# 在 Oracle® Solaris 11.2 中管理 TCP/IP 网络、IPMP 和 IP 隧道

文件号码 E53803
2014 年 7 月

ORACLE

# 目录

# 使用本文档

- **概述** – 介绍在 Oracle Solaris 操作系统 (operating system, OS) 中管理 TCP/IP 网络、IPMP 和 IP 隧道的任务。
- **目标读者** – 系统管理员。
- **必备知识** – 对网络管理（包括管理 TCP/IP 网络和高级网络功能，例如 IPMP 和 IP 隧道）具有丰富经验。

## 产品文档库

有关本产品的最新信息和已知问题均包含在文档库中，网址为：http://www.oracle.com/pls/topic/lookup?ctx=E56344。

## 获得 Oracle 支持

Oracle 客户可通过 My Oracle Support 获得电子支持。有关信息，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info；如果您听力受损，请访问 http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs。

## 反馈

可以在 http://www.oracle.com/goto/docfeedback 上提供有关此文档的反馈。

♦ ♦ ♦ 第 1 章

# 1 管理 TCP/IP 网络

本章介绍如何在安装了 Oracle Solaris 的系统上管理 TCP/IP 协议。本章中的任务假设您的站点拥有正常运行的 TCP/IP 网络，该网络仅启用了 IPv4 或启用了 IPv4/IPv6 双栈。

有关规划网络部署的信息，请参见《在 Oracle Solaris 11.2 中规划网络部署 》。

有关网络配置任务，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》。

有关对 TCP/IP 网络进行故障排除的一般信息，请参见《在 Oracle Solaris 11.2 中排除网络管理问题 》。

本章包含以下主题：



## 定制 TCP/IP 属性

可以使用 ipadm 命令配置大多数 TCP/IP 属性，也称为可调参数。ipadm 命令替换了 ndd 命令，用作设置可调参数的主工具。有关这些更改的更多信息，请参见《从 Oracle Solaris 10 转换至 Oracle Solaris 11.2 》中的“将 ndd 命令与 ipadm 命令进行比较”。

TCP/IP 属性可以是基于接口的或全局的，这些属性可以应用于特定接口，也可全局性地应用于区域中的所有接口。全局属性在不同的非全局区域中还可以有不同的值。有关支持的协议属性的完整列表，请参见 ipadm(1M) 手册页。

通常情况下，TCP/IP 协议的缺省值足以使网络正常工作。然而，如果这些值对于特定网络拓扑来说是不够的，则可以通过使用下面三个 ipadm 子命令来定制属性：

- `ipadm show-prop -p` *property protocol* – 显示协议的属性及属性的当前值。如果不使用 -p *property* 选项，则会显示协议的所有属性。如果不指定协议，则会显示所有协议的所有属性。
- `ipadm set-prop -p` *property=value protocol* – 为协议的属性指定值。
  - 要为某个协议属性指定多个值，请使用下面的语法：

    `ipadm set-prop [-t] -p` *property=value*`[,...] protocol`
  - 要从给定属性的一组值中删除一个值，可以使用 -= 限定符：

    `ipadm set-prop -p` *property-=value2*
- 将特定协议属性重置为其缺省值，如下所示：

  `ipadm reset-prop -p` *property protocol*

有关定制 IP 接口属性的信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“定制 IP 接口属性和地址”。

有关定制 IP 地址属性的信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“定制 IP 地址属性”。

# 全局启用包转发

当通过使用 ipadm set-ifprop 命令在某个 IP 接口上启用转发时，只会为该接口启用转发，所有其他接口上的转发保持不变。通过在单个 IP 接口属性上设置包转发，您可以选择性地在系统中的特定接口上实现此功能。有关更多信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“启用包转发”。

要在整个系统上启用包转发（不考虑 IP 接口的数量），请使用 protocol 属性。forwarding 属性是用于管理转发的全局 IP 属性，该属性与用于管理单个 IP 接口转发的属性的名称相同。因为您可以为 IPv4 或/和 IPv6 协议启用转发，所以必须单独管理每个协议。

例如，可以为系统上的所有 IPv4 和 IPv6 通信启用包转发，如下所示：

```
# ipadm show-prop -p forwarding ip
PROTO   PROPERTY     PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4    forwarding   rw     off       --           off       on,off
ipv6    forwarding   rw     off       --           off       on,off
```

```
# ipadm set-prop -p forwarding=on ipv4
# ipadm set-prop -p forwarding=on ipv6

# ipadm show-prop -p forwarding ip
PROTO   PROPERTY     PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
ipv4    forwarding   rw     on        on           off       on,off
ipv6    forwarding   rw     on        on           off       on,off
```

注 - IP 接口的 `forwarding` 属性与协议的该属性不是互斥的。您可以同时为接口和协议设置该属性。例如，可以在协议上全局性地启用包转发，然后在系统上定制每个 IP 接口的包转发。因此，尽管全局启用包转发，但您仍可以选择性地基于接口来管理包转发。

# 管理缺省地址选择

Oracle Solaris 可以让单个接口拥有多个 IP 地址。例如，使用 IPMP 之类的技术，可以将多个网络接口卡 (Network Interface Card, NIC) 连接到同一 IP 链路层。此链路可以具有一个或多个 IP 地址。此外，启用了 IPv6 的系统上的接口具有一个链路本地 IPv6 地址，至少具有一个 IPv6 路由地址，并且至少一个接口具有 IPv4 地址。

当系统启动事务时，应用程序将调用 `getaddrinfo` 套接字。`getaddrinfo` 套接字将搜索目标系统上正在使用的可能地址。然后，内核将设置此列表的优先级，以便找到包的最佳目标。此过程称为目标地址排序。然后，如果确定了包的最佳目标地址，Oracle Solaris 内核将选择相应的源地址格式。此过程称为地址选择。有关目标地址排序的更多信息，请参见 `getaddrinfo(3SOCKET)` 手册页。

仅启用了 IPv4 的系统和启用了 IPv4/IPv6 双栈的系统都必须执行缺省地址选择。大多数情况下，不需要更改缺省地址选择机制。但是，您可能需要更改地址格式的优先级，以便支持 IPMP 或首选使用 6to4 地址格式等。

## `/etc/inet/ipaddrsel.conf` 配置文件说明

`/etc/inet/ipaddrsel.conf` 文件包含 IPv6 缺省地址选择策略表。如果在安装 Oracle Solaris 时启用了 IPv6，则该文件包含表 1-1 “IPv6 地址选择策略表” 中所示的内容。

可以编辑 `/etc/inet/ipaddrsel.conf` 的内容。但是，在大多数情况下，应当避免修改此文件。如果一定要进行修改，请参阅如何管理 IPv6 地址选择策略表 [13]过程。有关 `ippaddrsel.conf` 的更多信息，请参见“修改 IPv6 地址选择策略表的原因” [12]和 `ipaddrsel.conf(4)` 手册页。

## ipaddrsel 命令说明

使用 ipaddrsel 命令，可以修改 IPv6 缺省地址选择策略表。

Oracle Solaris 内核使用 IPv6 缺省地址选择策略表为 IPv6 数据包头执行目标地址排序和源地址选择。/etc/inet/ipaddrsel.conf 文件包含该策略表。

下表列出了缺省地址的格式以及它们的策略表优先级。有关 IPv6 地址选择的技术详细信息，请参见 inet6(7P) 手册页。

表 1-1 IPv6 地址选择策略表

| 前缀 | 优先级 | 定义 |
|---|---|---|
| ::1/128 | 50 | 回送 |
| ::/0 | 40 | 缺省 |
| 2002::/16 | 30 | 6to4 |
| ::/96 | 20 | 与 IPv4 兼容 |
| ::ffff:0:0/96 | 10 | IPv4 |

在该表中，IPv6 前缀（::1/128 和 ::/0）优先于 6to4 地址 (2002::/16)、IPv4 地址（::/96 和 ::ffff:0:0/96）。因此，在缺省情况下，内核将为转至另一个 IPv6 目标的包选择接口的全局 IPv6 地址。接口的 IPv4 地址具有较低的优先级，对于转至 IPv6 目标的包尤其如此。如果给出了选定的 IPv6 源地址，内核针对目标地址也使用 IPv6 格式。

## 修改 IPv6 地址选择策略表的原因

在许多情况下，您不必更改 IPv6 缺省地址选择策略表。如果确实需要管理策略表，请使用 ipaddrsel 命令。

出于以下原因，您可能希望修改策略表：

- 如果系统中有一个用于 6to4 隧道的接口，可以为 6to4 地址指定更高的优先级。
- 如果希望与特定的目标地址进行通信时仅使用特定的源地址，可以将这些地址添加到策略表中。然后，可以使用 ipadm 命令将这些地址标记为首选地址。有关更多信息，请参见 ipadm(1M) 手册页。
- 如果希望 IPv4 地址优先于 IPv6 地址，可以将 ::ffff:0:0/96 的优先级更改为较大的数字。
- 如果需要为过时的地址指定较高的优先级，可以将过时的地址添加到策略表中。例如，现在，本地站点地址在 IPv6 中已过时。这些地址的前缀为 fec0::/10。可以更改策略表，以便为本地站点地址指定更高的优先级。

有关 ipaddrsel 命令的详细信息，请参见 ipaddrsel(1M) 手册页。

# ▼ 如何管理 IPv6 地址选择策略表

以下过程介绍如何修改地址选择策略表。有关 IPv6 缺省地址选择的概念性信息，请参见 ipaddrsel 命令说明。

注意 - 除非由于以下过程中提供的原因，否则不要更改 IPv6 地址选择策略表。这样做会导致网络因策略表构造不正确而出现问题。另外，请确保保存策略表的备份副本，如以下过程所示。

1. 成为管理员。
2. 查看当前的 IPv6 地址选择策略表。

```
# ipaddrsel
# Prefix                  Precedence Label
::1/128                           50 Loopback
::/0                              40 Default
2002::/16                         30 6to4
::/96                             20 IPv4_Compatible
::ffff:0.0.0.0/96                 10 IPv4
```

3. 备份缺省地址策略表的副本。

```
# cp /etc/inet/ipaddrsel.conf /etc/inet/ipaddrsel.conf.orig
```

4. 向 **/etc/inet/ipaddrsel.conf** 文件中添加任何定制项。

```
# pfedit /etc/inet/ipaddrsel.conf
```

针对 /etc/inet/ipaddrsel 中的各项使用以下语法：

*prefix/prefix-length precedence label [# comment ]*

有关您可能进行的一些常见修改的示例，请参见例 1-1 “修改缺省 Pv6 地址选择策略表”。

5. 将已修改的策略表加载到内核。

```
# ipaddrsel -f /etc/inet/ipaddrsel.conf
```

6. 如果已修改的策略表存在问题，请恢复缺省 IPv6 地址选择策略表。

```
# ipaddrsel -d
```

例 1-1 修改缺省 Pv6 地址选择策略表

下面是一些您可能希望对策略表进行的常见修改：

- 为 6to4 地址指定最高优先级。

```
2002::/16                       50 6to4
::1/128                         45 Loopback
```

6to4 地址格式现在具有最高优先级 50，而先前优先级为 50 的回送现在的优先级变为 45。其他地址格式保持不变。

- 指定与特定目标地址进行通信的特定源地址。

```
::1/128                         50 Loopback
2001:1111:1111::1/128           40 ClientNet
2001:2222:2222::/48             40 ClientNet
::/0                            40 Default
```

对于仅有一个物理接口的主机，此特定项非常有用。此处，`2001:1111:1111::1/128` 是发往网络 `2001:2222:2222::/48` 中目标的所有包的首选源地址。优先级 40 使得源地址 `2001:1111:1111::1/128` 的优先级高于为接口配置的其他地址格式。

- IPv4 地址优先于 IPv6 地址。

```
::ffff:0.0.0.0/96               60 IPv4
::1/128                         50 Loopback
.
.
```

IPv4 格式 `::ffff:0.0.0.0/96` 的优先级已从缺省的 10 更改为 60，这是表中的最高优先级。

## ▼ 如何仅修改当前会话的 IPv6 地址选择表

编辑 `/etc/inet/ipaddrsel.conf` 文件时，所做的任何修改即使在重新引导系统之后也都会保留下来。如果希望已修改的策略表仅存在于当前会话中，请使用以下过程。

1. **成为管理员。**

2. **将 `/etc/inet/ipaddrsel` 文件的内容复制到 *filename*，其中 *filename* 表示您选择的任何名称。**

```
# cp /etc/inet/ipaddrsel filename
```

3. **根据需要使用 `pfedit` 命令在 *filename* 中编辑策略表。**

4. **将已修改的策略表加载到内核。**

```
# ipaddrsel -f filename
```

内核将使用新的策略表，直到重新引导系统。

# 管理传输层服务

以下传输层协议是 Oracle Solaris 的标准部分：传输控制协议 (Transmission Control Protocol, TCP)、流控制传输协议 (Stream Control Transmission Protocol, SCTP) 和用户数据报协议 (User Datagram Protocol, UDP)。这些协议通常无需进行干预即可正常运行。但是，站点上的具体情况可能要求您记录或修改通过这些传输层协议运行的服务。在这种情况下，必须通过使用服务管理工具 (Service Management Facility, SMF) 命令来修改这些服务的配置文件。

`inetd` 守护进程负责在系统引导时启动标准 Internet 服务。这些服务包括将 TCP、SCTP 或 UDP 用作传输层协议的应用程序。可以使用 SMF 命令修改现有的 Internet 服务或添加新服务。

涉及传输层协议的操作包括：

- 设置特权端口
- 实施通信拥塞控制
- 记录所有的传入 TCP 连接
- 添加通过传输层协议（例如 SCTP）运行的服务
- 为访问控制配置 TCP 包装器工具
- 更改 TCP 接收缓冲区大小

有关更多信息，请参见《在 Oracle Solaris 11.2 中管理系统服务 》中的“修改 inetd 控制的服务”和 `inetd(1M)` 手册页。

## 设置特权端口

在传输协议（例如 TCP、UDP 和 SCTP）上，端口 `1-1023` 是缺省的特权端口。要绑定到特权端口，必须使用 `root` 权限运行该过程。缺省情况下，大于 `1023` 的端口为非特权端口。可以使用 `ipadm` 命令扩展特权端口的范围，也可以将非特权范围内的特定端口标记为特权端口。

要管理特权端口的范围，可以定制以下传输协议属性：

`smallest_nonpriv_port` 指定一个值，该值指示一般用户可以绑定到的非特权端口号的范围起始值。可以将非特权范围内的各个端口设置为特权端口。使用 `ipadm show-prop` 命令可显示属性的值。

`extra_priv_ports` 指定还为特权范围外的哪些端口授予特权。使用 `ipadm set-prop` 命令可指定要限制的端口。可为该属性指定多个值。

例如，假设要将 TCP 端口 `3001` 和 `3050` 设置为特权端口，并且仅限 `root` 角色访问。`smallest_nonpriv_port` 属性指示 `1024` 是非特权端口的最小端口号。因此，可以按如下方式将指定的端口 `3001` 和 `3050` 更改为特权端口：

```
# ipadm show-prop -p smallest_nonpriv_port tcp
PROTO PROPERTY               PERM   CURRENT   PERSISTENT   DEFAULT   POSSIBLE
tcp   smallest_nonpriv_port   rw     1024      --           1024      1024-32768

# ipadm show-prop -p extra_priv_ports tcp
PROTO   PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp     extra_priv_ports   rw     2049,4045   --           2049,4045   1-65535

# ipadm set-prop -p extra_priv_ports+=3001 tcp
# ipadm set-prop -p extra_priv_ports+=3050 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO   PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp     extra_priv_ports   rw     2049,4045   3001,3050    2049,4045   1-65535
                                  3001,3050
```

可以按如下方式删除特权端口（例如 4045）：

```
# ipadm set-prop -p extra_priv_ports-=4045 tcp
# ipadm show-prop -p extra_priv_ports tcp
PROTO   PROPERTY           PERM   CURRENT     PERSISTENT   DEFAULT     POSSIBLE
tcp     extra_priv_ports   rw     2049,3001   3001,3050    2049,4045   1-65535
                                  3050
```

# 实施通信拥塞控制

当节点发送的包数超过网络可以容纳的量时，通常会出现路由器缓冲区溢出形式的网络拥塞。各种算法通过对发送系统实施控制来防止通信拥塞。Oracle Solaris 中支持这些算法，并且可以轻松地在操作系统中添加或直接插入这些算法，下表介绍了所支持的内置算法。

| 算法 | Oracle Solaris 名称 | 说明 |
|---|---|---|
| NewReno | `newreno` | Oracle Solaris 中的缺省算法。此控制机制包括发送者的拥塞窗口、慢速启动和拥塞避免。 |
| HighSpeed | `highspeed` | 针对高速网络的 NewReno 的最广为人知和最简单的修改版之一。 |
| CUBIC | `cubic` | Linux 2.6 中当前的缺省算法。将拥塞避免阶段从线性窗口增加更改为 cubic 函数。 |
| Vegas | `vegas` | 基于延迟的经典算法，该算法尝试预测拥塞而不会引起实际包损失。 |

通过设置以下控制相关的 TCP 属性来启用拥塞控制。尽管下面列出的属性为 TCP 的属性，但由这些属性启用的控制机制也适用于 SCTP 通信。

`cong_enabled` 包含系统中当前运行的算法列表（以逗号分隔）。您可以添加或删除相应算法以仅启用要使用的那些算法。此属性可以有多个值。因

此，必须使用 += 限定符或 -= 限定符，具体取决于您要做出的更改。

`cong_default` 当应用程序未在套接字选项中显式指定算法时，缺省情况下使用该算法。`cong_default` 属性的值同时适用于全局和非全局区域。

以下示例演示如何将拥塞控制算法添加到协议中：

```
# ipadm set-prop -p cong_enabled+=algorithm tcp
```

按照以下方式删除算法：

```
# ipadm set-prop -p cong_enabled-=algorithm tcp
```

按照以下方式替换缺省算法：

```
# ipadm set-prop -p cong_default=algorithm tcp
```

---

注 - 添加或删除算法时不遵循任何序列规则。您可以先删除属性的算法，然后再将其他算法添加到该属性。但是，`cong_default` 属性必须始终具有已定义的算法。

