Oracle® Solaris 11.2 での TCP/IP ネットワーク、IPMP、および IP トンネルの管理

Part No: E53801
2014 年 7 月

ORACLE

目次

このドキュメントの使用法

- **概要** – Oracle Solaris オペレーティングシステム (OS) で TCP/IP ネットワーク、IPMP、IP トンネルを管理するためのタスクについて説明します。
- **対象読者** – システム管理者。
- **前提知識** - TCP/IP ネットワークおよび高度なネットワーク機能 (IPMP および IP トンネルなど) の管理を含む、高度なネットワーク管理の経験。

製品ドキュメントライブラリ

この製品の最新情報や既知の問題は、ドキュメントライブラリ (http://www.oracle.com/pls/topic/lookup?ctx=E56342) に含まれています。

Oracle サポートへのアクセス

Oracle のお客様は、My Oracle Support を通じて電子的なサポートを利用することができます。詳細は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=info を参照してください。聴覚に障害をお持ちの場合は、http://www.oracle.com/pls/topic/lookup?ctx=acc&id=trs を参照してください。

フィードバック

このドキュメントに関するフィードバックを http://www.oracle.com/goto/docfeedback からお聞かせください。

♦ ♦ ♦ 第 1 章

1 TCP/IP ネットワークの管理

この章では、Oracle Solaris がインストールされているシステムで TCP/IP プロトコルを管理する方法について説明します。この章のタスクでは、サイトで TCP/IP ネットワークが、IPv4 専用、または IPv4/IPv6 のデュアルスタックで稼働しているものとします。

ネットワーク配備計画の詳細については、『Oracle Solaris 11.2 でのネットワーク配備の計画 』を参照してください。

ネットワークの構成タスクについては、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』を参照してください。

TCP/IP ネットワークのトラブルシューティングに関する一般的な情報については、『Oracle Solaris 11.2 でのネットワーク管理のトラブルシューティング 』を参照してください。

この章の内容は、次のとおりです。

TCP/IP プロパティーのカスタマイズ

TCP/IP プロトコルの大半のプロパティー (*チューニング可能値*とも呼ばれる) を構成するには、ipadm コマンドを使用します。ipadm コマンドは、チューニング可能値の主要な設定ツールとして、ndd コマンドに置き換わるものです。これらの変更点の詳細については、『Oracle Solaris 10 から Oracle Solaris 11.2 への移行 』の「ndd コマンドと ipadm コマンドの比較」を参照してください。

TCP/IP プロパティーには、インタフェースベースのものとグローバルなものがあり、特定のインタフェースに適用したり、ゾーン内のすべてのインタフェースにグローバルに適用したりできます。グローバルなプロパティーは、非大域ゾーンごとに異なる値を持つことができます。サポートされるプロトコルプロパティーの一覧については、ipadm(1M)のマニュアルページを参照してください。

ネットワークが機能するには、通常、TCP/IP インターネットプロトコルのデフォルト値で十分です。ただし、特定のネットワークトポロジで使用するのに、これらの値では不十分な場合は、次の 3 つの ipadm サブコマンドを使用してプロパティーをカスタマイズできます。

- `ipadm show-prop -p` *property protocol* – プロトコルとその現在の値を表示します。-p *property* オプションを使用しなかった場合は、プロトコルのすべてのプロパティーが表示されます。プロトコルを指定しなかった場合は、すべてのプロトコルのすべてのプロパティーが表示されます。
- `ipadm set-prop -p` *property*=*value protocol* – プロトコルのプロパティーに値を割り当てます。
  - プロトコルのプロパティーに複数の値を割り当てるには、次の構文を使用します。

    `ipadm set-prop [-t] -p` *property*=*value*`[,...] protocol`
  - 特定のプロパティーの値のセットから 1 つの値を削除するには、次のように -= 修飾子を使用します。

    `ipadm set-prop -p` *property*`-=`*value2*
- 特定のプロトコルプロパティーをデフォルト値にリセットする方法は次のとおりです。

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

IP インタフェースプロパティーのカスタマイズに関する詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「IP インタフェースプロパティーとアドレスのカスタマイズ」を参照してください。

IP アドレスプロパティーのカスタマイズに関する詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「IP アドレスプロパティーのカスタマイズ」を参照してください。

グローバルなパケット転送の有効化

`ipadm set-ifprop` コマンドを使用して IP インタフェース上で転送を有効にした場合は、そのインタフェースでのみ転送が有効になり、ほかのすべてのインタフェース上での転送は変わらないままです。個々の IP インタフェースプロパティーでパケット転送を設定することにより、システムの特定のインタフェースで選択的にこの機能を実装できます。詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「パケット転送の有効化」を参照してください。

システム全体でパケット転送を有効にするには、IP インタフェースの数に関係なく、`protocol` プロパティーを使用します。`forwarding` プロパティーは転送を管理するために使用されるグローバルな IP プロパティーで、個々の IP インタフェースでの転送の管理に使用されるプロパティーと同じ名前です。転送は、IPv4 または IPv6 プロトコル (あるいはその両方) で有効にできるため、各プロトコルを個別に管理する必要があります。

たとえば、次のように、システム上のすべての IPv4 および IPv6 トラフィックでパケット転送を有効にします。

```
# 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
```

注記 - IP インタフェースとプロトコルの `forwarding` プロパティーは、どちらも排他的ではありません。インタフェースとプロトコルのプロパティーを同時に設定できます。たとえば、プロトコル上でグローバルにパケット転送を有効にしてから、システム上の IP インタフェースごとにパケット転送をカスタマイズできます。したがって、パケット転送は、グローバルに有効化できるものの、インタフェースごとに選択的に管理することもできます。

デフォルトアドレス選択の管理

Oracle Solaris では、単一のインタフェースに複数の IP アドレスを付与できます。たとえば、IPMP のようなテクノロジを使用すると、複数のネットワークインタフェースカード (NIC) が同じ IP リンク層に接続できます。このようなリンクは 1 つまたは複数の IP アドレスを持つことができます。さらに、IPv6 が有効なシステム上のインタフェースは、1 つの IPv6 リンクローカルアドレス、少なくとも 1 つの IPv6 ルーティングアドレス、および (少なくとも 1 つのインタフェースに) 1 つの IPv4 アドレスを持ちます。

システムがトランザクションを開始すると、アプリケーションは `getaddrinfo` ソケットを呼び出します。`getaddrinfo` ソケットは、宛先のシステムで使用されている可能性のあるアドレスを検出します。そのあと、カーネルはこのリストに優先度を付けて、パケットに使用するのに最適な宛先を見つけます。このプロセスのことを「*宛先アドレス順番付け*」と呼びます。そのあと、Oracle Solaris カーネルは、パケットに最適な宛先アドレスに対して、適切なソースアドレスの書式を選択します。このプロセスのことを*アドレス選択*と呼びます。宛先アドレス順番付けの詳細については、`getaddrinfo`(3SOCKET)のマニュアルページを参照してください。

IPv4 専用システムとデュアルスタック IPv4/IPv6 システムは両方とも、デフォルトアドレス選択を実行する必要があります。ほとんどの状況では、デフォルトアドレス選択メカニズムを変更する必要はありません。しかし、IPMP をサポートしたり、6to4 アドレス書式を選択したりする場合は、アドレス書式の優先度を変更する必要があります。

`/etc/inet/ipaddrsel.conf` 構成ファイルの説明

`/etc/inet/ipaddrsel.conf` ファイルには、IPv6 デフォルトアドレス選択ポリシーテーブルが含まれます。Oracle Solaris をインストールしたときに IPv6 を有効にした場合、このファイルには、表1-1「IPv6 アドレス選択ポリシーテーブル」 に示す内容が含まれます。

`/etc/inet/ipaddrsel.conf` ファイルの内容は編集できます。しかし、このファイルを変更することは極力避けるべきです。どうしても変更が必要な場合、手順については、14 ページの「IPv6 アドレス選択ポリシーテーブルを管理する方法」を参照してください。`ippaddrsel.conf` の詳細については、13 ページの「IPv6 アドレス選択ポリシーテーブルを変更する理由」および`ipaddrsel.conf`(4)のマニュアルページを参照してください。

ipaddrsel コマンドの説明

ipaddrsel コマンドを使用すると、IPv6 デフォルトアドレス選択ポリシーテーブルを変更できます。

Oracle Solaris カーネルは IPv6 デフォルトアドレス選択ポリシーテーブルを使用して、IPv6 パケットヘッダーに対して、宛先アドレス順番付けやソースアドレス選択を実行します。/etc/inet/ipaddrsel.conf ファイルには、このポリシーテーブルが含まれます。

次の表に、このポリシーテーブルのデフォルトアドレス書式とその優先度のリストを示します。IPv6 アドレス選択に関する技術的な詳細については、inet6(7P)のマニュアルページを参照してください。

表 1-1 IPv6 アドレス選択ポリシーテーブル

接頭辞	優先度	定義
::1/128	50	ループバック
::/0	40	デフォルト
2002::/16	30	6to4
::/96	20	IPv4 互換
::ffff:0:0/96	10	IPv4

この表では、IPv6 接頭辞 (::1/128 と ::/0) は 6to4 アドレス (2002::/16) と IPv4 アドレス (::/96 と ::ffff:0:0/96) よりも優先されます。したがって、カーネルは、別の IPv6 宛先に向かうパケットに対して、インタフェースのグローバル IPv6 アドレスをデフォルトで選択します。インタフェースの IPv4 アドレスの優先度は、特に IPv6 宛先に向かうパケットに対しては低くなります。選択した IPv6 ソースアドレスを考えて、カーネルは宛先アドレスにも IPv6 書式を使用します。

IPv6 アドレス選択ポリシーテーブルを変更する理由

ほとんどの場合、IPv6 デフォルトアドレス選択ポリシーテーブルを変更する必要はありません。ポリシーテーブルを管理する必要がある場合は、ipaddrsel コマンドを使用します。

ポリシーテーブルは、次のような理由で変更する必要があることがあります。

- システムが 6to4 トンネル用のインタフェースを持っている場合、6to4 アドレスにより高い優先度を与えることができます。

- 特定の宛先アドレスと通信するときだけ特定のソースアドレスを使用したい場合、これらのアドレスをポリシーテーブルに追加します。そのあと、`ipadm` コマンドを使用して、これらのアドレスが優先されるようにフラグを立てることができます。詳細は、`ipadm`(1M)のマニュアルページを参照してください。
- IPv4 アドレスを IPv6 アドレスよりも優先させたい場合、`::ffff:0:0/96` の優先度をより大きな値に変更します。
- 旧式のアドレスにより高い優先度を割り当てる必要がある場合は、旧式のアドレスをポリシーテーブルに追加します。たとえば、IPv6 内でサイトのローカルアドレスが旧式であると仮定します。これらのアドレスには、`fec0::/10` という接頭辞があります。サイトのローカルアドレスにより高い優先度を与えるようにポリシーテーブルを変更できます。

`ipaddrsel` コマンドの詳細については、`ipaddrsel`(1M)のマニュアルページを参照してください。

▼ IPv6 アドレス選択ポリシーテーブルを管理する方法

次の手順では、アドレス選択ポリシーテーブルを変更する方法について説明します。IPv6 デフォルトアドレス選択の概念については、ipaddrsel コマンドの説明を参照してください。

注意 - 次の手順に示した理由がある場合を除き、IPv6 アドレス選択ポリシーテーブルを変更しないでください。ポリシーテーブルを間違って変更すると、ネットワーク上で問題が発生する可能性があります。また、次の手順に示すように、必ずポリシーテーブルのバックアップを保存してください。

1. **管理者になります。**

2. **現在の IPv6 アドレス選択ポリシーテーブルを調査します。**

```
# 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. **デフォルトアドレス選択ポリシーテーブルのバックアップを作成します。**

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

4. **/etc/inet/ipaddrsel.conf にカスタマイズを追加します。**

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

/etc/inet/ipaddrsel のエントリには、次の構文を使用します。

prefix/prefix-length precedence label [# comment]

よく行われる変更の例については、例1-1「デフォルトの IPv6 アドレス選択ポリシーテーブルの変更」を参照してください。

5. **変更したポリシーテーブルをカーネルにロードします。**

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

6. **変更したポリシーテーブルに問題がある場合は、IPv6 デフォルトアドレス選択ポリシーテーブルを復元します。**

```
# ipaddrsel -d
```

例 1-1 デフォルトの IPv6 アドレス選択ポリシーテーブルの変更

次に、デフォルトアドレス選択ポリシーテーブルに対してよく行われる変更の例を示します。

- 6to4 アドレスに最高の優先度を付ける。

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

6to4 アドレス書式の優先度は現在、最高の 50 です。Loopback の優先度は、以前は 50 でしたが、現在は 45 です。ほかのアドレス書式の優先度は変わりません。

- 特定の宛先アドレスとの通信において、特定のソースアドレスを使用するように指示する場合。

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

このエントリは、物理インタフェースが 1 つしかないホストの場合に役立ちます。ここで、2001:1111:1111::1/128 は、ネットワーク 2001:2222:2222::/48 内にある宛先に向けられたすべてのパケットの送信元アドレスとして優先されます。優先度 40 は、このインタフェースに構成されたほかのアドレス書式よりも、送信元アドレス 2001:1111:1111::1/128 を優先することを指示します。

- IPv6 アドレスよりも IPv4 アドレスを優先する場合。

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

このテーブルでは、IPv4 書式 `::ffff:0.0.0.0/96` の優先度をデフォルトの 10 からテーブル内で最高の 60 に変更しています。

▼ 現在のセッションだけの IP6 アドレス選択テーブルを変更する方法

`/etc/inet/ipaddrsel.conf` ファイルを編集すると、その変更はリブート後も適用されます。変更したポリシーテーブルを現在のセッションだけに適用する場合、次の手順を実行します。

1. **管理者になります。**

2. **`/etc/inet/ipaddrsel` の内容を *filename* にコピーします (*filename* は自分が選択した名前)。**

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

3. **必要に応じて、`pfedit` コマンドを使用して *filename* 内のポリシーテーブルを編集します。**

4. **変更したポリシーテーブルをカーネルにロードします。**

```
# ipaddrsel -f filename
```

システムをリブートするまで、カーネルは新しいポリシーテーブルを使用します。

トランスポート層サービスの管理

Transmission Control Protocol (TCP)、Stream Control Transmission Protocol (SCTP)、および User Datagram Protocol (UDP) のトランスポート層プロトコルは、Oracle Solaris の標準部品です。通常、これらのプロトコルは、ユーザーの介入なしで正常に動作します。ただし、サイトの条件によっては、トランスポート層プロトコルの上で動作するサービスを記録

または変更しなければならない場合があります。その場合は、サービス管理機能 (SMF) コマンドを使用して、これらのサービスのプロファイルを変更する必要があります。

inetd デーモンは、システムがブートされると、標準的なインターネットサービスを起動します。これらのサービスは、TCP、SCTP、または UDP をトランスポート層プロトコルとして使用するアプリケーションなどです。SMF コマンドを使えば、既存のインターネットサービスを変更したり、新しいサービスを追加したりできます。

トランスポート層プロトコルが関係する操作には、次の操作があります。

- 特権ポートを設定する
- トラフィックの輻輳制御を実装する
- すべての着信 TCP 接続をロギングする
- トランスポート層プロトコル (たとえば、SCTP) の上で動作するサービスを追加する
- アクセス制御のために TCP ラッパー機能を構成する
- TCP 受信バッファーサイズを変更する

詳細は、『Oracle Solaris 11.2 でのシステムサービスの管理 』の「inetd で制御されるサービスの変更」および**inetd**(1M)のマニュアルページを参照してください。

特権ポートの設定

TCP、UDP、SCTP などのトランスポートプロトコルでは、ポート 1-1023 はデフォルトの特権ポートです。特権ポートにバインドするには、プロセスを root 権限で実行する必要があります。1023 より大きいポートは、デフォルトで非特権になっています。ipadm コマンドを使用して特権ポートの範囲を拡張したり、非特権範囲にある特定のポートを特権ポートとしてマークしたりできます。

特権ポートの範囲を管理するには、次のトランスポートプロトコルプロパティーをカスタマイズします。

smallest_nonpriv_port
: 非特権ポート番号の範囲 (通常のユーザーがバインドできるポート) の開始値を指定します。非特権ポート範囲内の個々のポートを特権ポートとして設定できます。プロパティーの値を表示するには、ipadm show-prop コマンドを使用します。

extra_priv_ports
: 特権範囲外のポートのうち特権にするポートを指定します。制限するポートを指定するには、ipadm set-prop コマンドを使用します。このプロパティーには複数の値を割り当てることができます。

たとえば、TCP ポート 3001 および 3050 に対するアクセスを、root 役割だけに与えるように制限するとします。smallest_nonpriv_port プロパティーは、1024 が非特権ポートの最小ポート番号であることを示しています。したがって、指定されたポート 3001 と 3050 を特権ポートに変更するには、次のようにします。

```
# 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
```

たとえば、特権ポート 4045 を削除するには、次のようにします。

```
# 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
```

トラフィックの輻輳制御の実装

ネットワークの輻輳は通常、ネットワークが対応できる量を上回るパケットをノードが送信したときに、ルーターバッファーオーバーフローの形で発生します。トラフィックの輻輳は、さまざまなアルゴリズムによって送信側システムに対する制御を設定することで回避されます。これらのアルゴリズムは Oracle Solaris でサポートされており、サポートされている組み込みのアルゴリズムを説明した次の表に示すとおり、OS に簡単に追加したり、直接プラグインしたりできます。

アルゴリズム	Oracle Solaris 名	説明
NewReno	newreno	Oracle Solaris のデフォルトのアルゴリズム。この制御メカニズムには、送信側の輻輳ウィンドウ、スロースタート、および輻輳回避があります。
HighSpeed	highspeed	高速ネットワーク用のもっとも有名かつシンプルな NewReno の修正版の 1 つ。

アルゴリズム	Oracle Solaris 名	説明
CUBIC	cubic	Linux 2.6 の現在のデフォルトアルゴリズム。輻輳回避フェーズを線形的なウィンドウ増加から cubic 関数に変更します。
Vegas	vegas	実際のパケットロスを発生させずに輻輳を予測しようとする典型的な遅延ベースのアルゴリズム。

輻輳制御を有効にするには、次に示す制御関連の TCP プロパティーを設定します。これらのプロパティーは TCP について表示されますが、これらのプロパティーによって有効になる制御メカニズムは、SCTP トラフィックにも適用されます。

`cong_enabled` システムで現在動作中のアルゴリズムをコンマで区切ったリストが格納されます。アルゴリズムを追加または削除して、使用するアルゴリズムのみを有効にすることができます。このプロパティーには複数の値を設定できます。このため、行う変更に応じて `+=` 修飾子または `-=` 修飾子を使用する必要があります。

`cong_default` アプリケーションが socket オプションで明示的にアルゴリズムを指定しない場合に、デフォルトで使用されます。`cong_default` プロパティーの値は大域ゾーンと非大域ゾーンの両方に適用されます。

次の例は、プロトコルに輻輳制御のアルゴリズムを追加する方法を示しています。

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

次のようにアルゴリズムを削除します。

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

次のようにデフォルトのアルゴリズムを置き換えます。

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

注記 - アルゴリズムを追加または削除するときに従う順序の規則はありません。ほかのアルゴリズムをプロパティーに追加する前に、アルゴリズムを削除できます。ただし、`cong_default` プロパティーには定義済みのアルゴリズムが常に設定されている必要があります。

次の例は、輻輳制御を実装する方法を示しています。この例では、TCP プロトコルのデフォルトのアルゴリズムが `newreno` から `cubic` に変更されています。次に、`vegas` アルゴリズムが有効なアルゴリズムのリストから削除されています。

```
# 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
```

すべての着信 TCP 接続の IP アドレスをロギングする方法

すべての着信 TCP 接続の IP アドレスは、`syslog` サービスによって、`daemon.notice` ファシリティーとレベルで記録されます。この情報は、ローカルの `syslog.conf` ファイルの設定を変更していないかぎり、`/var/adm/messages` に置かれています。`syslog.conf` ファイルの設定を変更すると、この情報の格納先が変わる可能性があります。詳細は、`syslog.conf`(4)のマニュアルページを参照してください。

次のように、`inetd` デーモンで管理されるすべてのサービスに対して TCP トレースを `TRUE` (使用可能) にします。

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

SCTP プロトコルを使用するサービスの追加

Stream Control Transmission Protocol (SCTP) プロトコルは、TCP に類似した方法でアプリケーション層プロトコルにサービスを提供します。ただし、SCTP では 2 つのシステム (一方または両方がマルチホームでもよい) 間での通信が可能です。SCTP 接続は「*アソシエーション*」と呼ばれます。アソシエーションでは、アプリケーションがデータを分割し、1 つまたは複数のメッセージストリームとして伝送します (*マルチストリーム化*)。SCTP 接続は、複数の IP アドレスを持つエンドポイントに到達できます。これは、テレフォニーアプリケーションにとって特に重要です。IP Filter や IPsec を使用する場合、SCTP のマルチホーム機能はセキュリティーの点で考慮を要します。考慮点については、`sctp`(7P)のマニュアルページを参照してください。

▼ SCTP プロトコルを使用するサービスを追加する方法

デフォルトで SCTP は Oracle Solaris に組み込まれており、構成を別に行う必要はありません。ただし、SCTP を使用するためには、一定のアプリケーション層サービスを明示的に構成しなければならない場合があります。例として、`echo` や `discard` などのアプリケーションがあります。次の手順では、SCTP 1 対 1 スタイルのソケットを使用する echo サービスを追加する方法を説明します。TCP および UDP 用のサービスを追加する場合も同じ手順を使用できます。

次のタスクでは、`inetd` デーモンによって管理される SCTP `inet` サービスを SMF リポジトリに追加する方法を説明します。このタスクでは、次に、SMF コマンドを使用してサービスを追加する方法について説明します。

始める前に 次の手順を実行する前に、サービスのマニフェストファイルを作成してください。この手順では、例として、`echo` サービスのマニフェスト `echo.sctp.xml` を使用しています。

1. **システムファイルに対する書き込みアクセス権を持つユーザーアカウントでローカルシステムにログインします。**

2. **`pfedit` コマンドを使用して `/etc/services` ファイルに新しいサービスの定義を追加します。**

   `pfedit`(1M)のマニュアルページを参照してください。

   サービスを定義する構文は次のとおりです。

   service-name port/protocol aliases

3. **サービスマニフェストが格納されているディレクトリに移動して、サービスマニフェストをインポートします。**

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

   たとえば、次のように、`service.dir` ディレクトリにあるマニフェスト `echo.sctp.xml` を使用して、新しい SCTP `echo` サービスを追加します。

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

4. **サービスマニフェストが追加されているか確認します。**

   ```
   # svcs FMRI
   ```

   FMRI 引数には、サービスマニフェストの Fault Managed Resource Identifier (FMRI) を使用します。

5. **サービスのプロパティーを一覧表示して、変更を加える必要があるかどうかを判断します。**

   ```
   # inetadm -l FMRI
   ```

6. **新しいサービスを使用可能にします。**

   ```
   # inetadm -e FMRI
   ```

7. **サービスが使用可能になっていることを確認します。**

例 1-2 SCTP トランスポートプロトコルを使用するサービスの追加

次の例は、echo サービスに SCTP を使用させるために必要なコマンドとファイルエントリを示しています。

