管理 Oracle® Solaris 11.1 网络性能

文件号码 E36609–02
2013 年 9 月

ORACLE

130924@25097

目录

前言

欢迎阅读《管理 Oracle Solaris 11.1 网络性能》。本书是“建立 Oracle Solaris 11.1 网络”系列的一部分，该系列包含配置 Oracle Solaris 网络的基本主题和过程。本书假定您已经安装 Oracle Solaris。您应该已经可以配置网络，或者已经可以配置网络上所需的任何网络软件。

目标读者

本书适用于所有负责管理在网络中配置的、运行 Oracle Solaris 的系统的人员。要使用本书，您应当至少具备两年的 UNIX 系统管理经验。参加 UNIX 系统管理培训课程可能会对您有所帮助。

获取 Oracle 支持

Oracle 客户可以通过 My Oracle Support 获取电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info，或访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs（如果您听力受损）。

印刷约定

下表介绍了本书中的印刷约定。

表 P–1 印刷约定

字体或符号	含义	示例
`AaBbCc123`	命令、文件和目录的名称；计算机屏幕输出	编辑 `.login` 文件。 使用 `ls -a` 列出所有文件。 `machine_name% you have mail.`
`AaBbCc123`	用户键入的内容，与计算机屏幕输出的显示不同	`machine_name%` **`su`** `Password:`

表 P–1 印刷约定 （续）

字体或符号	含义	示例
aabbcc123	要使用实名或值替换的命令行占位符	删除文件的命令为 `rm` *filename*。
AaBbCc123	保留未译的新词或术语以及要强调的词	这些称为 *Class* 选项。 **注意**：有些强调的项目在联机时以粗体显示。
新词术语强调	新词或术语以及要强调的词	**高速缓存**是存储在本地的副本。 请**勿**保存文件。
《书名》	书名	阅读《用户指南》的第 6 章。

命令中的 shell 提示符示例

下表显示了 Oracle Solaris OS 中包含的缺省 UNIX shell 系统提示符和超级用户提示符。请注意，在命令示例中显示的缺省系统提示符可能会有所不同，具体取决于 Oracle Solaris 发行版。

表 P–2 shell 提示符

shell	提示符
Bash shell、Korn shell 和 Bourne shell	`$`
Bash shell、Korn shell 和 Bourne shell 超级用户	`#`
C shell	`machine_name%`
C shell 超级用户	`machine_name#`

◆ ◆ ◆ 第 1 章

网络性能管理介绍

本章是绪论篇，将概述网络性能管理。本章还将介绍本书其余部分中所述的能够提高网络性能的功能。

在执行本书中所述的任一配置前，必须完成基本网络配置，以将系统连接到网络。有关基本网络配置的信息，请参见《在 Oracle Solaris 11.1 中使用反应性网络配置连接系统》和《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》。

有关 Oracle Solaris 11 系统上网络的一般概述，请参见《Oracle Solaris 11 联网介绍》

用于提高网络性能的功能

配置网络接口可将系统连接到网络。但是，Oracle Solaris 11 中的其他功能可提高网络的总体性能。网络性能管理是指使用这些功能和技术微调系统处理网络通信的方式。配置了这些技术的系统可以更好地管理网络通信，这有助于提高网络的总体性能。这些功能处理网络操作的不同方面。但是，它们可以提供共同的优势，如下所示：

- 网络连通性－某些功能（如链路聚合和 IPMP）可确保系统对网络进行连续访问。
- 效率－某些功能（如负载平衡）使系统能够在所有可用资源之间分配网络处理负载以提高效率。
- 网络管理－虚拟 LAN (virtual LAN, VLAN) 等功能可简化管理。使用其他改善网络的功能也能简化网络管理。
- 成本－所有这些 Oracle Solaris 11 技术都能在无需额外购买昂贵硬件的情况下提高网络性能。

本书介绍了可改进系统承载和处理网络通信方式的功能。请参考以下示例：

- 可以将数据链路和接口配置到一个单元中。合并的资源可专用于网络处理，从而提高网络的吞吐量。有关详细信息，请参见第 2 章，使用链路聚合或第 5 章，IPMP 介绍。

- 可以将局域网细分为多个虚拟子网以简化管理。有关详细信息，请参见第 3 章，使用 VLAN。
- 可以使系统通过最短的连接路由将包传送到其目的地。有关详细信息，请参见第 4 章，管理桥接网络（任务）。
- 在包含不同系统和配置的网络上，可以启用一种机制，向网络上的所有对等方通告各方的配置。因此，可基于两个对等方同时支持的一组协商配置交换网络包。对等协商是自动进行的。因此，不需要进行手动配置。有关详细信息，请参见第 7 章，使用 LLDP 交换网络连接信息。

本书中介绍的不同技术的使用取决于具体情况。此外，某些硬件配置可能会要求您使用特定类型的功能。例如，如果系统具有配置为属于同一个子网的多个接口，则必须使用 IP 多路径 (IP multipathing, IPMP) 技术。因此，您不需要完成本书中介绍的所有配置过程，而应该选择和部署能满足您的网络需求的技术。

另请参阅 Oracle Solaris 11.1 文档库，查找有助于改进网络的其他与网络相关的系统配置。例如，通过使用虚拟网络、区域和资源管理，能够部署多种集成网络性能，同时最大限度地利用可用的网络资源。有关详细信息，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》和《Oracle Solaris 11.1 管理：Oracle Solaris Zones、Oracle Solaris 10 Zones 和资源管理》。

◆ ◆ ◆ 第 2 章

使用链路聚合

通过链路聚合，可以将要管理的多个数据链路资源合并为一个单元。通过合并多个数据链路的资源，并将其专用于系统的网络操作，系统的性能会得到很大提高。本章介绍用于配置和维护链路聚合的过程。

本章包含以下主题：

链路聚合概述

链路聚合也称为**中继**，由系统上的多个接口组成，这些接口被配置为一个逻辑单元，以提高网络通信的吞吐量。下图显示了系统中配置的链路聚合的示例。

图 2–1 链路聚合配置

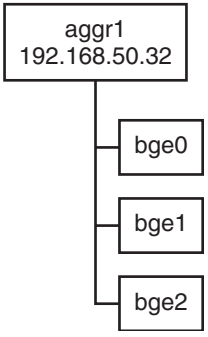


图 2–1 显示了由三个底层数据链路（net0 到 net2）组成的聚合 aggr1。这些数据链路专用于为通过该聚合遍历系统的通信提供服务。底层链路对外部应用程序是隐藏的。相反，可以访问逻辑数据链路 aggr1。

链路聚合具备以下功能：

- **增加了带宽**－将多个链路的容量组合到一个逻辑链路中。
- **自动故障转移和故障恢复**－通过支持基于链路的故障检测，将来自故障链路的通信故障转移到聚合中的其他工作链路。
- **改进了管理**－所有底层链路作为一个单元进行管理。
- **减少了网络地址池消耗**－可以将一个 IP 地址指定给整个聚合。
- **链路保护**－可以配置为流经聚合的包启用链路保护的数据链路属性。
- **资源管理**－可以通过网络资源的数据链路属性以及流定义控制应用程序的网络资源使用。有关资源管理的更多信息，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》中的第 3 章 “在 Oracle Solaris 中管理网络资源”。

注 – 链路聚合执行与 IP 多路径 (IP multipathing, IPMP) 类似的功能以提高网络性能和可用性。有关这两种技术的比较，请参见附录 B，链路聚合和 IPMP：功能比较。

Oracle Solaris 支持两种类型的链路聚合：

- 中继聚合
- 数据链路多路径 (Datalink multipathing, DLMP) 聚合

要快速查看这两种链路聚合类型之间的区别，请参见附录 A，链路聚合类型：功能比较。

以下各节将更详细地介绍每种类型的链路聚合。

中继聚合

中继聚合对具有不同通信负载的各种网络都有益处。例如，如果网络中的系统运行具有分布式大通信流量的应用程序，可以将中继聚合专用于该应用程序的通信以增加带宽。对于具有有限的 IP 地址空间但仍需要很大带宽的站点，大的接口聚合仅需要一个 IP 地址。对于需要隐藏内部接口的存在的站点，聚合的 IP 地址对外部应用程序隐藏其接口。

在 Oracle Solaris 中，创建聚合时缺省情况下会配置中继聚合。通常，配置了链路聚合的系统还会使用外部交换机连接到其他系统。请参见下图。

图 2–2 使用交换机的链路聚合

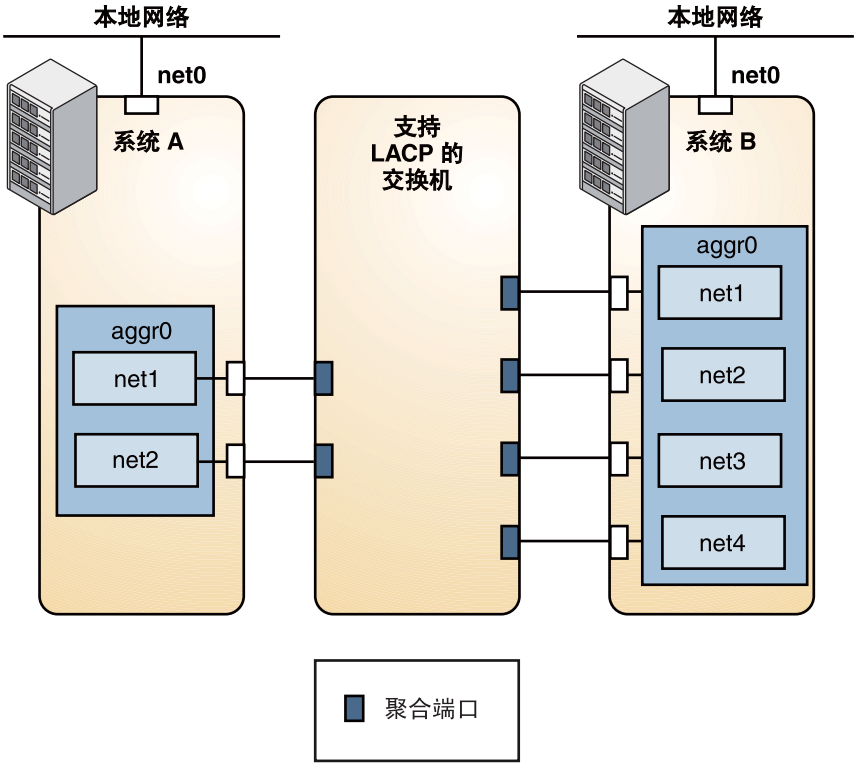


图 2–2 描述了包含两个系统的本地网络，其中每个系统都配置了一个聚合。这两个系统通过配置了链路聚合控制协议 (Link Aggregation Control Protocol, LACP) 的交换机连接在一起。

系统 A 的聚合由两个接口 `net1` 和 `net2` 组成。这些接口通过聚合端口连接到交换机。系统 B 的聚合由四个接口（即 `net1` 至 `net4`）组成。这些接口也连接到交换机上的聚合端口。

在该链路聚合拓扑中，交换机必须支持聚合技术。相应地，其交换机端口必须配置为管理来自这些系统的通信。

中继聚合还支持背对背配置。不使用交换机，将两个系统直接连接到一起以运行并行聚合，如下图所示。

图 2–3 背对背链路聚合配置

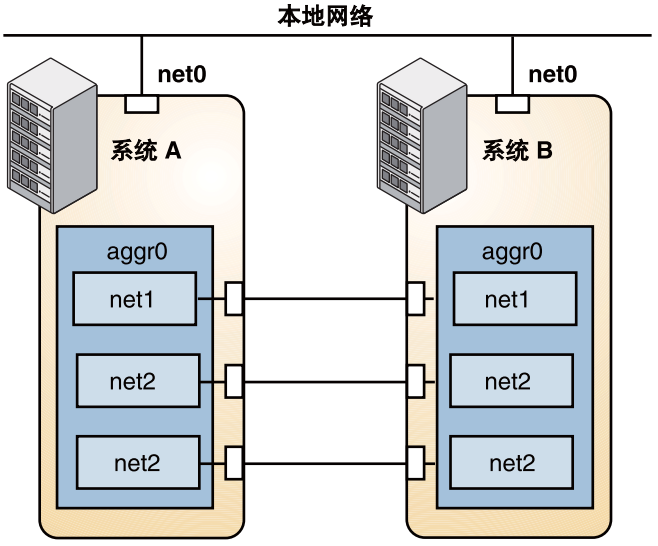


图 2–3 显示了系统 A 上的链路聚合 `aggr0` 直接与系统 B 上的链路聚合 `aggr0` 连接（通过各自底层数据链路之间的相应链路）。这样，系统 A 和 B 可以提供冗余和高可用性，以及这两个系统之间的高速通信。每个系统还将 `net0` 配置为用于本地网络内的通信流。

背对背链路聚合最常见的应用是镜像数据库服务器配置。这两个服务器必须一起更新，因此对带宽、高速通信流和可靠性要求很高。最常使用背对背链路聚合的是数据中心。

注 – DLMP 聚合不支持背对背配置。

以下各节介绍中继聚合特有的其他功能。请勿在创建 DLMP 聚合时配置这些功能。

策略和负载平衡

如果计划使用中继聚合，请考虑定义传出通信的策略。此策略可以指定希望如何在聚合的可用链路之间分配包，从而建立负载平衡。以下是可能用于聚合策略的层说明符及其意义：

- *L2* – 通过散列每个包的 MAC (L2) 头来确定传出链路
- *L3* – 通过散列每个包的 IP (L3) 头来确定传出链路
- *L4* – 通过散列每个包的 TCP、UDP 或其他 ULP (L4) 头来确定传出链路

这些策略的任意组合也是有效的。缺省策略是 L4。

聚合 LACP 模式和交换机

如果中继聚合的设置中包含交换机，则必须注意该交换机是否支持 LACP。如果交换机支持 LACP，则必须为交换机和聚合配置 LACP。可将聚合的 LACP 设为以下三个值之一：

- `off`－聚合的缺省模式。不生成 LACP 包（称为 *LACPDU*）。
- `active`－系统按固定的时间间隔（您可以指定该时间间隔）生成 LACPDU。
- `passive`－系统仅在收到来自交换机的 LACPDU 时才生成 LACPDU。如果聚合和交换机均在被动模式下进行配置，则它们无法交换 LACPDU。

数据链路多路径聚合

中继聚合一般可以满足网络设置的要求。但是，一个中继聚合只能与一个交换机配合使用。因此，该交换机成为系统聚合的单点故障。以前使聚合能够跨多个交换机的解决方案存在其自身的缺点：

- 在交换机上实现的解决方案是特定于供应商的，而非标准化的解决方案。如果使用了多个供应商提供的多种交换机，则某个供应商的解决方案可能不适用于其他供应商的产品。
- 使用 IP 多路径 (IP multipathing, IPMP) 合并链路聚合非常复杂，尤其是在包含全局区域和非全局区域的网络虚拟化环境中。扩大配置范围后复杂程度会增加，例如，在包含大量系统、区域、NIC、虚拟 NIC (virtual NIC, VNIC) 和 IPMP 组的方案中。该解决方案还需要您对各个系统上的全局区域和各个非全局区域进行配置。
- 即使实现了链路聚合与 IPMP 的合并，该配置也不能利用只在链路层工作的其他优点，例如，链路保护、用户定义的流以及定制链路属性（例如带宽）的能力。

DLMP 聚合可以克服这些缺点。下图显示了 DLMP 聚合的工作原理。

图 2–4 DLMP 聚合

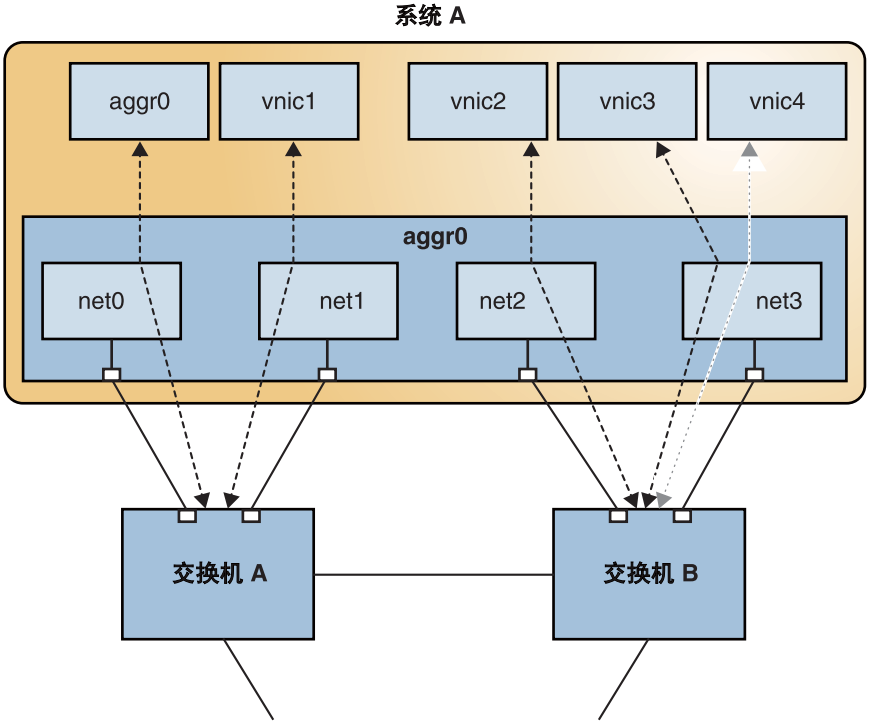


图 2–4 显示了具有链路聚合 aggr0 的系统 A。该聚合由 net0 到 net3 四个底层链路组成。除 aggr0 外，主接口、VNIC（vnic1 到 vnic4）也在该聚合上进行配置。该聚合连接到交换机 A 和交换机 B，这两个交换机反过来连接到范围更广的网络中的其他目标系统。

在中继聚合中，每个端口都与聚合上配置的各个数据链路关联。在 DLMP 聚合中，端口与聚合上配置的任意数据链路以及该聚合上的主接口和 VNIC 相关联。

如果 VNIC 数超过底层链路数，则单个端口与多个数据链路相关联。例如，图 2–4 显示 vnic4 与 vnic3 共享一个端口。

类似地，如果聚合的端口出现故障，则会将使用该端口的所有数据链路都分发到其他端口。例如，如果 net0 出现故障，则 aggr0 将与某个其他数据链路共享一个端口。聚合端口间的分发是对用户透明的，并独立于连接到聚合的外部交换机。

如果交换机出现故障，则聚合会使用其他交换机继续为其数据链路提供连接。因此，一个 DLMP 聚合可以使用多个交换机。

总的来说，DLMP 聚合支持以下功能：

- 聚合可跨越多个交换机。

- 没有什么交换机配置是必需的或必须对交换机执行的。
- 如果仅使用特定类型支持的选项，可以通过使用 `dladm modify-aggr` 命令在中继聚合与 DLMP 聚合之间进行切换。

注 – 如果从中继聚合切换为 DLMP 聚合，则必须删除之前为中继聚合创建的交换机配置。

链路聚合的要求

链路聚合配置必须符合以下要求：

- 不必在要配置到聚合中的数据链路上配置 IP 接口。
- 聚合中的所有数据链路必须以相同的速度和全双工模式运行。
- 对于 DLMP 聚合，必须至少有一个交换机，以将聚合连接到其他系统的端口。配置 DLMP 聚合时不能使用背对背设置。
- 在基于 SPARC 的系统上，各个数据链路都必须有各自唯一的 MAC 地址。有关说明，请参阅《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何确保每个接口的 MAC 地址是唯一的”。

如 IEEE 802.3ad 链路聚合标准中所定义，设备必须支持链路状态通知，以让端口连接至聚合或与聚合分离。不支持链路状态通知的设备只能通过使用 `dladm create-aggr` 命令的 `-f` 选项进行聚合。对于此类设备，始终将链路状态报告为 `UP`（活动）。有关使用 `-f` 选项的信息，请参见第 22 页中的 “如何创建链路聚合”。

管理链路聚合

本节包含用于配置和管理链路聚合的不同过程。请注意，过程中的一些步骤是配置中继聚合和 DLMP 聚合的通用步骤。每种类型特有的步骤已明确指出。

- 第 22 页中的 “如何创建链路聚合”
- 第 24 页中的 “如何切换链路聚合类型”
- 第 25 页中的 “如何修改中继聚合”
- 第 26 页中的 “如何将链路添加到聚合”
- 第 26 页中的 “如何从聚合中删除链路”
- 第 27 页中的 “如何删除链路聚合”

▼ 如何创建链路聚合

开始之前

注 – 链路聚合仅对以相同速度运行的全双工点对点链路起作用。确保聚合中的接口符合此要求。

如果要在聚合拓扑中使用交换机并创建中继聚合，请确保在该交换机上执行了以下操作：

- 配置了要用作聚合的端口
- 以主动模式或被动模式配置 LACP（如果交换机支持 LACP）

这些先决条件不适用于 DLMP 聚合。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **为确定要聚合的数据链路，请显示数据链路信息。**

   ```
   # dladm show-link
   ```

3. **确保要配置到聚合中的数据链路未被任何应用程序打开。**

   例如，如果在该数据链路上创建了一个 IP 接口，请先删除该 IP 接口。

   a. **要确定链路是否被任何应用程序使用，请检查 `ipadm show-if` 命令的输出。**

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

      该输出指示数据链路 `net0` 上存在一个 IP 接口。

   b. **要删除 IP 接口，请键入以下命令：**

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

      其中，*interface* 指定链路上的 IP 接口。

4. **通过发出以下命令之一创建链路聚合：**

   - 要创建中继聚合，请发出以下命令：

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

     `-f` 强制创建聚合。当试图聚合不支持链路状态通知的设备时，使用此选项。

     `-P` *policy* 指定聚合的负载平衡策略。

`-L` *lacpmode*	指定 LACP（如果使用）的模式，可以为 `off`、`active` 或 `passive`。请参见第 19 页中的 “聚合 LACP 模式和交换机”。
`-T` *time*	指定 LACP 的时间。
`-u` *address*	指定聚合的固定单播地址。
`-l` *linkn*	指定要聚合的数据链路。
aggr	指定聚合的名称，可以是任意定制名称。有关用于指定名称的规则，请参阅《Oracle Solaris 11 联网介绍》中的“有效链路名称的规则”。

- 要创建 DLMP 聚合，请发出以下命令：

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

`-l` *linkn*	指定要聚合的数据链路。
aggr	指定聚合的名称。

5. **（可选）检查刚创建的聚合的状态。**

   ```
   # dladm show-aggr
   ```

   聚合的状态应该是 `UP`。

6. **执行聚合的进一步配置，例如创建 IP 接口、VNIC 等等。**

示例 2–1 创建中继聚合

本示例说明如何使用命令创建包含两个底层数据链路（net0 和 net1）的链路聚合。该聚合还配置为传送 LACP 包。该示例以删除底层数据链路上的现有 IP 接口开始。

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

# ipadm delete-ip net0
# ipadm delete-ip net1
# dladm create-aggr -L active -l net0 -l net1 aggr0

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

示例 2–2 创建 DLMP 聚合

本示例说明如何创建 DLMP 聚合。该聚合具有三个底层数据链路。

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

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

▼ 如何切换链路聚合类型

要在中继聚合与 DLMP 聚合之间切换聚合类型，请使用 `dladm modify-aggr` 命令修改聚合的模式。请注意，切换聚合的类型会更改整个配置。因此，与仅修改其他链路聚合属性相比，此过程对聚合的影响更广。

1 成为管理员。

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 确定链路聚合的当前类型。

```
# dladm show-aggr
```

输出的 `MODE` 字段指示聚合的当前类型。如果是中继聚合，则 `MODE` 值为 `standard`；如果是 DLMP 聚合，则值为 `haonly`。

3 将聚合切换为 DLMP 聚合。

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

其中，*mode* 为 `standard`（如果要切换为中继聚合）或 `haonly`（如果要切换为 DLMP 聚合）。

4 根据新的链路聚合类型的要求，对交换机进行调整。

5 （可选）显示链路聚合配置。

```
# dladm show-aggr
```

示例 2–3 从中继聚合切换为 DLMP 聚合

本示例说明如何将聚合从中继聚合更改为 DLMP 聚合。

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

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

接下来，将在交换机端删除先前适用于中继聚合的交换机配置。

▼ 如何修改中继聚合

此过程说明如何仅修改中继聚合的选定属性。DLMP 聚合不支持这些属性。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **修改聚合的策略。**

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

   policy 表示策略 L2、L3 和 L4 中的一个或多个，如第 18 页中的 “策略和负载平衡”中所述。

   aggr 指定要修改其策略的聚合。

3. **修改聚合的 LACP 模式。**

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

   `-L` *lacpmode* 指示运行聚合的 LACP 模式。值包括 `active`、`passive` 和 `off`。

   `-T` *time* 指示 LACP 计时器值（`short` 或 `long`）。

   aggr 指定要修改其策略的聚合。

示例 2–4 修改中继聚合

本示例说明如何将链路聚合 `aggr0` 的策略修改为 L2，然后打开主动 LACP 模式。

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

▼ 如何将链路添加到聚合

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **确保要添加的链路上没有激活的 IP 接口。**

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

其中，*interface* 是数据链路上配置的 IP 接口。

3 **将链路添加到聚合。**

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

其中，*link* 代表要添加到聚合中的数据链路，而 *aggr* 是聚合的名称。

4 **（仅针对中继聚合）如果链路聚合未配置 LACP，请重新配置交换机以容纳其他数据链路。**

参考交换机文档，在交换机上执行任何重新配置任务。

示例 2–5 将链路添加到聚合

本示例说明如何将链路添加到聚合 `aggr0`。

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

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

▼ 如何从聚合中删除链路

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **从聚合中删除链路。**

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

示例 2–6 从聚合中删除链路

本示例说明如何从聚合 `aggr0` 中删除链路。

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

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

▼ 如何删除链路聚合

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **删除在链路聚合上配置的 IP 接口。**

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

其中 *IP-aggr* 是链路聚合上的 IP 接口。

3 **删除链路聚合。**

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

示例 2–7 删除链路聚合

本示例说明如何删除聚合 `aggr0`。该删除具有永久性。

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

◆ ◆ ◆ 第 3 章

3 使用 VLAN

利用虚拟 LAN，可以将您的网络划分为子网，且无需添加到物理网络环境。因此，子网是虚拟的，您仍使用相同的物理网络资源。由于小型组更易于维护，因此，VLAN 使得网络管理更加容易。以下各节将更详细地讨论 VLAN 的功能及优点。

本章介绍配置和维护虚拟局域网 (virtual local area networks, VLAN) 的过程。本章包括以下几个主题：

部署 VLAN：概述

虚拟局域网 *(virtual local area network, VLAN)* 是在协议栈的数据链路层上对局域网的细分。可以为采用交换机技术的局域网创建 VLAN。通过将用户组指定给 VLAN，可以加强整个本地网络的网络管理和安全性。还可以将同一系统上的接口指定给不同的 VLAN。

以下各节简要概述了 VLAN。

何时使用 VLAN

如果您需要执行以下任务，请考虑将本地网络划分为 VLAN：

- 创建工作组的逻辑分区。

  例如，假定某楼层的所有主机都连接到一个基于交换的本地网络。可以为该楼层的每个工作组创建单独的 VLAN。
- 对各个工作组强制实施不同的安全策略。

例如，财务部和信息技术部的安全需求大不相同。如果这两个部门的系统共享同一本地网络，则可以为每个部门创建单独的 VLAN。然后，基于每个 VLAN 强制实施适当的安全策略。

- 将工作组拆分为易管理的广播域。

  使用 VLAN 可减小广播域的大小并提高网络效率。

VLAN 与定制名称

VLAN 演示了使用通用名称或定制名称的优势。在先前的发行版中，VLAN 通过物理连接点 (physical point of attachment, PPA)（需要将数据链路基于硬件的名称与 VLAN ID 组合在一起）进行标识。在 Oracle Solaris 11 中，可以选择更有意义的名称来标识 VLAN。名称必须符合《Oracle Solaris 11 联网介绍》中的“有效链路名称的规则”中提供的数据链路命名规则。例如，指定给 VLAN 的名称可以是 `sales0` 或 `marketing1`。

VLAN 名称与 VLAN ID 结合使用。局域网中的每个 VLAN 通过 VLAN ID（也称为 VLAN 标记）标识。VLAN ID 在 VLAN 配置过程中指定。配置交换机以支持 VLAN 时，需要为每个端口指定一个 VLAN ID。端口的 VLAN ID 必须与为连接到该端口的接口所指定的 VLAN ID 相同。

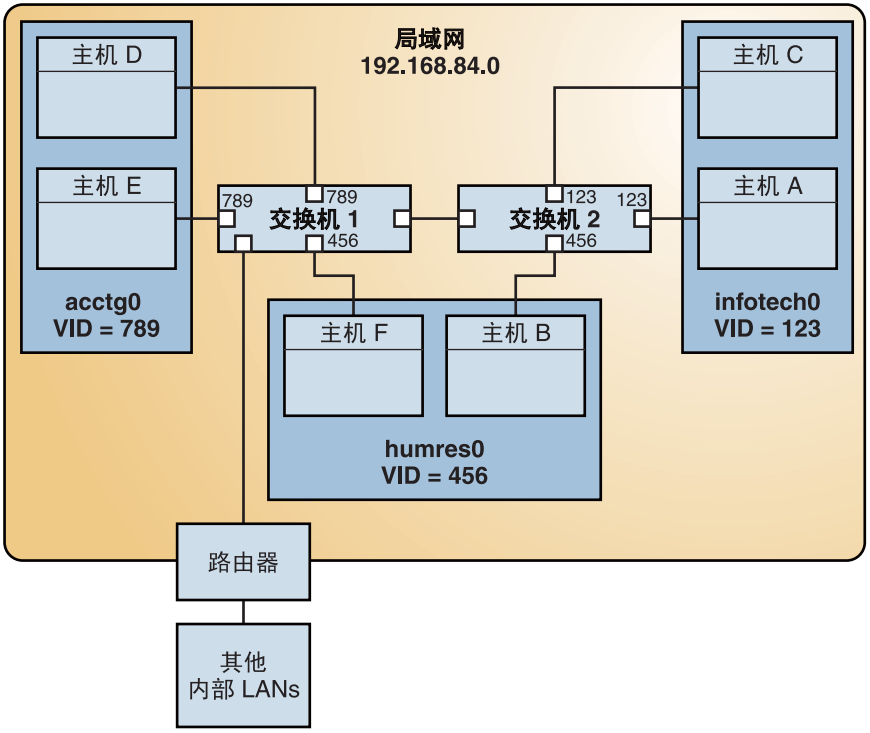
定制名称和 VLAN ID 的使用将在下一节进行说明。

VLAN 拓扑

使用交换 LAN 技术，可以将本地网络中的系统组织到 VLAN 中。将本地网络划分为 VLAN 之前，必须先获取支持 VLAN 技术的交换机。可以对交换机上的所有端口进行配置，使其为单个 VLAN 或多个 VLAN 提供服务，具体取决于 VLAN 拓扑设计。配置交换机端口的过程因交换机制造商而异。

下图显示了被划分为三个 VLAN 的局域网。

图 3–1 具有三个 VLAN 的局域网

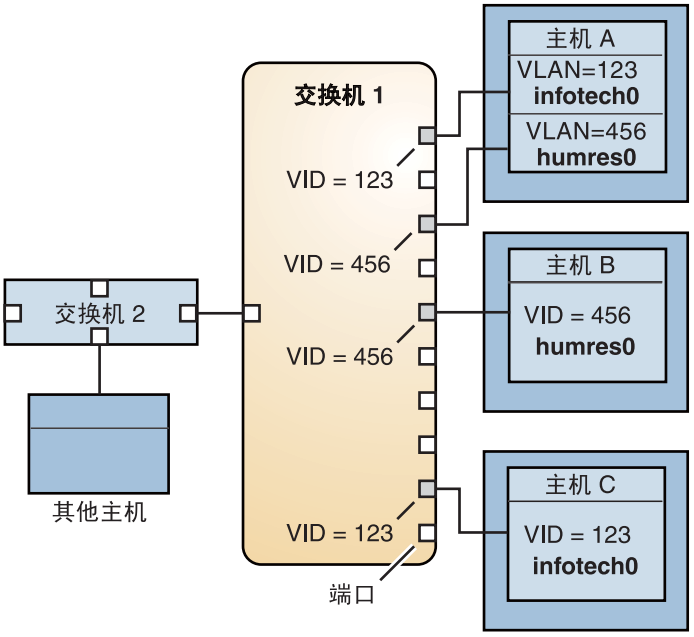


在图 3–1 中，LAN 的子网地址为 192.168.84.0。此 LAN 被细分为三个 VLAN，对应于三个工作组：

- VLAN ID 为 789 的 acctg0 — 会计组。此组拥有主机 D 和主机 E。
- VLAN ID 为 456 的 humres0 — 人力资源组。此组拥有主机 B 和主机 F。
- VLAN ID 为 123 的 infotech0 — 信息技术组。此组拥有主机 A 和主机 C。

图 3–2 显示了图 3–1 的一种变体，其中仅使用一个交换机，属于不同 VLAN 的多个主机均连接到此单个交换机。

图 3–2 包含 VLAN 的网络的交换机配置



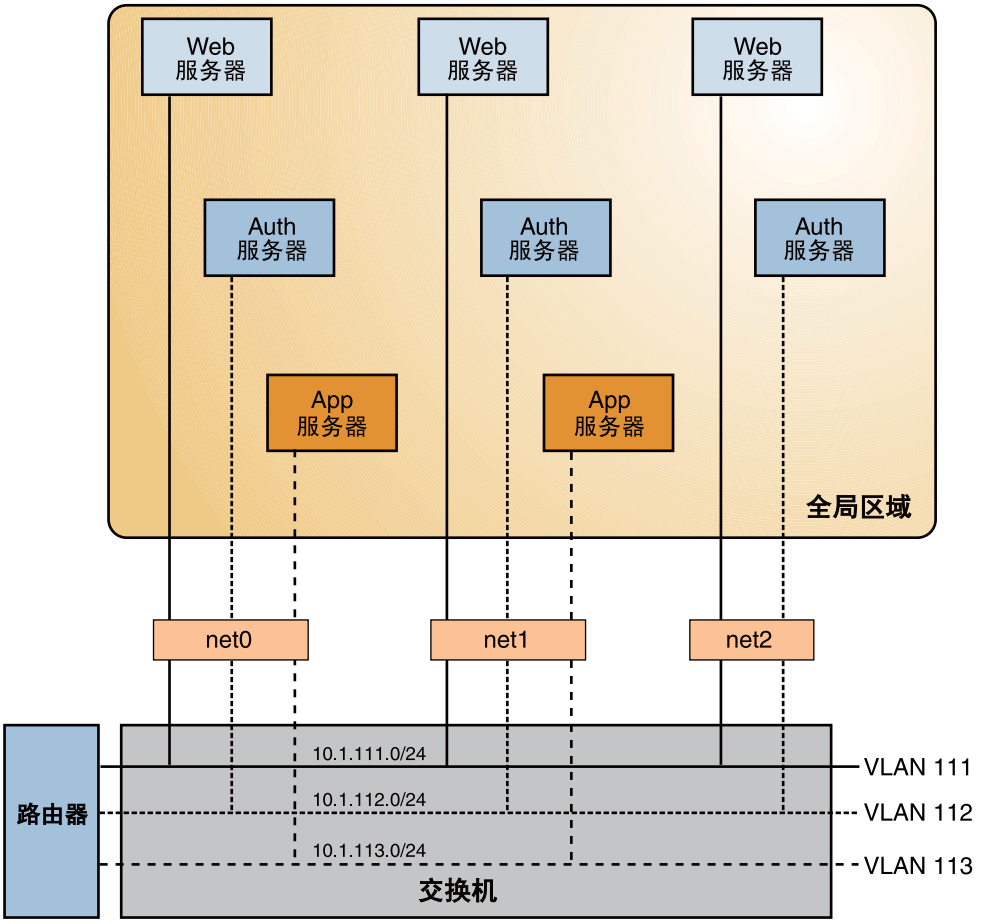
在图 3–2 中，主机 A 和主机 C 属于信息技术 VLAN，其 VLAN ID 为 123。因此，主机 A 的其中一个接口使用 VLAN ID 123 进行配置。此接口连接到交换机 1 上的端口 1，此端口也使用 VLAN ID 123 进行配置。主机 B 是人力资源 VLAN 的成员，其 VLAN ID 为 456。主机 B 的接口连接到交换机 1 上的端口 5，此端口使用 VLAN ID 456 进行配置。最后，主机 C 的接口使用 VLAN ID 123 进行配置。此接口连接到交换机 1 上的端口 9。端口 9 也使用 VLAN ID 123 进行配置。

图 3–2 还显示单个主机可以属于多个 VLAN。例如，主机 A 通过其接口配置了两个 VLAN。第二个 VLAN 使用 VLAN ID 456 进行配置并连接到端口 3（也使用 VLAN ID 456 进行配置）。因此，主机 A 同时是 `infotech0` 和 `humres0` 两个 VLAN 的成员。

使用 VLAN 和区域

通过将 VLAN 与 Oracle Solaris Zones 结合使用，可以在单个网络单元（如交换机）内配置多个虚拟网络。以具有三个物理网卡（ `net0`、`net1` 和 `net2`）的系统为例，如下图所示。