---

以下示例演示您可能实施拥塞控制的方式。在此示例中，将 TCP 协议的缺省算法从 `newreno` 更改为 `cubic`。然后，从已启用的算法列表中删除 `vegas` 算法。

```
# ipadm show-prop -p extra_priv_ports tcp
PROTO PROPERTY              PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
tcp   extra_priv_ports      rw   2049,4045    --           2049,4045    1-65535

# ipadm show-prop -p cong_default,cong_enabled tcp
PROTO PROPERTY              PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
tcp   cong_default          rw   newreno      --           newreno      newreno,cubic,
                                                                       highspeed,vegas
tcp   cong_enabled          rw   newreno,cubic, newreno,cubic, newreno  newreno,cubic,
                                 highspeed,   highspeed,                highspeed,vegas
                                 vegas        vegas

# ipadm set-prop -p cong_enabled-=vegas tcp
# ipadm set-prop -p cong_default=cubic tcp

# ipadm show-prop -p cong_default,cong_enabled tcp
PROTO PROPERTY              PERM CURRENT      PERSISTENT   DEFAULT      POSSIBLE
tcp   cong_default          rw   cubic        cubic        newreno      newreno,cubic,
                                                                       highspeed
tcp   cong_enabled          rw   newreno,cubic, newreno,cubic, newreno  newreno,cubic,
                                 highspeed    highspeed                highspeed,vegas
```

# 记录所有传入 TCP 连接的 IP 地址

所有传入 TCP 连接的 IP 地址由 `syslog` 服务在 `daemon.notice` 工具和级别上进行记录。除非您更改了本地 `syslog.conf` 文件设置，否则该信息位于 `/var/adm/messages` 中。请

注意，更改 syslog.conf 文件设置会影响该信息的存储位置。有关更多详细信息，请参见 syslog.conf(4) 手册页。

对于 inetd 守护进程管理的所有服务，按如下方式将 TCP 跟踪设置为 TRUE（启用）：

```
# inetadm -M tcp_trace=TRUE
```

# 添加使用 SCTP 协议的服务

流控制传输协议 (Stream Control Transmission Protocol, SCTP) 以与 TCP 类似的方式为应用层协议提供服务。但是，SCTP 允许单方或双方为多宿主系统的两个系统进行通信。SCTP 连接称为关联。在关联中，应用程序将要传输的数据分为一个或多个消息流，也称为多流化。SCTP 连接可以转到有多个 IP 地址的端点，这对电话应用程序尤其重要。如果站点使用 IP 过滤器或 IPsec，则 SCTP 的多宿主功能是出于安全考虑。sctp(7P) 手册页介绍了其中一些安全方面的注意事项。

## ▼ 如何添加使用 SCTP 协议的服务

缺省情况下，SCTP 包括在 Oracle Solaris 中，且不需要其他配置。但是，可能需要显式配置某些应用层服务才能使用 SCTP。echo 和 discard 就是这样的应用程序示例。以下过程说明如何添加使用 SCTP 一对一样式套接字的回显服务。可以使用同一过程为 TCP 和 UDP 添加服务。

以下任务说明如何将 inetd 守护进程管理的 SCTP inet 服务添加到 SMF 系统信息库。然后，该任务说明如何使用 SMF 命令添加该服务。

开始之前 执行以下过程之前，请为服务创建清单文件。该过程以 echo 服务的清单 echo.sctp.xml 为例。

1. **使用拥有系统文件的写入特权的用户帐户，登录到本地系统。**

2. **通过使用 pfedit 命令将新服务的定义添加到 /etc/services 文件中。**

   请参见 pfedit(1M) 手册页。

   对于服务定义，使用以下语法：

   *service-name port/protocol aliases*

3. **转到存储服务清单的目录，然后导入服务清单。**

   ```
   # cd dir-name
   # svccfg import service-manifest-name
   ```

   例如，可以按照以下方式，使用位于 service.dir 目录中的清单 echo.sctp.xml 添加新 SCTP echo 服务：

```
# cd service.dir
# svccfg import echo.sctp.xml
```

4. 验证是否已添加服务清单。

   ```
   # svcs FMRI
   ```

   对于 *FMRI* 参数，使用服务清单的故障管理资源标识符 (Fault Managed Resource Identifier, FMRI)。

5. 列出服务的属性以确定是否需要进行修改。

   ```
   # inetadm -l FMRI
   ```

6. 启用新服务。

   ```
   # inetadm -e FMRI
   ```

7. 验证是否已启用该服务。

例 1-2 添加使用 SCTP 传输协议的服务

以下示例给出要使用的命令以及使 `echo` 服务使用 SCTP 所需的文件项。

```
# cat /etc/services
.
.
echo            7/tcp
echo            7/udp
echo             7/sctp

# cd service.dir

# svccfg import echo.sctp.xml

# svcs network/echo*
STATE          STIME    FMRI
disabled       15:46:44 svc:/network/echo:dgram
disabled       15:46:44 svc:/network/echo:stream
disabled       16:17:00 svc:/network/echo:sctp_stream

# inetadm -l svc:/network/echo:sctp_stream
SCOPE    NAME=VALUE
         name="echo"
         endpoint_type="stream"
         proto="sctp"
         isrpc=FALSE
         wait=FALSE
         exec="/usr/lib/inet/in.echod -s"
         user="root"
default  bind_addr=""
default  bind_fail_max=-1
```

```
default  bind_fail_interval=-1
default  max_con_rate=-1
default  max_copies=-1
default  con_rate_offline=-1
default  failrate_cnt=40
default  failrate_interval=60
default  inherit_env=TRUE
default  tcp_trace=FALSE
default  tcp_wrappers=FALSE

# inetadm -e svc:/network/echo:sctp_stream

# inetadm | grep echo
disabled  disabled       svc:/network/echo:stream
disabled  disabled       svc:/network/echo:dgram
enabled   online         svc:/network/echo:sctp_stream
```

# 配置 TCP 包装器

TCP 包装器介于守护进程和传入的服务请求之间，为服务守护进程提供了安全措施。TCP 包装器记录成功的和不成功的连接尝试。此外，TCP 包装器可以提供访问控制，根据发出请求的位置来允许或拒绝连接。可以使用 TCP 包装器来保护安全 Shell (Secure Shell, SSH)、Telnet 和文件传输协议 (File Transfer Protocol, FTP) 之类的守护进程。`sendmail` 应用程序也可以使用 TCP 包装器，如《在 Oracle Solaris 11.2 中管理 sendmail 服务》中的“sendmail 版本 8.12 支持 TCP 包装”中所述。

## ▼ 如何使用 TCP 包装器控制对 TCP 服务的访问

`tcpd` 程序可实现 *TCP* 包装器。

1. **成为 `root` 角色。**
2. **将 TCP 包装器设置为“启用”。**

   ```
   # inetadm -M tcp_wrappers=TRUE
   ```

3. **配置 TCP 包装器的访问控制策略。**

   有关说明，请参见 `/usr/sfw/man` 目录中的 `hosts_access`(3) 手册页。

# 更改 TCP 接收缓冲区大小

TCP 接收缓冲区大小是通过使用 TCP 属性 `recv_buf` 来设置的，缺省情况下为 128 KB。但是，应用程序不会均匀地使用可用带宽。因此，如果出现连接延迟，就可能会

要求您更改缺省大小。例如，使用 Oracle Solaris 的安全 Shell 功能会产生带宽使用开销，因为有其他校验和与加密进程在数据流上执行。因此，可能需要增加缓冲区大小。同样，为使执行批量传输的应用程序能有效使用带宽，也需要进行相同的缓冲区大小调整。

通过估算带宽延迟乘积 (bandwidth delay product, BDP)，可以计算要使用的正确接收缓冲区大小。要计算 BDP，请将可用带宽乘以连接延迟的值。

使用 `ping -s` *host* 命令可获取连接延迟的值。

合适的接收缓冲区大小接近 BDP 的值。但是，带宽使用情况还取决于各种条件。共享基础结构或争用带宽的应用程序和用户的数量可能会影响该估算值。

按照以下方式更改缓冲区大小的值：

```
# ipadm set-prop -p recv_buf=value tcp
```

以下示例显示了如何将缓冲区大小增加到 164 KB：

```
# ipadm show-prop -p recv_buf tcp
PROTO PROPERTY   PERM CURRENT   PERSISTENT   DEFAULT  POSSIBLE
tcp   recv_buf   rw   128000       --        128000   2048-1048576

# ipadm set-prop -p recv_buf=164000 tcp

# ipadm show-prop -p recv_buf tcp
PROTO PROPERTY   PERM CURRENT   PERSISTENT   DEFAULT  POSSIBLE
tcp   recv_buf   rw   164000    164000       128000   2048-1048576
```

缓冲区大小没有任何设置值是最佳的，因为最佳大小随具体情况而不同。请考虑以下示例，其中为特定条件下的每种网络设置了不同的 BDP 值：

| | |
|---|---|
| 典型的 1 Gbps 局域网 (local area network, LAN)，其中缓冲区大小的缺省值为 128 KB： | `BDP = 128 MBps * 0.001 s = 128 kB` |
| 理论上的 1Gbps 广域网 (wide area network, WAN)，有 100 毫秒的延迟： | `BDP = 128 MBps * 0.1 s = 12.8 MB` |
| 欧洲到美国链路（带宽由 `uperf` 测量） | `BDP = 2.6 MBps * 0.175 = 470 kB` |

如果无法计算 BDP，请使用以下指导：

- 对于通过 LAN 的批量传输，缓冲区大小缺省值 (128 KB) 已够用。

- 对于大部分 WAN 部署，接收缓冲区大小应在 2 MB 范围内。

注意 - 增加 TCP 接收缓冲区大小的同时会增加许多网络应用程序的内存资源占用。

# 使用 netstat 命令监视网络状态

可以使用 netstat 命令生成包含网络状态和协议统计信息的显示内容。可以用表格形式显示 TCP、SCTP 和 UDP 端点的状态，还可以显示路由表和接口信息。

netstat 命令可显示各种类型的网络数据，具体取决于您指定的命令行选项。显示的信息对于系统管理特别有用。本章中介绍了用于一些更经常执行的任务的选项。有关完整的详细信息，请参见 netstat(1M) 手册页。

本节包含以下主题：



## 按地址类型筛选 netstat 输出

可以使用 netstat 命令显示 IPv4 和 IPv6 网络状态。可通过在 /etc/default/inet_type 文件中设置 DEFAULT_IP 值或者使用 -f 选项来选择要显示的协议信息。通过永久设置 DEFAULT_IP 值，可以确保 netstat 命令仅显示 IPv4 信息。要覆盖此设置，请在命令行中使用 -f 选项。有关 inet_type 文件的更多信息，请参见 inet_type(4) 手册页。

用户还可以使用 -f 选项来指定 inet、inet6、unix（针对用于内部通信的 Unix 域套接字）或 sdp（套接字说明协议）参数。

## 显示套接字的状态

使用 netstat 命令可以查看本地主机上套接字的状态。您可以一般用户的身份指定若干个 netstat 命令选项。

按照以下方式显示连接套接字的状态：

```
% netstat
```

按照以下方式显示所有套接字（包括未连接的监听程序套接字）的状态：

```
% netstat -a
```

例 1-3 显示连接的套接字

netstat 命令的输出显示详细的统计信息。以下示例演示如何将该输出仅限于 IPv4 套接字。

```
% netstat -f inet

TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q    State
-------------------- -------------------- ----- ------ ----- ------ -----------
system-1.ssh          remote.38474     128872      0 128872      0 ESTABLISHED
system2.40721         remote.ldap      49232       0 128872      0 ESTABLISHED
```

# 按协议显示统计信息

netstat -s 选项显示 UDP、TCP、SCTP、Internet 控制消息协议 (Internet Control Message Protocol, ICMP) 和 IP 协议的统计信息。

按照以下方式显示协议状态：

```
% netstat -s
```

使用 -P 选项可以按协议筛选 netstat 命令的输出。此选项不限于传输协议。
可以使用此选项指定以下值：

- icmp
- icmpv6
- igmp
- ipv6tcp
- rawip
- sctp
- tcp
- udp

例如，可以按照以下方式按 UDP 协议筛选 netstat 输出：

```
% netstat -s -P udp

UDP     udpInDatagrams      =  3704     udpInErrors         =     0
        udpOutDatagrams     =  3703     udpOutErrors        =     0
```

**例 1-4** 显示 TCP 协议的状态

以下示例演示如何通过指定 -P 选项来显示 TCP 协议的结果。

```
% netstat -P tcp

TCP: IPv4
Local Address     Remote Address    Swind Send-Q  Rwind Recv-Q      State
----------------- -------------------- ----- ------  ----- ------     -------
lhost-1.login      abc.def.local.Sun.COM.980 49640      0     49640    0 ESTABLISHED
lhost.login        ghi.jkl.local.Sun.COM.1020 49640     1     49640    0 ESTABLISHED
remhost.1014       mno.pqr.remote.Sun.COM.nfsd 49640    0     49640    0 TIME_WAIT

TCP: IPv6
Local Address    Remote Address        Swind Send-Q Rwind Recv-Q   State If
---------------- -------------------- ------ ----- ------ ----------- -----
localhost.38983   localhost.32777       49152      0 49152      0 ESTABLISHED
localhost.32777   localhost.38983       49152      0 49152      0 ESTABLISHED
localhost.38986   localhost.38980       49152      0 49152      0 ESTABLISHED
```

# 显示网络接口状态

使用 netstat 命令的 i 选项可以显示本地系统上配置的网络接口的状态。可以使用此选项确定系统在每个网络中传输和接收的包数。

按照以下方式显示网络上的接口的状态：

```
% netstat -i
```

以下示例演示如何使用带有筛选选项的 netstat 命令来限制特定接口的输出：

```
% netstat -i -I net0 -f inet
  Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
  net0  1500 abc.oracle.com abc.oracle.com 231001 0     55856  0     0      0
```

**例 1-5** 显示网络接口状态

以下示例显示了流经主机接口的 IPv4 和 IPv6 包流的状态。例如，每次客户机尝试引导时，显示的服务器输入包计数 (Ipkts) 都会增加，而输出包计数 (Opkts) 保持不变。这种情况表示服务器正在识别来自客户机的引导请求包。但是，服务器却不知道对它们做出响应。这种混乱可能是由 hosts 或 ethers 数据库中的错误地址引起的。

如果输入包计数在一段时间内保持不变，则说明计算机根本未查看包。这种情况说明出现了其他类型的故障，如硬件问题。

```
Name  Mtu  Net/Dest      Address        Ipkts  Ierrs Opkts  Oerrs Collis Queue
lo0   8232 loopback      localhost      142    0     142    0     0      0
net0  1500 host58        host58        1106302 0     52419  0     0      0

Name  Mtu  Net/Dest      Address                   Ipkts  Ierrs Opkts  Oerrs Collis
```

```
lo0   8252 localhost     localhost                   142    0     142    0     0
net0  1500 fe80::a00:20ff:feb9:4c54/10 fe80::a00:20ff:feb9:4c54 1106305 0 52422 0  0
```

# 显示用户和进程信息

netstat 命令包含新的 -u 选项。此选项提供有关系统上使用特定端口的用户和进程的信息。使用 netstat -u 命令可以显示用户、进程 ID，以及创建了网络端点的程序或当前控制网络端点的程序。

要使 netstat 命令的输出更好地对齐，可以一起指定 -n 选项和 -u 选项。有关更多信息和详细示例，请参见 netstat(1M) 手册页。

# 显示已知路由的状态

结合使用 -r 选项和 netstat 命令可以显示本地主机的路由表。此表显示了主机已知的所有路由的状态。

```
% netstat -r
```

例 1-6 使用 netstat 命令显示路由表输出

以下示例显示 netstat -r 命令的输出。

```
Routing Table: IPv4
Destination           Gateway           Flags  Ref   Use   Interface
-------------------- -------------------- ----- ----- ------ ---------
host15               myhost               U         1  31059  net0
10.0.0.14            myhost               U         1      0  net0
default              distantrouter        UG        1      2  net0
localhost            localhost            UH        42019361  lo0

Routing Table: IPv6
Destination/Mask      Gateway                           Flags Ref  Use   If
--------------------- --------------------------          ----- --- ------ -----
2002:0a00:3010:2::/64 2002:0a00:3010:2:1b2b:3c4c:5e6e:abcd U     1   0      net0:1
fe80::/10             fe80::1a2b:3c4d:5e6f:12a2            U     1   23     net0
ff00::/8              fe80::1a2b:3c4d:5e6f:12a2            U     1   0      net0
default               fe80::1a2b:3c4d:5e6f:12a2            UG    1   0      net0
localhost             localhost                            UH    9   21832  lo0
```

下表介绍了 netstat -r 命令输出中显示的信息。

| 参数 | 说明 |
| --- | --- |
| Destination<br><br>Destination/Mask | 指定作为路由目标端点的主机。请注意，IPv6 路由表将 6to4 隧道端点 (2002:0a00:3010:2::/64) 的前缀显示为路由目标端点。 |

| 参数 | 说明 |
| --- | --- |
| Gateway | 指定用于转发包的网关。 |
| Flags | 指示路由的当前状态。U 标志指示路由处于运行状态。G 标志指示路由指向网关。 |
| Use | 显示已发送的包数。 |
| Interface | 指示作为传输源端点的本地主机上的特定接口。 |

## 使用 netstat 命令显示其他网络状态

可以结合使用 netstat 命令和各种其他命令行选项来显示本章中说明的网络状态之外的其他网络状态。此命令有 10 种形式，每种都会生成一个不同的表，分别提供有关网络子系统各部分的统计信息。

您可以获取的一些其他统计信息包括：

| | |
| --- | --- |
| netstat -g | 显示多播组成员资格 |
| netstat -P | 显示网络层到介质层的表：ARP 和 NDP 映射 |
| netstat -m | 显示 STREAMS 内存统计信息 |
| netstat -M | 显示多播路由表 |
| netstat -D | 显示 DHCP 租用状态 |
| netstat -d | 显示目标高速缓存表 |