```
# 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
```

TCP ラッパーの構成

TCP ラッパーは、着信サービス要求とサービスデーモンの間で動作することによって、サービスデーモンにセキュリティー対策を追加します。TCP ラッパーは、正常および異常な接続の試みを記録します。さらに、TCP ラッパーはアクセス制御の機能を備えているため、要求の発行元に応じて接続を許可または拒否できます。TCP ラッパーを使用すると、Secure Shell (SSH)、Telnet、ファイル転送プロトコル (FTP) などのデーモンを保護できます。Support for TCP Wrappers From Version 8.12 of sendmail『Oracle Solaris 11.2 での sendmail サービスの管理 』の「sendmail の version 8.12 からの TCP ラッパーのサポート」アプリケーションでも TCP ラッパーを使用できます。

▼ TCP ラッパーを使って TCP サービスのアクセスを制御する方法

*TCP ラッパー*は `tcpd` プログラムによって実装されます。

1. **`root` の役割になります。**

2. **TCP ラッパーを使用可能にします。**

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

3. **TCP ラッパーのアクセス制御ポリシーを構成します。**

   手順については、`/usr/sfw/man` ディレクトリにある `hosts_access`(3) のマニュアルページを参照してください。

TCP 受信バッファーサイズの変更

TCP 受信バッファーのサイズは、TCP プロパティー `recv_buf` (デフォルトでは 128K バイト) を使用して設定します。しかし、アプリケーションは利用可能な帯域幅を均等に使用しません。このため、接続待機時間に応じてデフォルトのサイズを変更する必要があります。たと

えば、Oracle Solaris の Secure Shell 機能を使用すると、データストリームに対して追加のチェックサム処理や暗号化処理が実行されるため、帯域幅の使用量にオーバーヘッドが発生します。したがって、バッファーサイズを増やす必要がある可能性があります。同様に、一括転送を行うアプリケーションが帯域幅を効率的に使用できるようにするためにも、同じバッファーサイズ調整が必要です。

次のように、帯域幅遅延積 (BDP) を推定することで、使用する正しい受信バッファーサイズを計算できます。BDP は、使用可能な帯域幅に接続待機時間の値を乗算して算出します。

接続待機時間の値を取得するには、`ping -s` *host* コマンドを使用します。

適切な受信バッファーサイズは、BDP の値に近似します。ただし、帯域幅の使用量はさまざまな条件によって左右されます。インフラストラクチャーの共有や、帯域幅の使用で競合するアプリケーションとユーザーの数によって、その推定値は変化します。

次のように、バッファーサイズの値を変更します。

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

次の例は、バッファーサイズを 164K バイトに増やす方法を示しています。

```
# 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
```

推奨されるバッファーサイズは状況によって異なるため、推奨設定値はありません。次の例では、固有の条件を持つネットワークごとに BDP に異なる値を設定しています。

バッファーサイズのデフォルト値が 128K バイトである標準的な 1G ビット/秒ローカルエリアネットワーク (LAN):	`BDP = 128 MBps * 0.001 s = 128 kB`
待機時間が 100 ミリ秒である架空の 1G ビット/秒広域ネットワーク (WAN):	`BDP = 128 MBps * 0.1 s = 12.8 MB`

欧州米国間リンク (uperf で帯域幅を測定)	BDP = 2.6 MBps * 0.175 = 470 kB

BDP を計算できない場合は、次のガイドラインに従います。

- LAN 経由の一括転送では、バッファーサイズのデフォルト値は 128K バイトで十分です。
- ほとんどの WAN 配備では、受信バッファーサイズは 2M バイトの範囲内にすべきです。

注意 - TCP 受信バッファーサイズを増やすと、多くのネットワークアプリケーションのメモリーフットプリントも増加します。

netstat コマンドによるネットワークステータスのモニタリング

ネットワークのステータスとプロトコル統計を表示するには、netstat コマンドを使用します。TCP、SCTP、および UDP エンドポイントのステータス、およびルーティングテーブルとインタフェース情報を表形式で表示できます。

netstat コマンドは、指定したコマンド行オプションに応じて、さまざまな種類のネットワークデータを表示します。表示される情報はシステム管理に特に役立ちます。この章では、よく実行されるいくつかのタスクで使用するオプションについて説明します。詳細は、netstat(1M)のマニュアルページを参照してください。

このセクションには、次のトピックが含まれています。

- 25 ページの「アドレスタイプによる netstat 出力のフィルタリング」
- 26 ページの「ソケットのステータスの表示」
- 26 ページの「プロトコル別の統計情報の表示」
- 27 ページの「ネットワークインタフェースステータスの表示」
- 27 ページの「ネットワークインタフェースステータスの表示」
- 29 ページの「既知ルートのステータスの表示」

アドレスタイプによる netstat 出力のフィルタリング

netstat コマンドを使用して、IPv4 ネットワークと IPv6 ネットワークの両方のステータスを表示できます。表示するプロトコル情報を選択するには、/etc/default/inet_type ファイルに DEFAULT_IP 値を設定するか、-f オプションを使用します。DEFAULT_IP 値を永続的に設定することによって、netstat に IPv4 情報だけを表示させることができます。この設定をオーバーラ

イドするには、コマンド行から -f オプションを使用します。inet_type ファイルの詳細については、inet_type(4)のマニュアルページを参照してください。

-f オプションを使用して、inet、inet6、unix (内部通信に使用する Unix ドメインソケット用)、または sdp (Socket Description Protocol) の各引数を指定できます。

ソケットのステータスの表示

netstat コマンドを使用すると、ローカルホスト上にあるソケットのステータスを表示できます。通常のユーザーとして、複数の netstat コマンドオプションを指定できます。

次のようにして、接続ソケットのステータスを表示します。

```
% netstat
```

次のように、未接続のリスナーソケットも含め、すべてのソケットのステータスを表示します。

```
% netstat -a
```

例 1-3 接続済みソケットの表示

netstat コマンドの出力には、大量の統計が含まれます。次の例は、出力を IPv4 ソケットのみに制限する方法を示しています。

```
% 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
```

プロトコル別の統計情報の表示

netstat の -s オプションは、UDP、TCP、SCTP、Internet Control Message Protocol (ICMP)、および IP のプロトコルについて、プロトコル別の統計情報を表示します。

次のようにプロトコルのステータスを表示します。

```
% netstat -s
```

-P オプションを使用すると、netstat コマンドの出力をプロトコル別にフィルタリングできます。このオプションはトランスポートプロトコルに限定されません。

このオプションを使用して次の値を指定できます。

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

たとえば、次のように、netstat の出力を UDP プロトコルでフィルタリングします。

```
% netstat -s -P udp

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

例 1-4 TCP プロトコルのステータスの表示

次の例は、P -**オプション** を指定して TCP プロトコルの結果を表示する方法を示しています。

```
% 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
```

ネットワークインタフェースステータスの表示

netstat コマンドの i オプションを使用すると、ローカルシステムに構成されているネットワークインタフェースの状態を表示できます。このオプションを使用すると、各ネットワーク上で送受信しているパケット数がわかります。

次のように、ネットワーク上にあるインタフェースのステータスを表示します。

```
% netstat -i
```

次の例は、netstat コマンドにフィルタリングオプションを指定して、特定のインタフェースの出力を制限する方法を示しています。

```
% 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 ネットワークインタフェースステータスの表示

次の例は、ホストのインタフェースを通過する IPv4 と IPv6 のパケットのステータスを表示しています。たとえば、サーバーについて表示される入力パケットカウント (Ipkts) はクライアントがブートを試みるたびに増加しているのに、出力パケットカウント (Opkts) が変化しないことがあります。これは、サーバーがクライアントからのブート要求パケットを認識していることを意味します。しかし、サーバーはそれらのパケットに応答する方法を知りません。この混乱は、hosts または ethers データベース内の誤ったアドレスが原因である可能性があります。

入力パケットカウントが長時間にわたり変化しない場合は、マシンがパケットを認識していません。この場合は、上記と違って、ハードウェアの問題の可能性が高くなります。

```
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
```

ユーザーとプロセスの情報の表示

netstat コマンドには、新しい -u オプションがあります。このオプションは、特定のポートを使用しているシステム上のユーザーとプロセスに関する情報を提供します。netstat -u コマンドを使用すると、ユーザー、プロセス ID、およびネットワークエンドポイントを作成したプログラムまたはネットワークのエンドポイントを現在制御しているプログラムを表示できます。

netstat コマンドの出力をさらに整列するには、-n オプションと -u オプションをともに指定します。詳細な情報と例については、netstat(1M)のマニュアルページを参照してください。

既知ルートのステータスの表示

netstat コマンドの -r オプションを使用すると、ローカルホストのルーティングテーブルを表示できます。このテーブルには、ホストが認識しているすべてのルートのステータスが表示されます。

```
% netstat -r
```

例 1-6 netstat コマンドによるルーティングテーブルの出力の表示

次に、netstat - r コマンドの出力例を示します。

```
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
```

次の表では、netstat -r コマンドの出力に表示される情報について説明します。

パラメータ	説明
Destination	ルートの宛先エンドポイントであるホストを指定します。IPv6 ルーティングテーブルには、6to4 トンネルのエンドポイントの接頭辞 (2002:0a00:3010:2::/64) がルートの宛先エンドポイントとして示されていることに注目してください。
Destination/Mask	
Gateway	パケットの転送に使用するゲートウェイを指定します。
Flags	ルートの現在のステータスを示します。U フラグはルートが up 状態であること、G フラグはルートがゲートウェイへのものであることを示します。
Use	送信したパケットの数を示します。
Interface	転送元のエンドポイントである、ローカルホスト上の特定のインタフェースを示します。

netstat コマンドによる追加のネットワークステータスの表示

netstat コマンドにさまざまなコマンド行オプションを付けて使用すると、この章には記載されていないその他のネットワークステータスを表示できます。このコマンドには 10 の形式があり、各形式は、ネットワークサブシステムのさまざまな部分に関する異なる統計情報の表を生成します。

取得できる追加の統計情報の一部を次に示します。

netstat -g	マルチキャストグループメンバーシップを表示
netstat -P	net-to-media テーブルを表示: ARP と NDP のマッピング
netstat -m	STREAMS メモリー統計情報を表示
netstat -M	複数のルーティングテーブルを表示
netstat -D	DHCP リースのステータスを表示
netstat -d	宛先キャッシュテーブルを表示

詳細は、netstat(1M)のマニュアルページを参照してください。

netcat ユーティリティーを使用した TCP と UDP の管理の実行

netcat (nc) ユーティリティーを使用すると、TCP および UDP の管理に関するさまざまなタスクを実行できます。このコマンドは、IPv4 と IPv6 の両方のネットワークで使用できます。

netcat ユーティリティーを使用して次のタスクを実行できます。

- TCP 接続を開く
- UDP パケットを送信する
- 任意の TCP および UDP ポートを待機する
- ポートスキャンを実行する

telnet コマンドでは各エラーメッセージが個別に標準出力に書き出されますが、nc スクリプトによって生成されたエラーメッセージは 1 つの標準エラー出力にまとめられます (このほうが効率的です)。

netcat ユーティリティーでは新しい -M オプションがサポートされており、ソケットごとに Service Level Agreement (SLA) プロパティーを指定できます。適切なプロパティーを -M オプションとともに指定すると、そのソケットの MAC フローが作成されます。-M オプションは、たとえば、次のように使用します。

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

上の例に示したとおり、-M オプションを使用すると、*name=value* のペアをコンマで区切ったリストとして SLA プロパティーを指定できます。

一部のインストール方法では、netcat ソフトウェアパッケージがデフォルトではインストールされません。このパッケージがシステムにインストールされているかどうかは、次のようにして確認します。

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

パッケージがインストールされていない場合は、次のようにしてインストールします。

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

詳細は、netcat(1)のマニュアルページを参照してください。

ネットワークステータス表示の管理と記録

次のタスクでは、既知のネットワークコマンドを使用して、ネットワークのステータスをチェックする方法を説明します。

- 31 ページの「IP 関連コマンドの表示出力を制御する方法」
- 33 ページの「IPv4 ルーティングデーモンのアクションのロギング」
- 33 ページの「IPv6 近傍検索デーモンの活動をトレースする方法」

▼ IP 関連コマンドの表示出力を制御する方法

netstat コマンドの出力を制御すると、IPv4 情報だけを表示したり、IPv4 と IPv6 の両方の情報を表示したりできます。

1. **/etc/default/inet_type ファイルを作成します。**

2. **ネットワークの要求に基づいて、次のエントリのうちの 1 つを/etc/default/inet_type ファイルに追加します。**

   ■ **IPv4 情報だけを表示するには、次の変数を設定します。**

   ```
   DEFAULT_IP=IP_VERSION4
   ```

   ■ **IPv4 と IPv6 の両方の情報を表示するには、次のいずれかの変数を設定します。**

   ```
   DEFAULT_IP=BOTH

   DEFAULT_IP=IP_VERSION6
   ```

   詳細は、inet_type(4)マニュアルページを参照してください。

注記 - netstat コマンドの -f オプションは、inet_type ファイルに設定された値をオーバーライドします。

例 1-7 IPv4 情報と IPv6 情報を表示する出力の制御

次の例に、inet_type ファイルに DEFAULT_IP=BOTH または DEFAULT_IP=IP_VERSION6 変数を指定したときの出力を示します。

```
# 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
```

inet_type ファイルで、DEFAULT_IP=IP_VERSION4 変数を定義すると、次の出力が得られます。

```
# 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
```

IPv4 ルーティングデーモンのアクションのロギング

IPv4 ルーティングデーモン routed が誤動作している疑いがある場合は、このデーモンのアクティビティーをトレースするログを開始します。routed デーモンを起動すると、このログにすべてのパケット転送が記録されます。

次のように、ルーティングデーモンのアクションをトレースするログファイルを作成します。

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

注意 - ビジー状態のネットワークでは、このコマンドによりほとんど絶え間なく出力が生じることがあります。

次の例に、33 ページの「IPv4 ルーティングデーモンのアクションのロギング」の手順を実行したときに作成されたログの先頭部分を示します。

```
-- 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>
```

▼ IPv6 近傍検索デーモンの活動をトレースする方法

IPv6 の in.ndpd デーモンの動作が疑わしい場合、このデーモンの活動をトレースするログを開始できます。このトレースは、終了されるまで標準出力に表示されます。in.ndpd デーモンを起動すると、このトレースにはすべてのパケット転送が記録されます。

1. **in.ndpd デーモンのトレースを起動します。**

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

2. **Ctrl-C を押すといつでもトレースを終了できます。**

例 1-8 in.ndpd デーモンのトレース

次の例に、in.ndpd デーモンのトレースの開始部分を示します。

```
# /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
```

ping コマンドによるリモートホストのプローブ

ping コマンドを使用すると、システムがリモートホストと通信できるかどうかを確認できます。ping コマンドを実行すると、ICMP プロトコルは、指定されたホストにデータグラムを送って、応答を求めます。ICMP は、TCP/IP ネットワーク上のエラー処理を担当するプロトコルです。ping コマンドを使用すると、ホストと特定のリモートホスト間で IP パケットを交換できるかどうかを確認できます。

次に、ping コマンドの基本構文を示します。

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

ここで、*host* は、リモートホストの名前です。オプションの *timeout* 引数は、ping コマンドがリモートホストに到達しようと試行する秒数を示します。デフォルトは 20 秒です。詳細は、ping(1M)のマニュアルページを参照してください。

IPv6 をサポートするための ping コマンドの変更

ping コマンドは、IPv4 プロトコルと IPv6 プロトコルの両方で、ターゲットホストをプローブするために使用できます。プロトコル選択は、指定のターゲットホストのネームサーバーが戻すアドレスに依存します。デフォルトでネームサーバーによってターゲットホストの IPv6 アドレスが返

されると、ping コマンドは IPv6 プロトコルを使用します。サーバーが IPv4 アドレスだけを戻すと、ping コマンドは IPv4 プロトコルを使用します。この動作は、-A オプションを使用してプロトコルを指定することでオーバーライドできます。

詳細は、ping(1M)のマニュアルページを参照してください。

リモートホストが到達可能かどうかの確認

リモートホストが到達可能かどうかを確認するには、次のように svcs コマンドを使用します。

```
% ping hostname
```

ホストが ICMP 伝送を受け入れると、次のメッセージが表示されます。

```
hostname is alive
```

このメッセージは、ホストが ICMP 要求に応答したことを示します。しかし、ホストがダウン状態にあるかまたは ICMP パケットを受け取れなかった場合、またはホストとリモートホストとの間でルーティング障害が発生している場合は、ping コマンドから次の応答が返されます。

```
no answer from hostname
```

ホストとリモートホスト間でパケットが失われているかどうかの確認

パケットが失われると、破棄されたデータを再送信するために余分な時間がかかるため、ネットワークの接続速度が遅くなったように感じます。ping コマンドの -s オプションを使用すると、ホストとリモートホストの間でパケットが失われていないか確認できます。

```
% ping -s hostname
```

次の例では、ping -s *hostname* コマンドが、割り込み文字が送信されるかタイムアウトが発生するまで、指定されたホストにパケットを継続的に送信しています。

```
% 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
```

パケットロス統計情報は、ホストがパケットを破棄したかどうかを示します。ping コマンドがパケットロスが発生したことを示している場合、ipadm コマンドと netstat コマンドを使用して、ネットワークのステータスをチェックします。詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「IP インタフェースとアドレスのモニタリング」および25 ページの「netstat コマンドによるネットワークステータスのモニタリング」を参照してください。

traceroute コマンドによるルーティング情報の表示

traceroute コマンドは、IP パケットが通るリモートシステムまでのルートをトレースします。traceroute コマンドを使用すると、ルーティングの誤構成やルーティングパスの異常を発見できます。特定のホストが到達不可能な場合には、traceroute を使用して、パケットがリモートホストに到達するまでにたどる経路と障害が起きている可能性がある場所を調べることができます。

また、traceroute コマンドは、経路に沿った各ゲートウェイのターゲットホストとの間の往復時間も表示します。この情報は、2 つのホスト間のどこでトラフィックが遅くなっているのかを分析する際に利用できます。

traceroute コマンドの詳細は、traceroute(1M)のマニュアルページを参照してください。

このセクションには、次のトピックが含まれています。

IPv6 をサポートするための traceroute コマンドの変更

traceroute コマンドは、特定のホストへの IPv4 ルートと IPv6 ルートの両方をトレースするために使用できます。プロトコルの観点から見ると、traceroute は、ping と同じアルゴリズムを使

用します。この動作をオーバーライドするには、-A コマンド行オプションを使用します。マルチホームホストの各アドレスまでのそれぞれのルートは -a コマンド行オプションを使用してトレースできます。

リモートホストまでのルートの発見

次のように traceroute コマンドを使用すると、リモートシステムまでのルートを発見できます。

```
% traceroute destination-hostname
```

次の traceroute コマンドからの出力は、パケットがローカルシステム nearhost からリモートシステム farhost までに通る 7 ホップのパスを示しています。この出力は、パケットが各ホップを通過するのに要する時間も示しています。

```
% 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
```

すべてのルートのトレース

traceroute コマンドに -a オプションを付けてローカルシステム上で使用すると、すべてのルートをトレースできます。

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

次の例は、デュアルスタックホストまでの考えられるすべてのルートを表示しています。

```
% 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 *
```

TShark および Wireshark アナライザによるネットワークトラフィックの解析

TShark は、稼働中のネットワークからパケットデータをキャプチャーしたり、以前に保存した*キャプチャーファイル*からパケットを読み取ったりするコマンド行ネットワークトラフィックアナライザで、パケットをデコードされた形式で標準出力に出力したり、パケットをファイルに書き出すことができます。オプションを何も指定しないと、TShark は `tcpdump` コマンドと同じように動作し、ライブキャプチャーファイルにも同じ形式 `libpcap` を使用します。また、TShark は、Wireshark でサポートされているのと同じキャプチャーファイルの検出、読み取り、書き込みを実行できます。

Wireshark は、グラフィカルユーザーインタフェース (GUI) を採用したサードパーティー製のネットワークプロトコルアナライザで、対話形式によるネットワークトラフィックのダンプと解析に使用されます。`snoop` コマンドと同様、Wireshark を使用すると、稼働中のネットワーク上のパケットデータ、または以前にキャプチャーファイルに保存したパケットデータを参照できます。Wireshark は、デフォルトでは、ファイルキャプチャーに (`tcpdump` ユーティリティーやその他の類似ツールでも使用されている) `libpcap` 形式を使用します。Wireshark を使用する主な利点は、`libpcap` 形式以外にも、複数のファイル形式の読み込みとインポートが可能な点です。

TShark と Wireshark のどちらも、次に示すいくつかの固有の機能を備えています。

- TCP 通信のすべてのパケットの組み立て、ASCII、EBCDIC、または 16 進形式での通信データの表示が可能
- ほかのネットワークプロトコルアナライザよりも多くのフィルタ可能なフィールド
- ほかのネットワークプロトコルアナライザよりも多機能のフィルタ作成用構文

TShark および Wireshark を Oracle Solaris システム上で使用するには、まず、ソフトウェアパッケージがインストールされているかどうかを確認し、必要に応じて、次のようにそれらをインストールします。

```
# pkg install tshark
```

```
# pkg install wireshark
```

詳細は、`tshark`(1)および`wireshark`(1)のマニュアルページを参照してください。

Wireshark のドキュメント (http://www.wireshark.org/) も参照してください。

snoop コマンドによるパケット転送のモニタリング

snoop コマンドを使用すると、ネットワークトラフィックをモニタリングできます。snoop コマンドは、ネットワークパケットをキャプチャーし、指定された形式でその内容を表示します。パケットについては、受信してすぐに表示することも、ファイルに保存することも可能です。snoop コマンドが中間ファイルに書き込む場合、トレースのビジー状態でパケットロスが発生することはほとんどなく、snoop コマンドを使用して解釈されます。

デフォルトのインタフェースにおいて、パケットをプロミスキャスモードでキャプチャーするには、Network Management 権限プロファイルを取得するか、または root 役割になる必要があります。サマリー形式では、snoop コマンドは最上位レベルのプロトコルに関連するデータだけを表示します。たとえば NFS パケットでは、NFS 情報のみが表示されます。ベースとなるリモートプロシージャコール (RPC)、UDP、IP、および Ethernet のフレーム情報は非表示になりますが、verbose (詳細表示) オプションのいずれかを選択すると表示できます。

頻繁かつ定期的に snoop を使用して、システムの正常な動作を把握してください。snoop コマンドの詳細は、snoop(1M)のマニュアルページを参照してください。

このセクションには、次のトピックが含まれています。

- 40 ページの「すべてのインタフェースからのパケットをチェックする方法」
- 40 ページの「IPMP グループからのパケットをチェックする方法」
- 41 ページの「snoop の出力をファイルにキャプチャーする方法」
- 42 ページの「IPv4 サーバー/クライアント間のパケットを確認する方法」
- 42 ページの「IPv6 ネットワークトラフィックのモニタリング」
- 43 ページの「IP 層デバイスを使用したパケットモニタリング」

IPv6 をサポートするための snoop コマンドの変更

snoop コマンドは、IPv4 パケットと IPv6 パケットの両方をキャプチャーできます。このコマンドは、IPv6 ヘッダー、IPv6 拡張ヘッダー、ICMPv6 ヘッダー、近傍検索 (ND) プロトコルデータを表示できます。デフォルトで、snoop コマンドは、IPv4 パケットと IPv6 パケットの両方を表示します。ip または ip6 のプロトコルキーワードを指定した場合、このコマンドは IPv4 パケットまたは IPv6 パケットだけを表示します。IPv6 フィルタオプションでは、すべてのパケットをフィルタの対象にでき (IPv4 と IPv6 の両方)、IPv6 パケットだけが表示されます。42 ページの「IPv6 ネットワークトラフィックのモニタリング」および snoop(1M)のマニュアルページを参照してください。

▼ すべてのインタフェースからのパケットをチェックする方法

1. **システムに接続されているインタフェースについての情報を出力します。**

   ```
   # ipadm show-if
   ```

   snoop コマンドは通常、最初の非ループバックデバイス (通常はプライマリネットワークインタフェース) を使用します。

2. **snoop を引数なしで入力してパケットのキャプチャーを開始します。**

   ```
   # snoop
   ```

3. **Ctrl + C キーを押してプロセスを停止します。**

例 1-9 基本的な snoop 出力の表示

次の例は、デュアルスタックホストの基本的な snoop コマンドの出力を示しています。