图 3–3 具有多个 VLAN 的系统



在没有 VLAN 情况下，您需要配置不同的系统来执行特定功能并将这些系统连接到单独的网络。例如，将 Web 服务器连接到一个 LAN，将验证服务器连接到另一个 LAN，并将应用服务器连接到第三个 LAN。使用 VLAN 和区域，您可以“折叠”所有八个系统并将它们作为区域配置在单个系统中。然后，使用 VLAN ID 将 VLAN 指定给执行相同功能的每组区域。图中提供的信息可列表如下。

功能	区域名称	VLAN 名称	VLAN ID	IP 地址	NIC
Web 服务器	webzone1	web1	111	10.1.111.0	net0
验证服务器	authzone1	auth1	112	10.1.112.0	net0
应用服务器	appzone1	app1	113	10.1.113.0	net0

功能	区域名称	VLAN 名称	VLAN ID	IP 地址	NIC
Web 服务器	webzone2	web2	111	10.1.111.0	net1
验证服务器	authzone2	auth2	112	10.1.112.0	net1
应用服务器	appzone2	app2	113	10.1.113.0	net1
Web 服务器	webzone3	web3	111	10.1.111.0	net2
验证服务器	authzone3	auth3	112	10.1.112.0	net2

要创建图中所示的配置，请参阅示例 3–1。

管理 VLAN

本节包含用于配置和管理 VLAN 的过程。

▼ 如何规划 VLAN 配置

1 **检查 LAN 拓扑，并确定在何处划分 VLAN 比较合适。**

有关此类拓扑的基本示例，请参阅图 3–1。

2 **创建 VLAN ID 的编号方案，并为每个 VLAN 指定 VLAN ID。**

注 – VLAN 编号方案可能已存在于网络中。如果是这样，则必须在现有的 VLAN 编号方案范围内创建 VLAN ID。

---

3 **在每个系统上，确定哪些接口将是特定 VLAN 的组件。**

a. **确定系统上配置了哪些接口。**

```
# dladm show-link
```

b. **确定将为系统上的各个数据链路关联哪个 VLAN ID。**

c. **创建 VLAN。**

4. **检查接口与网络交换机的连接。**
   记下每个接口的 VLAN ID 以及每个接口所连接到的交换机端口。

5. **使用与交换机连接的接口相同的 VLAN ID 配置交换机上的每个端口。**
   有关配置说明，请参阅交换机制造商的文档。

▼ 如何配置 VLAN

开始之前 本过程假定已在系统上创建了区域。本过程不包含创建区域以及为区域指定接口的步骤。有关区域配置的更多信息，请参阅《Oracle Solaris 11.1 管理：Oracle Solaris Zones、Oracle Solaris 10 Zones 和资源管理》中的第 17 章 “规划和配置非全局区域（任务）”。

1. **成为管理员。**
   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **确定在系统中使用的链路的类型。**

   ```
   # dladm show-link
   ```

3. **在数据链路上创建一个 VLAN 链路。**

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

   link 指定正在其上创建 VLAN 接口的链路。

   vid 表示 VLAN ID 号。

   vlan-link 指定 VLAN 的名称，也可以是通过管理行为选择的名称。

4. **验证 VLAN 配置。**

   ```
   # dladm show-vlan
   ```

5. **在 VLAN 上创建 IP 接口。**

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

   其中 *interface* 使用 VLAN 名称。

6. **使用一个 IP 地址配置 IP 接口。**

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

示例 3–1 配置 VLAN

本示例说明如何创建图 3–3 所示的 VLAN 配置。本示例假定您已在系统中配置不同的区域。有关配置区域的更多信息，请参见《Oracle Solaris 11.1 管理： Oracle Solaris Zones、Oracle Solaris 10 Zones 和资源管理》中的第 II 部分, “Oracle Solaris Zones”。

管理员首先检查可用于配置 VLAN 的可用链路，然后在特定链路上创建 VLAN。

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

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

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

当显示链路信息时，列表中包含 VLAN。

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

接下来，管理员将 VLAN 指定给其各自的区域。指定 VLAN 后，将针对各区域显示与以下内容类似的信息：

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

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

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

属性 physical 的值表示为给定区域设置的 VLAN。

接下来，管理员登录到每个非全局区域以配置 VLAN 的 IP 地址。

在 webzone1 中：

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

在 webzone2 中：

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

在 webzone3 中：

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

在 authzone1 中：

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

在 authzone2 中：

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

在 authzone3 中：

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

在 appzone1 中：

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

在 appzone2 中：

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

配置完所有 VLAN 的 IP 地址后，配置过程即完成。三个 VLAN 可以运行并可以承载其各自区域的通信。

▼ 如何通过链路聚合配置 VLAN

与通过接口配置 VLAN 的方式一样，也可以基于链路聚合创建 VLAN。第 2 章，使用链路聚合中介绍了链路聚合。本节综合介绍了如何配置 VLAN 和链路聚合。

1. **列出系统上配置的链路聚合。**

   ```
   # dladm show-link
   ```

2. **对于要在选定链路聚合上创建的每个 VLAN，发出以下命令：**

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

   link 指定正在其上创建 VLAN 接口的链路。在此过程中，链路指链路聚合。

   vid 表示 VLAN ID 号

   vlan-link 指定 VLAN 的名称，也可以是通过管理行为选择的名称。

3. **对于上一步中创建的每个 VLAN，创建该 VLAN 上的 IP 接口。**

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

   其中 *interface* 使用 VLAN 名称。

4. **为 VLAN 上的各个 IP 接口配置有效的 IP 地址。**

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

示例 3–2 通过链路聚合配置多个 VLAN

在此示例中，基于链路聚合配置了两个 VLAN。为 VLAN 指定的 VLAN ID 分别为 193 和 194。

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

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

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

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

▼ 如何在传统设备上配置 VLAN

某些传统设备只处理最大传输单元 (maximum transmission unit, MTU) 大小（也称为帧大小）为 1514 字节的包。帧大小超出此最大限制的包将被丢弃。对于这种情况，请按照第 35 页中的 “如何配置 VLAN”中列出的过程执行操作。但是，当创建 VLAN 时，请使用 `-f` 选项强制创建 VLAN。

1. **使用 `-f` 选项创建 VLAN。**

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

   link 指定正在其上创建 VLAN 接口的链路。在此过程中，链路指传统设备。

   vid 表示 VLAN ID 号

   vlan-link 指定 VLAN 的名称，也可以是通过管理行为选择的名称。

2. **为最大传输单元 (maximum transmission unit, MTU) 设置一个较低的大小，如 1496 字节。**

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

   较低的 MTU 值为链路层提供了在传输之前插入 VLAN 头的空间。

3. **执行与步骤 2 相同的步骤，为 VLAN 中各节点的 MTU 大小设置相同的较低值。**

   有关更改链路属性值的更多信息，请参阅《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“基本 dladm 命令”。

显示 VLAN 信息

由于 VLAN 是数据链路，因此可以使用 `dladm show-link` 命令查看有关 VLAN 的信息。但是，对于特定于 VLAN 的信息，需要使用 `dladm show-vlan` 命令。

以下示例比较了通过这两个命令所获取的信息类型。第一个输出为使用 `dladm show-link` 命令后的输出，显示了系统上的所有数据链路，包括非 VLAN 的数据链路。第二个输出为使用 `dladm show-vlan` 命令后的输出，显示了仅与 VLAN 相关的数据链路信息子集。

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

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

修改 VLAN

通过使用 `dladm modify-vlan` 命令，可以按照以下方式修改 VLAN：

- 更改 VLAN 的 VLAN ID
- 将 VLAN 迁移到另一个底层链路

要更改 VLAN 的 VLAN ID，请使用以下命令之一：

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

  在此命令中，*vid* 指定要分配给 VLAN 的新 VLAN ID。*Datalink* 指在其上配置 VLAN 的底层链路。如果数据链路上仅存在单个 VLAN，则可以使用此命令语法。如果对配置了多个 VLAN 的数据链路使用此命令语法，则命令会失败，因为数据链路上的 VLAN 必须具有唯一的 VLAN ID。

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

  使用此命令可更改单个数据链路上多个 VLAN 的唯一 VLAN ID。数据链路上的每个 VLAN 具有唯一的 VLAN ID。因此，一次只能更改一个 VLAN ID。在图 3–3 中，假定您想更改在 `net0` 上配置的 `web1`、`auth1` 和 `app1` 的 VLAN ID。要更改这些 VLAN ID，应继续执行以下命令：

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

您可以将 VLAN 从一个底层数据链路迁移到另一个底层数据链路，而无需删除和重新配置 VLAN。底层链路可以是物理链路、链路聚合或 etherstub。有关 etherstub 的更多信息，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》中的“网络虚拟化组件”。

要成功迁移 VLAN，VLAN 要移动到的底层数据链路必须能够接纳此 VLAN 的数据链路属性。如果不支持这些属性，则迁移将会失败并通知用户。成功迁移后，如果 VLAN 仍然连接到网络，使用此 VLAN 的所有应用程序将继续正常运行。

迁移 VLAN 后，某些与硬件相关的属性可能会改变。例如，VLAN 始终与其底层数据链路共享相同的 MAC 地址。因此，迁移 VLAN 后，VLAN 的 MAC 地址将更改为目标数据链路的主 MAC 地址。其他可能会受影响的属性包括数据链路状态、链路速度、MTU 大小等等。但是，应用程序不间断地继续运行。

注 – 迁移后的 VLAN 不保留任何原始数据链路的硬件通道统计信息。目标数据链路上 VLAN 的可用硬件通道将成为统计信息的新来源。不过，`dlstat` 命令缺省显示的软件统计信息将会保留。

您可以全局性地或有选择性地执行 VLAN 迁移。全局迁移意味着将某个数据链路上的所有 VLAN 迁移到另一个数据链路。要执行全局迁移，只需指定源数据链路和目标数据链路。以下示例将 `ether0` 上的所有 VLAN 移动到 `net1`：

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

其中

- `-L` 指在其上配置 VLAN 的原始数据链路。
- `-l` 指 VLAN 要迁移到的目标数据链路。

注 – 目标数据链路必须在源数据链路之前指定。

要执行有选择性的 VLAN 迁移，需要指定要移动的 VLAN。以下示例基于图 3–3，将 VLAN 从 `net0` 移动到 `net3`。

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

注 – 有选择性地迁移 VLAN 时，请省略 `-L` 选项，此选项仅适用于全局迁移。

执行迁移时，可以更改 VLAN 的 VLAN ID。以下示例基于图 3–3，说明了如何同时迁移多个 VLAN 并更改其 VLAN ID。

```
# 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     -----
```

注 – 并行子命令 `dladm modify-vnic` 用于迁移配置为 VLAN 的 VNIC。必须根据要迁移的是 VLAN 还是配置为 VLAN 的 VNIC，使用正确的子命令。对于可通过 `dladm show-vlan` 子命令显示的 VLAN，使用 `modify-vlan` 子命令。对于可通过 `dladm show-vnic` 子命令显示的 VNIC（包括那些具有 VLAN ID 的 VNIC），使用 `modify-vnic` 子命令。要修改 VNIC，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》中的“网络虚拟化组件”。

删除 VLAN

使用 `dladm delete-vlan` 命令删除系统上的 VLAN 配置。

注 – 必须首先删除要删除的 VLAN 上的所有现有 IP 配置，才能删除此 VLAN。如果 VLAN 上存在 IP 接口，VLAN 删除将会失败。

示例 3–3 删除 VLAN 配置

要删除 VLAN 配置，可执行与以下示例类似的步骤：

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

示例 3–3 删除 VLAN 配置 （续）

```
app2      113     net1      ----
web3      111     net2      ----
auth3     113     net2      ----

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

使用案例：结合使用链路聚合与 VLAN 配置

本节提供一个示例，说明如何创建使用链路聚合和 VLAN（在其上创建 IP 接口）的网络配置组合。可在以下位置找到介绍其他网络方案的文章：http://www.oracle.com/us/sun/index.htm。

在以下示例中，使用四个 NIC 的系统必须配置为针对八个单独子网的路由器。为了实现这一目标，将配置八个链路，分别用于每个子网。首先，在所有四个 NIC 上创建一个链路聚合。此不带标记的链路将成为缺省路由指向的网络的缺省不带标记的子网。

然后，在链路聚合上为其他子网配置 VLAN 接口。子网的命名基于一种颜色编码方案。因此，VLAN 名称也采用类似命名方式以对应其各自的子网。最终配置包括分别针对八个子网的八个链路：一个不带标记的链路，以及七个带标记的 VLAN 链路。该示例首先验证数据链路上是否已经存在 IP 接口。必须先删除这些接口，才能将数据链路组合为聚合。

管理员首先删除数据链路上已配置的任何 IP 接口。

```
# 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
```

然后，管理员创建链路聚合 `default0`。

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

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

```
net3        phys      1500  up       --         --
default0    aggr      1500  up       --         net0 net1 net2 net3
```

接下来，管理员在 `default0` 上创建 VLAN。

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

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

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

最后，管理员在 VLAN 链路上创建 IP 接口并为这些接口分配 IP 地址。

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

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

◆ ◆ ◆ 第 4 章

4 管理桥接网络（任务）

本章介绍桥接网络及如何管理桥接网络。本章包含以下主题：

桥接概述

网桥用于连接不同的网段，是两个节点之间的通路。不同网段通过网桥连接后，连接的网段进行通信时，就如同是一个网段一样。桥接是在网络栈的数据链路层 (L2) 中实现的。网桥使用包转发机制将子网连接在一起。

虽然桥接和路由都可用于分发关于网络中资源位置的信息，但它们在多个方面存在区别。路由在 IP 层 (L3) 实现并使用路由协议。数据链路层不使用路由协议，而通过检查连接到网桥的链路中接收的网络通信流量确定转发包的目的地。

收到包后，将检查其源地址。包从某个节点被发送到接收该包的链路，包的源地址与该节点相关联。此后，当接收的包使用同一地址作为目标地址时，网桥将包通过链路转发到该地址。

与源地址关联的链路可能是连接到桥接网络中另一个网桥的中间链路。随着时间的推移，桥接网络中所有网桥都将“学会”通过哪个链路向给定的节点发送包。因此，包的目标地址用于通过逐跳桥接的方式将包导向其最终目的。

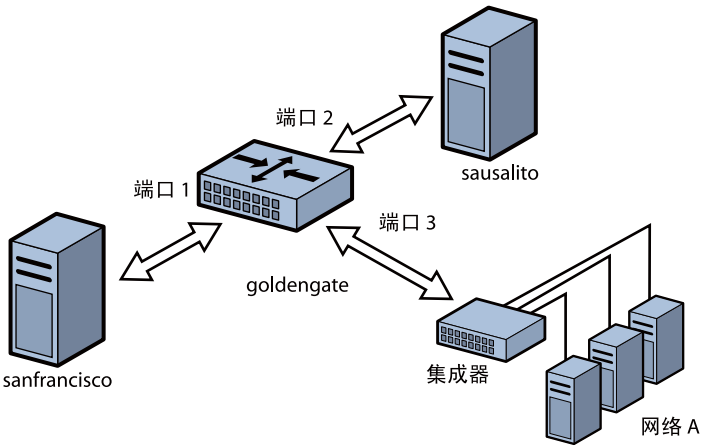
本地“链路断开”通知指示给定的链路上的所有节点都不再可访问。在此情况下，到该链路的包转发将停止，而通过该链路的所有转发条目将被清除。随时间的推移，还将清除早期转发条目。当恢复一个链路时，此链路上接收的包被视为新包。基于包源地址的“学习”过程再次开始。通过此过程，当该地址用作目标地址时，网桥能够通过该链路正确转发包。

要将包转发到目的地，网桥必须以混杂模式侦听连接到该网桥的每个链路。以混杂模式侦听导致网桥容易发生转发循环，发生转发循环时，包不停地以全线速循环。因此，桥接使用生成树协议 (Spanning Tree Protocol, STP) 机制防止网络循环使子网不可用。

除了对网桥使用 STP 和快速生成树协议 (Rapid Spanning Tree Protocol, RSTP) 之外，Oracle Solaris 还支持多链路透明互连 (Transparent Interconnection of Lots of Links, TRILL) 保护增强。缺省情况下使用 STP，但您可以通过为桥接命令指定 `-P trill` 选项来使用 TRILL。

通过将网络中的各个节点连接到一个单独的网络，使用网桥配置简化了对这些节点的管理。采用通过网桥连接这些区段的方式，所有节点都可共享单个广播网络。因此，通过使用网络协议（如 IP）而不是路由器来跨网段转发通信，每个节点都可以访问其他节点。如果不使用网桥，就必须配置 IP 路由以允许节点之间的 IP 通信转发。

图 4–1 简单的桥接网络

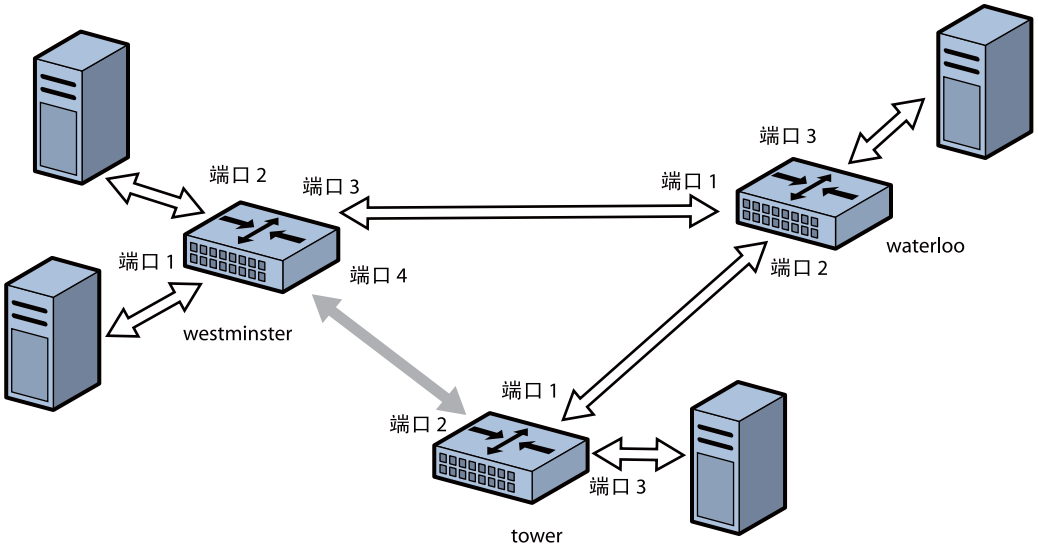


上图显示了一个简单的桥接网络配置。网桥 `goldengate` 是一个已配置了桥接的 Oracle Solaris 系统。系统 `sanfrancisco` 和 `sausalito` 物理连接到该网桥。网络 A 使用一个集线器，该集线器一侧物理连接到网桥，另一侧连接到计算机系统。网桥端口是链路，例如 `bge0`、`bge1` 和 `bge2`。

桥接的网络可以形成环，将多个网桥物理连接在一起。这种配置在网络中很常见。这种类型的配置可能导致因旧包无休止地在环中循环而使网络链路饱和的问题。为了防止出现这种循环情况，Oracle Solaris 网桥实现 STP 和 TRILL 协议。请注意，大多数硬件网桥还实现 STP 循环保护。

下图显示配置成一个环的桥接网络。此配置显示三个网桥。两个系统物理连接到 `westminster` 网桥。一个系统物理连接到 `waterloo` 网桥。一个系统物理连接到 `tower` 网桥。网桥通过网桥端口彼此物理连接。

图 4–2 桥接的网络环



当 STP 或 RSTP 用于循环保护时，通过防止循环中的一个连接转发包，降低了出现物理循环的风险。图 4–2 显示 `westminster` 和 `tower` 网桥之间的物理链路不用于转发包。

请注意，因为采用关闭可用物理链路来执行循环保护的方式，STP 和 RSTP 会消耗您的带宽。

与 STP 和 RSTP 不同，TRILL 不通过关闭物理链路来防止出现循环。相反，TRILL 计算网络中的每个 TRILL 节点的最短路径信息，并使用该信息将包转发到各个目的地。

因此，TRILL 使系统可以保持**所有**链路始终处于使用状态。循环不是问题，因为采用了 IP 处理循环的方式来处理循环，即 TRILL 根据需要创建路由并使用转发跳数限制来避免出现导致瞬时循环状态的问题。

注意 – 请勿在 SPARC 平台上设置 `local-mac-address?=false`。如果这样做，系统会错误地对同一个网络中的多个端口使用同一 MAC 地址。

注意 – 如果要求最高级别的性能，请**不要**将链路配置到网桥中。桥接**要求**底层接口处于混杂模式下，而该模式会禁用对硬件、驱动程序和系统其他层的多种重要优化。禁用这些性能增强功能是桥接机制不可避免的结果。

如果系统中的**某些**链路不是桥接的，因而不会受到这些限制，您就可以在该系统中使用网桥。这些性能问题只影响那些被配置为网桥的一部分的链路。

有关 STP 的信息，请参见 IEEE 802.1 D-1998。有关 RSTP 的信息，请参见 IEEE 820.1Q-2004。有关 TRILL 的信息，请参见 Internet 工程任务组 (Engineering Task Force, IETF) TRILL 草稿 (`http://tools.ietf.org/wg/trill`)。

链路属性

以下链路属性可以通过 `dladm show-linkprop` 命令查看并通过 `dladm set-linkprop` 和 `reset-linkprop` 命令修改：

`default_tag` 为通过某链路发送或接收的不带标记的包定义缺省虚拟局域网 (virtual local area network, VLAN) ID。有效值从 `0` 到 `4094`。缺省值为 `1`。只有非 VLAN 和非虚拟网络接口卡 (network interface card, VNIC) 类型的链路具有此属性。将此值设置为 `0` 将禁用以该端口为转发目的地或源的不带标记的包的转发。（这是一个 MAC 属性。）

注 – 此属性还在桥接范围之外使用，用于指定链路的 IEEE 端口 VLAN 标识符 (port VLAN identifier, PVID)。当 `default_tag` 非零时，不能在链路上创建具有相同 ID 的 VLAN，因为基本链路本身自动表示 PVID。

例如，如果 PVID 在 `net0` 上设置为 `5`，则无法在 `net0` 上创建 ID 为 `5` 的 VLAN。在这种情况下，要指定 VLAN `5`，请使用 `net1`。

不能将 `default_tag` 设置为与该链路上创建的任何现有 VLAN 的 ID 相同。例如，以下命令在 `net0` 上创建 VLAN 22：

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

在这种情况下，不能将 `default_tag` 设置为 `22`，因为这将使 `net0` 和 `myvlan0` 表示同一个 VLAN。

通过将 `default_tag` 设置为 `0`，您可以使 `net0` 上不带标记的包与任何 VLAN 都不关联。这样可以防止此类包被已配置的网桥转发。

`forward`	启用和禁用通过网桥的通信转发。除了 VNIC 链路外的所有链路都具有此属性。有效值是 `1` (true) 和 `0` (false)。缺省值为 `1`。禁用通信转发时，与链路实例关联的 VLAN 不会通过网桥转发通信。禁用转发相当于传统网桥中的从“允许的组”删除 VLAN。这意味着从本地客户端到底层链路的基于 VLAN 的 I/O 将继续，但不执行基于网桥的转发。
`stp`	启用和禁用 STP 和 RSTP。有效值是 `1` (true) 和 `0` (false)。缺省值是 `1`，这将启用 STP 和 RSTP。当该属性设置为 `0` 时，链路不使用 STP 或 RSTP，并始终处于转发模式下。转发模式使用网桥协议数据单元（Bridge Protocol Data Unit, BPDU）保护。当您要配置连接到结束节点的点对点链路时，请禁用 STP 和 RSTP。只有非 VLAN 和非 VNIC 类型的链路才具有此属性。
`stp_cost`	表示与使用此链路对应的 STP 和 RSTP 成本值。有效值介于 `1` 至 `65535` 之间。缺省值为 `0`，用于指示成本由链路类型自动计算。以下值分别表示几种链路类型的成本：`100` 对应 10 兆位/秒，`19` 对应 100 兆位/秒，`4` 对应 1 千兆位/秒，`2` 对应 10 千兆位/秒。
`stp_edge`	指定该端口是否连接到其他网桥。有效值是 `1` (true) 和 `0` (false)。缺省值为 `1`。如果该属性设置为 `0`，守护进程会假定端口连接到其他网桥（即使检测不到任何类型的 BPDU）。
`stp_p2p`	指定连接模式类型。有效值为 `true`、`false` 和 `auto`。缺省值是 `auto`，它会自动发现点对点连接。指定 `true` 可强制采用点对点模式。指定 `false` 可强制采用正常多点模式。
`stp_priority`	设置 STP 和 RSTP 端口优先级值。有效值介于 `0` 至 `255` 之间。缺省值为 `128`。STP 和 RSTP 端口优先级值用于附加到 PVID 之前来确定网桥的首选根端口。该数值越低，优先级越高。

STP 守护进程

使用 `dladm create-bridge` 命令创建的各网桥被表示为 `svc:/network/bridge` 的同名服务管理工具 (Service Management Facility, SMF) 实例。每个实例运行实现 STP 的 `/usr/lib/bridged` 守护进程的一个副本。

例如，以下命令创建名为 `pontevecchio` 的网桥：

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

系统将创建名为 `svc:/network/bridge:pontevecchio` 的 SMF 服务实例和名为 `/dev/net/pontevecchio0` 的可观测节点。

出于安全目的，缺省情况下所有端口都运行标准 STP。不运行某种形式的桥接协议（如 STP）的网桥，可能会在网络中形成不终结的转发循环。因为以太网包上没有跃点计数或晶体管-晶体管逻辑 (transistor-to-transistor logic, TTL)，所以任何此类循环对网络来说都是致命的。

如果特定端口未连接到另一网桥（例如，由于该端口有到主机系统的直接点对点连接），您可以从管理角度禁用该端口的 STP。即使网桥上的所有端口都禁用了 STP，STP 守护进程也仍然运行。守护进程继续运行的原因如下：

- 处理添加的任何新端口
- 实现 BPDU 保护
- 根据需要在端口上启用或禁用转发

当端口已禁用 STP 时，`bridged` 守护进程会继续侦听 BPDU（BPDU 保护）。此守护进程使用 `syslog` 标记所有错误并在端口上禁用转发来指示严重错误的网络配置。当链路从不可用重新变为可用，或者将链路手动删除再重新添加时，链路将重新启用。

如果禁用网桥的 SMF 服务实例，随着 STP 守护进程的停止，这些端口上的网桥转发也将停止。如果重新启动该实例，STP 将从初始状态启动。

TRILL 守护进程

使用 `dladm create-bridge` 命令创建的各网桥表示为 `svc:/network/bridge` 和 `svc:/network/routing/trill` 的同名 SMF 实例。`svc:/network/routing/trill` 的每个实例运行实现 TRILL 协议的 `/usr/lib/trilld` 守护进程的一个副本。

例如，以下命令创建名为 `bridgeofsighs` 的网桥：

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

系统将创建两个分别名为 `svc:/network/bridge:bridgeofsighs` 和 `svc:/network/routing/trill:bridgeofsighs` 的 SMF 服务。另外，系统将创建名为 `/dev/net/bridgeofsighs0` 的可观测节点。

调试网桥

为每个网桥实例指定一个**可观测节点**，此节点显示在 `/dev/net/` 目录下，用网桥名称加后缀 `0` 的形式命名。

可观测节点用于与 `snoop` 和 `wireshark` 实用程序配合使用。此节点行为类似于标准的以太网接口，不同的是，此节点传递包时会无提示丢弃。不能在可观测节点的顶部激活 IP，也不能执行绑定请求 (`DL_BIND_REQ`)，除非您使用被动选项。

使用该选项时，可观测节点会为每个由用户可用的网桥处理的包生成一个未经修改的副本，用于于监视和调试。此行为类似于传统网桥上的监视端口，并遵循普通的数据链路提供者接口 (data link provider interface, DLPI) 混杂模式规则。您可以使用 `pfmod` 命令或者 `snoop` 和 `wireshark` 实用程序中的功能基于 VLAN ID 对包进行过滤。

传送的包表示网桥接收的数据。

注 – 如果在桥接过程中添加、删除或修改了 VLAN 标记，`dlstat` 命令显示的数据将描述此过程发生之前的状态。如果不同的链路中使用不同的 `default_tag` 值（这种情况极为少见），可能会造成混乱。

要查看特定链路（桥接过程完成后）传送和接收的包，请在单个链路上（而不是在网桥的可观测节点上）运行 `snoop`。

您也可以使用 `dlstat` 命令获取有关网络包如何使用链路上的网络资源的统计信息。有关信息，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》中的第 4 章 “监视 Oracle Solaris 中的网络通信和资源使用情况”。

使用网桥时链路的行为如何变化

以下各节描述当网络配置中使用网桥时，链路的行为将如何变化。

有关标准链路行为的信息，请参见第 29 页中的 “部署 VLAN： 概述”。

DLPI 行为

下面介绍启用网桥时链路行为的差异：

- 链路接通 (`DL_NOTE_LINK_UP`) 和链路关闭 (`DL_NOTE_LINK_DOWN`) 通知是作为一个整体提供的。这意味着当所有的外部链路均显示链路关闭状态时，使用 MAC 层的较高级别客户机也将看到链路关闭事件。当网桥上的任何外部链路显示链路接通状态时，所有较高级别的客户机将看到链路接通事件。

  执行此整体链路接通和链路关闭报告的原因如下：

  - 当显示为链路关闭状态时，此链路上的节点将不再可访问。这种情况在桥接代码仍可以通过另一个链路发送和接收包时是不真实的。需要链路实际状态的管理应用程序可以使用现有的 MAC 层内核统计信息来显示状态。这些应用程序与普通的客户机（如 IP）不同，它们报告硬件状态信息，但不参与转发包。
  - 当所有外部链路处于关闭状态时，显示的状态就仿佛网桥本身被关闭。在这种特殊情况下，系统将认为不可访问任何节点。折中办法是在所有接口为“真实的”（非虚拟的）并且所有链路全部断开的情况下，网桥不能用于允许仅本地通信。

- 所有链路特定的功能都是通用功能。支持特殊的硬件加速功能的链路无法使用这些硬件加速功能，因为实际的输出链路不是完全由客户端确定的。转发功能的网桥必须基于目标 MAC 地址选择输出链路，此输出链路可以是网桥上的任何链路。

管理桥接网络上的 VLAN

缺省情况下，系统中配置的 VLAN 在网桥实例的所有端口之间转发包。如果底层链路是网桥的一部分，则调用 `dladm create-vlan` 或 `dladm create-vnic -v` 命令时该命令还将启用此网桥链路上指定 VLAN 的包转发。

要在一个链路上配置 VLAN 并禁用通过网桥上其他链路发送或接收的包转发，就必须通过使用 `dladm set-linkprop` 命令设置 VLAN 的 `forward` 属性来禁用转发。

当底层链路配置为网桥的一部分时，使用 `dladm create-vlan` 命令会自动启用桥接的 VLAN。

在符合标准的 STP 中，VLAN 将被忽略。桥接协议通过使用无标记 BPDU 消息只计算一个无环拓扑，并使用此树拓扑启用和禁用链路。必须配置网络中预分配的所有重复链路，以使配置的 VLAN 在 STP 自动禁用这些重复链路时不会断开连接。这意味着您必须在桥接主干中的所有位置运行所有 VLAN 或仔细检查所有冗余链路。

TRILL 协议不遵循复杂的 STP 规则。相反，TRILL 自动封装有完整 VLAN 标记的包并通过网络传递它们。这意味着如果在一个桥接网络内重用了相同的 VLAN ID，TRILL 会将分离的 VLAN 绑定。

这一与 STP 的重要区别意味着您可以在网络的各不同部分中重用 VLAN 标记来管理大于 4094 限制的 VLAN 集。虽然不能使用 TRILL 以这种方式管理网络，但您可能能够实现其他解决方案，例如基于提供者的 VLAN。

在具有 VLAN 的 STP 网络中，当 STP 禁用了“错误的”链路时，可能很难配置故障转移特征来防止 VLAN 分隔开。分离的 VLAN 的功能损失相对较小，与 TRILL 模型的稳健相比是次要的。

VLAN 行为

网桥通过检查允许的 VLAN 集和每个链路的 `default_tag` 属性来执行包转发。一般过程如下所示：

1. **确定输入 VLAN。**链路中的系统收到传入包时，此过程即开始。收到包时，将检查包的 VLAN 标记。如果该标记不存在，或标记只用于优先级（设置为零），会将该链路上配置的 `default_tag` 属性值（如果未设置为零）用作内部 VLAN 标记。如果该标记不存在或为零，并且 `default_tag` 为零，将忽略该包。将执行不带标记的转发。如果该标记存在且等于 `default_tag` 值，也将忽略该包。否则，该标记被视为代表传入 VLAN 包。

2. **检查链路成员身份。**如果输入 VLAN 未配置为此链路上的允许 VLAN，将忽略该包。然后计算转发，并对输出链路进行相同的检查。
3. **更新标记。**如果 VLAN 标记（此时非零）等于输出链路上的 `default_tag` 值，将删除包上的标记（如果有），而不管优先级如何。如果 VLAN 标记不等于输出链路上的 `default_tag` 值，且当前没有标记，则添加一个标记，并为传出包设置该标记，并将当前优先级复制到该包。

注 – 在转发向多个接口（用于广播、多播和未知目标）发送包的情况下，必须对每个输出链路独立执行输出链路检查和标记更新。某些传送可能已加标记，而有些则未标记。

查看网桥配置

以下示例显示如何查看有关网桥配置和桥接服务的信息：

- 可通过运行以下命令查看有关网桥的信息：

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

- 可通过运行以下命令查看网桥的 TRILL 昵称信息：

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

管理网桥

在 Oracle Solaris 中，使用 `dladm` 命令和 SMF 功能管理网桥。可以使用 SMF 命令通过实例 `svc:/network/bridge` 的故障管理资源标识符 (fault-managed resource identifier, FMRI) 来启用、禁用和监视网桥实例。可以使用 `dladm` 命令创建或销毁网桥，以及将链路指定给网桥或从网桥中删除链路。

管理网桥（任务列表）

下表显示了可用于管理网桥的任务。

任务	说明	参考
查看有关配置的网桥的信息。	使用 `dladm show-bridge` 命令查看系统中配置的网桥的信息。可以查看有关配置的网桥、链路、统计和内核转发条目的信息。	第 54 页中的 “如何查看配置的网桥的信息”
查看已连接到网桥的链路的配置信息。	使用 `dladm show-link` 命令查看系统中配置的链路的信息。如果链路已关联到网桥，请查看 `BRIDGE` 字段的输出。	第 56 页中的 “如何查看关于网桥链路的配置信息”
创建网桥。	使用 `dladm create-bridge` 命令创建网桥并添加可选的链路。 缺省情况下，网桥是使用 STP 创建的。要改为使用 TRILL 创建网桥，请将 `-P trill` 添加到 `dladm create-bridge` 命令行。或者，使用 `dladm modify-bridge` 命令启用 TRILL。	第 56 页中的 “如何创建网桥”
修改网桥的保护类型。	使用 `dladm modify-bridge` 命令修改网桥的保护类型。	第 57 页中的 “如何修改网桥的保护类型”
向网桥添加链路。	使用 `dladm add-bridge` 命令将一个或多个链路添加到现有的网桥。	第 57 页中的 “如何向现有网桥中添加一个或多个链路”
从网桥删除链路。	使用 `dladm remove-bridge` 命令从网桥删除链路。直到删除了网桥的所有链路后，才能删除网桥。	第 58 页中的 “如何从网桥删除链路”
从系统中删除网桥。	使用 `dladm delete-bridge` 命令从系统删除网桥。	第 58 页中的 “如何从系统中删除网桥”

▼ 如何查看配置的网桥的信息

此过程说明如何配合不同选项使用 `dladm show-bridge` 命令来显示所配置的网桥的各种信息。

有关 `dladm show-bridge` 命令选项的更多信息，请参见 `dladm(1M)` 手册页。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **查看某个网桥或所有已配置的网桥的信息。**

- 查看网桥的列表。

  ```
  # dladm show-bridge
  ```

- 显示与链路相关的网桥状态。

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

- 显示网桥的统计信息。

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

  注 – 报告的网桥统计信息的名称和定义可能会有变化。

- 显示网桥的与链路相关的统计信息。

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

- 显示网桥的内核转发条目。

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

- 显示网桥的 TRILL 信息。

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

示例 4–1 查看网桥信息

以下是配合不同选项使用 `dladm show-bridge` 命令的示例。

- 以下命令显示有关系统中配置的所有网桥的信息：

  ```
  # 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
  ```

- 以下命令显示单个网桥实例 `tower` 的与链路相关的状态信息。要查看已配置的属性，请使用 `dladm show-linkprop` 命令。

  ```
  # 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
  ```

- 以下命令显示指定网桥 `terabithia` 的统计信息：

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

- 以下命令显示指定网桥 `london` 上所有链路的统计信息：

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

- 以下命令显示指定网桥 `avignon` 的内核转发条目：