有关完整的详细信息，请参见 netstat(1M) 手册页。

# 使用 netcat 实用程序执行 TCP 和 UDP 管理

使用 netcat (nc) 实用程序可以执行与 TCP 或 UDP 管理相关联的各种任务。可以对 IPv4 和 IPv6 网络使用此命令。

可以使用 netcat 实用程序来执行以下任务：

- 打开 TCP 连接
- 发送 UDP 包
- 监听任意的 TCP 和 UDP 端口
- 执行端口扫描

与 telnet 命令将每条错误消息分别发出到标准输出不同，nc 脚本生成的错误消息会合并为一个标准错误输出，这种方法更有效。

netcat 实用程序支持新的 -M 选项，这使您能够指定每个套接字的服务级别协议 (Service Level Agreement, SLA) 属性。与 -M 选项一起指定相应的属性时，将创建套接字的 MAC 流。例如，您可以按照以下方式使用 -M 选项：

```
% nc -M maxbw=50M host.example.com 7777
% nc -l -M priority=high,inherit=on 2222
```

如以上示例所示，-M 选项可用于指定 SLA 属性的 *name=value* 对的逗号分隔列表。

缺省情况下，一些安装方法并不会安装 netcat 软件包。按照以下方式检查您的系统上是否安装了该软件包：

```
% pkg list network/netcat
```

如果未安装该软件包，请按照以下方式安装该软件包：

```
% pfexec pkg install pkg:/network/netcat
```

有关更多详细信息，请参见 netcat(1) 手册页。

# 管理和记录网络状态显示

以下任务说明如何使用已知的网络命令来检查网络状态：

- 如何控制与 IP 相关的命令的显示输出 [27]
- “记录 IPv4 路由选择守护进程的操作” [28]
- 如何跟踪 IPv6 相邻节点搜索守护进程的活动 [29]

## ▼ 如何控制与 IP 相关的命令的显示输出

可以将 netstat 命令的输出控制为仅显示 IPv4 信息或同时显示 IPv4 和 IPv6 信息。

1. **创建 /etc/default/inet_type 文件。**
2. **根据您的网络需要，将以下某一项添加到 /etc/default/inet_type 文件：**

   - **要仅显示 IPv4 信息，请设置以下变量：**

     ```
     DEFAULT_IP=IP_VERSION4
     ```

   - **要同时显示 IPv4 和 IPv6 信息，请设置以下变量之一：**

     ```
     DEFAULT_IP=BOTH
     ```

```
DEFAULT_IP=IP_VERSION6
```

有关更多信息，请参见 inet_type(4) 手册页。

---

注 - netstat 命令的 -f 选项将覆盖 inet_type 文件中设置的值。

---

例 1-7 将输出控制为显示 IPv4 和 IPv6 信息

以下示例显示了在 inet_type 文件中指定 DEFAULT_IP=BOTH 或 DEFAULT_IP=IP_VERSION6 变量时的输出：

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
net0: flags=100001004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,PHYSRUNNING> mtu 1500 index
 603
        inet 10.46.86.54 netmask ffffff00 broadcast 10.46.8.255
        ether 0:1e:68:49:98:80
lo0: flags=2002000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv6,VIRTUAL> mtu 8252 index 1
        inet6 ::1/128
net0: flags=120002000841<UP,RUNNING,MULTICAST,IPv6,PHYSRUNNING> mtu 1500 index 603
        inet6 fe80::21e:68ff:fe49:9880/10
        ether 0:1e:68:49:98:80
net0:1: flags=120002080841<UP,RUNNING,MULTICAST,ADDRCONF,IPv6,PHYSRUNNING> mtu 1500 index 603
        inet6 2001:b400:414:6059:21e:68ff:fe49:9880/64
```

当您在 inet_type 文件中指定 DEFAULT_IP=IP_VERSION4 变量时，您应该会看到以下输出：

```
# ifconfig -a
lo0: flags=2001000849<UP,LOOPBACK,RUNNING,MULTICAST,IPv4,VIRTUAL> mtu 8232 index 1
        inet 127.0.0.1 netmask ff000000
net0: flags=100001004843<UP,BROADCAST,RUNNING,MULTICAST,DHCP,IPv4,PHYSRUNNING> mtu 1500 index
 603
        inet 10.46.86.54 netmask ffffff00 broadcast 10.46.8.255
        ether 0:1e:68:49:98:80
```

# 记录 IPv4 路由选择守护进程的操作

如果怀疑 IPv4 路由选择守护进程 routed 不能正常运行，则可以启动跟踪此守护进程活动的日志。此日志包括启动 routed 守护进程时的所有包传送。

按照以下方式创建跟踪路由选择守护进程操作的日志文件：

```
# /usr/sbin/in.routed /var/log-file-name
```

---

注意 - 在繁忙的网络中，此命令生成的输出几乎是连续的。

---

以下示例显示了执行 “记录 IPv4 路由选择守护进程的操作” [28] 过程时创建的日志的开始部分。

```
-- 2003/11/18 16:47:00.000000 --
Tracing actions started
RCVBUF=61440
Add interface lo0  #1   127.0.0.1      -->127.0.0.1/32
<UP|LOOPBACK|RUNNING|MULTICAST|IPv4> <PASSIVE>
Add interface net0 #2   10.10.48.112    -->10.10.48.0/25
<UP|BROADCAST|RUNNING|MULTICAST|IPv4>
turn on RIP
Add    10.0.0.0        -->10.10.48.112      metric=0  net0  <NET_SYN>
Add    10.10.48.85/25  -->10.10.48.112      metric=0  net0  <IF|NOPROP>
```

# ▼ 如何跟踪 IPv6 相邻节点搜索守护进程的活动

如果您怀疑 IPv6 `in.ndpd` 守护进程不能正常运行，则可以启动跟踪此守护进程的活动的日志。此跟踪显示在标准输出中，直到终止。此跟踪包括启动 `in.ndpd` 守护进程时的所有包传送。

1. 启动对 **`in.ndpd`** 守护进程的跟踪。

   ```
   # /usr/lib/inet/in.ndpd -t
   ```

2. 根据需要按 Ctrl-C 组合键终止跟踪。

例 1-8 跟踪 `in.ndpd` 守护进程

以下输出显示了对 `in.ndpd` 守护进程的跟踪的开始部分。

```
# /usr/lib/inet/in.ndpd -t
Nov 18 17:27:28 Sending solicitation to  ff02::2 (16 bytes) on net0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:b9:4c:54>
Nov 18 17:27:28 Received valid advert from fe80::a00:20ff:fee9:2d27 (88 bytes) on net0
Nov 18 17:27:28         Max hop limit: 0
Nov 18 17:27:28         Managed address configuration: Not set
Nov 18 17:27:28         Other configuration flag: Not set
Nov 18 17:27:28         Router lifetime: 1800
Nov 18 17:27:28         Reachable timer: 0
Nov 18 17:27:28         Reachable retrans timer: 0
Nov 18 17:27:28         Source LLA: len 6 <08:00:20:e9:2d:27>
Nov 18 17:27:28         Prefix: 2001:08db:3c4d:1::/64
Nov 18 17:27:28                 On link flag:Set
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
Nov 18 17:27:28         Prefix: 2002:0a00:3010:2::/64
Nov 18 17:27:28                 On link flag:Set
```

```
Nov 18 17:27:28                 Auto addrconf flag:Set
Nov 18 17:27:28                 Valid time: 2592000
Nov 18 17:27:28                 Preferred time: 604800
```

# 使用 ping 命令探测远程主机

可以使用 ping 命令确定您的系统是否可以与远程主机通信。运行 ping 命令时，ICMP 协议会将数据报发送到指定的主机，并请求响应。ICMP 是负责 TCP/IP 网络中错误处理的协议。当您使用 ping 命令时，可以确定能否在您的主机与指定的远程主机之间交换 IP 包。

以下是 ping 命令的基本语法：

**/usr/sbin/ping** *host* **[***timeout***]**

其中，*host* 是远程主机的名称。可选的 *timeout* 参数指示 ping 命令继续尝试到达远程主机所用的时间（以秒为单位）。缺省值为 20 秒。有关更多信息，请参见 ping(1M) 手册页。

## 为支持 IPv6 而对 ping 命令进行的修改

ping 命令既可以使用 IPv4 协议又可以使用 IPv6 协议来探测目标主机。具体选择哪个协议取决于由特定目标主机的名称服务器所返回的地址。缺省情况下，如果名称服务器返回目标主机的 IPv6 地址，ping 命令将使用 IPv6 协议。如果名称服务器仅返回 IPv4 地址，ping 命令将使用 IPv4 协议。可以使用 -A 选项指定要使用的协议以覆盖此行为。

有关详细信息，请参见ping(1M)手册页。

## 确定远程主机是否可访问

要确定远程主机是否可访问，请使用 ping 命令，如下所示：

% **ping** *hostname*

如果主机正在接受 ICMP 传输，则会显示以下消息：

*hostname* is alive

此消息指示主机对 ICMP 请求做出了响应。但是，如果主机出现故障，无法接收 ICMP 包，或者您的主机与远程主机之间存在路由问题，则会从 ping 命令接收到以下响应：

no answer from *hostname*

## 确定是否丢弃了您的主机与远程主机之间的包

包丢失会导致网络连接显得缓慢，因为需要花费额外的时间来重新传输丢弃的包。使用 ping 命令的 -s 选项可以确定是否丢弃了您的主机与远程主机之间的包：

```
% ping -s hostname
```

在以下示例中，ping -s *hostname* 命令连续不断地将包发送到指定的主机，直到您发送中断字符或出现超时为止：

```
% ping -s host1.domain8
PING host1.domain8 : 56 data bytes
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=0. time=1.67 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=1. time=1.02 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=2. time=0.986 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=3. time=0.921 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=4. time=1.16 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.00 ms
64 bytes from host1.domain8.COM (172.16.83.64): icmp_seq=5. time=1.980 ms

^C

----host1.domain8  PING Statistics----
7 packets transmitted, 7 packets received, 0% packet loss
round-trip (ms)  min/avg/max/stddev = 0.921/1.11/1.67/0.26
```

包丢失统计信息指示主机是否丢失了任何包。如果 ping 命令指示丢失了包，请使用 ipadm 和 netstat 命令检查网络的状态。有关更多详细信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“监视 IP 接口和地址”和“使用 netstat 命令监视网络状态” [22]。

# 使用 traceroute 命令显示路由信息

traceroute 命令将跟踪发往远程系统的 IP 包所经过的路由。可以使用 traceroute 命令查找所有的路由配置错误以及路由路径错误。如果无法到达特定的主机，则可以使用 traceroute 来查看发往远程主机的包所经由的路径以及可能出现故障的位置。

traceroute 命令还显示在通向目标主机的路径上每个网关的往返时间。此信息对于分析两个主机之间的网络通信在何处变慢非常有用。

有关 traceroute 命令的更多详细信息，请参见 traceroute(1M) 手册页。

本节包含以下主题：

## 为支持 IPv6 而对 traceroute 命令进行的修改

可以使用 traceroute 命令跟踪到特定主机的 IPv4 和 IPv6 路由。从协议的角度看，traceroute 命令与 ping 命令使用相同的算法。使用 -A 命令行选项可覆盖此行为。另外，还可以使用 -a 命令行选项跟踪到多宿主主机的每个地址的各个单独路由。

## 搜索通向远程主机的路由

按以下方式使用 traceroute 命令可搜索通向远程系统的路由：

```
% traceroute destination-hostname
```

以下 traceroute 命令输出显示了包从本地系统 nearhost 到达远程系统 farhost 所经由的具有七个跃点的路径。此输出还显示包遍历每个跃点所用的时间。

```
% traceroute farhost
traceroute to farhost (172.16.64.39), 30 hops max, 40 byte packets
1  frbldg7c-86 (172.16.86.1)  1.516 ms  1.283 ms  1.362 ms
2  bldg1a-001 (172.16.1.211)  2.277 ms  1.773 ms  2.186 ms
3  bldg4-bldg1 (172.16.4.42)  1.978 ms  1.986 ms  13.996 ms
4  bldg6-bldg4 (172.16.4.49)  2.655 ms  3.042 ms  2.344 ms
5  ferbldg11a-001 (172.16.1.236)  2.636 ms  3.432 ms  3.830 ms
6  frbldg12b-153 (172.16.153.72)  3.452 ms  3.146 ms  2.962 ms
7  farhost (172.16.64.39)  3.430 ms  3.312 ms  3.451 ms
```

## 跟踪所有路由

在本地系统上使用带 -a 选项的 traceroute 命令可以跟踪所有路由：

```
% traceroute -a host-name
```

以下示例显示了通向双栈主机的所有可能路由：

```
% traceroute -a v6host
traceroute: Warning: Multiple interfaces found; using 2001:db8:4a3a:1:56:a0:a8 @ net0:2
 traceroute to v6host (2001:db8:4a3b:5:102:a00:fe79:19b0),30 hops max, 60 byte packets
1 v6-rout86 (2001:db8:4a3b:1:56:a00:fe1f:59a1) 35.534 ms 56.998 ms *
2 2001:db8::255:0:c0a8:717 32.659 ms 39.444 ms *
3  farhost (2001:db8:4a3b:2:103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
4  distant (2001:db8:4a3b:3:100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
5  v6host (2001:db8:4a3b:5:102:a00:fe79:19b0)  66.111 ms *  36.965 ms *

traceroute to v6host  (192.168.10.75),30 hops max,40 byte packets
1  v6-rout86 (172.16.86.1)  4.360 ms  3.452 ms  3.479 ms
2  flrmpj17u (172.16.17.131)  4.062 ms  3.848 ms  3.505 ms
3  farhost (10.0.0.23)  4.773 ms *  4.294 ms
4  distant (192.168.10.104)  5.128 ms  5.362 ms *
5  v6host  (192.168.15.85)  7.298 ms  5.444 ms *
```

# 使用 TShark 和 Wireshark 分析器分析网络通信

*TShark* 是命令行网络通信分析器，利用此分析器可以从实时网络捕获包数据或者从以前保存的捕获文件读取包，方法是将已解码形式的包打印到标准输出或者将这些包写入到文件中。不带任何选项的情况下，TShark 的工作方式与 `tcpdump` 命令类似，并且还使用相同的实时捕获文件格式 `libpcap`。此外，TShark 还可以检测、读取和写入与 Wireshark 支持的文件相同的捕获文件。

*Wireshark* 是第三方图形用户界面 (graphical user interface, GUI) 网络协议分析器，用于以交互方式转储和分析网络通信。与 `snoop` 命令类似，您可以使用 Wireshark 在实时网络上或以前保存的捕获文件中浏览包数据。缺省情况下，Wireshark 将 `libpcap` 格式用于文件捕获，该格式也由 `tcpdump` 实用程序和其他类似工具使用。使用 Wireshark 的关键优势在于，它能够读取和导入 `libpcap` 格式之外的其他几种文件格式。

TShark 和 Wireshark 都提供了几项独特的功能，其中包括：

- 能够将一个 TCP 会话中的所有包组合起来，并以 ASCII、EBCDIC 或 hex 格式显示该会话中的数据
- 包含比其他网络协议分析器更多的可筛选字段
- 使用比其他网络协议分析器更丰富的语法创建筛选器

要在 Oracle Solaris 系统上使用 TShark 和 Wireshark，请先检查是否安装了这些软件包，如有必要，按以下方式安装它们：

```
# pkg install tshark

# pkg install wireshark
```

有关更多信息，请参见 `tshark(1)` 和 `wireshark(1)` 手册页。

另请参见位于 http://www.wireshark.org/ 的 Wireshark 文档。

# 使用 snoop 命令监视包传送

可以使用 `snoop` 命令监视网络通信。`snoop` 命令可捕获网络包并以指定的格式显示内容。一旦收到包或将其保存到文件之后，便可以立即显示这些包。当 `snoop` 命令向中间文件执行写入时，在紧密跟踪的情况下不可能丢失包。然后，就可以使用 `snoop` 命令来解释该文件。

要以混杂模式捕获进出缺省接口的包，您必须具有“网络管理”权限配置文件或 `root` 角色。在汇总表单中，`snoop` 命令仅显示与最高层协议有关的数据。例如，NFS 包仅显示 NFS 信息，而不会显示底层远程过程调用 (remote procedure call, RPC)、UDP、IP 和以太网帧信息，但是如果使用了两个详细选项之一，则也会显示这些信息。

持续不断地使用 `snoop` 命令可使您熟悉常规的系统行为。有关 `snoop` 命令的更多详细信息，请参见 `snoop(1M)` 手册页。

本节包含以下主题：



## 为支持 IPv6 而对 snoop 命令进行的修改

snoop 命令可以捕获 IPv4 和 IPv6 包。此命令可以显示 IPv6 数据包头、IPv6 扩展头、ICMPv6 数据包头和相邻节点搜索 (Neighbor Discovery, ND) 协议数据。缺省情况下，snoop 命令既可以显示 IPv4 包又可以显示 IPv6 包。如果您指定了 ip 或 ip6 协议关键字，此命令将只显示 IPv4 包或 IPv6 包。使用 IPv6 的过滤选项，可以对所有的 IPv4 和 IPv6 包进行过滤，以便仅显示 IPv6 包。请参见“监视 IPv6 网络通信” [36]和 snoop(1M) 手册页。

## ▼ 如何检查来自所有接口的包

1. 列显有关连接到系统的接口的信息。

   ```
   # ipadm show-if
   ```

   snoop 命令通常使用第一个非回送设备，通常为主网络接口。

2. 通过键入不带参数的 **snoop** 开始捕获包。

   ```
   # snoop
   ```

3. 按 Ctrl-C 组合键停止此进程。

例 1-9 显示基本 snoop 输出

以下示例显示双栈主机的基本 snoop 命令输出。