```
# 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)
```

上記の出力では、キャプチャーされたパケットは、DNS 問合せと応答、ローカルルーターからの定期的なアドレス解決プロトコル (ARP) パケット、IPv6 リンクローカルアドレスの in.ripngd デーモンへの広告を示しています。

▼ IPMP グループからのパケットをチェックする方法

Oracle Solaris 10 では、IPMP グループからのパケットをキャプチャーする場合、その IPMP グループの各インタフェース上で snoop コマンドを手動で実行する必要があります。Oracle Solaris 11 リリースでは、snoop コマンドに -I オプションを指定して使用することで、IPMP グループに属するすべてのインタフェースからのパケットをキャプチャーできます。こうすると、出力は単一の出力ストリームに統合されます。

snoop コマンドは通常、最初の非ループバックデバイス (通常はプライマリネットワークインタフェース) を使用します。ただし、-I オプションを指定すると、snoop コマンドは、(/dev/ipnet/* にある) IP インタフェース (ipadm show-if コマンドによって表示される) を使用します。このオプションを使用すると、ループバックトラフィックおよびその他の IP 専用構造体をスヌーピングすることもできます。-d オプションは、dladm show-link コマンドの出力に表示されるデータリンクインタフェースを使用します。

1. **システムに接続されているインタフェースについての情報を出力します。**

   ```
   # ipadm show-if
   ```

2. **指定した IPMP グループに対するパケットキャプチャーを開始します。**

   # snoop -I *ipmp-group*

3. **Ctrl + C キーを押してプロセスを停止します。**

▼ snoop の出力をファイルにキャプチャーする方法

1. **snoop セッションをファイルに取り込みます。次に例を示します。**

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

   この例では、30 個のパケットが /tmp/cap というファイルにキャプチャーされています。ファイルは、十分なディスク領域のあるどのディレクトリにでも格納できます。取り込んだパケットの数はコマンド行に表示され、Ctrl-C を押せばいつでも終了できます。

   snoop コマンドによってホストマシン上に高いネットワーク負荷が生じるので、結果に誤差が生じる場合があります。実際の結果を表示するには、第 3 のシステムから snoop コマンドを実行します。

2. **snoop 出力キャプチャーファイルを検査します。**

   # snoop -i *filename*

例 1-10 snoop 出力キャプチャーの表示

次の出力は、snoop -i コマンドから出力として返されるキャプチャーを示しています。

```
# 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
```

▼ IPv4 サーバー/クライアント間のパケットを確認する方法

1. **クライアントまたはサーバーのいずれかに接続されたハブ (またはスイッチ上のミラーポート) から切り離して、snoop システムを確立します。**

   この第 3 のシステム (snoop システム) はサーバーとクライアント間のすべてのトラフィックを監視するので、snoop のトレースには実際のネットワーク上の状態が反映されます。

2. **snoop を適切なオプション付きで入力して、出力をファイルに保存します。**

3. **出力を検査および解釈します。**

   snoop キャプチャーファイルの詳細については、RFC 1761, Snoop Version 2 Packet Capture File Format (http://www.rfc-editor.org/rfc/rfc1761.txt) を参照してください。

IPv6 ネットワークトラフィックのモニタリング

次のように snoop コマンドを使用して、IPv6 パケットをキャプチャーできます。

```
# snoop ip6
```

次に、ノード上で snoop ip6 コマンドを実行したときに表示される典型的な出力の例を示します。

```
# 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
```

snoop コマンドの詳細については、snoop(1M)のマニュアルページを参照してください。

IP 層デバイスを使用したパケットモニタリング

IP 層の可観測性を高めるため、IP 層デバイスが Oracle Solaris に導入されています。これらのデバイスを使用すると、システムのネットワークインタフェースに関連付けられたアドレスを持つすべてのパケットにアクセスできます。アドレスには、ローカルアドレスのほか、非ループバックインタフェースまたは論理インタフェースにホストされたアドレスも含まれます。監視可能なトラフィックには、IPv4、IPv6 のどちらのアドレスが含まれていてもかまいません。したがって、システムに向かうすべてのトラフィックをモニターできます。トラフィックには、ループバック IP トラフィック、リモートマシンからのパケット、システムから送信されるパケット、またはすべての転送トラフィックが含まれる場合があります。

Oracle Solaris の大域ゾーンの管理者は IP 層デバイスを使用することで、ゾーン間のトラフィックやゾーン内のトラフィックをモニターできます。非大域ゾーンの管理者も、そのゾーンによって送受信されるトラフィックを監視できます。

IP 層のトラフィックをモニターするには、snoop コマンドに新しい -I オプションを指定して使用します。このオプションは、コマンドが、ベースとなるリンク層デバイスではなく新しい IP 層デバイスを使用してトラフィックデータを表示することを指定します。

▼ IP 層でパケットをチェックする方法

1. **(オプション) 必要であれば、システムに接続されているインタフェースについての情報を出力します。**

   ```
   # ipadm show-if
   ```

2. **特定のインタフェースの IP トラフィックを取得します。**

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

パケットをチェックする方法

以降のすべての例は次のシステム構成に基づいています。

```
# 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
```

2 つのゾーン sandbox と toybox が次の IP アドレスを使用しているとします。

- sandbox – 172.0.0.3
- toybox – 172.0.0.1

snoop -I コマンドは、システム上のさまざまなインタフェースに対して使用できます。表示されるパケット情報は、ユーザーが大域ゾーン、非大域ゾーンのいずれの管理者であるかに依存します。

例 1-11 ループバックインタフェース上のトラフィックの監視

次に、ループバックインタフェースに対する netstat コマンドの出力例を示します。

```
# 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)
```

詳細な出力を生成するには、-v オプションを使用します。

```
# 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
...
```

IP 層でのパケット監視を支援するために、監視対象パケットの前に付く新しいヘッダー ipnet が導入されています。発信元 ID と着信先 ID の両方が示されます。ID が 0 の場合は、トラフィックが大域ゾーンから生成されていることを示します。

例 1-12 ローカルゾーン内の net0 デバイスにおけるパケットフローの監視

次の例は、システム内の異なるゾーン内で発生したトラフィックを示しています。ローカルでほかのゾーンに配信されるパケットも含め、net0 の IP アドレスに関連するすべてのパケットを表示

できます。詳細出力を生成すれば、パケットのフローに関連するゾーンを確認することもできます。

```
# 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:
```

上記の出力で、ipnet ヘッダーは、パケットが大域ゾーン (ID 0) から sandbox (ID 1) に送信されていることを示しています。

例 1-13 ゾーンを特定したネットワークトラフィックの監視

次の例は、ゾーンを特定してネットワークトラフィックを監視する方法 (複数のゾーンを持つシステムにはきわめて有用) を示しています。現在のところ、ゾーンを特定するには、ゾーン ID を使用するしかありません。ゾーン名を指定して snoop コマンドを使用することはできません。

```
# 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
```

ipstat および tcpstat コマンドを使用したネットワークトラフィックの監視

このリリースでは、サーバー上でさまざまなタイプのネットワークトラフィックを監視する 2 つのコマンド ipstat と tcpstat が導入されています。

ipstat コマンドは、コマンド構文で指定された選択した出力モードとソート順序に基づいて、サーバー上で IP トラフィックに関する統計情報を収集および報告するために使用します。このコマンドを使用すると、送信元、宛先、上位層プロトコル、およびインタフェースごとに集計された IP 層のネットワークトラフィックを監視できます。このコマンドは、特定のサーバーとそのほかのサーバーの間のトラフィック量を監視する必要があるときに使用します。

tcpstat コマンドは、コマンド構文で指定された選択した出力モードとソート順序に基づいて、サーバー上で TCP および UDP トラフィックに関する統計情報を収集および報告するために使用します。このコマンドを使用すると、トランスポート層、具体的には TCP と UDP のネット

ワークトラフィックを監視できます。送信元および宛先 IP アドレスに加え、送信元および宛先の TCP または UDP ポート、トラフィックを送受信しているプロセスの PID、プロセスが動作しているゾーンの名前を監視できます。

次に、tcpstat コマンドの使い方の例をいくつか挙げます。

- サーバー上でもっともトラフィック量の多い TCP および UDP の送信元を特定する。
- 特定のプロセスによって生成されているトラフィックを調査する。
- 特定のゾーンで生成されているトラフィックを調査する。
- ローカルポートにバインドされているプロセスを確認する。

注記 - 上記の使い方がすべてではありません。tcpstat コマンドにはほかにも使い方があります。詳細は、tcpstat(1M)のマニュアルページを参照してください。

---

ipstat および tcpstat コマンドを使用するには、次のいずれかの権限が必要です。

- root 役割になります。
- dtrace_kernel 権限が明示的に割り当てられていること
- Network Management または Network Observability 権利プロファイルが割り当てられていること

次の例に、この 2 つのコマンドを使用してネットワークトラフィックを監視するさまざまな方法を示します。詳細は、tcpstat(1M)およびipstat(1M)のマニュアルページを参照してください。

次の例は、-c オプションを指定して実行したときの ipstat コマンドからの出力を示しています。-c オプションを使用すると、前のレポートを上書きすることなく、前のレポート以降の新しいレポートを出力できます。この例の数字 3 は、データを表示する間隔を指定します (これは、ipstat 3 としてコマンドを起動した場合と同じです)。

```
# 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
```

比較のために、次の例に、tcpstat コマンドを -c オプション付きで使用した場合の出力を示します。

```
# 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
```

さらに次の例は、ipstat および tcpstat コマンドを使用して、ネットワーク上のトラフィックを監視するほかの方法を示しています。

例 1-14 ipstat コマンドを使用した 5 つのもっともアクティブな IP トラフィックフローの監視

次の例は、もっともアクティブな 5 つの IP トラフィックフローを報告しています。-l *nlines* オプションには、報告ごとの出力データ行数を指定します。

```
# 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 ipstat コマンドを使用したタイムスタンプの表示

次の例は、最上位の IP トラフィックを標準日付書式 (-d d) のタイムスタンプ付きで報告しています。タイムスタンプが、秒または Unix 時間 (-d u) で出力されるように指定できます。間隔は 10 秒に設定されます。

```
# 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 tcpstat コマンドを使用した 5 つのもっともアクティブなトラフィックフローの監視

次の例は、サーバーでもっともアクティブな 5 つの TCP トラフィックフローを報告しています。

```
# 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 tcpstat コマンドを使用したタイムスタンプ情報の表示

次の例は、tcpstat コマンドを使用して、サーバー上の TCP ネットワークトラフィックのタイムスタンプ情報を標準の日付書式で表示しています。

```
# 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 IPMP の管理について

IP ネットワークマルチパス (IPMP) は、複数の IP インタフェースを単一の論理インタフェースにグループ化することを可能にするレイヤー 3 (L3) テクノロジです。障害検出、透過的なアクセスフェイルオーバー、パケット負荷分散などの機能を提供する IPMP は、システムに対してネットワークを常に使用可能に保つことで、ネットワークパフォーマンスを向上させます。

この章の内容は、次のとおりです。

注記 - この章と第3章「IPMP の管理」では、*インタフェース*という用語はすべて、明確に *IP インタフェース*を意味しています。ネットワークインタフェースカード (NIC) のように、修飾語がこの用語の異なる使用法を明示的に示す場合を除き、この用語は常に、IP 層で構成されたインタフェースを指します。

IPMP の新機能

IPMP 構成の変更

このリリースの IPMP は、Oracle Solaris 10 とは異なる方法で動作します。1 つの重要な変更は、IP インタフェースが 1 つの*仮想* IP インタフェース (たとえば、`ipmp0`) にグループ化されるようになったことです。仮想 IP インタフェースはすべてのデータ IP アドレスを処理するのに

対して、プローブベースの障害検出に使用される検査用アドレスは net0 などのベースとなるインタフェースに割り当てられます。詳細は、59 ページの「IPMP の動作方法」を参照してください。

既存の IPMP 構成から新しい IPMP モデルに移行する際には、次の一般的なワークフローを参照してください。

1. IPMP を構成する前に、DefaultFixed プロファイルがシステムで有効になっていることを確認します。『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「プロファイルを有効および無効にする」を参照してください。
2. SPARC ベースのシステムでの MAC アドレスが一意であることを確認します。『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「各インタフェースの MAC アドレスが一意であることを確認する方法」を参照してください。
3. dladm コマンドは、データリンクを構成するために使用します。IPMP 構成内で同じ物理ネットワークデバイスを使用するには、最初に各デバイスインスタンスに関連付けられたデータリンクを識別する必要があります。

```
# 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. これまで、IPMP の構成で、e1000g0 と e1000g1 を使用していた場合は、今後、net2 と net3 を使用できるようになります。データリンクは、物理リンクのみでなくアグリゲーション、VLAN、VNIC などをベースにすることもできます。詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「システムのデータリンクの表示」を参照してください。
5. ipadm コマンドを使用して、次のタスクを実行します。
   - ネットワーク層を構成します。
   - IP インタフェースを作成します。
   - IP インタフェースを IPMP グループに追加します。
   - データアドレスを IPMP グループに追加します。

このリリースにおけるネットワーク管理コマンドの変更点の詳細については、『Oracle Solaris 10 から Oracle Solaris 11.2 への移行 』の「ネットワーク管理コマンドの変更」を参照してください。

Oracle Solaris の IPMP サポート

IPMP は次のサポートを提供します。

- IPMP により、複数の IP インタフェースを、IPMP グループと呼ばれる単一のグループに構成できます。IPMP グループは、そのベースとなる複数の IP インタフェースを含め、全体で単一の *IPMP インタフェース*として表されます。このインタフェースは、ネットワークスタックの IP 層のほかのインタフェースとまったく同様に扱われます。IP 管理タスク、ルーティングテーブル、アドレス解決プロトコル (ARP) テーブル、ファイアウォール規則、およびその他の IP 関連手順はすべて、IPMP インタフェースを参照することで IPMP グループを操作します。
- システムは、ベースとなるアクティブインタフェース間でデータアドレスの分配を処理します。IPMP グループを作成すると、データアドレスは*アドレスプール*として IPMP インタフェースに属します。続いてカーネルが自動的かつランダムに、グループのデータアドレスとベースとなるアクティブインタフェースをバインドします。
- IPMP グループに関する情報を取得するには、主に `ipmpstat` コマンドを使用します。このコマンドは、グループのベースとなる IP インタフェース、検査用アドレスとデータアドレス、使用される障害検出のタイプ、故障したインタフェースなど、IPMP 構成のあらゆる側面についての情報を提供します。`ipmpstat` の機能、使用できるオプション、および各オプションが生成する出力についてはすべて、94 ページの「IPMP 情報のモニタリング」で説明しています。
- IPMP インタフェースにカスタマイズされた名前を割り当てて、IPMP グループをより簡単に識別できます。73 ページの「IPMP グループの構成」を参照してください。

IPMP を使用する利点

インタフェースの故障や、インタフェースが保守のためにオフラインになっている場合など、さまざまな要因によりインタフェースが使用不可能になる可能性があります。IPMP を使用しないと、その使用不可能になったインタフェースに関連付けられたどの IP アドレスを使用しても、システムと通信できなくなります。さらに、それらの IP アドレスを使用する既存の接続が切断されます。

IPMP を使用すると、複数の IP インタフェースを 1 つの *IPMP グループ*に構成できます。このグループは、ネットワークトラフィックを送受信するデータアドレス付きの IP インタフェースのように機能します。グループ内のベースとなるインタフェースの 1 つで障害が発生すると、グルー

プ内の残りのアクティブなベースとなるインタフェースの間でデータアドレスが再分配されます。したがって、インタフェースの 1 つが故障しても、グループはネットワークの接続性を維持します。IPMP では、グループで最低 1 つのインタフェースが使用可能であれば、ネットワーク接続を常に使用できます。

IPMP は、IPMP グループ内のインタフェースセット全体にアウトバウンドネットワークトラフィックを自動的に分散させることにより、全体的なネットワークパフォーマンスを向上させます。このプロセスは、*アウトバウンド負荷分散*と呼ばれます。システムはさらに、アプリケーションによって発信元 IP アドレスが指定されなかったパケットに対して発信元アドレス選択を実行することにより、インバウンド負荷分散も間接的に制御します。ただし、アプリケーションが発信元 IP アドレスを明示的に選択した場合は、システムはその発信元アドレスを変更しません。

IPMP がインバウンドおよびアウトバウンドの負荷分散のために実施するポリシーに関する次の重要な情報を確認してください。

- オンリンク IP アドレスの場合、IPMP は、その IP アドレスに到達するための単一のアクティブな IP インタフェースをランダムに選択します。特定のオンリンク IP アドレスに対して複数の別個の接続が存在する場合は、それらすべての接続で同一のアウトバウンド IP インタフェースが使用されます。さらに、IP インタフェースが時間とともに変化する場合、その IP アドレスへのすべての接続が影響を受けます。
- オフリンク IP アドレスの場合、IPMP は、そのオフリンク IP アドレスに到達する際に経由するオンリンク IP ルーターの IP アドレスに到達するための単一の IP インタフェースをランダムに選択します。このポリシーは、事実上、特定の IPMP グループのすべてのオフリンク IP アドレスが単一の IP インタフェースを使用することを意味します。

注記 - 現在のインバウンドおよびアウトバウンドの負荷分散ポリシーは変更される可能性があります。

リンクアグリゲーションは IPMP と同様の機能を実行して、ネットワークのパフォーマンスと可用性を向上させます。この 2 つのテクノロジの比較については、『Oracle Solaris 11.2 でのネットワークデータリンクの管理 』の付録 A「リンクアグリゲーションと IPMP: 機能比較」を参照してください。

IPMP を使用するための規則

IPMP グループの構成はシステム構成によって決まります。

次の IPMP 構成規則に従ってください。

1. 同じ LAN 上の複数の IP インタフェースは、1 つの IPMP グループに構成される必要があります。LAN は、同じリンク層ブロードキャストドメインに属するノードから成る VLAN や有線と無線の両方のローカルネットワークなど、さまざまなローカルネットワーク構成を広く指します。

注記 - 同じリンク層 (L2) ブロードキャストドメイン上の複数の IPMP グループはサポートされていません。通常、L2 ブロードキャストドメインは特定のサブネットにマップされます。したがって、サブネットごとに IPMP グループを 1 つだけ構成する必要があります。このルールには例外があり、たとえば Oracle によって提供される特定のエンジニアドシステムなどは適用外となります。詳細は、Oracle サポート担当者にお問い合わせください。

---

2. IPMP グループのベースとなる IP インタフェースが異なる LAN にまたがってはいけません。

   たとえば、3 つのインタフェースを備えたシステムが 2 つの別個の LAN に接続されているとします。一方の LAN に 2 つの IP インタフェースが接続し、他方の LAN に単一の IP インタフェースが接続します。この場合、最初の規則により、1 つ目の LAN に接続する 2 つの IP インタフェースは 1 つの IPMP グループとして構成される必要があります。2 番目の規則のため、2 つ目の LAN に接続する単一の IP インタフェースがその IPMP グループのメンバーになることはできません。単一の IP インタフェースでは、IPMP 構成は必須ではありません。ただし、その単一のインタフェースの可用性をモニターするために、そのインタフェースを 1 つの IPMP グループに構成してもかまいません。単一インタフェースの IPMP 構成の詳細については、57 ページの「IPMP インタフェース構成のタイプ」を参照してください。

   別のケースとして、1 つ目の LAN へのリンクが 3 つの IP インタフェースから構成され、もう 1 つのリンクが 2 つのインタフェースから構成される場合を考えます。この設定は 2 つの IPMP グループの構成を必要とします (1 つ目の LAN に接続する 3 つのインタフェースから成るグループと、2 つ目の LAN に接続する 2 つのインタフェースから成るグループ)。
3. 同じグループのすべてのインタフェースは、同じ順番で構成された STREAMS モジュールを持っていなければなりません。IPMP グループを計画する際には、まず、今後 IPMP グループに含められるすべてのインタフェース上の STREAMS モジュールの順番をチェックし、各インタフェースのモジュールを IPMP グループの標準の順番でプッシュします。STREAMS モジュールの一覧を出力するには、`ifconfig` *interface* `modlist` コマンドを使用します。たとえば、`net0` インタフェースの `ifconfig` 出力は次のようになります。

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

上の出力が示しているように、インタフェースは通常、IP モジュールのすぐ下のネットワークドライバとして存在します。これらのインタフェースでは、追加の構成は必要ありません。ただし、特定のテクノロジは、IP モジュールとネットワークドライバの間に STREAMS モジュールとしてプッシュされます。STREAMS モジュールがステートフルである場合、グループ内のすべてのインタフェースに同じモジュールをプッシュしている場合でも、フェイルオーバーで予期しない動作が発生する可能性があります。ただし、IPMP グループのすべてのインタフェースに同じ順番で転送している場合は、処理状態を把握できない STREAMS モジュールを使用できます。

次の例は、各インタフェースのモジュールを IPMP グループの標準の順番でプッシュするときに使用するコマンドを示しています。

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

IPMP グループを計画する際の詳細な段階については、74 ページの「IPMP グループの計画を立てる方法」を参照してください。

IPMP のコンポーネント

IPMP のソフトウェアコンポーネントは次のとおりです。

- **マルチパスデーモン** `in.mpathd` - インタフェースの障害や修復を検出します。ベースとなるインタフェースで検査用アドレスが構成されている場合、このデーモンは、リンクベースの障害検出とプローブベースの障害検出の両方を実行します。このデーモンは、使用される障害検出手法のタイプに応じて、インタフェース上で適切なフラグを設定または解除し、そのインタフェースが故障しているかどうかや修復されたかどうかを示します。オプションとして、IPMP グループに属するように構成されていないインタフェースも含め、すべてのインタフェースの可用性をモニターするようにこのデーモンを構成することもできます。障害検出については、66 ページの「IPMP での障害検出」を参照してください。

  `in.mpathd` デーモンは、IPMP グループ内のアクティブインタフェースの指定も制御します。このデーモンは、IPMP グループの作成時に最初に構成されたのと同じ数のアクティブインタフェースを維持しようとします。したがって、`in.mpathd` は、ベースとなるインタフェースを必要に応じてアクティブ化または非アクティブ化して、管理者が構成したポリシーとの一貫性

を保ちます。in.mpathd デーモンがベースとなるインタフェースのアクティブ化を管理する方法の詳細については、59 ページの「IPMP の動作方法」を参照してください。このデーモンの詳細については、in.mpathd(1M)のマニュアルページを参照してください。

- **IP カーネルモジュール** - IPMP グループ内で使用可能な一連の IP データアドレスをグループ内で使用可能な一連のベースとなる IP インタフェースに分配することで、アウトバウンド負荷分散を管理します。さらにこのモジュールは、発信元アドレス選択を実行してインバウンド負荷分散を管理します。このモジュールのどちらの役割も、ネットワークトラフィックのパフォーマンスを改善します。
- **IPMP 構成ファイル** (/etc/default/mpathd) - デーモンの動作を定義します。

  次のパラメータを設定するには、このファイルをカスタマイズします。

  - プローブベースの障害検出の実行中にプローブされるターゲットインタフェース
  - 障害を検出するためにターゲットをプローブする時間
  - 障害のあるインタフェースが修復されたあと、そのインタフェースに付けられるステータス
  - モニターする IP インタフェースの範囲 (IPMP グループに属するように構成されていない、システム内の IP インタフェースも含めるかどうか)

