to display the probes of the respective probe type.

```
# dlstat show-aggr -n -P t,i aggr1
   TIME     AGGR     PORT   LOCAL        TARGET       PROBE  NETRTT     RTT
   0.53s    aggr1    net0   net0         net1         t16148  --         --
   0.53s    aggr1    net0   net0         net1         t16148  0.62ms     0.87ms
   1.17s    aggr1    net1   net1         net0         t16148  --         --
   1.17s    aggr1    net1   net1         net0         t16148  0.72ms     0.99ms
   2.24s    aggr1    net1   192.168.0.1  192.168.0.2  i15535  --         --
   2.24s    aggr1    net1   192.168.0.1  192.168.0.2  i15535  0.11ms     0.55ms
```

`TIME` — Time at which the probe is sent in seconds. This time is relative to the time when you issue the `dlstat` command. If the probe is sent before you issue the `dlstat` command, the time is negative.

`AGGR` — Aggregation name for which the probe is sent.

`LOCAL` — ICMP probes: Source IP address of the probes.

Transitive probes: Port name from which the transitive probe originated.

`TARGET` — ICMP probes: Destination IP address of the probes.

Transitive probes: Port name of the probe that is targeted.

`PROBE` — Identifier number representing the probe. The prefix `t` is for transitive probes and the prefix `i` is for ICMP probes.

`NETRTT` — Network round-trip-time for the probe. This value is the time period between sending the probe and receiving the acknowledgment by the DLMP aggregation.

`RTT` — Total round-trip-time for the probe. This value is the time period between sending the probe and completing the process of acknowledgment by the DLMP aggregation.

For more information, see the `dlstat`(1M) man page.

EXAMPLE 2-8 Displaying Detailed Information About the Aggregated Port

The following example displays the detailed aggregation information on each underlying port.

```
# dladm show-aggr -x
LINK        PORT    SPEED  DUPLEX   STATE    ADDRESS            PORTSTATE
aggr1       --      100Mb  full     up       1e:34:db:fa:50:a2  --
            net0    100Mb  full     up       1e:34:db:fa:50:a2  attached
            net1    100Mb  full     up       b2:c0:6a:3e:c5:b5  attached
```

For more information, see the `dladm`(1M) man page.

EXAMPLE 2-9 Displaying the State of the Aggregated Ports

The following example shows the state of the ports of the aggregation and the target IP addresses of the ports.

```
# dladm show-aggr -S -n
LINK        PORT       FLAGS   STATE    TARGETS        XTARGETS
aggr1       net1       u--3    active   192.168.0.2    net0
--          net0       u-2-    active   --             net1
```

EXAMPLE 2-10 Displaying `probe-ip` Property Values

The following example displays the details about the link property `probe-ip` for the specified DLMP aggregation.

```
# dladm show-linkprop -p probe-ip aggr1
LINK     PROPERTY         PERM   VALUE         EFFECTIVE    DEFAULT      POSSIBLE
aggr1    probe-ip         rw     192.168.0.2   192.168.0.2  --           --
```

EXAMPLE 2-11 Displaying the IP Address and the State of the Aggregation

The following example displays the IP address and the state of the aggregation.

```
# ipadm show-addr aggr1
ADDROBJ           TYPE     STATE        ADDR
aggr1/local1      static   ok           192.168.0.1/24
```

Deleting a Link Aggregation

You can delete a link aggregation by using the `dladm delete-aggr` command. You must remove the IP interface and VNICs configured over the link aggregation before deleting the aggregation.

▼ How to Delete a Link Aggregation

1. **Become an administrator.**

   For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

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 2-12 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
```

Switching Between Trunk and DLMP Aggregations

Switching between a trunk aggregation and a DLMP aggregation changes the entire configuration, so it affects the aggregation in a more comprehensive way than simply modifying the other link aggregation properties.

▼ How to Switch Between Link Aggregation Types

Before You Begin

- 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.
- If you switch from a DLMP aggregation, you must ensure that all the links are on the same switch and the switch is configured for the aggregation.

1. **Become an administrator.**

   For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

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

```
# dladm show-aggr
```

   The output `MODE` field indicates the current type of the aggregation. The value of `MODE` is `trunk` for a trunk aggregation and `dlmp` for a DLMP aggregation.

3. **Switch the aggregation.**

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

where *mode* is `trunk` if you are switching to a trunk aggregation or `dlmp` if you are switching to a DLMP aggregation, and *aggr* is the name of the aggregation.

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

   See the switch manufacturer's documentation for configuring the switch.

5. **(Optional) Verify the current link aggregation configuration.**

   ```
   # dladm show-aggr
   ```

Example 2-13 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   trunk      L2      auto            active         short

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

Once the aggregation is switched, you must remove the previous switch configuration that was applied to the trunk aggregation.

Use Case: Configuring a Link Aggregation

The following end-to-end use case shows how to accomplish the following actions:

- Create a DLMP aggregation.
- Add links to the aggregation.
- Configure an IP interface over the aggregation.
- Configure a VNIC over the aggregation.
- Configure probe-based failure detection for the aggregation.
- Configure the target IP address in the routing table.
- Monitor the ICMP and transitive probes.

1. Become an administrator.

   For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

2. Display datalink information to identify the datalinks for aggregation.

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

3. Ensure that the datalinks that you want aggregate do not have IP interfaces configured over the link. Delete the interface if any interface is configured on any of the links.

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

4. Create a DLMP aggregation with the links `net0` and `net1`.

```
# dladm create-aggr -m dlmp -l net0 -l net1 aggr1
```

5. Add another link, `net2`, to the aggregation.

```
# dladm add-aggr -l net2 aggr1
```

   Reconfigure the switch to accommodate the new links if the existing switch configuration requires it. See the switch manufacturer's documentation.

6. Configure an IP interface on top of the aggregation `aggr1`.

```
# ipadm create-ip aggr1
# ipadm create-addr -T static -a local=10.10.10.1 aggr1/v4
```

7. Create a VNIC on top of the aggregation.

```
# dladm create-vnic -l aggr1 vnic1
```

8. Configure probe-based failure detection for the aggregation.

```
# dladm set-linkprop -p probe-ip=+ aggr1
```

   Source IP address and the target IP address of the probes are not specified. Hence, you need to configure the target in the routing table in order to enable probing.

9. Configure the target in the routing table to be in the same subnet as the specified IP address.

```
# route add -host 10.10.10.2 10.10.10.2 -static
```

10. Display the state of the aggregated ports and the targets.

```
# dladm show-aggr -S
LINK       PORT        FLAGS   STATE     TARGETS         XTARGETS
aggr1      net0        u--3    active    10.10.10.2      net2 net1
--         net1        u-2-    active       --           net2 net0
--         net2        u-2-    active       --           net0 net1
```

11. Monitor the ICMP probe statistics.

```
# dlstat show-aggr -n -P i
   TIME     AGGR      PORT   LOCAL          TARGET        PROBE NETRTT  RTT
   1.16s    aggr1     net0   10.10.10.1     10.10.10.2    i33   --      --
   1.16s    aggr1     net0   10.10.10.1     10.10.10.2    i33   0.08ms  0.33ms
   2.05s    aggr1     net0   10.10.10.1     10.10.10.2    i34   --      --
   2.05s    aggr1     net0   10.10.10.1     10.10.10.2    i34   0.01ms  0.64ms
   4.05s    aggr1     net0   10.10.10.1     10.10.10.2    i35   --      --
   4.05s    aggr1     net0   10.10.10.1     10.10.10.2    i35   0.10ms  0.35ms
   5.54s    aggr1     net0   10.10.10.1     10.10.10.2    i36   --      --
   5.54s    aggr1     net0   10.10.10.1     10.10.10.2    i36   0.08ms  0.34ms
```

12. Monitor the transitive probe statistics between the ports.