```
# snoop
Using device /dev/net (promiscuous mode)
router5.local.com -> router5.local.com ARP R 10.0.0.13, router5.local.com is
0:10:7b:31:37:80
router5.local.com -> BROADCAST     TFTP Read "network-confg" (octet)
myhost -> DNSserver.local.com      DNS C 192.168.10.10.in-addr.arpa. Internet PTR ?
DNSserver.local.com  myhost        DNS R 192.168.10.10.in-addr.arpa. Internet PTR
```

```
niserve2.
.
.
.
fe80::a00:20ff:febb:e09 -> ff02::9 RIPng R (5 destinations)
```

在以上输出中，捕获到的包显示 DNS 查询和响应，以及来自本地路由器的定期地址解析协议 (Address Resolution Protocol, ARP) 包和向 `in.ripngd` 守护进程发出的 IPv6 链路本地地址的通告。

## ▼ 如何检查 IPMP 组中的包

在 Oracle Solaris 10 中，如果要捕获 IPMP 组中的包，您需要手动对 IPMP 组的每个接口运行 `snoop` 命令。在 Oracle Solaris 11 发行版中，您可以使用带 `-I` 选项的 `snoop` 命令从 IPMP 组中的所有接口捕获包。然后，该输出将会合并到单个输出流。

`snoop` 命令通常使用第一个非回送设备，通常为主网络接口。然而，当使用了 `-I` 选项时，`snoop` 命令将使用 IP 接口（来自 `/dev/ipnet/*`），如 `ipadm show-if` 命令所示。您还可以使用此选项来窥探回送通信和其他纯 IP 的结构。`-d` 选项使用在 `dladm show-link` 命令输出中显示的数据链路接口。

1. 列显有关连接到系统的接口的信息。

   ```
   # ipadm show-if
   ```

2. 开始捕获指定 IPMP 组的包。

   ```
   # snoop -I ipmp-group
   ```

3. 按 Ctrl-C 组合键停止此进程。

## ▼ 如何将 snoop 输出捕获到文件

1. 将 **snoop** 会话捕获到文件。例如：

   ```
   # snoop -o /tmp/cap
   Using device /dev/eri (promiscuous mode)
   30 snoop: 30 packets captured
   ```

   在以上示例中，在名为 `/tmp/cap` 的文件中捕获到了 30 个包。可以将此文件放在任何具有足够磁盘空间的目录中。捕获的包数显示在命令行中，您可以随时按 Ctrl-C 组合键中止捕获。

   `snoop` 命令将在主机上产生大量网络负载，这会使结果失真。要查看实际结果，请从第三方系统运行 `snoop` 命令。

2. **检查 snoop 输出捕获文件。**

```
# snoop -i filename
```

例 1-10 显示 snoop 输出捕获

以下输出显示了可能会作为 snoop -i 命令输出接收到的捕获。

```
# snoop -i /tmp/cap
1   0.00000 fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375
ICMPv6 Neighbor advertisement
...
10  0.91493    10.0.0.40 -> (broadcast)  ARP C Who is 10.0.0.40, 10.0.0.40 ?
34  0.43690 nearserver.here.com  -> 224.0.1.1  IP  D=224.0.1.1 S=10.0.0.40 LEN=28,
ID=47453, TO =0x0, TTL=1
35  0.00034  10.0.0.40 -> 224.0.1.1    IP  D=224.0.1.1 S=10.0.0.40 LEN=28, ID=57376,
TOS=0x0, TTL=47
```

## ▼ 如何检查 IPv4 服务器和客户机之间的包

1. **在远离与客户机或服务器相连的集线器（或交换机上的镜像端口）的位置建立 snoop 系统。**
   第三方系统（snoop 系统）将检查所有干预通信，以便 snoop 跟踪会反映网络上实际出现的情况。

2. **键入带有相应选项的 snoop 命令并将输出保存到文件。**

3. **检查并解释输出。**
   有关 snoop 捕获文件的详细信息，请参见 RFC 1761, Snoop Version 2 Packet Capture File Format (http://www.rfc-editor.org/rfc/rfc1761.txt)（RFC 1761，Snoop 版本 2 包捕获文件格式）。

# 监视 IPv6 网络通信

可以按照以下方法使用 snoop 命令捕获 IPv6 包：

```
# snoop ip6
```

以下示例显示了在节点上运行 snoop ip6 命令时可能显示的典型输出。

```
# snoop ip6
fe80::a00:20ff:fecd:4374 -> ff02::1:ffe9:2d27 ICMPv6 Neighbor solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
solicitation
fe80::a00:20ff:fee9:2d27 -> fe80::a00:20ff:fecd:4375 ICMPv6 Neighbor
solicitation
```

```
fe80::a00:20ff:febb:e09 -> ff02::9      RIPng R (11 destinations)
fe80::a00:20ff:fee9:2d27 -> ff02::1:ffcd:4375 ICMPv6 Neighbor solicitation
```

有关 snoop 命令的更多信息，请参见 snoop(1M) 手册页。

# 使用 IP 层设备监视包

Oracle Solaris 中引入了用于增强 IP 观察功能的 IP 层设备。通过这些设备，可以访问具有与系统网络接口关联的地址的所有包。这些地址包括本地地址以及位于非回送接口或逻辑接口上的地址。可观察的通信流量可以是 IPv4 和 IPv6 地址。因此，可以监视以系统为目标的所有通信流量。通信流量可以是回送 IP 通信流量、来自远程计算机的包、要从系统发送的包或转发的所有通信流量。

使用 IP 层设备，Oracle Solaris 全局区域的管理员可以监视区域之间以及区域内的通信流量。非全局区域的管理员也可以观察由该区域发送和接收的通信流量。

要监视 IP 层上的通信流量，请使用带有较新的 -I 选项的 snoop 命令。此选项指定此命令将使用新的 IP 层设备，而不是底层链路层设备来显示通信流量数据。

## ▼ 如何检查 IP 层上的包

1. （可选） 如有必要，列显有关连接到系统的接口的信息。

   ```
   # ipadm show-if
   ```

2. 捕获特定接口上的 IP 通信流量。

   ```
   # snoop -I interface [-V | -v]
   ```

### 用于检查包的方法

以下所有示例均基于此系统配置：

```
# ipadm show-addr

ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           dhcp     ok           10.153.123.225/24
lo0/v6            static   ok           ::1/128
net0/v6           addrconf ok           fe80::214:4fff:2731:b1a9/10
net0/v6           addrconf ok           2001:0db8:212:60bb:214:4fff:2731:b1a9/64
net0/v6           addrconf ok           2001:0db8:56::214:4fff:2731:b1a9/64
```

假设有两个区域 sandbox 和 toybox，它们使用以下 IP 地址：

- sandbox — 172.0.0.3

- toybox — 172.0.0.1

您可以对系统上的不同接口使用 snoop -I 命令。显示的包信息取决于您是全局区域的管理员还是非全局区域的管理员。

**例 1-11** 观察回送接口上的通信流量

以下示例显示了用于回送接口的 snoop 命令的输出。

```
# snoop -I lo0
Using device ipnet/lo0 (promiscuous mode)
localhost -> localhost    ICMP Echo request (ID: 5550 Sequence number: 0)
localhost -> localhost    ICMP Echo reply (ID: 5550 Sequence number: 0)
```

要生成详细输出，请使用 -v 选项。

```
# snoop -v -I lo0
Using device ipnet/lo0 (promiscuous mode)
IPNET:  ----- IPNET Header -----
IPNET:
IPNET:  Packet 1 arrived at 10:40:33.68506
IPNET:  Packet size = 108 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 4
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 0
IPNET:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
...
```

为了支持观察 IP 层上的包，引入了新的 ipnet 数据包头，置于要观察的包之前。将同时指示源和目标 ID。ID 为 0 指示将从全局区域生成通信流量。

**例 1-12** 观察本地区域内的 net0 设备中的包流

以下示例显示了系统内不同区域中发生的通信。您可以查看与 net0 IP 地址关联的所有包，包括那些从本地传递到其他区域的包。如果生成详细输出，则还可以查看包流中涉及的区域。

```
# snoop -I net0
Using device ipnet/net0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=62117 Syn Seq=195630514 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=62117 S=22 Syn Ack=195630515 Seq=195794440 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=62117 Ack=195794441 Seq=195630515 Len=0 Win=49152
sandbox -> toybox TCP D=62117 S=22 Push Ack=195630515 Seq=195794441 Len=20 Win=491

# snoop -I net0 -v port 22
IPNET:  ----- IPNET Header -----
IPNET:
```

```
IPNET:  Packet 5 arrived at 15:16:50.85262
IPNET:  Packet size = 64 bytes
IPNET:  dli_version = 1
IPNET:  dli_type = 0
IPNET:  dli_srczone = 0
IPNET:  dli_dstzone = 1
IPNET:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 40 bytes
IP:   Identification = 22629
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 6 (TCP)
IP:   Header checksum = 0000
IP:   Source address = 172.0.0.1, 172.0.0.1
IP:   Destination address = 172.0.0.3, 172.0.0.3
IP:   No options
IP:
TCP:  ----- TCP Header -----
TCP:
TCP:  Source port = 46919
TCP:  Destination port = 22
TCP:  Sequence number = 3295338550
TCP:  Acknowledgement number = 3295417957
TCP:  Data offset = 20 bytes
TCP:  Flags = 0x10
TCP:        0... .... = No ECN congestion window reduced
TCP:        .0.. .... = No ECN echo
TCP:        ..0. .... = No urgent pointer
TCP:        ...1 .... = Acknowledgement
TCP         .... 0... = No push
TCP         .... .0.. = No reset
TCP:        .... ..0. = No Syn
TCP:        .... ...0 = No Fin
TCP:  Window = 49152
TCP:  Checksum = 0x0014
TCP:  Urgent pointer = 0
TCP:  No options
TCP:
```

在以上输出中，`ipnet` 数据包头指示该包是从全局区域 (ID `0`) 发送至 `sandbox` (ID `1`) 的。

**例 1-13** 通过标识区域观察网络通信流量

以下示例演示如何通过标识区域来观察网络通信流量，这种方法对于具有多个区域的系统非常有用。当前，只能使用区域 ID 标识区域。不支持将 snoop 命令与区域名称一起使用。

```
# snoop -I hme0 sandboxsnoop -I net0 sandbox
Using device ipnet/hme0 (promiscuous mode)
toybox -> sandbox TCP D=22 S=61658 Syn Seq=374055417 Len=0 Win=49152 Options=<mss
sandbox -> toybox TCP D=61658 S=22 Syn Ack=374055418 Seq=374124525 Len=0 Win=49152
toybox -> sandbox TCP D=22 S=61658 Ack=374124526 Seq=374055418 Len=0 Win=49152
```

# 使用 ipstat 和 tcpstat 命令观察网络通信流量

在此发行版中引入了以下两个新命令，用于观察服务器上各种类型的网络通信流量：ipstat 和 tcpstat。

ipstat 命令用于根据命令语法中指定的所选输出模式和排序顺序收集并报告有关服务器上 IP 通信流量的统计信息。使用此命令可以观察 IP 层的网络通信流量（按源、目标、高层协议和接口进行聚合）。当您希望观察一个服务器与其他服务器之间的通信量时，使用此命令。

tcpstat 命令用于根据命令语法中指定的所选输出模式和排序顺序收集并报告有关服务器上 TCP 和 UDP 通信流量的统计信息。使用此命令可以观察传输层的网络通信流量（专门用于 TCP 和 UDP）。除源和目标的 IP 地址以外，您还可以观察源和目标的 TCP 或 UDP 端口、发送或接收通信流量的进程的 PID，以及运行该进程的区域的名称。

可通过下面一些方法使用 tcpstat 命令：

- 标识服务器上 TCP 和 UDP 通信流量的最大源。
- 检查特定进程生成的通信流量。
- 检查从特定区域生成的通信流量。
- 确定哪个进程绑定到本地端口。

---

注 - 以上列表并非全部。还可以通过其他几种方法使用 tcpstat 命令。有关更多信息，请参见 tcpstat(1M) 手册页。

---

要使用 ipstat 和 tcpstat 命令，需要下面某种权限：

- 承担 root 角色
- 被显式指定 dtrace_kernel 权限
- 被指定 "Network Management"（网络管理）或"Network Observability"（网络观察）权限配置文件

以下示例演示了可使用这两个命令观察网络通信流量的各种方法。有关详细信息，请参见 tcpstat(1M) 和 ipstat(1M) 手册页。

以下示例显示使用 -c 选项运行时 ipstat 命令的输出。使用 -c 选项可以在以前的报告之后打印更新的报告，而不覆盖以前的报告。此示例中的数字 3 指定数据的显示时间间隔，与以 ipstat 3 形式调用此命令一样。

```
# ipstat -c 3
SOURCE                    DEST                      PROTO    INT        BYTES
zucchini                  antares                   TCP      net0       72.0
zucchini                  antares                   SCTP     net0       64.0
antares                   zucchini                  SCTP     net0       56.0
amadeus.foo.example.com   10.6.54.255               UDP      net0       40.0
antares                   zucchini                  TCP      net0       40.0
zucchini                  antares                   UDP      net0       16.0
antares                   zucchini                  UDP      net0       16.0
Total: bytes in: 192.0  bytes out: 112.0
```

作为对比，以下示例显示与 -c 选项一起使用时 tcpstat 命令的输出：

```
# tcpstat -c 3
ZONE         PID PROTO  SADDR             SPORT DADDR             DPORT   BYTES
global    100680 UDP    antares           62763 agamemnon          1023   76.0
global    100680 UDP    antares             775 agamemnon          1023   38.0
global    100680 UDP    antares             776 agamemnon          1023   37.0
global    100680 UDP    agamemnon          1023 antares           62763   26.0
global    104289 UDP    zucchini          48655 antares            6767   16.0
global    104289 UDP    clytemnestra      51823 antares            6767   16.0
global    104289 UDP    antares            6767 zucchini         48655   16.0
global    104289 UDP    antares            6767 clytemnestra     51823   16.0
global    100680 UDP    agamemnon          1023 antares             776   13.0
global    100680 UDP    agamemnon          1023 antares             775   13.0
global    104288 TCP    zucchini          33547 antares            6868    8.0
global    104288 TCP    clytemnestra      49601 antares            6868    8.0
global    104288 TCP    antares            6868 zucchini         33547    8.0
global    104288 TCP    antares            6868 clytemnestra     49601    8.0
Total: bytes in: 101.0  bytes out: 200.0
```

下面的其他一些示例演示了可使用 ipstat 和 tcpstat 命令观察网络上的通信流量的其他方法。

**例 1-14** 使用 ipstat 命令观察最活跃的五个 IP 通信流

以下示例报告最活跃的五个 IP 通信流。-l *nlines* 选项指定每个报表要输出的数据行数。

```
# ipstat -l 5
SOURCE                    DEST                      PROTO    IFNAME     BYTES
charybdis.foo.example.com achilles.exampl           UDP      net0        6.6K
eratosthenes.example.com  aeneas.example.c          TCP      tun0        6.1K
achilles.exampl           charybdis.foo.example.com UDP      net0      964.0
aeneas.example.c          eratosthenes.example.com  TCP      tun0      563.0
odysseus.example.         255.255.255.255           UDP      net0       66.0
```

```
Total: bytes in: 12.6K bytes out:  2.2K
```

**例 1-15** 使用 ipstat 命令显示时间戳

以下示例报告首要的 IP 通信流量并显示标准日期格式 (-d d) 的时间戳。可以指定以秒为单位或以 Unix 时间 (-d u) 形式打印时间戳。时间间隔设置为 10 秒。

```
# ipstat -d d -c 10
Monday, March 26, 2012 08:34:07 PM EDT
SOURCE                     DEST                       PROTO    IFNAME     BYTES
charybdis.foo.example.com  achilles.exampl            UDP      net0       15.1K
eratosthenes.example.com   aeneas.example.c           TCP      tun0       13.9K
achilles.exampl            charybdis.foo.example.com  UDP      net0        2.4K
aeneas.example.c           eratosthenes.example.com   TCP      tun0        1.5K
odysseus.example.          255.255.255.255            UDP      net0       66.0
cassiopeia.foo.example.com aeneas.example.c           TCP      tun0       29.0
aeneas.example.c           cassiopeia.foo.example.com TCP      tun0       20.0
Total: bytes in: 29.1K bytes out:  3.8K
```

**例 1-16** 使用 tcpstat 命令观察最活跃的五个通信流

以下示例报告服务器的最活跃的五个 TCP 通信流：

```
# tcpstat -l 5
ZONE            PID PROTO  SADDR            SPORT DADDR            DPORT   BYTES
global        28919 TCP    achilles.exampl  65398 aristotle.exampl   443   33.0
zone1          6940 TCP    ajax.example.com  6868 achilles.exampl  61318    8.0
zone1          6940 TCP    achilles.exampl  61318 ajax.example.com  6868    8.0
global         8350 TCP    ajax.example.com  6868 achilles.exampl  61318    8.0
global         8350 TCP    achilles.exampl  61318 ajax.example.com  6868    8.0
Total: bytes in: 16.0  bytes out: 49.0
```

**例 1-17** 使用 tcpstat 命令显示时间戳信息

在以下示例中，tcpstat 命令用于以标准日期格式显示服务器上 TCP 网络通信流量的时间戳信息：