  構成ファイルの変更方法の詳細については、92 ページの「IPMP デーモンの動作を構成する方法」を参照してください。
- **ipmpstat コマンド** - IPMP の全体としてのステータスに関するさまざまなタイプの情報を提供します。さらにこのツールは、各 IPMP グループのベースとなる IP インタフェースに関するその他の情報や、グループで構成されているデータアドレスや検査用アドレスも表示します。このコマンドの詳細は、94 ページの「IPMP 情報のモニタリング」および ipmpstat(1M)のマニュアルページを参照してください。

IPMP インタフェース構成のタイプ

IPMP 構成は、通常同じ LAN に接続された同じシステムの複数の物理インタフェースで構成されます。

これらのインタフェースは、次のいずれかの構成の IPMP グループに属することができます。

- **アクティブ - アクティブ構成** - ベースとなるインタフェースのすべてがアクティブである IPMP グループ。「*アクティブインタフェース*」とは、IPMP グループによって現時点で使用可能な IP インタフェースのことです。

注記 - デフォルトでは、あるベースとなるインタフェースを IPMP グループの一部になるように構成すると、そのインタフェースはアクティブになります。

- **アクティブ - スタンバイ構成** – 少なくとも 1 つのインタフェースが*スタンバイインタフェース*として管理上構成されている IPMP グループ。スタンバイインタフェースはアイドル状態になっていますが、その構成方法に応じて、可用性を追跡するためにマルチパスデーモンによってモニターされます。インタフェースによってリンク障害通知がサポートされている場合は、リンクベースの障害検出が使用されます。インタフェースで検査用アドレスが構成されている場合は、プローブベースの障害検出も使用されます。いずれかのアクティブインタフェースが故障すると、スタンバイインタフェースが必要に応じて自動的に配備されます。1 つの IPMP グループには、スタンバイインタフェースを必要な数だけ構成できます。

単一のインタフェースを、そのインタフェースだけの IPMP グループとして構成することもできます。単一インタフェース IPMP グループは、複数のインタフェースを持つ IPMP グループと同じように動作します。ただし、この IPMP 構成は、ネットワークトラフィックの高可用性を提供しません。ベースとなるインタフェースが故障すると、システムはトラフィックを送受信する機能をすべて失います。単一インタフェースの IPMP グループを構成する目的は、障害検出を使用してインタフェースの可用性をモニターすることです。インタフェースで検査用アドレスを構成することにより、プローブベースの障害検出を使用してマルチパスデーモンでそのインタフェースを追跡できます。

単一インタフェースの IPMP グループ構成は通常、Oracle Solaris Cluster ソフトウェアなど、より幅広いフェイルオーバー機能を備えたほかのテクノロジとともに使用されます。システムは引き続き、ベースとなるインタフェースのステータスをモニターできますが、Oracle Solaris Cluster ソフトウェアは、障害発生時にネットワークの可用性を保証するための機能を提供します。Oracle Solaris Cluster ソフトウェアの詳細は、『Oracle Solaris Cluster Concepts Guide 』を参照してください。

ベースとなるインタフェースが削除されたグループなど、ベースとなるインタフェースを持たない IPMP グループも存在できます。この IPMP グループは破棄はされませんが、このグループを使用してトラフィックを送受信することはできません。このグループでベースとなるインタフェースがオンラインになると、それらのインタフェースに IPMP インタフェースのデータアドレスが割り当てられ、システムがネットワークトラフィックのホスティングを再開します。

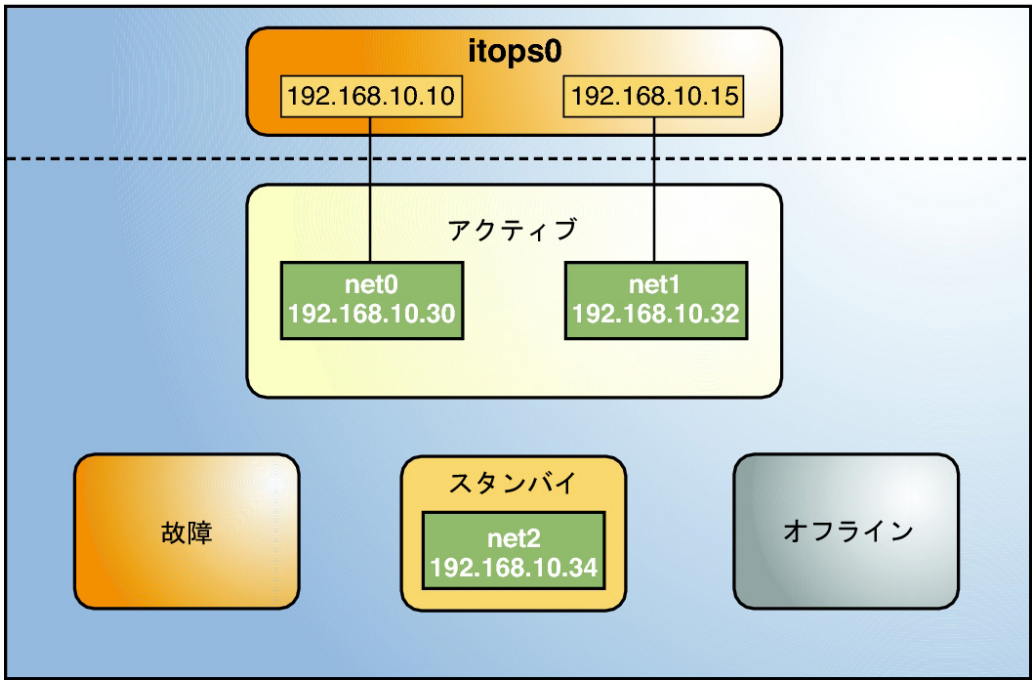
IPMP の動作方法

IPMP は、IPMP グループの作成時に構成されたアクティブインタフェースとスタンバイインタフェースの元の数を保持しようとすることによって、ネットワークの可用性を維持します。

グループ内の特定のベースとなる IP インタフェースの可用性を判定するための IPMP 障害検出は、リンクベースまたはプローブベース、あるいはその両方にすることができます。あるベースとなるインタフェースが故障したと IPMP が判定した場合、そのインタフェースは故障としてフラグが付けられ、使用できなくなります。次に、故障したインタフェースに関連付けられていたデータ IP アドレスが、グループ内で機能している別のインタフェースに再分配されます。さらに、使用可能な場合は、スタンバイインタフェースも配備され、アクティブインタフェースの元の数を維持します。

次の図に示すような、3 つのインタフェースを含むアクティブ - スタンバイ構成の IPMP グループ `itops0` を考えます。

図 2-1 IPMP アクティブ-スタンバイ構成



IPMP グループ itops0 は次のように構成されています。

- このグループには、2 つのデータアドレス 192.168.10.10 と 192.168.10.15 が割り当てられています。
- ベースとなる 2 つのインタフェースがアクティブインタフェースとして構成され、柔軟なリンク名 net0 と net1 が割り当てられています。
- このグループにはスタンバイインタフェースが 1 つ含まれており、これにも柔軟なリンク名 net2 が割り当てられています。
- プローブベースの障害検出が使用されるため、アクティブインタフェースとスタンバイインタフェースは次のような検査用アドレスで構成されています。
  - net0: 192.168.10.30
  - net1: 192.168.10.32
  - net2: 192.168.10.34

注記 - 図2-1「IPMP アクティブ-スタンバイ構成」、図2-2「IPMP でのインタフェースの故障」、図2-3「IPMP でのスタンバイインタフェースの故障」、および図2-4「IPMP の回復プロセス」のアクティブ、オフライン、スタンバイ、および障害発生の各領域は、ベースとなるインタフェースのステータスを示しているだけであり、物理的な場所を示しているわけではありません。この IPMP 実装内では、インタフェースまたはアドレスの物理的な移動や IP インタフェースの転送は一切発生しません。これらの領域の役割は、ベースとなるインタフェースのステータスが故障、修復のいずれかの結果としてどのように変化するかを示すことだけです。

さまざまなオプションとともに ipmpstat コマンドを使用して、既存の IPMP グループに関する特定の種類の情報を表示できます。その他の例については、94 ページの「IPMP 情報のモニタリング」を参照してください。

次のコマンドは、図2-1「IPMP アクティブ-スタンバイ構成」の IPMP 構成に関する情報を表示します。

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

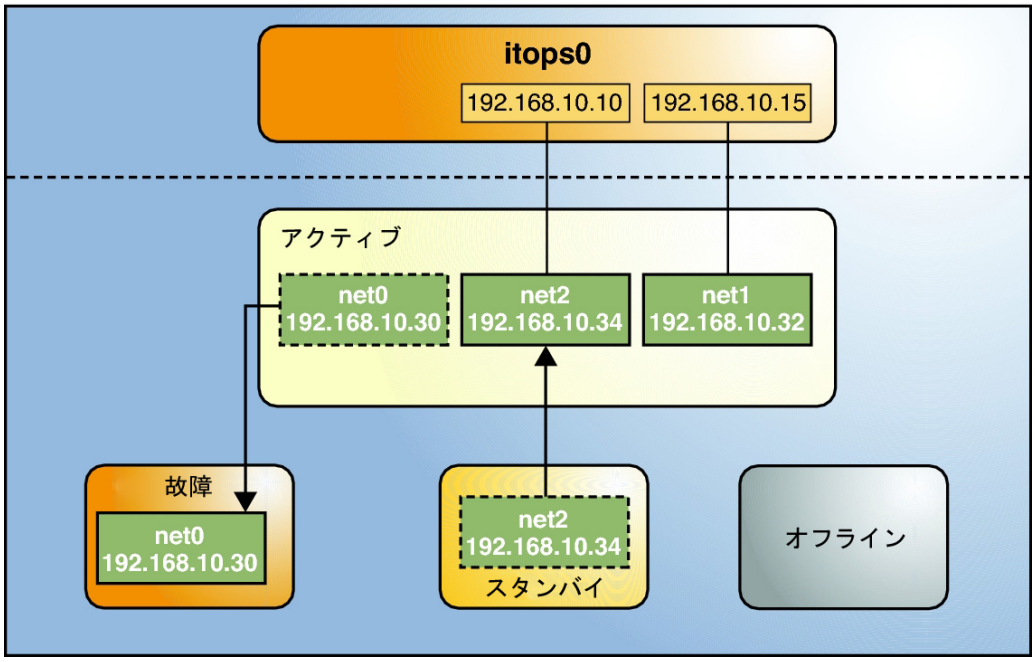
グループのベースとなるインタフェースに関する情報を表示するには、次のように入力します。

```
# 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 は、アクティブインタフェースの元の数を維持できるようにベースとなるインタフェースを管理することで、ネットワークの可用性を維持します。したがって、net0 が故障すると、IPMP グ

ループが引き続き 2 つのアクティブインタフェースを持てるように、net2 が配備されます。net2 のアクティブ化を次の図に示します。

図 2-2 IPMP でのインタフェースの故障



注記 - 図2-2「IPMP でのインタフェースの故障」のデータアドレスとアクティブインタフェースとの 1 対 1 のマッピングは、図を単純化するためのものにすぎません。IP カーネルモジュールは、データアドレスとインタフェースとの間の 1 対 1 の関係に必ずしも縛られることなく、データアドレスをランダムに割り当てることができます。

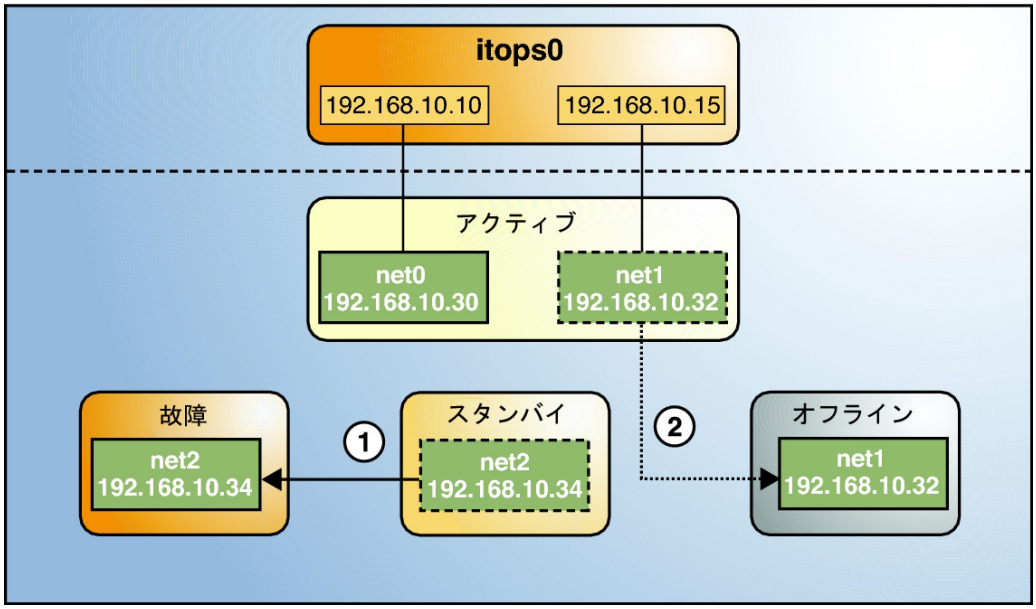
ipmpstat コマンドは、この図の情報を次のように表示します。

```
# 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
```

net0 が修復されると、アクティブインタフェースとしてのステータスに戻ります。一方、net2 は元のスタンバイステータスに戻されます。

別の障害シナリオを図2-3「IPMP でのスタンバイインタフェースの故障」に示します。このシナリオでは、スタンバイインタフェース net2 が故障します (1)。そのあと、アクティブインタフェースの 1 つである net1 が管理者によってオフラインに切り替えられます (2)。その結果、この IPMP グループには、機能しているインタフェース net0 が 1 つ残されます。

図 2-3 IPMP でのスタンバイインタフェースの故障



ipmpstat コマンドは、この図の情報を次のように表示します。

```
# 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
```

この障害では、インタフェースが修復されたあとの回復の動作が異なります。この回復プロセスは、修復後の構成と比較した IPMP グループのアクティブインタフェースの元の数に依存します。次の図は、回復プロセスを表します。

図 2-4 IPMP の回復プロセス

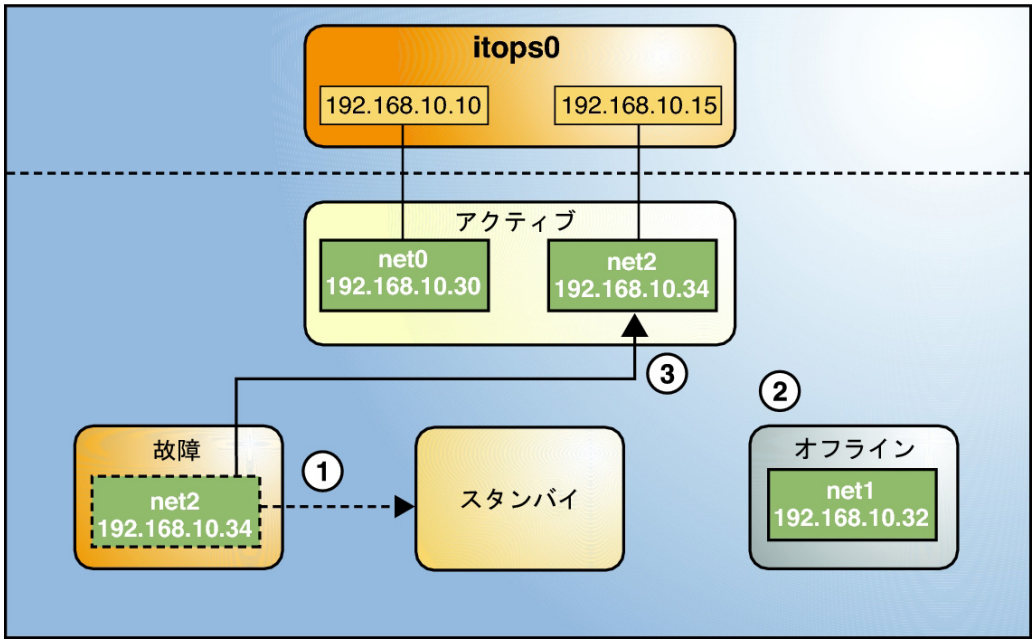


図2-4「IPMP の回復プロセス」で、net2 が修復されると、通常、スタンバイインタフェースとしての元のステータスに戻ります (1)。ところが、この IPMP グループは依然として、アクティブインタフェースの元の数である 2 個を反映していません。これは、net1 が引き続きオフラインのままになっているからです (2)。したがって、IPMP は代わりに net2 をアクティブインタフェースとして配備します (3)。

ipmpstat コマンドは、修復後の IPMP シナリオを次のように表示します。

```
# 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
```

FAILBACK=no モードも構成されたアクティブインタフェースが故障に関与している場合にも、同様の回復プロセスが発生します。その場合、故障したアクティブインタフェースが修復されても、自動的にはアクティブステータスに戻りません。図2-2「IPMP でのインタフェースの故障」の net0 が FAILBACK=no モードに構成されているとします。そのモードでは、修復された

net0 は、最初はアクティブインタフェースであったとしても、スタンバイインタフェースになります。この IPMP グループのアクティブインタフェースの元の数である 2 個を維持するように、インタフェース net2 はアクティブのままになります。

ipmpstat コマンドは、回復情報を次のように表示します。