```
# dlstat show-aggr -n -P t
  TIME     AGGR   PORT        LOCAL       TARGET  PROBE NETRTT  RTT
  0.30s    aggr1  net2        net2        net0    t38   --      --
  0.30s    aggr1  net2        net2        net0    t38   0.46ms  0.59ms
  0.46s    aggr1  net0        net0        net1    t39   --      --
  0.46s    aggr1  net0        net0        net1    t39   0.46ms  0.50ms
  0.48s    aggr1  net1        net1        net0    t39   --      --
  0.48s    aggr1  net1        net1        net0    t39   0.34ms  0.38ms
  0.72s    aggr1  net2        net2        net1    t38   --      --
  0.72s    aggr1  net2        net2        net1    t38   0.38ms  0.42ms
  0.76s    aggr1  net0        net0        net2    t39   --      --
  0.76s    aggr1  net0        net0        net2    t39   0.33ms  0.38ms
  0.87s    aggr1  net1        net1        net2    t39   --      --
  0.87s    aggr1  net1        net1        net2    t39   0.32ms  0.38ms
  1.95s    aggr1  net2        net2        net0    t39   --      --
  1.95s    aggr1  net2        net2        net0    t39   0.36ms  0.42ms
  1.97s    aggr1  net2        net2        net1    t39   --      --
  1.97s    aggr1  net2        net2        net1    t39   0.32ms  0.38ms
  1.99s    aggr1  net0        net0        net1    t40   --      --
  1.99s    aggr1  net0        net0        net1    t40   0.31ms  0.36ms
  2.12s    aggr1  net1        net1        net0    t40   --      --
  2.12s    aggr1  net1        net1        net0    t40   0.34ms  0.40ms
  2.14s    aggr1  net0        net0        net2    t40   --      --
```

The aggregation `aggr0` with an IP interface configured over it is created. The VNIC `vnic1` is configured on top of the aggregation `aggr0`. Probe-based failure detection is configured without specifying either the source IP address or the target IP address of the probes. To enable probing, the target in the routing table is configured with an IP address, `10.10.10.2`, on the same subnet as the specified IP address, `10.10.10.1`. The ICMP and transitive probe statistics are monitored.

Comparing Trunk and DLMP Aggregation

This section presents a general comparison of the two types of link aggregation.

TABLE 2-1 Feature Comparison of Trunk and DLMP Aggregation

Feature	Trunk Aggregations	DLMP Aggregations
Link-based failure detection	Supported	Supported
LACP	Supported	Not supported
Use of standby interfaces	Not supported	Not supported[1]
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 of the aggregation's ports	Supported	Limited[2]
User-defined flows for resource management	Supported	Supported
Link protection	Supported	Supported
Back-to-back configuration	Supported	Not supported[3]

[1] Each DLMP client is associated with exactly one DLMP port. The remaining ports act as available ports for the DLMP clients, but you cannot configure these available ports.

[2] The aggregation spreads its VNICs across all ports. However, individual VNICs cannot spread the load on multiple ports.

[3] DLMP aggregations must always use an intermediary switch to send packets to other destination systems. However, no switch configuration is required for DLMP.

♦ ♦ ♦ CHAPTER 3

Configuring Virtual Networks by Using Virtual Local Area Networks

This chapter discusses the features and advantages of virtual local area networks (VLANs). This chapter also describes the procedures to configure and modify VLANs.

This chapter contains the following topics:

Overview of Deploying VLANs

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. You can assign interfaces on the same system to different VLANs.

When to Use VLANs

You can deploy VLANs if you need to do the following:

- Create a logical division of workgroups.

For example, if all hosts on a floor of a building are connected on one switch-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. You can create a separate VLAN for each department and enforce the appropriate security policy on a per-VLAN basis.
- Reduce the size of broadcast domain and improve network efficiency. You can split workgroups into manageable broadcast domains.

  For example, in a broadcast domain consisting of 25 users, if the broadcast traffic is intended only for 12 users, then setting up a separate VLAN for those 12 users can reduce traffic and improve network efficiency.

Assigning VLAN 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. However, now in Oracle Solaris, 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 “Configuring and Administering Network Components in Oracle Solaris 11.2 ”. For example, you can assign a custom VLAN name such as `sales0` or `marketing1`.

VLAN names work in conjunction with VLAN IDs. Each VLAN in a local area network is identified by a VLAN ID, which is a part of the 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.

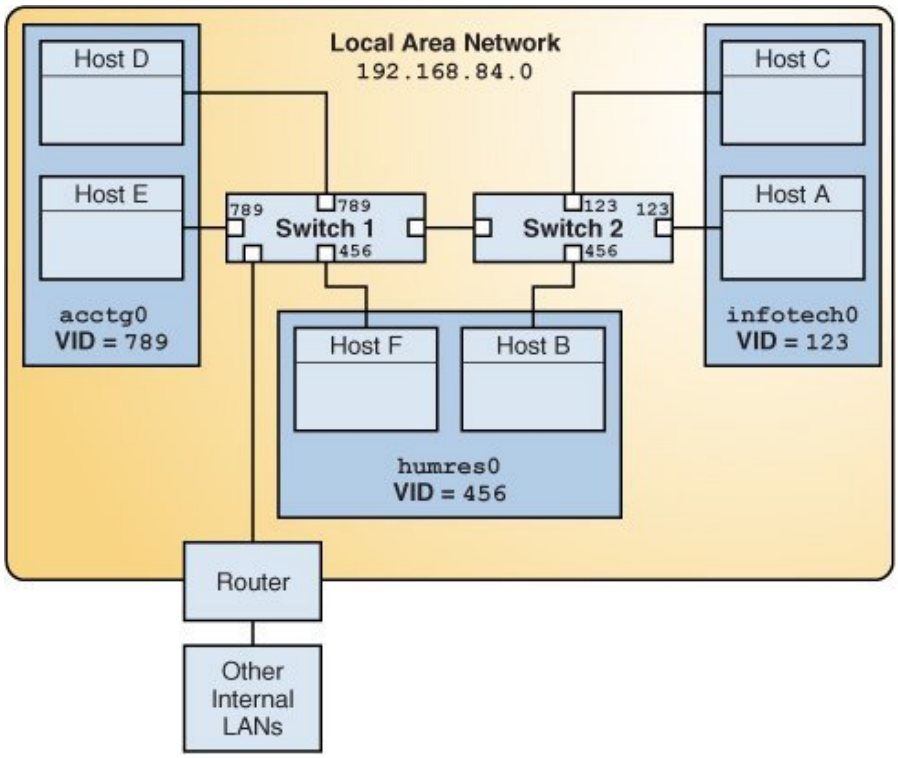
By default, each port has a VLAN ID called Port VLAN ID. The packets that belong to this VLAN ID are not tagged with a VLAN tag. In Oracle Solaris, you can use the datalink property `default_tag` to display and change Port VLAN ID on an interface.

VLAN Topology

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



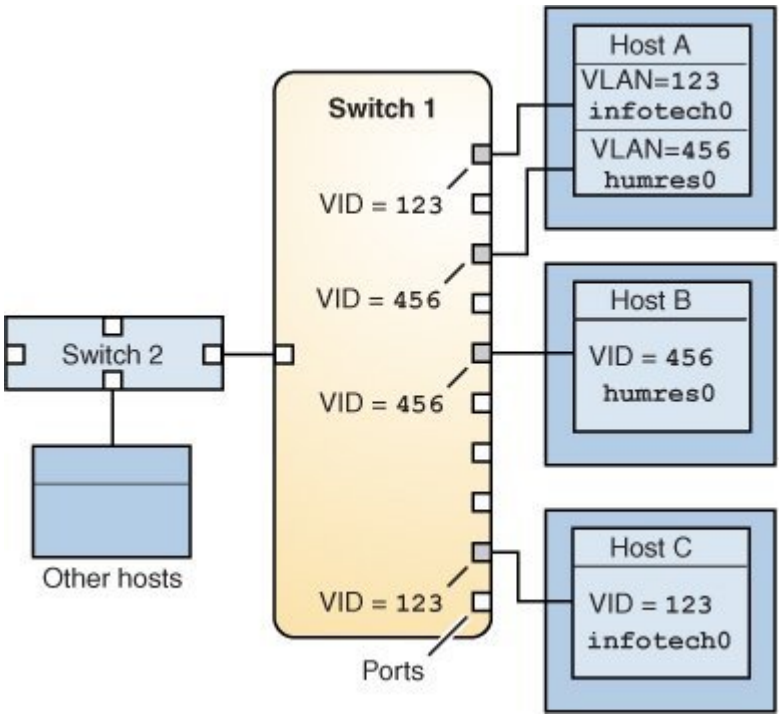
In the illustration, the LAN has the subnet address `192.168.84.0`.

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 this figure is shown in the following figure, where only one switch is used and multiple hosts belonging to different VLANs connect to that single switch.

FIGURE 3-2 A Switch Connecting Multiple Hosts of Different VLANs



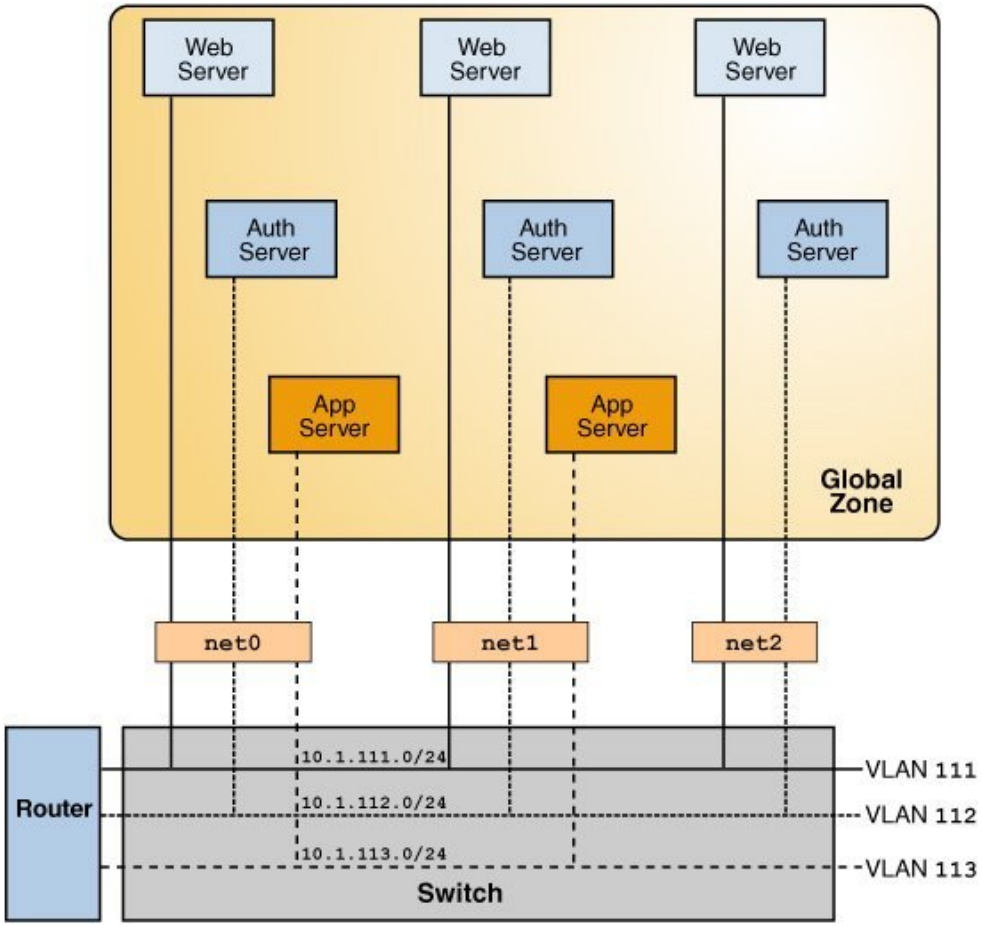
In the figure, Host A and Host C belong to the Information Technology VLAN with the VLAN ID `123`. One of Host A's interface is configured with the VLAN ID `123`. This interface connects to Port 1 on Switch 1, which is also configured with the VLAN ID `123`. Host B is a member of the Human Resources 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`.

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

Using VLANs With 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 figure, which illustrates 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 combine all eight systems and configure them as zones in a single

system. Then you can use VLAN IDs to assign a VLAN to each set of zones that perform the same functions. The following table lists the information provided in the figure.

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`
Application server	`appzone1`	`app1`	`113`	`10.1.113.0`	`net0`
Web server	`webzone2`	`web2`	`111`	`10.1.111.1`	`net1`
Authentication server	`authzone2`	`auth2`	`112`	`10.1.112.1`	`net1`
Application server	`appzone2`	`app2`	`113`	`10.1.113.1`	`net1`
Web server	`webzone3`	`web3`	`111`	`10.1.111.2`	`net2`
Authentication server	`authzone3`	`auth3`	`112`	`10.1.112.2`	`net2`

To create the configuration shown in the figure, refer to Example 3-2.

Planning a VLAN Configuration

Planning a VLAN configuration involves the following steps:

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 - If VLAN numbering scheme already exists on the network, you must create VLAN IDs within the existing VLAN numbering scheme.

---

3. On each system, determine which interfaces are the components of a particular VLAN.
   a. Determine which links are configured on the system by using the `dladm show-link` command.
   b. Determine which VLAN ID will be associated with each datalink on the system.
   c. Create the VLAN.
4. Check the connections of the datalinks to the switches of the network.

Note the VLAN ID of each datalink 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.

Configuring a VLAN

The following procedure describes how to create a VLAN over a datalink by using the `dladm` command. You can create an IP interface over a VLAN and configure the interface with an IP address by using the `ipadm` command. For information about the `dladm` and `ipadm` commands, see the `dladm`(1M) and `ipadm`(1M) man pages.

▼ How to Configure a VLAN

Before You Begin This procedure assumes that the zones are already created on the system. For information about zone configuration, refer to Chapter 1, "How to Plan and Configure Non-Global Zones," in "Creating and Using Oracle Solaris Zones ".

1. Become an administrator.

For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

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 a meaningful custom name. For information about VLAN names, see "Assigning VLAN Names" on page 42.

4. Verify the VLAN configuration.

```
# dladm show-vlan
```

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

   `# ipadm create-ip` *interface*

   where *interface* provides the VLAN name.

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

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

Example 3-1 Creating a VLAN

This example shows how to create the VLAN configuration that is illustrated in Figure 3-1.

1. Check the available links and create the VLANs over the specific links.

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

2. Host A:

```
# dladm create-vlan -l net0 -v 123 infotech0
```

   Host C:

```
# dladm create-vlan -l net0 -v 123 infotech0
```

   Host F:

```
# dladm create-vlan -l net0 -v 456 humres0
```

   Host B:

```
# dladm create-vlan -l net0 -v 456 humres0
```

   Host D:

```
# dladm create-vlan -l net0 -v 789 acctg0
```

   Host E:

```
# dladm create-vlan -l net0 -v 789 acctg0
```

3. Display the VLANs created.

```
# dladm show-vlan
LINK           VID     OVER        FLAGS
infotech0      123     net0        ----
infotech0      123     net0        ----
```

```
humres0         456     net0        ----
humres0         456     net0        ----
acctg0          789     net0        ----
acctg0          789     net0        ----
```

Example 3-2 Configuring a VLAN With Zones

This example shows how to create the VLAN configuration that is illustrated in Figure 3-3. This example assumes that you have already configured different zones in the system. For more information about configuring zones, see “Creating and Using Oracle Solaris Zones ”.

1. Check the available links that can be used for configuring VLANs and then create the VLANs over the specific links.

```
global# dladm show-link
LINK     CLASS     MTU     STATE     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     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
```

2. Assign the VLANs to their respective zones and display information for each zone similar to the following:

```
global# zonecfg -z webzone1 info net
net:
address not specified
physical: web1
net:
address not specified
physical: web2
net:
address not specified
physical: web3

global# zonecfg -z authzone1 info net
net:
address not specified
physical: auth1
net:
address not specified
physical: auth2
net:
address not specified
physical: auth3

global# zonecfg -z appzone2 info net
net:
address not specified
physical: app1
net:
address not specified
physical: app2
```

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

3. Display the assigned VLANs in the zones.

```
global# dladm show-vlan
```

```
LINK            VID  OVER  FLAGS
webzone1/web1   111  net0  --
authzone1/auth1 112  net0  --
appzone1/app1   113  net0  --
webzone1/web2   111  net1  --
authzone1/auth2 112  net1  --
appzone1/app2   113  net1  --
webzone1/web3   111  net2  --
authzone2/auth3 111  net2  --
```

4. Log 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 10.1.111.1/24 web2
   ipadm: web2/v4
   ```

   In webzone3:

   ```
   webzone3# ipadm create-ip web3
   webzone3# ipadm create-addr -a 10.1.111.2/24 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 10.1.112.1/24 auth2
   ipadm: auth2/v4
   ```

   In authzone3:

   ```
   authzone3# ipadm create-ip auth3
   authzone3# ipadm create-addr -a 10.1.112.2/24 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 10.1.113.1/24 app2
ipadm: app2/v4
```

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

Configuring VLANs Over a Link Aggregation

You can create VLANs on a link aggregation in a manner that is similar to configuring VLANs over an interface. Link aggregations are described in Chapter 2, “Configuring High Availability by Using Link Aggregations”. This section combines configuring VLANs and link aggregations.

▼ How to Configure VLANs Over a Link Aggregation

Before You Begin Create the link aggregation. For information about how to create link aggregations, refer to “How to Create a Link Aggregation” on page 26.

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

   ```
   # dladm show-aggr
   ```

2. **For every VLAN that you want to create over the link aggregation you selected, you can use the following command:**

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

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

   Note - In this procedure, the link refers to the link aggregation.

   vid — Indicates the VLAN ID number.

   VLAN-link — Specifies the name of the VLAN.