```
# 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
```

- 以下命令显示指定网桥 key 的 TRILL 信息：

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

▼ 如何查看关于网桥链路的配置信息

dladm show-link 输出中包含 BRIDGE 字段。如果链路是网桥成员，此字段标识该链路所属的网桥的名称。缺省情况下显示该字段。对于不属于网桥的链路，如果使用 -p 选项，此字段为空。否则，该字段显示 -- 。

网桥的可观测节点也作为一个单独的链路显示在 dladm show-link 输出中。对于此节点，现有 OVER 字段会列出那些是网桥成员的链路。

1 成为管理员。

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 查看是网桥成员的任何链路的相关配置信息。

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

-p 选项生成可解析的格式的输出。

▼ 如何创建网桥

此过程说明如何使用 STP（缺省协议）创建网桥。有关网桥创建选项的更多信息，请参见 dladm(1M) 手册页中的 dladm create-bridge 命令说明。

注 – 要使用 TRILL 创建网桥，请向 dladm create-bridge 命令添加 -P trill。或者，使用 dladm modify-bridge 命令启用 TRILL。

---

dladm create-bridge 命令创建一个网桥实例，并可以选择将一个或多个网络链路指定给该新网桥。因为缺省情况下系统中不存在网桥实例，所以，缺省情况下 Oracle Solaris 不在网络链路之间创建网桥。

要在链路之间创建网桥，必须创建至少一个网桥实例。每个网桥实例是独立的。网桥之间并不包括转发连接，一个链路至多是一个网桥的成员。

网桥名称可以是任意字符串，但必须是合法的 SMF 服务实例名称。此名称是没有转义序列的 FMRI 组成部分，这意味着不能包含空格、ASCII 控制字符和以下字符：

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

保留名称 `default`，同时保留所有以 `SUNW` 字符串开头的名称。保留具有数字后缀的名称以创建用于调试的可观测设备。由于可观测设备的使用，进一步将合法的网桥实例名称约束为合法的 `dlpi` 名称。该名称必须以字母字符或下划线字符开始和结束。其余的名称可以包含字母数字和下划线字符。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **创建网桥。**

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

`-l` *link* 选项将链路添加到网桥。如果无法添加任何指定的链路，该命令将失败，且不会创建网桥。

以下示例说明如何通过连接 `net0` 和 `net1` 链路创建 `brooklyn` 网桥：

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

▼ 如何修改网桥的保护类型

此过程说明如何使用 `dladm modify-bridge` 命令将保护类型从 STP 修改为 TRILL，或从 TRILL 修改为 STP。

- **修改网桥的保护类型。**

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

`-P` *protection-type* 选项指定要使用的保护类型。缺省情况下，保护类型是 STP (`-P stp`)。要使用 TRILL 保护类型，请使用 `-P trill` 选项。

以下示例显示如何将 `brooklyn` 网桥的保护类型从缺省的 STP 改为 TRILL：

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

▼ 如何向现有网桥中添加一个或多个链路

此过程说明如何向网桥实例添加一个或多个链路。

一个链路只能是一个网桥的成员。因此，如果要将链路从一个网桥实例移动到另一网桥实例，必须先从当前网桥删除该链路，然后再将其添加到另一网桥。

指定给网桥的链路不能是 VLAN、VNIC 或隧道。仅可将被视为聚合的一部分的链路或本身就是聚合的链路指定给网桥。

指定给同一网桥的链路必须具有相同的 MTU 值。请注意，Oracle Solaris 允许您更改现有链路的 MTU 值。不过，网桥实例将进入维护状态，直到您删除或更改了指定的链路，以便重新启动网桥之前 MTU 值匹配。

指定给网桥的链路必须是以太网类型，其中包括 802.3 和 802.11 介质。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **向现有网桥添加新链路。**

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

以下示例说明如何将 net2 链路添加到现有网桥 rialto：

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

▼ 如何从网桥删除链路

此过程说明如何从网桥实例中删除一个或多个链路。如果要删除网桥，请按此过程进行。只有先删除网桥的所有链路后，才能将网桥删除。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **从网桥删除链路。**

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

以下示例说明如何从网桥 charles 删除 net0、net1 和 net2 链路：

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

▼ 如何从系统中删除网桥

此过程说明如何删除网桥实例。必须先通过运行 dladm remove-bridge 命令取消激活所有已连接的链路，然后才能删除网桥。请参见第 58 页中的 “如何从网桥删除链路”。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 从系统中删除网桥。

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

以下示例说明如何先从 coronado 网桥删除 net0、net1 和 net2 链路，再从系统删除该网桥本身：

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

◆ ◆ ◆ 第 5 章

IPMP 介绍

IP 网络多路径 (IP network multipathing, IPMP) 是第 3 层技术，通过该技术可以将多个 IP 接口分组到一个逻辑接口。通过故障检测、透明访问故障转移和包负荷分配等功能，IPMP 确保网络始终对系统可用，从而提高了网络性能。

本章包含以下主题：

注 – 在本章和第 6 章，管理 IPMP（任务）涉及 IPMP 说明的所有内容中，对术语**接口**的所有引用均特指 *IP* **接口**。除非限定条件明确指示了术语的不同用途，如网络接口卡 (network interface card, NIC)，否则该术语始终指在 IP 层上配置的接口。

Oracle Solaris 中的 IPMP

Oracle Solaris 中的 IPMP 包含以下功能：

- 通过 IPMP，可以将多个 IP 接口配置到一个组（称为 IPMP 组）中。IPMP 组及其多个底层 IP 接口作为一个整体表示为单个 *IPMP* **接口**。对此接口的处理与对网络栈的 IP 层上的其他任何接口一样。所有 IP 管理任务、路由表、地址解析协议 (Address Resolution Protocol, ARP) 表、防火墙规则和其他与 IP 相关的过程均通过引用 IPMP 接口来使用 IPMP 组。
- 系统负责处理数据地址在各底层活动接口间的分发。创建 IPMP 组时，数据地址作为一个地址池属于对应的 IPMP 接口。然后，内核会自动将数据地址随机绑定到组的底层活动接口。

- `ipmpstat` 工具是用于获取有关 IPMP 组的信息的主要工具。此命令提供有关 IPMP 配置的所有方面的信息，例如组的底层 IP 接口、测试地址和数据地址、所使用的故障检测的类型，以及哪些接口已经出现故障。`ipmpstat` 的功能、您可使用的选项以及每个选项生成的输出都在第 92 页中的 “监视 IPMP 信息”中描述。
- 可以为 IPMP 接口指定一个定制名称，以便更容易识别 IPMP 组。请参见第 77 页中的 “配置 IPMP 组”。

使用 IPMP 的益处

多种因素可导致接口不可用，例如接口出现故障或接口处于脱机状态以进行维护。如果没有 IPMP，便无法再使用与不可用的接口相关联的任何 IP 地址联系系统。而且，使用这些 IP 地址的现有连接也会中断。

通过 IPMP，可以将多个 IP 接口配置到一个 *IPMP* **组**中。组在功能上类似于使用数据地址发送或接收网络通信流量的 IP 接口。如果组中的一个底层接口出现故障，数据地址会在组中的其余底层活动接口之间重新分配。因此，尽管某个接口出现故障，组仍能保持网络连接。使用 IPMP 时，只要组中至少有一个接口可用，网络连接就始终可用。

通过自动在 IPMP 组的一组接口中分配传出网络通信流量，IPMP 提高了总体网络性能。此过程称为传出**负荷分配**。系统还通过为应用程序未指定其 IP 源地址的包执行源地址选择，间接控制传入负荷分配。但是，如果应用程序明确选择了 IP 源地址，则系统不会改变该源地址。

注 – 链路聚合执行与 IPMP 类似的功能以提高网络性能和可用性。有关这两种技术的比较，请参见附录 B，链路聚合和 IPMP：功能比较。

用于使用 IPMP 的规则

IPMP 组的配置取决于您的系统配置。

使用 IPMP 时，请遵守以下规则：

- 同一 LAN 上的多个 IP 接口必须配置到一个 IPMP 组中。LAN 广泛地指各种本地网络配置，包括 VLAN 以及其节点属于同一链路层广播域的有线和无线本地网络。

  注 – 不支持同一个链路层 (L2) 广播域上存在多个 IPMP 组。L2 广播域通常对应于特定子网。因此，对于每个子网，只能配置一个 IPMP 组。

- IPMP 组的底层 IP 接口不能跨越不同的 LAN。

例如，假定具有三个接口的系统已连接到两个单独的 LAN。其中两个 IP 接口连接到一个 LAN，而另一个 IP 接口连接到另一个 LAN。在这种情况下，根据第一条规则的要求，连接到第一个 LAN 的两个 IP 接口必须配置为一个 IPMP 组。根据第二条规则，连

接到第二个 LAN 的单个 IP 接口不能成为该 IPMP 组的成员。该单个 IP 接口不需要 IPMP 配置。但是，您可以将该单个接口配置到一个 IPMP 组中以监视该接口的可用性。单接口 IPMP 配置在第 64 页中的 “IPMP 接口配置的类型”中进一步讨论。

考虑另一种情况，其中第一个 LAN 的链路包含三个 IP 接口，另一个链路包含两个接口。此设置需要配置两个 IPMP 组：一个三接口组连接到第一个 LAN，一个两接口组连接到第二个 LAN。

IPMP 组件

以下是 IPMP 软件组件：

- **多路径守护进程** `in.mpathd` – 检测接口故障并修复。如果为底层接口配置了测试地址，该守护进程将同时执行基于链路的故障检测和基于探测器的故障检测。根据使用的故障检测方法的类型，该守护进程在接口上设置或清除相应标志，以指示该接口是出现故障还是已修复。作为一个选项，还可以配置该守护进程以监视所有接口（包括未配置为属于 IPMP 组的接口）的可用性。有关故障检测的说明，请参见第 70 页中的 “IPMP 中的故障检测”。

  `in.mpathd` 守护进程还控制 IPMP 组中的活动接口指定。该守护进程试图保持创建 IPMP 组时最初配置的活动接口的数量。因此，`in.mpathd` 会根据需要激活或取消激活底层接口，以符合管理员配置的策略。有关 `in.mpathd` 守护进程如何管理底层接口激活的更多信息，请参阅第 64 页中的 “IPMP 的工作原理”。有关该守护进程的更多信息，请参阅 `in.mpathd(1M)` 手册页。

- **IP 内核模块** – 通过将 IPMP 组中可用的一组 IP 数据地址分配给组中可用的一组底层 IP 接口，管理传出负荷分配。该模块还将执行源地址选择以管理传入负荷分配。该模块的两个角色都可以提高网络通信性能。

- **IPMP 配置文件** (`/etc/default/mpathd`) – 用于定义守护进程的行为。

  应定制该文件以设置以下参数：

  - 运行基于探测器的故障检测时要探测的目标接口
  - 探测目标以检测故障的持续时间
  - 用于在出现故障的接口修复后标记该接口的状态
  - 要监视的 IP 接口范围，是否也包括系统中未配置为属于 IPMP 组的 IP 接口

  有关修改配置文件的过程，请参阅第 90 页中的 “如何配置 IPMP 守护进程的行为”。

- `ipmpstat` **命令** – 全面提供了有关 IPMP 状态的各种不同类型的信息。该工具还显示有关每个 IPMP 组的底层 IP 接口的其他信息，以及已为该组配置的数据地址和测试地址。有关此命令的更多信息，请参见第 92 页中的 “监视 IPMP 信息”和 `ipmpstat(1M)` 手册页。

IPMP 接口配置的类型

一个 IPMP 配置通常由同一系统上连接到同一 LAN 的两个或更多物理接口组成。这些接口可以属于一个 IPMP 组并采用以下配置之一：

- **活动/活动配置** – 所有底层接口都处于活动状态的 IPMP 组。**活动接口**是当前可供 IPMP 组使用的 IP 接口。

  注 – 缺省情况下，当您将一个底层接口配置为属于某个 IPMP 组时，该接口将变为活动状态。

- **活动/备用配置** – 至少有一个接口出于管理目的被配置为**备用接口**的 IPMP 组。虽然备用接口处于空闲状态，但多路径守护进程仍会监视该接口以跟踪其可用性，具体取决于该接口的配置方式。如果该接口支持链路故障通知，则使用基于链路的故障检测。如果为该接口配置了测试地址，则还将使用基于探测器的故障检测。如果一个活动接口出现故障，会自动根据需要部署备用接口。您可以根据需要，为一个 IPMP 组配置任意数量的备用接口。

也可以将一个接口单独配置在一个 IPMP 组中。单接口 IPMP 组的行为与具有多个接口的 IPMP 组相同。然而，这种 IPMP 配置不提供网络通信的高可用性。如果底层接口出现故障，系统将完全丧失发送或接收通信的能力。配置单接口 IPMP 组的目的是通过使用故障检测监视接口的可用性。通过在接口上配置测试地址，多路径守护进程可以使用基于探测器的故障检测跟踪该接口。

通常，单接口 IPMP 组配置与具有更强的故障转移功能的其他技术（例如，Oracle Solaris Cluster 软件）结合使用。系统可以持续监视底层接口的状态，而 Oracle Solaris Cluster 软件提供的功能可确保发生故障时网络的可用性。有关 Oracle Solaris Cluster 软件的更多信息，请参见《Oracle Solaris Cluster Concepts Guide》。

无底层接口的 IPMP 组也可以存在，例如，其底层接口已被删除的一个组。该 IPMP 组未被破坏，但该组不能用于发送和接收通信。当组的底层接口被置于联机状态时，IPMP 接口的数据地址将被分配给这些接口，系统继续承载网络通信。

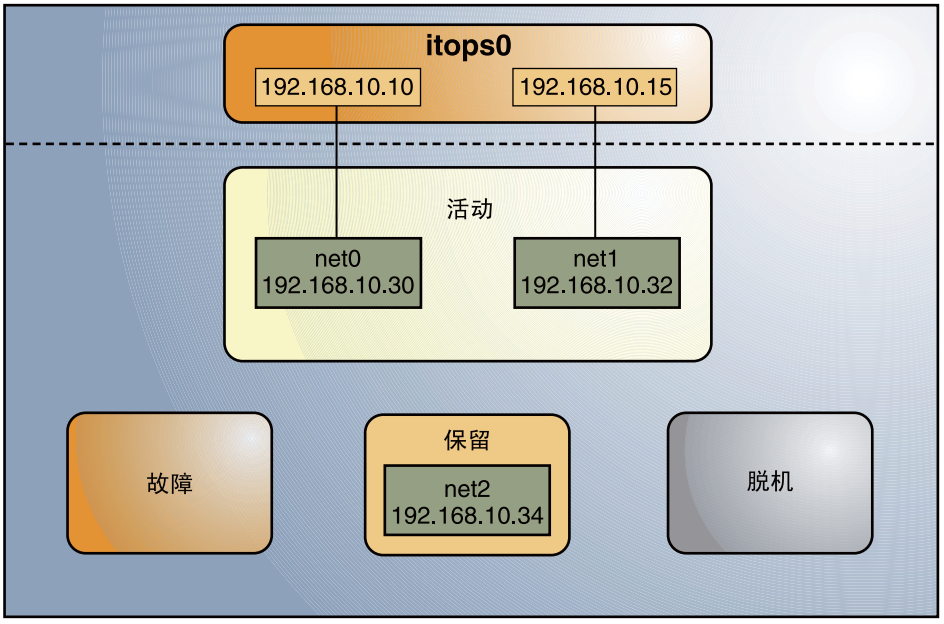
IPMP 的工作原理

IPMP 试图保留与创建 IPMP 组时最初配置的数量相同的活动和备用接口，从而保持网络可用性。

IPMP 故障检测可以基于链路，也可以基于探测器，或同时基于两者，以确定组中特定底层 IP 接口的可用性。如果 IPMP 确定某底层接口出现故障，则该接口被标记为出现故障，并且不再可用。然后，与故障接口相关联的数据 IP 地址被重新分配给组中另一个能正常工作的接口。还会部署一个备用接口（如果可用）以保持活动接口的原始数量。

以一个使用活动/备用配置的三接口 IPMP 组 `itops0` 为例，如下图所示。

图 5–1 IPMP 活动/备用配置



IPMP 组 `itops0` 的配置如下所示：

- 两个数据地址指定给组 `192.168.10.10` 和 `192.168.10.15`。
- 两个底层接口配置为活动接口，并对其指定了灵活的链路名称：`net0` 和 `net1`。
- 该组有一个备用接口，同样具有灵活的链路名称 `net2`。
- 使用了基于测试器的故障检测，因此为活动接口和备用接口配置了测试地址，如下所示：
  - `net0`：`192.168.10.30`
  - `net1`：`192.168.10.32`
  - `net2`：`192.168.10.34`

注 – 图 5–1、图 5–2、图 5–3 和图 5–4 中的“活动”、“脱机”、“备用”和“故障”区域仅表示底层接口（而不是物理位置）的状态。此 IPMP 实现中没有发生接口或地址的物理移动或任何 IP 接口转移。这些区域只是为了显示在出现故障或修复后，底层接口是如何改变状态的。

可以将 ipmpstat 命令与不同的选项结合使用来显示有关现有 IPMP 组的特定类型的信息。有关其他示例，请参见第 92 页中的 “监视 IPMP 信息”。

以下 ipmpstat 命令显示有关图 5–1 中的 IPMP 配置的信息：

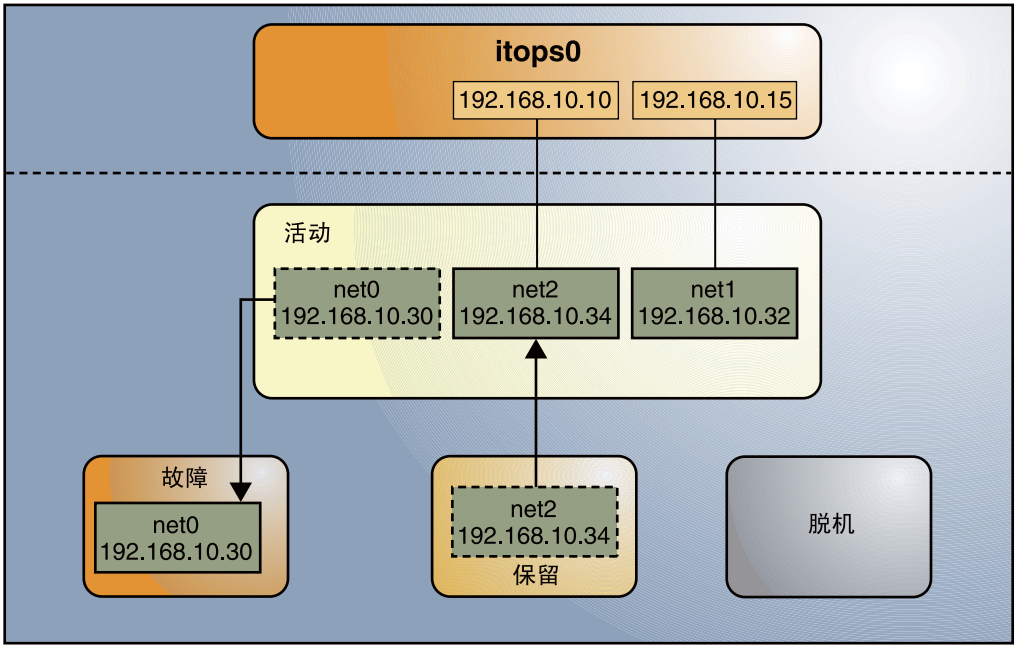
```
# ipmpstat -g
GROUP     GROUPNAME     STATE     FDT       INTERFACES
itops0    itops0        ok        10.00s    net1 net0 (net2)
```

要显示有关该组的底层接口的信息，请键入以下内容：

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

IPMP 通过管理底层接口以保留活动接口的原始数量来维护网络可用性。因此，如果 net0 出现故障，则会部署 net2 以确保该 IPMP 组仍有两个活动接口。net2 的激活如下图所示。

图 5–2 IPMP 中的接口故障



注 – 图 5–2 中数据地址与活动接口存在一对一对应关系，这仅是为了简化示意图。IP 内核模块可以随机指定数据地址，数据地址和接口之间不一定符合一对一关系。

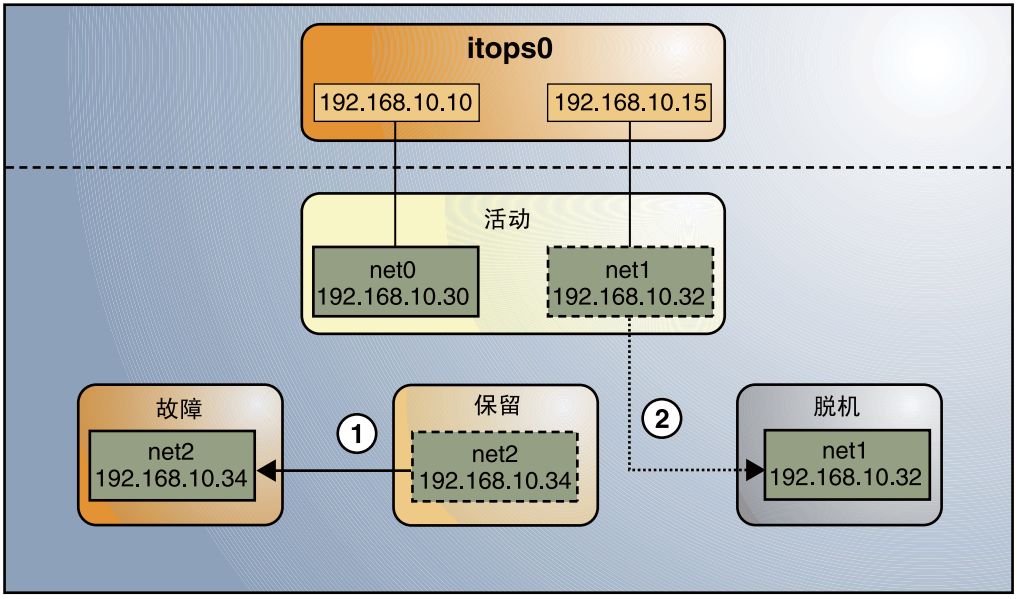
`ipmpstat` 命令显示了图 5–2 中的信息，如下所示：

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

在修复 `net0` 后，它的状态将恢复为活动接口。`net2` 也依次返回其原始的备用状态。

图 5–3 显示了一种不同的故障情形，其中备用接口 `net2` 出现故障 (1)。然后，管理员将一个活动接口 `net1` 置于脱机状态 (2)。结果，该 IPMP 组就只剩下一个正常工作的接口 `net0`。

图 5–3 IPMP 中的备用接口故障



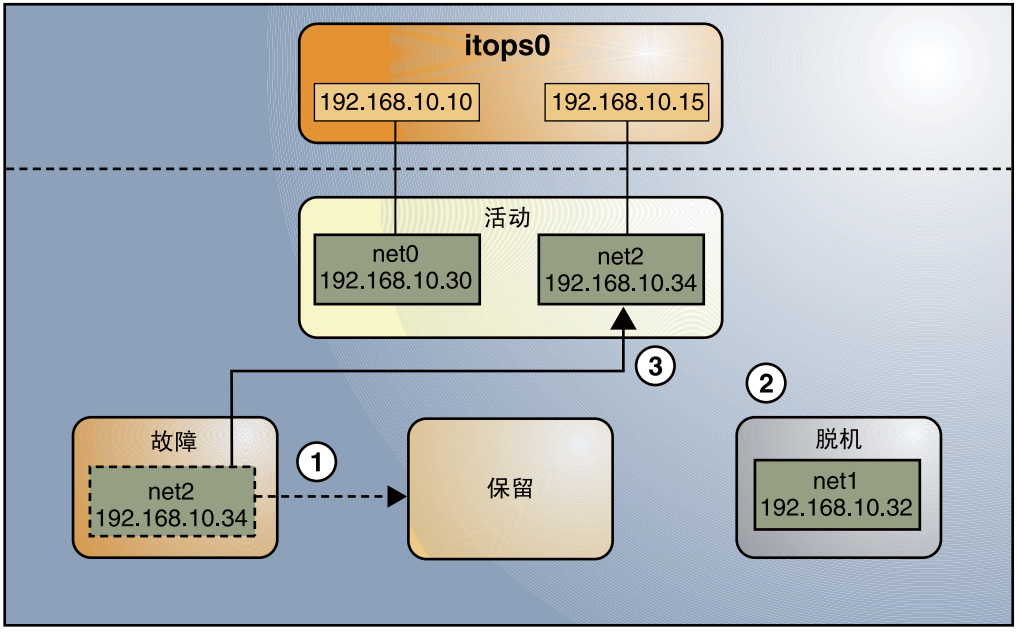
`ipmpstat` 命令显示了图 5–3 中的信息，如下所示：

```
# ipmpstat -i
INTERFACE   ACTIVE     GROUP      FLAGS       LINK        PROBE      STATE
net0        yes        itops0     -------     up          ok         ok
```