```
# 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
```

このタイプの構成の詳細については、70 ページの「FAILBACK=no モード」を参照してください。

IPMP アドレス指定

IPMP 障害検出は、IPv4 ネットワークと、IPv4 および IPv6 のデュアルスタックネットワークの両方で構成できます。IPMP で構成されたインタフェースは、データアドレスと検査用アドレスという 2 種類のアドレスをサポートしています。IP アドレスは IPMP インタフェース (グループ) 上に*のみ*存在し、データアドレスとして指定されます。検査用アドレスは、ベースとなるインタフェース上に存在する IP アドレスです。

データアドレス

*データアドレス*とは従来の IPv4 および IPv6 アドレスのことであり、ブート時に DHCP サーバーによって IP インタフェースに動的に割り当てられるか、あるいは ipadm コマンドを使用して手動で割り当てられます。データアドレスは IPMP インタフェース (グループ) に*のみ*割り当てられます。標準の IPv4 パケットトラフィックおよび IPv6 パケットトラフィック (該当する場合) は*データトラフィック*とみなされます。データトラフィックは、IPMP インタフェースでホストされているデータアドレスを使用し、その IPMP インタフェースまたはグループのアクティブインタフェースを通過します。

検査用アドレス

*検査用アドレス*とは、プローブベースの障害および修復検出を実行するために in.mpathd デーモンによって使用される、IPMP 固有のアドレスのことです。検査用アドレスもやはり、DHCP

サーバーによる動的な割り当て、または ipadm コマンドを使用した手動での割り当てが可能です。IPMP グループのベースとなるインタフェースには、検査用アドレスのみ割り当てられます。あるベースとなるインタフェースが故障した場合、そのインタフェースの検査用アドレスが引き続き in.mpathd デーモンによってプローブベースの障害検出のために使用され、そのインタフェースがその後修復されたかどうかがチェックされます。

注記 - プローブベースの障害検出を使用する場合のみ、検査用アドレスを構成します。それ以外の場合は、推移的プローブを有効にすることで、検査用アドレスを使用しなくても障害を検出できます。検査用アドレスを使用する場合としない場合のプローブベースの障害検出の詳細については、66 ページの「プローブベースの障害検出」を参照してください。

以前の IPMP 実装では、特にインタフェースの障害発生時にアプリケーションによって使用されないように、検査用アドレスは DEPRECATED としてマークされる必要がありました。現在の実装では、検査用アドレスはベースとなるインタフェース内に存在しています。したがって、IPMP を認識しないアプリケーションによってこれらのアドレスが間違って使用されることはなくなりました。ただし、これらのアドレスがデータパケットの発信元の候補として考慮されないように、システムは自動的に、NOFAILOVER フラグの付いたすべてのアドレスを DEPRECATED としてマークします。

サブネット上の任意の IPv4 アドレスを検査用アドレスとして使用できます。IPv4 アドレスは、多くのサイトでは限定リソースなので、ルート指定できない RFC 1918 プライベートアドレスを検査用 IP アドレスとして指定したい場合もあります。 in.mpathd デーモンは、ICMP 検査信号を検査用アドレスと同じサブネットのホストとしか交換しません。RFC 1918 形式の検査用アドレスを使用していない場合は、ネットワーク上のほかのシステム (ルーターが望ましい) を適切な RFC 1918 サブネットのアドレスで必ず構成してください。この構成により、in.mpathd デーモンは、ターゲットシステムと正常に検査信号を交換できます。RFC 1918 プライベートアドレスの詳細については、RFC 1918, Address Allocation for Private Internets (http://www.rfc-editor.org/rfc/rfc1918.txt) を参照してください。

有効な IPv6 検査用 IP アドレスは、物理インタフェースのリンクローカルアドレスだけです。IPMP 検査用 IP アドレスとして機能する別の IPv6 アドレスは必要ありません。IPv6 リンクローカルアドレスは、インタフェースのメディアアクセスコントロール (MAC) アドレスに基づいています。リンクローカルアドレスは、インタフェースがブート時に IPv6 を使用できるようになったり、インタフェースが ipadm コマンドによって手動で構成されたりした場合に、自動的に構成されます。

IPMP グループですべてのグループのインタフェースに IPv4 と IPv6 の両方が使用される場合には、別個の IPv4 検査用アドレスを構成する必要はありません。in.mpathd デーモンは、IPv6 リンクローカルアドレスを検査用 IP アドレスとして使用します。

IPMP での障害検出

IPMP は、トラフィックを送受信するネットワークの継続的な可用性を保証するために、IPMP グループのベースとなる IP インタフェースに対して障害検出を実行します。故障したインタフェースは、修復されるまで使用不可能なままになります。残りのアクティブインタフェースが機能し続ける一方で、既存のスタンバイインタフェースが必要に応じて配備されます。

`in.mpathd` デーモンは次のタイプの障害検出を処理します。

- プローブベースの障害検出の 2 つのタイプ:
  - 検査用アドレスが構成されない (推移的プローブ)。
  - 検査用アドレスが構成される。
- リンクベースの障害検出 (NIC ドライバがサポートしている場合)

プローブベースの障害検出

プローブベースの障害検出では、ICMP プローブを使用してインタフェースが故障しているかどうかをチェックします。この障害検出手法の実装は、検査用アドレスが使用されるかどうかによって決まります。

検査用アドレスを使用するプローブベースの障害検出

この障害検出手法では、検査用アドレスを使用する ICMP 検査信号メッセージを送受信します。*プローブトラフィック*または*テストトラフィック*とも呼ばれるこれらのメッセージは、インタフェース経由で同じローカルネットワーク上の 1 つ以上のターゲットシステムに送信されます。`in.mpathd` デーモンは、プローブベースの障害検出用に構成されたすべてのインタフェースを経由してすべてのターゲットを個別にプローブします。ある特定のインタフェースで、連続する 5 つのプローブに対して応答がない場合、`in.mpathd` デーモンはそのインタフェースに障害があるものとみなします。プローブを発信する頻度は、*障害検出時間* (*FDT*) に依存します。障害検出時間のデフォルト値は 10 秒です。ただし、FDT は IPMP 構成ファイルで調整できます。手順については、92 ページの「IPMP デーモンの動作を構成する方法」を参照してください。

プローブベースの障害検出を最適化するには、`in.mpathd` デーモンからのプローブを受信する複数のターゲットシステムを設定する必要があります。複数のターゲットシステムを設定することで、報告された障害の性質をより正確に判定できます。たとえば、唯一定義されたター

ゲットシステムから応答がない場合、そのターゲットシステムの障害を示している可能性もあれば、IPMP グループのインタフェースの 1 つの障害を示している可能性もあります。これに対し、いくつかのターゲットシステムのうちの 1 つのシステムだけがプローブに応答しない場合は、IPMP グループ自体ではなく、ターゲットシステムで障害が発生している可能性があります。

`in.mpathd` デーモンは、プローブするターゲットシステムを動的に決定します。まず、デーモンはルーティングテーブル内で、IPMP グループのインタフェースに関連付けられた検査用アドレスと同じサブネット上にあるターゲットシステムを検索します。そのようなターゲットが見つかった場合、デーモンはそれらをプローブのターゲットとして使用します。同じサブネット上でターゲットシステムが見つからない場合、デーモンは、リンク上の近くのホストをプローブするマルチキャストパケットを送信します。ターゲットシステムとして使用するホストを決定するために、すべてのホストのマルチキャストアドレス (IPv4 では `224.0.0.1`、IPv6 では `ff02::1`) にマルチキャストパケットが送信されます。エコーパケットに応答する最初の 5 つのホストが、プローブのターゲットとして選択されます。マルチキャストプローブに応答したルーターまたはホストを検出できない場合、デーモンはプローブベースの障害を検出できません。この場合、`ipmpstat -i` コマンドはプローブの状態を `unknown` として報告します。

ホストルートを使用して、`in.mpathd` デーモンが使用するターゲットシステムのリストを明示的に構成できます。手順については、89 ページの「プローブベースの障害検出の構成」を参照してください。

検査用アドレスを使用しないプローブベースの障害検出

検査用アドレスを使用しないこの手法は、2 種類のプローブを使用して実装されています。

- **ICMP プローブ**

  ICMP プローブは、ルーティングテーブルに定義されたターゲットをプローブするために、IPMP グループ内のアクティブインタフェースによって送信されます。*アクティブ*インタフェースとは、そのインタフェースのリンク層 (L2) アドレス宛てのインバウンド IP パケットを受信できるベースとなるインタフェースのことです。ICMP プローブは、データアドレスをそのプローブの発信元アドレスとして使用します。ICMP プローブがそのターゲットに到達し、ターゲットから応答が得られた場合、そのアクティブインタフェースは動作しています。

- **推移的プローブ**

  推移的プローブは、アクティブインタフェースをプローブするために、IPMP グループ内の代替インタフェースによって送信されます。代替インタフェースとは、インバウンド IP パケットを能動的に受信しないベースとなるインタフェースのことです。

たとえば、4 つのベースとなるインタフェースと 1 つのデータアドレスから成る IPMP グループを考えます。この構成では、アウトバウンドパケットはベースとなるインタフェースをすべて使用できます。一方、インバウンドパケットは、データアドレスがバインドされたインタフェースによってのみ受信できます。インバウンドパケットを受信できない残り 3 つのベースとなるインタフェースが、*代替インタフェース*となります。

代替インタフェースがアクティブインタフェースへのプローブの送信と応答の受信に成功した場合、そのアクティブインタフェースは機能しており、推論により、プローブを送信した代替インタフェースも機能しています。

注記 - Oracle Solaris では、プローブベースの障害検出は検査用アドレスを使用して動作します。検査用アドレスなしでプローブベースの障害検出を選択するには、推移的プローブを手動で有効にする必要があります。手順については、91 ページの「障害検出手法の選択」を参照してください。

グループ障害

*グループ障害*は、IPMP グループ内のすべてのインタフェースで同時に障害が発生したと思われる場合に発生します。この場合、ベースとなるインタフェースは一切使用できません。また、すべてのターゲットシステムで同時に障害が発生したときに、プローブベースの障害検出が有効になっていた場合、`in.mpathd` デーモンはその現在のターゲットシステムをすべてフラッシュし、新しいターゲットシステムに対してプローブします。

検査用アドレスを持たない IPMP グループでは、アクティブインタフェースをプローブできる単一のインタフェースが*プローバ*として指定されます。この指定されたインタフェースには、`FAILED` フラグと `PROBER` フラグが両方とも設定されます。このインタフェースにデータアドレスがバインドされるため、このインタフェースは引き続き、ターゲットをプローブして回復を検出できます。

リンクベースの障害検出

リンクベースの障害検出は、インタフェースがその種の障害検出をサポートしている場合は、常に有効です。

サードパーティーのインタフェースがリンクベースの障害検出をサポートしているかどうかを判定するには、`ipmpstat -i` コマンドを使用します。ある特定のインタフェースの出力の `LINK` 列に `unknown` ステータスが含まれる場合、そのインタフェースはリンクベースの障害検出をサポートし

ません。デバイスに関するより具体的な情報については、製造元のドキュメントを参照してください。

リンクベースの障害検出をサポートするネットワークドライバは、インタフェースのリンク状態をモニターし、リンク状態が変わったときにネットワークサブシステムに通知します。変更を通知されると、ネットワークサブシステムは、インタフェースの RUNNING フラグを適宜設定または解除します。インタフェースの RUNNING フラグが解除されたことを検出すると、in.mpathd デーモンは即座にインタフェースに障害があるものとみなします。

障害検出と匿名グループ機能

IPMP は匿名グループでの障害検出をサポートしています。デフォルトでは、IPMP は IPMP グループに属するインタフェースのステータスのみをモニターします。ただし、どの IPMP グループにも属さないインタフェースのステータスも追跡するように IPMP デーモンを構成することもできます。したがって、これらのインタフェースは*匿名グループ*の一部とみなされます。ipmpstat -g コマンドを発行した場合、匿名グループは二重ダッシュ (--) として表示されます。匿名グループ内のインタフェースでは、データアドレスが検査用アドレスとしても機能します。これらのインタフェースは名前付きの IPMP グループに属していないため、これらのアドレスはアプリケーションから可視となります。IPMP グループの一部でないインタフェースの追跡を有効にする方法については、92 ページの「IPMP デーモンの動作を構成する方法」を参照してください。

物理インタフェースの回復検出

「*修復検出時間*」は障害検出時間の 2 倍です。障害検出のデフォルト時間は 10 秒です。したがって、修復検出のデフォルト時間は 20 秒です。故障したインタフェースが RUNNING フラグでふたたびマークされ、障害検出手法がそのインタフェースを修復済みとして検出すると、in.mpathd デーモンはそのインタフェースの FAILED フラグを解除します。修復されたインタフェースは、管理者が最初に設定したアクティブインタフェースの数に応じて再配備されます。

ベースとなるインタフェースが故障したときに、プローブベースの障害検出が使用されていた場合、in.mpathd デーモンは、検査用アドレスが構成されていない場合は指定されたプローバ経由で、またはそのインタフェースの検査用アドレスを使用して、プローブを継続します。

インタフェース修復時の回復プロセス処理は、次のように、障害が発生したインタフェースが最初に構成された方法によって決まります。

- 故障したインタフェースが最初はアクティブインタフェースだった場合、修復されたインタフェースは元のアクティブステータスに戻ります。その IPMP グループで、十分な数 (システム管理者によって定義された数) のインタフェースがアクティブになっていれば、障害時に代わりに機能していたスタンバイインタフェースは元のスタンバイステータスに切り替えられます。

注記 - 例外は、修復されたアクティブインタフェースが `FAILBACK=no` モードでも構成されていた場合です。詳細は、70 ページの「`FAILBACK=no` モード」を参照してください。

- 故障したインタフェースが最初はスタンバイインタフェースだった場合、IPMP グループにアクティブインタフェースの元の数が反映されていれば、修復されたインタフェースは元のスタンバイステータスに戻ります。それ以外の場合、スタンバイインタフェースはアクティブインタフェースになります。

インタフェースの障害や修復時の IPMP の動作方法の図を用いた説明については、59 ページの「IPMP の動作方法」を参照してください。

FAILBACK=no モード

デフォルトでは、障害発生後に修復されたアクティブインタフェースは自動的に、IPMP グループ内で元のアクティブインタフェースに戻ります。この動作は、`in.mpathd` デーモンの構成ファイル内の `FAILBACK` パラメータの値によって制御されます。ただし、データアドレスが修復されたインタフェースに再マッピングされるときに発生する短い中断でさえ許容できないことがあります。その場合は、アクティブ化されたスタンバイインタフェースをそのままアクティブインタフェースとして機能させたほうが望ましいこともあります。IPMP では、デフォルト動作をオーバーライドして、インタフェースが修復時に自動的にアクティブにならないようにすることができます。これらのインタフェースは `FAILBACK=no` モードで構成する必要があります。関連する手順については、92 ページの「IPMP デーモンの動作を構成する方法」を参照してください。

`FAILBACK=no` モードのアクティブインタフェースが故障したあと修復された場合、`in.mpathd` デーモンは IPMP の構成を次のように復元します。

- IPMP グループがアクティブインタフェースの元の構成を反映している場合、デーモンはこのインタフェースの `INACTIVE` ステータスを維持します。
- 修復時点での IPMP の構成が、グループのアクティブインタフェースの元の構成を反映していない場合、`FAILBACK=no` ステータスであっても、修復されたインタフェースがアクティブインタフェースとして再配備されます。

注記 - FAILBACK = NO モードは、インタフェースごとのチューニング可能なパラメータとしてではなく、IPMP グループ全体に設定されます。

IPMP と動的再構成

Oracle Solaris の動的再構成 (DR) 機能によって、システムの実行中にインタフェースなどのシステムハードウェアを再構成できます。DR は、この機能をサポートするシステムでのみ使用できます。DR をサポートするシステム上では、IPMP は RCM (Reconfiguration Coordination Manager) フレームワークに統合されています。したがって、NIC の接続、切り離し、または再接続を安全に行うことができ、RCM がシステムコンポーネントの動的再構成を管理します。たとえば、新しいインタフェースを接続して plumb したあと、それを既存の IPMP グループに追加できます。これらのインタフェースは、構成されたあとすぐに IPMP で使用可能となります。

NIC を切断するすべてのリクエストは、まず接続性を保持できるかどうかチェックされます。たとえば、デフォルトでは、IPMP グループ内にない NIC を切断することはできません。IPMP グループ内の機能中のインタフェースだけを含む NIC も切断できません。ただし、システムコンポーネントを削除する必要がある場合は、cfgadm(1M)のマニュアルページに説明されている cfgadm コマンドの -f オプションを使用して、この動作をオーバーライドできます。

チェックが成功すると、in.mpathd デーモンはインタフェースに OFFLINE フラグを設定します。インタフェース上の検査用アドレスがすべて構成解除されます。次に、NIC はシステムを unplumb します。

これらの段階のいずれかが失敗した場合、または同じシステムコンポーネントのその他のハードウェアの DR で障害が発生した場合は、永続的な構成のみリストアされます。この場合は、次のエラーメッセージがログに記録されます。

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

それ以外の場合、切断要求は正常に完了しています。システムからコンポーネントを削除しても、既存の接続は中断されません。

注記 - NIC を交換するときは、同じ種類 (Ethernet など) のカードであることを確認してください。NIC が交換されたら、持続的な IP インタフェース構成がその NIC に適用されます。

♦ ♦ ♦ 第 3 章

3 IPMP の管理

この章では、Oracle Solaris リリースで、IPMP を使用してインタフェースグループを管理する方法について説明します。この章のタスクでは、`ipadm` コマンド (Oracle Solaris 10 で IPMP を構成するときに使用する `ifconfig` コマンドを置換する) を使用して IPMP を構成する方法について説明します。これら 2 つのコマンドを互いにマップする方法の詳細については、『Oracle Solaris 10 から Oracle Solaris 11.2 への移行 』の「ifconfig コマンドと ipadm コマンドの比較」を参照してください。

IPMP の概念モデルの変更の詳細な説明については、51 ページの「IPMP の新機能」を参照してください。

この章の内容は、次のとおりです。

IPMP グループの構成

ここからは、IPMP グループを計画および構成する方法について説明します。IPMP グループのインタフェースとしての実装については、第2章「IPMP の管理について」の概要で説明しています。この章では、用語 *IPMP グループ*と *IPMP インタフェース*が同義として使用されます。

このセクションで説明するタスクは次のとおりです。

■ 80 ページの「アクティブ - スタンバイ IPMP グループを構成する方法」

▼ IPMP グループの計画を立てる方法

次の手順には、必要となる計画タスクと IPMP グループを構成する前に収集する情報が含まれています。これらのタスクは順番に実行する必要はありません。

IPMP の構成は、システム上でホストされるタイプのトラフィックを処理するためのネットワークの要件によって決まります。IPMP はアウトバウンドのネットワークパケットを IPMP グループのインタフェース間で分散するため、ネットワークのスループットを改善します。ただし、ある特定の TCP 接続では、インバウンドトラフィックは通常、アウトオブオーダーのパケットを処理するリスクを最小限に抑えるために、1 つの物理パスのみをたどります。

したがって、ネットワークが大量のアウトバウンドトラフィックを処理する場合、多数のインタフェースを 1 つの IPMP グループに構成すると、ネットワークのパフォーマンスを改善できます。そうではなく、システムが大量のインバウンドトラフィックをホストする場合は、グループ内のインタフェースの数を増やしても、トラフィックの負荷分散によってパフォーマンスが改善されるとはかぎりません。ただし、ベースとなるインタフェースの数を増やせば、インタフェースで障害が発生した際のネットワークの可用性を保証しやすくなります。

注記 - 各サブネットまたは L2 ブロードキャストドメインに対して、IPMP グループを 1 つだけ構成する必要があります。詳細は、54 ページの「IPMP を使用するための規則」を参照してください。

---

1. **ユーザーの要求に適した全般的な IPMP 構成を決定します。**

   使用する IPMP 構成を決定するときのガイダンスとして、この手順のタスクのサマリー情報を参照してください。

2. **(SPARC のみ) グループ内の各インタフェースが一意の MAC アドレスを持っていることを確認します。**

   システムの各インタフェースに一意の MAC アドレスを構成するには、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「各インタフェースの MAC アドレスが一意であることを確認する方法」を参照してください。

3. **STREAMS モジュールの同じセットが構成されており、IPMP グループ内のすべてのインタフェースにプッシュされていることを確認します。**

ガイドラインと使用するコマンド構文については、54 ページの「IPMP を使用するための規則」を参照してください。

4. **IPMP グループのすべてのインタフェースで同じ IP アドレス指定形式を使用します。**

   1 つのインタフェースが IPv4 向けに構成されている場合は、その IPMP グループのすべてのインタフェースを IPv4 向けに構成する必要があります。同様に、1 つのインタフェースに IPv6 アドレス指定を追加した場合は、IPMP グループ内のすべてのインタフェースを IPv6 をサポートするように構成する必要があります。

5. **実装する障害検出のタイプを決定します。**

   たとえば、プローブベースの障害検出を実装する場合は、ベースとなるインタフェースで検査用アドレスを構成する必要があります。66 ページの「IPMP での障害検出」を参照してください。

6. **IPMP グループ内のすべてのインタフェースが同じローカルネットワークに接続されていることを確認します。**

   たとえば、同じ IP サブネット上の Ethernet スイッチを 1 つの IPMP グループに構成できます。1 つの IPMP グループにいくつでもインタフェースを構成できます。

   注記 - たとえば、システムに物理インタフェースが 1 つだけ存在する場合は、単一インタフェースの IPMP グループを構成することもできます。57 ページの「IPMP インタフェース構成のタイプ」を参照してください。

   ---

7. **IPMP グループに、別のネットワークメディアタイプのインタフェースが含まれていないことを確認します。**

   グループ化するインタフェースは、同じインタフェースタイプである必要があります。たとえば、1 つの IPMP グループに Ethernet インタフェースとトークンリングインタフェースを組み合わせることはできません。別の例としては、同じ IPMP グループに、トークンバスインタフェースと非同期転送モード (ATM) インタフェースを組み合わせることはできません。

8. **ATM インタフェースを持つ IPMP の場合は、LAN エミュレーションモードで ATM インタフェースを構成します。**

   RFC 1577 (http://www.rfc-editor.org/rfc/rfc1577.txt) および RFC 2225 (http://www.rfc-editor.org/rfc/rfc2225.txt) で定義されている Classical IP over ATM テクノロジを使用するインタフェースでは、IPMP はサポートされていません。

▼ DHCP を使用して IPMP グループを構成する方法

複数インタフェースの IPMP グループは、アクティブ - アクティブインタフェースで構成することも、アクティブ - スタンバイインタフェースで構成することもできます。57 ページの「IPMP インタフェース構成のタイプ」を参照してください。次の手順では、DHCP を使用してアクティブ - スタンバイ IPMP グループを構成する方法について説明します。

始める前に 下記の手順を実行する前に、次の点を確認してください。

- 想定 IPMP グループに含める予定の IP インタフェースが、システムのネットワークデータリンク上で正しく構成されていることを確認します。手順については、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』を参照してください。IPMP インタフェースは、ベースとなる IP インタフェースがまだ作成されていなくても作成できます。ただし、ベースとなる IP インタフェースを作成しないと、この IPMP インタフェースでのその後の構成は失敗します。
- さらに、SPARC ベースのシステムを使用する場合は、インタフェースごとに一意の MAC アドレスを構成する必要があります。『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「各インタフェースの MAC アドレスが一意であることを確認する方法」を参照してください。
- 最後に、DHCP を使用する場合は、ベースとなるインタフェースの*リースが無限になっている*ことを確認します。そうしないと、IPMP グループの障害が発生した場合に、検査用アドレスが期限切れとなり、`in.mpathd` デーモンがプローブベースの障害検出を無効化し、リンクベースの障害検出が使用されます。リンクベースの障害検出によってインタフェースが機能していることがわかると、デーモンは誤ってインタフェースが修復されたと報告する可能性があります。DHCP の構成方法の詳細は、『Oracle Solaris 11.2 での DHCP の作業 』を参照してください。

1. **`root` の役割になります。**

2. **IPMP インタフェースを作成します。**

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

   ここで、*ipmp-interface* は IPMP インタフェースの名前を指定します。IPMP インタフェースには、意味のある任意の名前を割り当てることができます。IP インタフェースと同様に、名前は `ipmp0` のように文字列と数字から構成されます。

3. **ベースとなる IP インタフェースがまだ存在しない場合はそれらを作成します。**

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

ここで、*under-interface* は、IPMP グループに追加する IP インタフェースを表します。

4. **IPMP グループに、検査用アドレスを含むベースとなる IP インタフェースを追加します。**

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

   IP インタフェースは、システムで使用可能な数だけ IPMP グループに追加できます。

5. **IPMP インタフェース上のデータアドレスを DHCP によって構成および管理することを指定します。**

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

   直前の手順は、DHCP サーバーによって提供されたアドレスをアドレスオブジェクトに関連付けます。アドレスオブジェクトは、たとえば `ipmp0/v4` のように、*interface*/*address-type* という形式を使用して IP アドレスを一意に識別します。アドレスオブジェクトの詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「IPv4 インタフェースを構成する方法」を参照してください。

6. **検査用アドレスによるプローブベースの障害検出を使用する場合は、ベースとなるインタフェース上の検査用アドレスを DHCP によって管理することを指定します。**

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

   段階 6 で自動的に作成されるアドレスオブジェクトは、たとえば `net0/v4` のように、*under-interface*/*address-type* という形式を使用します。

7. **(オプション) IPMP グループのベースとなるインタフェースごとに、段階 6 を繰り返します。**

例 3-1 DHCP を使用した IPMP グループの構成

この例は、DHCP を使用したアクティブ - スタンバイ IPMP グループの構成を示しており、次のシナリオに基づいています。

- 3 つのベースとなるインタフェース `net0`、`net1`、および `net2` が IPMP グループに構成されます。
- IPMP インタフェース `ipmp0` は、同じ名前を IPMP グループと共有します。
- `net2` が指定されたスタンバイインタフェースになります。
- ベースとなるインタフェースのすべてに検査用アドレスが割り当てられます。

最初に IPMP インタフェースが作成されます。

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

ベースとなる IP インタフェースが作成され、IPMP インタフェースに追加されます。

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

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

DHCP によって管理された IP アドレスが IPMP インタフェースに割り当てられます。IPMP インタフェースに割り当てられる IP アドレスはデータアドレスです。この例では、IPMP インタフェースには 2 つのデータアドレスがあります。

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

次に、DHCP によって管理された IP アドレスが IPMP グループのベースとなる IP インタフェースに割り当てられます。ベースとなるインタフェースに割り当てられる IP アドレスは、プローブベースの障害検出に使用される検査用アドレスです。

```
# 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
```

最後に、net2 インタフェースがスタンバイインタフェースとして構成されます。

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

▼ アクティブ - アクティブ IPMP グループを構成する方法

次の手順では、アクティブ - アクティブ IPMP グループを手動で構成する方法について説明します。この手順の手順 1 から 4 では、リンクベースのアクティブ - アクティブ IPMP グループを構成する方法について説明します。手順 5 では、リンクベースの構成をプローブベースにする方法について説明します。

始める前に 想定 IPMP グループに含める予定の IP インタフェースが、システムのネットワークデータリンク上で正しく構成されていることを確認します。手順については、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「IPv4 インタフェースを構成する方法」を参照して

ください。IPMP インタフェースは、ベースとなる IP インタフェースがまだ存在していなくても作成できます。ただし、この IPMP インタフェースでのその後の構成は失敗します。

さらに、SPARC ベースのシステムを使用する場合は、インタフェースごとに一意の MAC アドレスを構成します。『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「各インタフェースの MAC アドレスが一意であることを確認する方法」を参照してください。

1. **root の役割になります。**

2. **IPMP インタフェースを作成します。**

   # ipadm create-ipmp *ipmp-interface*

   ここで、*ipmp-interface* は IPMP インタフェースの名前を指定します。IPMP インタフェースには、意味のある任意の名前を割り当てることができます。IP インタフェースと同様に、名前は `ipmp0` のように文字列と数字から構成されます。

3. **ベースとなる IP インタフェースをグループに追加します。**

   # ipadm add-ipmp -i *under-interface*`1` **[-i** *underinterface*`2` **...]** *ipmp-interface*

   ここで、*under-interface* は、IPMP グループのベースとなるインタフェースを表します。IP インタフェースは、システムで使用可能な数だけ追加できます。

   注記 - デュアルスタック環境では、特定のグループにインタフェースの IPv4 インスタンスを配置すると、IPv6 インスタンスが自動的に同じグループに配置されます。

4. **IPMP インタフェースにデータアドレスを追加します。**

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

   ここで、*address* は CIDR 表記にすることができます。

   注記 - IPMP グループ名または IP アドレスの DNS アドレスのみが必要です。

5. **検査用アドレスによるプローブベースの障害検出を使用する場合は、ベースとなるインタフェースに検査用アドレスを追加します。**

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

   ここで、*address* は CIDR 表記にすることができます。IPMP グループのすべての検査用 IP アドレスは、1 つの IP サブネットに属していなければならず、したがって同じネットワーク接頭辞を使用する必要があります。

▼ アクティブ - スタンバイ IPMP グループを構成する方法

次の手順では、1 つのインタフェースがスタンバイインタフェースとして保持される IPMP グループを構成する方法について説明します。このインタフェースは、グループ内のアクティブインタフェースが故障した場合にのみ配備されます。

スタンバイインタフェースの概要については、57 ページの「IPMP インタフェース構成のタイプ」を参照してください。

1. **root の役割になります。**

2. **IPMP インタフェースを作成します。**

   # ipadm create-ipmp *ipmp-interface*

   ここで、*ipmp-interface* は IPMP インタフェースの名前を指定します。

3. **ベースとなる IP インタフェースをグループに追加します。**

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

   ここで、*under-interface* は、IPMP グループのベースとなるインタフェースを表します。IP インタフェースは、システムで使用可能な数だけ追加できます。

   注記 - デュアルスタック環境では、特定の IPMP グループにインタフェースの IPv4 インスタンスを配置すると、IPv6 インスタンスが自動的に同じグループに配置されます。

4. **IPMP インタフェースにデータアドレスを追加します。**

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

   ここで、*address* は CIDR 表記にすることができます。

5. **検査用アドレスによるプローブベースの障害検出を使用する場合は、ベースとなるインタフェースに検査用アドレスを追加します。**

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

   ここで、*address* は CIDR 表記にすることができます。IPMP グループのすべての検査用 IP アドレスは、1 つの IP サブネットに属していなければならず、したがって同じネットワーク接頭辞を使用する必要があります。

6. **ベースとなるインタフェースの 1 つをスタンバイインタフェースとして構成します。**

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

例 3-2 アクティブ - スタンバイ IPMP グループの構成

次の例では、アクティブ - スタンバイ IPMP 構成を作成する方法を示します。

最初に IPMP インタフェースが作成されます。

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

次に、ベースとなる IP インタフェースが作成され、IPMP インタフェースに追加されます。

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

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

次に、IP アドレスが IPMP インタフェースに割り当てられます。IPMP インタフェースに割り当てられる IP アドレスはデータアドレスです。この例では、IPMP インタフェースには 2 つのデータアドレスがあります。