```
# tcpstat -d d -c 10
Saturday, March 31, 2012 07:48:05 AM EDT
ZONE            PID PROTO  SADDR            SPORT DADDR            DPORT   BYTES
global         2372 TCP    penelope.example 58094 polyphemus.examp    80   37.0
zone1          6940 TCP    ajax.example.com  6868 achilles.exampl  61318    8.0
zone1          6940 TCP    achilles.exampl  61318 ajax.example.com  6868    8.0
global         8350 TCP    ajax.example.com  6868 achilles.exampl  61318    8.0
global         8350 TCP    achilles.exampl  61318 ajax.example.com  6868    8.0
Total: bytes in: 16.0  bytes out: 53.0
```

♦ ♦ ♦ 第 2 章

# 2 关于 IPMP 管理

IP 网络多路径 (IP network multipathing, IPMP) 是第 3 层 (L3) 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 3 章 管理 IPMP 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。

## IPMP 的新增功能

以下是本发行版中新增或已更改的功能。

### IPMP 配置更改

本发行版中的 IPMP 和 Oracle Solaris 10 中的 IPMP 的工作原理不同。一个显著的变化是，现在 IP 接口分组到虚拟 IP 接口（例如 `ipmp0`）中。虚拟 IP 接口提供所有的数据 IP 地址，而用于基于探测器的故障检测的测试地址将分配给底层接口（例如 `net0`）。有关更多信息，请参见 “IPMP 的工作原理” [48]。

从现有 IPMP 配置转换到新的 IPMP 模型时请参阅以下常规工作流程：

1. 在配置 IPMP 之前，请确保在系统上启用了 `DefaultFixed` 配置文件。请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“启用和禁用配置文件”。

2. 确保基于 SPARC 的系统上 MAC 地址是唯一的。请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“如何确保每个接口的 MAC 地址是唯一的”。
3. 使用 `dladm` 命令配置数据链路。要在您的 IPMP 配置中使用相同的物理网络设备，将需要首先确定与每个设备实例关联的数据链路。

```
# dladm show-phys
LINK              MEDIA               STATE      SPEED  DUPLEX    DEVICE
net1              Ethernet            unknown    0      unknown   bge1
net0              Ethernet            up         1000   full      bge0
net2              Ethernet            unknown    1000   full      e1000g0
net3              Ethernet            unknown    1000   full      e1000g1
```

4. 如果您以前将 `e1000g0` 和 `e1000g1` 用于 IPMP 配置，则现在将使用 `net2` 和 `net3`。请注意，数据链路不但可以基于物理链路，而且可以基于聚合、VLAN、VNIC 等。有关更多信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“显示系统的数据链路”。
5. 使用 `ipadm` 命令可执行以下任务：
   - 配置网络层。
   - 创建 IP 接口。
   - 将 IP 接口添加到 IPMP 组中。
   - 将数据地址添加到 IPMP 组中。

有关本发行版中的网络管理命令更改的更多信息，请参见《从 Oracle Solaris 10 转换至 Oracle Solaris 11.2 》中的“网络管理命令更改”。

# Oracle Solaris 中的 IPMP 支持

IPMP 提供了以下支持：

- 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。
- 主要使用 `ipmpstat` 命令来获取有关 IPMP 组的信息。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` 命令的功能、您可使用的选项以及每个选项生成的输出都在“监视 IPMP 信息” [75]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“配置 IPMP 组” [59]。

# 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 还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。

请注意以下有关 IPMP 为传入和传出负荷分配强制实施的策略的重要信息：

- 对于链路上 IP 地址，IPMP 随机选择一个活动 IP 接口来到达该 IP 地址。在给定链路上 IP 地址具有多个不同连接的情况下，这些所有连接将使用同一个传出 IP 接口。此外，如果 IP 接口随时间发生更改，则这种更改将影响到该 IP 地址的所有连接。
- 对于链路下 IP 地址，IPMP 随机选择一个 IP 接口来到达链路上 IP 路由器的 IP 地址，通过该路由器访问链路下 IP 地址。此策略实际上意味着给定 IPMP 组的所有链路下 IP 地址使用单个 IP 接口。

---

注 - 当前的传入和传出负荷分配策略可能会发生更改。

---

链路聚合执行与 IPMP 类似的功能以提高网络性能和可用性。要比较这两项技术，请参见《在 Oracle Solaris 11.2 中管理网络数据链路 》中的附录 A “链路聚合和 IPMP：功能比较”。

# 用于使用 IPMP 的规则

IPMP 组配置取决于您的特定系统配置。

遵守以下 IPMP 配置规则：

1. 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。另请注意，此规则有一些例外适用，例如，适用于 Oracle 提供的某些工程系统。有关进一步说明，请与您的 Oracle 支持代表联系。

2. 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 “IPMP 接口配置的类型” [47]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。

3. 同一组中的所有接口必须按相同顺序配置相同的 STREAMS 模块。当规划某个 IPMP 组时，请首先检查目标 IPMP 组中所有接口上的 STREAMS 模块顺序，然后按标准顺序为该 IPMP 组推送每个接口的模块。要打印 STREAMS 模块的列表，请使用 `ifconfig`*interface*`modlist` 命令。例如，以下是 `net0` 接口的 `ifconfig` 输出：

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

   如以上输出所示，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。

   以下示例显示了按标准顺序为该 IPMP 组推送每个接口的模块时可能使用的命令：

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

   有关规划 IPMP 组的逐步说明，请参见如何规划 IPMP 组 [59]。

# 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“IPMP 中的故障检测” [54]。

  `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，请参见 “IPMP 的工作原理” [48]。有关该守护进程的更多信息，请参见 `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

  有关如何修改配置文件的信息，请参见如何配置 IPMP 守护进程的行为 [74]。
- **`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“监视 IPMP 信息” [75]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 IPMP 组所需，为其配置任意数量的备用接口。

也可以将单个接口配置到其自己的 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。

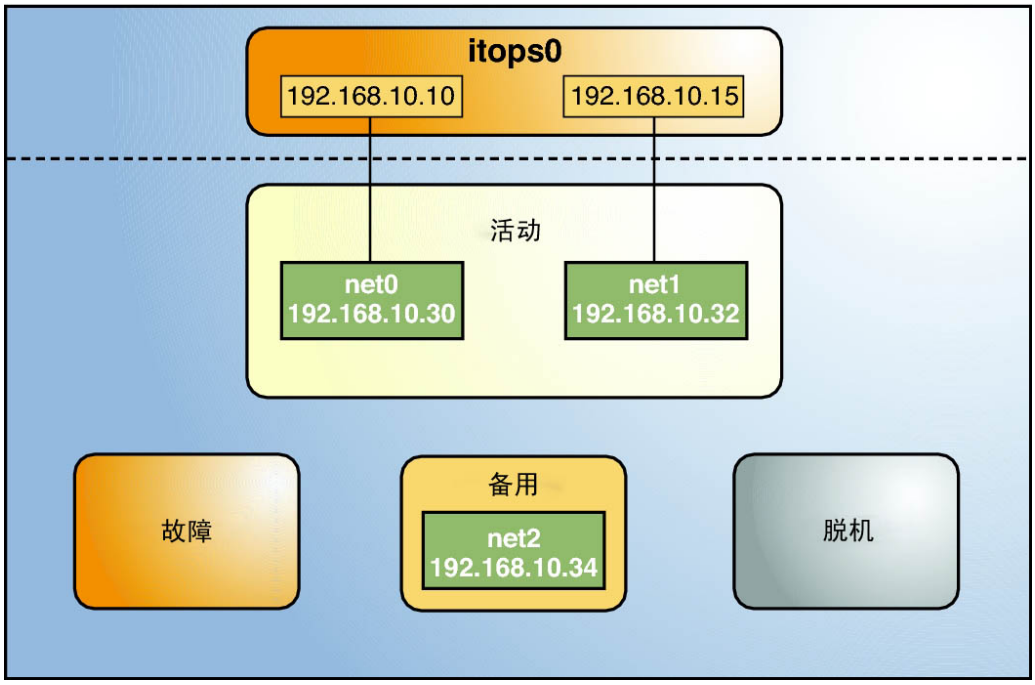
# IPMP 的工作原理

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 2-1 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 2-1 “IPMP 活动/备用配置”、图 2-2 “IPMP 中的接口故障”、图 2-3 “IPMP 中的备用接口故障” 和图 2-4 “IPMP 恢复过程” 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。如需更多示例，请参见“监视 IPMP 信息” [75]。

以下命令显示有关图 2-1 “IPMP 活动/备用配置” 中的 IPMP 配置的信息：

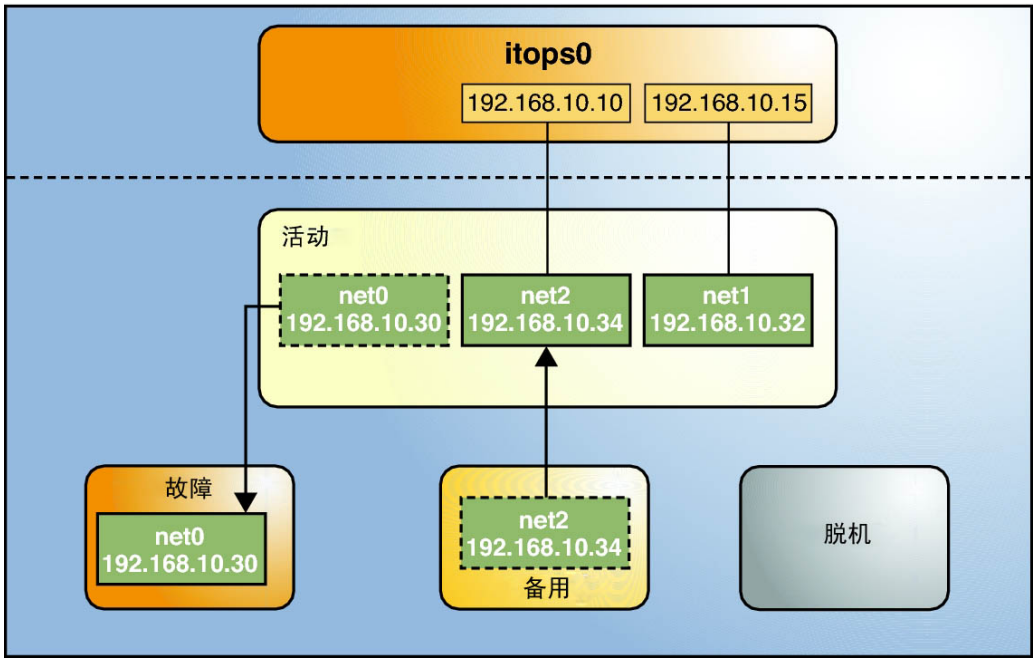
```
# ipmpstat -g
GROUP     GROUPNAME     STATE     FDT       INTERFACES
itops0    itops0        ok        10.00s    net1 net0 (net2)
```

显示有关该组的底层接口的信息，如下所示：

```
# 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。下图中显示了 net2 激活。

图 2-2 IPMP 中的接口故障



注 - 图 2-2 “IPMP 中的接口故障” 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。

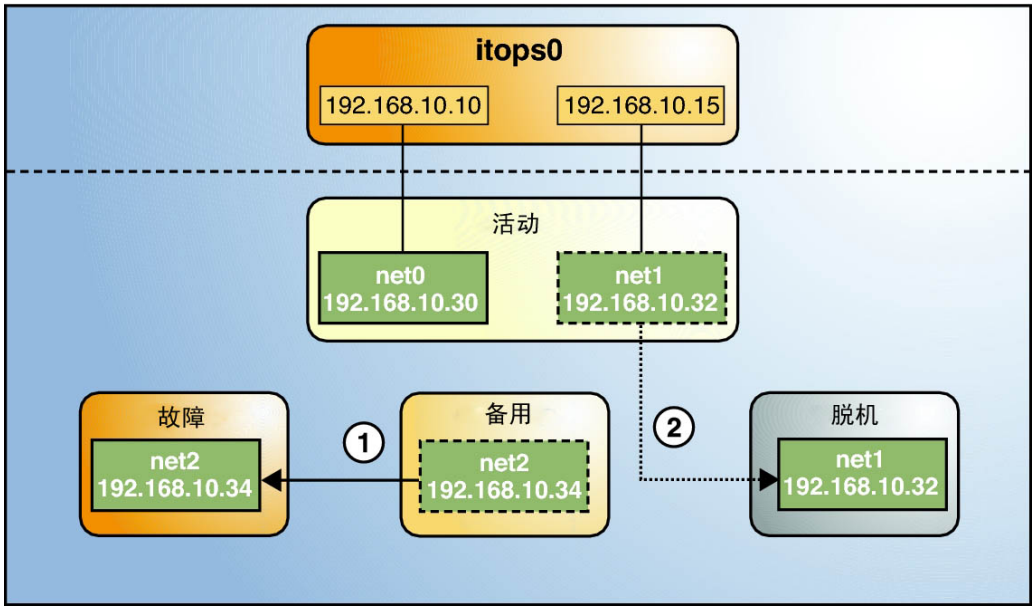
ipmpstat 命令显示了此图中的信息，如下所示：

```
# 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 2-3 “IPMP 中的备用接口故障” 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，IPMP 组就只剩下一个正常工作的接口 net0。

图 2-3 IPMP 中的备用接口故障



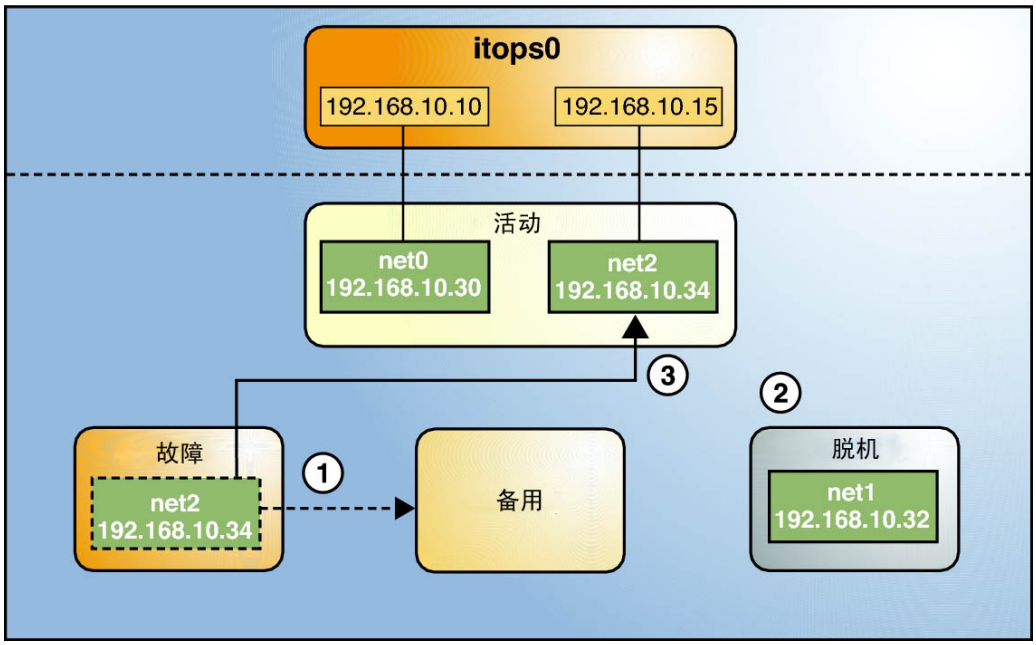

ipmpstat 命令显示了此图中的信息，如下所示：

```
# 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，接口修复后的恢复过程有所不同。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。下图显示了该恢复过程。

图 2-4 IPMP 恢复过程




在图 2-4 “IPMP 恢复过程” 中，net2 在得到修复后，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 2-2 “IPMP 中的接口故障” 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“FAILBACK=no 模式” [57]。

# 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。使用 IPMP 配置的接口支持以下两种类型的地址：数据地址和测试地址。IP 地址仅驻留在 IPMP 接口（组）上且被指定为数据地址。测试地址是驻留在底层接口的 IP 地址。

## 数据地址

数据地址是 DHCP 服务器在引导时动态指定给 IP 接口的或使用 ipadm 命令手动指定的传统 IPv4 和 IPv6 地址。仅将数据地址指定给 IPMP 接口（或组）。标准的 IPv4 包通信和 IPv6 包通信（如果适用）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，请参见“基于探测器的故障检测” [54]。

---

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在同一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.rfc-editor.org/rfc/rfc1918.txt)。

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：

- 两种类型的基于探测器的故障检测：
  - 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 驱动程序支持该故障检测）

## 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` 守护进程认为

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，请参见如何配置 IPMP 守护进程的行为 [74]。

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，请参见“配置基于探测器的故障检测” [72]。

## 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和一个数据地址的 IPMP 组为例。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。有关说明，请参见“选择故障检测方法” [73]。

---

### 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，请参见如何配置 IPMP 守护进程的行为 [74]。

# 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，恢复过程如何继续取决于故障接口最初的配置方式，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 “FAILBACK=no 模式” [57]。

---

- 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。

有关 IPMP 在接口故障及修复过程中行为的图形表示，请参见 “IPMP 的工作原理” [48]。

### 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，可能也是不可接受的。在这种情况下，您可能更希望允许激活的备用接口继续作为活动接口。IPMP 允许您覆盖缺省行为，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，请参见如何配置 IPMP 守护进程的行为 [74]。

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 - 会为整个 IPMP 组设置 FAILBACK=NO 模式，而不是作为每接口可调参数。

---

## 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（例如接口）。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，则仅会恢复持久性配置。这种情况下，将记录以下错误消息：

```
"IP: persistent configuration is restored for <ifname>"
```

否则，分离请求成功完成。您可以从系统中移除该组件，并且任何现有连接都未中断。

---

注 - 更换 NIC 时，确保更换卡为同一类型（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 3 章

# 3 管理 IPMP

本章介绍如何在 Oracle Solaris 发行版中使用 IPMP 管理接口组。本章中的各项任务介绍如何使用 `ipadm` 命令配置 IPMP，该命令替代了 Oracle Solaris 10 中用于配置 IPMP 的 `ifconfig` 命令。要了解有关这两条命令如何相互映射的更多信息，请参见《从 Oracle Solaris 10 转换至 Oracle Solaris 11.2 》中的“将 ifconfig 命令与 ipadm 命令进行比较”。

有关 IPMP 概念模型中的更改的详细说明，请参见 “IPMP 的新增功能” [43]。

本章包含以下主题：



## 配置 IPMP 组

以下信息介绍如何规划和配置 IPMP 组。第 2 章 关于 IPMP 管理中的概述介绍了如何将 IPMP 组实现为接口。在本章中，术语 *IPMP* 组和 *IPMP* 接口可互换使用。

本节包含以下任务：



### ▼ 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。您不需要按顺序执行这些任务。

IPMP 配置取决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荷分配而提高性能。不过，如果有更多的底层接口，有助于在接口出现故障时保证网络的可用性。