```
net1        no          itops0    --mb-d-     up          ok          offline
net2        no          itops0    is-----     up          failed      failed
```

对于此特定故障，接口修复后的恢复行为也有所不同。恢复过程取决于 IPMP 组的活动接口的原始数量与修复后的配置的对比情况。恢复过程以图形表示在下图中：

图 5–4 IPMP 恢复过程



在图 5–4 中，net2 在修复后通常会恢复到其原始状态，即备用接口 (1)。但是，IPMP 组仍不反映两个活动接口的原始数量，因为 net1 仍保持脱机状态 (2)。因此，IPMP 将 net2 部署为活动接口 (3)。

ipmpstat 命令显示了修复后的 IPMP 情形，如下所示：

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

如果一个同时配置为 FAILBACK=no 模式（在此模式下，出现故障的活动接口在修复后不会自动恢复为活动状态）的活动接口出现故障，会发生类似的恢复过程。假定图 5–2 中的 net0 配置为 FAILBACK=no 模式。在该模式下，修复后的 net0 会成为备用接口，尽管它原来是一个活动接口。接口 net2 仍保持活动以维持 IPMP 组的两个活动接口的原始数量。

ipmpstat 命令显示了恢复信息，如下所示：

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

有关这种类型的配置的更多信息，请参见第 73 页中的 “FAILBACK=no 模式”。

IPMP 寻址

可以在 IPv4 网络以及双栈 IPv4 和 IPv6 网络中配置 IPMP 故障检测。配置了 IPMP 的接口支持两种类型的地址，这两种地址在以下各节中进行了说明。从 Oracle Solaris 11 开始，IP 地址**仅**位于 IPMP 接口中，并被指定为数据地址，而测试地址位于底层接口中。

数据地址

数据地址是 DHCP 服务器在引导时动态分配给 IP 接口的或使用 ipadm 命令手动指定的传统 IPv4 和 IPv6 地址。数据地址指定给 IPMP 接口。标准的 IPv4 包通信（如果适用的话，还包括 IPv6 包通信）被视为**数据通信**。数据通信使用 IPMP 接口上承载的数据地址，并流经该 IPMP 接口或组的活动接口。

测试地址

测试地址是 in.mpathd 守护进程用于执行基于探测器的故障和修复检测的特定于 IPMP 的地址。测试地址可由 DHCP 服务器动态指定，也可使用 ipadm 命令手动指定。指定给 IPMP 组底层接口的只能是测试地址。当一个底层接口出现故障时，该接口的测试地址继续由 in.mpathd 守护进程用来进行基于探测器的故障检测，以检查该接口的后续修复。

注 – 仅当需要使用基于探测器的故障检测时，才需要配置测试地址。否则，您可以启用传递式探测来检测故障，而无需使用测试地址。有关使用或不使用测试地址进行基于探测器的故障检测的更多信息，请参阅第 70 页中的 “基于探测器的故障检测”。

在以前的 IPMP 实现中，必须将测试地址标记为 DEPRECATED 以免被应用程序使用（尤其是在接口出现故障时）。在当前的实现中，测试地址位于底层接口中。因此，不知晓 IPMP的应用程序不再能够意外地使用这些地址。然而，为了确保这些地址不被视为可能的数据包源，系统自动将具有 NOFAILOVER 标志的所有地址标记为 DEPRECATED。

可以将子网中的任何 IPv4 地址用作测试地址。由于 IPv4 地址是许多站点的有限资源，因此您可能希望将不可路由的 RFC 1918 专用地址用作测试地址。请注意，`in.mpathd` 守护进程与测试地址所在子网中的其他主机仅交换 ICMP 探测器。如果使用 RFC 1918 样式的测试地址，确保使用适当的 RFC 1918 子网中的地址配置网络上的其他系统（首选路由器）。然后 `in.mpathd` 守护进程便可以成功地与目标系统交换探测器。有关 RFC 1918 专用地址的更多信息，请参阅 RFC 1918, Address Allocation for Private Internets (`http://www.ietf.org/rfc/rfc1918.txt?number=1918`)。

唯一的有效 IPv6 测试地址是物理接口的链路本地地址。无需将单独的 IPv6 地址用作 IPMP 测试地址。IPv6 链路本地地址基于接口的介质访问控制 (Media Access Control, MAC) 地址。当引导时接口变为启用了 IPv6 的接口或通过 `ipadm` 命令手动配置接口时，将自动配置链路本地地址。

如果在 IPMP 组的所有接口上同时激活了 IPv4 和 IPv6，则无需配置单独的 IPv4 测试地址。`in.mpathd` 守护进程可以将 IPv6 链路本地地址用作测试地址。

IPMP 中的故障检测

为确保网络可持续用于发送或接收通信流量，IPMP 在 IPMP 组的底层 IP 接口上执行故障检测。出现故障的接口在修复之前不可用。其余活动接口继续工作，同时根据需要部署任何现有的备用接口。

`in.mpathd` 守护进程处理以下类型的故障检测：

- 基于探测器的故障检测，有两种类型：
  - 未配置测试地址（传递式探测）。
  - 配置了测试地址。
- 基于链路的故障检测（如果 NIC 驱动程序支持）

基于探测器的故障检测

基于探测器的故障检测包括使用 ICMP 探测器检查接口是否已经出现故障。此故障检测方法的实现取决于是否使用了测试地址。

使用测试地址的基于探测器的故障检测

这种故障检测方法涉及发送和接收使用测试地址的 ICMP 探测器消息。这些消息也称为**探测器通信**或**测试通信**，它们通过接口发送到同一本地网络上的一个或多个目标系统。`in.mpathd` 守护进程通过已为基于探测器的故障检测配置的所有接口分别探测所有目标。如果给定接口对五个连续的探测器未做出任何响应，则 `in.mpathd` 认为该接口已出现故障。探测速率取决于**故障检测时间** (failure detection time, *FDT*)。故障检测时间的缺省值是 10 秒。不过，您可以在 IPMP 配置文件中调整 FDT。有关说明，请转到第 90 页中的 “如何配置 IPMP 守护进程的行为”。

要优化基于探测器的故障检测，您必须将多个目标系统设置为接收来自 `in.mpathd` 守护进程的探测器。通过使用多个目标系统，您可以更好地确定报告的故障的性质。例如，唯一定义的目标系统没有响应，则表示故障可能在目标系统中，也可能在 IPMP 组的接口之一中。相比之下，如果几个目标系统中只有一个系统没有响应探测器，则故障可能在目标系统中，而不在 IPMP 组本身中。

`in.mpathd` 守护进程确定要动态探测哪些目标系统。首先，守护进程在路由表中搜索与 IPMP 组的接口相关联的测试地址所在子网中的目标系统。如果找到这样的目标，则守护进程使用它们作为探测目标。如果没有发现位于同一子网上的目标系统，则该守护进程将发送多播包以探测链路上的相邻主机。多播包将发送到所有主机多播地址（在 IPv4 中为 `224.0.0.1`，在 IPv6 中为 `ff02::1`），以确定要用作目标系统的主机。对回显包作出响应的前五个主机将被选作探测目标。如果守护进程找不到响应多播探测器的路由器或主机，则该守护进程将无法检测基于探测器的故障。在这种情况下，`ipmpstat -i` 命令将探测器状态报告为 `unknown`。

可以使用主机路由明确配置 `in.mpathd` 守护进程要使用的目标系统的列表。有关说明，请参阅第 88 页中的 “配置基于探测器的故障检测”。

不使用测试地址的基于探测器的故障检测

在没有测试地址的情况下，使用两种类型的探测器实现此方法：

- **ICMP 探测器**

  ICMP 探测器由 IPMP 组中的活动接口发送，用于探测在路由表中定义的目标。**活动**接口是可以接收发送到该接口的链路层 (L2) 地址的传入 IP 包的底层接口。ICMP 探测器使用数据地址作为探测器的源地址。如果 ICMP 探测器到达目标并从目标获得响应，则活动接口能正常工作。

- **传递式探测器**

  传递式探测器由 IPMP 组中的备用接口发送，用于探测活动接口。备用接口是不主动接收任何传入 IP 包的底层接口。

  以包含四个底层接口的一个 IPMP 组为例。对该组配置了一个数据地址，但没有配置测试地址。在此配置中，传出包可以使用所有底层接口。然而，传入包只能由绑定到该数据地址的接口接收。其余三个不能接收传入包的底层接口就是**备用**接口。

  如果备用接口可以成功地将探测器发送到活动接口并收到响应，则活动接口能正常工作，同时也推断出该备用接口发送了探测器。

注 – 在 Oracle Solaris 中，使用测试地址执行基于探测器的故障检测。要选择不使用测试地址的基于探测器的故障检测，必须手动启用传递式探测。有关过程，请参见第 89 页中的 “如何选择要使用的故障检测方法”。

组故障

当 IPMP 组中的所有接口同时出现故障时，则发生**组故障**。在这种情况下，没有可用的底层接口。此外，如果所有目标系统同时出现故障并且启用了基于探测器的故障检测，in.mpathd 守护进程会刷新其所有当前目标系统并探测新的目标系统。

在没有测试地址的 IPMP 组中，将可以探测活动接口的单一接口指定为探测器。此指定的接口会同时设置 FAILED 标志和 PROBER 标志。数据地址绑定到此接口，从而允许接口继续探测目标以检测恢复。

基于链路的故障检测

只要接口支持基于链路的故障检测，会始终启用该类故障检测。

要确定一个第三方接口是否支持基于链路的故障检测，请使用 ipmpstat -i 命令。如果给定接口的输出中的 LINK 列包含 unknown 状态，则该接口不支持基于链路的故障检测。参考制造商的文档，了解有关设备的更多特定信息。

支持基于链路的故障检测的网络驱动程序会监视接口的链路状态，并在该链路状态变化时通知网络子系统。收到更改通知后，网络子系统会根据需要设置或清除该接口的 RUNNING 标志。如果 in.mpathd 守护进程检测到接口的 RUNNING 标志已被清除，会立即使该接口失效。

故障检测和匿名组功能

IPMP 支持匿名组中的故障检测。缺省情况下，IPMP 只监视属于 IPMP 组的接口的状态。但是，可以将 IPMP 守护进程配置为同时跟踪不属于任何 IPMP 组的接口的状态。因此，这些接口被视为属于某个“匿名组”。当发出 ipmpstat -g 命令时，匿名组将显示双短划线 (--)。在匿名组中，接口的数据地址也用作测试地址。由于这些接口不属于任何指定的 IPMP 组，这些地址对应用程序是可见的。要启用对不属于 IPMP 组的接口的跟踪，请参见第 90 页中的 “如何配置 IPMP 守护进程的行为”。

检测物理接口修复

修复检测时间是故障检测时间的两倍。故障检测的缺省时间为 10 秒。因此，修复检测的缺省时间为 20 秒。在再次使用 RUNNING 标志标记出现故障的接口，并且故障检测方法检测到该接口已修复后，in.mpathd 守护进程将清除该接口的 FAILED 标志。根据管理员最初设置的活动接口的数量，重新部署已修复的接口。

当一个底层接口出现故障并且使用了基于探测器的故障检测时，in.mpathd 守护进程将使用指定的探测器（如果没有配置测试地址）或使用接口的测试地址继续探测。在接口修复期间，恢复过程取决于出现故障的接口的原始配置，如下所示：

- 如果出现故障的接口原来是活动接口，修复后的接口将恢复为其原始的活动状态。如果 IPMP 组中活动接口的数量足以达到系统管理员所定义的数量，则在故障期间充当“替补”的备用接口将切换回备用状态。

  注 – 一个例外是当修复的活动接口同时配置为 FAILBACK=no 模式时。有关更多信息，请参见第 73 页中的 “FAILBACK=no 模式”。

- 如果出现故障的接口原来是备用接口，则只要 IPMP 组反映活动接口的原始数量，修复后的接口就会恢复为其原始的备用状态。否则，备用接口将变为活动接口。

要查看 IPMP 在接口故障及修复过程中行为的图形表示，请参见第 64 页中的 “IPMP 的工作原理”。

FAILBACK=no 模式

缺省情况下，出现故障然后又被修复的活动接口将自动重新成为 IPMP 组中的活动接口。此行为由 in.mpathd 守护进程的配置文件中的 FAILBACK 参数值控制。但是，在数据地址重新映射到修复的接口时即使发生非关键性中断，对一些管理员来说可能也是不可接受的。这些管理员可能更希望允许激活的备用接口继续作为活动接口。IPMP 允许管理员覆盖缺省行为，以防止修复后的接口自动成为活动接口。这些接口必须在 FAILBACK=no 模式中配置。有关过程，请参见第 90 页中的 “如何配置 IPMP 守护进程的行为”。

当处于 FAILBACK=no 模式的活动接口出现故障然后又被修复后，in.mpathd 守护进程恢复 IPMP 配置的方式如下所示：

- 只要 IPMP 组反映活动接口的原始配置，守护进程就保留该接口的 INACTIVE 状态。
- 如果 IPMP 配置在修复时刻并不反映该组的活动接口的原始配置，则已修复的接口将作为活动接口重新部署，尽管它具有 FAILBACK=no 状态。

注 – FAILBACK=NO 模式是针对整个 IPMP 组设置的。它不是每个接口的可调参数。

IPMP 和动态重新配置

使用 Oracle Solaris 的动态重新配置 (dynamic reconfiguration, DR) 功能，可以在系统运行的同时重新配置系统硬件（如接口）。DR 只能在支持该功能的系统上使用。在支持 DR 的系统上，IPMP 集成到了重新配置协调管理器 (Reconfiguration Coordination Manager, RCM) 框架中。因此，您可以安全地连接、分离或重新连接 NIC 和 RCM 以管理系统组件的动态重新配置。例如，您可以连接、激活新接口，然后将其添加到现有的 IPMP 组。配置这些接口后，它们即可供 IPMP 使用。

对于要分离 NIC 的所有请求，首先需要进行检查以确保可以保持连接。例如，缺省情况下，无法分离不在 IPMP 组中的 NIC。也无法分离包含 IPMP 组中仅有的工作接口的 NIC。但是，如果必须移除系统组件，可以使用 `cfgadm` 命令的 `-f` 选项覆盖此行为，如 `cfgadm(1M)` 手册页中所述。

如果检查成功，`in.mpathd` 守护进程将为接口设置 `OFFLINE` 标志。接口上的所有测试地址都要取消配置。然后，从系统中取消激活 NIC。如果上述任一步骤失败，或者同一系统组件上其他硬件的 DR 失败，则会将先前的配置恢复到其初始状态。将显示有关此事件的状态消息。否则，分离请求成功完成。可以从系统中移除组件。未断开任何现有连接。

注 – 更换 NIC 时，确保所使用的卡为同一类型（例如以太网）。更换 NIC 之后，持久性 IP 接口配置将应用于该 NIC。

◆ ◆ ◆ 第 6 章

6 管理 IPMP（任务）

本章介绍在 Oracle Solaris 11 发行版中使用 IP 网络多路径 (IP network multipathing, IPMP) 管理接口组的任务。

本章包含以下主题：

注 – 本章中的任务介绍如何使用 `ipadm` 命令配置 IPMP，该命令替代了 Oracle Solaris 10 和以前发行版中使用的 `ifconfig` 命令。要了解有关这两个命令相互之间的对应关系的更多信息，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的附录 A “对应关系比较：ifconfig 和 ipadm 命令”。

Oracle Solaris 11 还为 IPMP 引入了新的概念模型。有关详细解释，请参见第 64 页中的 “IPMP 的工作原理”。

部署 IPMP 时维护路由

配置 IPMP 组时，IPMP 接口会继承其底层接口的 IP 地址以将它们用作数据地址。然后，底层接口接收 IP 地址 `0.0.0.0`。因此，如果特定 IP 接口随后添加到 IPMP 组，则使用这些接口定义的路由将会丢失。

配置 IPMP 时丢失路由通常涉及缺省路由并在 Oracle Solaris 安装过程中发生。在安装过程中，您需要定义一个缺省路由，为此您要使用系统中的一个接口，例如主接口。随后，您使用定义缺省路由时所用的相同接口配置 IPMP 组。配置 IPMP 后，系统将不能再路由网络包，因为该接口的地址已转换为 IPMP 接口。

要确保在使用 IPMP 时保留缺省路由，必须在不指定接口的情况下定义该路由。在这种方式下，任何接口（包括 IPMP 接口）都可用于进行路由。因此，系统可以继续路由通信。

注 – 本节使用主接口作为定义缺省路由的示例。然而，路由丢失情况适用于可用于进行路由的任何接口，该接口稍后会成为 IPMP 组的一部分。

▼ 如何在使用 IPMP 时定义路由

以下过程说明如何在配置 IPMP 时保留缺省路由。

1. **使用控制台登录到系统。**

   必须使用控制台执行此过程。如果使用 `ssh` 或 `telnet` 命令登录，则在执行后续步骤时连接将丢失。

2. **可选显示路由表中定义的路由。**

   ```
   # netstat -nr
   ```

3. **删除绑定到特定接口的路由。**

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

4. **在不指定接口的情况下添加路由。**

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

5. **可选显示重新定义的路由。**

   ```
   # netstat -nr
   ```

6. **可选如果路由表中的信息未更改，请重新启动路由服务，然后重新检查路由表中的信息以确保已正确重新定义路由。**

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

示例 6–1 为 IPMP 定义路由

本示例假定在安装过程中为 `net0` 定义了缺省路由。

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

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

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

配置 IPMP 组

本节提供了用于规划和配置 IPMP 组的过程。第 5 章，IPMP 介绍中的概述介绍了如何将 IPMP 组实现为接口。因此，在本章中，术语 *IPMP* **组**和 *IPMP* **接口**可互换使用。

- 第 77 页中的 “如何规划 IPMP 组”
- 第 79 页中的 “如何配置使用 DHCP 的 IPMP 组”
- 第 81 页中的 “如何手动配置活动/活动 IPMP 组”
- 第 82 页中的 “如何手动配置活动/备用 IPMP 组”

▼ 如何规划 IPMP 组

以下过程包括在配置 IPMP 组之前要收集的必要规划任务和信息。这些任务不必按顺序执行。

注 – 必须为每个子网或 L2 广播域只配置一个 IPMP 组。有关更多信息，请参见第 62 页中的 “用于使用 IPMP 的规则”。

---

1 **确定满足您的需要的常规 IPMP 配置。**

IPMP 配置取决于用于处理系统上承载的通信类型的网络要求。IPMP 将传出网络包分配到 IPMP 组的各个接口，从而提高了网络吞吐量。然而，对于给定的 TCP 连接，传入通信通常只使用一个物理路径，以尽量降低处理无序包的风险。

因此，如果您的网络处理大量传出通信，将多个接口配置到一个 IPMP 组中可以提高网络性能。相反，如果系统承载大量传入通信，则在组中配置大量接口不一定能通过对通信流量进行负载分配而提高性能。不过，有更多的底层接口有助于在接口出现故障时保证网络的可用性。

2 **验证组中的每个接口是否具有唯一的 MAC 地址。**

要为系统中的每个接口配置唯一的 MAC 地址，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何确保每个接口的 MAC 地址是唯一的”。

3 **确保在 IPMP 组的所有接口上配置并推送了同一组 STREAMS 模块。**

同一组中的所有接口必须按相同顺序配置相同的 STREAMS 模块。

a. **检查即将包含在 IPMP 组中的所有接口上 STREAMS 模块的顺序。**

您可以通过使用 `ifconfig` *interface* `modlist` 命令打印 STREAMS 模块的列表。例如，以下是 `net0` 接口的 `ifconfig` 输出：

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

如输出所示，接口通常作为网络驱动程序存在于 IP 模块的下方。这些接口不需要额外的配置。

然而，某些技术被作为 STREAMS 模块推送到 IP 模块和网络驱动程序之间。如果 STREAMS 模块是有状态的，即使将相同模块推送到组中的所有接口上，在进行故障转移时仍可能会出现意外行为。但是，只要按相同顺序将 STREAMS 模块推送到 IPMP 组中的所有接口上，就可以使用无状态 STREAMS 模块。

b. **按 IPMP 组的标准顺序推送每个接口的模块。**

例如：

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

4 **在 IPMP 组中的所有接口上使用相同的 IP 寻址格式。**

如果为一个接口配置了 IPv4，则必须为 IPMP 组中的所有接口配置 IPv4。例如，如果您为一个接口添加了 IPv6 寻址，则必须将 IPMP 组中的所有接口配置为支持 IPv6。

5 **确定您要实现的故障检测的类型。**

例如，如果您要实现基于探测器的故障检测，则必须在底层接口上配置测试地址。如需相关信息，请参见第 70 页中的 “IPMP 中的故障检测”。

6 **确保 IPMP 组中的所有接口都连接到同一本地网络。**

例如，您可以将同一 IP 子网上的以太网交换机配置到一个 IPMP 组中。您可以将任意数量的接口配置到一个 IPMP 组中。

注 – 您还可以配置一个只包含一个接口的 IPMP 组，例如，如果您的系统只有一个物理接口。如需相关信息，请参见第 64 页中的 “IPMP 接口配置的类型”。

---

7 **确保 IPMP 组不包含具有不同网络介质类型的接口。**

分组在一起的接口必须采用相同的接口类型。例如，不能将以太网接口和令牌环接口组合在一个 IPMP 组中。此外，不能将令牌总线接口与异步传输模式 (asynchronous transfer mode, ATM) 接口组合在同一 IPMP 组中。

8 **对于具有 ATM 接口的 IPMP，请在 LAN 仿真模式下配置 ATM 接口。**

使用经典 IP over ATM 技术的接口不支持 IPMP，如 IETF RFC 1577 (http://datatracker.ietf.org/doc/rfc1577/) 和 IETF RFC 2225 (http://datatracker.ietf.org/doc/rfc2225/) 中所定义。

▼ 如何配置使用 DHCP 的 IPMP 组

可以使用活动/活动接口或活动/备用接口配置多接口 IPMP 组。如需相关信息，请参见第 64 页中的 “IPMP 接口配置的类型”。以下过程介绍了如何通过使用 DHCP 配置活动/备用 IPMP 组。

开始之前 确保即将包含在目标 IPMP 组中的 IP 接口已在系统的网络数据链路上正确配置。有关配置链路和 IP 接口的过程，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何配置 IP 接口”。即使底层 IP 接口尚未创建，您仍可以创建一个 IPMP 接口。然而，如果未创建底层 IP 接口，对该 IPMP 接口的后续配置将会失败。

此外，如果您使用的是基于 SPARC 系统，请为每个接口配置一个唯一的 MAC 地址。有关过程，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何确保每个接口的 MAC 地址是唯一的”。

最后，如果您使用 DHCP，请确保底层接口具有无限租用期。否则，如果 IPMP 组出现故障，测试地址将到期，`in.mpathd` 守护进程将随后禁用基于探测器的故障检测，而使用基于链路的故障检测。如果基于链路的故障检测发现接口运行正常，守护进程可能会错误地报告接口已修复。有关配置 DHCP 的更多信息，请参阅《在 Oracle Solaris 11.1 中使用 DHCP》 。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **创建一个 IPMP 接口。**

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

其中，*ipmp-interface* 指定 IPMP 接口的名称。您可以为 IPMP 接口指定任何有意义的名称。与任何 IP 接口一样，该名称包含一个字符串和一个数字，例如 `ipmp0`。

3 **创建底层 IP 接口（如果尚不存在）。**

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

其中 *under-interface* 指将添加到 IPMP 组的 IP 接口。

4 **将要包含测试地址的底层 IP 接口添加到 IPMP 组。**

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

您可以为 IPMP 组添加系统中可用的所有 IP 接口。

5 **让 DHCP 配置和管理 IPMP 接口上的数据地址。**

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

步骤 5 将 DHCP 服务器提供的地址与地址对象关联。地址对象使用 *interface*/*address-type* 格式（例如，`ipmp0/v4`）唯一标识 IP 地址。有关地址对象的更多信息，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何配置 IP 接口”。

6 **如果您使用基于探测器的故障检测和测试地址，请让 DHCP 管理底层接口的测试地址。**

对 IPMP 组的每个底层接口发出以下命令。

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

步骤 6 自动创建的地址对象使用 *under-interface*/*address-type* 格式，例如，`net0/v4`。

示例 6–2 使用 DHCP 配置 IPMP 组

本示例显示如何使用 DHCP 配置活动/备用 IPMP 组，并且基于以下方案：

- 一个 IPMP 组中配置了三个底层接口（`net0`、`net1` 和 `net2`）。
- IPMP 接口 `ipmp0` 与 IPMP 组共享相同的名称。
- `net2` 是指定的备用接口。
- 所有的底层接口均指定了测试地址。

首先，管理员创建 IPMP 接口。

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

接下来，管理员创建底层 IP 接口并将其添加到该 IPMP 接口。

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

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

接下来，管理员将 DHCP 管理的 IP 地址指定给该 IPMP 接口。指定给该 IPMP 接口的 IP 地址是数据地址。在本示例中，IPMP 接口有两个数据地址。

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

接下来，管理员将 DHCP 管理的 IP 地址指定给该 IPMP 组的底层 IP 接口。指定给底层接口的 IP 地址是要用于基于探测器的故障检测的测试地址。

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

最后，管理员将 net2 配置为备用接口。

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

▼ 如何手动配置活动/活动 IPMP 组

以下过程介绍了如何手动配置活动/活动 IPMP 组。在此过程中，步骤 1-4 介绍了如何配置基于链路的活动/活动 IPMP 组。步骤 5 介绍了如何使基于链路的配置基于探测器。

开始之前 确保即将包含在目标 IPMP 组中的 IP 接口已在系统的网络数据链路上正确配置。有关配置链路和 IP 接口的过程，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何配置 IP 接口”。即使底层 IP 接口尚不存在，您仍可以创建一个 IPMP 接口。然而，随后在此 IPMP 接口上的配置将失败。

此外，如果您使用的是基于 SPARC 系统，请为每个接口配置一个唯一的 MAC 地址。有关过程，请参见《在 Oracle Solaris 11.1 中使用固定网络配置连接系统》中的“如何确保每个接口的 MAC 地址是唯一的”。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **创建一个 IPMP 接口。**

   # ipadm create-ipmp *ipmp-interface*

   其中，*ipmp-interface* 指定 IPMP 接口的名称。您可以为 IPMP 接口指定任何有意义的名称。与任何 IP 接口一样，该名称包含一个字符串和一个数字，例如 ipmp0。

3. **将底层 IP 接口添加到组中。**

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

   其中 *under-interface* 指 IPMP 组的底层接口。您可以添加系统中可用的所有 IP 接口。

   注 – 在双栈环境中，如果将某个接口的 IPv4 实例放入特定组中，则 IPv6 实例将自动放入同一组中。

4. **将数据地址添加到 IPMP 接口。**

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

   其中，*address* 可以采用 CIDR 表示法。

   注 – 只有 IPMP 组名称的 DNS 地址或 IP 地址是必需的。

5 **如果您使用基于探测器的故障检测和测试地址，请为底层接口添加测试地址。**

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

其中，*address* 可以采用 CIDR 表示法。IPMP 组中的所有测试 IP 地址必须属于单个 IP 子网并使用相同的网络前缀。

▼ 如何手动配置活动/备用 IPMP 组

有关备用接口的信息，请参见第 64 页中的 “IPMP 接口配置的类型”。以下过程介绍了如何配置一个 IPMP 组，其中将一个接口保留为备用。仅当组中的一个活动接口出现故障时，才部署此接口。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **创建一个 IPMP 接口。**

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

其中，*ipmp-interface* 指定 IPMP 接口的名称。您可以为 IPMP 接口指定任何有意义的名称。与任何 IP 接口一样，该名称包含一个字符串和一个数字，例如 `ipmp0`。

3 **将底层 IP 接口添加到组中。**

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

其中 *under-interface* 指 IPMP 组的底层接口。您可以添加系统中可用的所有 IP 接口。

注 – 在双栈环境中，如果将某个接口的 IPv4 实例放入特定的 IPMP 组中，则 IPv6 实例将自动放入同一组中。

---

4 **将数据地址添加到 IPMP 接口。**

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

其中，*address* 可以采用 CIDR 表示法。

5 **如果您使用基于探测器的故障检测和测试地址，请为底层接口添加测试地址。**

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

其中，*address* 可以采用 CIDR 表示法。IPMP 组中的所有测试 IP 地址必须属于单个 IP 子网并使用相同的网络前缀。

6 **将一个底层接口配置为备用接口。**

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

示例 6–3 配置活动/备用 IPMP 组

本示例说明如何手动创建活动/备用 IPMP 配置。

首先，管理员创建 IPMP 接口。

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

接下来，管理员创建底层 IP 接口并将其添加到该 IPMP 接口。

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

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

接下来，管理员将 IP 地址指定给该 IPMP 接口。指定给该 IPMP 接口的 IP 地址是数据地址。在本示例中，IPMP 接口有两个数据地址。

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

本示例中的 IP 地址包括 `prefixlen` 属性，以十进制数表示。IP 地址的 `prefixlen` 部分指定地址的 IPv4 网络掩码或 IPv6 前缀所包含的地址的最左侧的连续位数。其余低阶位定义地址的主机部分。将 `prefixlen` 属性转换为地址的文本表现形式时，对于用于网络部分的位位置，此地址包含 1，对于主机部分，此地址包含 0。`dhcp` 地址对象类型不支持此属性。有关更多信息，请参见 `ipadm(1M)` 手册页。

接下来，管理员将 IP 地址指定给该 IPMP 组的底层 IP 接口。指定给底层接口的 IP 地址是要用于基于探测器的故障检测的测试地址。

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

最后，管理员将 `net2` 配置为备用接口。

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

管理员可以使用 `ipmpstat` 命令查看 IPMP 配置。

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

# ipmpstat -t
INTERFACE  MODE     TESTADDR        TARGETS
```

```
net0        routes    192.168.10.30   192.168.10.1
net1        routes    192.168.10.32   192.168.10.1
net2        routes    192.168.10.34   192.168.10.5
```

维护 IPMP

本节包含用于维护您在系统上创建的 IPMP 组的过程。

▼ 如何将接口添加到 IPMP 组

开始之前 确保您添加到组的接口满足所有要求。有关要求列表，请参见第 77 页中的 “如何规划 IPMP 组”。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **如果底层 IP 接口尚不存在，则创建该接口。**

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

3. **将 IP 接口添加到 IPMP 组中。**

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

   其中，*ipmp-interface* 指要添加底层接口的 IPMP 组。

示例 6–4 将接口添加到 IPMP 组

以下示例向 IPMP 组 `ipmp0` 添加接口 `net4`：

```
# ipadm create-ip net4
# ipadm add-ipmp -i net4 ipmp0
# ipmpstat -g
GROUP    GROUPNAME    STATE      FDT       INTERFACES
ipmp0    ipmp0        ok         10.00s    net0 net1 net4
```

▼ 如何从 IPMP 组中删除接口

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **从 IPMP 组中删除一个或多个接口。**

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

其中，*under-interface* 指要从 IPMP 组中删除的 IP 接口，*ipmp-interface* 指要从中删除底层接口的 IPMP 组。

您可以根据需要在单个命令中删除任意数量的底层接口。删除所有底层接口并不会删除 IPMP 接口。相反，它作为一个空 IPMP 接口或空组存在。

示例 6–5 从 IPMP 组中删除接口

以下示例从 IPMP 组 `ipmp0` 中删除接口 `net4`：

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

▼ 如何添加 IP 地址

要添加 IP 地址，可使用 `ipadm create-addr` 子命令。请注意，在 IPMP 中，IP 地址可以是数据地址或测试地址。数据地址添加到 IPMP 接口。测试地址添加到 IPMP 接口的底层接口。以下过程介绍了如何添加 IP 地址（测试地址或数据地址）。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **要将数据地址添加到 IPMP 组，请键入以下命令：**

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

将自动为您刚刚创建的 IP 地址指定地址对象。地址对象是 IP 地址的唯一标识符。地址对象的名称使用 *interface*/*random-string* 命名约定。因此，数据地址的地址对象名称中将会包含 IPMP 接口。

3 **要为 IPMP 组的底层接口添加测试地址，请键入以下命令：**

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

将自动为您刚刚创建的 IP 地址指定地址对象。地址对象是 IP 地址的唯一标识符。地址对象的名称使用 *interface*/*random-string* 命名约定。因此，测试地址的地址对象名称中将会包含底层接口。

▼ 如何删除 IP 地址

要删除 IP 地址，可使用 `ipadm delete-addr` 子命令。请注意，在 IPMP 中，IPMP 接口承载数据地址，底层接口承载测试地址。以下过程说明了如何删除 IP 地址（数据地址或测试地址）。

1. **成为管理员。**

   有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2. **确定要删除的地址。**

   - 要显示数据地址列表，请键入以下命令：

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

   - 要显示测试地址列表，请键入以下命令：

     ```
     # ipadm show-addr
     ```

     测试地址由名称中包含配置了地址的底层接口的地址对象进行标识。

3. **要从 IPMP 组中删除数据地址，请键入以下命令：**

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

   其中，*addrobj* 必须包含 IPMP 接口的名称。如果您键入的地址对象不包含 IPMP 接口名称，则要删除的地址不是数据地址。

4. **要从 IPMP 组中删除测试地址，请键入以下命令：**

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

   其中，*addrobj* 必须包含正确的底层接口名称，以便删除正确的测试地址。

示例 6–6 从接口删除测试地址

以下示例使用示例 6–3 中的活动/备用 IPMP 组 `ipmp0` 的配置。该示例从底层接口 `net1` 中删除测试地址。

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

# ipadm delete-addr net1/v4
```

▼ 如何将接口从一个 IPMP 组移至另一个 IPMP 组

如果某个接口属于现有的 IPMP 组，则可以将该接口放入新的 IPMP 组中。无需从当前 IPMP 组中删除该接口。接口放入新组中后，该接口将自动从任何现有的 IPMP 组中删除。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **将接口移动到新的 IPMP 组。**

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

其中，*under-interface* 指要移动的底层接口，*ipmp-interface* 指要将底层接口移动到的 IPMP 接口。

示例 6–7 将接口移动到其他 IPMP 组

在本示例中，IPMP 组的底层接口包括 `net0`、`net1` 和 `net2`。以下命令从 IPMP 组 `ipmp0` 中删除 `net0` 接口，然后将 `net0` 放入 IPMP 组 `cs-link1`：

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

▼ 如何删除 IPMP 组

如果您不再需要特定 IPMP 组，请使用此过程。

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **确定要删除的 IPMP 组和底层 IP 接口。**

```
# ipmpstat -g
```

3 **删除当前属于 IPMP 组的所有 IP 接口。**

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

其中，*under-interface* 指要删除的底层接口，*ipmp-interface* 指要从中删除底层接口的 IPMP 接口。

注 – 要成功地删除 IPMP 接口，不能存在作为 IPMP 组的一部分的 IP 接口。

---

4 **删除 IPMP 接口。**

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

删除 IPMP 接口后，与该接口相关联的任何 IP 地址将从系统中删除。

示例 6–8 删除 IPMP 接口

以下示例删除具有底层 IP 接口 `net0` 和 `net1` 的接口 `ipmp0`：

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

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

# ipadm delete-ipmp ipmp0
```

配置基于探测器的故障检测

基于探测器的故障检测涉及目标系统的使用，如第 70 页中的 “基于探测器的故障检测”中所述。在确定基于探测器的故障检测的目标时，`in.mpathd` 守护进程在两种模式下运行：路由器目标模式或多播目标模式。在路由器目标模式中，守护进程探测在路由表中定义的目标。如果没有定义目标，则守护进程在多播目标模式下运行，其中发出多播包以探测 LAN 上的相邻主机。

您最好设置供 `in.mpathd` 守护进程探测的目标系统。对于一些 IPMP 组，缺省路由器作为目标就足够了。但是，对于另一些 IPMP 组，则可能需要为基于探测器的故障检测配置特定目标。要指定目标，请将路由表中的主机路由设置为探测器目标。在路由表中配置的任何主机路由会列在缺省路由器的前面。IPMP 使用显式定义的主机路由来选择目标。因此，您应设置主机路由以配置特定的探测器目标，而不是使用缺省路由器。

要在路由表中设置主机路由，请使用 `route` 命令。您可以将此命令与 `-p` 选项结合使用来添加持久性路由。例如，使用 `route -p add` 添加的路由在您重新引导系统后仍保留在路由表中。因此，您可以使用 `-p` 选项添加持久性路由，从而无需使用任何特殊脚本在每次系统启动时重新创建这些路由。要以最佳方式使用基于探测器的故障检测，确保您设置多个目标来接收探测器。

第 90 页中的 “如何为基于探测器的故障检测手动指定目标系统”过程显示了为基于探测器的故障检测将持久性路由添加到目标的确切语法。有关 `route` 命令的选项的更多信息，请参阅 `route(1M)` 手册页。

本节包含以下主题：

为基于探测器的故障检测选择目标的要求

在评估网络中哪些主机可以用作合适的目标时，请遵循以下条件列表：

- 确保将来的目标可用并且正在运行。建立其 IP 地址的列表。
- 确保目标接口与要配置的 IPMP 组位于同一网络中。
- 目标系统的网络掩码和广播地址必须与 IPMP 组中的地址相同。
- 目标主机必须能够应答使用基于探测器的故障检测的接口发出的 ICMP 请求。

配置基于探测器的故障检测（任务列表）

以下任务列表列出了为 IPMP 组配置基于探测器的故障检测的过程。

任务	说明	参考
选择基于探测器的故障检测的工作方式。	确定基于探测器的故障检测应该使用传递式探测器还是测试地址。	第 89 页中的 “如何选择要使用的故障检测方法”
选择要用于基于探测器的故障检测的目标系统。	指定系统的 IP 地址，基于探测器的故障检测将使用该地址来测试 IPMP 组的底层接口的状态。	第 90 页中的 “如何为基于探测器的故障检测手动指定目标系统”
选择如何在 IPMP 组中重新部署已修复的接口。	确定是应该在 IPMP 组中自动重新激活已修复的 IP 接口，还是使其保持取消激活状态。	第 90 页中的 “如何配置 IPMP 守护进程的行为”

▼ 如何选择要使用的故障检测方法

缺省情况下，使用测试地址运行基于探测器的故障检测。如果 NIC 驱动程序支持基于链路的故障检测，还将自动启用它。

如果 NIC 驱动程序支持基于链路的故障检测，您无法禁用此方法。但是，您可以选择要实现哪种类型的基于探测器的故障检测。

开始之前 确保您的探测器目标满足第 89 页中的 “为基于探测器的故障检测选择目标的要求”中列出的要求。

1 要仅使用传递式探测，请执行以下步骤：

a. 使用适当的 SMF 命令打开 IPMP 属性 `transitive-probing`。

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

有关设置此属性的更多信息，请参见 `in.mpathd(1M)` 手册页。

b. **删除已为 IPMP 组配置的任何现有测试地址。**

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

其中，*addrobj* 必须是承载测试地址的底层接口。

2 **要使用测试地址来探测故障，请执行以下步骤：**

a. **如有必要，请关闭传递式探测。**

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

b. **将测试地址指定给 IPMP 组的底层接口。**

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

其中，*address* 可以采用 CIDR 表示法，而 *under-interface* 是 IPMP 组的底层接口。

▼ 如何为基于探测器的故障检测手动指定目标系统

此过程介绍了在使用测试地址执行基于探测器的故障检测时如何添加特定目标。

开始之前 确保您的探测器目标满足第 89 页中的“为基于探测器的故障检测选择目标的要求”中列出的要求。

1 **使用您的用户帐户登录到要在其中配置基于探测器的故障检测的系统。**

2 **将路由添加到要用作基于探测器的故障检测中的目标的特定主机。**

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

其中 *destination-IP* 和 *gateway-IP* 是要用作目标的主机的 IPv4 地址。例如，可以键入以下内容指定目标系统 `192.168.10.137`，该目标系统与 IPMP 组 `ipmp0` 中的接口位于同一子网中：

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

每次重新启动系统时，将自动配置此新路由。如果只想为基于探测器的故障检测定义一个到目标系统的临时路由，请不要使用 `-p` 选项。

3 **将路由添加到网络中要用作目标系统的其他主机。**

▼ 如何配置 IPMP 守护进程的行为

可使用 IPMP 配置文件 `/etc/default/mpathd` 为 IPMP 组配置以下系统范围的参数。

- `FAILURE_DETECTION_TIME`
- `FAILBACK`

- TRACK_INTERFACES_ONLY_WITH_GROUPS

1 **成为管理员。**

有关更多信息，请参见《Oracle Solaris 11.1 管理：安全服务》中的“如何使用指定给您的管理权限”。

2 **编辑 /etc/default/mpathd 文件。**

更改这三个参数中的一个或多个参数的缺省值。

a. **为 FAILURE_DETECTION_TIME 参数键入新值。**

```
FAILURE_DETECTION_TIME=n
```

其中 *n* 是 ICMP 探测器用来检测是否发生接口故障的时间（以秒为单位）。缺省值是 10 秒。

b. **为 FAILBACK 参数键入新值。**

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

- yes – yes 值是 IPMP 的故障恢复行为的缺省值。当检测到故障接口修复时，网络访问故障恢复到已修复的接口，如第 72 页中的 “检测物理接口修复”中所述。
- no – no 值指示数据通信不返回到已修复的接口。当检测到某个故障接口已修复时，会为此接口设置 INACTIVE 标志。此标志表示该接口当前不用于数据通信。但该接口仍可用于探测器通信。

  例如，假定 IPMP 组 ipmp0 包含两个接口，net0 和 net1。在 /etc/default/mpathd 文件中，已设置 FAILBACK=no 参数。如果 net0 出现故障，则它被标记为 FAILED 并变得不可用。修复后，接口标记为 INACTIVE，但由于 FAILBACK=no 值，它仍保持不可用状态。

  如果 net1 出现故障并且只有 net0 处于 INACTIVE 状态，则会清除 net0 的 INACTIVE 标志并且该接口变为可用。如果 IPMP 组中还有其他接口也处于 INACTIVE 状态，则当 net1 出现故障时，这些处于 INACTIVE 状态的任一接口（不一定是 net0）会被清除标志并变为可用。

c. **为 TRACK_INTERFACES_ONLY_WITH_GROUPS 参数键入新值。**

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

注 – 有关此参数和匿名组功能的信息，请参见第 72 页中的 “故障检测和匿名组功能”。

- yes – yes 值是 IPMP 行为的缺省值。此值使 IPMP 忽略未配置到 IPMP 组中的网络接口。

- no – no 值为**所有**网络接口设置故障和修复检测，无论它们是否配置到 IPMP 组中。不过，在未配置到 IPMP 组中的接口上检测到故障或修复时，在 IPMP 中不触发操作以维持该接口的网络功能。因此，no 值仅用于报告故障，并不能直接提高网络可用性。

3 **重新启动 in.mpathd 守护进程。**

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

守护进程将重新启动，新参数值生效。

监视 IPMP 信息

本节中的示例使用 ipmpstat 命令，从而使您能够监视系统中 IPMP 组的不同方面。您可以将 IPMP 组作为一个整体来观察其状态，也可以观察其底层 IP 接口的状态。您还可以验证 IPMP 组的数据地址和测试地址的配置。通过使用 ipmpstat 命令还可以获取有关故障检测的信息。有关 ipmpstat 命令及其选项的更多信息，请参见ipmpstat(1M) 手册页。

当您使用 ipmpstat 命令时，缺省情况下，将显示 80 列可容纳的最有意义的字段。在输出中，将显示特定于与 ipmpstat 命令结合使用的选项的所有字段（使用 ipmpstat -p 语法时除外）。

缺省情况下，输出中显示主机名而不是数字 IP 地址（只要主机名存在）。要在输出中列出数字 IP 地址，请将 -n 选项与其他选项结合使用以显示特定的 IPMP 组信息。

注 – 在以下示例中，ipmpstat 命令的使用不需要系统管理员特权，除非另有说明。

请将以下选项与 ipmpstat 命令结合使用以确定要显示的信息：

- -g 显示有关系统上 IPMP 组的信息。请参见示例 6–9。
- -a 显示为 IPMP 组配置的数据地址。请参见示例 6–10。
- -i 显示与 IPMP 配置相关的 IP 接口的信息。请参见示例 6–11。
- -t 显示有关用于故障检测的目标系统的信息。该选项还显示 IPMP 组使用的测试地址。请参见示例 6–12。
- -p 显示有关用于故障检测的探测器的信息。请参见示例 6–13。

以下示例显示可以通过 ipmpstat 命令获取的有关系统中 IPMP 配置的信息。

示例 6–9 获取 IPMP 组信息

-g 选项列出系统上各 IPMP 组的状态，包括其底层接口的状态。如果为特定组启用了基于探测器的故障检测，则该命令还包括该组的故障检测时间。

示例 6–9 获取 IPMP 组信息 （续）

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

输出字段提供以下信息：

GROUP　指定 IPMP 接口名称。对于匿名组，此字段为空。有关匿名组的更多信息，请参见 in.mpathd(1M) 手册页。

GROUPNAME　指定 IPMP 组的名称。对于匿名组，此字段将为空。

STATE　表示一个 IPMP 组的当前状态，可以是以下值之一：

- ok 表示 IPMP 组的所有底层接口都可用。
- degraded 表示该组中的部分底层接口不可用。
- failed 表示该组的所有接口都不可用。

FDT　指定故障检测时间（如果启用了故障检测）。如果禁用了故障检测，则此字段为空。

INTERFACES　指定属于该 IPMP 组的底层接口。在此字段中，首先列出活动接口，然后是非活动接口，最后列出不可用的接口。接口的状态由列出方式表示：

- *interface*（不带圆括号或方括号）表示活动接口。系统使用活动接口发送或接收数据通信。
- *(interface)*（带圆括号）表示能正常工作但处于非活动状态的接口。根据管理策略的定义，该接口并未使用。
- *[interface]*（带方括号）表示接口不可用，因为它已出现故障或处于脱机状态。

示例 6–10 获取 IPMP 数据地址信息

-a 选项显示数据地址以及每个地址所属的 IPMP 组。显示的信息还包括哪些地址可用，具体取决于是否通过 ipadm [up-addr/down-addr] 命令切换了地址的状态。您还可以确定一个地址可以在哪个传入或传出接口上使用。

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

示例 6–10 获取 IPMP 数据地址信息 （续）

输出字段提供以下信息：

ADDRESS 指定主机名或数据地址（如果 -n 选项与 -a 选项结合使用）。

STATE 指示 IPMP 接口上的地址状态是 up（因此可用）还是 down（因此不可用）。

GROUP 指定承载特定数据地址的 IPMP 接口。通常，在 Oracle Solaris 中，IPMP 组的名称是 IPMP 接口。

INBOUND 标识接收给定地址的包的接口。根据外部事件，字段信息可能会发生更改。例如，如果数据地址已关闭或者 IPMP 组中没有活动的 IP 接口，此字段为空。空字段指示系统当前没有接受发送到给定地址的 IP 包。

OUTBOUND 标识发送使用给定地址作为源地址的包的接口。与 INBOUND 字段一样，OUTBOUND 信息也可能根据外部事件而变化。空字段指示系统当前未发送具有给定的源地址的包。该字段可能为空，原因是该地址已关闭或组中没有活动的 IP 接口。

示例 6–11 获取有关 IPMP 组的底层 IP 接口的信息

-i 选项显示有关 IPMP 组的底层 IP 接口的信息。

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

输出字段提供以下信息：

INTERFACE 指定每个 IPMP 组的每个底层接口。

ACTIVE 指示该接口是正常工作并在使用中 (yes) 还是没有 (no)。

GROUP 指定 IPMP 接口名称。对于匿名组，此字段为空。有关匿名组的更多信息，请参见 in.mpathd(1M) 手册页。

FLAGS 指示每个底层接口的状态，可以是以下各项之一或这些项的任意组合：

- i 指示为接口设置了 INACTIVE 标志。因此，该接口不用于发送或接收数据通信。
- s 指示接口配置为备用接口。
- m 指示接口由系统指定用于发送和接收 IPMP 组的 IPv4 多播通信。

示例 6–11 获取有关 IPMP 组的底层 IP 接口的信息 （续）

- b 指示接口由系统指定用于接收 IPMP 组的广播通信。
- M 指示接口由系统指定用于发送和接收 IPMP 组的 IPv6 多播通信。
- d 指示接口已关闭，因此不可用。
- h 指示接口与另一个接口共享一个重复的物理硬件地址并且已处于脱机状态。h 标志指示接口不可用。

LINK 指示基于链路的故障检测的状态，为以下状态之一：

- up 或 down 分别指示链路可用或不可用。
- unknown 指示驱动程序不支持链路是 up 还是 down 的通知，因此不检测链路状态的变化。

PROBE 指定使用测试地址配置的接口的基于探测器的故障检测的状态，如下所示：

- ok 指示探测器正常工作并处于活动状态。
- failed 表示基于探测器的故障检测已检测到接口未正常工作。
- unknown 指示找不到合适的探测器目标，因此无法发送探测器。
- disabled 指示在接口上没有配置 IPMP 测试地址。因此，基于探测器的故障检测处于禁用状态。

STATE 指定接口的整体状态，如下所示：

- ok 指示接口已联机并正在基于故障检测方法的配置正常工作。
- failed 指示接口当前未工作，原因是接口的链路已关闭或探测器检测已确定该接口无法发送或接收通信。
- offline 指示接口不可用。通常，在以下情况下会将接口置于脱机状态：
  - 正在测试接口。
  - 正在执行动态重新配置。
  - 接口与另一个接口共享一个重复的硬件地址。
- unknown 指示无法确定 IPMP 接口的状态，因为找不到基于探测器的故障检测的探测器目标。

示例 6–12 获取 IPMP 探测器目标信息

-t 选项标识与 IPMP 组中每个 IP 接口相关联的探测器目标。第一个输出是为基于探测器的故障检测使用测试地址的 IPMP 配置示例。

```
$ ipmpstat -nt
INTERFACE   MODE          TESTADDR        TARGETS
net0        routes        192.168.85.30   192.168.85.1 192.168.85.3
net1        disabled      --              --
```

示例 6–12 获取 IPMP 探测器目标信息 （续）

```
net3        disabled      --              --
net4        routes        192.1.2.200     192.1.2.1
net2        multicast     128.9.0.200     128.0.0.1 128.0.0.2
net6        multicast     128.9.0.201     128.0.0.2 128.0.0.1
net5        multicast     128.9.0.202     128.0.0.1 128.0.0.2
net7        multicast     128.9.0.203     128.0.0.1 128.0.0.2
```