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 3-3 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     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
```

Configuring VLANs on a Legacy Device

Certain legacy devices handle only packets whose maximum transmission unit (MTU) size, also known as frame size, is a maximum of 1514 bytes. Packets whose frame sizes exceed this maximum limit are dropped. For such cases, follow the same procedure listed in "How to Configure a VLAN" on page 47. However, when creating the VLAN, use the `-f` option to force the creation of the VLAN.

▼ How to Configure VLANs on a Legacy Device

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

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

`-f`	Forces the creation of VLAN. Use this option when you are creating a VLAN on devices that do not allow frame sizes large enough to include a VLAN header.
`-l` *link*	Specifies the link on which the VLAN interface is created. In this procedure, the link refers to the legacy device.
`-v` *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`).**

   In the following example, `mtu` is set to `1496`.

   ```
   # dladm set-linkprop -p mtu=1496 VLAN-link
   ```

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

3. **Repeat Step 2 to set a MTU value for each node in the VLAN.**

   For more information about changing link property values, refer to “Administering Datalink Properties” in “Configuring and Administering Network Components in Oracle Solaris 11.2 ”.

Displaying VLAN Information

You can use the `dladm show-link` command to display information about VLANs because VLANs are datalinks. Use the `dladm show-vlan` command to display specific information about the VLANs.

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

```
# dladm show-link
LINK     CLASS    MTU      STATE     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

You can modify a VLAN by using the `dladm modify-vlan` command in the following ways:

- Change the VLAN ID of a VLAN
- Migrate a VLAN to another underlying link

Modifying the VLAN ID of a VLAN

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

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

  where *vid* specifies the new VLAN ID that you are assigning to the VLAN and *datalink* refers to the underlying link over which the VLAN is configured.

Note - You can use the `dladm modify-vlan -v` *vid* `-L` *datalink* command syntax only if a single VLAN exists on the datalink. The command fails if you use it on a datalink that has multiple configured VLANs because each VLAN on a datalink must have unique VLAN IDs.

If you modify the VLAN ID on the link, you must also configure the switch port for the new VLAN ID.

- `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 so you must change the

VLAN IDs one at a time. In the setup shown in Figure 3-3, you would change the VLAN IDs of `web1`, `auth1`, and `app1` that are configured over `net0` as follows:

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

Migrating a VLAN to Another Underlying Link

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 a Virtual Network" in "Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ".

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 the 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. Therefore, 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, and MTU size. 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

Global migration is used to migrate all of the VLANs that are configured over one datalink to another datalink. To perform a global migration, you need to specify only the source and target datalinks. The following example shows how to move all of the VLANs on `ether0` to `net1`.

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

`-l` Refers to the target datalink to which the VLANs are migrated.

`-L` Refers to the original datalink over which the VLANs are configured.

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

Selective Migration

Selective migration is used to migrate only selected VLANs. To perform selective VLAN migration, you specify the VLANs that you want to move. In the following example, which is 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, do not include 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 command, `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` command. Use the `modify-vnic` subcommand on VNICs, including those with VLAN IDs, that are displayed in the output of the `dladm show-vnic` command. For information about how to modify VNICs, see "Modifying the VLAN IDs of VNICs" in "Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ".

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 fails if IP interfaces exist over the VLAN.

EXAMPLE 3-4 Deleting a VLAN Configuration

This example shows how to delete a VLAN configuration.

```
# 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.

In the following example, a system that uses four NICs must be configured to be a router for eight separate subnets. Therefore, eight links are configured, one for each subnet. First, a trunk aggregation is created on all four NICs. This untagged link that does not include a VLAN tag in the outgoing frame becomes the default untagged subnet for the network to which the default route points.

VLAN interfaces are then 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.

1. Remove 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
```

2. Create the trunk aggregation `default0`.

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

# dladm show-link
LINK        CLASS     MTU   STATE    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
```

3. Configure an IP interface over the aggregation.

```
# ipadm create-ip default0
# ipadm create-addr -a 10.2.3.4/24 default0
```

4. Create 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    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    -----
```

5. Create IP interfaces over the VLAN links and assign 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 10.2.3.5/24 orange0
# ipadm create-addr -a 10.2.3.6/24 green0
# ipadm create-addr -a 10.2.3.7/24 blue0
# ipadm create-addr -a 10.2.3.8/24 white0
# ipadm create-addr -a 10.2.3.9/24 yellow0
# ipadm create-addr -a 10.2.3.10/24 red0
# ipadm create-addr -a 10.2.3.11/24 cyan0
```

♦ ♦ ♦ CHAPTER 4

Administering Bridging Features

You can use bridging is used to connect separate network segments so that they communicate as if they were a single network segment. This chapter describes how to configure and administer bridged networks.

This chapter contains the following topics:

Overview of Bridged Networks

Bridges connect various nodes in the network into a single network. The network segments share a single broadcast network and communicate as if they were a single network segment when connected. As a result, each node can reach the other nodes by using network protocols such as IP rather than 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.

Although you can use both bridging and routing 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.

Bridging is used to distribute information about the locations of resources on the network. In a bridged network, the destinations of forwarded packets are determined by examining the

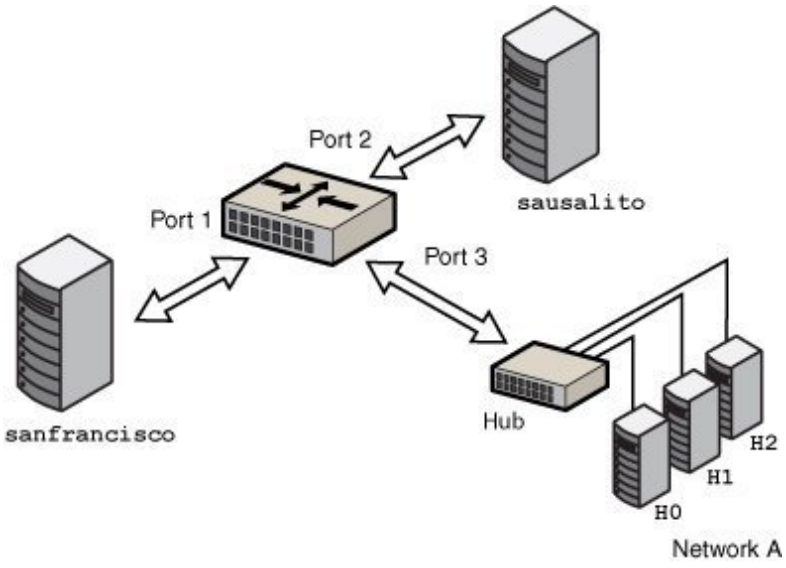
network traffic that is received on the links that are attached to the bridge. A bridged network uses protocols, such as Spanning Tree Protocol (STP) and Transparent Interconnection of Lots of Links (TRILL). For more information, see “Bridging Protocols” on page 65.

Caution - Do not set the `local-mac-address?` property to `false` by using the `eeprom` command on SPARC® based systems that use bridging. Doing so causes these systems to incorrectly use the same MAC address on multiple ports and on the same network.

Simple Bridged Network

The following figure shows a simple bridged network configuration.

FIGURE 4-1 Simple Bridged Network

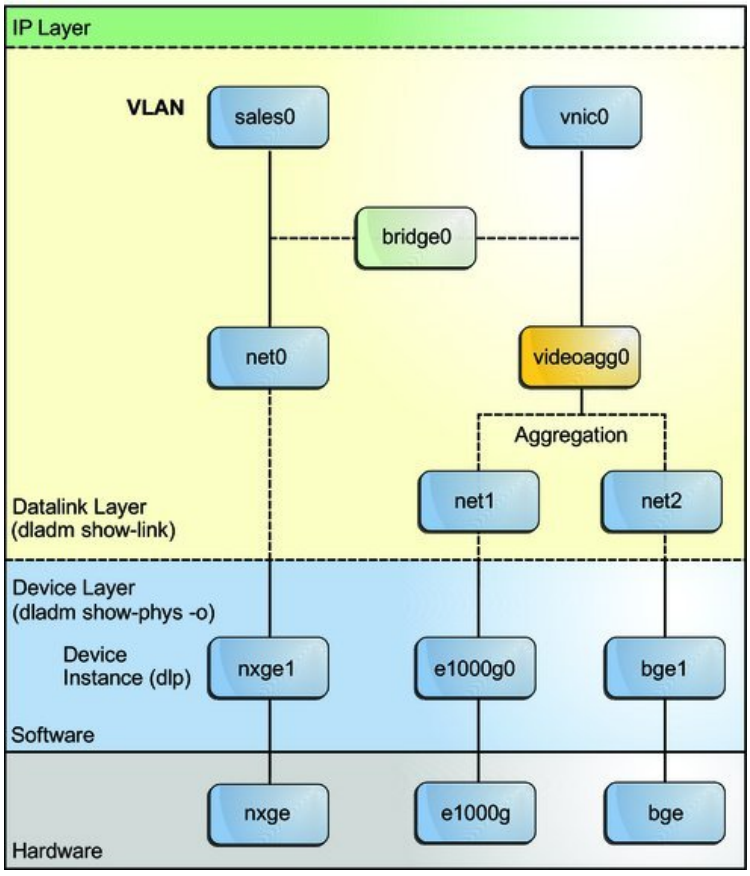


The bridge, `goldengate`, is an Oracle Solaris system that has bridging configured. The systems `sanfrancisco` and `sausalito` are physically connected to the bridge. Network A uses a hub that is physically connected to the bridge on one side and to three computer systems on the other side. The bridge ports are the links `net0`, `net1`, and `net2`.

How Oracle Solaris Bridges Are Implemented in the Network Stack

In Oracle Solaris, you can configure bridges on the datalink layer of the same network stack implementation, as shown in the following figure.

FIGURE 4-2 Bridges in the Network Stack for Oracle Solaris

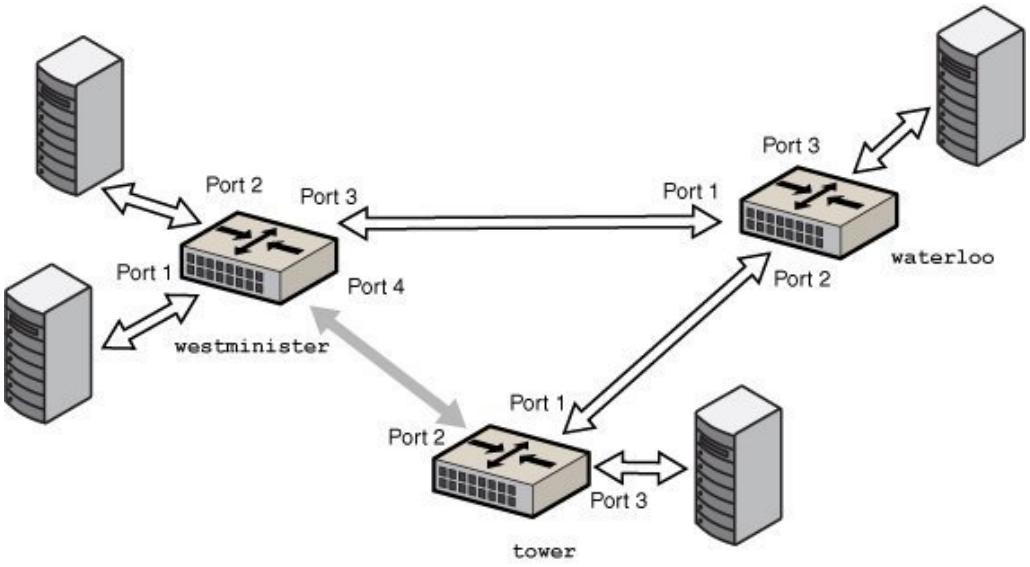


Two interfaces, `net0` and `videoagg0`, are configured as a bridge, `bridge0`. Packets that are received on one interface are forwarded to the other interface. After the bridge configuration, both interfaces can still be used to configure VLANs and IP interfaces.

Bridged Network Ring

Bridged networks can be formed into rings that physically connect several bridges together. The following figure shows a bridged network ring configuration.

FIGURE 4-3 Bridged Network Ring



The figure shows a bridged network that is configured in a ring. The configuration shows three bridges. Two systems are physically connected to the `westminister` 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.

This type of configuration can 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.

How a Bridged Network Works

When a packet is received by the bridge, its source address is examined. The source address of the packet associates the node from which the packet was sent with 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 that is 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. Therefore, destination address of the packet 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 begins again, based on the source address of a packet. This process enables the bridge to properly forward packets over that link when the address is used as the destination address.

Bridging Protocols

Bridged networks use the following protocols:

- Spanning Tree Protocol (STP)

  STP is the default protocol that is used by the bridged networks. Bridging uses the STP mechanism to prevent network loops that potentially render the subnetworks unusable. 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 infinitely circle at full-line rate.

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

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

- Transparent Interconnection of Lots of Links (TRILL)

  Oracle Solaris supports the TRILL protection enhancement, which avoids loops without disabling links. TRILL helps to load-balance the traffic between several paths to the destination.

When STP is used for loop protection, the physical loop is mitigated by preventing one of the connections in the loop from forwarding packets. Figure 4-3 shows that the physical link between the `westminster` and `tower` bridges is not used to forward packets.

Unlike STP, 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.

You can use TRILL by specifying the `-P trill` option in the `dladm create-bridge` or `dladm modify-bridge` commands. For more information, see "Creating a Bridge" on page 67 and "Modifying the Protection Type for a Bridge" on page 69.

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

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`. The observability node is intended for use with the `snoop` command and the `wireshark` packet analyzer. You can use the `dlstat` command to obtain the run time statistics of the bridge.

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 time-to-live (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 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 add it again.

If you disable the SMF service instance for a bridge, the bridge 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 by 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`.

Creating a Bridge

In Oracle Solaris, 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, and to assign links to bridges or to remove links from them. The links that are assigned to the bridge must be an Ethernet type, which includes 802.3 and 802.11 media.

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 a maximum of one bridge.

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, use the following command:

```
# dladm create-bridge [–P protect] [–p priority] [–d forward-delay] [–l link...] bridge-name
```

`-P` *protect* — Specifies the protection method. It can be set to one of the following values.

- `stp` – STP protection method (the default)
- `trill` – TRILL protection method

`-p` *priority* — Specifies an IEEE STP priority value for a bridge to determine the root bridge node in the network. The default value is `32768`. Valid values are from `0` (highest priority) to `61440` (lowest priority), in increments of 4096.

`-d` *forward-delay* — Specifies the STP forward delay parameter for the bridge. When the bridge that is created is the root node, all the bridges in the network use this timer to sequence the link states when a port is enabled. The default value is `15` seconds. Valid values are from `4` to `30` seconds.

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

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` and all names beginning with the `SUNW` string are reserved. 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.

For more information about bridge creation options, see the description of the `dladm create-bridge` command in the `dladm`(1M) man page.

EXAMPLE 4-1 Creating a Bridge

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

```
# dladm create-bridge -P stp -d 12 -l net0 -l net1 brooklyn
# dladm show-bridge
BRIDGE       PROTECT ADDRESS           PRIORITY DESROOT
goldengate   stp     32768/8:0:20:bf:f 32768    8192/0:d0:0:76:14:38
brooklyn     stp     32768/8:0:20:e5:8 32768    8192/0:d0:0:76:14:38
```

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

```
# dladm create-bridge -P trill -l net0 -l net1 westminister
# dladm show-bridge
BRIDGE       PROTECT ADDRESS           PRIORITY DESROOT
goldengate   stp     32768/8:0:20:bf:f 32768    8192/0:d0:0:76:14:38
westminister trill   32768/8:0:20:e5:8 32768    8192/0:d0:0:76:14:38
```

Modifying the Protection Type for a Bridge

STP is a mechanism to prevent network loops that could render the subnetworks unusable. In addition to using STP for bridges, Oracle Solaris supports the TRILL protection enhancement. STP is used by default, but you can use TRILL by specifying the `-P trill` option for the bridging commands.

To modify the protection type from STP to TRILL or from TRILL to STP, use the following command:

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

The `-P` *protection-type* option specifies which protection type to use, either `stp` (the default) or `trill`.

EXAMPLE 4-2 Modifying the Protection Type for a Bridge

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
```

The following example shows how to change the protection type for the `brooklyn` bridge from TRILL to STP.

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

Adding Links to an Existing Bridge

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.

Links that are assigned to the same bridge must have the same MTU value. Although you can change the MTU value on an existing link, 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.

Note - 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 can be assigned to a bridge.

To add a new link to an existing bridge, use the following command:

```
# 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
```

Removing Links From a Bridge

Before you delete any bridge, all of its links must first be removed. To remove links, use the following command:

```
# 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
```

Setting Link Properties for a Bridge

You can set the following link properties for bridges:

`default_tag`	The default VLAN ID for untagged packets that are sent to a link and received from a link. The valid values are `0` to `4094`. The default value is `1`.
`forward`	Enables and disables traffic forwarding through the bridge. This property exists on all links except VNIC links. The valid values are `1` (true) and `0` (false). The default value is `1`.
`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.

`stp_cost`	Represents the 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.
`stp_edge`	Specifies whether the port is connected to other bridges. Valid values are `1` (true) and `0` (false). The default value is `1`.
`stp_p2p`	Specifies the connection mode type. Valid values are `true`, `false`, and `auto`. The default value is `auto`.
`stp_priority`	Sets the STP and RSTP port priority value. Valid values are from `0` to `255`. The default value is `128`.

For more information, see the `dladm`(1M) man page.

To modify the link properties of a bridge, use the following command:

```
dladm set-linkprop -p prop=value link
```

EXAMPLE 4-3 Setting Link Properties for a Bridge

The following example shows how to disable traffic forwarding and set the connection mode type. To set the properties for the bridge, you must set the properties on the links that connect the bridge.

```
# dladm create-bridge -P stp -d 12 -l net0 -l net1 brooklyn
# dladm set-linkprop -p forward=0 net0
# dladm set-linkprop -p stp_p2p=true net1
```

The following example shows how to reset multiple properties of a bridge.

```
# dladm reset-linkprop -p default_tag,stp_priority brooklyn
```

Displaying Bridge Configuration Information

You can display bridge configuration information by using the `dladm show-bridge` command.

Displaying Information About Configured Bridges

You can use the `dladm show-bridge` and `dlstat show-bridge` commands to show different kinds of information about configured bridges.

Use the following command options:

- To view the list of bridges:

```
# dladm show-bridge
```

```
# 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
```

- To show the link-related status of a bridge:

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

- To show link-related statistics for the bridge:

```
# dlstat show-bridge bridge-name
```

- To show kernel forwarding entries for the bridge:

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

- To show TRILL information about the bridge:

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

- To show statistics of each bridge and statistics of the links connected to each bridge:

```
# dlstat show-bridge
```

```
BRIDGE     LINK       IPKTS    RBYTES     OPKTS    OBYTES     DROPS  FORWARDS
rbblue0     --        1.93K   587.29K     2.47K     3.30M         0         0
           simblue1     72     4.32K     2.12K     2.83M         0        --
           simblue2  1.86K   582.97K       348   474.04K         0        --
stbred0     --         975   976.69K     3.44K     1.13M         0        38
           simred3    347   472.54K     1.86K   583.03K         0        --
           simred4    628   504.15K     1.58K   551.51K         0        --
```

- To show all statistics of each bridge and statistics of the links connected to each bridge:

```
# dlstat show-bridge -o all
```

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

EXAMPLE 4-4 Displaying Bridge Information

The following examples show how to use the `dladm show-bridge` command with various options.

- 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 the 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 the 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     --          --
```

Displaying Configuration Information About Bridge Links

You use the `dladm show-link` command with the `-o all` option to display the `BRIDGE` field in the output. If a link is a member of a bridge, this field identifies the name of the bridge of which it is a member. For links that are not part of a bridge, the field is blank if the `-p` option is used. Otherwise, the field shows `--`.

The observability node of the bridge 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.

Use the following command to 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.

Deleting a Bridge From the System

Before you delete a bridge, you must first remove any links attached to the bridge.

▼ How to Delete a Bridge From the System

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Remove any links attached to the bridge.**

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

3. **Delete the bridge from the system.**

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

Example 4-5 Deleting a Bridge From the System

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
```

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 a part of a bridge, the command also enables packet forwarding of the specified VLAN on that bridge link. For more information about VLANs, see Chapter 3, "Configuring Virtual Networks by Using Virtual Local Area Networks".

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. For more information, see "Setting Link Properties for a Bridge" on page 70.

▼ How to Configure VLANs Over a Datalink That Is Part of a Bridge

Before You Begin This procedure assumes that the bridge already exists. For information about how to create a bridge, see "Creating a Bridge" on page 67.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **List the link-related information of the bridge to determine the links that are a part of the bridge.**

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

3. **Create the VLAN over the link that is a part of the bridge.**

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

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

   Note - In this procedure, the link should part of the bridge that you have created.

   vid Indicates the VLAN ID number.

   VLAN-link Specifies the name of the VLAN.

4. **Repeat this command for every VLAN you want to create. For every VLAN that you created, create an IP interface over the VLAN.**

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

   where *interface* is the VLAN name.

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

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

VLANs and the STP and TRILL Protocols

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. 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.

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`, for example, `/dev/net/bridgeofsighs0`.

The observability node is intended for use with the `snoop` command and the `wireshark` packet analyzer. 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, which enables you only to receive packets and not to send them.

The observability node makes a single unmodified copy of every packet handled by the bridge. It is available to the user for monitoring and debugging. This behavior is similar to monitoring a port on a traditional bridge and is subject to the usual datalink provider interface (DLPI) promiscuous mode rules. You can also use the `pfmod` command or features in the `snoop` command and the `wireshark` packet analyzer to filter packets based on the VLAN ID.

The delivered packets, which are the packets that are sent to the observability node, represent the data received by the bridge.

Note - When the bridging process adds, removes, or modifies a VLAN tag, the data shown by the `snoop` command and the `wireshark` packet analyzer describes the state prior to the processes 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 the `snoop` command on the individual links rather than on the bridge observability node.

You can also obtain statistics about how network packets use network resources on links by using the `dlstat` command. For information, see Chapter 8, “Monitoring Network Traffic and Resource Usage,” in “Managing Network Virtualization and Network Resources in Oracle Solaris 11.2 ”.

♦ ♦ ♦ CHAPTER 5

Exchanging Network Connectivity Information With Link Layer Discovery Protocol

This chapter describes how to enable systems to exchange system and network connectivity information throughout the local network by using the Link Layer Discovery Protocol (LLDP).

This chapter contains the following topics:

Overview of LLDP

LLDP is used by systems in a local area network (LAN) to exchange configuration and management information with each other. 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 systems to receive similar information about other systems that are on the same local network.

On any LAN, 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.

When you manually configure each system, switch, and other components, ensuring compatibility among the components is a challenge. The manual configuration of systems is risky and can easily cause misconfigurations, particularly if different administrators work independently on different systems. A better alternative is to use LLDP, which enables systems to transmit their individual configuration information to peer systems and helps to detect any misconfigurations.

Oracle Solaris supports the use of LLDP to promote the exchange of system and network connectivity information between systems on the network, which reduces the risk of misconfigured network resources.

In this release, LLDP is used by the network diagnostics service to automatically detect problems that could lead to limited or degraded network connectivity, or both. Enabling the LLDP service enhances the ability to perform network diagnostics on your Oracle Solaris system. For more information about network diagnostics, see Chapter 4, “Performing Network Diagnostics With the network-monitor Transport Module Utility,” in “Troubleshooting Network Administration Issues in Oracle Solaris 11.2 ”.

In Oracle Solaris, LLDP is also used to exchange data center bridging exchange protocol (DCBX) Type-Length-Value (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 6, “Managing Converged Networks by Using Data Center Bridging”.

With LLDP, the system administrator can easily detect faulty system configurations, particularly in complex networks such as virtual local area networks (VLANs) and link aggregations. Information about the network topology can be obtained readily without having 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:

- **LLDP package** – You install this package to enable the LLDP. This package includes the LLDP daemon, command-line utilities, the service manifest and scripts, and other components that are required for LLDP to operate.
- **LLDP service** – You can enable the LLDP service by using the `svcadm` command. This service uses the fault management resource identifier (FMRI) of the service management facility (SMF) service instance, `svc:/network/lldp:default`, to manage the LLDP daemon, `lldpd`. This LLDP service is responsible for starting, stopping, restarting, or refreshing the `lldpd` daemon. This service is automatically enabled after you install the LLDP package.
- `lldpadm` **command** – You can use this command to administer LLDP on individual links and to configure the operating mode of LLDP, to specify TLV units that are transmitted, and to configure DCBX TLV units. For information about TLV units, see “Information the LLDP Agent Advertises” on page 80.

  You must use this command to set the per-agent LLDP properties and global LLDP properties and to get LLDP information for a specific agent or its peer.

  The `lldpadm` subcommands are described in the following sections. For more information about the `lldpadm` command, see the `lldpadm`(1M) man page.

- **LLDP daemon** – The LLDP services manage LLDP agents on the system. They also interact with `snmpd`, the daemon for the Simple Network Management Protocol (SNMP), to retrieve LLDP information that is received on the system through SNMP.
- **LLDP agents** – LLDP agents are the LLDP instances that are associated with a physical datalink on which LLDP is enabled. LLDP agents transmit information about the datalink to its peer and also receive information from the peer. You can configure an LLDP agent to advertise specific information about the associated physical datalink. You can enable LLDP only on physical datalinks.

Information Sources of the LLDP Agent

The LLDP agent transmits and receives LLDP data units (LLDPDUs). The agent manages and stores information that is contained in these LLDPDUs in the following types of data stores:

- **Local management information base (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. Therefore, 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

The LLDP agent operates in the following modes:

- Transmit only (`txonly`) – The LLDP agent does not process incoming LLDPDUs. Therefore, the remote MIB is empty.
- Receive only (`rxonly`) – 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`) – The agent transmits local information and processes incoming LLDPDUs and therefore maintains both local and remote MIBs.
- Disabled (`disable`) – The agent does not exist.