---

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“用于使用 IPMP 的规则” [45]。

---

1. **确定满足您的需要的常规 IPMP 配置。**
   有关确定要使用的 IPMP 配置的指导，请参见此过程的任务摘要中的信息。

2. **（仅限 SPARC） 验证组中的每个接口是否具有唯一的 MAC 地址。**
   要为系统上的每个接口配置唯一的 MAC 地址，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》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。**
   有关准则和要使用的命令语法，请参见“用于使用 IPMP 的规则” [45]。

4. **在 IPMP 组中的所有接口上使用同一 IP 寻址格式。**
   如果为 IPv4 配置了一个接口，则必须为 IPv4 配置 IPMP 组中的所有接口。同样，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。请参见 “IPMP 中的故障检测” [54]。

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。请参见 “IPMP 接口配置的类型” [47]。

---

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 RFC 1577 (http://www.rfc-editor.org/rfc/rfc1577.txt) 和 RFC 2225 (http://www.rfc-editor.org/rfc/rfc2225.txt) 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。请参见 “IPMP 接口配置的类型” [47]。以下过程介绍了如何使用 DHCP 配置活动/备用 IPMP 组。

开始之前 在执行以下过程之前，请执行以下操作：

- 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。有关过程的信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》。即使尚未创建底层 IP 接口，您也可以创建一个 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，必须每个接口配置一个唯一的 MAC 地址。请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》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，请参见《在 Oracle Solaris 11.2 中使用 DHCP 》。

1. 成为 **root** 角色。
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. 为 IPMP 组添加将包含测试地址的底层 IP 接口。

   **#** **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. 指定 DHCP 配置和管理 IPMP 接口上的数据地址。

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

上一步将 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 11.2 中配置和管理网络组件 》中的“如何配置 IPv4 接口”。

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

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

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

7. （可选） 针对 IPMP 组的每个底层接口重复执行步骤 6。

例 3-1 使用 DHCP 配置 IPMP 组

以下示例显示了使用 DHCP 的活动/备用 IPMP 组配置，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。

首先创建 IPMP 接口。

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

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
```

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
```

然后，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，将 net2 接口配置为备用接口。

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

## ▼ 如何配置活动/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。有关说明，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“如何配置 IPv4 接口”。即使底层 IP 接口尚不存在，您也可以创建一个 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。请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“如何确保每个接口的 MAC 地址是唯一的”。

1. **成为 root 角色。**

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。

## ▼ 如何配置活动/备用 IPMP 组

以下过程介绍如何配置一个 IPMP 组，其中将一个接口保留为备用接口。仅当组中的一个活动接口出现故障时，才部署此接口。

有关备用接口的概述信息，请参见 “IPMP 接口配置的类型” [47]。

1. 成为 **root** 角色。

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。

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 -m ip under-interface
```

例 3-2 配置活动/备用 IPMP 组

以下示例说明如何创建活动/备用 IPMP 配置。

首先创建 IPMP 接口。

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

然后创建底层 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，将 IP 地址指定给 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。

然后将 IP 地址指定给 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，将 `net2` 接口配置为备用接口。

```
# 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
```

# 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，为此您要使用系统上的一个接口，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 - 以下任务使用主接口作为定义缺省路由的示例。然而，这种类型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 时保留缺省路由

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） 显示路由表中当前定义的路由。**

   ```
   # 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） 显示路由表中重新定义的路由。

```
# netstat -nr
```

6. （可选） 如果该信息未更改，请重新启动路由服务，然后重新检查路由表中的信息，以确保已正确地重新定义路由。

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

例 3-3 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
```

# 管理 IPMP

本节包含以下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 确保添加到组的接口满足所有的所需要求。有关要求列表，请参见如何规划 IPMP 组 [59]。

1. **成为 root 角色。**
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。

例 3-4 将接口添加到 IPMP 组

以下示例说明如何将 net4 接口添加到 IPMP 组 ipmp0。

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

## ▼ 如何从 IPMP 组中删除接口

1. **成为 root 角色。**
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。

例 3-5 从 IPMP 组中删除接口

以下示例说明如何从 IPMP 组 ipmp0 中删除 net4 接口。

```
# ipadm remove-ipmp net4 ipmp0
```

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

## ▼ 如何将 IP 地址添加到 IPMP 组

要将 IP 地址添加到 IPMP 组，请使用 `ipadm create-addr` 命令。对于 IPMP 配置，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，而测试地址则添加到 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 `root` 角色。**

2. **将 IP 地址添加到 IPMP 组。**

   - **将数据地址添加到 IPMP 组中，如下所示：**

     ```
     # 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。

   - **将测试地址添加到 IPMP 组的底层接口，如下所示：**

     ```
     # 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。

## ▼ 如何从 IPMP 接口中删除 IP 地址

要从 IPMP 组中删除 IP 地址，请使用 `ipadm delete-addr` 命令。对于 IPMP 配置，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 `root` 角色。**

2. **确定要删除的 IP 地址。**

   - **按照以下方式显示数据地址的列表：**

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

- 按照以下方式显示测试地址的列表：

  ```
  # 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. 从 IPMP 组中删除 IP 地址。

   - 按照以下方式删除数据地址：

     ```
     # 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。

   - 按照以下方式删除测试地址：

     ```
     # 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。

例 3-6 从接口删除测试地址

以下示例使用例 3-2 “配置活动/备用 IPMP 组” 中显示的活动/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 **root** 角色。

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。

例 3-7 将接口移动到其他 IPMP 组

在以下示例中，IPMP 组的底层接口是 `net0`、`net1` 和 `net2`。该示例说明如何从 IPMP 组 `ipmp0` 中删除 `net0` 接口，然后将 `net0` 放在 IPMP 组 `cs-link1` 中。

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

## ▼ 如何删除 IPMP 组

如果您不再需要特定 IPMP 组，请使用以下过程。

1. 成为 **root** 角色。
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 地址也将从系统中删除。

例 3-8 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

本节包含以下主题：



## 关于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“基于探测器的故障检测” [54]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。

`route` 命令既作用于 IPv4 路由又作用于 IPv6 路由，IPv4 路由是缺省设置。如果紧跟 `route` 命令之后使用 `-inet6` 选项，系统将针对 IPv6 路由执行操作。

如何为基于探测器的故障检测手动指定目标系统 [73]过程显示了用于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。有关可与 `route` 命令一起使用的选项的更多信息，请参见 `route`(1M) 手册页和“部署 IPMP 时维护路由” [66]。

## 为基于探测器的故障检测选择目标的要求

请参考以下要求来确定网络上的哪些主机可能用作好的目标：

- 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。

## 选择故障检测方法

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。

在选择基于探测器的检测方法之前，请确保您的探测器目标满足“为基于探测器的故障检测选择目标的要求” [72]中列出的要求。

要仅使用传递式探测，请执行以下操作：

1. 使用 SMF 命令启用 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。
2. 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。

要使用测试地址来探测故障，请执行以下操作：

1. 如有必要，请使用 SMF 命令禁用传递式探测。

   ```
   # svccfg -s svc:/network/ipmp setprop config/transitive-probing=false
   # svcadm refresh  svc:/network/ipmp:default
   ```
2. 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

以下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“为基于探测器的故障检测选择目标的要求” [72]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为 `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

使用 IPMP `/etc/default/mpathd` 配置文件为 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 **root** 角色。
2. 编辑 **/etc/default/mpathd** 文件。

   ```
   # pfedit /etc/default/mpathd
   ```

   有关说明，请参见 `pfedit(1M)` 手册页。

   更改以下三个参数中的一个或多个参数的缺省值：

   - 为 `FAILURE_DETECTION_TIME` 参数键入新值，如下所示：

     ```
     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。
   - 为 `FAILBACK` 参数键入新值，如下所示：

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

yes — 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“检测物理接口修复” [56]中所述。

no — 表示数据通信不返回到修复后的接口。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。

- 为 TRACK_INTERFACES_ONLY_WITH_GROUPS 参数键入新值，如下所示：

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

yes — 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 — 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。

有关此参数和匿名组功能的更多信息，请参见“故障检测和匿名组功能” [56]。

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

以下示例说明如何使用 ipmpstat 命令监视系统上 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。您还可以使用相同的命令来获取有关故障检测的信息。有关更多信息，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（ipmpstat 与 -p 选项结合使用的情况除外）。

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。

---

可以将 ipmpstat 命令与以下选项结合使用来显示所需的信息：

-g　显示有关系统上 IPMP 组的信息。请参见例 3-9 “获取 IPMP 组信息”。

-a　显示为 IPMP 组配置的数据地址。请参见例 3-10 “获取 IPMP 数据地址信息”。

-i　显示与 IPMP 配置相关的 IP 接口的信息。请参见例 3-11 “获取有关 IPMP 组的底层 IP 接口的信息”。

-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。请参见例 3-12 “获取 IPMP 探测器目标信息”。

-p　显示有关用于故障检测的探测器的信息。请参见例 3-13 “观察 IPMP 探测器”。

以下其他示例说明如何使用 ipmpstat 命令来显示有关系统中 IPMP 配置的信息。

**例 3-9　获取 IPMP 组信息**

-g 选项显示系统上各 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。

```
% 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，首先显示活动接口，然后是非活动接口，最后显示不可用的接口。接口的状态由显示方式表示：

- *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。

例 3-10　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。显示的信息还包括那些可供使用的地址，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        --          --
192.168.10.31   up       field2        net2    net2 net7
192.168.10.32   up       field2        net7    net2 net7
192.168.10.33   down     field2        --          --
```

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 IP 接口。

**例 3-11**　　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：

- b – 表示接口由系统指定用于接收 IPMP 组的广播通信。
- d – 表示接口已关闭，因此不可用。
- h – 表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。
- i – 表示为接口设置了 INACTIVE 标志。因此，该接口不用于发送或接收数据通信。
- m – 表示接口由系统指定用于发送和接收 IPMP 组的 IPv4 多播通信。
- M – 表示接口由系统指定用于发送和接收 IPMP 组的 IPv6 多播通信。
- s – 表示接口配置为备用接口。

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 – 分别表示链路可用或不可用。
- unknown – 表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 – 表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 – 表示找不到合适的探测器目标，因此无法发送探测器。
- disabled – 表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 – 表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 – 表示接口当前未工作，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 – 表示接口不可用。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 – 表示无法确定 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。

例 3-12　　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。以下示例中的输出显示为基于探测器的故障检测使用测试地址的 IPMP 配置。

```
% ipmpstat -nt
INTERFACE   MODE          TESTADDR        TARGETS
net0        routes        192.168.85.30   192.168.85.1 192.168.85.3
net1        disabled      --              --
net3        disabled      --              --
net4        routes        192.1.2.200      192.1.2.1
net2        multicast     192.168.10.200   192.168.10.1 192.168.10.2
net6        multicast     192.168.10.201   192.168.10.2 192.168.10.1
net5        multicast     192.168.10.202   192.168.10.1 192.168.10.2
net7        multicast     192.168.10.203   192.168.10.1 192.168.10.2
```

以下输出显示使用传递式探测（即，没有测试地址的基于探测器的故障检测）的 IPMP 配置。

```
% 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 – 表示使用系统路由表查找探测器目标。
- mcast – 表示使用多播 ICMP 探测器查找目标。
- disabled – 表示已为接口禁用基于探测器的故障检测。
- transitive – 表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“基于探测器的故障检测” [54]。

TESTADDR　　指定主机名；或者如果 -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。

例 3-13　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 Ctrl-C 组合键终止命令。您必须成为 root 角色或具有适当的权限才能运行此命令。

下面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     192.168.10.1
0.26s   net6        602       --        --        --         192.168.10.2
0.25s   net5        601       0.62ms    1.20ms    1.00ms     192.168.10.1
0.26s   net7        603       0.79ms    1.11ms    1.10ms     192.168.10.1
1.66s   net4        613       --        --        --         192.1.2.1
1.70s   net0        603       0.63ms    1.10ms    1.10ms     192.168.85.3
^C
```

下面是使用传递式探测（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。

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

使用 -o 选项可以定制 ipmpstat 命令的输出。将此选项与之前提到的其他 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
- 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 all 参数中包含 -o 选项。

## 在脚本中使用 ipmpstat 命令

当您通过脚本或通过使用命令别名运行 ipmpstat 命令时，尤其是在您还希望生成计算机可解析的输出时，-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。
- 当请求了多个字段时，如果某个字段包含文字冒号 (:) 或反斜杠 (\)，您可以通过为这些字符添加反斜杠 (\) 前缀来对其进行转义或排除它们。