第二个输出是使用传递式探测（即，没有测试地址的基于探测器的故障检测）的 IPMP 配置示例。

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

输出字段提供以下信息：

INTERFACE 指定 IPMP 组的每个底层接口。

MODE 指定用于获取探测器目标的方法。

- routes 指示使用系统路由表查找探测器目标。
- mcast 指示使用多播 ICMP 探测器查找目标。
- disabled 指示已为接口禁用基于探测器的故障检测。
- transitive 指示使用传递式探测检测故障，如第二个示例中所示。请注意，无法在同时使用传递式探测器和测试地址的情况下实现基于探测器的故障检测。如果您不希望使用测试地址，则必须启用传递式探测。如果您不希望使用传递式探测，则必须配置测试地址。有关概述，请参见第 70 页中的 “基于探测器的故障检测”。

TESTADDR 指定主机名；如果 -n 选项与 -t 选项结合使用，则指定分配给用于发送和接收探测器的接口的 IP 地址。

如果使用传递式探测，则接口名称指当前未用于接收数据的底层 IP 接口。这些名称还指示传递式测试探测器是使用这些指定接口的源地址发送的。对于接收数据的活动底层 IP 接口，显示的 IP 地址指示传送 ICMP 探测器的源地址。

注 – 如果某 IP 接口同时配置了 IPv4 和 IPv6 测试地址，分别显示每个测试地址的探测器目标信息。

TARGETS 在空格分隔的列表中列出当前探测器目标。探测器目标以主机名或 IP 地址形式显示。如果 -n 选项与 -t 选项结合使用，将会显示 IP 地址。

示例 6–13 观察 IPMP 探测器

-p 选项使您能够观察正在运行的探测器。将该选项和 ipmpstat 命令一起使用时，会持续显示有关系统上探测器活动的信息，直到您按 Control-C 组合键终止命令。您必须具有主管理员权限才能运行此命令。

第一个输出是为基于探测器的故障检测使用测试地址的 IPMP 配置示例。

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

第二个输出是使用传递式探测（即，没有测试地址的基于探测器的故障检测）的 IPMP 配置示例。

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

输出字段提供以下信息：

TIME　指定发送探测器的时间（相对于发出 ipmpstat 命令的时间）。如果探测器在 ipmpstat 开始之前已启动，则显示的时间为负值（即相对于发出命令的时间）。

INTERFACE　指定在其上发送探测器的接口。

PROBE　指定代表探测器的标识符。如果使用传递式探测进行故障检测，则标识符具有前缀 t（对于传递式探测器）或 i（对于 ICMP 探测器）。

NETRTT　指定探测器的总网络往返时间（以毫秒为单位）。NETRTT 指从 IP 模块发送探测器到 IP 模块接收到来自目标的 ack 包的时间。如果 in.mpathd 守护进程已确定探测器丢失，则该字段为空。

RTT　指定探测器的总往返时间（以毫秒为单位）。RTT 指从 in.mpathd 守护进程执行代码以发送探测器到守护进程完成对来自目标的 ack 包的处理之间的时间。如果守护进程已确定探测器丢失，则该字段为空。如果 RTT 出现峰值而 NETRTT 没有出现，则可能表明本地系统过载。

RTTAVG　指定接口上的探测器在本地系统和目标之间的平均往返时间。平均往返时间可以帮助确定速度慢的目标。如果数据不足以计算平均值，此字段为空。

示例 6–13 观察 IPMP 探测器 （续）

TARGET 指定主机名；如果 -n 选项与 -p 选项结合使用，则指定向其发送探测器的目标地址。

定制 ipmpstat 命令的输出

ipmpstat 命令的 -o 选项允许您定制输出。将此选项与之前的其他 ipmpstat 选项结合使用可选择除主要选项通常显示的全部字段之外要显示的特定字段。

例如，-g 选项提供以下信息：

- IPMP 组
- IPMP 组名称
- 组状态
- 故障检测时间
- IPMP 组的底层接口

假定您希望仅显示系统上 IPMP 组的状态。您可以结合使用 -o 和 -g 选项并指定 groupname 和 state 字段，如以下示例所示：

```
$ ipmpstat -g -o groupname,state
GROUPNAME  STATE
ipmp0      ok
accgt1     failed
field2     degraded
```

要显示 ipmpstat 命令关于特定类型信息的所有字段，请在语法中包含 -o all。例如，要列出提供组信息的所有字段，请键入 ipmpstat -g -o all。

在脚本中使用 ipmpstat 命令

当您通过脚本或使用命令别名发出命令时，尤其是如果您还想生成计算机可解析的输出，-o 选项非常有用。

要生成计算机可解析的信息，请将 -P 和 -o 选项与其他 ipmpstat 主要选项之一以及要显示的特定字段结合使用。计算机可解析的输出在以下几个方面有别于常规输出：

- 省略了列标题。
- 字段以冒号 (:) 分隔。
- 具有空值的字段为空，而不是用双短划线 (--) 填充。
- 在请求多个字段时，如果字段中包含原义冒号 (:) 或反斜杠 (\)，可以为这些字符添加反斜杠 (\) 前缀对其做转义或排除处理。

为了正确使用 ipmpstat -P 语法，请遵循以下规则：

- 将 -o *option field* 与 -P 选项结合使用。使用逗号分隔多个选项字段。
- 不将 -o all 与- P 选项结合使用。

忽略上述任一规则将导致 ipmpstat -P 失败。

以下示例显示了使用 -P 选项时的信息格式。

```
$ ipmpstat -P -o -g groupname,fdt,interfaces
ipmp0:10.00s:net0 net1
acctg1::[net3 net4]
field2:20.00s:net2 net7 (net5) [net6]
```

组名称、故障检测时间和底层接口是组信息字段。因此，应将 -o -g 选项与 -P 选项结合使用。

-P 选项设计用于脚本中。以下示例显示了如何通过脚本发出 ipmpstat 命令。此脚本用于显示 IPMP 组的故障检测时间。

```
getfdt() {
         ipmpstat -gP -o group,fdt | while IFS=: read group fdt; do
             [[ "$group" = "$1" ]] && { echo "$fdt"; return; }
         done
     }
```

◆ ◆ ◆ 第 7 章

7 使用 LLDP 交换网络连接信息

在任一局域网上，都不会孤立地配置诸如系统和交换机之类的单个组件。要有效地承载网络通信，网络中的系统配置必须相互协调。因此，包交换基于组件的功能、链路属性配置、带宽限制等。可以手动配置各个系统、交换机以及其他组件以确保这些组件之间的兼容性。但是，这种方法存在风险，很容易导致配置错误，尤其是当不同的管理员在不同的系统上独立工作时。更好的选择是使用一种允许系统将各自的配置信息传送到对等方系统的技术。通过该技术可降低风险并使网络管理变得更加容易。Oracle Solaris 支持针对此特定用途的链路层发现协议 (Link Layer Discovery Protocol, LLDP)。

本章介绍如何使系统通过使用 LLDP 在整个本地网络内交换系统和网络连接信息。本章包含以下主题：

Oracle Solaris 中的 LLDP 概述

LLDP 在 LAN 中通告信息，以进行拓扑搜索。使用此协议，系统可以向网络上的其他系统通告连接和管理信息。此信息可以包括系统功能、管理地址以及其他与网络操作相关的信息。此协议还使该系统能够接收有关同一本地网络上其他系统的类似信息。

在 Oracle Solaris 中，LLDP 还用于交换数据中心桥接交换 (data center bridging exchange, DCBX) 协议 TLV 单元。DCBX 提供了有关基于优先级的流控制 (priority-based flow control, PFC) 和增强传输选择 (enhanced transmission selection, ETS) 之类的 DCB 功能的配置信息。有关 DCB 的更多信息，请参见第 8 章，使用 Oracle Solaris 中的数据中心桥接功能。

使用 LLDP，系统管理员可以轻松检测有故障的系统配置，尤其是在包含虚拟局域网 (virtual local area network, VLAN)、链路聚合等的复杂网络中。可以轻松得到有关拓扑的信息，而无需跟踪构成网络的服务器、交换机以及其他设备之间的物理连接。

LLDP 实现的组件

LLDP 是使用以下组件实现的：

- 必须安装 LLDP 软件包以启用 LLDP 功能。此软件包提供 LLDP 守护进程、命令行实用程序、服务清单和脚本以及 LLDP 运行所需的其他组件。
- `lldp` 服务由 `svcadm` 命令启用。此服务管理 LLDP 守护进程和负责启动、停止、重新启动或刷新该守护进程。安装 LLDP 软件包后，此服务将自动启用。
- `lldpadm` 命令用于管理各个链路上的 LLDP 以及其他用途，例如配置 LLDP 的操作模式、指定要传送的类型长度值 (Type-Length-Value, TLV) 单元以及配置 DCBX TLV 单元。特别是，该命令用于设置每个代理的 LLDP 属性以及 LLDP 的全局属性。`lldpadm` 命令的常规子命令对应于 `dladm` 和 `ipadm` 命令的子命令。
  - `lldpadm set-*` 指定要执行的操作（其中为给定的 LLDP 属性设置一个或多个值）。
  - `lldpadm show-*` 显示为指定的 LLDP 属性设置的值。
  - `lldpadm reset-*` 将指定的 LLDP 属性的配置重置为其缺省值。

  后续各节说明了这些子命令的用法。有关 `lldpadm` 命令的更多信息，请参阅 `lldpadm(1M)` 手册页。
- LLDP 库 (`liblldp.so`) 提供的 API 可用于检索链路上的 LLDP 信息、解析 LLDP 包以及执行其他功能。
- LLDP 代理是与启用了 LLDP 的网络接口卡相关联的 LLDP 实例。LLDP 代理控制关联的 NIC 上的 LLDP 行为。LLDP 代理只能在 NIC 或物理链路上配置，并使用这些链路的端口通告信息。因此，在此 LLDP 文档中，LLDP 代理、启用该代理的物理链路以及端口的名称都相同。
- LLDP 守护进程 (`lldpd`) 管理系统上的 LLDP 代理。它还与 `snmpd`（简单网络管理协议 (Simple Network Management Protocol, SNMP) 的守护进程）交互，以通过 SNMP 检索系统上接收的 LLDP 信息。此外，该守护进程还发布 LLDP `sysevents` 信息以及响应来自 LLDP 库的查询。

LLDP 代理的信息源

LLDP 代理传送并接收 LLDP 数据单元 (LLDP data unit, LLDPDU)。代理在两种类型的数据存储中管理和存储这些 LLDPDU 中所包含的信息：

- **本地管理信息库，即本地 MIB**－此数据存储包含与系统中启用 LLDP 代理的特定链路相关的网络信息。本地 MIB 既包含公用信息，也包含独特信息。例如，机箱 ID 是在系统上的所有 LLDP 代理之间共享的公用信息。不过，系统的各个数据链路的端口 ID 是不同的。因此，每个代理管理它自己的本地 MIB。
- **远程 MIB**－此数据存储中的信息接收自对等主机的 LLDP 代理。

LLDP 代理的操作模式

LLDP 代理在以下模式下运行：

- 仅传送 (`txonly`)：在此模式下，LLDP 代理不处理传入 LLDPDU。因此，远程 MIB 为空。
- 仅接收 (`rxonly`)：在此模式下，代理仅处理传入 LLDPDU 并将信息存储在远程 MIB 中。不过，不从本地 MIB 传送信息。
- 传送和接收 (`both`)：在此模式下，代理传送本地信息并处理传入 LLDPDU，因此同时维护本地和远程 MIB。如果底层链路支持 DCB 功能，则自动为受支持的 DCB 功能启用 DCBX TLV 单元。
- 禁用 (`disable`)：在此模式下，代理不存在。

LLDP 的 SMF 属性

服务管理工具 (service management facility, SMF) 属性 `auto-enable-agents` 控制在系统上启用 LLDP 的方式。通过该属性，可以选择在所有物理链路上全局性地启用 LLDP，或者一次仅在一个物理链路上启用 LLDP。该属性可以使用以下三个可能值之一：

- `yes` 是 SMF 属性的缺省值。使用该值会在所有端口上以 Rx 和 Tx 模式全局性地启用 LLDP（如果端口上不存在之前的 LLDP 配置）。如果端口上已存在一个配置，则将保留该端口的配置。例如，如果端口上先前配置了仅 Rx 模式的 LLDP，则 LLDP 服务不会将代理切换为在 Rx 和 Tx 模式下运行。该端口上的 LLDP 继续处于 Rx 模式。
- `force` 在所有端口上以 Rx 和 Tx 模式启用 LLDP，并覆盖任何端口上的任何现有 LLDP 配置。例如，如果端口上先前的 LLDP 配置在仅 Rx 模式下运行，则 LLDP 代理会切换为在 Rx 和 Tx 模式（缺省的 LLDP 模式）下运行。
- `no` 将禁止在所有端口上自动启用 LLDP，但那些已存在现有 LLDP 配置的端口除外。在这些端口上，现有 LLDP 配置会保留。

请注意，每次定制 `auto-enable-agents` 属性后，必须重新启动 LLDP SMF 服务才能使新值生效。

LLDP 代理通告的信息

LLDP 代理在 LLDP 包或 LLDPDU 中传送系统和连接信息。这些包包含以类型长度值 (Type-Length-Value, TLV) 格式单独格式化的信息单元。因此，各信息单元也称为 *TLV* **单元**。某些 TLV 单元是强制性的，在缺省情况下，在启用 LLDP 时它们就包含在 LLDP 包中。无法使用 `lldpadm` 命令排除这些单元中的任何单元。强制性 TLV 单元如下所示：

- 机箱 ID
- 端口 ID
- TTL（生存时间）
- PDU 的结尾

机箱 ID 与通过 `hostid` 命令生成的信息相同。端口 ID 是物理 NIC 的 MAC 地址。根据链路的数量，可以在单个系统中启用多个 LLDP 代理。机箱 ID 和端口 ID 的组合唯一标识了代理，将其与系统上的其他代理区分开来。

可以将可选的 TLV 单元添加到 LLDP 包。供应商可通过这些可选 TLV 单元插入要通告的特定于供应商的 TLV 单元。TLV 单元由各个组织唯一标识符 (organization unique identifier, OUI) 标识，并且根据这些 OUI 是遵循 IEEE 802.1 规范还是 IEEE 802.3 规范而具有不同的类型。可以配置 LLDP 代理属性以启用或禁用这些可选 TLV 单元的传输。

下表列出了各个 TLV 类型或组、其对应的属性名称以及每个属性的 TLV 单元及其说明。您配置上述任一属性以指定在启用 LLDP 时要包含在包中的 TLV 单元。

表 7–1 可以为 LLDP 代理启用的 TLV 单元

TLV 类型	属性名称	TLV 单元	说明
基本管理	`basic-tlv`	`sysname`、`portdesc`、`syscapab`、 `sysdesc`、`mgmtaddr`	指定要通告的系统名称、端口说明、系统功能、系统说明和管理地址。
802.1 OUI	`dot1-tlv`	`vlanname`、`pvid`、`linkaggr`、`pfc`、`appln`、`evb`、 `etscfg`	指定以下要通告的内容：VLAN 名称、端口 VLAN ID、链路聚合、针对基于优先级的流控制的 TLV 单元、应用程序、增强传输选择以及边缘虚拟桥接。
802.3 OUI	`dot3-tlv`	`max-framesize`	指定要通告的最大帧大小。
特定于 Oracle 的 OUI（它被定义为 `0x0003BA`）	`virt-tlv`	`vnic`	指定要通告的 VNIC（如果配置了虚拟网络）。

TLV 单元及其属性

每个 TLV 单元都有您可以使用特定值进一步配置的属性。如果将 TLV 单元启用为 LLDP 代理的属性，则仅使用指定的值在网络中通告该 TLV 单元。以通告系统功能的 TLV 值 syscapab 为例。这些功能可能包括对路由器、网桥、中继器、电话和其他设备的支持。但是，您可以将 syscapab 设置为仅通告在您的特定系统中实际支持的功能，例如路由器和网桥。

配置 TLV 单元的过程取决于要配置**全局** *TLV* **单元**还是**每代理** *TLV* **单元**。

全局 TLV 单元将应用于系统上的所有 LLDP 代理。下表显示了全局 TLV 值及其相应的可能的配置。

表 7–2 全局 TLV 单元及其属性

TLV 名称	TLV 属性名称	可能的属性值	值说明
syscapab	supported	other、repeater、bridge、wlan-ap、router、telephone、docsis-cd、station、cvlan、sylvan、tpmr	代表系统的主要支持的功能。缺省值是 router、station 和 bridge。
	enabled	为 supported 列出的值的子集	代表系统的已启用的功能。
mgmtaddr	ipaddr	ipv4 或 ipv6	指定将与本地 LLDP 代理相关联的 IP 地址的类型。地址将用于到达更高的层实体，并有助于网络管理的发现。只允许指定一种类型。

不能有全局值的 TLV 单元在 LLDP 代理级别进行管理。使用每代理 TLV 单元，通过特定 LLDP 代理启用 TLV 单元传输时将使用您提供的值。

下表显示了一个 LLDP 代理的 TLV 值及其相应的可能的配置。

表 7–3 每代理 TLV 单元及其属性

TLV 名称	TLV 属性名称	可能的属性值	值说明
pfc	willing	on、off	设置 LLDP 代理以接受或拒绝来自远程计算机的与基于优先级的流控制相关的配置信息。

表 7–3 每代理 TLV 单元及其属性 （续）

TLV 名称	TLV 属性名称	可能的属性值	值说明
appln	apt	值是从应用程序优先级表中定义的信息中获取的。	配置应用程序优先级表。此表包含应用程序 TLV 单元及其相应的优先级的列表。应用程序由 id/selector 对标识。表的内容使用以下格式： id/selector/priority
etscfg	willing	on、off	设置 LLDP 代理以接受或拒绝来自远程计算机的与增强传输选择相关的配置信息。

有关每代理 TLV 单元的讨论，请参见第 8 章，使用 Oracle Solaris 中的数据中心桥接功能。

在系统上启用 LLDP

以下过程介绍如何配置 LLDP 以与网络上的其他主机或对等方交换系统信息。

▼ 如何部署 LLDP

以下过程介绍如何使用系统上的 LLDP 启动通告系统功能。缺省情况下，安装完 LLDP 软件包后，LLDP 即已启用并可以使用。如果满足缺省 LLDP 配置，则大部分步骤都是可选的。

开始之前 必须安装 LLDP 软件包才能使用 LLDP。要安装该软件包，请键入以下命令：

```
# pkg install lldp
```

1 确保已启动 LLDP 服务。

```
# svcs lldp
STATE          STIME    FMRI
online         Jul_10   svc:/network/lldp:default
```

如果已禁用 LLDP 服务，请使用以下命令启动该服务：

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

2 执行以下步骤之一。

- 如果希望在系统上全局性地启用 LLDP 服务，请指定 LLDP 代理将通告的 TLV 单元。

  ```
  # lldpadm set-agentprop -p property=value agent
  ```

  其中，*agent* 是 LLDP 代理，通过启用该代理的物理链路进行标识。因此，如果在 `net0` 上启用了 LLDP，则代理为 `net0`。

  注 – 发出 `lldpadm` 子命令时，可以使用缩写形式。例如，对于 `lldpadm set-agentprop`，也可键入 `lldpadm set-ap`。请参阅 `lldpadm(1M)` 手册页，了解子命令及其缩写形式。

  有关 LLDP 代理的属性的说明，请参见第 104 页中的 “LLDP 代理通告的信息”。

  要获取 LLDP 代理的属性列表，请键入 `lldpadm show-agentprop`。或者，请参阅表 7–1。

  有关说明，请参见第 109 页中的 “如何为代理的 LLDP 包指定 TLV 单元”。

- 如果希望只在选定的端口上启用 LLDP 服务，请执行以下步骤。

  a. 将 SMF `auto-enable-agents` 属性更改为 `no`。

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

     该 SMF 属性决定在系统上启用 LLDP 的方式。它有三个可能的值：`yes`、`force` 和 `no`。缺省情况下，该属性设置为 `yes`。有关这些值以及这些值所对应的 LLDP 代理的后续行为的说明，请参见第 103 页中的 “LLDP 的 SMF 属性”。

  b. 重新启动 LLDP 服务。

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

  c. 在选定的端口或链路上启用 LLDP 代理。

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

     其中，*agent* 是 LLDP 代理，通过启用该代理的物理链路进行标识。因此，如果在 `net0` 上启用 LLDP，则代理为 `net0`。

     可将属性 `mode` 设置为以下四个可能值（代表 LLDP 代理的操作模式）之一：`tx`、`rx`、`both` 和 `disable`。有关这些值的说明，请参见第 103 页中的 “LLDP 代理的操作模式”。

  d. 指定 LLDP 代理将通告的 TLV 单元。

     ```
     # lldpadm set-agentprop -p property=value agent
     ```

     有关 LLDP 代理的属性的说明，请参见第 104 页中的 “LLDP 代理通告的信息”。

     要获取 LLDP 代理除 `mode` 属性之外的其他属性的列表，请键入 `lldpadm show-agentprop`。或者，请参阅表 7–1。

有关说明，请参见第 109 页中的 “如何为代理的 LLDP 包指定 TLV 单元”。

3 **如有必要，定制全局 TLV 单元。**

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

其中，*property* 指全局 TLV 单元的属性。

有关全局 TLV 单元的说明，请参见第 105 页中的 “TLV 单元及其属性”。

要获取全局 TLV 的列表，请键入 `lldpadm show-tlvprop`。或者，请参阅表 7–2。

有关说明，请参见第 110 页中的 “如何定义 TLV 值”。

4 **如有必要，定制每代理 TLV 单元。**

```
# lldpadm set-agenttlvprop -p property=value -a agent per-agent-tlv
```

其中，*property* 指每代理 TLV 单元的属性。

有关每代理 TLV 单元的说明，请参见第 105 页中的 “TLV 单元及其属性”。

要获取每代理 TLV 的列表，请键入 `lldpadm show-tlvprop`。或者，请参阅表 7–2。

有关说明，请参见第 110 页中的 “如何定义 TLV 值”。

示例 7–1 定制 auto-enable-agents SMF 属性

以下示例显示了更改 SMF 属性值后的不同 LLDP 启用方式。假定一个系统上有四个端口，按如下所示在两个端口上配置 LLDP：

- `net0`：Rx 和 Tx 模式
- `net1`：仅 Rx 模式
- `net2` 和 `net3`：无

SMF 属性设为缺省值 `yes` 时，会在 `net2` 和 `net3` 上自动启用 LLDP。LLDP 配置显示如下：

```
# 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
```

如果将 SMF 属性切换为 `no`，则当重新启动该服务时，配置会发生变化。

```
# 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
```

在此输出样例中，net2 和 net3 的 LLDP 模式（先前已经自动启用）现在标记为禁用。但是，先前已配置了 LLDP 代理的 net0 和 net1 没有变化。

示例 7–2 在多个数据链路上启用 LLDP

本示例说明如何选择性地启用 LLDP。一个系统具有两个数据链路（net0 和 net1）。在 net0 上，您希望代理传送和接收 LLDP 包。在 net1 上，您希望代理只传送 LLDP 包。可键入以下命令：

```
# 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
```

▼ 如何为代理的 LLDP 包指定 TLV 单元

此过程说明如何在代理传送的 LLDP 包中指定要通告的 TLV 单元。要指定 TLV 单元，请使用 lldpadm set-agentprop 子命令。

1 如有必要，标识可以包含您要添加的 TLV 单元的 LLDP 代理属性。

此子命令还用于显示已为每个属性设置的 TLV 单元。

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

如果不指定属性，此子命令将显示所有 LLDP 代理属性及其 TLV 值。

2 将 TLV 单元添加到属性。

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

+|- 限定符用于接受多个值的属性。这些限定符使您能够在列表中添加 (+) 或删除 (-) 值。如果您不使用限定符，则您设置的值将取代以前为该属性定义的所有值。

3 （可选的）显示属性的新值。

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

示例 7–3 将可选 TLV 单元添加到 LLDP 包

在本示例中，LLDP 代理 net0 已配置为在其 LLDP 包中通告 VLAN 信息。您希望要通告的信息还包括系统功能、链路聚合和网络虚拟化信息。不过，您希望从包中删除 VLAN 说明。

```
# lldpadm show-agentprop net0
AGENT    PROPERTY    PERM    VALUE           DEFAULT    POSSIBLE
net0     mode        rw      both            disable    txonly,rxonly,both,
                                                        disable
net0     basic-tlv   rw      sysname,         none      none,portdesc,
                             sysdesc                    sysname,sysdesc,
                                                        syscapab,mgmtaddr,
                                                        all
net0     dot1-tlv    rw      vlanname,        none      none,vlanname,pvid,
                             pvid,pfc                   linkaggr,pfc,appln,
                                                        evb,etscfg,all
net0     dot3-tlv    rw      max-framesize    none      none, max-framesize,
                                                        all
net0     virt-tlv    rw      none             none      none,vnic,all


# lldpadm set-agentprop -p basic-tlv+=syscapab,dot1-tlv+=linkaggr,virt-tlv=vnic net0
# lldpadm set-agentprop -p dot1-tlv-=vlanname net0
# lldpadm show-agentprop -p net0
AGENT    PROPERTY    PERM    VALUE           DEFAULT    POSSIBLE
net0     mode        rw      both            disable    txonly,rxonly,both,
                                                        disable
net0     basic-tlv   rw      sysname,         none      none,portdesc,
                             sysdesc,                   sysname,sysdesc,
                             syscapab                   syscapab,mgmtaddr,
                                                        all
net0     dot1-tlv    rw      pvid,            none      none,vlanname,pvid,
                             linkaggr                   linkaggr,pfc,appln,
                                                        evb,etscfg,all
net0     dot3-tlv    rw      max-framesize    none      none, max-framesize,
                                                        all
net0     virt-tlv    rw      vnic             none      none,vnic,all
```

▼ 如何定义 TLV 值

此过程说明如何为特定 TLV 单元提供值。使用以下子命令之一：

- `lldpadm set-tlvprop`，用于配置全局 TLV 单元。
- `lldpadm set-agenttlvprop`，用于配置每代理 TLV 单元。

1 根据是要配置全局 TLV 单元还是每代理单元，执行以下步骤之一：

- 要配置全局 TLV 单元，请设置适当的 TLV 属性以包含您要通告的值。

  # lldpadm set-tlvprop -p *tlv-property*=*value*[,*value*,*value*,...] *tlv-name*

  其中，*tlv-name* 是全局 TLV 单元的名称，而 *tlv-property* 是该 TLV 单元的一个属性。可为该属性分配多个值。有关参考信息，请参见表 7–2。

- 要配置每代理 TLV 单元，请配置 LLDP 代理的相应 TLV 属性以包含希望代理通告的值。

  ```
  # lldpadm set-agenttlvprop -p tlv-property[+|-]=value[,value,value,...] -a agent tlv-name
  ```

  其中，*tlv-name* 是代理 TLV 单元的名称，而 *tlv-property* 是该 TLV 单元的一个属性。可为该属性分配多个值。有关参考信息，请参见表 7–3。

2 （可选）通过执行以下步骤之一，显示刚配置的 TLV 属性的值：

- 要显示全局 TLV 属性值，请使用以下命令：

  ```
  # lldpadm show-tlvprop
  ```

- 要显示代理的 TLV 属性值，请使用以下命令：

  ```
  # lldpadm show-agenttlvprop
  ```

示例 7–4 指定系统的功能和管理 IP 地址

本示例实现两个目标：

- 提供有关要在 LLDP 包中通告的系统功能的特定信息。为了实现此目标，必须同时配置 `syscapab` TLV 单元的 `supported` 和 `enabled` 属性。
- 提供在通告中使用的管理 IP 地址。

```
# llpdadm set-tlvprop -p supported=bridge,router,repeater syscapab
# llpdadm set-tlvprop -p enabled=router syscapab
# llpdadm set-tlvprop -p ipaddr=192.168.1.2 mgmtaddr
# llpdadm show-tlvprop
TLVNAME    PROPERTY   PERM  VALUE        DEFAULT         POSSIBLE
syscapab   supported  rw    bridge,      bridge,router,  other,router,
                            router,      station         repeater,bridge,
                            repeater                     wlan-ap,telephone,
                                                         docis-cd,station,
                                                         cvlan,svlan,tpmr
syscapab   enabled    rw    router       none            bridge,router,
                                                         repeater
mgmtaddr   ipaddr     rw    192.162.1.2  none            --
```

禁用 LLDP

要在各个端口上选择性地禁用 LLDP，请使用以下命令之一：

- `lldpadm set-agentprop -p mode=disable` *agent*

  其中，*agent* 是 LLDP 代理，通过启用该代理的物理链路进行标识。因此，如果在 `net0` 上启用 LLDP，则代理为 `net0`。此命令通过更改代理的模式禁用 LLDP。

- `lladpadm reset-agentprop`

  在此命令中，不设置 `mode` 属性的值。此命令通过从端口中删除 LLDP 配置来禁用 LLDP。

注意 – 子命令 `lldpadm reset–agentprop` 将从端口中彻底删除 LLDP 配置。如果设置为 `no` 的 `auto-enable-agents` 切换回 `yes`，则 LLDP 的行为与单纯禁用该端口上的代理模式时不同。

要在所有系统接口中全局性地禁用 LLDP，请执行以下步骤。

1. 将 SMF LLDP 属性更改为 `no`。

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

2. 重新启动 LLDP 服务。

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

3. 在保留了先前 LLDP 配置的各个端口上禁用 LLDP。

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

监视 LLDP 代理

`lldpadm show-agent` 子命令显示由 LLDP 代理通告的完整信息。相对于给定系统，通告可以是传送到网络其余部分的有关本地系统的信息。通告也可以是系统接收的来自同一网络上的其他系统的信息。

本节介绍两个过程：

- 第 112 页中的“如何显示通告”
- 第 114 页中的“如何显示 LLDP 统计信息”

▼ 如何显示通告

此过程说明如何显示由 LLDP 代理通告的信息。信息可以是本地信息，也可以是远程信息。**本地**信息来自本地系统。**远程**信息来自网络上的其他系统，由本地系统接收。

- **将 `lldpadm show-agent` 子命令和适当的选项结合使用可显示所需的信息。**

  - **要显示由 LLDP 代理通告的本地信息，请键入以下命令：**

    ```
    # lldpadm show-agent -l agent
    ```

  - **要显示 LLDP 代理接收的远程信息，请键入以下命令：**

    ```
    # lldpadm show-agent -r agent
    ```

  - **要详细显示本地或远程信息，请键入以下命令：**

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

示例 7–5 获取通告的 LLDP 代理信息

以下示例说明如何显示 LLDP 代理以本地或远程方式通告的信息。缺省情况下，信息以短格式显示。通过使用 -v 选项，您可以获取详细信息。

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

# lldpadm show-agent -lv net0
                    Agent: net0
         Chassis ID Subtype: Local(7)
            Port ID Subtype: MacAddress(3)
                    Port ID: 00:14:4f:01:77:5d
           Port Description: net0
               Time to Live: 81 (seconds)
                System Name: hosta.example.com
         System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
     Supported Capabilities: bridge,router
       Enabled Capabilities: router
         Management Address: 192.168.1.2
         Maximum Frame Size: 3000
               Port VLAN ID: --
               VLAN Name/ID: vlan25/25
        VNIC PortID/VLAN ID: 02:08:20:72:71:31
    Aggregation Information: Capable, Not Aggregated
                PFC Willing: --
                    PFC Cap: --
                    PFC MBC: --
                 PFC Enable: --
                PFC Pending: --
Application(s) (ID/Sel/Pri): --
    Information Valid Until: 117 (seconds)

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

# llpdadm show-agent -rv net0
                     Agent: net0
         Chassis ID Subtype: Local(7)
            Port ID Subtype: MacAddress(3)
                    Port ID: 00:14:4f:01:59:ab
           Port Description: net0
               Time to Live: 121 (seconds)
                System Name: hostb.example.com
         System Description: SunOS 5.11 dcb-clone-x-01-19-11 i86pc
     Supported Capabilities: bridge,router
       Enabled Capabilities: router
         Management Address: 192.168.1.3
         Maximum Frame Size: 3000
               Port VLAN ID: --
               VLAN Name/ID: vlan25/25
        VNIC PortID/VLAN ID: 02:08:20:72:71:31
    Aggregation Information: Capable, Not Aggregated
                PFC Willing: --
                    PFC Cap: --
                    PFC MBC: --
```

```
                PFC Enable: --
Application(s) (ID/Sel/Pri): --
    Information Valid Until: 117 (seconds)
```

▼ 如何显示 LLDP 统计信息

您可以显示 LLDP 统计信息，以获取有关由本地系统或远程系统通告的 LLDP 包的信息。统计信息引用涉及 LLDP 包传送和接收的重大事件。

1 **要显示有关 LLDP 包传送和接收的所有统计信息，请使用以下命令：**

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

2 **要显示所选的统计信息，请使用 -o 选项。**

```
# lldpadm show-agent -s -o field[,field,...]agent
```

其中 *field* 指 `show-agent -s` 命令的输出中的任何字段名称。

示例 7–6 显示 LLDP 包统计信息

本示例说明如何显示有关 LLDP 包通告的信息。

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

命令输出提供以下信息：

- `AGENT` 指定 LLDP 代理的名称，它与在其上启用 LLDP 代理的数据链路相同。
- `IFRAMES`、`IEER` 和 `IDISCARD` 显示有关正在接收的包、有错误的传入包和丢弃的传入包的信息。
- `OFRAMES` 和 `OLENERR` 指传出包以及有长度错误的包。
- `TLVDISCARD` 和 `TLVUNRECOG` 显示有关被丢弃的和未识别的 TLV 单元的信息。
- `AGEOUT` 指已超时的包。

该示例表明在系统收到的 9 个帧中，有 5 个 TLV 单元无法识别，原因可能是不符合标准。该示例还显示本地系统向网络传送了 14 个帧。

◆ ◆ ◆ 第 8 章

使用 Oracle Solaris 中的数据中心桥接功能

与第 7 章，使用 LLDP 交换网络连接信息中所介绍的 LLDP 类似，数据中心桥接 (data center bridging, DCB) 也涉及与网络上对等方的信息交换。信息指影响网络包完整性的配置，尤其是在通信流量很大的环境（如数据中心）中。DCB 通过协调与这些中心中的高速通信相关的组件配置，实现高效的网络通信流量交换。

本章包含以下主题：

数据中心桥接 (Data Center Bridging, DCB) 概述

数据中心桥接是一组可增强传统以太网功能以管理通信的功能，尤其适用于网络通信流量和传输率都很高的环境中。光纤通道可专用于承载此类型的通信。但是，如果使用专用链路来仅提供光纤通道通信，则成本可能会很高。因此，更多情况下使用以太网光纤通道 (fiber channel traffic over Ethernet, FCoE)。DCB 功能可满足光纤通道对遍历以太网时包丢失的敏感度要求。

DCB 允许对等方基于优先级区分通信。通过区分优先级，主机可确保在主机之间发生拥塞时保持较高优先级通信的包完整性。使用 DCB 交换 (DCB exchange, DCBX) 协议，通信主机可以交换会影响高速网络通信的配置信息。然后，对等方可对公用配置进行协商，确保通信流不中断，同时防止高优先级包出现包丢失。

在 Oracle Solaris 中，LLDP 用于交换 DCBX TLV 单元。如果底层 NIC 支持 DCB，则可以与网络上的对等主机共享 DCB 功能，如基于优先级的流量控制 (priority-based flow control, PFC) 和增强传输选择 (enhanced transmission selection, ETS)。

- 基于优先级的流量控制 (priority-based flow control, PFC) 通过实施一种机制来防止包丢失，该机制可暂停具有定义的服务类 (class of service, CoS) 优先级的包的通信流。请参见第 117 页中的 “基于优先级的流量控制”。有关 CoS 的更多信息，请参阅 `dladm`(1M) 手册页中有关 `cos` 链路属性的说明。

- 增强传输选择 (enhanced transmission selection, ETS) 支持基于定义的 CoS 优先级在包之间共享带宽。请参见第 121 页中的 “增强传输选择”。

与使用 LLDP 交换的所有系统信息一样，在主机上存在两种类型的 DCB 信息：本地 DCB 信息和远程 DCB 信息。要使 PFC 功能生效，主机上用于 PFC 的这两种 DCB 信息必须对称。通常，本地主机必须能够匹配它从对等方接收到的 DCB 信息。在启用 DCB 的 Oracle Solaris 系统中，也将启用与对等方同步 DCB 信息的功能。

注 – 仅当物理 NIC 支持 DCB 时，才可以在 Oracle Solaris 11 系统上使用 DCB 功能。此外，必须将该卡配置为在 DCB 模式下运行。

▼ 如何启用 DCBX

启用 LLDP 时，将自动启用对 DCBX 的支持。此过程提供了备用的手动步骤，以防某些自动过程失败。在此过程中，假定对 `net0` 执行相应步骤。

1 安装 LLDP 软件包。

```
# pkg install lldp
```

2 验证 LLDP 服务是否正在运行。

```
# svcs lldp
```

如果已禁用 LLDP 服务，请使用以下命令启动该服务：

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

3 确保 LLDP 代理正在以 Rx 和 Tx 模式运行。

```
# lldpadm show-agentprop -p mode net0
```

如果并非以这两种模式启用 LLDP 代理，请键入以下命令：

```
# lldpadm set-agentprop -p mode=both net0
```

有关更多信息，请参见第 103 页中的 “LLDP 的 SMF 属性”。

有关 LLDP 代理的其他可能配置，请参见第 106 页中的 “在系统上启用 LLDP”。

4 确保底层 NIC 支持 DCB。