For information about setting agent modes, see “How to Enable LLDP for Specific Ports” on page 85.

Information the LLDP Agent Advertises

The LLDP agent transmits system and connectivity information in the LLDP packets or LLDPDUs. These packets contain information units that are individually formatted in TLV format. The information units are also called *TLV units*.

Mandatory TLV Units

Certain TLV units are mandatory and are included in the LLDP packets by default when LLDP is enabled. You cannot use the `lldpadm` command to exclude any of these units.

The following TLV units are mandatory:

- Chassis ID – Information that is generated by the `hostid` command
- Port ID – MAC address of the physical NIC
- TTL (time to live)
- End of protocol data unit (PDU)

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.

EXAMPLE 5-1 Displaying the Chassis ID and Port ID

The following example displays the chassis ID and port ID for an LLDP agent.

```
# hostid
004e434e

# dladm show-phys -m net4
LINK                SLOT     ADDRESS            INUSE CLIENT
net4                primary  0:1b:21:87:8b:b4   yes   net4

# lldpadm show-agent -l net4
AGENT               CHASSISID           PORTID
net4                004e434e            00:1b:21:87:8b:b4
```

Oracle Solaris LLDP agents use `hostid` as the chassis ID and the port's MAC address as the port ID.

Optional TLV Units

Optional TLV units can be added to an LLDP packet. These optional TLV units enable vendors to insert vendor-specific TLV units to be advertised. LLDP enables defining additional TLV

units by using organization unique identifiers (OUIs). An OUI identifies the category for a TLV unit depending on whether the OUI follows the IEEE 802.1 or IEEE 802.3 standard. The LLDP agent properties can be configured to enable or disable the transmission of these optional TLV units.

The following table lists each TLV group, its corresponding name, and 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 5-1 Optional TLV Units for an LLDP Agent

TLV Group	TLV 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`, `etsreco`	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 Unit Properties

Each TLV unit has properties that you can further configure with specific values. If 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 unit `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*. For information about how to configure TLV units, see “Specifying TLV Units and Values for the LLDP Packet of an Agent” on page 87.

Global TLV units apply to all LLDP agents on the system. The following table lists the global TLV units and their corresponding possible configurations.

TABLE 5-2 Global TLV Units and Their Properties

TLV Unit	Property Name	Possible Property Values	Value Description
`syscapab`	`supported`	`other`, `repeater`, `bridge`, `wlan-ap`, `router`, `telephone`, `docsis-cd`, `station`, `cvlan`, `sylvan`, `tpmr`	Represents 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 are associated with the local LLDP agent. The addresses are used to reach higher layer entities and assist in discovery by network management. Only one type can be specified.

TLV units that are specific to LLDP agent are managed on a per-agent basis. 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 lists the TLV values and their corresponding possible configurations for an LLDP agent.

TABLE 5-3 Per-Agent TLV Units and Their Properties

TLV Unit	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` For more information, see "Application Priority Configurations" on page 106.
`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 information about per-agent TLV units, see Chapter 6, “Managing Converged Networks by Using Data Center Bridging”.

Enabling LLDP on the System

You can configure LLDP to exchange system information with other hosts or peers on the network.

The SMF property `auto-enable-agents` controls the way in which you can enable LLDP agents 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 SMF property `auto-enable-agents` can have one of the following three possible values:

- `yes` enables LLDP on all ports in the transmit and receive (`both`) mode 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 the `rxonly` mode, then the LLDP service will not switch the agent to run in the transmit and receive (`both`) mode. LLDP on that port continues to be in the `rxonly` mode. This is the default value of the SMF property `auto-enable-agents`.
- `force` enables LLDP in the transmit and receive (`both`) mode on all ports and overrides any existing LLDP configurations on any port. For example, if a previous LLDP configuration on a port runs in the `rxonly` mode, the LLDP agent is switched to run in the transmit and receive (`both`) mode, which is the default LLDP mode.
- `no` disables automatic enabling of LLDP on all ports except those with existing LLDP configurations. On these ports, the existing LLDP configuration is retained.

Note - Every time you customize the `auto-enable-agents` property, you must restart the LLDP service for the new value to become effective.

▼ How to Install the LLDP Package

By default, LLDP is enabled and ready to be used after you have completed installing the LLDP package.

1. **Become an administrator.**

   For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

2. **Install the package.**

```
# pkg install lldp
```

3. **Determine whether 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
```

▼ How to Enable LLDP Globally

Before You Begin In order to enable LLDP, you must first install the LLDP package. For more information, see "How to Install the LLDP Package" on page 83.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Change the SMF `auto-enable-agents` property to `yes` if it is set to `no`.**

```
# svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "yes"
```

   By default, this property is set to `yes`.

3. **Restart the LLDP service.**

```
# svcadm restart svc:/network/lldp:default
```

4. **(Optional) Customize global TLV units.**

```
# lldpadm set-tlvprop -p property=value  global-TLV
```

   where *property* refers to the property of the global TLV unit.

Next Steps For an explanation of global TLV units, see "TLV Unit Properties" on page 81.

To display a list of global TLVs, type `lldpadm show-tlvprop` or refer to Table 5-2.

For instructions about how to define TLV values, see "How to Define TLV Units" on page 89.

For information about the `lldpadm` command, see the `lldpadm`(1M) man page.

How to Enable LLDP for Specific Ports

Before You Begin In order to enable LLDP, you must first install the LLDP package. For more information, see "How to Install the LLDP Package" on page 83.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Change the SMF `auto-enable-agents` property to `no` if it is set to `yes`.**

   ```
   # svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
   ```

   By default, this property is set to `yes`.

3. **Restart the LLDP service if you have changed the SMF property `auto-enable-agents` in step 2.**

   ```
   # svcadm restart svc:/network/lldp:default
   ```

4. **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. For example, 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: `txonly`, `rxonly`, `both`, and `disable`. For an explanation of these values, see "LLDP Agent Modes" on page 79.

5. **Specify the TLV units that the LLDP agent can 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 80.

   To display a list of the other properties of the LLDP agent, type `lldpadm show-agentprop` or refer to Table 5-1.

   For instructions about how to specify TLV units for LLDP packet of an agent, see "How to Specify TLV Units for the LLDP Packet of an Agent" on page 87.

6. **(Optional) Customize the 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 Unit Properties" on page 81.

To display a list of per-agent TLVs, type `lldpadm show-agenttlvprop` or refer to Table 5-3.

For instructions about how to define TLV values, see "How to Define TLV Units" on page 89.

For information about the `lldpadm` command, see the `lldpadm`(1M) man page.

Example 5-2 Customizing the `auto-enable-agents` SMF Property

The following example shows the different way in which LLDP is enabled if you change the value of the SMF property `auto-enable-agents`. For example, given a system with four ports, LLDP is configured on two ports as follows:

- `net0`: `both` mode
- `net1`: `rxonly` mode
- `net2` and `net3`: none

If the SMF property `auto-enable-agents` has the default value `yes`, LLDP is automatically enabled on `net2` and `net3`. You can display the LLDP configuration 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 5-3 Enabling LLDP on Multiple Datalinks

This example shows how to enable LLDP selectively. A system has two datalinks, `net0` and `net1`. On `net0`, to set the agent to transmit and receive LLDP packets, and on `net1`, to set the agent to only transmit LLDP packets, 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
# lldpadm show-agentprop -p mode
AGENT   PROPERTY   PERM   VALUE    DEFAULT   POSSIBLE
net0    mode       rw     both     disable   txonly,rxonly,both,disable
net1    mode       rw     txonly   disable   txonly,rxonly,both,disable
```

Specifying TLV Units and Values for the LLDP Packet of an Agent

You can specify TLV units such as `dot1-tlv` and `basic-tlv` as property values for an LLDP agent. You can further configure these property values. To specify TLV units, use the `lldpadm set-agentprop` command. For more information, see `lldpadm`(1M) man page. When the TLV unit is specified as a property of an LLDP agent, then that TLV unit is advertised in the network only with the values that you have specified for the TLV units. For information about TLV units and properties see, "TLV Unit Properties" on page 81.

▼ How to Specify TLV Units for the LLDP Packet of an Agent

This procedure explains how to specify TLV units to be advertised in an LLDP packet that an agent transmits.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **(Optional) Identify the LLDP agent property that can contain the TLV unit that you want to add by displaying the TLV units.**

   ```
   # lldpadm show-agentprop agent
   ```

   This command helps you to see the TLV units that are already set for each property. If you do not specify a property, this command displays all the LLDP agent properties and their TLV values. For a list of agent properties, see Table 5-1.

3. **Add or remove the TLV unit from the property.**

   ```
   # lldpadm set-agentprop -p property[+|-]=value[,...] agent
   ```

You can use qualifiers to add (+) or remove (-) values from the list of values for properties that accept multiple values.

If you do not use the add (+) or remove (-) qualifiers, then the value that you set replaces all the values that were previously defined for the property.

4. (Optional) Display the new values for the property.

```
# lldpadm show-agentprop -p property agent
```

Example 5-4 Adding Optional TLV Units to an LLDP Packet

In the following example, the LLDP agent has `net0` configured to advertise VLAN information in its LLDP packet. The LLDP packet is further configured to include system capabilities, link aggregation, and virtual NIC information as items that the LLDP can advertise. Later, the VLAN description is removed from the packet.

1. Display the existing agent properties.

```
# 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,etsreco,all
net0    dot3-tlv   rw     max-framesize   none     none, max-framesize,
                                                   all
net0    virt-tlv   rw     none            none     none,vnic,all
```

   The output displays the existing, default, and possible values for each property of the LLDP agent.

2. Set system capabilities, link aggregation, and network virtualization information as items to advertise over the network.

```
# lldpadm set-agentprop -p basic-tlv+=syscapab,dot1-tlv+=linkaggr,virt-tlv=vnic net0
```

3. Remove the VLAN description from the packet.

```
# lldpadm set-agentprop -p dot1-tlv-=vlanname net0
```

4. Display the agent properties.

```
# 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,pfc        none   none,vlanname,pvid,
                           linkaggr               linkaggr,pfc,appln,
                                                  evb,etscfg,etsreco,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 Units

This procedure explains how to provide values for specific TLV units.

Tip - You can reset the TLV properties to their default values by using the `lldpadm reset-tlvprop` command for global TLV units and `lldpadm reset-agenttlvprop` command for per-agent TLV units.

---

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Configure a global or a per-agent TLV 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 a list of global TLV units and their properties, see Table 5-2.

   - **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 a list of per-agent TLV units and their properties, see Table 5-3.

You can use qualifiers to add (+) or remove (-) values from the list of values for properties that accept multiple values.

3. **(Optional) Display the values of the TLV property that you have configured.**

   - **To display the global TLV property values:**

     ```
     # lldpadm show-tlvprop
     ```

   - **To display the values of the TLV property of an agent:**

     ```
     # lldpadm show-agenttlvprop
     ```

Example 5-5 Defining TLV Values for the `syscapab` and `mgmtaddr` TLV Units

In the following example, specific information about the capabilities of the system to be advertised in the LLDP packet and management IP address is configured.

1. Configure both `supported` and `enabled` properties of the `syscapab` TLV unit.

   ```
   # lldpadm set-tlvprop -p supported=bridge,router,repeater syscapab
   # lldpadm set-tlvprop -p enabled=router syscapab
   ```

2. Specify the management IP address for the `mgmtaddr` TLV unit.

   ```
   # lldpadm set-tlvprop -p ipaddr=192.168.1.2 mgmtaddr
   ```

3. Display the TLV values of the agent properties.

   ```
   # lldpadm 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            --
   ```

   The output includes the default values of the TLV units and also the possible values that can be set for the property.

For information about configuring per-agent TLV properties, see Chapter 6, “Managing Converged Networks by Using Data Center Bridging”.

Disabling LLDP

This section describes how to disable LLDP selectively on individual ports.

How to Disable LLDP

To disable LLDP across all of the system's interfaces, perform the following steps.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

2. **Change the SMF LLDP property to `no`, which disables automatic enabling of LLDP on all ports except those with existing LLDP configurations.**

   ```
   # svccfg -s svc:/network/lldp:default setprop lldp/auto-enable-agents = "no"
   ```

3. **Restart the LLDP service.**

   ```
   # svcadm restart svc:/network/lldp:default
   ```

4. **Disable LLDP on each port whose previous LLDP configuration is retained.**

   - **To disable LLDP by changing the mode of the agent:**

     ```
     # 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. For example, if you enable LLDP on `net0`, the agent is `net0`.

   - **To disable LLDP by removing the LLDP configuration from the port:**

     ```
     # lldpadm reset-agentprop -p mode agent
     ```

     In this command, you do not set a value for the `mode` property.

Caution - 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.

Monitoring LLDP Agents

The `lldpadm show-agent` command displays 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 information that is received by the system from other systems on the same network.

Displaying the Advertised Information

The information can be either local or remote. *Local* information comes from the local LLDP agent. *Remote* information comes from other LLDP agents on the network that is received by the local LLDP agent.

Use the `lldpadm show-agent` command to display the advertised information.