为了正确使用 ipmpstat-P 命令，请遵循以下规则：

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

注意 - 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 选项的正确语法：

```
% 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，您将 -o 和 -g 选项与 -P 选项结合使用。

-P 选项设计用于脚本中。以下示例说明如何通过脚本运行 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 4 章

# 关于 IP 隧道管理

本章概述了 Oracle Solaris 中的 IP 隧道管理。有关与任务相关的信息，请参见第 5 章 管理 IP 隧道。

本章包含以下主题：



## IP 隧道管理中的新增功能

IP 隧道管理已经过修订，以使其与用于 Oracle Solaris 11 中的网络数据链路管理的新模型一致。在此发行版中，可以使用 `dladm` 命令创建和配置 IP 隧道。隧道也可以使用此发行版中支持的其他数据链路功能。例如，通过支持以管理方式选择的名称，可以为隧道指定更有意义的名称。有关更多信息，请参见 `dladm(1M)` 手册页。

## IP 隧道功能摘要

当中间网络不支持域中的协议时，IP 隧道（在本书中也简称为隧道）提供了一种在这些域之间传输数据包的方式。例如，在大多数网络使用 IPv4 协议的环境中，IPv6 网络需要一种方式来与其边界外部通信。可以使用隧道进行这种通信。IP 隧道在可通过使用 IP 访问的两个节点之间提供虚拟链路。因此，可使用该链路在 IPv4 网络上传输 IPv6 包，从而在两个 IPv6 站点之间实现 IPv6 通信。

### 隧道类型

隧道连接涉及将 IP 包封装到其他包中。通过此封装操作，包可以通过不支持包协议的中间网络到达其目标。根据使用的包封装的类型，隧道会有所不同。

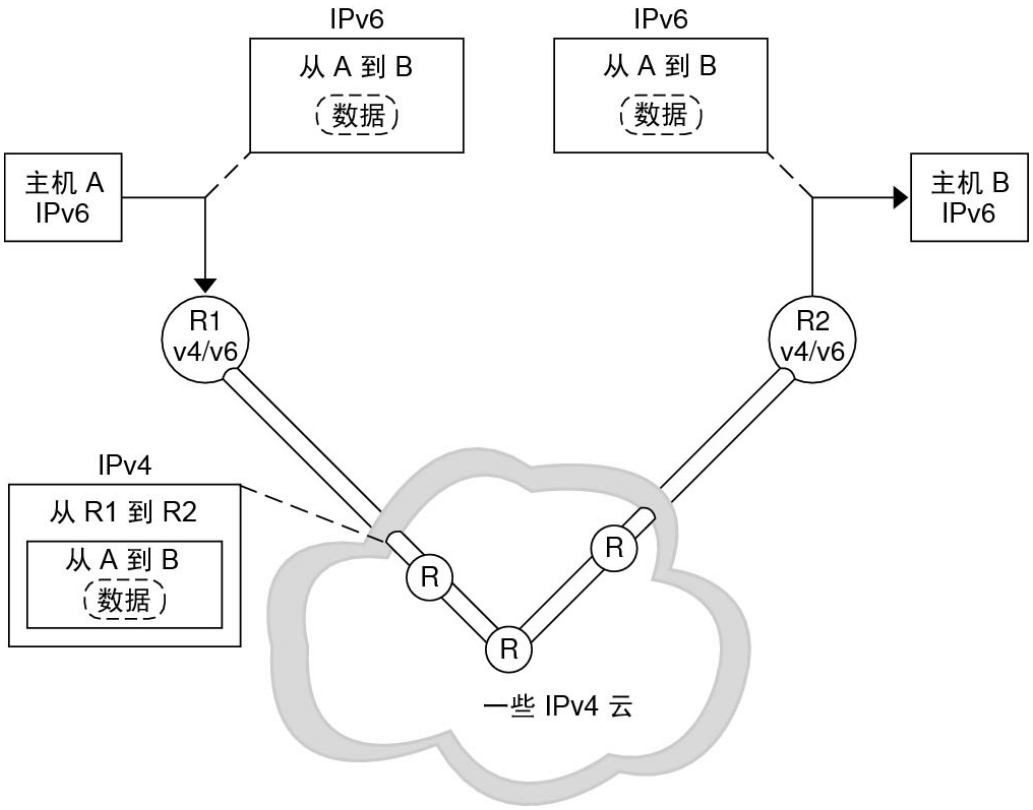
Oracle Solaris 支持以下类型的隧道：

- **IPv4 隧道** – IPv4 包将封装于 IPv4 数据包头中并发送到预配置的单播 IPv4 目标。为了更具体地说明流经隧道的包，IPv4 隧道也称为 *IPv4 over IPv4* 隧道或 *IPv6 over IPv4* 隧道。
- **6to4 隧道** – IPv6 包将封装于 IPv4 数据包头中并发送到自动按包确定的 IPv4 目标。此确定的依据基于 *6to4* 协议中定义的算法。
- **IPv6 隧道** – IPv6 包将封装于 IPv4 数据包头中并发送到自动按包确定的 IPv4 目标。确定的依据基于 6to4 协议中定义的算法。

## IPv6 和 IPv4 的组合网络环境中的隧道

在 IPv6 部署的早期阶段，具有 IPv6 域的许多站点可能都需要通过遍历 IPv4 网络与其他 IPv6 域通信。下图演示了通过 IPv4 路由器的两个 IPv6 主机之间的隧道连接机制（在此图中用 "R" 指示）。

图 4-1 IPv6 隧道连接机制




在上图中，隧道由两个路由器组成，这两个路由器之间配置了通过 IPv4 网络建立的虚拟的点对点链路。

IPv6 包封装于 IPv4 包内。IPv6 网络的边界路由器可以设置经由各种 IPv4 网络到达目标 IPv6 网络的边界路由器的点对点隧道。包通过隧道传输到目标边界路由器，将在该路由器中对包取消封装，然后，路由器将单独的 IPv6 包转发到目标节点。

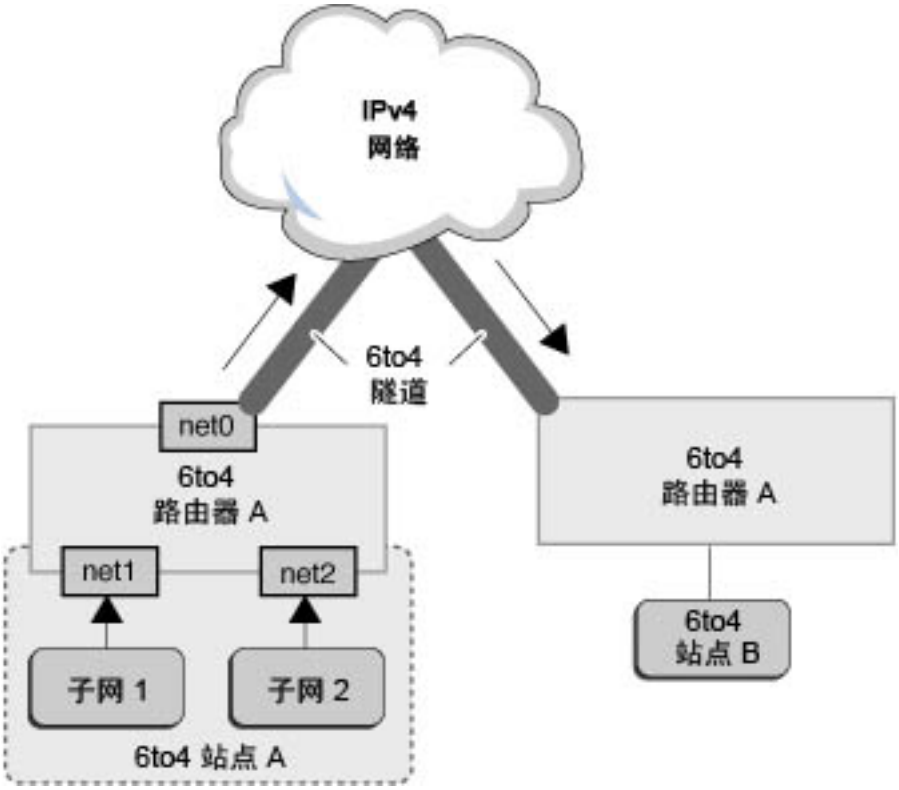
## 6to4 隧道

Oracle Solaris 包括 6to4 隧道作为中间方法，以实现从 IPv4 到 IPv6 寻址的转换。通过 *6to4* 隧道，隔离的 IPv6 站点可以通过不支持 IPv6 的 IPv4 网络跨自动隧道进行通信。要使用 6to4 隧道，必须首先将 IPv6 网络上的边界路由器配置为 6to4 自动隧道的一个端点。这样，6to4 路由器便可以参与通往另一个 6to4 站点的隧道，如果需要的话，还可以参与通往本地非 6to4 IPv6 站点的隧道。

## 6to4 隧道的拓扑

6to4 隧道提供到任何位置的所有 6to4 站点的 IPv6 连接。如果 6to4 隧道配置为向中继路由器转发，该隧道也同样提供到所有 IPv6 站点的连接（包括本地 IPv6 Internet）。下图显示了 6to4 隧道如何提供两个 6to4 站点之间的连接。

图 4-2 两个 6to4 站点之间的隧道




上图描述了两个隔离的 6to4 网络，即站点 A 和站点 B。每个站点都配置了具有到 IPv4 网络的外部连接的路由器。跨 IPv4 网络的 6to4 隧道可以连接两个 6to4 站点。

必须至少配置一个路由器接口来支持 6to4，才能让 IPv6 站点成为 6to4 站点。此接口必须提供与 IPv4 网络的外部连接。在上图中，边界路由器 A 的 `net0` 接口将站点 A 连接到 IPv4 网络。在 `net0` 上配置的地址必须全局唯一。必须为 `net0` 接口配置一个 IPv4 地址，然后才能配置隧道接口以在路由器上支持 6to4。

在上图中，6to4 站点 A 由两个子网组成，这些子网连接到路由器 A 上的 `net1` 和 `net2` 接口。站点 A 的任一子网上的所有 IPv6 主机会在接收到来自路由器 A 的通告时自动重新配置为具有 6to4 派生的地址。

站点 B 是另一个隔离的 6to4 站点。为了正确地从站点 A 接收通信，必须在站点 B 上配置边界路由器以支持 6to4。否则，路由器从站点 A 接收的包将因无法识别而被丢弃。

### `6to4relay` 命令

使用 6to4 隧道连接，可以在相互隔离的 6to4 站点之间进行通信。但是，要使用本地的非 6to4 IPv6 站点传输包，6to4 路由器必须使用 6to4 中继路由器建立一个隧道。然后，*6to4* 中继路由器将 6to4 包转发到 IPv6 网络，并最终将其传输到本地 IPv6 站点。如果启用了 6to4 的站点必须与本地 IPv6 站点交换数据，请使用 `6to4relay` 命令启用相应的隧道。

---

注 - 由于使用中继路由器不安全，因此 Oracle Solaris 在缺省情况下会禁用与中继路由器的隧道连接。在部署该方案之前，请认真考虑在建立通往 6to4 中继路由器的隧道时所涉及的问题。有关详细信息，请参见“启用 6to4 中继路由器隧道的注意事项” [90]。如果决定启用 6to4 中继路由器支持，可以参阅如何创建和配置 IP 隧道 [96]中的相关操作步骤。

---

有关更多信息，请参见 `6to4`(7M) 手册页。

## 通过 6to4 隧道的包流

本节介绍从一个 6to4 站点上的主机到远程 6to4 站点上的主机之间的包流。此方案使用图 4-2 “两个 6to4 站点之间的隧道” 中显示的拓扑。而且，它还假定已经配置了 6to4 路由器和 6to4 主机。

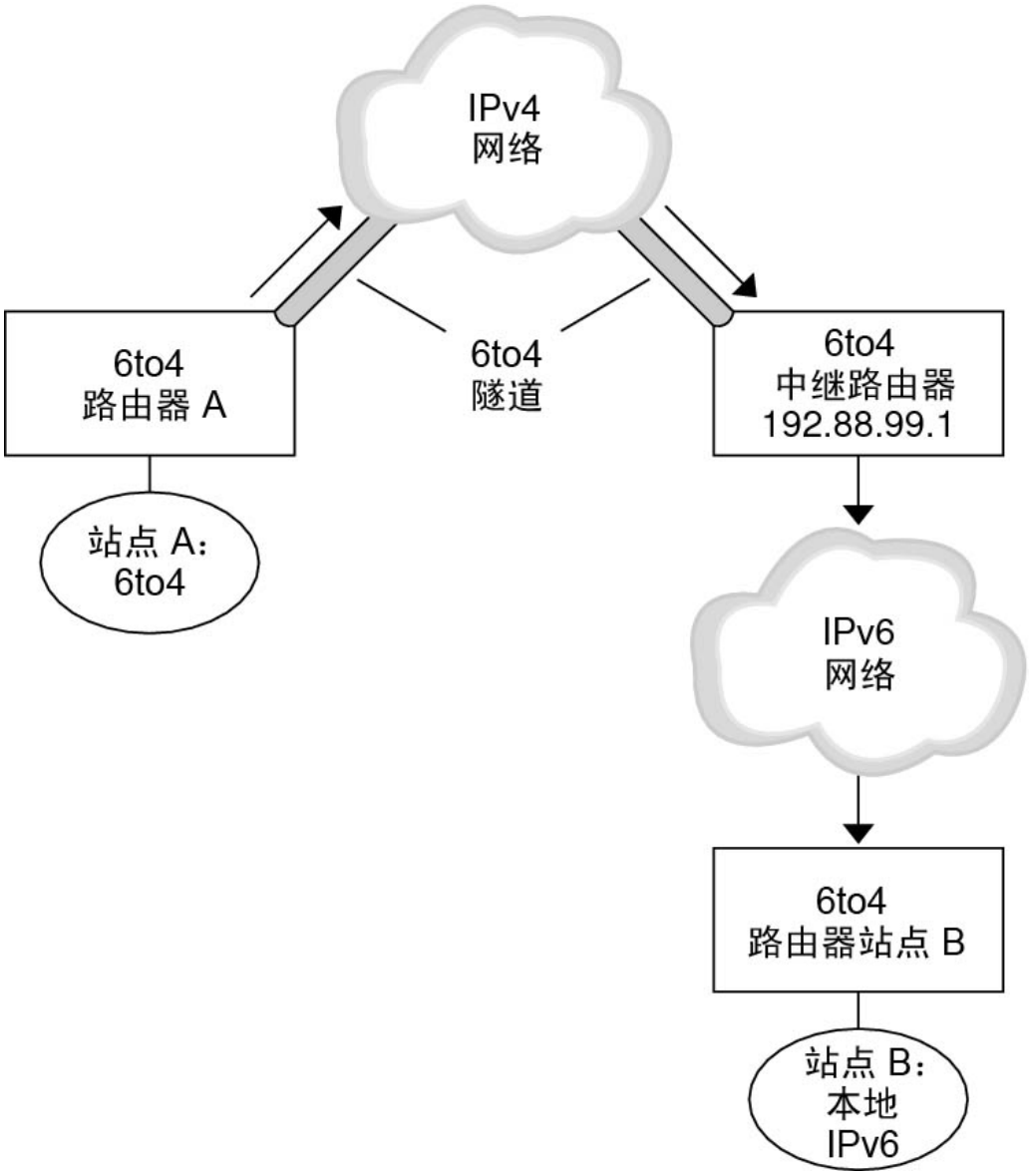
以下是包流：

1. 6to4 站点 A 的子网 1 上的主机发送传输请求，6to4 站点 B 上的主机作为目标。每个数据包头中均有 6to4 派生的源地址和 6to4 派生的目标地址。
2. 站点 A 的路由器将每个 6to4 包封装到 IPv4 数据包头中。在该过程中，路由器将 IPv4 封装数据包头的目标地址设置为站点 B 的路由器地址。对于每个流经隧道接口的 IPv6 包，其 IPv6 目标地址同时也包含 IPv4 目标地址。因此，路由器将能够确定 IPv4 封装数据包头上设置的 IPv4 目标地址。路由器随后使用标准的 IPv4 路由过程，通过 IPv4 网络转发包。
3. 包通过的任何 IPv4 路由器都使用包的 IPv4 目标地址进行转发。此地址是路由器 B 上某个接口的全局唯一 IPv4 地址，该接口还充当 6to4 伪接口。
4. 来自站点 A 的包到达路由器 B，路由器 B 对 IPv4 数据包头中的 IPv6 包取消封装。
5. 路由器 B 随后使用 IPv6 包中的目标地址将包转发到站点 B 上的接收主机。

## 启用 6to4 中继路由器隧道的注意事项

6to4 中继路由器充当某些隧道的一个端点，这些隧道的另一个端点是需要与本地非 6to4 IPv6 网络通信的 6to4 路由器。本质上，中继路由器是 6to4 站点和本地 IPv6 站点之间的桥梁。因为此解决方案可能不安全，所以，在缺省情况下，Oracle Solaris 不启用对 6to4 中继路由器的支持。但是，如果站点需要这样的隧道，可以使用 `6to4relay` 命令来启用隧道连接，如下图所示。

图 4-3 从 6to4 站点到 6to4 中继路由器的隧道




在图 4-3 “从 6to4 站点到 6to4 中继路由器的隧道” 中，6to4 站点 A 需要与本地 IPv6 站点 B 上的节点通信。该图说明了从站点 A 到 IPv4 网络上 6to4 隧道的通信路径。该隧道

将 6to4 路由器 A 和 6to4 中继路由器作为其端点。IPv6 站点 B 所连接到的 IPv6 网络位于 6to4 中继路由器的外部。

### 6to4 站点和本地 IPv6 站点之间的包流

本节介绍从 6to4 站点到本地 IPv6 站点之间的包流。此方案使用图 4-3 “从 6to4 站点到 6to4 中继路由器的隧道” 中显示的拓扑。

以下是包流：

1. 6to4 站点 A 上的主机发送一个将本地 IPv6 站点 B 上的主机指定为目标的传输信号。每个包头中具有作为其源地址的 6to4 派生地址。目标地址是标准的 IPv6 地址。
2. 站点 A 的 6to4 路由器将每个包封装到 IPv4 数据包头中，该 IPv4 数据包头将 6to4 中继路由器的 IPv4 地址作为其目标地址。6to4 路由器随后使用标准的 IPv4 路由过程，通过 IPv4 网络转发包。包遇到的任何 IPv4 路由器都会将包转发到 6to4 中继路由器。
3. 物理位置距离站点 A 最近的任播 6to4 中继路由器检索以 `192.88.99.1` 任播组为目标的包。

---

注 - 6to4 中继路由器属于 6to4 中继路由器任播组，它的 IP 地址为 `192.88.99.1`。此任播地址是 6to4 中继路由器的缺省地址。如果您需要使用特定的 6to4 中继路由器，则可以覆盖缺省设置并指定该路由器的 IPv4 地址。

---

4. 该中继路由器会对 6to4 包中的 IPv4 数据包头取消封装，并显示本地 IPv6 目标地址。
5. 然后，中继路由器仅将 IPv6 包发送到 IPv6 网络中，站点 B 中的路由器最终将在该网络中检索到这些包。然后，路由器将这些包转发到目标 IPv6 节点。

# 关于部署 IP 隧道

要正确部署 IP 隧道，需要执行两个主要任务。首先创建隧道链路，然后配置隧道上的 IP 接口。下面是创建隧道及其对应 IP 接口的要求。

## 创建 IP 隧道的要求

要成功创建 IP 隧道，必须遵守以下要求：

- 如果使用主机名而不是字面值 IP 地址，则这些名称必须解析为与隧道类型兼容的有效 IP 地址。

- 所创建的 IPv4 或 IPv6 隧道不得与配置的其他隧道具有相同的隧道源地址和隧道目标地址。
- 所创建的 IPv4 或 IPv6 隧道不得与现有的 6to4 隧道具有相同的隧道源地址。
- 如果创建 6to4 隧道，则该隧道不得与配置的其他隧道具有相同的隧道源地址。

有关更多信息，请参见《在 Oracle Solaris 11.2 中规划网络部署 》中的“网络中使用隧道的规划”。

# IP 隧道和 IP 接口的要求

每个隧道类型都对隧道上配置的 IP 接口具有特定的 IP 地址要求。下表中汇总了这些要求。

表 4-1 隧道和 IP 接口要求

| 隧道类型 | 隧道上允许的 IP 接口 | IP 接口要求 |
|---|---|---|
| IPv4 隧道 | IPv4 接口 | 手动指定本地和远程地址。 |
| | IPv6 接口 | 发出 `ipadm create-addr -T addrconf` 命令时，将自动设置本地和远程本地链路地址。请参见 ipadm(1M) 手册页。 |
| IPv6 隧道 | IPv4 接口 | 手动指定本地和远程地址。 |
| | IPv6 接口 | 发出 `ipadm create-addr -T addrconf` 命令时，将自动设置本地和远程本地链路地址。请参见 ipadm(1M) 手册页。 |
| 6to4 隧道 | 仅 IPv6 接口 | 发出 `ipadm create-ip` 命令时，将自动选择缺省的 IPv6 地址。请参见 ipadm(1M) 手册页。 |

您可以通过使用 `ipadm create-addr -T addrconf` 命令指定本地和远程 `interface-id` 来覆盖自动为 IPv6 或 IPv4 隧道上的 IPv6 接口设置的链路本地地址。

♦ ♦ ♦ 第 5 章

# 5 管理 IP 隧道

本章介绍在 Oracle Solaris 中管理 IP 隧道的任务。有关 Oracle Solaris 中的 IP 隧道管理的概述，请参见第 4 章 关于 IP 隧道管理。

本章包含以下主题：



## Oracle Solaris 中的 IP 隧道管理

在此 Oracle Solaris 发行版中，隧道管理将与 IP 接口配置分开。可以使用 `dladm` 命令管理 IP 隧道的数据链路方面，使用 `ipadm` 命令管理配置的 IP 方面（包括 IP 隧道的配置）。

可以使用下面的 `dladm` 子命令配置 IP 隧道：

- `create-iptun`
- `modify-iptun`
- `show-iptun`
- `delete-iptun`
- `set-linkprop`

有关更多信息，请参见 `dladm`(1M) 手册页。

---

注 - IP 隧道管理与 IPsec 配置密切相关。例如，IPsec 虚拟专用网络 (virtual private network, VPN) 是主要使用 IP 隧道的网络之一。有关 Oracle Solaris 中的网络安全性的更多信息，请参见《在 Oracle Solaris 11.2 中确保网络安全 》中的第 6 章 “关于 IP 安全体系结构”。要配置 IPsec，请参见《在 Oracle Solaris 11.2 中确保网络安全 》中的第 7 章 “配置 IPsec”。

---

# 管理 IP 隧道

本节包含以下主题：



## ▼ 如何创建和配置 IP 隧道

1. **成为 `root` 角色。**

2. **创建隧道。**