```
# dladm show-linkprop -p ntcs net0
```

大于零 (`0`) 的属性值表示 NIC 支持 DCB。

基于优先级的流量控制

PFC 扩展了标准 PAUSE 帧以包含 IEEE 802.1p CoS 值。使用 PFC，当发送 PAUSE 帧时不会停止链路上的所有通信，而是仅暂停 PFC 帧中启用的 CoS 值所对应的通信。为启用的优先级（具备此优先级的通信需要暂停）发送一个 PFC 帧。发送主机将停止该优先级的通信，而其他禁用的优先级的通信不受影响。经过 PFC 帧中指定的时间间隔之后，或者在发送主机接收到另一个 PFC 帧之后，将恢复这些包的传输。基于优先级暂停可确保不会丢弃该优先级的包。对于未定义任何优先级的包，不会发送 PAUSE 帧。因此，通信将继续进行，但在通信拥塞时可能会丢弃包。

优先级通过 `pfcmap` 数据链路属性中的 8 位掩码 (0–7) 表示。最低位表示优先级 0，而最高位表示优先级 7。此掩码中每位均可表示是否已为对应的优先级启用 PFC。缺省情况下，`pfcmap` 设置为 `1111111`，表示已为所有优先级启用 PFC。对于通过链路传输的任何包，如果在接收主机上产生拥塞，则将向发送主机发送 PFC 帧。

与 PFC 相关的数据链路属性

除了 `pfcmap` 属性，以下属性也提供有关优先级定义和映射的信息：

- `pfcmap-lcl-effective` 指本地主机上的有效 PFC 映射。此属性具有只读权限。此属性可反映 `pfcmap` 属性或 `pfcmap-rmt-effective` 属性的值。
- `pfcmap-rmt-effective` 指远程对等方上的有效 PFC 映射。此属性也具有只读权限。

要使 PFC 帧能够正常发送，通信主机必须具有对称的 DCB 配置信息。Oracle Solaris 11 系统可以自动调整其 PFC 配置，以匹配远程对等方的 PFC 配置。

以上列出的两种属性间接指示对等方之间的 PFC 信息是否同步。对于本地和远程对等方之间 PFC 信息匹配的数据链路，不管将 `pfcmap` 设置为何值，`pfcmap-lcl-effective` 和 `pfcmap-rmt-effective` 的值均相同。如果在本地主机上禁用了同步功能，则 `pfcmap-lcl-effective` 将反映本地主机的 `pfcmap` 属性的值。

有关这些属性配置提供的 PFC 信息的示例，请参见第 118 页中的 “获取 PFC 配置信息”。

基于优先级的流量控制 TLV 单元

PFC TLV 单元控制与从对等主机接收的信息有关的主机行为。此 TLV 单元只有一个可配置属性 `willing`。缺省情况下，该属性设置为 `on`，从而使本地主机可以将其 PFC 优先级定义与远程对等方上的 PFC 定义同步。通过将该属性切换为 `off`，可以防止自动同步特定代理的信息，如下所示：

```
# lldpadm set-agenttlvprop -p willling=off -a agent pfc
```

其中，*agent* 通过启用代理的数据链路进行标识。

▼ 如何为 DCB 定制基于优先级的流量控制

大多数情况下，PFC 的缺省配置已足够。启用 LLDP 时会自动设置此配置。但是，为了显示配置 PFC 时可以使用的其他选项，此过程列出了手动配置 PFC 的步骤。这些步骤假定不存在任何自动配置。为了便于理解这些步骤，将对 `net0` 执行所有配置。

1. **确保已启用 DCBX。**

   请参见第 116 页中的 “如何启用 DCBX”。

2. **（可选）定制您要启用的 DCB 功能。**

   缺省情况下，PFC、ETS 和边缘虚拟桥接 (edge virtual bridging, EVB) 已启用。假定您希望仅使用 PFC。那么，您必须从 LLDP 代理的 `dot1–tlv` 属性中删除其他两个值。有关 `dot1–tlv` 的可能值的列表，请参阅表 7–3。

   ```
   # lldpadm set-agenttlvprop -p dot1-tlv-=etscfg,evb net0
   ```

3. **确保数据链路的 `flowctrl` 属性设置为 `pfc`。**

   ```
   # dladm show-linkprop -p flowctrl net0
   ```

   如果该属性的值列表中不包含 `pfc`，请发出以下命令：

   ```
   # dladm set-linkprop -p flowctrl=pfc net0
   ```

4. **如果不想使用缺省值 `11111111`，请根据需要设置 `pfcmap` 属性。**

   例如，要仅启用 CoS 优先级 6，请键入以下命令：

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

5. **确保主机可以将其 PFC 信息与远程对等方上的 PFC 信息同步。**

   ```
   # lldpadm show-agenttlvprop -p willing -a net0 pfc
   ```

   如果 PFC TLV 属性 `willing` 设置为 `off`，请发出以下命令：

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

获取 PFC 配置信息

本节包含配置 LLDP 和 DCB 后与 PFC 相关的信息的多个示例。

以下命令显示与 PFC 相关的信息：

- `dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective` *datalink*
  此命令显示优先级定义以及数据链路上的有效 PFC 映射。
- `dladm show-phys -D pfc` *datalink*
  此命令显示物理链路上与该 NIC 上启用的优先级有关的 PFC 信息。
- `lldpadm show-agenttlvprop -a` *agent* `pfc`

其中，*agent* 通过启用 LLDP 的数据链路进行标识。因此，LLDP 代理的名称与数据链路的名称相同。此命令显示 PFC TLV 属性，该属性控制主机将其 PFC 映射与对等方同步的功能。

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

  此命令提醒您本地主机和对等方之间的 PFC 映射信息不匹配。

以下示例说明先前列出的命令所显示的信息类型。

示例 8–1 显示与 PFC 相关的数据链路属性

此示例说明如何显示与基于优先级的流量控制相关的数据链路属性的状态。

```
# dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective net0
LINK    PROPERTY               PERM    VALUE      DEFAULT    POSSIBLE
net0    pfcmap                 rw      11111111   11111111   00000000-11111111
net0    pfcmap-lcl-effective   r-      11111111   --         --
net0    pfcmap-rmt-effective   r-      01000000   --         --
```

该输出指示本地主机上的 PFC 映射使用缺省值，其中所有 8 个优先级都已启用。`pfcmap-lcl-effective` 和 `pfcmap-rmt-effective` 的值不匹配表示本地主机未将其 PFC 信息与远程对等方同步。此不匹配可能是由允许关闭同步的属性导致的。或者，对等方没有将 PFC TLV 单元发送到网络中。可以通过键入以下命令确认此配置：

示例 8–2 显示本地主机同步 PFC 信息的功能

此示例说明如何显示主机用于适应对等方 PFC 配置的功能的当前状态。

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

要启用同步，请发出以下命令：

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

# dladm show-linkprop -p pfcmap,pfc-lcl-effective,pfc-rmt-effective net0
LINK    PROPERTY               PERM    VALUE      DEFAULT    POSSIBLE
net0    pfcmap                 rw      11111111   11111111   00000000-11111111
net0    pfcmap-lcl-effective   r-      01000000   --         --
net0    pfcmap-rmt-effective   r-      01000000   --         --
```

在第二个输出中，本地主机丢弃了自己的 PFC 映射 (11111111)，而是与对等方同步，现在其生效 PFC 映射与对等方的 PFC 映射完全一致。通过这种值的聚合，主机之间可以成功交换 PFC PAUSE 帧。

示例 8–3 验证主机和对等方之间的 PFC 信息的对称性

此示例说明如何在实际运行时验证主机和对等方之间的 PFC 信息是否同步，或者是否发生不匹配。

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

示例 8–3 验证主机和对等方之间的 PFC 信息的对称性 （续）

如果主机和对等方之间的 PFC 信息未聚合，则 `PFC Pending` 会返回 `True` 状态。解决不匹配问题之后，`PFC Pending` 的状态将恢复为 `False`。

要显示代理所通告的所有信息，请使用 `lldpadm show-agent` 命令的 `verbose` 选项：

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

示例 8–4 显示 CoS 优先级定义

此示例说明如何显示特定数据链路上基于 `pfcmap` 属性值的当前 CoS 优先级定义。例如，假定 `pfcmap` 配置为 `01000000`。要显示物理链路上相应的优先级映射，请执行如下命令：

```
# dladm show-phys -D pfc net0
LINK        COS   PFC   PFC_EFFECT   CLIENTS
ixgbe0      0     YES           NO   net0,vnic1
            1     YES          YES   vnic2
            2     YES           NO   vnic3
            3     YES           NO   vnic4
            4     YES           NO   vnic5
            5     YES           NO   vnic6
            6     YES           NO   vnic7
            7     YES           NO   vnic8
```

对于物理链路 `net0`，为数据链路上配置的所有 VNIC 客户机启用了优先级。但是，本地主机将其 PFC 映射调整为对等方的 PFC 映射，如 `PFC_EFFECT` 字段的值所示，其中，已对 CoS `0` 和 `2-7` 禁用优先级。因此，不会为除 `vnic2` 之外的任何 VNIC 上的通信交换 PFC 帧，无论资源是否可用。此配置允许对流经 `vnic2` 之外的任何 VNIC 的通信丢弃包。对于 `vnic2` 上的通信，在出现通信拥塞时将发送 PFC PAUSE 帧，以防止此客户机上发生包丢失。

应用程序 TLV 单元

应用程序 TLV 单元包含有关要用于主机上某应用程序的优先级的信息。该优先级在应用程序优先级表中进行定义。表中的每一个条目都包含应用程序的名称以及指定给该应用程序的优先级。应用程序 TLV 使用此表向其他主机传输应用程序优先级信息。

表中的条目使用以下格式：

protocol-id/*selector*/ *priority*

protocol-id/*selector* 对标识应用程序。*Priority* 包含 0 到 7 之间的值，用于标识相应应用程序的优先级。

要与其他主机交换此有关某应用程序优先级的信息，可按如下所示设置应用程序 TLV：

```
# lldpadm set-agenttlvprop -p property=value -a agent appln
```

例如，对于 FCoE 通信，协议 ID 为 `0x8906`，选择器 ID 为 `1`。假定为此应用程序指定优先级 `4`。根据表 7–3（列出了用于设置应用程序 TLV 的参数），键入以下命令：

```
# 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   --        --
```

增强传输选择

ETS 是一项 DCB 功能，该功能允许根据应用程序的 DCB 优先级为其分配 NIC 上的带宽。DCB 优先级是包含一个 3 位优先级字段的 VLAN 头。优先级字段的值用于区分网络中的以太网包。DCB 使用该优先级值（也称为 802.1p 优先级）将通信与其他 DCB 属性（如 PFC 配置和链路带宽）关联。将 DCB 配置为根据包的优先级值设置要分配给包的特定带宽。

要使用 ETS，NIC 必须支持 DCB 并在 DCB 模式下运行。

与 ETS 相关的数据链路属性

引用 PFC 信息的数据链路属性适用于基于为包定义的 CoS 优先级来防止包丢失。引用 ETS 信息的属性适用于基于相同的 CoS 优先级为包分配共享带宽。可通过以下数据链路属性配置 ETS：

- `cos` 指定数据链路的服务类。该属性表示以太网优先级。该属性的值（介于 0 到 7）应用于数据链路上的传出包。该值是在传出包的 VLAN 标记中设置的。如果对物理链路本身设置此属性，则优先级仅应用于该链路的主客户机上的通信。不会在其他辅助客户机（如 VNIC）上设置此优先级。缺省情况下，如果 NIC 在 DCB 模式下运行，或者链路为 VLAN，则将 `cos` 设置为 `0`。
- `etsbw-lcl` 指示为数据链路的 TX 端分配的 ETS 带宽。仅当底层物理 NIC 具有 DCB 功能并支持 ETS 时，此属性才可配置。可通过指定要分配给辅助数据链路或客户机的底层 NIC 总带宽的百分比来设置该值。只要该链路的 `cos` 未设置为零 (`0`)，就可以设置该属性。

注 – 配置为聚合且在 DCB 模式下运行的物理链路当前不支持 ETS。

在 `etsbw-lcl` 中定义的带宽百分比不是仅用于辅助客户机的保留量。如果分配的带宽未使用，则该带宽可由具有类似配置的其他客户机使用。此外，仅在主机通信的传输端强制进行带宽分配。

除了前面列表中的属性，以下只读属性也提供有关本地主机与其对等方之间所交换的带宽数据的信息：

- `etsbw-lcl-advice` 指定建议的带宽份额。此数据链路建议带宽由远程对等方发送至本地主机。
- `etsbw-lcl-effective` 指在本地主机的数据链路上实施的实际带宽份额。此属性可反映 `etsbw-lcl` 属性或 `etsbw-lcl-advice` 属性的值。
- `estbw-rmt-effective` 指在远程对等方中配置的带宽份额。

对于具有特定优先级的包要使用的相应带宽，最好使通信主机之间的 ETS 信息对称或进行同步。确切地说，本地系统应该能够将其带宽份额调整为 `etsbw-lcl-advice` 的值。Oracle Solaris 11 系统可以自动调整其 ETS 配置，以匹配远程对等方的 ETS 配置。

`estbw-lcl-effective` 属性间接指示本地主机用于与对等方匹配 ETS 信息的功能是否已启用。如果此属性的值匹配 `etsbw-lcl-advice` 的值，则该功能已启用。否则，`etsbw-lcl-effective` 和 `etsbw-lcl` 属性的值将相同。

增强传输选择 TLV 单元

ETS TLV 单元 `etscfg` 控制与从对等主机接收的信息有关的主机行为。此 TLV 单元只有一个可配置属性 `willing`。缺省情况下，此属性设置为 `on`，从而使本地主机可以将其 ETS 配置与远程对等方的 ETS 配置同步。如果需要防止同步某特定代理的信息，请将 `willing` 属性设置为 `off`，如下所示：

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

其中，*agent* 通过启用代理的数据链路进行标识。

▼ 如何为 DCB 定制增强传输选择

大多数情况下，系统上的缺省 ETS 配置已足够。如果启用了 LLDP，底层链路支持 DCB，并且底层链路在 DCB 模式下运行，则将自动设置此配置。但是，为了显示配置 ETS 时可以使用的其他选项，此过程列出了手动配置 ETS 的步骤。这些步骤假定不存在任何自动配置，并且在虚拟客户机 `vnic1` 上执行配置。在 LLDP 代理 `net0` 上配置虚拟客户机。

1 确保已启用 DCBX。

请参见第 116 页中的 “如何启用 DCBX”。

2 （可选）定制您要启用的 DCB 功能。

缺省情况下，PFC、ETS 和边缘虚拟桥接 (edge virtual bridging, EVB) 已启用。假定您希望禁用 EVB。那么，从 LLDP 代理的 `dot1-tlv` 属性中删除其他两个值。

```
# lldpadm set-agenttlvprop -p dot1-tlv-=evb net0
```

3 为 VNIC 设置 CoS 优先级定义。

```
# dladm set-linkprop -p cos=value vnic1
```

4 **设置 VNIC 带宽占物理链路总带宽的比重。**

```
# dladm set-linkprop -p etsbw-lcl=value vnic1
```

指定给 `etsbw-lcl` 属性的值表示底层链路的带宽总量的百分比。为客户机指定的所有已分配带宽值的总和不得超过 100%。

5 **验证主机可以将其 ETS 信息与远程对等方的 ETS 信息同步。**

```
# lldpadm show-agenttlvprop -p willing -a net0 etscfg
```

如果 `willing` 属性设置为 `off`，请发出以下命令：

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

获取 ETS 配置信息

本节包含配置 LLDP 和 DCB 后与 ETS 配置相关的信息的多个示例。

以下命令显示有关 ETS 配置的信息：

- `dladm show-linkprop -p etsbw-lcl,etsbw-advise,etsbw-lcl-effective,etsbw-rmt-effective` *datalink*

  此命令显示带宽分配定义以及数据链路上实施的有效分配。

- `dladm show-phys -D ets` *datalink*

  此命令显示物理链路上与链路间的带宽分配和分发有关的 ETS 配置。

- `lldpadm show-agenttlvprop -a` *agent* `etscfg`

  其中，*agent* 通过启用 LLDP 的数据链路进行标识。此命令显示 ETS TLV 属性，该属性控制主机将 ETS 信息与对等方同步的功能。

以下示例说明列出的命令所显示的信息类型。

示例 8–5 显示与 ETS 相关的数据链路属性

此示例说明如何显示与增强传输选择相关的数据链路属性的状态。

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

该输出显示 `vnic1` 配置为使用物理链路总可用带宽的 20% 带宽份额。VNIC 的 802.1p 优先级（通过 `cos` 属性指示）设置为二。

示例 8–6 显示本地主机同步 ETS 信息的功能

此示例说明如何显示本地主机用于适应对等方的 ETS 配置的功能的当前状态。

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

要启用同步，请发出以下命令：

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

# dladm show-linkprop -p etsbw-lcl,etsbw-lcl-advise, \
etsbw-lcl-effective,etsbw-rmt-effective vnic0
LINK    PROPERTY              PERM  VALUE      DEFAULT    POSSIBLE
vnic1   cos                   rw    2          0          0-7
vnic1   etsbw-lcl             rw    20         0          --
vnic1   etsbw-lcl-advise      r-    15         --         --
vnic1   etsbw-lcl-effective   r-    15         --         --
vnic1   etsbw-rmt-effective   r-    25         --         --
```

尽管 `vnic1` 的 `estbw-lcl` 设置为 20%，但为匹配从对等方接收到的建议带宽，该 VNIC 的有效带宽份额为 15%。此调整是由于将 `etscfg` TLV 单元的 willing 属性切换为 on。

以下示例说明物理链路上的优先级映射：

```
# dladm show-phys -D ets net0
LINK       COS   ETSBW   ETSBW_EFFECT   CLIENTS
ixgbe0     0     20      20             <default,mcast>,net0
           1     15      15             vnic2
           2     20      20             vnic1
           3     30      30             vnic5
           4     15      15             vnic3
           5     0       0              vnic4
           6     0       0              vnic6
           7     0       0              vnic7
```

在此示例中，为不同的 VNIC 设置了各自对应的 `cos` 值。根据之前的输出，`vnic1` 的 `cos` 属性设置为二。在 `ETSBW` 字段下，客户机 `vnic1` 的有效带宽份额为 15% 以匹配从对等方接收到的建议值（显示在 `ETSBW_EFFECT` 字段下）。此示例还显示最大份额的带宽分配给了 `vnic5`。请注意，分配给 `vnic4`、`vnic6` 和 `vnic7` 的带宽为 0% 并不表示这些客户机完全没有带宽份额。更确切地说，如果其他客户机正在使用分配给这些客户机的带宽，则这些客户机不会接收到带宽。

◆ ◆ ◆ 第 9 章

9 Oracle Solaris 中的边缘虚拟桥接

本章介绍了边缘虚拟桥接 (edge virtual bridging, EVB) 功能。EVB 进一步扩展了以下各章所述的信息交换功能：

- 第 7 章，使用 LLDP 交换网络连接信息
- 第 8 章，使用 Oracle Solaris 中的数据中心桥接功能

通过 EVB 可交换与系统上虚拟链路相关的信息。本章包含以下主题：

边缘虚拟桥接概述

边缘虚拟桥接是不断发展的 IEEE 标准，用于供主机与外部交换机交换虚拟链路信息。通过 EVB，可以在网络上通告更多有关虚拟链路配置的信息，例如 DCB 功能提供的物理链路的带宽共享或优先级定义等，不一而足。

总的来说，EVB 可用于以下操作：

- 在外部网桥端口上启用反射中继。请参见第 126 页中的 “反射中继功能”。
- 自动在网桥上配置虚拟端口。请参见第 126 页中的 “自动在网桥上配置虚拟端口”。

要了解 EVB 机制，请注意 EVB 使用的以下术语。

- **站**指系统或主机。
- **网桥**指连接到站的外部交换机。
- **虚拟站实例 (virtual station instance, VSI)**指站中配置的 VNIC。
- **虚拟机 (virtual machine, VM)** 是一个通用术语，指系统上的区域、Oracle VM VirtualBox 以及其他通过软件实现的计算机。

以下各节详细介绍了 EVB 的功能。

反射中继功能

借助于网络虚拟化，可通过单个物理 NIC 为站配置多个虚拟网络接口卡 (virtual network interface card, VNIC)。这些 VNIC 将分配给站中的虚拟机。在此设置中，包无需离站即可在虚拟机之间通信，而是由物理链路的虚拟交换机将包从一个 VM 路由到另一 VM。有关包含多个 VNIC、一个虚拟交换机和多个虚拟机的系统配置的说明，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》中的“网络虚拟化组件”。

在某些情况下，VM 之间的内部通信可能需要使用外部网桥。例如，内部通信可能需要遵守外部网桥上配置的访问控制列表 (access control list, ACL)。因此，来自源 VM 的包可能通过外部网桥的端口退出站。然后，这些包将从网桥发送回站中的接收 VM。

缺省情况下，网桥不能使用接收包的端口发送包。因此，要在使用外部网桥的 VM 之间进行通信，网桥必须具有反射中继功能。通过该功能，网桥可使用它接收包的同一链路将来自发送 VM 的包转发到接收 VM。

EVB 定义了新的特定于组织的 LLDP TLV 单元，以通知网络对等方有关反射中继功能的信息，同时 EVB TLV 单元还充当信息媒介。信息交换通过一个站请求网桥启用反射中继功能（如果网桥支持该功能）而启动。然后，网桥启用该功能（如果支持），并通知发出请求的主机已启用该功能。如果网桥不支持反射中继，则会向站发送回禁用状态。在这种情况下，虚拟机之间的通信只能使用物理链路上的虚拟交换机。

自动在网桥上配置虚拟端口

通过 LLDP 和 DCBX，站可以与最接近的网桥交换配置信息。通过此交换，网桥可以检测诸如为站上的通信类定义的优先级之类的信息。根据该信息，可自动将网桥配置为根据包的 802.1p 优先级值对其进行处理。

如果没有信息交换和自动配置，则需要手动单独（不能与站一起）配置网桥以反映站配置中的通信类优先级。手动配置将增加网桥配置错误风险，并且可能导致站和网桥之间出现不一致。

使用 EVB 可将交换机制扩展为包括有关站用于网桥的 VSI 的信息。通过这种方式，可将 VNIC 配置扩展到网桥。例如，假定已为 VNIC 配置特定的带宽限制。通过 EVB，网桥可对目标为该 VNIC 的包实施该带宽限制。

用于 VSI 信息交换的 EVB 组件

站可通过以下 EVB 组件将 VSI 信息通告到网桥：

- **VSI 配置文件**，包括为特定 VNIC 配置的链路属性。因此，站最多可以拥有与已配置的 VNIC 同等数量的 VSI 配置文件。
- VSI 标识符，由一个 **VSI 类型 ID** 和 **VSI 版本 ID** 对组成，可唯一标识一个 VSI 配置文件。
- **VSI 管理器**，通过将 VSI 类型 ID-VSI 版本 ID 标识符与一组特定的 VNIC 属性相对应来管理站的多个 VSI 配置文件。

- **VSI 管理器 ID**，用于标识与特定 VSI 类型 ID - VSI 版本对相关的 VSI 管理器。VSI 管理器 ID 表示为 IPv6 地址。

VSI 管理器 ID、VSI 类型 ID 和 VSI 版本组合起来构成一个元组，用于标识特定 VNIC 的属性集。

VSI 信息使用 VSI 搜索和配置协议 (VDP) 进行交换，而 VDP TLV 单元充当信息媒介。网桥从站接收 VDP TLV 单元。然后，网桥使用 TLV 单元中包含的元组获取与 VSI 关联的属性集。获取 VSI 配置文件或类型的属性后，网桥可将属性配置应用到该 VSI 的包。

必须满足以下要求才能将 VSI 信息通告到网桥：

- 站必须了解发送 VSI 搜索协议请求时要使用的 VSI 管理器 ID。
- 网桥必须能够识别并支持站发送的 VSI 管理器 ID。

Oracle Solaris 中的 EVB支持

目前没有用于定义 VSI 配置文件（例如应包含在配置文件中的特定属性）的既定标准。此外，VSI 类型的定义已紧密链接到 VSI 管理器 ID（通常特定于供应商）。

Oracle Solaris 使用 3 字节编码 `oracle_v1` 定义 VSI 管理器。此 VSI 管理器支持以下数据链路属性：

- 带宽限制
- 底层链路的链路速度
- 通信类
- VNIC 的最大传输单元 (maximum transmission unit, MTU)

`oracle_v1` 编码定义如下：

位	属性
0-4	**链路带宽限制** 00000-10100：链路速度的 0-100% 范围，以 5% 递增。 其余：保留

位	属性
5–7	**链路速度**
	000－未知
	001－10 Mbps
	010－100 Mbps
	011－1 Gbps
	100－10 Gbps
	101－40 Gbps
	110－100 Gbps
	111－保留
8–12	**保留**
13-15	**通信类 (0-7)**
16-17	**链路 MTU**
	00－1500 字节
	01－9000 字节
	10－定制
	11－保留

在 Oracle Solaris 中直接将 3 字节编码用作 VSI 类型 ID。

因此，在 Oracle Solaris 中通告到网桥的元组是 Oracle VSI 管理器和组合的 VSI 类型 ID-VSI 版本 ID 对。VSI 信息交换机制遵循第 126 页中的 “用于 VSI 信息交换的 EVB 组件”中所述的过程。配置网桥以识别 Oracle VSI 管理器。然后，网桥使用 Oracle VSI 管理器 ID 和 VSI 类型 ID-VSI 版本 ID 组合获取与 VSI 配置文件关联的属性集。获取属性信息后，网桥可将属性配置应用到该 VNIC 的包。

在传输 VSI 管理器 ID TLV 之后，发送 Oracle 提供的特定于组织的 OUI TLV 单元。OUI TLV 指示它所遵循的用于 VSI 管理器 ID 的任何编码（如果有）。如果网桥识别 Oracle 定义的 VSI 管理器 ID，则网桥在回复发出请求的站时将包含该 TLV 单元。网桥响应中缺少特定于 Oracle 的 TLV 单元表示该交换机未识别也不支持 Oracle VSI 管理器。

与 EVB 相关的数据链路属性

以下是与 EVB 相关的可配置数据链路属性的列表：

- `vsi-mgrid` 指定为物理链路或 VNIC 设置的 VSI 管理器 ID。在 Oracle Solaris 中，该属性与缺省的 VSI 管理器 ID `ORACLE_VSIMGR_V1` 相关联。

如果希望使用 IPv6 地址，您还必须定义 VSI 类型 ID 和 VSI 版本 ID。否则，Oracle Solaris 无法识别元组。此外，还必须手动配置与 VSI 类型 ID-VSI 版本 ID/VSI 管理器 ID 元组对应的适当数据链路属性。

使用 EVB 时最好使用缺省的 Oracle VSI 管理器 ID。这样，Oracle VSI 管理器可自动为站的 VSI 配置文件生成 VSI 类型 ID 和 VSI 版本 ID。

- `vsi-mgrid-enc` 指示与 VSI 管理器 ID 关联的编码。缺省情况下，该属性设置为 `oracle_v1`。如果不希望将 `oracle_v1` 与 VSI 管理器 ID 关联，请将该属性的值设置为 `none`。
- `vsi-typeid` 指定 VSI 类型 ID。VSI 类型 ID 和要与 VSI 配置文件关联的 VSI 版本 ID 成对。如果 `vsi-mgrid` 和 `vsi-mgrid-enc` 使用缺省值，则将自动生成该 3 字节值。否则，您必须明确地为该属性指定一个值。
- `vsi-vers` 指定 VSI 版本 ID。VSI 版本 ID 和要与 VSI 配置文件关联的 VSI 类型 ID 成对。如果 `vsi-mgrid` 和 `vsi-mgrid-enc` 使用缺省值，则将自动生成该 1 字节值。否则，您必须明确地为该属性指定一个值。

注 – 可为所有 VNIC 手动配置所有这些属性，但只能为物理链路配置 `vsi-mgrid` 和 `vsi-mgrid-enc` 属性。

除了以上列表中的属性，以下只读属性也提供有关系统上实际实施的 EVB 配置的信息：

- `vsi-mgrid-effective` 指定虚拟链路或 VNIC 上的 VSI 管理器 ID。
- `vsi-mgrid-enc-effective` 指定用于虚拟链路或 VNIC 的 VSI 管理器 ID 编码，是 VSI 管理器 ID 的基础。
- `vsi-typeid-effective` 指定虚拟链路或 VNIC 上的有效 VSI 类型 ID。
- `vsi-vers-effective` 指定链路上的有效 VSI 版本。

在站中使用 EVB

要在站中使用 EVB，必须安装 EVB 软件包。键入以下命令：

```
# pkg install evb
```

最好接受安装软件包后自动启用的缺省 EVB 配置。EVB 配置要使用 Oracle VSI 管理器才能启用 EVB。通过接受缺省的 EVB 配置，站可以立即与网桥交换有关站中已配置的任何 VNIC 的 VSI 信息。

以下示例显示物理链路上与 EVB 相关的属性：

```
# dladm show-linkprop -p vsi-mgrid,vsi-mgrid-enc
LINK     PROPERTY            PERM VALUE        DEFAULT      POSSIBLE
net4     vsi-mgrid           rw   --           ::           --
net4     vsi-mgrid-enc       rw   --           oracle_v1    none,oracle_v1
```

该输出显示 Oracle Solaris 11 中的缺省 EVB 配置。通过使用 oracle_v1 编码，Oracle VSI 管理器可管理它所识别并支持的 VSI 及其数据链路属性。

如果不想使用缺省配置，请将编码更改为 none。

```
# dladm set-linkprop -p vsi-mgrid-enc=none net4
```

此后，必须手动提供要用作 VSI 管理器 ID 的 IPv6 地址、定义 VSI 类型 ID 以及其他所有与 EVB 相关的组件及其属性。

以下示例显示 VSI 或 VNIC 上与 EVB 相关的属性：

```
# dladm show-linkprop vnic0
LINK     PROPERTY                PERM VALUE      DEFAULT  POSSIBLE
...
vnic0    vsi-typeid              rw   --         --       --
vnic0    vsi-typeid-effective    r-   65684      --       --
vnic0    vsi-vers                rw   --         --       --
vnic0    vsi-vers-effective      r-   0          --       --
vnic0    vsi-mgrid               rw   --         --       --
vnic0    vsi-mgrid-effective     r-   ::         --       --
vnic0    vsi-mgrid-enc-effective r-   oracle_v1  --       --
...
```

该输出显示的值基于 Oracle VSI 管理器。VSI 的有效 VSI 管理器 ID 编码是 oracle_v1。因此，将自动生成类型 ID 65684，且该 ID 对 vnic0 有效。

以下示例显示有关站中已启用 EVB 时物理以太网链路的 VDP 状态的信息。要仅显示单个链路的信息，请在命令中指定该链路。否则，将显示有关所有以太网链路的 VDP 信息。

```
# dladm show-ether -P vdb
VSI     LINK   VSIID             VSI-TYPEID VSI-STATE   CMD-PENDING
vnic0   net4   2:8:20:2c:ed:f3   65684/0    TIMEDOUT    NONE
vnic1   net4   2:8:20:df:73:77   65684/0    TIMEDOUT    NONE
```

该输出显示已通过链路 net4 配置了两个 VSI。其特定的 VSI ID 指向各自的 MAC 地址。基于缺省的 vsi-mgrid 值，两个 VSI 具有相同的 VSI 类型 ID (65684)。

要获得有关传出或传入的 VDP 包的统计信息，请使用以下命令：