```
# lldpadm show-agent -[l|r][v] agent
```

- `-l` displays local information advertised by the local LLDP agent.
- `-r` displays remote information received by the LLDP agent.
- `-v` displays detailed local or remote information.

EXAMPLE 5-6 Displaying Advertised LLDP Agent Information

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.

To display local information that is advertised by the LLDP agent:

```
# lldpadm show-agent -l net0
AGENT   CHASSISID   PORTID
net0    004bb87f    00:14:4f:01:77:5d
```

To display remote information that is advertised by the LLDP agent:

```
# lldpadm show-agent -r net0
AGENT   SYSNAME   CHASSISID   PORTID
net0    hostb     0083b390    00:14:4f:01:59:ab
```

To display local information in the verbose mode, use the -v option:

```
# lldpadm show-agent -l -v net4
                                    Agent: net4
                          Chassis ID Subtype: Local(7)
```

```
                                Chassis ID: 00843300
                           Port ID Subtype: MacAddress(3)
                                   Port ID: 00:1b:21:89:03:d0
                          Port Description: --
                              Time to Live: 21 (seconds)
                               System Name: --
                        System Description: --
                    Supported Capabilities: --
                      Enabled Capabilities: --
                        Management Address: --
                        Maximum Frame Size: --
                              Port VLAN ID: --
                              VLAN Name/ID: vlan1/22
                       VNIC PortID/VLAN ID: 02:08:20:63:2d:9d,02:08:20:e5:6c:af/21
                  Aggregation Information: --
                               PFC Willing: On
                                   PFC Cap: 8
                                   PFC MBC: False
                                PFC Enable: 4
                               PFC Pending: True
             Application(s)(ID/Sel/Pri): --
                               ETS Willing: On
                      ETS Configured CBS: 0
                      ETS Configured TCS: 8
                      ETS Configured PAT: 0,1,2,3,4,5,6,7
                      ETS Configured BAT: 40,20,0,40,0,0,0,0
                      ETS Configured TSA: 2,2,2,2,2,2,2,2
                     ETS Recommended PAT: 0,1,2,3,4,5,6,7
                     ETS Recommended BAT: 40,20,0,40,0,0,0,0
                     ETS Recommended TSA: 2,2,2,2,2,2,2,2
                                 EVB Mode: Station
                    EVB GID (Station): Not Supported
               EVB ReflectiveRelay REQ: Not Requested
            EVB ReflectiveRelay Status: RR Not Enabled
                      EVB GID (Bridge): Not Supported
          EVB ReflectiveRelay Capable (RRCAP): Not Supported
          EVB ReflectiveRelay Control (RRCTR): Not Enabled
                       EVB max Retries (R): 0
      EVB Retransmission Exponent (RTE): 0
          EVB Remote or Local(ROL) and
              Resource Wait Delay (RWD): Local
          EVB Resource Wait Delay (RWD): 0
         EVB Remote or Local (ROL) and
                Reinit Keep Alive (RKA): Local
            EVB Reinit Keep Alive (RKA): 0
               Next Packet Transmission: 4 (seconds)
```

To display remote information in the verbose mode, use the -v option:

```
# lldpadm show-agent -r -v net4
                                   Agent: net4
                         Chassis ID Subtype: Local(7)
                                 Chassis ID: 00843300
                            Port ID Subtype: MacAddress(3)
                                    Port ID: 00:1b:21:89:03:d0
                           Port Description: --
                               Time to Live: 21 (seconds)
                                System Name: --
```

```
                System Description: --
             Supported Capabilities: --
               Enabled Capabilities: --
                  Management Address: --
                  Maximum Frame Size: --
                        Port VLAN ID: --
                        VLAN Name/ID: vlan1/22
                   VNIC PortID/VLAN ID: 02:08:20:63:2d:9d,02:08:20:e5:6c:af/21
              Aggregation Information: --
                           PFC Willing: On
                               PFC Cap: 8
                               PFC MBC: False
                            PFC Enable: 4
         Application(s)(ID/Sel/Pri): --
                           ETS Willing: On
                  ETS Configured CBS: 0
                  ETS Configured TCS: 8
                  ETS Configured PAT: 0,1,2,3,4,5,6,7
                  ETS Configured BAT: 40,20,0,40,0,0,0,0
                  ETS Configured TSA: 2,2,2,2,2,2,2,2
                 ETS Recommended PAT: 0,1,2,3,4,5,6,7
                 ETS Recommended BAT: 40,20,0,40,0,0,0,0
                 ETS Recommended TSA: 2,2,2,2,2,2,2,2
                             EVB Mode: Station
                   EVB GID (Station): Not Supported
      EVB ReflectiveRelay REQ: Not Requested
    EVB ReflectiveRelay Status: RR Not Enabled
                EVB GID (Bridge): Not Supported
EVB ReflectiveRelay Capable (RRCAP): Not Supported
EVB ReflectiveRelay Control (RRCTR): Not Enabled
               EVB max Retries (R): 0
  EVB Retransmission Exponent (RTE): 0
      EVB Remote or Local(ROL) and
          Resource Wait Delay (RWD): Local
     EVB Resource Wait Delay (RWD): 0
    EVB Remote or Local (ROL) and
           Reinit Keep Alive (RKA): Local
       EVB Reinit Keep Alive (RKA): 0
           Information Valid Until: 19 (seconds)
```

Displaying 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.

- To display all statistics about LLDP packet transmission and reception:

  ```
  # lldpadm show-agent -s agent
  ```

- 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 5-7 Displaying LLDP Packet Statistics

This example shows how to display information about LLDP packet advertisement.

```
# lldpadm show-agent -s net0
AGENT IFRAMES IERR IDISCARD OFRAMES OLENERR TLVDISCARD TLVUNRECOG AGEOUT
net0        9    0        0      14       0          4          5      0
```

This 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`, `IERR`, 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 and packets that have length errors.
- `TLVDISCARD` and `TLVUNRECOG` display information about TLV units that are discarded and 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.

EXAMPLE 5-8 Displaying Selected LLDP Packet Statistics

This example shows how to display selected statistics information.

```
# # lldpadm show-agent -s -o iframes,oframes net4
IFRAMES  OFRAMES
0        10
```

♦ ♦ ♦ CHAPTER 6

Managing Converged Networks by Using Data Center Bridging

Traditionally, different networks are used for traffic management based on the application requirements, and load distribution of network traffic that is based on the available bandwidth. For example, local area network (LAN) uses Ethernet and storage area network (SAN) uses fibre channel. However, data center bridging enhances the Ethernet making it more suitable for running different types of traffic, which is converged traffic, and also to support features such as losslessness. DCB enables efficient network infrastructure by consolidating SAN and LAN and thereby reducing operational and management costs in data centers.

This chapter contains the following topics:

Overview of Data Center Bridging

Data center bridging is used to manage the bandwidth, relative priority, and flow control of multiple traffic types when sharing the same network link, for example, when sharing a datalink between networking and storage protocols. Fibre channel can be dedicated to host this type of traffic. However, using dedicated links to service only fibre channel traffic can be costly. Therefore, fibre channel over Ethernet (FCoE) is more commonly used. DCB addresses the sensitivity of fibre channels to packet loss while traversing an Ethernet network.

DCB distinguishes traffic based on priorities, which are also called class of service (CoS) priorities. The host and the next hop use the DCB exchange protocol (DCBX) to negotiate a network configuration, such as no traffic loss and minimum bandwidth share, based on priorities. This process enables packets from different applications on the host and in the network to be treated according to their priorities and the corresponding configuration to be negotiated by using DCBX.

Each packet in a DCB network has a VLAN header that contains a DCB 3-bit priority value, which is a DCB priority. This IEEE 802.1p priority value differentiates each Ethernet packet in the network from the other packets. Depending on the priority values of the packets, you can configure DCB to allocate specific bandwidth to the packets. For example, all packets with a priority `1` must have PFC enabled and all packets with a priority `2` must have PFC disabled and a bandwidth share of 10%.

You can configure DCB features such as priority-based flow control (PFC) and enhanced transmission selection (ETS) based on priorities. For more information about PFC and ETS, see "Priority-Based Flow Control" on page 99 and "Enhanced Transmission Selection" on page 99.

The DCB `cos` datalink property enables you to specify the CoS or priority of the datalink. The `cos` value that is set on a primary datalink does not apply to the VNICs that are created over this physical link. For information about customizing PFC based on the `cos` property, see "Customizing Priority-Based Flow Control for DCB" on page 101. For information about customizing ETS based on the `cos` property, see "Customizing Enhanced Transmission Selection for DCB" on page 107.

In Oracle Solaris, LLDP is used to exchange DCBX type-length-value (TLV) units. For more information about LLDP, see Chapter 5, "Exchanging Network Connectivity Information With Link Layer Discovery Protocol". Provided that the underlying network interface card (NIC) supports DCB features such as priority-based flow control and enhanced transmission selection, configuration information for these features can be shared with peer hosts on the network, as follows:

- PFC prevents packet loss by implementing a mechanism that pauses traffic flow for packets with a defined class of service (CoS). For more information about CoS, see the description of the `cos` link property in the `dladm`(1M) man page.
- ETS enables bandwidth sharing among packets based on the defined CoS. See "Enhanced Transmission Selection" on page 99.

Considerations When Using DCB

Note the following considerations for using DCB:

- DCB is *only* supported on Intel Niantic physical NICs.

  To verify whether the NIC supports DCB, issue the following command:

```
# dladm show-linkprop -p ntcs agent
```

A property value that is greater than zero (`0`) indicates that the NIC supports DCB.

- DCB ports that are configured in DCB mode cannot be aggregated (trunk or DLMP mode).
- Because DCB supports only the IEEE version of DCBX (not the CEE or CIN versions), external bridges must support the IEEE version to interoperate with Oracle Solaris DCB.
- DCB supports ETS configuration and recommendation TLVs.
- DCB is only supported in the eight traffic class configuration.
- DCB does not support congestion notification (CN).

Priority-Based Flow Control

Priority-based flow control (PFC) extends the standard IEEE 802.3x 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 `cos` property value for which traffic needs to be paused. The sending host stops traffic for that `cos` property value while traffic for other disabled `cos` property values are unaffected. After a time interval that is specified in the PFC frame, transmission resumes for the paused packets.

Pausing based on CoS values ensures that packets are not dropped for that `cos` property value. For packets without any defined CoS value or with CoS values that do not have PFC enabled, no PAUSE frames are sent. Therefore, traffic continues to flow, and packets might be dropped during traffic congestion. Handling loss of packets depends on the protocol stack, for example, TCP.

Two types of DCB information exist on the host: local DCB information and remote DCB information. For the PFC features to be effective, the local and remote types of DCB information for PFC on the host must be symmetric. The local host must be able to match the DCB information that it receives from the peer. If you enable DCB on your system, DCB synchronizes the DCB information with the peer.

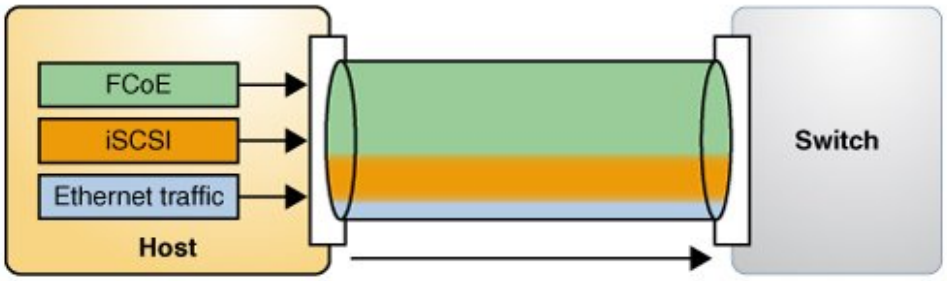
In most cases, the default configuration for PFC is sufficient. This configuration is automatically set up when you enable LLDP. However, you can adjust different options when configuring PFC. For more information, see “Customizing Priority-Based Flow Control for DCB” on page 101 and “Displaying PFC Configuration Information” on page 104.

Enhanced Transmission Selection

ETS is a DCB feature that enables you to allocate bandwidth on a NIC to applications based on their DCB priority.

The following figure shows the ETS feature of DCB in a network.

FIGURE 6-1 Enhanced Transmission Selection in DCB



The host in the figure has different types of traffic, such as FCoE and iSCSI, that share the link bandwidth. In the figure, the priority and bandwidth as shown in the following table are assigned for different types of traffic.

Traffic	Priority	Bandwidth
FCoE	3	60%
iSCSI	4	30%
Ethernet (non iSCSI) traffic	0	10%

The DCBX ETS TLV with the corresponding ETS bandwidth allocation is as follows:

Priority	0	1	2	3	4	5	6	7
Bandwidth Allocation Percentage	10	0	0	60	30	0	0	0

To use the ETS feature, the NIC must support the feature and run in the DCB mode. When you enable LLDP, the default configuration for the ETS feature is automatically set up if the underlying link supports DCB. However, you can modify the default configuration. For more information, see “Customizing Enhanced Transmission Selection for DCB” on page 107, “Recommending ETS Configuration to the Peer” on page 110, and “Displaying ETS Configuration Information” on page 112.

Enabling DCBX

Support for DCBX is automatically enabled when you enable LLDP. This procedure provides alternative manual steps in case certain automatic processes fail.

▼ How to Enable the Data Center Bridging Exchange Feature Manually

Before You Begin Ensure that you install LLDP. For more information about enabling LLDP, see "Enabling LLDP on the System" on page 83.

1. **Become an administrator.**

   For more information, see "Using Your Assigned Administrative Rights" in "Securing Users and Processes in Oracle Solaris 11.2 ".

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 agent
   ```

   If the LLDP agent is not enabled on both modes, type the following command:

   ```
   # lldpadm set-agentprop -p mode=both agent
   ```

   For the other possible configurations of the LLDP agents, see "Enabling LLDP on the System" on page 83.