```
# 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
```

この例の IP アドレスには、10 進数で表される `prefixlen` プロパティーが含まれています。IP アドレスの `prefixlen` 部分は、アドレスの左から何ビットがアドレスの IPv4 ネットマスクまたは IPv6 接頭部に該当するかを指定します。残りの下位ビットは、アドレスのホスト部を定義します。`prefixlen` プロパティーをアドレスのテキスト表現に変換すると、アドレスのネットワーク部として使用するビット位置には 1 が、ホスト部として使用するビット位置には 0 が含まれています。このプロパティーは、`dhcp` アドレスオブジェクトタイプではサポートされません。詳細は、`ipadm`(1M)のマニュアルページを参照してください。

次に、IPMP グループのベースとなる IP インタフェースに IP アドレスが割り当てられます。ベースとなるインタフェースに割り当てられる IP アドレスは、プローブベースの障害検出に使用される検査用アドレスです。

```
# 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
```

最後に、net2 インタフェースがスタンバイインタフェースとして構成されます。

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

管理者は ipmpstat コマンドを使用して IPMP 構成を表示できます。

```
# 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
```

IPMP の配備時にルーティングを維持

IPMP グループの構成時に、IPMP インタフェースは、ベースとなるインタフェースの IP アドレスを継承して、これらのアドレスをデータアドレスとして使用します。次に、ベースとなるインタフェースは、IP アドレス **0.0.0.0** を受信します。そのため、特定のインタフェースを使用して定義されたルートがあるとき、そのインタフェースが IPMP グループに追加されると、そのルートは失われます。

IPMP の構成時にルーティングが失われる場合は通常、デフォルトルートが関与し、Oracle Solaris のインストールに関連して発生します。インストール時に、デフォルトルート (プライマリインタフェースなど、システム上のインタフェースを使用する) を定義する必要があります。そのあとで、デフォルトルートを定義した同じインタフェースを使用して IPMP グループを構成します。IPMP の構成後、インタフェースのアドレスは IPMP インタフェースに転送されたため、システムはネットワークパケットをルーティングできなくなります。

IPMP の使用中にデフォルトルートが保持されるようにするには、インタフェースを指定せずにルートを定義する必要があります。この方法で、IPMP インタフェースを含む任意のインタフェースをルーティングに使用できます。その結果、システムはトラフィックのルーティングを続行できます。

注記 - 次のタスクでは、デフォルトルートとして定義されたプライマリインタフェースを例として使用します。ただし、このようにルーティングが失われる事例は、ルーティングに使用されており、あとで IPMP グループの一部になるすべてのインタフェースに当てはまります。

▼ IPMP 使用時にデフォルトルートを保持する方法

次の手順では、IPMP を構成した場合に、デフォルトルートを保持する方法を説明します。

1. **コンソールを使用してシステムにログインします。**
   この手順を実行するには、コンソールを使用する必要があります。ssh または telnet コマンドを使用してログインした場合、以降の段階を実行すると接続が失われます。

2. **(オプション) ルーティングテーブルに現在定義されているルートを表示します。**

   ```
   # netstat -nr
   ```

3. **特定のインタフェースにバインドされているルートを削除します。**

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

4. **インタフェースを指定せずにルートを追加します。**

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

5. **(オプション) ルーティングテーブルの再定義されたルートを表示します。**

   ```
   # netstat -nr
   ```

6. **(オプション) 情報が変更されていない場合、ルーティングサービスを再起動し、ルーティングテーブル内の情報を再度調べて、ルートが正しく再定義されていることを確認します。**

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

例 3-3 IPMP のルートの定義

この例では、インストール時に net0 のデフォルトルートが定義されたことを前提としています。

```
# 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
```

IPMP の管理

このセクションでは、システム上に作成した IPMP グループを保守する次の手順について説明します。

- 84 ページの「IPMP グループにインタフェースを追加する方法」
- 85 ページの「IPMP グループからインタフェースを削除する方法」
- 85 ページの「IPMP グループに IP アドレスを追加する方法」
- 86 ページの「IPMP インタフェースから IP アドレスを削除する方法」
- 87 ページの「インタフェースを 1 つの IPMP グループから別の IPMP グループに移動する方法」
- 88 ページの「IPMP グループを削除する方法」

▼ IPMP グループにインタフェースを追加する方法

始める前に グループに追加するインタフェースが、必要な要件をすべて満たしていることを確認します。要件の一覧については、74 ページの「IPMP グループの計画を立てる方法」を参照してください。

1. **root の役割になります。**

2. **ベースとなる IP インタフェースがまだ存在していない場合は、インタフェースを作成します。**

   # ipadm create-ip *under-interface*

3. **IPMP グループへ IP インタフェースを追加します。**

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

   ここで、*ipmp-interface* はベースとなるインタフェースを追加する IPMP グループを表します。

例 3-4 IPMP グループへのインタフェースの追加

次の例では、net4 インタフェースを IPMP グループ ipmp0 に追加する方法を示します。

```
# 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
```

▼ IPMP グループからインタフェースを削除する方法

1. **root の役割になります。**

2. **IPMP グループから 1 つ以上のインタフェースを削除します。**

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

   ここで、*under-interface* は IPMP グループから削除する IP インタフェースを表し、*ipmp-interface* はベースとなるインタフェースを削除する IPMP グループを表します。
   単一のコマンドで、ベースとなるインタフェースを必要な数だけ削除できます。ベースとなるインタフェースをすべて削除しても、IPMP インタフェースは削除されません。代わりに、空の IPMP インタフェースまたはグループとして存在します。

例 3-5 IPMP グループからのインタフェースの削除

次の例では、net4 インタフェースを IPMP グループ ipmp0 から削除する方法を示します。

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

▼ IPMP グループに IP アドレスを追加する方法

IPMP グループに IP アドレスを追加するには、ipadm create-addr コマンドを使用します。IPMP 構成では、IP アドレスはデータアドレスまたは検査用アドレスのどちらかです。データアドレスは IPMP インタフェースに追加され、検査用アドレスは、IPMP インタフェースのベースとなるインタフェースに追加されます。次の手順では、検査用アドレスとデータアドレスのいずれかである IP アドレスを追加する方法について説明します。

1. **root の役割になります。**

2. **IP アドレスを IPMP グループに追加します。**

   ■ **次のように、IPMP グループにデータアドレスを追加します。**

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

IP アドレスを作成すると、アドレスオブジェクトが自動的に割り当てられます。アドレスオブジェクトは、IP アドレスの一意識別子です。アドレスオブジェクトの名前には、*interface*/*random-string* という命名規則が使用されます。したがって、データアドレスのアドレスオブジェクトの名前には、IPMP インタフェースが含まれることになります。

- **次のように、IPMP グループのベースとなるインタフェースに検査用アドレスを追加します。**

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

IP アドレスを作成すると、アドレスオブジェクトが自動的に割り当てられます。アドレスオブジェクトは、IP アドレスの一意識別子です。アドレスオブジェクトの名前には、*interface*/*random-string* という命名規則が使用されます。したがって、検査用アドレスのアドレスオブジェクトの名前には、ベースとなるインタフェースが含まれることになります。

▼ IPMP インタフェースから IP アドレスを削除する方法

IPMP グループから IP アドレスを削除するには、`ipadm delete-addr` コマンドを使用します。IPMP 構成では、データアドレスは IPMP インタフェースでホストされ、検査用アドレスはベースとなるインタフェースでホストされます。次の手順は、データアドレスと検査用アドレスのいずれかである IP アドレスを削除する方法を示しています。

1. **`root` の役割になります。**
2. **削除する IP アドレスを決定します。**
   - **次のように、データアドレスの一覧を表示します。**

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

   - **次のように、検査用アドレスの一覧を表示します。**

     ```
     # ipadm show-addr
     ```

     検査用アドレスを識別するアドレスオブジェクトの名前には、アドレスが構成されているベースとなるインタフェースが含まれています。
3. **IP アドレスを IPMP グループから削除します。**

■ **次のように、データアドレスを削除します。**

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

ここで、*addrobj* には IPMP インタフェースの名前が含まれている必要があります。入力したアドレスオブジェクトに IPMP インタフェース名が含まれていない場合、削除されるアドレスはデータアドレスではありません。

■ **次のように、検査用アドレスを削除します。**

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

ここで、正しい検査用アドレスを削除するには、*addrobj* にベースとなる正しいインタフェースの名前が含まれている必要があります。

例 3-6　インタフェースからの検査用アドレスの削除

次の例は、例3-2「アクティブ - スタンバイ IPMP グループの構成」に示したアクティブ - スタンバイ IPMP グループ `ipmp0` の構成を使用しています。この例では、ベースとなるインタフェース `net1` から検査用アドレスを削除します。

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

# ipadm delete-addr net1/v4
```

▼ インタフェースを 1 つの IPMP グループから別の IPMP グループに移動する方法

インタフェースが既存の IPMP グループに属している場合は、新しい IPMP グループにインタフェースを配置できます。この場合、現在の IPMP グループからインタフェースを削除する必要はありません。新しいグループに追加されたインタフェースは、既存の IPMP グループから自動的に削除されます。

1. **`root` の役割になります。**

2. **新しい IPMP グループへインタフェースを移動します。**

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

ここで、*under-interface* は移動するベースとなるインタフェースを表し、*ipmp-interface* はベースとなるインタフェースの移動先となる IPMP インタフェースを表します。

例 3-7 別の IPMP グループへのインタフェースの移動

次の例では、IPMP グループのベースとなるインタフェースは、`net0`、`net1`、および `net2` です。次の例は、IPMP グループ `ipmp0` から `net0` インタフェースを削除し、IPMP グループ `cs-link1` に `net0` を追加する方法を示しています。

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

▼ IPMP グループを削除する方法

この手順は、特定の IPMP グループが不要になったときに使用します。

1. **`root` の役割になります。**

2. **削除する IPMP グループおよびベースとなる IP インタフェースを特定します。**

   ```
   # ipmpstat -g
   ```

3. **IPMP グループに現在属している IP インタフェースをすべて削除します。**

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

   ここで、*under-interface* は削除するベースとなるインタフェースを表し、*ipmp-interface* はベースとなるインタフェースを削除する IPMP インタフェースを表します。

   注記 - IPMP インタフェースを正常に削除するには、その IPMP グループの一部として IP インタフェースが存在していてはいけません。

4. **IPMP インタフェースを削除します。**

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

   IPMP インタフェースを削除すると、そのインタフェースに関連付けられた IP アドレスもすべてシステムから削除されます。

例 3-8 IPMP インタフェースの削除

次の例は、ベースとなる IP インタフェース net0 および net1 を含むインタフェース ipmp0 を削除しています。

```
# 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
```

プローブベースの障害検出の構成

このセクションには、次のトピックが含まれています。

プローブベースの障害検出について

プローブベースの障害検出では、66 ページの「プローブベースの障害検出」で説明されているようにターゲットシステムを使用します。プローブベースの障害検出のターゲットを特定するとき、in.mpathd デーモンは、*ルーターターゲットモード*または*マルチキャストターゲットモード*の 2 つのうちどちらかのモードで動作します。ルーターターゲットモードでは、デーモンはルーティングテーブルに定義されたターゲットをプローブします。ターゲットが 1 つも定義されていない場合、このデーモンはマルチキャストターゲットモードで動作します。この場合、LAN 上の近くのホストをプローブするためにマルチキャストパケットが送出されます。

できれば、in.mpathd デーモンがプローブするターゲットシステムを設定するようにしてください。一部の IPMP グループでは、デフォルトルーターはターゲットとして十分です。ただし、一部の IPMP グループでは、検査信号ベースの障害検出用に特定のターゲットを構成したほうが良いこともあります。ターゲットを指定するには、ルーティングテーブル内にホストのルートをプローブターゲットとして設定します。ルーティングテーブルに構成されているすべてのホストルートは、デ

フォルトルーターの前に一覧化されます。IPMP はターゲットを選択するために、明示的に定義されたホストルートを使用します。したがって、デフォルトルーターを使用するのではなく、ホストのルートを設定して特定のプローブターゲットを構成するようにしてください。

ホストのルートをルーティングテーブルに設定するには、route コマンドを使用します。このコマンドで -p オプションを使用して、永続的なルートを追加できます。たとえば、route -p add はシステムのリブート後もルーティングテーブル内に残るルートを追加します。したがって、-p オプションを使用すると永続的なルートを追加できるため、システムが起動するたびにこれらのルートを作成し直す特殊なスクリプトは一切必要ありません。プローブベースの障害検出を最適なかたちで使用するには、プローブを受信するターゲットを必ず複数設定してください。

route コマンドは IPv4 ルートと IPv6 ルートの両方 (デフォルトでは IPv4 ルート) で動作します。route コマンドのすぐあとに -inet6 オプションを指定した場合、route コマンドは IPv6 ルート上で動作します。

92 ページの「プローブベースの障害検出のターゲットシステムを手動で指定する方法」に示した手順は、プローブベースの障害検出でターゲットへの永続的なルートを追加するために使用する正確な構文を示しています。route コマンドで使用できるオプションの詳細については、route(1M)のマニュアルページおよび82 ページの「IPMP の配備時にルーティングを維持」を参照してください。

プローブベースの障害検出のターゲットを選択するための要件

ネットワーク上で適切なターゲットとして使用できるホストを決定するために次の要件を参照してください。

- 予想されるターゲットが使用可能で、実行されていることを確認します。IP アドレスの一覧を作成します。
- ターゲットインタフェースが、構成中の IPMP グループと同じネットワークにあることを確認します。
- ターゲットシステムのネットマスクとブロードキャストアドレスは、IPMP グループ内のアドレスと同じでなければなりません。
- ターゲットホストは、プローブベースの障害検出を使用しているインタフェースからの ICMP 要求に応答できなければなりません。

障害検出手法の選択

デフォルトでは、プローブベースの障害検出は検査用アドレスを使用して実行されます。NIC ドライバがリンクベースの障害検出をサポートしている場合、この障害検出も自動的に有効になります。

NIC ドライバによってリンクベースの障害検出がサポートされている場合、この手法を無効にすることはできません。ただし、実装するプローブベースの障害検出のタイプは選択可能です。

プローブベースの検出手法を選択する前に、90 ページの「プローブベースの障害検出のターゲットを選択するための要件」に示された要件をプローブターゲットが満たしていることを確認してください。

推移的プローブのみを使用する場合は、次の手順を実行します。

1. SMF コマンドを使用して IPMP プロパティー `transitive-probing` を有効にします。

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

   このプロパティーの設定方法の詳細については、`in.mpathd`(1M)のマニュアルページを参照してください。
2. IPMP グループ用に構成された既存の検査用アドレスをすべて削除します。

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

   ここで、*addrobj* は検査用アドレスをホストしているベースとなるインタフェースを表す必要があります。

検査用アドレスを使用して障害をプローブする場合は、次の手順を実行します。

1. 必要に応じて、SMF コマンドを使用して推移的プローブを無効にします。

   ```
   # svccfg -s svc:/network/ipmp setprop config/transitive-probing=false
   # svcadm refresh  svc:/network/ipmp:default
   ```
2. IPMP グループのベースとなるインタフェースに検査用アドレスを割り当てます。

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

   ここで、*address* は CIDR 表記にすることができ、*under-interface* は IPMP グループのベースとなるインタフェースです。

▼ プローブベースの障害検出のターゲットシステムを手動で指定する方法

次の手順では、検査用アドレスを使用してプローブベースの障害検出を実装する場合に、特定のターゲットを追加する方法について説明します。

始める前に 90 ページの「プローブベースの障害検出のターゲットを選択するための要件」に示された要件をプローブのターゲットが満たしていることを確認してください。

1. **プローブベースの障害検出を構成するシステムにユーザーアカウントでログインします。**

2. **プローブベースの障害検出でターゲットとして使用される特定のホストにルートを追加します。**

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

   ここで、*destination-IP* と *gateway-IP* は、ターゲットとして使用されるホストの IPv4 アドレスです。たとえば、IPMP グループ `ipmp0` のインタフェースと同じサブネット上のターゲットシステム `192.168.10.137` を指定するには、次のように入力します。

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

   この新しいルートは、システムを再起動するたびに自動的に構成されます。プローブベースの障害検出用のターゲットシステムへの一時的なルートを定義するだけの場合は、`-p` オプションを使用しないでください。

3. **ターゲットシステムとして使用されるネットワーク上の追加ホストにルートを追加します。**

▼ IPMP デーモンの動作を構成する方法

IPMP グループに関連する次のシステム共通パラメータを設定するには、IPMP 構成ファイル `/etc/default/mpathd` を使用します。

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

1. **`root` の役割になります。**

2. **`/etc/default/mpathd` ファイルを編集します。**

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

手順については、pfedit(1M)のマニュアルページを参照してください。

次の 3 つのパラメータのうち、1 つ以上のデフォルト値を変更します。

- 次のように FAILURE_DETECTION_TIME パラメータの新しい値を入力します。

  ```
  FAILURE_DETECTION_TIME=n
  ```

  ここで、*n* は ICMP 検証がインタフェースの障害が発生していないかどうかを検出する時間 (秒単位) です。デフォルトは 10 秒です。

- 次のように FAILBACK パラメータの新しい値を入力します。

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

  yes

  IPMP のフェイルバック動作のデフォルトです。障害が発生したインタフェースの修復が検出されると、ネットワークアクセスはこの修復されたインタフェースに復帰します。詳細は、69 ページの「物理インタフェースの回復検出」を参照してください。

  no

  データトラフィックが、修復されたインタフェースに戻らないことを示します。障害が発生したインタフェースの回復が検出されると、そのインタフェースに INACTIVE フラグが設定されます。このフラグは、現時点でそのインタフェースをデータトラフィックに使用すべきでないことを示します。ただし、そのインタフェースを検査信号トラフィックに使用することはできます。

  たとえば、IPMP グループ ipmp0 が 2 つのインタフェース net0 と net1 から構成されているとします。/etc/default/mpathd ファイルで FAILBACK=no パラメータが設定されています。net0 が故障すると、FAILED としてフラグが付けられて使用不可能になります。修復後、このインタフェースは INACTIVE としてフラグが付けられ、値 FAILBACK=no が設定されているため、使用不可能なままとなります。

  net1 が故障し、net0 のみが INACTIVE 状態である場合には、net0 の INACTIVE フラグが解除され、このインタフェースが使用可能になります。IPMP グループに同じく INACTIVE 状態のインタフェースがほかにも含まれている場合、net1 の障害発生時にそれらの INACTIVE

インタフェースのいずれか 1 つ (必ずしも net0 とはかぎらない) がクリアーされ、使用可能となります。

- 次のように TRACK_INTERFACES_ONLY_WITH_GROUPS パラメータの新しい値を入力します。

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

yes
: IPMP 動作のデフォルトです。この値を指定した場合、IPMP は、IPMP グループ内に構成されていないネットワークインタフェースを無視します。

no
: すべてのネットワークインタフェース (IPMP グループ内に構成されているかどうかは無関係) で障害と回復を検出するように設定します。ただし、IPMP グループ内に構成されていないインタフェースで障害や修復が検出されても、IPMP では、そのインタフェースのネットワーク機能を維持するためアクションは一切起動されません。したがって、値 no を指定することは、障害の報告には役立ちますが、ネットワークの可用性を直接向上させることはありません。

  このパラメータと匿名グループ機能については、69 ページの「障害検出と匿名グループ機能」を参照してください。

3. **in.mpathd デーモンを再起動します。**

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

デーモンは、新しいパラメータ値が有効になった状態で再起動します。

IPMP 情報のモニタリング

次の各例は、ipmpstat コマンドを使用してシステム上の IPMP グループのさまざまな側面をモニタリングする方法を示しています。IPMP グループ全体のステータスやそのベースとなる IP インタフェースのステータスを監視できます。IPMP グループのデータアドレスや検査用アドレスの構成を確認することもできます。また、同じコマンドを使用して障害検出に関する情報を取得できます。詳細は、ipmpstat(1M)のマニュアルページを参照してください。

ipmpstat コマンドを使用する場合はデフォルトで、80 列に収まるもっとも意味のあるフィールドが表示されます。出力では、ipmpstat - コマンドが p オプション付きで使用されている場合を除き、ipmpstat コマンドで使用したオプションに固有のフィールドがすべて表示されます。

デフォルトでは、ホスト名が存在する場合、数値 IP アドレスではなくホスト名が出力に表示されます。出力に数値 IP アドレスを表示するには、-n オプションを、IPMP グループの特定の情報を表示するためのほかのオプションとともに使用します。

注記 - 次の各例では、特に明記していないかぎり、ipmpstat コマンドの使用時にシステム管理者の特権は必要ありません。

---

ipmpstat コマンドに次のオプションを付けて使用することで、必要な情報を表示できます。

-g　システム上の IPMP グループに関する情報を表示します。例3-9「IPMP グループ情報の取得」を参照してください。

-a　IPMP グループに構成されているデータアドレスを表示します。例 3-10「IPMP のデータアドレス情報の取得」を参照してください。

-i　IPMP 構成に関連する IP インタフェースに関する情報を表示します。例 3-11「IPMP グループのベースとなる IP インタフェースに関する情報の取得」を参照してください。

-t　障害検出に使用されるターゲットシステムに関する情報を表示します。このオプションは、IPMP グループで使用される検査用アドレスも表示します。例3-12「IPMP のプローブターゲット情報の取得」を参照してください。

-p　障害検出に使用されているプローブに関する情報を表示します。例 3-13「IPMP のプローブの監視」を参照してください。

次の追加例では、ipmpstat コマンドを使用してシステムの IPMP 構成に関する情報を表示する方法を示します。

例 3-9　IPMP グループ情報の取得

-g オプションは、システム上のさまざまな IPMP グループのステータスを、そのベースとなるインタフェースのステータスも含めて表示します。ある特定のグループでプローブベースの障害検出が有効になっていると、コマンドはそのグループの障害検出時間も含めます。

```
% 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]
```

出力フィールドでは、次の情報が提供されます。

GROUP
: IPMP インタフェースの名前を指定します。匿名グループの場合、このフィールドは空になります。匿名グループの詳細については、in.mpathd(1M)のマニュアルページを参照してください。