```
# dlstat show-ether -P vdb
```

◆ ◆ ◆ 第 10 章

10 集成负载平衡器（概述）

本章介绍 Oracle Solaris 的一种功能：集成负载平衡器 (Integrated Load Balancer, ILB)。ILB 可为基于 SPARC 和基于 x86 的系统上安装的 Oracle Solaris 提供第 3 层和第 4 层负载平衡能力。ILB 会拦截客户机的传入请求，根据负载平衡规则确定哪个后端服务器应处理该请求，然后将该请求转发到选定的服务器。ILB 将执行可选的运行状况检查，并为负载平衡算法提供数据，以验证选定的服务器能否处理传入请求。通过执行以上功能，ILB 将定向到服务器的工作负荷分配到多个服务器。这样做可以提高可靠性、最大限度地缩短响应时间，通常还可提高服务器性能。

本章包含以下主题：

ILB 功能

ILB 的主要功能包括：

- 支持 IPv4 和 IPv6 的无状态服务器直接返回 (Direct Server Return, DSR) 和网络地址转换 (Network Address Translation, NAT) 操作模式
- 允许通过命令行界面 (command-line interface, CLI) 进行 ILB 管理
- 通过运行状况检查提供服务器监视功能

以下列表介绍了 ILB 的其他功能：

- **允许客户机对虚拟 IP (virtual IP, VIP) 地址执行 ping 操作**－ILB 可响应客户机对虚拟服务 IP 地址的 Internet 控制消息协议 (Internet Control Message Protocol, ICMP) 回显请求。ILB 为 DSR 和 NAT 操作模式提供此功能。
- **允许您在不中断服务的情况下在服务器组中添加和删除服务器**－您可以在服务器组中动态添加和删除服务器，而无需中断与后端服务器建立的现有连接。ILB 为 NAT 操作模式提供此功能。
- **允许您配置会话持久性（粘性）**－对于许多应用程序，务必将来自同一客户机的一系列连接和/或包发送到同一后端服务器。您可以通过使用 `-p` 选项并在子命令 `ilbadm create-rule` 中指定 `pmask`，为虚拟服务配置会话持久性（即源地址持久性）。有关更多信息，请参见第 159 页中的 “如何创建 ILB 规则”。在创建持久性映射后，对具有匹配的客户机源 IP 地址的虚拟服务的连接、包或这两者的后续请求将转发到同一后端服务器。对于 IPv4，无类域间路由 (Classless Inter-Domain Routing, CIDR) 表示法中的前缀长度是一个介于 `0-32` 之间的值；对于 IPv6，该前缀长度是一个介于 `0-128` 之间的值。DSR 和 NAT 操作模式均支持会话持久性。
- **允许您执行连接排空**－ILB 仅为基于 NAT 的虚拟服务的服务器提供对此功能的支持。此功能可阻止将新连接发送到已禁用的服务器。此功能可用于在不中断活动连接或会话的情况下关闭服务器。到服务器的现有连接将继续正常运行。终止该服务器的所有连接后，便可以将其关闭以进行维护。当服务器准备好处理请求后，服务器将启用以便负载平衡器能够向其转发新连接。通过此功能，可在不中断活动连接或会话的情况下将服务器关闭以进行维护。
- **支持对 TCP 和 UDP 端口进行负载平衡**－ILB 可在不同服务器集中对给定 IP 地址的所有端口进行负载平衡，而不会要求您为每个端口设置显式规则。ILB 为 DSR 和 NAT 操作模式提供此功能。
- **允许您为同一服务器组中的虚拟服务指定独立端口**－通过此功能，您可以针对 NAT 操作模式为同一服务器组中的各个服务器指定不同的目标端口。
- **允许您对简单端口范围进行负载平衡**－ILB 可以将 VIP 的端口范围负载平衡到给定的服务器组。为方便起见，您可以通过将同一 VIP 的各个端口范围负载平衡到不同的后端服务器集来节省 IP 地址。此外，在为 NAT 模式启用会话持久性后，ILB 还会将同一客户机 IP 地址对范围中的各个端口的请求发送到同一后端服务器。
- **支持端口范围移位和折叠**－端口范围移位和折叠取决于负载平衡规则中服务器的端口范围。因此，如果服务器端口范围与 VIP 端口范围不同，将自动实现端口移位。如果服务器端口范围为单个端口，则实现端口折叠。这些功能是为 NAT 操作模式提供的。

ILB 组件

ILB 具有以下三个主要组件：

- `ilbadm` CLI－可以使用此命令行界面配置负载平衡规则、执行可选的运行状况检查以及查看统计信息。
- `libilb` 配置库－`ilbadm` 和第三方应用程序可以使用 `libilb` 中实现的功能进行 ILB 管理。

- `ilbd` 守护进程－此守护进程执行以下任务：
  - 管理重新引导和软件包更新过程中的持久性配置
  - 处理配置信息并将其发送到 ILB 内核模块进行执行，从而提供对 ILB 内核模块的串行访问
  - 执行运行状况检查并向 ILB 内核模块发送结果，以便正确调整负载分布

ILB 操作模式

ILB 以单路拓扑和双路拓扑形式支持 IPv4 和 IPv6 的无状态服务器直接返回 (Direct Server Return, DSR) 和网络地址转换 (Network Address Translation, NAT) 操作模式。

- 无状态 DSR 拓扑
- NAT 模式（全 NAT 和半 NAT）拓扑

服务器直接返回拓扑

在 DSR 模式下，ILB 会平衡后端服务器的传入请求，而使从服务器返回到客户机的通信流量绕过它。但是，您也可以将 LIB 设置为后端服务器的路由器。在这种情况下，后端服务器对客户机的响应将通过正在运行 ILB 的系统进行路由。ILB 的当前 DSR 实现未提供 TCP 连接跟踪（意味着它是无状态的）。使用无状态 DSR，ILB 不会保存已处理包的任何状态信息，但基本统计信息除外。由于 ILB 在此模式下不保存任何状态，因此性能与正常的 IP 转发性能相当。此模式最适合无连接协议。

优点：

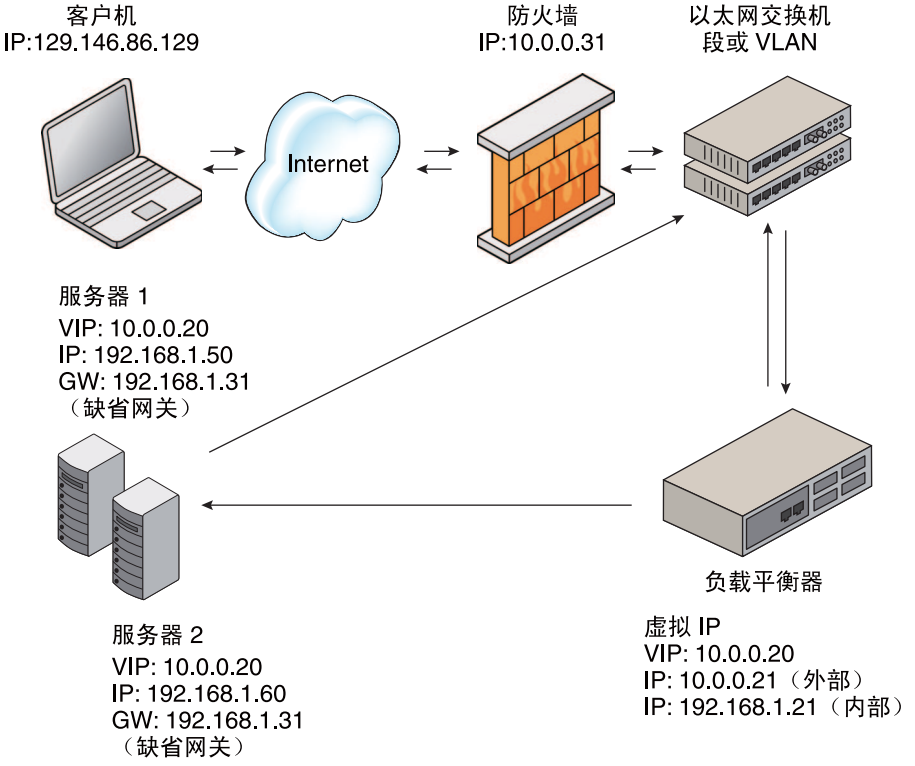
- 性能优于 NAT，因为只更改包的目标 MAC 地址，服务器将直接响应客户机。
- 服务器和客户机之间完全透明。服务器可直接看到来自客户机 IP 地址的连接，并通过缺省网关回复客户机。

缺点：

- 后端服务器必须响应其自身的 IP 地址（针对运行状况检查）以及虚拟 IP 地址（针对负载平衡通信）。
- 由于负载平衡器不维护连接状态（意味着它是无状态的），因此添加或删除服务器将导致连接中断。

下图显示了如何使用 DSR 拓扑实现 ILB。

图 10–1 服务器直接返回拓扑



在此图中，两个后端服务器与 ILB 机箱处于同一子网 (192.168.1.0/24) 中。服务器还连接到路由器，因此在收到 ILB 机箱转发的请求后，它们可以直接回复客户机。

网络地址转换拓扑

ILB 完全是为了实现负载平衡功能才使用独立模式的 NAT。在此模式下，ILB 会重写头信息并处理传入与传出通信流量。ILB 可在半 NAT 模式和全 NAT 模式下运行。但是，全 NAT 还会重写源 IP 地址，使服务器认为所有连接均源自负载平衡器。NAT 确实提供了 TCP 连接跟踪（意味着它是有状态的）。NAT 模式可提供附加安全性，最适合超文本传输协议 (Hypertext Transfer Protocol, HTTP)（或安全套接字层 (Secure Sockets Layer, SSL)）通信。

优点：

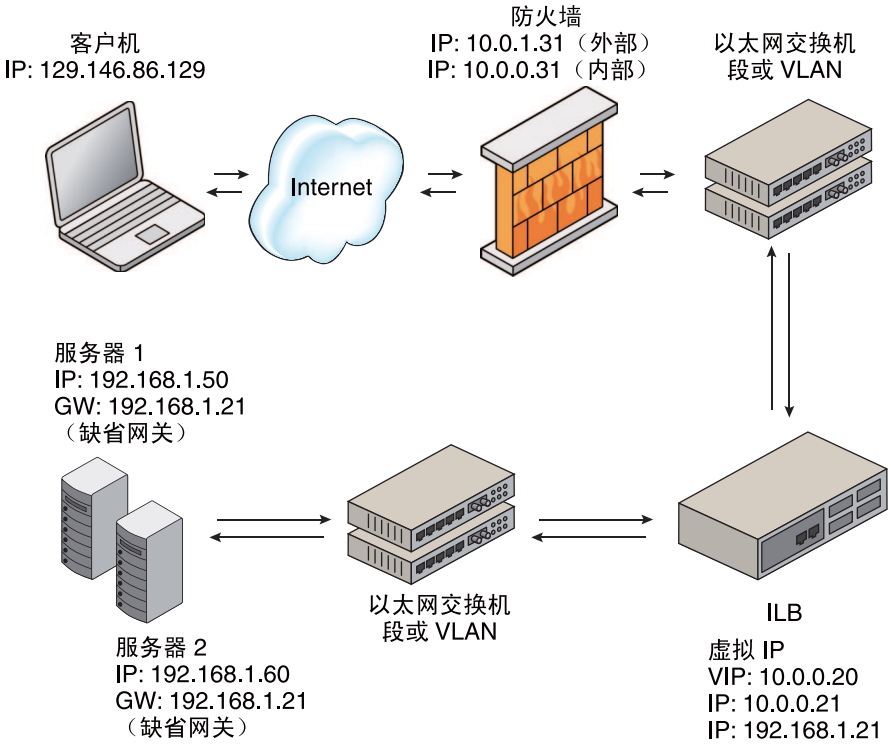
- 通过将缺省网关更改为指向负载平衡器，可使用所有后端服务器。
- 由于负载平衡器保持连接状态，因此可在不中断连接的情况下添加或删除服务器。

缺点：

- 性能低于 DSR，因为处理涉及到操作 IP 头，并且服务器向负载平衡器发送响应。
- 所有后端服务器都必须使用负载平衡器作为缺省网关。

NAT 拓扑的一般实现方式如下图所示。

图 10–2 网络地址转换拓扑



这种情况下，对 VIP 的所有请求均通过 ILB 机箱并转发到后端服务器。后端服务器的所有回复均通过 ILB 机箱以进行网络地址转换。

注意 – ILB 中实现的 NAT 代码路径不同于 Oracle Solaris 的 IP 过滤器功能中实现的代码路径。**请勿**同时使用这两种代码路径。

半 NAT 负载平衡拓扑

在 ILB 的半 NAT 操作模式下，ILB 仅重写包头中的目标 IP 地址。如果您使用的是半 NAT 实现方式，则无法从服务器所在的子网连接到该服务的虚拟 IP (virtual IP, VIP) 地址。下表显示了流经客户机和 ILB 之间以及 ILB 和后端服务器之间的包的 IP 地址。

表 10–1 服务器和客户机位于不同的网络中时，半 NAT 实现的请求流和响应流

	请求流	源 IP 地址	目标 IP 地址
1.	客户机 –> ILB	客户机	ILB 的 VIP
2.	ILB –> 服务器	客户机	服务器
	响应流		
3.	服务器 –> ILB	服务器	客户机
4.	ILB –> 客户机	ILB 的 VIP	客户机

如果您将客户机系统连接到这些服务器所在的网络，预定的服务器将直接响应该客户机。不会执行第四步。因此，服务器对客户机响应的源 IP 地址无效。当客户机向负载平衡器发送连接请求时，预定的服务器会进行响应。自此以后，客户机的 IP 栈将正确丢弃所有响应。

在这种情况下，请求流和响应流会按下表所示进行。

表 10–2 服务器和客户机位于同一网络中时，半 NAT 实现的请求流和响应流

	请求流	源 IP 地址	目标 IP 地址
1.	客户机 –> ILB	客户机	ILB 的 VIP
2.	ILB –> 服务器	客户机	服务器
	响应流		
3.	服务器 –> 客户机	服务器	客户机

全 NAT 负载平衡拓扑

在全 NAT 实现中，将会重写源 IP 地址和目标 IP 地址，以确保通信在两个方向通过负载平衡器。通过全 NAT 拓扑，可以从服务器所在的子网中连接到 VIP。

下表描述了使用全 NAT 拓扑时流经客户机和 ILB 之间以及 ILB 和后端服务器之间的包的 IP 地址。服务器中不需要使用 ILB 框的特殊缺省路由。但是请注意，全 NAT 拓扑需要管理员留出一个或一系列 IP 地址供 ILB 用作源地址，以便与后端服务器进行通信。假定使用的地址属于子网 C。在此方案中，ILB 充当代理。

表 10–3 全 NAT 实现的请求流和响应流

	请求流	源 IP 地址	目标 IP 地址
1.	客户机 –> ILB	客户机	ILB 的 VIP
2.	ILB –> 服务器	负载平衡器的接口地址（子网 C）	服务器

表 10–3 全 NAT 实现的请求流和响应流 （续）

	响应流		
3.	服务器 –> ILB	服务器	ILB 的接口地址（子网 C）
4.	ILB –> 客户机	ILB 的 VIP	客户机

ILB 的工作原理

本节介绍了 ILB 的工作原理：如何处理从客户机到 VIP 的请求，如何将请求转发到后端服务器以及如何处理响应。

客户机至服务器包处理：

1. ILB 收到客户机发送到 VIP 地址的传入请求，并将该请求与负载平衡规则匹配。
2. 如果 ILB 发现匹配的负载平衡规则，它将使用负载平衡算法将请求转发到后端服务器，具体取决于操作模式。
   - 在 DSR 模式下，ILB 会将传入请求的 MAC 头替换为选定的后端服务器的 MAC 头。
   - 在半 NAT 模式下，ILB 会将传入请求的目标 IP 地址和传输协议端口号替换为选定的后端服务器的对应项。
   - 在全 NAT 模式下，ILB 会将传入请求的源 IP 地址和传输协议端口号替换为负载平衡规则的 NAT 源地址。ILB 还会将传入请求的目标 IP 地址和传输协议端口号替换为选定的后端服务器的对应项。
3. ILB 将修改后的传入请求转发到选定的后端服务器。

服务器至客户机包处理：

1. 后端服务器向 ILB 发送回复，以响应客户机的传入请求。
2. ILB 在收到后端服务器的响应后采取的措施基于操作模式。
   - 在 DSR 模式下，后端服务器的响应会绕过 ILB 并直接转至客户机。但是，如果 ILB 还用作后端服务器的路由器，则后端服务器对客户机的响应将通过运行 ILB 的系统进行路由。
   - 在半 NAT 模式和全 NAT 模式下，ILB 会将后端服务器的响应与传入请求进行匹配，并将更改后的 IP 地址和传输协议端口号替换为原始传入请求的对应项。然后，ILB 将响应转发到客户机。

ILB 算法

ILB 算法用于控制通信流量分布，并为负载分布和服务器选择提供各种特征。ILB 为两种操作模式提供了以下算法：

- 循环－在循环算法中，负载平衡器将请求轮流指定给服务器组。在为某服务器指定请求后，该服务器将会移至列表末尾。
- *src IP* 散列－在源 IP 散列方法中，负载平衡器根据传入请求的源 IP 地址的散列值选择服务器。
- *src-IP, port* 散列－在源 IP、端口散列方法中，负载平衡器根据传入请求的源 IP 地址和源端口的散列值选择服务器。
- *src-IP, VIP* 散列－在源 IP、VIP 散列方法中，负载平衡器根据传入请求的源 IP 地址和目标 IP 地址的散列值选择服务器。

服务管理工具

ILB 由服务管理工具 (Service Management Facility, SMF) 服务 `svc:/network/loadbalancer/ilb:default` 进行管理。有关 SMF 的概述，请参见《在 Oracle Solaris 11.1 中管理服务和故障》中的第 1 章 “管理服务（概述）”。有关与 SMF 关联的逐步过程，请参见《在 Oracle Solaris 11.1 中管理服务和故障》中的第 2 章 “管理服务（任务）”。

ILB 命令行界面

ILB 命令行界面位于 `/usr/sbin/ilbadm` 目录中。CLI 包含用于配置负载平衡规则、服务器组和运行状况检查的子命令。此外，还包含用于显示统计信息以及查看配置详细信息的子命令。子命令可以分为两类：

- **配置子命令**－这些子命令用于执行以下任务：
  - 创建和删除负载平衡规则
  - 启用和禁用负载平衡规则
  - 创建和删除服务器组
  - 在服务器组中添加和删除服务器
  - 启用和禁用后端服务器
  - 为负载平衡规则中的服务器组创建和删除服务器运行状况检查

注 – 要管理配置子命令，您需要拥有特权。这些特权是通过 Oracle Solaris 基于角色的访问控制 (role-based access control, RBAC) 获取的。要创建相应的角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

- **查看子命令** – 这些子命令用于执行以下任务：
  - 查看已配置的负载平衡规则、服务器组和运行状况检查
  - 查看包转发统计信息
  - 查看 NAT 连接表
  - 查看运行状况检查结果
  - 查看会话持久性映射表

注 – 管理查看子命令不需要拥有特权。

有关 `ilbadm` 子命令列表，请参见第 139 页中的 “ILB 命令和子命令”。有关 `ilbadm` 子命令的更多详细信息，请参阅 `ilbadm(1M)` 手册页。

ILB 命令和子命令

您可以使用 `ilbadm` 及其子命令处理负载平衡规则。有关 `ilbadm` 子命令的更多详细信息，请参阅 `ilbadm(1M)` 手册页。

表 10–4 用于处理负载平衡规则的 ILB 子命令

ILB 子命令	说明
`ilbadm create-rule`	创建具有给定特征的 `rule name`。
`ilbadm show-rule`	显示指定规则的特征，或显示所有规则（如果未指定任何规则）。
`ilbadm delete-rule`	删除与 `rule name` 有关的所有信息。如果 `rule name` 不存在，该子命令将会失败。
`ilbadm enable-rule`	启用指定规则，或启用所有规则（如果未指定任何名称）。
`ilbadm disable-rule`	禁用指定规则，或禁用所有规则（如果未指定任何名称）。
`ilbadm show-statistics`	显示 ILB 统计信息。例如，将 `-t` 与该子命令结合使用会在每个头中包含时间戳。
`ilbadm show-hc-result`	显示与规则 `rule-name` 的指定名称关联的服务器的运行状况检查结果。如果未指定 `rule-name`，则显示所有规则的服务器运行状况检查结果。

表 10–4 用于处理负载平衡规则的 ILB 子命令 （续）

ILB 子命令	说明
`ilbadm show-nat`	显示 NAT 表信息。
`ilbadm create-servergroup`	创建包含一个或多个服务器的服务器组。可以使用 `ilbadm add-server` 添加其他服务器。
`ilbadm delete-servergroup`	删除服务器组。
`ilbadm show-servergroup`	列出某个服务器组，或列出所有服务器组（如果未指定任何服务器组）。
`ilbadm enable-server`	启用已禁用的服务器。
`ilbadm disable-server`	禁用指定的服务器。
`ilbadm add-server`	将指定的服务器添加到服务器组。
`ilbadm show-server`	显示与指定规则关联的服务器，或显示所有服务器（如果未指定规则名称）。
`ilbadm remove-server`	从服务器组中删除一个或多个服务器。
`ilbadm create-healthcheck`	设置可用于建立规则的运行状况检查信息。
`ilbadm show-healthcheck`	显示有关所配置的运行状况检查的详细信息。
`ilbadm delete-healthcheck`	删除运行状况检查信息。
`ilbadm show-persist`	显示会话持久性映射表。
`ilbadm export-config` *filename*	以适合使用 `ilbadm import` 进行导入的格式导出现有的 ILB 配置文件。如果未指定 *filename*，则 `ilbadm export` 会写入 `stdout`。
`ilbadm import-config -p` *filename*	导入某个文件并将现有 ILB 配置替换为该导入文件的内容。如果未指定 *filename*，则 `ilbadm import` 会从 `stdin` 中读取。

◆ ◆ ◆ 第 11 章

11 配置集成负载平衡器

本章介绍集成负载平衡器 (Integrated Load Balancer, ILB) 的安装，并提供了有关设置简单 ILB 配置的示例。本章包含以下主题：

安装 ILB

ILB 包含两部分，即内核部分和用户级部分。在安装 Oracle Solaris 11 的过程中，将会自动安装内核部分。要获取 ILB 的用户级部分，必须使用 `pkg install ilb` 命令手动安装 `ilb` 软件包。

启用 ILB

本节介绍用于启用 ILB 的过程。

▼ 如何启用 ILB

开始之前 确保系统中的基于角色的访问控制 (role-based access control, RBAC) 属性文件具有以下条目。如果不存在这些条目，请手动进行添加。

- 文件名：`/etc/security/auth_attr`
  - `solaris.network.ilb.config:::Network ILB Configuration::help=NetworkILBconf.html`

    - `solaris.network.ilb.enable:::Network ILB Enable Configuration::help=NetworkILBenable.html`
    - `solaris.smf.manage.ilb:::Manage Integrated Load Balancer Service States::help=SmfILBStates.html`
  - 文件名：`/etc/security/prof_attr`
    - `Network ILB:::Manage ILB configuration via ilbadm:auths=solaris.network.ilb.config,solaris.network.ilb.enable;help=RtNetILB.htm`
    - 该文件中的网络管理条目必须包含 `solaris.smf.manage.ilb`。
  - 文件名：`/etc/user_attr`
    - `daemon::::auths=solaris.smf.manage.ilb,solaris.smf.modify.application`

必须为 ILB 配置子命令设置用户授权。您必须拥有 `solaris.network.ilb.config` RBAC 授权，才能执行第 139 页中的 “ILB 命令和子命令”中列出的 ILB 配置子命令。

- 要为现有用户指定授权，请参见《Oracle Solaris 11.1 管理：安全服务》中的第 9 章 “使用基于角色的访问控制（任务）”。
- 您也可以在系统上创建新的用户帐户时提供授权。

  以下示例将创建一个组 ID 为 `10`、用户 ID 为 `1210` 并且具有管理系统中 ILB 的授权的用户 `ilbadm`。

  ```
  # useradd -g 10 -u 1210 -A solaris.network.ilb.config ilbadmin
  ```

  `useradd` 命令将向 `/etc/passwd`、`/etc/shadow` 和 `/etc/user_attr` 文件中添加一个新用户。`-A` 选项为该用户指定授权。

1. **承担拥有 ILB Management（ILB 管理）权限配置文件的角色，或者成为超级用户。**

   可以将 ILB Management（ILB 管理）权限配置文件指定给您创建的角色。要创建角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

2. **启用相应的转发服务 IPv4 或 IPv6，或同时启用这两种服务。**

   此命令在成功时不会生成输出。

   ```
   # ipadm set-prop -p forwarding=on ipv4
   # ipadm set-prop -p forwarding=on ipv6
   ```

3. **启用 ILB 服务。**

   ```
   # svcadm enable ilb
   ```

4. **验证是否已启用 ILB 服务。**

   ```
   # svcs ilb
   ```

配置 ILB

本节介绍用于设置 ILB 以使用半 NAT 拓扑对两台服务器之间的通信进行负载平衡的步骤。请参见第 133 页中的 “ILB 操作模式”中的 NAT 拓扑实现。

▼ 如何配置 ILB

1 **承担拥有 ILB Management（ILB 管理）权限配置文件的角色，或者成为超级用户。**

可以将 ILB Management（ILB 管理）权限配置文件指定给您创建的角色。要创建角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

2 **设置后端服务器。**

在此方案中，后端服务器设置为将 ILB 用作缺省路由器。可以通过在两台服务器上运行以下命令来实现此目的。

```
# route add -p default 192.168.1.21
```

执行此命令后，在两台服务器上启动服务器应用程序。假定它是侦听端口 5000 的 TCP 应用程序。

3 **在 ILB 中设置服务器组。**

有 2 台服务器，`192.168.1.50` 和 `192.169.1.60`。可以通过键入以下命令来创建包含这两台服务器的服务器组 `srvgrp1`。

```
# ilbadm create-sg -s servers=192.168.1.50,192.168.1.60 srvgrp1
```

4 **设置名为 `hc-srvgrp1` 的简单运行状况检查（可以通过键入以下命令来创建该运行状况检查）。**

简单 TCP 级别的运行状况检查用于检测服务器应用程序是否可访问。每隔 60 秒执行一次该检查。它将最多尝试 3 次，并且在两次尝试之间最多等待 3 秒，以查看服务器是否正常运行。如果所有 3 次尝试均失败，它会将服务器标记为 `dead`。

```
# ilbadm create-hc -h hc-test=tcp,hc-timeout=3, \
hc-count=3,hc-inerval=60 hc-srvgrp1
```

5 **通过键入以下命令来设置 ILB 规则。**

此规则中使用持久性（具有 32 位掩码），且负载平衡算法为 `round robin`。使用服务器组 `srvgrp1` 和运行状况检查机制 `hc-srvgrp1`。可以通过键入以下命令来创建该规则。

```
# ilbadm create-rule -e -p -i vip=10.0.2.20,port=5000 -m \
 lbalg=rr,type=half-nat,pmask=32 \
-h hc-name=hc-srvgrp1 -o servergroup=srvgrp1 rule1_rr
```

禁用 ILB

下节介绍禁用 ILB 的过程。

▼ 如何禁用 ILB

1 **承担拥有 ILB Management（ILB 管理）权限配置文件的角色，或者成为超级用户。**

可以将 ILB Management（ILB 管理）权限配置文件指定给您创建的角色。要创建角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

2 **禁用 ILB 服务。**

```
# svcadm disable ilb
```

3 **验证是否已禁用 ILB 服务。**

```
# svcs ilb
```

导入和导出配置

`ilbadm export` 子命令用于将当前 ILB 配置导出到用户指定的文件中。此信息随后可用作 `ilbadm import` 子命令的输入。

`ilbadm import` 子命令会在导入之前删除现有配置，除非明确指示保留该配置。如果省略文件名，则指示命令从 `stdin` 读取或写入 `stdout`。

要导出 ILB 配置，请使用 `export-config` 命令。以下示例以适合使用 `import` 子命令进行导入的格式将当前配置导出到文件 `/var/tmp/ilb_config` 中：

```
# ilbadm export-config /var/tmp/ilb_config
```

要导入 ILB 配置，请使用 `import-config` 命令。以下示例读取 `/var/tmp/ilb_config` 文件的内容并覆盖现有配置：

```
# ilbadm import-config /var/tmp/ilb_config
```

为 ILB 配置高可用性（仅限主动-被动模式）

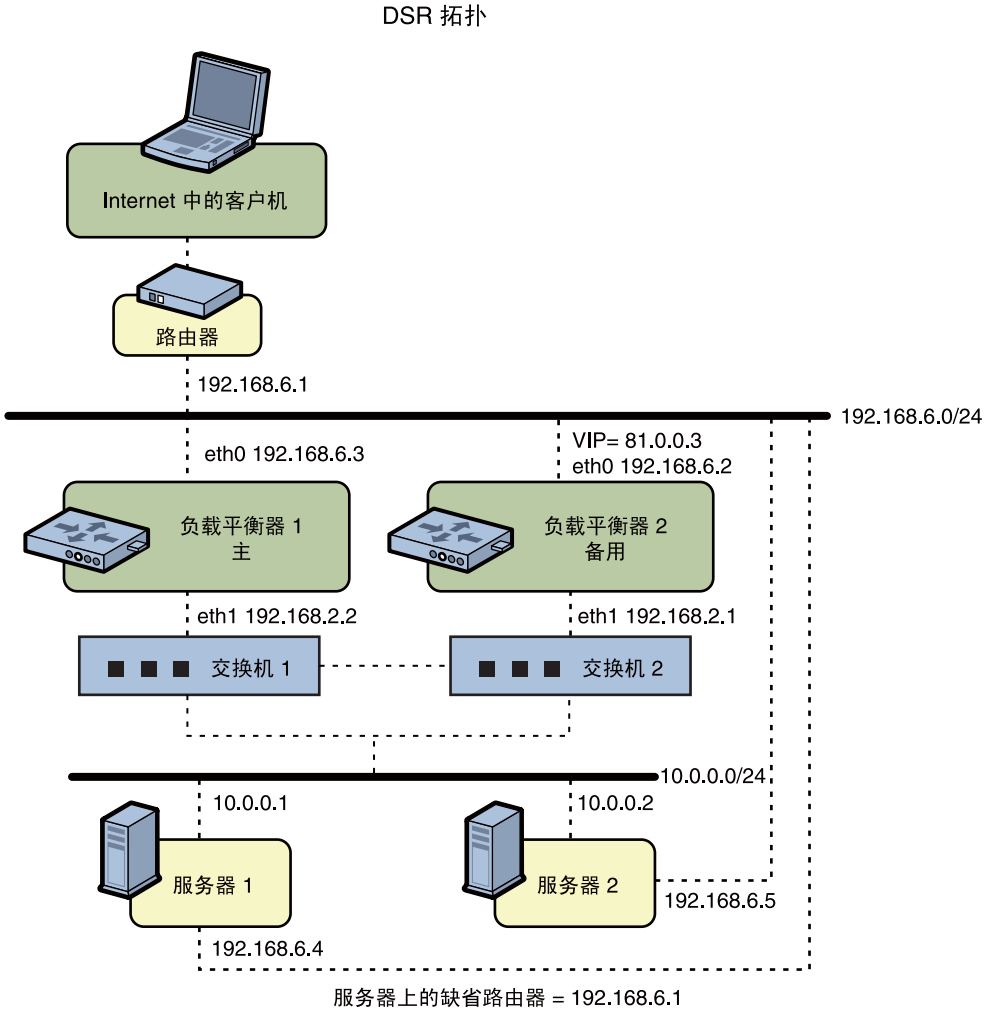
本节介绍了如何使用 DSR、半 NAT 拓扑对 ILB 进行高可用性 (high-availability, HA) 配置。

使用 DSR 拓扑为 ILB 配置高可用性

本节介绍了如何使用 DSR 拓扑设置 ILB 连接以实现高可用性 (high availability, HA)。您需要设置两个负载平衡器，一个作为主负载平衡器，另一个作为备用负载平衡器。如果主负载平衡器出现故障，备用负载平衡器将承担主负载平衡器的角色。

下图显示了用于配置 ILB 连接以实现 HA 的 DSR 拓扑。

图 11–1 使用 DSR 拓扑进行 ILB 高可用性配置



负载平衡器上的所有 VIP 都是在面向子网 192.168.6.0/24 的接口上进行配置的。

▼ 如何使用 DSR 拓扑配置 ILB 以实现高可用性

1 **承担拥有 ILB Management（ILB 管理）权限配置文件的角色，或者成为超级用户。**

可以将 ILB Management（ILB 管理）权限配置文件指定给您创建的角色。要创建角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

2 配置主负载平衡器和备用负载平衡器。

```
# ilbadm create-servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -i vip=81.0.0.3,port=9001 \
-m lbalg=hash-ip-port,type=DSR -o servergroup=sg1 rule1
```

3 确保在所有服务器的 lo0 接口上都配置了 VIP。

```
Server1# ipadm create-addr -d -a 81.0.0.3/24 lo0
Server2# ipadm create-addr -d -a 81.0.0.3/24 lo0
```

4 配置用于充当主负载平衡器的负载平衡器 1。

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# ipadm create-addr -d -a 81.0.0.3/24 vnic1
```

5 配置用于充当备用负载平衡器的负载平衡器 2。

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# ipadm create-addr -d -a 81.0.0.3/24 vincl
```

上述配置可防范以下故障情况：

- 如果负载平衡器 1 出现故障，负载平衡器 2 将成为主负载平衡器。然后负载平衡器 2 将负责 VIP 81.0.0.3 的地址解析，并处理由客户机发往目标 IP 地址 81.0.0.3 的所有包。

  当负载平衡器 1 恢复时，负载平衡器 2 会恢复为备用模式。
- 如果负载平衡器 1 的一个或两个接口出现故障，负载平衡器 2 会接替，充当主负载平衡器。然后负载平衡器 2 将负责 VIP 81.0.0.3 的地址解析，并处理由客户机发往目标 IP 地址 81.0.0.3 的所有包。

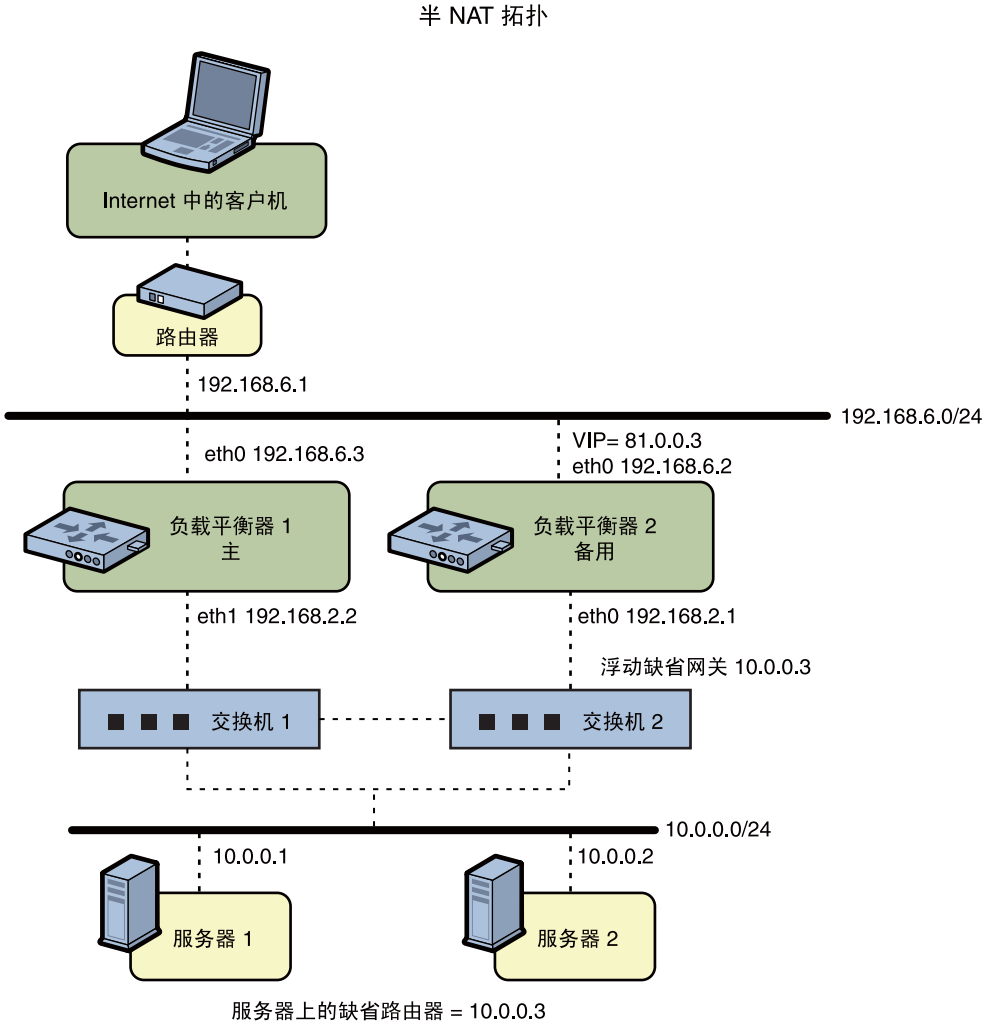
  当负载平衡器 1 的两个接口运行正常时，负载平衡器 2 会恢复为备用模式。

使用半 NAT 拓扑为 ILB 配置高可用性

本节介绍了如何使用半 NAT 拓扑设置 ILB 连接以实现高可用性 (high availability, HA)。您需要设置两个负载平衡器，一个作为主负载平衡器，另一个作为备用负载平衡器。如果主负载平衡器出现故障，备用负载平衡器将承担主负载平衡器的角色。

下图显示了用于配置 ILB 连接以实现 HA 的半 NAT 拓扑。

图 11–2 使用半 NAT 拓扑进行 ILB 高可用性配置



负载平衡器上的所有 VIP 都是在面向子网 192.168.6.0/24 的接口上进行配置的。

▼ 如何使用半 NAT 拓扑配置 ILB 以实现高可用性

1 **承担拥有 ILB Management（ILB 管理）权限配置文件的角色，或者成为超级用户。**

可以将 ILB Management（ILB 管理）权限配置文件指定给您创建的角色。要创建角色并将其指定给用户，请参见《Oracle Solaris 11.1 管理：安全服务》中的“初次配置 RBAC（任务列表）”。

2 配置主负载平衡器和备用负载平衡器。

```
# ilbadm create servergroup -s server=10.0.0.1,10.0.0.2 sg1
# ilbadm create-rule -ep -i vip=81.0.0.3,port=9001-9006,protocol=udp \
-m lbalg=roundrobin,type=HALF-NAT,pmask=24 \
-h hc-name=hc1,hc-port=9006 \
-t conn-drain=70,nat-timeout=70,persist-timeout=70 -o servergroup=sg1 rule1
```

3 配置用于充当主负载平衡器的负载平衡器 1。

```
LB1# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB1# ipadm create-addr -d -a 81.0.0.3/24 vnic1
LB1# vrrpadm create-router -V 1 -A inet -l eth0 -p 255 vrrp1
LB1# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB1# ipadm create-addr -d -a 10.0.0.3/24 vnic2
LB1# vrrpadm create-router -V 2 -A inet -l eth1 -p 255 vrrp2
```

4 配置用于充当备用负载平衡器的负载平衡器 2。

```
LB2# dladm create-vnic -m vrrp -V 1 -A inet -l eth0 vnic1
LB2# ipadm create-addr -d -a 81.0.0.3/24 vnic1
LB2# vrrpadm create-router -V 1 -A inet -l eth0 -p 100 vrrp1
LB2# dladm create-vnic -m vrrp -V 2 -A inet -l eth1 vnic2
LB2# ipadm create-addr -d -a 10.0.0.3/24 vnic2
LB2# vrrpadm create-router -V 2 -A inet -l eth1 -p 100 vrrp2
```

5 为两台服务器添加浮动缺省网关的 IP 地址。

```
# route add net 192.168.6.0/24 10.0.0.3
```

上述配置可防范以下故障情况：

- 如果负载平衡器 1 出现故障，负载平衡器 2 将成为主负载平衡器。然后负载平衡器 2 将负责 VIP 81.0.0.3 的地址解析，并处理由客户机发往目标 IP 地址 81.0.0.3 的所有包。此外，负载平衡器 2 还处理发送到浮动网关地址 10.0.0.3 的所有包。

  当负载平衡器 1 恢复时，负载平衡器 2 会恢复为备用模式。

- 如果负载平衡器 1 的一个或两个接口出现故障，负载平衡器 2 会接替，充当主负载平衡器。然后负载平衡器 2 将负责 VIP 81.0.0.3 的地址解析，并处理由客户机发往目标 IP 地址 81.0.0.3 的所有包。此外，负载平衡器 2 还处理发送到浮动网关地址 10.0.0.3 的所有包。

  当负载平衡器 1 的两个接口运行正常时，负载平衡器 2 会恢复为备用模式。

注 – 当前的 ILB 实现不会同步主负载平衡器和备用负载平衡器。当主负载平衡器出现故障而由备用负载平衡器接管时，现有连接将会失败。但是，在主负载平衡器出现故障的情况下，未同步的 HA 仍然有价值。

◆ ◆ ◆ 第 12 章

12 管理集成负载平衡器

本章介绍管理 ILB 服务器组的过程，如创建或删除服务器组、管理后端服务器（如在服务器组中添加、删除、重新启用或禁用服务器）、创建和删除规则以及显示统计信息。

本章包含以下主题：

管理 ILB 服务器组

本节介绍如何使用 `ilbadm` 命令创建、删除和列出 ILB 服务器组。

▼ 如何创建 ILB 服务器组

1. **为要创建的服务器组选择一个名称。**

2. **选择要包含在该服务器组中的服务器。**
   可以通过服务器的主机名或 IP 地址及可选端口来指定服务器。

3. **创建服务器组。**

```
# ilbadm create-servergroup -s servers= \
server1,server2,server3  servergroup
```

示例 12–1 创建 ILB 服务器组

以下示例创建了一个名为 webgroup 并包含三个服务器的服务器组：

```
# ilbadm create-servergroup -s servers=webserv1,webserv2,webserv3 webgroup
```

▼ 如何删除 ILB 服务器组

1 **在终端窗口中，键入 show-servergroup 子命令以获取有关特定服务器组或所有服务器组的信息。**

```
# ilbadm show-servergroup -o all
```

以下命令样例列出了有关所有服务器组的详细信息：

sgname	serverID	minport	maxport	IP_address
specgroup	_specgroup.0	7001	7001	199.199.68.18
specgroup	_specgroup.1	7001	7001	199.199.68.19
test123	_test123.0	7002	7002	199.199.67.18
test123	_test123.1	7002	7002	199.199.67.19

上表列出了两个服务器组，specgroup 和 test123。specgroup 包含两台服务器，199.199.68.18 和 199.199.68.19，这里的服务器使用端口 7001。类似地，test123 也包含两台服务器，199.199.67.18 和 199.199.67.19。这里的服务器使用端口 7002。

2 **选择要删除的服务器组。**

该服务器组不得正在由活动规则使用。否则，删除操作将会失败。

3 **在终端窗口中，使用以下命令删除该服务器组。**

```
# ilbadm delete-servergroup servergroup
```

示例 12–2 删除 ILB 服务器组

以下示例删除名为 webgroup 的服务器组：