4. **Verify that the underlying NIC supports DCB.**

   ```
   # dladm show-linkprop -p ntcs agent
   ```

   A property value that is greater than zero (`0`) indicates that the NIC supports DCB.

Customizing Priority-Based Flow Control for DCB

PFC and ETS are enabled by default only if the underlying NIC is in DCB mode. If you prefer to use only PFC, then you must remove `etscfg` from the `dot1 -tlv` property of the LLDP agent by using the following command:

```
# lldpadm set-agentprop -p dot1-tlv-=etscfg net0
```

For a list of possible values for `dot1 -tlv`, refer to Table 5-1.

Setting the PFC-Related Datalink Properties

The PFC feature of DCB provides the following datalink properties:

- `pfcmap` – Provides information about priority definitions and mappings. The `pfcmap` property refers to an 8-bit mask (`0`–`7`) that represent priorities. 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.
- `pfcmap-rmt` – Specifies the operative PFC mapping on the remote peer. This property is read-only.

In a DCB network, when a receiver is unable to keep up with the incoming rate of traffic, it sends a PFC frame to the sender requesting the sender to pause traffic for priorities that have PFC enabled. For any packet transmitted over a link, DCB sends a PFC frame to the sending host if traffic congestion accumulates on the receiving host. To send PFC frames properly, the communicating hosts must have symmetric DCB configuration information. A system can automatically adjust its PFC configurations to match the PFC configurations on the remote peer. You can determine the operative PFC mapping on the local host by using the `dladm show-linkprop` command that displays the `EFFECTIVE` value for the `pfcmap` property. For more information, see “Displaying Datalink Properties” on page 104.

▼ How to Customize Priority-Based Flow Control for DCB

1. **Become an administrator.**

   For more information, see “Using Your Assigned Administrative Rights” in “Securing Users and Processes in Oracle Solaris 11.2 ”.

2. **Ensure that the `flowctrl` property of the datalink is set to `pfc`.**

   ```
   # dladm show-linkprop -p flowctrl datalink
   ```

   If the property is not set to `pfc` or `auto`, use the following command:

   ```
   # dladm set-linkprop -p flowctrl=pfc datalink
   ```

3. **Set the `pfcmap` property to a value other than the default value `11111111`.**

```
# dladm set-linkprop -p pfcmap=value  datalink
```

For example, to enable priority only on CoS priority 6, type the following command:

```
# dladm set-linkprop -p pfcmap=01000000 net0
```

Setting the PFC TLV Units

DCB uses PFC TLV units to exchange PFC information between hosts. Either hosts must have the same value for the `pfcmap` property or at least one host must be willing to accept the configuration of its peer. You can set the `pfcmap` property by using the `dladm set-linkprop` command.

The TLV property `willing` indicates whether the host is ready to accept the peer's configuration if the host configuration differs from the peer's configuration. By default, the property value of `willing` is set to `on`, which indicates that the host will accept the peer's configuration.

To verify that the host can synchronize its PFC information with the PFC information on the remote peer, you must determine whether the `willing` is set to `on` by using the following command:

```
# lldpadm show-agenttlvprop -p willing -a agent pfc
```

If the PFC TLV property `willing` is set to `off`, type the following command to set the property `willing` to `on` and enable synchronization.

```
# lldpadm set-agenttlvprop -p willing=on -a agent pfc
```

where *agent* is the datalink on which the agent is enabled.

EXAMPLE 6-1 Enabling Synchronization Between the Host and the Peer

To enable synchronization for a `net0` datalink, type the following command:

```
# lldpadm set-agenttlvprop -p willing=on -a net0 pfc

# dladm show-linkprop -p pfcmap,pfcmap-rmt net0
LINK   PROPERTY    PERM  VALUE      EFFECTIVE  DEFAULT    POSSIBLE
net0   pfcmap      rw    11111111   00010000   11111111   00000000-11111111
net0   pfcmap-rmt  r-    --         00010000   11111111   --
```

In the example, the `pfcmap` and `pfcmap-rmt` properties have the value `00010000`. This indicates that the local host has synchronized with the peer. Hence, PFC is enabled with a priority of `4` on both the host and the peer.

For more information, see the `lldpadm`(1M) man page.

Displaying PFC Configuration Information

This section describes commands to display information related to PFC after LLDP and DCB are configured and provides examples to show the use of these commands.

Displaying Datalink Properties

The following command displays the priority definitions and the effective PFC mappings on the datalink:

```
# dladm show-linkprop -p pfcmap,pfcmap-rmt datalink
```

On a datalink with matching PFC information between the local and remote peers, the values of the `EFFECTIVE` column for `pfcmap` and `pfcmap-rmt` properties are identical regardless of the value set for the `pfcmap` property. If the ability to synchronize is disabled on the local host, then the `EFFECTIVE` field for the `pfcmap` property reflects the value of the `pfcmap` property for the local host.

EXAMPLE 6-2 Displaying PFC-Related Datalink Properties

This example shows how to display the status of physical datalink properties that are related to priority-based flow control.

```
# dladm show-linkprop -p pfcmap,pfcmap-rmt net0
LINK   PROPERTY      PERM   VALUE      EFFECTIVE  DEFAULT    POSSIBLE
net0   pfcmap        rw     11111111   11111111   11111111   00000000-11111111
net0   pfcmap-rmt    r-     --         --         --         --
```

In the example, the `value` field for the `pfcmap` property has the value of `11111111`. This value indicates that the PFC mapping on the local host has the default value where all eight priorities are enabled. The `EFFECTIVE` values for the `pfcmap` and `pfcmap-rmt` properties are `11111111` and `--`. These mismatched values for the `EFFECTIVE` field indicate that the local host has not synchronized its PFC information with the remote peer.

You can use the `lldpadm show-agenttlvprop` command to verify the value of the `willing` property and the `lldpadm show-agent -r` command to check the PFC TLV information from the peer.

Displaying the Capability of the Local Host to Synchronize PFC Information

The following command displays the PFC TLV property that controls a host's capability to synchronize its PFC mapping with a peer.

```
# lldpadm show-agenttlvprop -a agent pfc
```

where *agent* is identified by the datalink on which LLDP is enabled.

EXAMPLE 6-3 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 PFC configurations of the peer.

```
# lldpadm show-agenttlvprop -a net0 pfc
AGENT   TLVNAME   PROPERTY   PERM   VALUE   DEFAULT   POSSIBLE
net0    pfc       willing    rw     off     on        on,off
```

For more information, see "Setting the PFC TLV Units" on page 103.

Displaying PFC Mapping Information Between Host and Peer

The `PFC Pending` value 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`.

The following command alerts you to a mismatch of PFC mapping information between the local host and the peer.

```
# lldpadm show-agent -lv -o "PFC Pending" agent
```

```
# lldpadm show-agent -lv -o "PFC Pending" agent
```

EXAMPLE 6-4 Verifying Symmetry of PFC Information Between Host and Peer

The following example shows how to verify in actual running time whether PFC information is synchronized between the host and peer, or whether a mismatch occurs.

```
# lldpadm show-agent -lv -o "PFC Pending" net0
PFC Pending: True
```

To display all the information that the agent advertises, use the -v (verbose) option of the lldpadm show-agent command:

```
# lldpadm show-agent -v net0
```

Displaying Priority Definitions

The following command displays PFC information on the physical link with regards to enabled priorities on the NIC:

```
# dladm show-phys -D pfc datalink
```

EXAMPLE 6-5 Displaying CoS Priority Definitions

This example shows how to display the current priority definitions on a specific physical link 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
net0   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 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. As a result, no PFC frames would be exchanged for traffic on any VNIC 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 Priority Configurations

DCBX exchanges the priority information that is associated with the application. Application TLV units that are exchanged through DCBX 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 in the table contains the name of the application and the priority assigned to the application. When you set a priority for an application, all the DCB settings of the PFC and ETS of that priority are applicable to the application. The application TLV uses the table data for exchanging application priority information with the other hosts.

By default, the application feature accepts priority mappings from the peer. The `willing` property of the application TLV `appln` is similar to PFC and enables information exchange between the peers.

Entries on the table uses the following format:

protocol-ID/*selector*/*priority*

The pair *protocol-ID*/*selector* identifies the application. The priority for a corresponding application is identified by a priority that contains a value from `0` to `7`.

To exchange this information about the priority of an application 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 5-3 that lists the parameters for setting an application TLV you would 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   --        --
```

Customizing Enhanced Transmission Selection for DCB

The default configuration is automatically set up when LLDP is enabled and DCB is supported by the underlying link. In this default configuration, the `cos` value `0` is assigned all of the bandwidth. However, you can use the `dladm set-linkprop` command to configure the `cos` values on a datalink to assign part of the bandwidth to that datalink.

ETS configuration and recommendation TLVs are enabled by default for a NIC. For a list of possible values for `dot1-tlv`, refer to Table 5-1.

If you want to remove the `pfc` TLV, type the following command:

```
# lldpadm set-agenttlvprop -p dot1-tlv-=pfc agent
```

Setting the ETS-Related Datalink Properties

The properties of datalinks that refer to PFC information apply to the prevention of packet loss based on the priorities defined for the packets. The ETS properties relate to assigning shares of the underlying link's bandwidth based on priorities.

DCB provides the following ETS-related properties:

- `cos` – Specifies the class of service or priority of the datalink. The value of this property ranges from `0` to `7`. The default value is `0`. The `cos` value is set in the VLAN tag of the packets that are transmitted over this link.
- `etsbw-lcl` – Indicates the ETS bandwidth that is allocated on the transmit (Tx) side for the datalink. This property is configurable only if the underlying physical NIC has DCB capabilities and supports ETS and the link's `cos` property is not set to `0`. You set a value for this property on a datalink by specifying the percentage of total bandwidth of the underlying physical link. The sum of the values for the `etsbw-lcl` property for all the datalinks over the same physical NIC must not exceed 100%.

  The bandwidth percentage that is defined on `etsbw-lcl` is not reserved only for that datalink. If the allocated bandwidth is not used, then it can be used by other datalinks on that physical NIC. Further, the bandwidth allocation is enforced only on the transmission side of the host's traffic.
- `etsbw-rmt-advice` – Specifies the recommended ETS bandwidth value sent to the peer. By default, the locally configured value of the `etsbw-lcl` property is recommended to the peer. However, you can recommend a value that is different from the `etsbw-lcl` property by explicitly configuring the `etsbw-rmt-advice` datalink property.

  Configuring the `etsbw-rmt-advice` property is useful if the bandwidth assignment for a datalink is asymmetrical, which means that the receive (Rx) and transmit (Tx) bandwidth are different. When you explicitly set the `etsbw-rmt-advice` property, transmission of the ETS recommendation DCBX TLV starts automatically.
- `etsbw-lcl-advice` – Specifies the recommended bandwidth share for the datalink, which is sent by the peer to the local host. This is a read-only property.
- `etsbw-rmt` – Specifies the bandwidth share that is configured on the peer for the datalink. This is a read-only property.

To set the priority and to allocate a bandwidth to the VNIC, use the following commands:

- To set the priority to the VNIC:

  ```
  # dladm set-linkprop -p cos=value VNIC
  ```
- To allocate a percentage of the bandwidth of the underlying physical link to a VNIC:

  ```
  # dladm set-linkprop -p etsbw-lcl=value VNIC
  ```

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.

- To explicitly recommend a bandwidth that is sent to the peer:

```
# dladm set-linkprop -p etsbw-rmt-advice=value VNIC
```

You can determine the actual bandwidth share that is implemented on the local host's datalink and the bandwidth share that is configured on the peer's datalink by using the `dladm show-linkprop` command. The value in the `EFFECTIVE` field of the output for the `etsbw-lcl` and `etsbw-rmt` properties shows the actual bandwidth share implemented. For more information, see "Displaying ETS Configuration Information" on page 112.

For the appropriate bandwidth to be used for packets with specific priorities, symmetric or synchronized ETS information between the communicating hosts is preferable. Specifically, should be the local system that is able to adjust its bandwidth share to the value of `etsbw-lcl-advice`. An Oracle Solaris system can automatically adjust its ETS configurations to match the ETS recommendation from the peer.

Setting ETS TLV Units

The ETS TLV (`etscfg`) configuration determines how the host responds to ETS recommendations from the peer. 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 recommendation of the remote peer.

To verify that the host can synchronize its ETS information with the ETS information of the remote peer, use the following command:

```
# lldpadm show-agenttlvprop -p willing -a agent etscfg
```

If the `willing` property is set to `off`, type the following command to establish synchronization:

```
# lldpadm set-agenttlvprop -p willing=on -a agent etscfg
```

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 the datalink on which the agent is enabled.

Recommending ETS Configuration to the Peer

The configured ETS bandwidth values (`etsbw-lcl`) for each priority can be recommended to the peer so that the peer can configure the same values. You must enable the `etsreco` property in the `dot1-tlv` type of the LLDP agent on the NIC to recommend the ETS bandwidth values. The recommended values can be the same as the locally configured ETS values or you can also explicitly configure the recommended values by setting the new datalink property `etsbw-rmt-advice` by using the `dladm set-linkprop` command. Configuring the `etsbw-rmt-advice` property is useful if the assigned bandwidth for a datalink is asymmetrical, which means that the receive (Rx) and transmit (Tx) bandwidth are different.

By default, the configured values of the `etsbw-lcl` property are used to recommend values to the peer. However, you can recommend a different ETS value by setting a different value for the `etsbw-rmt-advice` property. For example, if the network traffic is more on the Tx, then you can configure a higher ETS value for the `etsbw-lcl` property (Tx on the host) and a lower value configured for the `etsbw-rmt-advice` property (Rx to the host).

EXAMPLE 6-6 Recommending an ETS Configuration to the Peer

1. Ensure that the `etsreco` property is enabled by displaying the `dot1-tlv` type property of the LLDP agent for `net5`.

```
# lldpadm show-agentprop -p dot1-tlv net5
AGENT        PROPERTY   PERM VALUE          DEFAULT        POSSIBLE
net5         dot1-tlv   rw   etsreco,etscfg none           none,vlanname,pvid,
                                                           linkaggr,pfc,appln,
                                                           evb,etscfg,etsreco,
                                                           all
```

2. Allocate a share of 20% of the underlying link's bandwidth for `vnic1`.

```
# dladm set-linkprop -p etsbw-lcl=20 vnic1
# dladm show-linkprop -p etsbw-lcl vnic1
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
vnic1    etsbw-lcl       rw   20           20           0         --
```

   By default, the same value is recommended for the peer.

```
# dladm show-linkprop -p etsbw-rmt-advice vnic1
LINK     PROPERTY         PERM VALUE  EFFECTIVE    DEFAULT   POSSIBLE
vnic1    etsbw-rmt-advice rw   --      20           0         --
```

3. Display the information exchanged by LLDP.

```
# lldpadm show-agent -l -v net5
```

4. Display the bandwidth recommended to the peer.

```
# dladm show-phys -D ets -r net5
LINK      COS ETSBW_RMT_EFFECT ETSBW_RMT_ADVICE CLIENTS
--         0   0                80               net5
           1   0                0                --
           2   0                0                --
           3   0                20               vnic1
           4   0                0                --
           5   0                0                --
           6   0                0                --
           7   0                0                --
```

   By default, the values configured for the `etsbw-lcl` property for each priority on `net5` are sent as the recommended values to the peer. The `ETSBW_RMT_ADVICE` shows the values recommended to the peer. The output also shows that the peer has not configured any ETS bandwidth on its end. You can also display bandwidth recommended to the peer by using the `lldpadm show-agent` command.

5. Recommend a different value to the peer.

```
# dladm set-linkprop -p etsbw-rmt-advice=10 vnic1
# dladm show-linkprop -p etsbw-rmt-advice vnic1
LINK     PROPERTY          PERM VALUE     EFFECTIVE    DEFAULT   POSSIBLE
vnic1    etsbw-rmt-advice  rw   10         10           0         --
```

6. Display the bandwidth recommended to the peer.

```
# dladm show-phys -D ets -r net5
LINK                COS ETSBW_RMT_EFFECT ETSBW_RMT_ADVICE CLIENTS
--                  0   0                90               net5
                    1   0                0                --
                    2   0                0                --
                    3   0                10               vnic2
                    4   0                0                --
                    5   0                0                --
                    6   0                0                --
                    7   0                0                --
```

   The `ETSBW_RMT_EFFECT` field shows value `0` for `vnic2`, which indicates that the peer has not set any bandwidth on its end, even though you have recommended bandwidth values. This situation means that peer might not have enabled LLDP or does not support ETS.

Displaying ETS Configuration Information

You can use the following commands to display information about the ETS configuration:

- `# dladm show-linkprop -p etsbw-lcl,etsbw-rmt,etsbw-lcl-advice,etsbw-rmt-advice` *datalink*

  This command displays the information related to ETS on a physical link.

- `# dladm show-phys -D ets` *phys-link*

  This command displays the local and remote 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 the datalink on which LLDP is enabled. This command displays the ETS TLV property that controls the capability of a host to synchronize ETS information with a peer.

EXAMPLE 6-7 Displaying ETS-Related Datalink Properties

This example shows how to display the status of datalink properties that are related to ETS before synchronization is enabled.

```
# dladm show-linkprop -p cos,etsbw-lcl,etsbw-rmt,etsbw-lcl-advice, \
etsbw-rmt-advice vnic1
LINK    PROPERTY          PERM  VALUE  EFFECTIVE  DEFAULT    POSSIBLE
vnic1   cos               rw    2      2          0          0-7
vnic1   etsbw-lcl         rw    10     10         0          --
vnic1   etsbw-rmt         r-    20     20         --         --
vnic1   etsbw-lcl-advice  r-    20     20         --         --
vnic1   etsbw-rmt-advice  rw    10     10         0          --
```

The output shows that the host has set and recommended an ETS value of 10% for `vnic1` with a `cos` value of `2`. However, the peer has set and recommended an ETS value of 20% with a `cos` value of `2` for `vnic1`. Because the synchronization is not enabled (`willing` is not enabled) the host has not accepted the peer's recommendation, which is reflected in the `EFFECTIVE` value of the `etsbw-lcl` property (locally configured value).

EXAMPLE 6-8 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 ETS configurations of the peer.

```
# lldpadm show-agenttlvprop -a net0 etscfg
AGENT   TLVNAME   PROPERTY   PERM   VALUE   DEFAULT   POSSIBLE
net0    etscfg    willing    rw     off     on        on,off
```

To enable synchronization, type the following commands:

```
# lldpadm set-agenttlvprop -p willing=on -a net0 etscfg

# dladm show-linkprop -p  cos,etsbw-lcl,etsbw-rmt, \
etsbw-lcl-advice,etsbw-rmt-advice vnic1
LINK    PROPERTY           PERM  VALUE   EFFECTIVE  DEFAULT    POSSIBLE
vnic1   cos                rw    2       2          0          0-7
vnic1   etsbw-lcl          rw    10      20         0          --
vnic1   etsbw-rmt          r-    20      20         --         --
vnic1   etsbw-lcl-advice   r-    20      20         --         --
vnic1   etsbw-rmt-advice   rw    10      10         0          --
```

Because the synchronization is enabled (the property `willing` is enabled), the host has accepted the peer's recommendation, which is reflected in the `EFFECTIVE` value of `etsbw-lcl`.

The following example shows effective ETS values on the host and the peer for each priority value on the physical link.

```
# dladm show-phys -D ets net4
LINK     COS ETSBW_LCL_EFFECT ETSBW_RMT_EFFECT ETSBW_LCL_SOURCE CLIENTS
net4      0   0                30               local            net4
          1   0                0                local            --
          2   0                0                local            --
          3   0                0                local            --
          4   0                70               local            --
          5   0                0                local            --
          6   0                0                local            --
          7   0                0                local            --
```

`ETSBW_LCL_EFFECT` — Displays the effective ETS bandwidth as a percentage for the priority.

`ETSBW_RMT_EFFECT` — Displays the effective ETS bandwidth as a percentage for the priority value on the peer.

`ETSBW_LCL_SOURCE` — Indicates the source for the `ETSBW_LCL_EFFECT` value. This value could be either `local`, which is the configured value, or `remote`, which is the recommended value.

The following example shows the local ETS information including the locally configured values, local effective values, and the values recommended by the peer.

```
# dladm show-phys -D ets -l net5
LINK               COS ETSBW_LCL ETSBW_LCL_EFFECT ETSBW_LCL_ADVICE CLIENTS
--                 0   80        80               0                net5
                   1   0         0                0                --
                   2   0         0                0                --
                   3   20        20               0                vnic2
                   4   0         0                0                --
                   5   0         0                0                --
                   6   0         0                0                --
                   7   0         0                0                --
```

Because the peer has not recommended any values, the local effective value (`ETSBW_LCL_EFFECT`) is set by using the local configured value (`ETSBW_LCL`).

The following example displays information about the peer.

```
# dladm show-phys -D ets -r net5
LINK                COS ETSBW_RMT_EFFECT ETSBW_RMT_ADVICE CLIENTS
--                  0   0                20               net5
                    1   0                0                --
                    2   0                0                --
                    3   0                80               vnic2
                    4   0                0                --
                    5   0                0                --
                    6   0                0                --
                    7   0                0                --
```

The output shows that the remote peer does not have any value set in the `ETSBW_RMT_EFFECT` field even though the host had recommended the peer to set 80% for priority `3`.

♦ ♦ ♦ APPENDIX A

Link Aggregations and IPMP: Feature Comparison

Link aggregation and IPMP are different technologies that achieve improved network performance and maintain network availability.

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

Feature	Link Aggregation	IPMP
Network technology type	Layer 2 (link layer).	Layer 3 (IP layer).
Configuration tool	`dladm`	`ipadm`
Link-based failure detection	Supported.	Supported.
Probe-based failure detection	Trunk: Based on LACP, targeting the immediate peer host or switch. DLMP: Supported. ICMP-based, targeting any defined systems in the same subnet as DLMP addresses, across multiple levels of intervening Layer 2 switches.	ICMP-based, targeting any defined system in the same IP subnet as test addresses across multiple levels of intervening Layer 2 switches.
Use of standby interfaces	Trunk: Not supported. DLMP: Not supported.	Supported. Standby interfaces can be configured.
Span multiple switches	Trunk: Supported. However, requires switch vendor extensions. DLMP: Supported.	Supported.
Switch configuration	Trunk: Required. DLMP: Not required.	Not required.
Back-to-back configuration	Supported.	Not supported.
Media types supported	Ethernet-specific.	Broadcast-capable.
Load-spreading support	Trunk: Supported and controlled by the administrator by using the `dladm`	Supported. Controlled by the kernel. Inbound load spreading is indirectly affected by the source address selection.

Feature	Link Aggregation	IPMP
	command. Inbound load spreading is supported. DLMP: Supported across clients and VNICs of the aggregation. However, load spreading by individual clients and VNICs over the aggregation is not supported.	
Level of support when integrating with VNICs	Excellent support. Aggregation is configured in the control domain or the global zone only and is transparent to the zones.	Supported. However, VNIC properties such as bandwidth limit, dedicated Rx or Tx rings, and link protection cannot be enforced on an IPMP group. Requires multiple VNICs to be assigned to the zones and needs to be configured in every zone.
User defined flows for resource management	Supported.	Not supported.
Link protection	Supported.	Not supported.
Protocol requirements	None.	None.

In link aggregations, incoming traffic is spread over the multiple links that comprise the aggregation in trunk mode. Therefore, 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.

♦ ♦ ♦ APPENDIX B

Packet Format of Transitive Probes

The transitive probe packet is a proprietary protocol packet with an Ethernet type, `ETHERTYPE_ORCL` (`0x881b`). For more information about the transitive probes, see "Probe-Based Failure Detection" on page 23. For an example to display the statistics of the probes, see Example 2-7. The following figure shows the transitive probe packet format.

FIGURE B-1 Transitive Probe Packet

Field	Description
`Dest MAC`	Destination MAC address.
`Src MAC`	Source MAC address.
`Proto#`	Protocol number. The Layer 2 payload of `ETHERTYPE_ORCL` packets must start with a 16-bit protocol number, which is `1` for the transitive probe packet.
`Typ`	Probe packet type. The probe packet type is `0` for request and `1` for response.
`Pad`	Padding (all-zero).

Field	Description
`Seq#`	Probe sequence number.
`Timestamp`	Probe timestamp.
`TL`	Target information length. For Ethernet, target length is 6 bits (Ethernet MAC address length).
`Target`	Target port MAC address.
`SL`	Source information length. For Ethernet, source length is 6 bits.
`Source`	Source port MAC address.

Index

E

R

S

T

V

W