```
# dladm create-iptun [-t] -T type -a [local|remote]=addr,... tunnel-link
```

`-t`
创建临时隧道。缺省情况下，该命令将创建一个持久性隧道。
如果要在隧道中配置一个持久性 IP 接口，则必须创建一个持久性隧道并且不使用 `-t` 选项。

`-T` *type*
指定要创建的隧道的类型。创建所有隧道类型都需要此参数。

`-a [local|remote]=`*address,...*
指定对应于本地地址和远程隧道地址的字面值 IP 地址或主机名。这些地址必须有效并且已在系统中创建。取决于隧道的类型，仅指定一个地址或同时指定本地和远程地址。如果指定本地和远程地址，则必须使用逗号分隔这两个地址。

- `IPv4` 隧道需要有本地和远程 IPv4 地址才能正常工作。
- `IPv6` 隧道需要有本地和远程 IPv6 地址才能正常工作。
- `6to4` 隧道需要有本地 IPv4 地址才能正常工作。

---

注 - 对于持久性 IP 隧道数据链路配置，如果对地址使用主机名，这些主机名将保存在配置存储中。在后续系统引导期间，如果名称解析到的 IP 地址不同于创建隧道时使用的 IP 地址，则隧道将获取新配置。

---

*tunnel-link*
指定 IP 隧道链路。如果此发行版的网络链路管理中支持有意义的名称，则隧道名称不再限制为要创建的隧道类型。相反，可以

为隧道指定任何通过管理方式选择的名称。隧道名称由字符串和物理连接点 (physical point of attachment, PPA) 编号组成，例如 *mytunnel0*。有关管理有意义名称的指定的规则，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“有效链路名称的规则”。

3. （可选） 设置跃点限制或封装限制的值。

```
# dladm set-linkprop -p [hoplimit=value] [encaplimit=value] tunnel-link
```

`hoplimit` 指定 IPv6 上隧道的隧道接口的跃点限制。*hoplimit* 与 IPv4 上隧道的 IPv4 生存时间 (time to live, TTL) 字段等效。

`encaplimit` 指定包允许的嵌套隧道的级数。此选项仅适用于 IPv6 隧道。

为 `hoplimit` 和 `encaplimit` 属性设置的值必须处于可接受的范围内。`hoplimit` 和 `encaplimit` 属性为隧道链路属性。因此，与其他链路属性一样，这些属性通过相同的 `dladm` 子命令管理。您使用的子命令为 `dladm set-linkprop`、`dladm reset-linkprop` 和 `dladm show-linkprop`。

4. 创建隧道上的 IP 接口。

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

其中，*tunnel-interface* 使用与隧道链路相同的名称。

5. 为隧道接口指定本地和远程 IP 地址。

```
# ipadm create-addr [-t] -a local=address,remote=address interface
```

其中，*interface* 指定隧道接口。

有关更多信息，请参见 `ipadm`(1M) 手册页和《在 Oracle Solaris 11.2 中配置和管理网络组件 》。

6. （可选） 验证隧道的 IP 接口配置的状态。

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

例 5-1 在 IPv4 隧道上创建 IPv6 接口

以下示例演示如何创建持久性 IPv6 over IPv4 隧道。

```
# dladm create-iptun -T ipv4 -a local=192.0.2.23,remote=203.0.113.14 private0
# dladm set-linkprop -p hoplimit=200 private0
# ipadm create-ip private0
# ipadm create-addr -T addrconf private0
private0/v6
```

```
# ipadm show-addr private0/
ADDROBJ           TYPE     STATE        ADDR
private0/v6       addrconf ok fe80::c000:217->fe80::cb00:710e
```

要添加备用地址，请使用相同语法。例如，可以按如下所示添加全局地址：

```
# ipadm create-addr -a local=2001:db8:4728::1,remote=2001:db8:4728::2 private0
private0/v6a
# ipadm show-addr private0/
ADDROBJ           TYPE     STATE        ADDR
private0/v6       addrconf ok fe80::c000:217->fe80::cb00:710e
private0/v6a      static   ok 2001:db8:4728::1->2001:db8:4728::2
```

请注意，IPv6 地址的前缀 `2001:db8` 为特殊的 IPv6 前缀，专门用于文档示例。

例 5-2 在 IPv4 隧道上创建 IPv4 接口

以下示例演示如何创建持久性 IPv4 over IPv4 隧道。

```
# dladm create-iptun -T ipv4 -a local=192.0.2.23,remote=203.0.113.14 vpn0
# ipadm create-ip vpn0
# ipadm create-addr -a local=10.0.0.1,remote=10.0.0.2 vpn0
vpn0/v4
# ipadm show-addr vpn0/
ADDROBJ           TYPE     STATE        ADDR
vpn0/v4           static   ok 10.0.0.1->10.0.0.2
```

您可以进一步配置 IPsec 策略来为流经此隧道的包提供安全连接。有关信息，请参见《在 Oracle Solaris 11.2 中确保网络安全 》中的第 7 章 “配置 IPsec”。

例 5-3 在 IPv6 隧道上创建 IPv6 接口

以下示例演示如何创建持久性 IPv6 over IPv6 隧道。

```
# dladm create-iptun -T ipv6 -a local=2001:db8:feed::1234,remote=2001:db8:beef::4321 tun0
# ipadm create-ip tun0
# ipadm create-addr -T addrconf tun0
tun0/v6
# ipadm show-addr tun0/
ADDROBJ           TYPE     STATE        ADDR
tun0/v6           addrconf ok fe80::1234->fe80::4321
```

要添加地址（例如全局地址或替代本地和远程地址），请按如下所示使用 `ipadm` 命令：

```
# ipadm create-addr -a local=2001:db8:cafe::1,remote=2001:db8:cafe::2 tun0
tun0/v6a
# ipadm show-addr tun0/
ADDROBJ           TYPE     STATE        ADDR
tun0/v6           addrconf ok fe80::1234->fe80::4321
tun0/v6a          static   ok 2001:db8:cafe::1->2001:db8:cafe::2
```

# ▼ 如何配置 6to4 隧道

配置 6to4 隧道，6to4 路由器必须充当网络的 6to4 站点中节点的 IPv6 路由器。因此，当配置 6to4 路由器时，还必须在其物理接口上将该路由器配置为 IPv6 路由器。有关将 Oracle Solaris 主机配置为路由器的更多信息，请参见《将 Oracle Solaris 11.2 系统配置为路由器或负载平衡器 》中的“配置 IPv6 路由器”。

1. 创建 6to4 隧道。

   ```
   # dladm create-iptun -T 6to4 -a local=address tunnel-link
   ```

   -a local=*address*　　指定隧道本地地址，该地址必须是系统中已存在的有效地址。

   *tunnel-link*　　指定 IP 隧道链路。如果网络链路管理中支持有意义的名称，则隧道名称不再限制为要创建的隧道类型。相反，可以为隧道指定任何通过管理方式选择的名称。隧道名称由字符串和 PPA 编号组成，例如 *mytunnel0*。有关更多信息，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的“有效链路名称的规则”。

2. 创建隧道 IP 接口。

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

   其中，*tunnel-interface* 使用与隧道链路相同的名称。

3. （可选） 为所使用的隧道添加替代 IPv6 地址。

4. 编辑 **/etc/inet/ndpd.conf** 文件。

   ```
   # pfedit /etc/inet/ndpd.conf
   ```

5. 通过在该文件中添加以下两行来通告 6to4 路由。

   ```
   if subnet-interface AdvSendAdvertisements 1
   IPv6-address subnet-interface
   ```

   其中，第一行指定接收通告的子网，*subnet-interface* 是指该子网连接到的链路。第二行中的 IPv6 地址必须具有 6to4 前缀 `2000`，该前缀用于 6to4 隧道中的 IPv6 地址。

   有关 `ndpd.conf` 文件的详细信息，请参见`ndpd.conf(4)`手册页。

6. 启用 IPv6 转发。

   ```
   # ipadm set-prop -p forwarding=on ipv6
   ```

7. 从下列选项中选择一个：

   - 重新引导路由器。

- 向 **/etc/inet/in.ndpd** 守护进程发出 **sighup**，以便开始发送路由器通告。
  要接收 6to4 前缀的每个子网上的 IPv6 节点自动配置有新的 6to4 派生地址。

8. 将节点的新 6to4 派生地址添加到在 6to4 站点上使用的名称服务中。
   有关说明，请参见《在 Oracle Solaris 11.2 中配置和管理网络组件 》中的第 4 章 “在 Oracle Solaris 客户机上管理命名和目录服务”。

例 5-4 创建 6to4 隧道

以下示例演示如何创建 6to4 隧道。请注意，在 6to4 隧道上只能配置 IPv6 接口。在此示例中，子网接口为 /etc/inet/ndpd.conf 引用的 net0。

```
# dladm create-iptun -T 6to4 -a local=192.0.2.23 tun0
# ipadm create-ip tun0
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           dhcp     ok           192.0.2.23/24
lo0/v6            static   ok           ::1/128
tun0/v6           static   ok           2002:c000:217::1/16

# ipadm create-addr -T addrconf net0
net0/v6
# ipadm create-addr -a 2002:c000:217:cafe::1 net0
net0/v6a
# ipadm show-addr
ADDROBJ           TYPE     STATE        ADDR
lo0/v4            static   ok           127.0.0.1/8
net0/v4           dhcp     ok           192.0.2.23/24
lo0/v6            static   ok           ::1/128
net0/v6           addrconf ok     fe80::214:4fff:fef9:b1a9/10
net0/v6a          static   ok           2002:c000:217:cafe::1/64
tun0/v6           static   ok           2002:c000:217::1/16

# vi /etc/inet/ndpd.conf
if net0 AdvSendAdvertisements on
prefix 2002:c000:217:cafe::0/64 net0

# ipadm set-prop -p forwarding=on ipv6
```

请注意，对于 6to4 隧道，IPv6 地址的前缀为 2002。

# ▼ 如何启用通往 6to4 中继路由器的 6to4 隧道

注意 - 由于重大安全问题，缺省情况下，在 Oracle Solaris 中禁用了 6to4 中继路由器支持。有关详细信息，请参见《在 Oracle Solaris 11.2 中排除网络管理问题 》中的“建立通往 6to4 中继路由器的隧道时的安全问题”。

开始之前 在启用通往 6to4 中继路由器的 6to4 隧道之前，需完成以下任务：

- 在您的站点中配置 6to4 路由器。请参见如何创建和配置 IP 隧道 [96]。
- 检查建立通往 6to4 中继路由器的隧道连接时涉及到的安全问题。

1. 使用以下方法之一启用通往 6to4 中继路由器的隧道：

   - 启用通往任播 6to4 中继路由器的隧道。

     ```
     # 6to4relay -e
     ```

     `-e` 选项可用于在 6to4 路由器和任播 6to4 中继路由器之间设置隧道。任播 6to4 中继路由器具有已知的 IPv4 地址 `192.88.99.1`。物理位置距离您的站点最近的任播中继路由器将成为 6to4 隧道的端点。该中继路由器随后将在 6to4 站点和本机 IPv6 站点之间转发包。

     有关详细信息，请参见 RFC 3068, "An Anycast Prefix for 6to4 Relay Routers" (http://www.rfc-editor.org/rfc/rfc3068.txt)（RFC 3068，6to4 中继路由器的任播前缀）。

   - 启用通往特定 6to4 中继路由器的隧道。

     ```
     # 6to4relay -e -a relay-router-address
     ```

     `-a` 选项表示后面将跟有一个特定路由器地址。请将 *relay-router-address* 替换为用以启用隧道的特定 6to4 中继路由器的 IPv4 地址。

   除非删除 6to4 隧道的伪接口，否则通往 6to4 中继路由器的隧道将一直保持活动状态。

2. 如果不再需要隧道，请删除通往 6to4 中继路由器的隧道。

   ```
   # 6to4relay -d
   ```

3. （可选） 使通往 6to4 中继路由器的隧道在重新引导过程中持续保留。

   您的站点可能迫切要求通往 6to4 中继路由器的隧道在 6to4 路由器每次重新引导时都进行恢复。要支持此方案，必须执行下列操作：

   a. 编辑 `/etc/default/inetinit` 文件。

      ```
      # pfedit /etc/default/inetinit
      ```

      要修改的行位于该文件的末尾。

b. 将 **ACCEPT6TO4RELAY=NO** 行中的 **NO** 值更改为 **YES**。

c. （可选） 创建通往特定 6to4 中继路由器的隧道，该隧道在重新引导过程中持续保留。

对于 RELAY6TO4ADDR 参数，请将 192.88.99.1 地址更改为要使用的 6to4 中继路由器的 IPv4 地址。

例 5-5 获取有关 6to4 中继路由器支持的状态信息

可以使用 6to4relay 命令来确定是否启用了对 6to4 中继路由器的支持。以下示例显示了禁用 6to4 中继路由器支持（此为 Oracle Solaris 中的缺省设置）时的输出。

```
# 6to4relay
6to4relay: 6to4 Relay Router communication support is disabled.
```

启用对 6to4 中继路由器的支持时，将显示以下输出：

```
# 6to4relay
6to4relay: 6to4 Relay Router communication support is enabled.
IPv4 remote address of Relay Router=192.88.99.1
```

# 修改 IP 隧道配置

可以使用下面的命令语法来更改隧道的配置：

```
# dladm modify-iptun -a [local|remote]=addr,... tunnel-link
```

无法修改现有隧道的类型。因此，不允许对此命令使用 -T *type* 选项。只能修改以下隧道参数：

-a [local|remote]=*address,...*

指定对应于本地地址和远程隧道地址的字面值 IP 地址或主机名。取决于隧道的类型，仅指定一个地址或同时指定本地和远程地址。如果指定本地和远程地址，则必须使用逗号分隔这两个地址。

- IPv4 隧道需要有本地和远程 IPv4 地址才能正常工作。
- IPv6 隧道需要有本地和远程 IPv6 地址才能正常工作。
- 6to4 隧道需要有本地 IPv4 地址才能正常工作。

对于持久性 IP 隧道数据链路配置，如果对地址使用主机名，这些主机名将保存在配置存储中。在后续系统引导期间，如果名称解析到的 IP 地址不同于创建隧道时使用的 IP 地址，则隧道将获取新配置。

如果要更改隧道的本地和远程地址，请确保这些地址与要修改的隧道类型一致。

- 要更改隧道链路的名称，请使用 `dladm rename-link` 命令而不是 `modify-iptun` 命令，如下所示：

  `# dladm rename-link` *old-tunnel-link new-tunnel-link*

- 要更改 `hoplimit` 或 `encaplimit` 等隧道属性，请使用 `dladm set-linkprop` 命令而不是 `modify-iptun` 命令。

例 5-6 修改隧道的地址和属性

以下示例由两个过程组成。首先，临时更改 IPv4 隧道 `vpn0` 的本地和远程地址。以后重新引导系统时，隧道将恢复为使用原始地址。第二条命令演示如何将 `vpn0` 的 `hoplimit` 更改为 60。

```
# dladm modify-iptun -t -a local=10.8.48.149,remote=192.168.2.3 vpn0

# dladm set-linkprop -p hoplimit=60 vpn0
```

# 显示 IP 隧道的配置

可以使用下面的命令语法来显示 IP 隧道的配置：

```
# dladm show-iptun [-p] -o fields [tunnel-link]
```

| | |
|---|---|
| `-p` | 显示计算机可解析格式的信息。此参数是可选的。 |
| `-o` *fields* | 显示提供特定隧道信息的选定字段。 |
| *tunnel-link* | 指定要显示其配置信息的隧道。此参数是可选的。如果省略隧道名称，该命令将显示有关系统中所有隧道的信息。 |

例 5-7 显示有关所有隧道的信息

在下面的示例中，系统上只存在一个隧道。

```
# dladm show-iptun
LINK     TYPE     FLAGS     LOCAL          REMOTE
tun0     6to4     --        192.168.35.10  --
vpn0     ipv4     --        10.8.48.149    192.168.2.3
```

例 5-8 显示计算机可解析格式的选定字段

在下面的示例中，只显示具有隧道信息的特定字段。

```
# dladm show-iptun -p -o link,type,local
tun0:6to4:192.168.35.10
vpn0:ipv4:10.8.48.149
```

## 显示 IP 隧道的属性

可以使用下面的命令语法来显示隧道链路的属性：

```
# dladm show-linkprop [-c] [-o fields] [tunnel-link]
```

-c　　显示计算机可解析格式的信息。此参数是可选的。

-o *fields*　　显示提供有关链路属性的特定信息的选定字段。

*tunnel-link*　　指定要显示其属性的隧道。此参数是可选的。如果省略隧道名称，该命令将显示有关系统中所有隧道的信息。

例 5-9　显示隧道的属性

以下示例演示如何显示隧道的所有链路属性。

```
# dladm show-linkprop iptun0
LINK     PROPERTY        PERM VALUE        EFFECTIVE    DEFAULT   POSSIBLE
iptun0   autopush        rw   --           --           --        --
iptun0   zone            rw   --           --           --        --
iptun0   state           r-   up           up           up        up,down
iptun0   mtu             rw   1480         1480         1480      1280-1480
iptun0   maxbw           rw   --           --           --        --
iptun0   cpus            rw   --           --           --        --
iptun0   rxfanout        rw   --           8            8         --
iptun0   pool            rw   --           --           --        --
iptun0   priority        rw   medium       medium       medium    low,medium,
                                                                  high
iptun0   hoplimit        rw   64           64           64        1-255
iptun0   protection      rw   --           --           --        mac-nospoof,
                                                                  restricted,
                                                                  ip-nospoof,
                                                                  dhcp-nospoof
iptun0   allowed-ips     rw   --           --           --        --
iptun0   allowed-dhcp-cids rw --           --           --        --
iptun0   rxrings         rw   --           --           --        --
iptun0   txrings         rw   --           --           --        --
iptun0   txringsavail    r-   0            0            --        --
iptun0   rxringsavail    r-   0            0            --        --
iptun0   rxhwclntavail   r-   0            0            --        --
iptun0   txhwclntavail   r-   0            0            --        --
```

## ▼ 如何删除 IP 隧道

1. 根据接口类型，取消激活隧道上配置的 IP 接口。

   ```
   # ipadm delete-ip tunnel-link
   ```

注 - 要成功删除隧道，隧道上应检测不到 IP 接口。

2. 删除 IP 隧道。

```
# dladm delete-iptun tunnel-link
```

此命令的唯一选项是 -t，将导致隧道被临时删除。重新引导系统后，将恢复该隧道。

例 5-10 删除为 IPv6 接口配置的 IPv6 隧道

在以下示例中，永久删除了持久性隧道。

```
# ipadm delete-ip ip6.tun0
# dladm delete-iptun ip6.tun0
```

# 索引