```
# ilbadm delete-servergroup webgroup
```

在 ILB 中管理后端服务器

本节介绍如何使用 ilbadm 命令添加、删除、启用和禁用服务器组中的一个或多个后端服务器。

▼ 如何向 ILB 服务器组添加后端服务器

- **将后端服务器添加到服务器组。**

  指定的服务器必须包含主机名或 IP 地址，还可以包含可选的端口或端口范围。服务器组中不允许存在 IP 地址相同的服务器项。

  ```
  # ilbadm add-server -s server=192.168.89.1,192.168.89.2 ftpgroup
  # ilbadm add-server -s server=[2001:7::feed:6]:8080 sgrp
  ```

  `-e` 选项用于启用已添加到服务器组中的服务器。

  注 – IPv6 地址必须括在方括号中。

示例 12–3 向 ILB 服务器组添加后端服务器

以下示例将后端服务器添加到服务器组 `ftpgroup` 和 `sgrp` 中，并启用这些服务器。

```
# ilbadm add-server -e -s \
server=192.168.89.1,192.168.89.2 ftpgroup
# ilbadm add-server -e -s server=[2001:7::feed:6]:8080 sgrp
```

▼ 如何从 ILB 服务器组中删除后端服务器

1. **要将后端服务器从服务器组中删除，请执行以下步骤：**

   a. **确定要从服务器组中删除的服务器的服务器 ID。**

   服务器 ID 是将服务器添加到服务器组中时指定给系统的 IP 地址的唯一名称。可以从 `show-servergroup -o all` 子命令的输出中获取服务器 ID。

   b. **删除服务器。**

   ```
   # ilbadm remove-server -s server=serverID servergroup
   ```

2. **要将后端服务器从所有服务器组中删除，请执行以下步骤：**

   a. **确定要删除的服务器的 IP 地址和主机名。**

   b. **通过 `ilbadm show-servergroup-o all` 命令的输出确定包含该服务器的服务器组。**

   c. **对于每个服务器组，运行以上子命令将该服务器从服务器组中删除。**

示例 12–4 从 ILB 服务器组中删除后端服务器

以下示例将服务器 ID 为 _sg1.2 的服务器从服务器组 sg1 中删除：

```
# ilbadm remove-server -s server=_sg1.2 sg1
```

请注意以下事项：

- 如果 NAT 或半 NAT 规则正在使用该服务器，请先通过 disable-server 子命令禁用该服务器，然后再进行删除。有关更多信息，请参见第 154 页中的 “如何重新启用或禁用 ILB 服务器组中的后端服务器”。当服务器被禁用时，将会进入连接排空状态。排空所有连接后，可以使用 remove-server 子命令将服务器删除。发出 disable-server 命令后，请定期检查 NAT 表（使用 show-nat 命令），了解相关服务器是否仍处于连接状态。排空所有连接（服务器不再显示在 show-nat 命令输出中）后，即可使用 remove-server 命令删除服务器。
- 如果设置了 conn-drain 超时值，连接排空状态将在超时期限到期后结束。conn-drain 超时的缺省值为 0，这意味着它将持续等待，直到连接正常关闭为止。

▼ 如何重新启用或禁用 ILB 服务器组中的后端服务器

1. **确定要重新启用或禁用的后端服务器的 IP 地址、主机名或服务器 ID。**

   如果指定了 IP 地址或主机名，则会针对与该 IP 地址或主机名关联的所有规则重新启用或禁用服务器。如果指定了服务器 ID，则会针对与该服务器 ID 关联的特定规则重新启用或禁用服务器。

   注 – 如果服务器属于多个服务器组，则可以具有多个服务器 ID。

2. **重新启用或禁用后端服务器。**

   ```
   # ilbadm enable-server webservergroup.1
   # ilbadm disable-server webservergroup.1
   ```

示例 12–5 重新启用和禁用 ILB 服务器组中的后端服务器

在以下示例中，将会先启用服务器 ID 为 websg.1 的服务器，然后再将其禁用：

```
# ilbadm enable-server websg.1
# ilbadm disable-server websg.1
```

管理 ILB 的运行状况检查

ILB 提供了以下可选类型的服务器运行状况检查，供您进行选择：

- 内置的 ping 探测器
- 内置的 TCP 探测器
- 内置的 UDP 探测器
- 用户提供的可作为运行状况检查运行的定制测试

缺省情况下，ILB 不会执行任何运行状况检查。您可以在创建负载平衡规则时为每个服务器组指定运行状况检查。对于每个负载平衡规则，只能配置一个运行状况检查。只要启用了虚拟服务，与启用的虚拟服务关联的服务器组的运行状况检查就会自动启动并定期重复执行。禁用虚拟服务后，运行状况检查将立即停止。重新启用虚拟服务后，先前的运行状况检查状态将不会保留。

如果您指定了 TCP、UDP 或定制测试探测器来执行运行状况检查，则 ILB 缺省情况下会先发送 ping 探测器以确定服务器是否可以访问，然后再向该服务器发送指定的 TCP、UDP 或定制测试探测器。ping 探测器是一种监视服务器运行状况的方法。如果 ping 探测器失败，则会禁用对应的服务器，并且运行状况检查状态为 `unreachable`。如果 ping 探测器成功，但 TCP、UDP 或定制测试探测器失败，则会禁用服务器，并且运行状况检查状态为 `dead`。

注 –

- 可以禁用缺省的 ping 探测器。
- 不能禁用 UDP 探测器的缺省 ping 探测器。因此，对于 UDP 运行状况检查，ping 探测器始终是缺省探测器。

创建运行状况检查

以下示例创建了两个运行状况检查对象 `hc1` 和 `hc-myscript`。第一个运行状况检查使用内置的 TCP 探测器。第二个运行状况检查使用定制测试 `/var/tmp/my-script`。

```
# ilbadm create-healthcheck \
-h hc-timeout=3,hc-count=2,hc-interval=8,hc-test=tcp hc1
# ilbadm create-healthcheck -h hc-timeout=3, \
hc-count=2,hc-interval=8,hc-test=/var/tmp/my-script hc-myscript
```

每个参数的说明如下：

`hc-timeout` 指定将运行状况检查视为失败（如果未完成）时的超时。

`hc-count` 指定尝试执行 `hc-test` 运行状况检查的次数。

`hc-interval` 指定连续的运行状况检查之间的间隔。为避免同步，将会随机生成介于 `0.5 * hc-interval` 和 `1.5 * hc-interval` 之间的实际间隔。

`hc-test` 指定运行状况检查的类型。

注 – `hc-test` 的端口规范是使用 `create-rule` 子命令中的 `hc-port` 关键字指定的。有关详细信息，请参阅 `ilbadm(1M)` 手册页。

用户提供的测试详细信息

用户提供的定制测试必须符合以下条件：

- 测试可以是二进制文件或脚本。
- 测试可以位于系统中的任何位置；当您使用 `create-healthcheck` 子命令时，必须指定绝对路径。

  在 `create-rule` 子命令中将测试（例如，`/var/tmp/my-script`）指定为运行状况检查规范的一部分时，`ilbd` 守护进程将派生一个进程并执行测试，如下所示：

  ```
  /var/tmp/my-script $1 $2 $3 $4 $5
  ```

  每个参数的说明如下：

  `$1` VIP（数值 IPv4 或 IPv6 地址）

  `$2` 服务器 IP（数值 IPv4 或 IPv6 地址）

  `$3` 协议（字符串形式的 UDP、TCP）

  `$4` 数字端口范围（用户指定的 `hc-port` 值）

  `$5` 测试在返回失败之前必须等待的最长时间（秒）。如果测试运行超出了指定的时间，则可能会停止测试并认为测试失败。该值是用户定义的并在 `hc-timeout` 中指定。

用户提供的测试 `my-script` 可能会也可能不会使用所有参数，但**必须**返回以下各项之一：

- 以微秒为单位的往返时间 (round-trip time, RTT)
- 0（如果测试不计算 RTT）
- -1（失败）

缺省情况下，运行状况检查测试使用以下特权运行：`PRIV_PROC_FORK`、`RIV_PROC_EXEC` 和 `RIV_NET_ICMPACCESS`。

如果需要更广泛的特权集，则必须在测试中实现 `setuid`。有关特权的更多详细信息，请参阅 `privileges(5)` 手册页。

显示运行状况检查

您可以使用以下 `ilbadm list-healthcheck` 子命令获取有关已配置的运行状况检查的详细信息：

```
# ilbadm list-healthcheck
```

以下输出样例列出了所配置的两个运行状况检查。

NAME	TIMEOUT	COUNT	INTERVAL	DEF_PING	TEST
hc1	3	2	8	Y	tcp
hc2	3	2	8	N	/var/usr-script

显示运行状况检查结果

您可以使用 `ilbadm list-hc-result` 子命令获取运行状况检查结果。如果未指定规则或运行状况检查，该子命令将列出所有运行状况检查。

以下示例显示了与名为 `rule1` 的规则关联的运行状况检查结果：

```
# ilbadm show-hc-result rule1
```

RULENAME	HCNAME	SERVERID	STATUS	FAIL	LAST	NEXT	RTT
rule1	hc1	_sg1:0	dead	10	11:01:19	11:01:27	941
rule1	hc1	_sg1:1	alive	0	11:01:20	11:01:34	1111

表中的 LAST 列显示完成服务器运行状况检查的时间。NEXT 列显示完成下一个服务器运行状况检查的时间。

删除运行状况检查

以下示例删除一个名为 *hc1* 的运行状况检查：

```
# ilbadm delete-healthcheck hc1
```

管理 ILB 规则

在 ILB 中，虚拟服务通过负载平衡规则表示，并通过以下参数进行定义。

- 虚拟 IP 地址
- 传输协议：TCP 或 UDP
- 端口号（或端口范围）
- 负载平衡算法
- 负载平衡模式的类型（DSR、全 NAT 或半 NAT）
- 包含一组后端服务器的服务器组
- 可对服务器组中的每个服务器执行的可选服务器运行状况检查
- 用于运行状况检查的可选端口

注 – 您可以对特定端口指定运行状况检查，也可以对 `ilbd` 守护进程从服务器端口范围中随机选择的任意端口指定运行状况检查。

- 用于表示虚拟服务的规则名称

本节介绍如何使用 `ilbadm` 命令创建、删除和列出负载平衡规则。

列出 ILB 规则

要列出规则的配置详细信息，请使用 `ilbadm show-rule` 子命令。如果未指定规则名称，将提供所有规则的信息。

```
# ilbadm show-rule
```

以下为命令输出样例。

RULENAME	STATUS	LBALG	TYPE	PROTOCOL	VIP	PORT
rule-http	E	hash-ip-port	HALF-NAT	TCP	10.0.0.1	80
rule-dns	D	hash-ip	DSR	UDP	10.0.0.1	53
rule-abc	D	roundrobin	NAT	TCP	2003::1	1024
rule-xyz	E	hash-ip-vip	NAT	TCP	2003::1	2048-2050

▼ 如何创建 ILB 规则

1. **创建一个包含相应的后端服务器的服务器组。**

   ```
   # ilbadm create-servergroup -s server=server1:port-range1,server2:port-range2  sg1
   ```

2. **如果要将服务器运行状况检查与规则关联，请创建一个运行状况检查。**

   ```
   # ilbadm create-healthcheck -h hc-test=protocol, \
   hc-timeout=value1,hc-count=value2 \
   ,hc-interval=value3 hc1
   ```

3. **确定要与规则关联的 VIP、端口和可选协议。**

   这些是使用 `-i` 选项指定的。

4. **选择要使用的操作（DSR、半 NAT 或全 NAT）。**

   如果选择了 NAT，则必须指定要用作 `proxy-src` 地址的 IP 地址范围。全 NAT 拓扑的范围不超过 10 个 IP 地址。

5. **选择要使用的负载平衡算法。**

   可以在 `-m` 选项中指定步骤 4 和步骤 5 中的参数。有关更多信息，请参见第 138 页中的 “ILB 算法”。

6. **选择其他可选功能。**

   有关更多信息，请参见 `ilbadm(1M)` 手册页。

7. **选择规则名称。**

8. **创建并启用规则。**

   有关各选项的更多信息，请参见 `ilbadm(1M)` 手册页。

   ```
   # ilbadm create-rule -e -i vip=ipaddr,port=port,protocol=protocol \
   -m lbalg=lb-algorithm,type=topology-type,proxy-src=ipaddr1-ipaddr2, \
   pmask=value4 -h hc-name=hc1 \
   -o servergroup=sg1  rule1
   ```

   以下示例显示用于创建包含运行状况检查的全 NAT 规则的步骤。

示例 12–6 创建具有运行状况检查会话持久性的全 NAT 规则

此示例创建一个名为 `hc1` 的运行状况检查，以及一个名为 `sg1` 的服务器组。该服务器组包含两个服务器，每个服务器具有一系列端口。最后一个命令创建并启用一个名为 `rule1` 的规则，并将该规则与服务器组和运行状况检查进行关联。此规则实施全 NAT 操作模式。请注意，创建该规则之前，必须先创建服务器组和运行状况检查。

```
# ilbadm create-healthcheck -h hc-test=tcp,hc-timeout=2, \
hc-count=3,hc-interval=10  hc1
# ilbadm create-servergroup -s server=60.0.0.10:6000-6009,60.0.0.11:7000-7009 sg1
```

```
# ilbadm create-rule -e -i vip=81.0.0.10,port=5000-5009, \
protocol=tcp -m lbalg=rr,type=NAT, \
proxy-src=60.0.0.101-60.0.0.104,persist=24 \
-h hc-name=hc1 -o servergroup=sg1 rule1
```

创建半 NAT 或全 NAT 规则时，请指定 connection-drain 超时的值。conn-drain 超时的缺省值为 0，这意味着它将持续等待，直到连接正常关闭为止。

删除 ILB 规则

要删除规则，请使用 ilbadm delete-rule 子命令。要删除所有规则，请使用 -a 选项。以下示例删除名为 rule1 的规则：

```
# ilbadm delete-rule rule1
```

显示 ILB 统计信息

本节介绍如何使用 ilbadm 命令获取相关信息，例如服务器的打印统计信息或规则的统计信息。还可以显示 NAT 表信息和会话持久性映射表。

获取统计信息

使用 ilbadm show-statistics 子命令可查看负载分布详细信息。以下示例显示了 show-statistics 子命令的用法：

```
# ilbadm show-statistics
 PKT_P   BYTES_P   PKT_U   BYTES_U   PKT_D   BYTES_D
  9       636       0       0       0       0
```

PKT_P　已处理的包数

BYTES_P　已处理的字节数

PKT_U　未处理的包数

BYTES_U　未处理的字节数

PKT_D　已丢弃的包数

BYTES_D　已丢弃的字节数

显示 NAT 连接表

使用 `ilbadm show-nat` 子命令可显示 NAT 连接表。在连续运行此命令时，不应对元素的相对位置进行任何假设。例如，执行 `{{ ilbadm show-nat 10}}` 两次并不能保证显示同样的 10 个项两次，尤其是在繁忙的系统上。如果未指定计数值，则会显示整个 NAT 连接表。

示例 12–7 NAT 连接表项

以下示例显示了 NAT 连接表中的五个项。

```
# ilbadm show-nat 5
UDP: 124.106.235.150.53688 > 85.0.0.1.1024 >>> 82.0.0.39.4127 > 82.0.0.56.1024
UDP: 71.159.95.31.61528 > 85.0.0.1.1024 >>> 82.0.0.39.4146 > 82.0.0.55.1024
UDP: 9.213.106.54.19787 > 85.0.0.1.1024 >>> 82.0.0.40.4114 > 82.0.0.55.1024
UDP: 118.148.25.17.26676 > 85.0.0.1.1024 >>> 82.0.0.40.4112 > 82.0.0.56.1024
UDP: 69.219.132.153.56132 > 85.0.0.1.1024 >>> 82.0.0.39.4134 > 82.0.0.55.1024
```

项格式如下：

```
T: IP1 > IP2 >>> IP3 > IP4
```

`T` 此项中使用的传输协议

`IP1` 客户机的 IP 地址和端口

`IP2` VIP 和端口

`IP3` 对于半 NAT 模式，是客户机的 IP 地址和端口。

对于全 NAT 模式，是客户机的 IP 地址和端口。

`IP4` 后端服务器的 IP 地址和端口。

显示会话持久性映射表

使用 `ilbadm show-persist` 子命令可显示会话持久性映射表。

示例 12–8 会话持久性映射表项

以下示例显示了会话持久性映射表中的五个项：

```
# ilbadm show-persist 5
rule2: 124.106.235.150 --> 82.0.0.56
rule3: 71.159.95.31 --> 82.0.0.55
rule3: 9.213.106.54 --> 82.0.0.55
rule1: 118.148.25.17 --> 82.0.0.56
rule2: 69.219.132.153 --> 82.0.0.55
```

项格式如下：

示例 12–8 会话持久性映射表项 （续）

```
R: IP1 --> IP2
```

R 与该持久性项绑定的规则。

IP1 客户机的 IP 地址。

IP2 后端服务器的 IP 地址。

◆ ◆ ◆ 第 13 章

13 虚拟路由器冗余协议（概述）

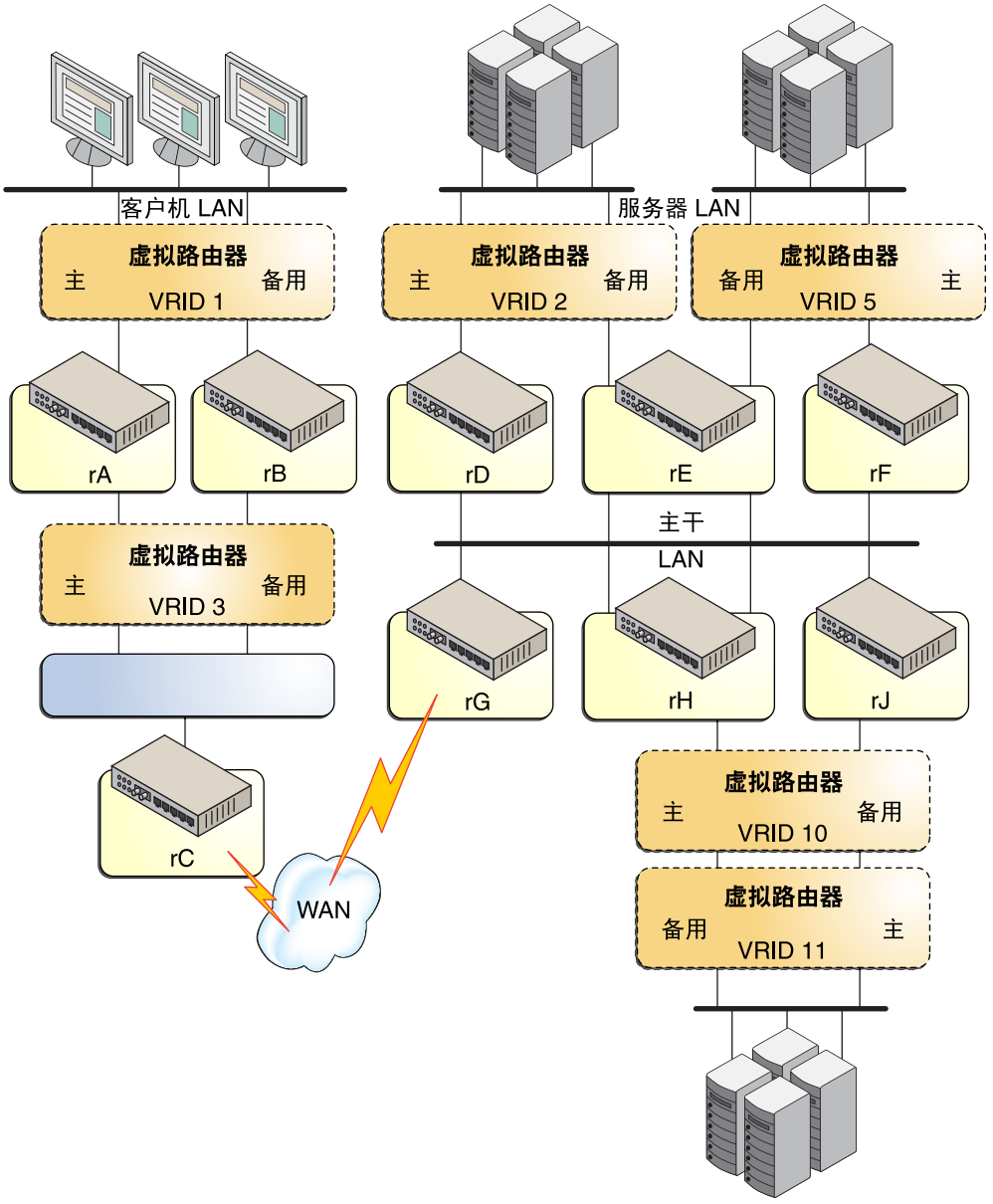
虚拟路由器冗余协议 (Virtual Router Redundancy Protocol, VRRP) 是 Virtual Router Redundancy Protocol Version 3 for IPv4 and IPv6（IPv4 和 IPv6 的虚拟路由器冗余协议版本 3）中指定的 Internet 标准协议。Oracle Solaris 中支持 VRRP 以提供高可用性。Oracle Solaris 提供了配置和管理 VRRP 服务的管理工具。

本章包含以下主题：

VRRP 的工作原理

下图显示了 VRRP 的工作原理。图后的文本对图中使用的 VRRP 组件进行了解释。

图 13–1 VRRP的工作原理



在上图中，VRRP 使用了以下组件：

- 路由器 rA 是虚拟路由器 VRID 1 的主路由器和 VRID 3 的备用路由器。路由器 rA 处理发送到 VRID 1 的 VIP 的包路由，并准备好承担 VRID 3 的路由职责。
- 路由器 rB 是虚拟路由器 VRID 3 的主路由器和 VRID 1 的备用路由器。路由器 rB 处理发送到 VRID 3 的 VIP 的包路由，并准备好承担 VRID 1 的路由职责。
- 路由器 rC 没有 VRRP 功能，但它使用 VRID 3 的 VIP 访问客户机 LAN 子网。
- 路由器 rD 是 VRID 2 的主路由器。路由器 rF 是 VRID 5 的主路由器。路由器 rE 是这两个 VRID 的备用路由器。如果 rD 或 rF 出现故障，rE 将成为该 VRID 的主路由器。rD 和 rF 可能会同时出现故障。VRRP 路由器是一个 VRID 的主路由器的事实不妨碍将其作为其他 VRID 的主路由器。
- 路由器 rG 是主干 LAN 的广域网 (wide area network, WAN) 网关。连接到主干的所有路由器使用动态路由协议（如开放式最短路径优先 (Open Shortest Path First, OSPF)）与 WAN 上的路由器共享路由信息。此方面不涉及 VRRP，尽管路由器 rC 通告客户机 LAN 子网的路径要通过 VRID 3 的 VIP。
- 路由器 rH 是 VRID 10 的主路由器，是 VRID 11 的备用路由器。同样，路由器 rJ 是 VRID 11 的主路由器，是 VRID 10 的备用路由器。此 VRRP 负载分担配置说明单个路由器接口上可以存在多个 VRID。

可以在网络设计中使用 VRRP 为网络中的所有系统提供几乎完全路由冗余。

在局域网中使用 VRRP

建立网络（如局域网 (local area network, LAN)）时，提供高可用性服务非常重要。提高网络可靠性的一种方式是为网络中的重要组件提供备份。向网络中添加路由器、交换机和链路之类的组件可确保出现故障时继续提供服务。在网络端点提供冗余是一项重要任务，可以使用 VRRP 轻松实现。可以使用 VRRP 在 LAN 中引入虚拟路由器，从而提供路由器故障恢复。

VRRP 是一种选举协议，用于将虚拟路由器的职责动态地分配给 LAN 内的某个 VRRP 路由器。VRRP 为 LAN 上静态配置的路由器提供一个或多个备用路由器。

称为**主路由器**的 VRRP 路由器控制与虚拟路由器关联的一个或多个 IPv4 或 IPv6 地址。虚拟路由器转发发送到主路由器的 IP 地址的包。

选举过程提供动态故障转移，同时转发发送到这些 IP 地址的包。VRRP 消除了静态缺省路由环境中固有的单点故障。

通过使用 Oracle Solaris 中的 VRRP 功能，可以为路由过程提供可用性更高的缺省路径，而无需在每个终端主机上配置动态路由或路由器搜索协议。

VRRP 路由器

VRRP 运行在每个 VRRP 路由器上，用于管理路由器的状态。一个主机可以配置多个 VRRP 路由器，其中每个 VRRP 路由器属于不同的虚拟路由器。

VRRP 路由器具有以下属性：

- **路由器名称**－系统范围的唯一标识符
- **虚拟路由器 ID (Virtual Router ID, VRID)**－用于在给定网段中标识虚拟路由器的唯一编号。VRID 标识 LAN 内的虚拟路由器。
- **主 IP 地址**－VRRP 通告的源 IP 地址
- **虚拟 IP 地址 (VRIP)**－与 VRID 关联的 IP 地址，其他主机可通过该地址获取网络服务。VRIP 由属于 VRID 的 VRRP 实例管理。
- **VRRP 参数**－包括优先级、通告间隔、抢占模式和接受模式
- **VRRP 状态信息和统计信息**

管理 VRRP 子命令

以下各节概述了 `vrrpadm` 子命令。有关详细信息，请参见 `vrrpadm(1M)` 手册页。除了 `vrrpadm show-router` 子命令外，所有子命令的结果都是持久性的。例如，`vrrpadm create-router` 创建的 VRRP 路由器在重新引导后将继续存在。

VRRP VNIC 创建

在系统的物理网络适配器之上配置的伪网络接口也称为网络接口卡 (network interface card, NIC)。一个物理接口可以有多个 VNIC。VNIC 是网络虚拟化的重要组件。有关更多信息，请参见《在 Oracle Solaris 11.1 中使用虚拟网络》。

现有的 `dladm create-vnic` 子命令已扩展为允许创建 VRRP VNIC。其语法如下所示：

```
# dladm create-vnic [-t] [-R root-dir] [-l link] [-m vrrp -V VRID -A \
{inet | inet6}] [-v vlan-id] [-p prop=value[,...]] vnic-link
```

引入了新的 VNIC 地址类型 `vrrp`。您必须使用此新 VNIC 地址类型指定 VRID 和地址族。

因此，将创建具有已知虚拟路由器 MAC 地址的 VNIC。

创建路由器

vrrpadm create-router 子命令使用指定的 VRID 和地址族及其他指定参数创建 VRRP 路由器。每个 VRRP 路由器都需要创建一个特殊的 VRRP VNIC，可以使用 dladm create-vnic 命令创建该 VNIC。有关更多信息，请参见 vrrpadm(1M) 手册页。其语法如下所示：

```
# vrrpadm create-router -V vrid -l link -A {inet | inet6} \
[-p priority] [-i adv-interval] [-o flags] router-name
```

-o 选项用于配置 VRRP 路由器的抢占模式和接受模式。值可以是：preempt、un_preempt、accept、no_accept。缺省情况下，这两种模式都设置为 true。

router-name 用作此 VRRP 路由器的唯一标识符并用于其他 vrrpadm 子命令中。路由器名称中允许使用的字符包括：字母数字字符 (a-z, A-Z, 0-9) 和下划线 ('_')。路由器名称的最大长度为 31 个字符。

启用路由器

已禁用的 VRRP 路由器可以使用 enable-router 子命令重新启用。启用路由器时，在其上创建 VRRP 路由器的底层数据链路（当使用 vrrpadm create-router 创建路由器时使用 -l 选项指定）和路由器的 VRRP VNIC 必须存在。否则，启用操作将失败。其语法如下所示：

```
# vrrpadm enable-router router-name
```

修改路由器

vrrpadm modify-router 子命令更改指定 VRRP 路由器的配置。其语法如下所示：

```
# vrrpadm modify-router [-p priority] [-i adv-interval] \
[-o flags] router-name
```

显示路由器配置

vrrpadm show-router 子命令显示指定 VRRP 路由器的配置和状态。有关更多详细信息，请参见 vrrpadm(1M) 手册页。其语法如下所示：

```
# vrrpadm show-router [-P | -x] [-p] [-o field[,...]] [router-name]
```

以下是 vrrpadm show-router 输出的示例：

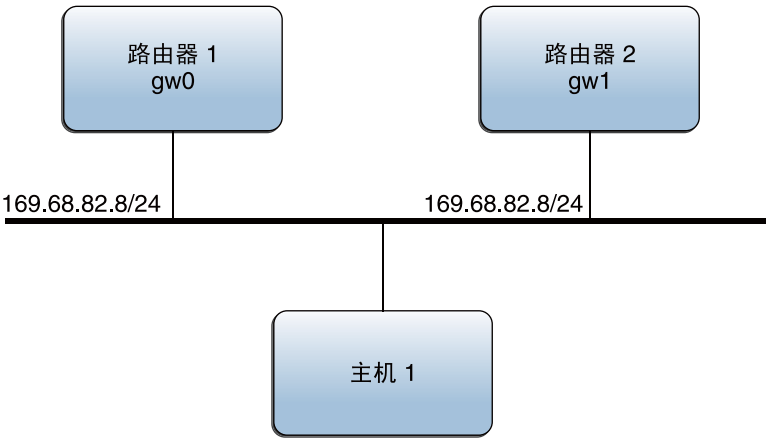
```
# vrrpadm show-router vrrp1
NAME VRID LINK AF PRIO ADV_INTV MODE STATE VNIC
vrrp1 1 bge1 IPv4 100 1000 e-pa- BACK vnic1
```

```
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK MAST 1m17s vnic1 10.0.0.100 10.0.0.1

# vrrpadm show-router -P vrrp1
NAME PEER P_PRIO P_INTV P_ADV_LAST M_DOWN_INTV
vrrp1 10.0.0.123 120 1000 0.313s 3609
```

示例 13–1 VRRP配置示例

下图显示了典型的VRRP配置。



在本示例中，IP地址169.68.82.8配置为host1的缺省网关。此IP地址是虚拟IP地址，受由以下两个VRRP路由器组成的虚拟路由器保护：router1和router2。在同一时间，两个路由器中只能有一个充当主路由器并承担虚拟路由器的职责，同时转发来自host1的包。

假定虚拟路由器的VRID是12。以下示例显示了用于在router1和router2上配置上述VRRP配置的命令。router1是虚拟IP地址169.68.82.8的所有者，其优先级为缺省值(255)。router2是优先级为100的备用路由器。

```
router1:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw0 vnic1
# vrrpadm create-router -V 12 -A inet -l gw0 vrrp1
# ipadm create-addr -d -a 169.68.82.8/24 vnic1/router1
# ipadm create-addr -d -a 169.68.82.100/24 gw0/router1
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
router2:
# dladm create-vnic -m vrrp -V 12 -A inet -l gw1 vnic1
# vrrpadm create-router -V 12 -A inet -l gw1 -p 100 vrrp1
# ipadm create-addr -d -a 169.68.82.8/24 vnic1/router2
```

示例 13–1 VRRP配置示例 （续）

```
# ipadm create-addr -d -a 169.68.82.101/24 gw1/router2
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 BACK INIT 2m32s vnic1 169.68.82.101 169.68.82.8
```

以 router1 的配置为例，必须在 gw0 上至少配置一个 IP 地址。在以下示例中， router1 的此 IP 地址为主 IP 地址，用于发送 VRRP 通告包：

```
# vrrpadm show-router -x vrrp1
NAME STATE PRV_STAT STAT_LAST VNIC PRIMARY_IP VIRTUAL_IPS
vrrp1 MAST BACK 1m17s vnic1 169.68.82.100 169.68.82.8
```

禁用路由器

VRRP 路由器只有启用后才会运行。缺省情况下，首次创建 VRRP 路由器时会将其启用。但有时临时禁用 VRRP 路由器也很有用，这样可以进行相应配置更改，然后重新启用该路由器。其语法如下所示：

```
# vrrpadm disable-router router-name
```

删除路由器

vrrpadm delete-router 子命令删除指定的 VRRP 路由器。其语法如下所示：

```
# vrrpadm delete-router router-name
```

VRRP 安全注意事项

配置 VRRP 服务需要 solaris.network.vrrp 授权。请注意，只读操作 vrrpadm showrouter 不需要此授权。

solaris.network.vrrp 授权是 Network Management（网络管理）配置文件的一部分。

VRRP 限制

本节介绍了 VRRP 限制。

专用 IP 区域支持

在每个专用 IP 区域中，当在该区域中创建任何 VRRP 路由器时会自动启用 VRRP 服务 `svc:/network/vrrp/default`。VRRP 服务管理该特定区域的 VRRP 路由器。

但是，由于以下原因，对专用 IP 区域的支持有限：

- 无法在非全局区域内创建 VNIC。所以，必须先在全局区域中创建 VRRP VNIC。然后必须将该 VNIC 指定给 VRRP 路由器所在的非全局区域。然后，可以使用 `vrrpadm` 命令在非全局区域中创建和启动 VRRP 路由器。
- 在单个 Oracle Solaris 系统上，无法在不同区域中创建两个 VRRP 路由器以参与同一虚拟路由器。原因是 Oracle Solaris 不允许创建两个具有相同 MAC 地址的 VNIC。

与其他网络功能的相互操作

VRRP 服务无法在 IP 网络多路径 (IP network multipathing, IPMP) 接口上工作。VRRP 需要特定的 VRRP MAC 地址，但 IPMP 完全在 IP 层中工作。

此外，VRRP 虚拟 IP 地址只能以静态方式配置，无法通过针对 IP 地址的两个现有自动配置工具来自动配置：`in.ndpd` 用于 IPv6 自动配置，`dhcpagent` 用于 DHCP 配置。因为主 VRRP 路由器和备用 VRRP 路由器 (VNIC) 共享同一 MAC 地址，因此 `in.ndpd` 和 `dhcpagent` 可能会无法区别。最终会发生意外结果。因此，VRRP VNIC 不支持 IPv6 自动配置和 DHCP 配置。如果在 VRRP VNIC 上配置 IPv6 自动配置或 DHCP，则尝试启动自动配置的 IP 地址时将失败，自动配置操作也将失败。

◆ ◆ ◆ 附 录 A

链路聚合类型：功能比较

中继聚合和数据链路多路径 (datalink multipathing, DLMP) 聚合支持几乎相同的功能，以提高网络性能。但是，二者确实存在差异。下表列出了对 Oracle Solaris 中两种链路聚合类型的一般比较。

功能	中继聚合	DLMP 聚合
基于链路的故障检测	支持	支持
链路聚合控制协议 (Link Aggregation Control Protocol, LACP)	支持	不支持
使用备用接口	不支持	支持
跨越多个交换机	除非使用供应商专有解决方案，否则不支持	支持
交换机配置	需要	不需要
负载平衡策略	支持	不适用
跨所有聚合端口进行负荷分配	支持	有限[1]
用于资源管理的用户定义流	支持	支持
链路保护	支持	支持
背对背并行配置	支持	不支持[2]

[1]聚合跨所有端口分配其 VNIC。但是，各个 VNIC 无法在多个端口上分配负荷。

[2]DLMP 聚合必须始终使用中间交换机将包发送到其他目标系统。但是，使用 DLMP 聚合时，不要为链路聚合配置交换机。

◆ ◆ ◆ 附 录 B

链路聚合和 IPMP：功能比较

IPMP 和链路聚合是用来改进网络性能和维护网络可用性的不同技术。

下表列出了链路聚合与 IPMP 的一般比较。

功能	IPMP	链路聚合
网络技术类型	第 3 层（IP 层）	第 2 层（链路层）
配置工具	`ipadm`	`dladm`
基于链路的故障检测	支持	支持
基于探测器的故障检测	基于 ICMP，面向与测试地址位于同一 IP 子网中的任何已定义系统，跨多个级别的干预第 2 层交换机	基于 LACP，面向直接对等主机或交换机 在 DLMP 聚合中不支持
使用备用接口	支持	在中继聚合中不支持 在 DLMP 聚合中支持
跨越多个交换机	支持	在中继聚合中不支持。一些供应商提供专有的非互操作解决方案以跨越多个交换机。 在 DLMP 聚合中支持
硬件支持	不需要	对于中继聚合为必需。例如，Oracle Solaris 系统上的中继聚合也要求聚合交换机上的对应端口。 在 DLMP 聚合中不需要
链路层要求	广播功能	特定于以太网
驱动程序框架要求	无	必须使用 GLDv3 框架

功能	IPMP	链路聚合
负荷分配支持	支持，由内核控制。传入负荷分配受源地址选择间接影响。	仅在中继聚合中支持，由管理员通过 `dladm` 命令控制。支持传入负荷分配。 在 DLMP 聚合中不支持按聚合上的各接口进行负荷分配。
与 VNIC 集成时的支持级别	有限支持	出色支持
用于资源管理的用户定义流	不支持	支持
链路保护	不支持	支持
协议要求	无	无

在链路聚合中，传入通信流量以标准模式在构成聚合的多个链路上分配。因此，由于安装了更多 NIC 以将链路添加到聚合，提高了网络性能。

DLMP 聚合跨越多个交换机。作为第 2 层技术，聚合与其他 Oracle Solaris 虚拟化技术进行了很好的集成。

IPMP 的通信使用 IPMP 接口的数据地址，因为它们绑定到了可用的活动接口。举例来说，如果所有数据通信只在两个 IP 地址之间流动，但不一定在同一连接上，则添加多个 NIC 不会借助 IPMP 提高性能，因为只有两个 IP 地址保持可用。

中继聚合和 IPMP 可以互补，可以一起部署，取长补短，综合提高网络性能和可用性。

索引

L