GROUPNAME
: IPMP グループの名前を指定します。匿名グループの場合、このフィールドは空になります。

STATE
: IPMP グループの現在のステータスを示します。ステータスは次のいずれかになります。

- ok - IPMP グループのベースとなるインタフェースがすべて使用可能であることを示します。
- degraded - グループ内のベースとなるインタフェースの一部が使用不可能であることを示します。
- failed - グループのインタフェースがすべて使用不可能であることを示します。

FDT
: 障害検出が有効になっている場合に、その障害検出時間を指定します。障害検出が無効になっている場合、このフィールドは空になります。

INTERFACES
: IPMP グループに属するベースとなるインタフェースを指定します。このフィールドでは、まずアクティブなインタフェースが表示され、次にアクティブでないインタフェースが表示され、最後に使用不可能なインタフェースが表示されます。インタフェースのステータスはその表示形式によって示されます。

- *interface* (丸括弧や角括弧なし) - アクティブなインタフェースを示します。アクティブなインタフェースは、システムでデータトラフィックの送受信に使用されています。
- *(interface)* (丸括弧付き) - 機能しているがアクティブではないインタフェースを示します。このインタフェースは、管理ポリシーの定義に従って未使用になっています。
- *[interface]* (角括弧付き) - インタフェースに障害が発生しているかオフラインになっているために使用不可能であることを示します。

例 3-10 IPMP のデータアドレス情報の取得

-a オプションは、データアドレスと各アドレスの所属先 IPMP グループを表示します。表示される情報には、使用可能なアドレス (ipadm [up-addr/down-addr] コマンドによって切り替えられたかどうかによって変わる) も含まれます。また、あるアドレスがどのインバウンドまたはアウトバウンドインタフェース上で使用できるかも判定できます。

```
% 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        --          --
```

出力フィールドでは、次の情報が提供されます。

ADDRESS　　-n オプションが -a オプションとともに使用された場合は、ホスト名またはデータアドレスを示します。

STATE　　IPMP インタフェースのアドレスが、up したがって使用可能、down したがって使用不可能、のいずれであるかを示します。

GROUP　　特定のデータアドレスをホストする IPMP インタフェースを示します。Oracle Solaris では通常、IPMP グループの名前は IPMP インタフェースです。

INBOUND　　特定のアドレスのパケットを受信するインタフェースを識別します。このフィールドの情報は、外部のイベントに応じて変わる可能性があります。たとえば、データアドレスが停止している場合や、IPMP グループ内にアクティブな IP インタフェースが 1 つも残っていない場合、このフィールドは空になります。空のフィールドは、この特定のアドレス宛ての IP パケットをシステムが受け入れていないことを示します。

OUTBOUND　　特定のアドレスを発信元アドレスとして使用したパケットを送信するインタフェースを識別します。INBOUND フィールドと同様に、OUTBOUND の情報も外部のイベントに応じて変わる可能性があります。空のフィールドは、システムがこの特定の発信元アドレスでパケットを送信していないことを示します。このフィールドが空である場合、それは、アドレスが停止しているか、またはグループ内にアクティブな IP インタフェースが 1 つも残っていないためです。

例 3-11　IPMP グループのベースとなる IP インタフェースに関する情報の取得

-i オプションは、IPMP グループのベースとなる IP インタフェースに関する情報を表示します。

```
% 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
```

出力フィールドでは、次の情報が提供されます。

INTERFACE　各 IPMP グループのベースとなる各インタフェースを示します。

ACTIVE　このインタフェースが機能していて使用中 (yes) であるか、それともそうではない (no) かを示します。

GROUP　IPMP インタフェースの名前を指定します。匿名グループの場合、このフィールドは空になります。匿名グループの詳細については、in.mpathd(1M)のマニュアルページを参照してください。

FLAGS　ベースとなる各インタフェースのステータスを示し、これは次のいずれか、またはその任意の組み合わせになります。

- b - このインタフェースがこの IPMP グループのブロードキャストトラフィックの受信用として、システムにより指定されていることを示します。
- d - このインタフェースが停止しており、したがって使用不可能であることを示します。
- h - このインタフェースが重複する物理ハードウェアアドレスを別のインタフェースと共有しており、オフラインになっていることを示します。h フラグは、このインタフェースが使用不可能であることを示します。
- i - このインタフェースに INACTIVE フラグが設定されていることを示します。したがって、このインタフェースはデータトラフィックの送受信に使用されません。
- m - このインタフェースがこの IPMP グループの IPv4 マルチキャストトラフィックの送受信用として、システムにより指定されていることを示します。
- M - このインタフェースがこの IPMP グループの IPv6 マルチキャストトラフィックの送受信用として、システムにより指定されていることを示します。
- s - このインタフェースがスタンバイインタフェースとして構成されていることを示します。

LINK　リンクベースの障害検出のステータスを示し、これは次のいずれかになります。

- up または down - リンクの使用可能または使用不可能を示します。
- unknown - リンクが up、down のどちらであるかの通知をドライバがサポートしておらず、したがってドライバがリンクの状態変化を検出しないことを示します。

PROBE

検査用アドレスが構成されたインタフェースについて、次のようにプローブベースの障害検出の状態を示します。

- ok - プローブが機能しており、アクティブであることを示します。
- failed - プローブベースの障害検出が、このインタフェースが動作していないことを検出したことを示します。
- unknown - 適切なプローブターゲットが見つからなかったため、プローブを送信できないことを示します。
- disabled - このインタフェースでは IPMP 検査用アドレスが構成されていないことを示します。したがって、プローブベースの障害検出は無効になっています。

STATE

次のように、このインタフェース全体の状態を指定します。

- ok - このインタフェースがオンラインになっており、障害検出手法の構成に基づいて正常に動作していることを示します。
- failed - このインタフェースのリンクが停止しているか、またはこのインタフェースはトラフィックを送受信できないとプローブ検出が判定したために、このインタフェースが動作していないことを示します。
- offline - このインタフェースが使用不可能であることを示します。通常、次の状況の下ではインタフェースがオフラインになります。
  - インタフェースがテスト中である。
  - 動的再構成が実行中である。
  - このインタフェースが、重複するハードウェアアドレスを別のインタフェースと共有している。
- unknown - プローブベースの障害検出のプローブターゲットが見つからなかったために IPMP インタフェースの状態を判定できないことを示します。

例 3-12 IPMP のプローブターゲット情報の取得

-t オプションは、IPMP グループ内の各 IP インタフェースに関連付けられたプローブターゲットを特定します。次の例の出力は、プローブベースの障害検出に検査用アドレスを使用する IPMP 構成を示しています。

```
% 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
```

次の出力は、推移的プローブまたはプローブベースの障害検出を検査用アドレスなしで使用する IPMP 構成を示しています。

```
% 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
```

出力フィールドでは、次の情報が提供されます。

`INTERFACE` IPMP グループのベースとなる各インタフェースを示します。

`MODE` プローブターゲットを取得するための方法を指定します。

- `routes` - プローブターゲットの検索にシステムのルーティングテーブルが使用されることを示します。
- `mcast` - ターゲットの検索にマルチキャスト ICMP プローブが使用されることを示します。
- `disabled` - このインタフェースでプローブベースの障害検出が無効になっていることを示します。
- `transitive` - 2 番目の例に示したように、障害検出に推移的プローブが使用されることを示します。推移的プローブと検査用アドレスを同時に使用してプローブベースの障害検出を実装することはできない点に注意してください。検査用アドレスを使用しない場合は、推移的プローブを有効にする必要があります。推移的プローブを使用しない場合は、検査用アドレスを構成する必要があります。概要については、66 ページの「プローブベースの障害検出」を参照してください。

TESTADDR　ホスト名、またはプローブの送受信のためにこのインタフェースに割り当てられた IP アドレス (-n オプションが -t オプションとともに使用された場合) を示します。

推移的プローブが使用された場合、インタフェースの名前は、データ受信用としてアクティブに使用されていない、ベースとなる IP インタフェースを表します。また、これらの名前は、指定されたこれらのインタフェースの発信元アドレスを使用して推移的テストプローブが送信されていることも示しています。データを受信するアクティブなベースとなる IP インタフェースの場合、表示される IP アドレスは、送信 ICMP プローブの発信元アドレスを示します。

注記 - IP インタフェースに IPv4 検査用アドレスと IPv6 検査用アドレスの両方が構成されている場合、プローブターゲットの情報は各検査用アドレスについて個別に表示されます。

TARGETS　現在のプローブターゲットを空白区切りリストとして一覧表示します。プローブターゲットは、ホスト名と IP アドレスのどちらかで表示されます。-n オプションが -t オプションとともに使用された場合は、IP アドレスが表示されます。

例 3-13　IPMP のプローブの監視

-p オプションを使用すると、進行中のプローブを監視できます。ipmpstat コマンドにこのオプションを使用すると、Control-C を押してコマンドを終了するまで、システム上のプローブのアクティビティーに関する情報が継続的に表示されます。このコマンドを実行するには、root 役割になるか、適切な権限が必要です。

次に示すのは、プローブベースの障害検出に検査用アドレスを使用する IPMP 構成の例です。

```
# 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
```

次に示すのは、推移的プローブまたはプローブベースの障害検出を検査用アドレスなしで使用する IPMP 構成の例です。

```
# 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
```

出力フィールドでは、次の情報が提供されます。

TIME　　ipmpstat コマンドが発行された時点を基準にしたプローブの送信時間を指定します。ipmpstat が開始される前にプローブが起動された場合、コマンドが発行された時点を基準にした負の値で時間が表示されます。

INTERFACE　　プローブの送信先となるインタフェースを指定します。

PROBE　　プローブを表す識別子を指定します。障害検出に推移的プローブが使用される場合は、この識別子に、推移的プローブの場合は t という接頭辞が、ICMP プローブの場合は i という接頭辞が付きます。

NETRTT　　プローブの合計ネットワーク往復時間を示します (ミリ秒で測定)。NETRTT は、IP モジュールがプローブを送信した時点から、IP モジュールがターゲットから ack パケットを受け取った時点までの時間をカバーします。プローブが失われたと in.mpathd デーモンが判定した場合、このフィールドは空になります。

RTT　　プローブの合計往復時間を示します (ミリ秒で測定)。RTT は、in.mpathd デーモンがプローブを送信するコードを実行した時点から、デーモンがターゲットからの ack パケットの処理を完了した時点までの時間が対象になります。プローブが失われたとデーモンが判定した場合、このフィールドは空になります。NETRTT に存在しないスパイクが RTT で発生する場合、それは、ローカルシステムが過負荷状態であることを示している可能性があります。

RTTAVG　　ローカルシステムとターゲットの間における、このインタフェース経由でのプローブの平均往復時間を示します。平均往復時間は、低速なターゲットの特定に役立ちます。データが不十分で平均を計算できない場合、このフィールドは空になります。

TARGET　　ホスト名を指定します。または -n オプションが -p とともに使用された場合は、プローブの送信先となるターゲットアドレスを示します。

ipmpstat コマンドの出力のカスタマイズ

-o オプションを使用すると、ipmpstat コマンドの出力をカスタマイズできます。前述した ipmpstat のほかのオプションとともにこのオプションを使用して、メインオプションで通常表示される合計フィールドの中から、表示する特定のフィールドを選択します。

たとえば、-g オプションでは次の情報が提供されます。

- IPMP グループ
- IPMP グループ名
- グループのステータス
- 障害検出時間
- IPMP グループのベースとなるインタフェース

システム上の IPMP グループのステータスだけを表示するとします。次の例に示すように、-o オプションと -g オプションを組み合わせ、フィールド groupname および state を指定します。

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

特定の種類の情報に関してのみ、ipmpstat コマンドのすべてのフィールドを表示するには、-o オプションに all 引数を指定します。

スクリプト内での ipmpstat コマンドの使用

-oオプションは、スクリプトから、またはコマンド別名を使用して ipmpstat コマンドを実行する場合で、特にマシン解析可能な出力を生成する必要がある場合に便利です。

マシンによる解析が可能な情報を生成するには、-P オプションと -o オプションを ipmpstat のほかのメインオプションの 1 つに組み合わせ、表示する特定のフィールドと一緒に指定します。

マシンによる解析が可能な出力は、次の点で通常の出力とは異なります。

- 列ヘッダーは省略されます。
- 各フィールドがコロン (:) で区切られます。
- 空の値を含むフィールドは、二重ダッシュ (--) が設定されるのではなく、空になります。
- 複数のフィールドが要求された場合で、あるフィールドにリテラルのコロン (:) またはバックスラッシュ (\) が含まれている場合は、これらの文字の前にバックスラッシュ (\) を付けることによって、これらの文字をエスケープまたは除外できます。

ipmpstat -P コマンドを正しく使用するには、次の規則に従います。

- -o *option fields* を -P オプションとともに使用する。複数のオプションフィールドはコンマで区切る。

- `-o all` オプションを `-P` オプションとともに使用しない。

注意 - これらの規則のいずれかを無視した場合は、`ipmpstat -P` が失敗します。

次の例は、`-P` オプションを使用するための正しい構文を示しています。

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

グループ名、障害検出時間、およびベースとなるインタフェースは、グループ情報フィールドです。したがって、`-o` および `-g` オプションを `-P` オプションとともに使用します。

`-P` オプションは、スクリプト内で使用するためのものです。次の例は、`ipmpstat` コマンドをスクリプトから実行する方法を示しています。このスクリプトは、IPMP グループの障害検出時間を表示します。

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

♦ ♦ ♦ 第 4 章

4 IP トンネルの管理について

この章では、Oracle Solaris での IP トンネル管理の概要を示します。タスク関連の情報については、第5章「IP トンネルの管理」を参照してください。

この章の内容は、次のとおりです。

IP トンネル管理の新機能

IP トンネルの管理は、Oracle Solaris 11 のネットワークデータリンクの管理に使用される新しいモデルに合わせて修正されました。このリリースでは、`dladm` コマンドを使用して、IP トンネルを作成および構成します。トンネルでは、このリリースでサポートされたほかのデータリンク機能も使用できます。たとえば、管理者によって選択された名前がサポートされたため、意味のあるトンネル名を割り当てることができます。詳細は、`dladm`(1M)のマニュアルページを参照してください。

IP トンネルの機能のサマリー

IP トンネル (このドキュメントでは簡単にトンネルと呼ぶこともある) は、ドメイン内のプロトコルが中間のネットワークでサポートされないときにドメイン間でデータパケットを転送するための手段を提供します。たとえば、大部分のネットワークで IPv4 プロトコルが使用されている環境では、IPv6 ネットワークは自身の境界の外側で通信する方法を必要とします。この通信を可能にするのがトンネルです。IP トンネルは、IP を使用して到達可能な 2 つのノード間で仮想リンクを実現します。したがって、このリンクを使用すれば IPv4 ネットワーク経由で IPv6 パケットを転送でき、2 つの IPv6 サイト間での IPv6 通信を実現できます。

トンネルのタイプ

トンネリングでは、IP パケットを別のパケット内にカプセル化する必要があります。このカプセル化によって、パケットは、パケットのプロトコルをサポートしない中間のネットワークを介して宛先に到達できます。トンネルは、使用されるパケットカプセル化のタイプによって異なります。

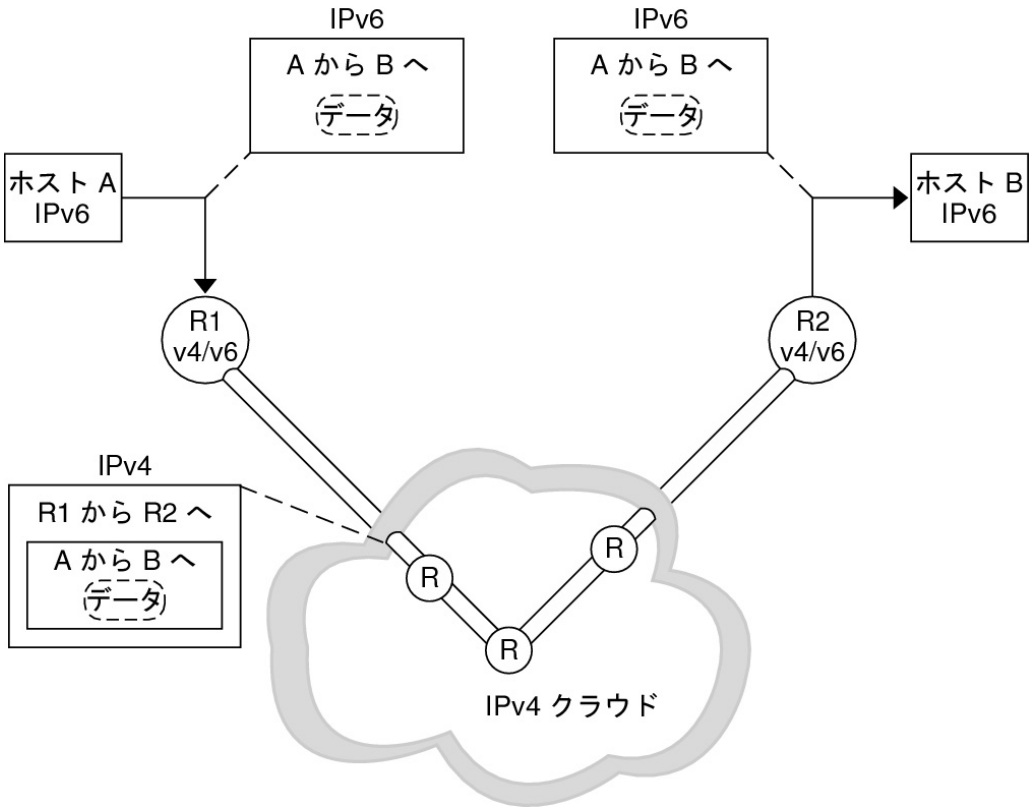
Oracle Solaris でサポートされるトンネルのタイプは、次のとおりです。

- **IPv4 トンネル** – IPv4 パケットが IPv4 ヘッダー内にカプセル化され、事前に構成されたユニキャスト IPv4 宛先に送信されます。トンネルを通過するパケットをより具体的に示すため、IPv4 トンネルは *IPv4 over IPv4 トンネル*または *IPv6 over IPv4 トンネル*とも呼ばれます。
- **6to4 トンネル** – IPv6 パケットが IPv4 ヘッダー内にカプセル化され、パケット単位で自動的に決定される IPv4 宛先に送信されます。この決定は、*6to4 プロトコル*内に定義されたアルゴリズムに基づきます。
- **IPv6 トンネル** – IPv6 パケットが IPv4 ヘッダー内にカプセル化され、パケット単位で自動的に決定される IPv4 宛先に送信されます。この決定は、6to4 プロトコル内に定義されたアルゴリズムに基づきます。

IPv6 と IPv4 を組み合わせたネットワーク環境でのトンネル

IPv6 ドメインを持つ多くのサイトは、IPv6 配備の初期フェーズにおいては、ほかの IPv6 ドメインと通信する際に IPv4 ネットワークを通過しなければならない場合があります。次の図に、IPv4 ルーターを介した 2 台の IPv6 ホスト間のトンネルメカニズム (図の「R」) を示します。

図 4-1 IPv6 トンネルメカニズム



上の図で、トンネルは 2 台のルーターから成り、IPv4 ネットワーク上の 2 台のルーター間の仮想のポイントツーポイントリンクが構成されています。

IPv6 パケットが IPv4 パケット内にカプセル化されます。IPv6 ネットワークの境界ルーターは、宛先 IPv6 ネットワークの境界ルーターに向かうさまざまな IPv4 ネットワークにポインツーポイントトンネルを設定します。パケットはトンネル経由で宛先の境界ルーターに転送され、そこでそのカプセル化が解除されます。次に、そのルーターは個々の IPv6 パケットを宛先のノードに転送します。

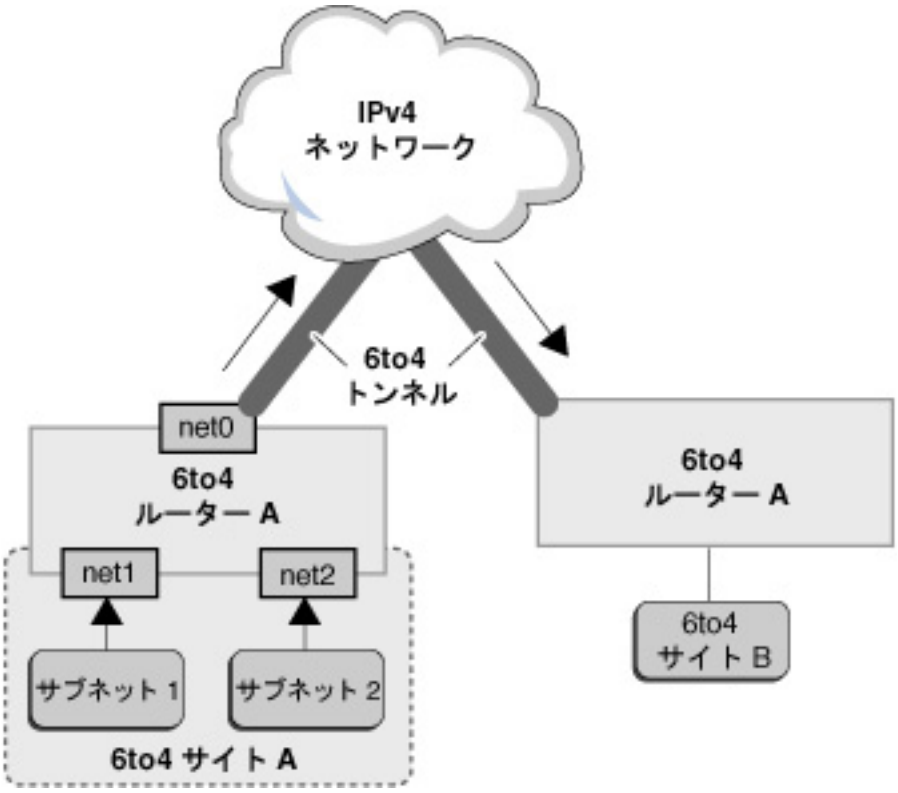
6to4 トンネル

Oracle Solaris には、IPv4 アドレス指定から IPv6 アドレス指定に移行するための暫定的な手段として 6to4 トンネルが含まれています。*6to4 トンネル*を使用すると、孤立した IPv6 サイトが、IPv6 をサポートしない IPv4 ネットワーク上の自動トンネルを介して通信できます。6to4 トンネルを使用するには、6to4 自動トンネルの一方のエンドポイントとして、境界ルーターを IPv6 ネットワーク上に構成する必要があります。そのあと、この 6to4 ルーターをほかの 6to4 サイトとの間のトンネルの構成要素として使用することも、あるいは必要に応じて 6to4 以外のネイティブ IPv6 サイトとの間のトンネルで使用することもできます。

6to4 トンネルのトポロジ

6to4 トンネルは、あらゆる場所にあるすべての 6to4 サイトに IPv6 接続を提供します。同様に、リレールーターに転送するようにトンネルが構成されている場合、トンネルはネイティブ IPv6 インターネットも含むすべての IPv6 サイトへのリンクとしても機能します。次の図は、6to4 トンネルが 6to4 サイト間にこの接続を提供する仕組みを示しています。

図 4-2 2 つの 6to4 サイト間のトンネル



上図には、孤立した 2 つの 6to4 ネットワークとしてサイト A とサイト B が描かれています。各サイトでは、IPv4 ネットワークへの外部接続を備えたルーターが構成されています。IPv4 ネットワークを越える 6to4 トンネルによって、6to4 サイトをリンクする接続が提供されています。

IPv6 サイトを 6to4 サイトにするには、6to4 をサポートするために最低 1 つのルーターインタフェースを構成する必要があります。このインタフェースは、IPv4 ネットワークに対する外部接続を提供する必要があります。前の図では、境界ルーター A のインタフェース `net0` がサイト A を IPv4 ネットワークに接続しています。`net0` で構成するアドレスは、一意 (世界で 1 つ) のものでなければなりません。ルーター上で 6to4 をサポートするようにトンネルインタフェースを構成する前に、IPv4 アドレスで `net0` インタフェースを構成する必要があります。

この図の 6to4 サイト A は、ルーター A のインタフェース `net1` と `net2` に接続された 2 つのサブネットから構成されています。サイト A のどちらかのサブネット上の IPv6 ホストはすべて、ルーター A からの広告の受信時に 6to4 派生アドレスで自動的に再構成されます。

サイト B は、もう 1 つの独立した 6to4 サイトです。サイト A からトラフィックを正しく受け取るには、サイト B 側の境界ルーターを 6to4 をサポートするように構成する必要があります。それ以外の場合、ルーターがサイト A から受け取るパケットが認識されずに削除されてしまいます。

`6to4relay` コマンド

6to4 トンネリングは、孤立した 6to4 サイト間の通信を可能にします。しかし、6to4 以外のネイティブ IPv6 サイトにパケットを転送する場合は、6to4 ルーターは 6to4 リレールーターとのトンネルを確立する必要があります。このトンネルが確立されると、「*6to4 リレールーター*」によって 6to4 パケットが IPv6 ネットワークに転送され、最終的にネイティブ IPv6 サイトに送信されます。6to4 有効化サイトがネイティブな IPv6 サイトとデータを交換する必要がある場合、`6to4relay` コマンドを使用して、適切なトンネルを有効にします。

注記 - リレールーターの使用はセキュアではないため、Oracle Solaris のデフォルト設定ではリレールーターとの間のトンネリングは無効になっています。このシナリオを実践に移す場合は、6to4 リレールーターとの間のトンネル構築に伴って発生する問題点をあらかじめ慎重に検討してください。詳細は、111 ページの「6to4 リレールーターとの間のトンネルを有効にする際の考慮事項」を参照してください。6to4 リレールーターのサポートを有効にする場合、その関連手順については、118 ページの「IP トンネルを作成および構成する方法」を参照してください。

詳細は、**`6to4`**(7M) のマニュアルページを参照してください。

6to4 トンネルを介したパケットフロー

このセクションでは、ある 6to4 サイトにあるホストから、リモートの 6to4 サイトにあるホストまでのパケットのフローについて説明します。このシナリオでは、図4-2「2 つの 6to4 サイト間のトンネル」で使用したトポロジを使用します。このシナリオは、6to4 ルーターと 6to4 ホストがすでに構成済みであることを想定しています。

パケットフローは次のとおりです。

1. 6to4 サイト A のサブネット 1 に存在するホストが伝送を行い、6to4 サイト B 上のホストが宛先として機能します。各パケットヘッダーには、送信元の 6to4 派生アドレスと宛先の 6to4 派生アドレスが含まれます。

2. サイト A のルーターは、IPv4 ヘッダー内で各 6to4 パケットをカプセル化します。このプロセスでルーターは、カプセル化ヘッダーの IPv4 宛先アドレスを、サイト B のルーターアドレスに設定します。トンネルインタフェースを通過する各 IPv6 パケットの IPv6 宛先アドレスには、この IPv4 宛先アドレスも含まれています。したがって、ルーターはカプセル化ヘッダーに設定されている IPv4 宛先アドレスを特定することができます。続いてサイト A のルーターは、標準の IPv4 ルーティング手続きを使用し IPv4 ネットワークを介してこのパケットを転送します。
3. パケットが遭遇する IPv4 ルーターが、パケットの IPv4 宛先アドレスを使用して転送を行います。このアドレスはルーター B のインタフェースに使用される一意の (世界に 1 つしかない) IPv4 アドレスであり、6to4 擬似インタフェースとしても機能します。
4. サイト A から送付されたパケットがルーター B に到着します。ルーター B は、IPv4 ヘッダーを削除して IPv6 パケットのカプセル化を解除します。
5. 続いてルーター B は、IPv6 パケット内の宛先アドレスを使用してサイト B の受信ホストにパケットを転送します。

6to4 リレールーターとの間のトンネルを有効にする際の考慮事項

6to4 リレールーターは、6to4 ではない ネイティブ IPv6 ネットワークと通信を行う必要がある 6to4 ルーターからのトンネルのエンドポイントとして機能します。本来、リレールーターは 6to4 サイトとネイティブ IPv6 サイトとの間のブリッジとして使用されます。この解決方法はセキュアではない場合があるため、Oracle Solaris のデフォルト設定では 6to4 リレールーターのサポートは無効になっています。しかし、サイトでこのようなトンネルが必要な場合には `6to4relay` コマンドを使用して次の図に示すようなトンネリングを有効にできます。

図 4-3 6to4 サイトと 6to4 リレールーター間のトンネル

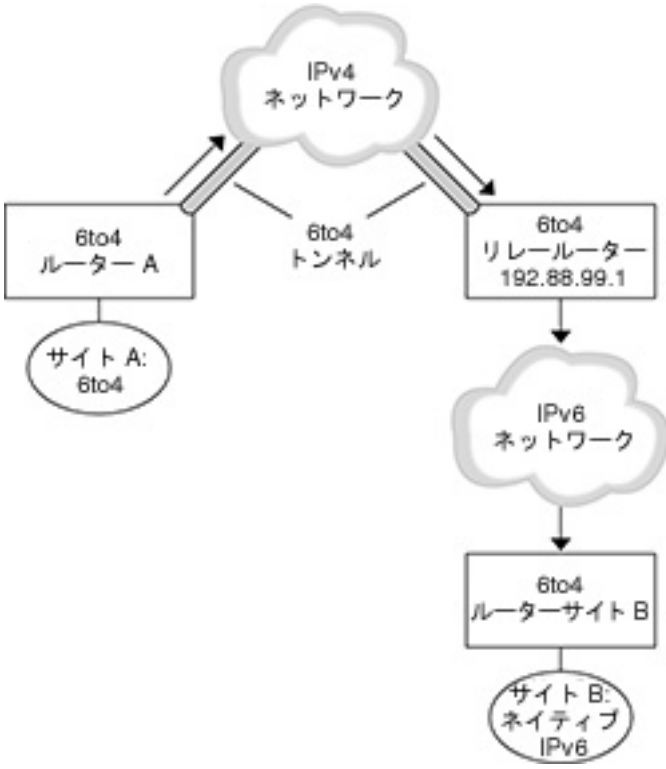


図4-3「6to4 サイトと 6to4 リレールーター間のトンネル」の 6to4 サイト A は、ネイティブ IPv6 サイト B のノードと通信する必要があります。図には、IPv4 ネットワーク経由でサイト A から 6to4 トンネルに向かうトラフィックのパスが示されています。このトンネルは、6to4 ルーター A と 6to4 リレールーターをエンドポイントとして使用しています。6to4 リレールーターより先は IPv6 ネットワークであり、IPv6 サイト B はこのネットワークに接続されています。

6to4 サイトとネイティブ IPv6 サイト間のパケットフロー

このセクションでは、6to4 サイトからネイティブな IPv6 サイトまでのパケットフローについて説明します。このシナリオでは、図4-3「6to4 サイトと 6to4 リレールーター間のトンネル」で使用したトポロジを使用します。

パケットフローは次のとおりです。

1. 6to4 サイト A のホストが、ネイティブ IPv6 サイト B のホストを宛先に指定して伝送を行います。各パケットヘッダーの発信元アドレスには 6to4 派生アドレスが含まれています。宛先アドレスは標準の IPv6 アドレスです。
2. サイト A の 6to4 ルーターは、各パケットを宛先である 6to4 ルーターの Ipv4 アドレスを持つ IPv4 ヘッダー内でカプセル化します。この 6to4 ルーターは、標準の IPv4 ルーティング手続きを使用し IPv4 ネットワークを介してこのパケットを転送します。パケットが遭遇する IPv4 ルーターが、6to4 リレールーターにパケットを転送します。
3. サイト A に物理的にもっとも近いエニーキャスト 6to4 リレールーターが、`192.88.99.1` エニーキャストグループ宛てのパケットを取り出します。

注記 - 6to4 リレールーターエニーキャストグループの一部である 6to4 リレールーターには、`192.88.99.1` という IP アドレスが割り当てられます。このエニーキャストアドレスは、6to4 リレールーターのデフォルトアドレスです。特定の 6to4 リレールーターを使用する必要がある場合は、デフォルトをオーバーライドしてそのルーターの IPv4 アドレスを指定できます。

---

4. このリレールーターは、IPv4 ヘッダーを取り除いて 6to4 パケットのカプセル化を解除し、ネイティブ IPv6 宛先アドレスを明らかにします。
5. 次に、リレールーターが IPv6 のみとなったパケットを IPv6 ネットワークに送信し、そこでサイト B のルーターがそのパケットを最終的に受け取ります。次に、ルーターがそのパケットを宛先の IPv6 ノードに転送します。

IP トンネルの配備について

IP トンネルを正しく配備するには、2 つの主要タスクを実行する必要があります。まず、トンネルリンクを作成してから、トンネル上で IP インタフェースを構成します。次に、トンネルとそれらに対応する IP インタフェースを作成するための要件について説明します。

IP トンネルを作成するための要件

IP トンネルを正常に作成するには、次の要件を遵守する必要があります。

- リテラル IP アドレスの代わりにホスト名を使用する場合、それらの名前はトンネルのタイプと互換性のある有効な IP アドレスに解決される必要があります。

- 作成する IPv4 または IPv6 トンネルは、構成済みの別のトンネルと同じトンネル発信元アドレスとトンネル着信先アドレスを共有することはできません。
- 作成する IPv4 または IPv6 トンネルは、既存の 6to4 トンネルと同じトンネル発信元アドレスを共有することはできません。
- 6to4 トンネルを作成する場合、そのトンネルは、構成済みの別のトンネルと同じトンネル発信元アドレスを共有することはできません。

詳細は、『Oracle Solaris 11.2 でのネットワーク配備の計画 』の「ネットワークでのトンネル使用の計画」を参照してください。

IP トンネルと IP インタフェースの要件

各トンネルタイプは、そのトンネル上で構成される IP インタフェースに対して独自の IP アドレス要件を持ちます。これらの要件は次の表に要約されています。

表 4-1 トンネルと IP インタフェースの要件

トンネルタイプ	トンネル上で許可される IP インタフェース	IP インタフェースの要件
IPv4 トンネル	IPv4 インタフェース	ローカルとリモートのアドレスは手動で指定されます。
	IPv6 インタフェース	`ipadm create-addr -T addrconf` コマンドの発行時に、ローカルとリモートのリンクローカルアドレスが自動的に設定されます。ipadm(1M)のマニュアルページを参照してください。
IPv6 トンネル	IPv4 インタフェース	ローカルとリモートのアドレスは手動で指定されます。
	IPv6 インタフェース	`ipadm create-addr -T addrconf` コマンドの発行時に、ローカルとリモートのリンクローカルアドレスが自動的に設定されます。ipadm(1M)のマニュアルページを参照してください。
6to4 トンネル	IPv6 インタフェースのみ	`ipadm create-ip` コマンドの発行時に、デフォルトの IPv6 アドレスが自動的に選択されま

トンネルタイプ	トンネル上で許可されるIP インタフェース	IP インタフェースの要件
		す。ipadm(1M)のマニュアルページを参照してください。

IPv6 または IPv4 トンネル上の IPv6 インタフェースに自動的に設定されたリンクローカルアドレスをオーバーライドするには、`ipadm create-addr -T addrconf` コマンドにローカルおよびリモートの `interface-id` を指定します。

♦ ♦ ♦ 第 5 章

5 IP トンネルの管理

この章では、Oracle Solaris で、IP トンネルを管理するためのタスクについて説明します。Oracle Solaris での IP トンネルの管理の概要については、第4章「IP トンネルの管理について」を参照してください。

この章の内容は、次のとおりです。

Oracle Solaris での IP トンネルの管理

Oracle Solaris のこのリリースでは、トンネル管理が IP インタフェース構成から切り離されています。IP トンネルのデータリンクの側面を管理するには `dladm` コマンドを使用し、IP (IP トンネルを含む) の構成の側面を管理するには `ipadm` コマンドを使用します。

IP トンネルを構成するには、`dladm` の次のサブコマンドを使用します。

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

詳細は、`dladm`(1M)のマニュアルページを参照してください。

注記 - IP トンネルの管理は、IPsec の構成と密接に関連しています。たとえば、IPsec 仮想プライベートネットワーク (VPN) は、IP トンネリングの主な用途の 1 つです。Oracle Solaris でのネットワークセキュリティーの詳細は、『Oracle Solaris 11.2 でのネットワークのセキュリティー保護 』の第 6 章「IP セキュリティーアーキテクチャーについて」を参照してください。IPsec を構成する場合は、『Oracle Solaris 11.2 でのネットワークのセキュリティー保護 』の第 7 章「IPsec の構成」を参照してください。

IP トンネルの管理

このセクションには、次のトピックが含まれています。

▼ IP トンネルを作成および構成する方法

1. **root の役割になります。**

2. **トンネルを作成します。**

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

`-t`	一時的なトンネルを作成します。このコマンドはデフォルトでは永続的なトンネルを作成します。 トンネル上で永続的 IP インタフェースを構成するには、永続的なトンネルを作成し、`-t` オプションは使用しないようにする必要があります。
`-T` *type*	作成するトンネルのタイプを指定します。この引数は、どのトンネルタイプを作成する場合も必要です。
`-a [local|remote]=`*address,...*	ローカルアドレスとリモートトンネルアドレスに対応するリテラル IP アドレスまたはホスト名を指定します。これらのアドレスは有効であり、かつシステム内ですでに作成されている必要があります。トンネルのタイプに応じ

て、アドレスを 1 つだけ指定するか、ローカルアドレスとリモートアドレスの両方を指定します。ローカルアドレスとリモートアドレスの両方を指定する場合は、それらのアドレスをコンマで区切ります。

- `IPv4` トンネルが機能するためには、ローカルとリモートの IPv4 アドレスが必要です。
- `IPv6` トンネルが機能するためには、ローカルとリモートの IPv6 アドレスが必要です。
- `6to4` トンネルが機能するためには、ローカル IPv4 アドレスが必要です。

注記 - 永続的な IP トンネルデータリンクの構成でホスト名をアドレスとして使用した場合、それらのホスト名が構成ストレージに保存されます。次回以降のシステムブート時に、トンネル作成時に使用された IP アドレスとは異なる IP アドレスに名前が解決された場合、トンネルは新しい構成を取得します。

---

tunnel-link IP トンネルリンクを指定します。このリリースでは、ネットワークリンク管理で意味のある名前がサポートされたため、トンネル名が作成対象トンネルのタイプに制限されなくなりました。代わりに、管理者によって選択された名前をトンネルに割り当てることができます。トンネル名は、*mytunnel0* のように、文字列と物理接続点 (PPA) 番号から構成されます。意味のある名前の割り当てに関する規則については、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「有効なリンク名のための規則」を参照してください。

3. **(オプション) ホップ制限またはカプセル化制限の値を設定します。**

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

`hoplimit` IPv6 上でのトンネリング用のトンネルインタフェースのホップ制限を指定します。*hoplimit* は、IPv4 上でのトンネリングの IPv4 生存時間 (TTL) フィールドに相当します。

`encaplimit` 1 つのパケットで許可される入れ子のトンネリングのレベル数を指定します。このオプションは IPv6 トンネルにのみ適用されます。

`hoplimit` と `encaplimit` に設定する値は、許容範囲内である必要があります。`hoplimit` および `encaplimit` プロパティーはトンネルリンクのプロパティーです。したがって、これらのプロパティーは、ほかのリンクプロパティーと同じように、`dladm` サブコマンドを使って管理します。サブコマンドは、`dladm set-linkprop`、`dladm reset-linkprop`、および `dladm show-linkprop` です。

4. **トンネル上で IP インタフェースを作成します。**

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

ここで、*tunnel-interface* ではトンネルリンクと同じ名前を使用します。

5. **ローカルおよびリモートの IP アドレスをトンネルインタフェースに割り当てます。**

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

ここで、*interface* はトンネルインタフェースを指定します。

詳細は、ipadm(1M)のマニュアルページ、および『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』を参照してください。

6. **(オプション) トンネルの IP インタフェース構成のステータスを確認します。**

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

例 5-1 IPv4 トンネル上での IPv6 インタフェースの作成

この例では、永続的な IPv6 over IPv4 トンネルを作成する方法を示します。

```
# 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
```

代替アドレスを追加するには、同じ構文を使用します。たとえば、次のようにしてグローバルアドレスを追加できます。

```
# 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
```

IPv6 アドレスの接頭辞 `2001:db8` は、ドキュメントの例で特別に使用される特殊な IPv6 接頭辞です。

例 5-2 IPv4 トンネル上での IPv4 インタフェースの作成

この例では、永続的な IPv4 over IPv4 トンネルを作成する方法を示します。

```
# 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
```

さらに、このトンネル上を通過するパケットに対してセキュリティー保護された接続を提供するために、IPsec ポリシーを構成することもできます。詳細は、『Oracle Solaris 11.2 でのネットワークのセキュリティー保護 』の第 7 章「IPsec の構成」を参照してください。

例 5-3 IPv6 トンネル上での IPv6 インタフェースの作成

この例では、永続的な IPv6 over IPv6 トンネルを作成する方法を示します。

```
# 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
```

グローバルアドレスや代替のローカルおよびリモートアドレスなどのアドレスを追加するには、次のように `ipadm` コマンドを使用します。

```
# 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
```

▼ 6to4 トンネルを構成する方法

6to4 トンネルを構成する場合、6to4 ルーターは、ネットワークの 6to4 サイト内のノードに対して IPv6 ルーターとして機能する必要があります。このため、6to4 ルーターを構成するときには、そのルーターを、その物理インタフェース上で IPv6 ルーターとしても構成する必要があります。Oracle Solaris ホストをルーターとして構成する方法の詳細については、『ルーターまたはロードバランサとしての Oracle Solaris 11.2 システムの構成 』の「IPv6 ルーターの構成」を参照してください。

1. **6to4 トンネルを作成します。**

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

-a local=*address*
: トンネルのローカルアドレスを指定します。これが有効なアドレスであるためには、それがすでにシステム内に存在している必要があります。

tunnel-link
: IP トンネルリンクを指定します。ネットワークリンク管理での意味のある名前のサポートにより、トンネル名が作成対象トンネルのタイプに制限されなくなりました。代わりに、管理者が選択した名前をトンネルに割り当てることができます。トンネル名は、*mytunnel0* のように、文字列と PPA 番号から構成されます。詳細は、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の「有効なリンク名のための規則」を参照してください。

2. **トンネルの IP インタフェースを作成します。**

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

ここで、*tunnel-interface* ではトンネルリンクと同じ名前を使用します。

3. **(オプション) トンネルで使用するための代替 IPv6 アドレスを追加します。**

4. **/etc/inet/ndpd.conf ファイルを編集します。**

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

5. **6to4 ルーティングを広告するためにファイルに次の 2 行を追加します。**

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

ここで第 1 行は広告を受信するサブネットを、*subnet-interface* はサブネットの接続先リンクを表します。2 行目の IPv6 アドレスの接頭辞は、6to4 トンネルの IPv6 アドレスで使用される 6to4 接頭辞 `2000` になっている必要があります。

`ndpd.conf` ファイルの詳細は、`ndpd.conf`(4)のマニュアルページを参照してください。

6. **IPv6 転送を有効にします。**

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

7. **次のいずれかのオプションを選択します。**

- **ルーターをリブートします。**
- **`sighup` を `/etc/inet/in.ndpd` デーモンに発行して、ルーター広告の送信を開始します。**

各サブネット上の 6to4 接頭辞を受信する IPv6 ノードは、新しい 6to4 派生アドレスで自動構成されます。

8. **ノードに使用される 6to4 派生の新しいアドレスを 6to4 サイトで使用されるネームサービスに追加します。**

   手順については、『Oracle Solaris 11.2 でのネットワークコンポーネントの構成と管理 』の第 4 章「Oracle Solaris クライアントでのネームサービスとディレクトリサービスの管理」を参照してください。

例 5-4 6to4 トンネルの作成

この例では、6to4 トンネルを作成する方法を示します。6to4 トンネル上で構成できるのは IPv6 インタフェースだけです。この例では、サブネットのインタフェース (`/etc/inet/ndpd.conf` 内で参照される) が `net0` になっています。

```
# 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
```

6to4 トンネルでは IPv6 アドレスの接頭辞は `2002` です。

▼ 6to4 リレールーターとの間の 6to4 トンネルを有効にする方法

注意 - セキュリティー上の大きな問題のため、Oracle Solaris では、6to4 リレールーターのサポートはデフォルトでは無効になっています。詳細は、『Oracle Solaris 11.2 でのネットワーク管理のトラブルシューティング 』の「6to4 リレールーターへのトンネルを作成するときのセキュリティー問題」を参照してください。

始める前に 6to4 リレールーターとの間の 6to4 トンネルを有効にする前に、次のタスクを完了します。

- 自サイトで 6to4 ルーターを構成します。118 ページの「IP トンネルを作成および構成する方法」を参照してください。
- 6to4 リレールーターとの間のトンネリングに伴うセキュリティー上の問題を検討します。

1. **次のどちらかの方法を使用して、6to4 リレールーターとの間のトンネルを有効にします。**

   - **エニーキャスト 6to4 リレールーターとの間のトンネルを有効にします。**

     ```
     # 6to4relay -e
     ```

     `-e` オプションは、6to4 ルーターとエニーキャスト 6to4 リレールーターの間にトンネルを設定します。エニーキャスト 6to4 リレールーターは既知の IPv4 アドレス `192.88.99.1` を持っています。サイトに物理的にもっとも近いエニーキャストリレールーターが、6to4 トンネルのエンドポイントになります。このリレールーターは、6to4 サイトとネイティブ IPv6 サイト間のパケット転送を処理します。

     詳細は、RFC 3068「An Anycast Prefix for 6to4 Relay Routers」(http://www.rfc-editor.org/rfc/rfc3068.txt) を参照してください。

   - **特定の 6to4 リレールーターとの間のトンネルを有効にします。**

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

     `-a` オプションは、特定のルーターアドレスが続くことを示します。*relay-router-address* には、トンネルを有効にするために使用する特定の 6to4 リレールーターの IPv4 アドレスを指定してください。

   6to4 リレールーターとの間のトンネルは、6to4 トンネル擬似インタフェースが削除されるまでアクティブな状態を維持します。

2. **6to4 リレールーターとの間のトンネルが不要になったら、トンネルを削除します。**

```
# 6to4relay -d
```

3. **(オプション) リブートを行なっても 6to4 リレールーターとの間のトンネルが永続するように設定します。**

   サイトによっては、6to4 ルーターがリブートするたびに 6to4 リレールーターとの間のトンネルを元に戻さざるをえない場合があるでしょう。このようなシナリオをサポートするには、次を行う必要があります。

   a. **/etc/default/inetinit ファイルを編集します。**

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

      変更が必要な行は、ファイルの最後にあります。

   b. **ACCEPT6TO4RELAY=NO という行の値 NO を YES に変更します。**

   c. **(オプション) 特定の 6to4 リレールーターとの間で、リブートを行なっても永続するトンネルを構築します。**

      RELAY6TO4ADDR パラメータのアドレス 192.88.99.1 を、使用する 6to4 リレールーターの IPv4 アドレスに変更します。

例 5-5 6to4 リレールーターサポートのステータス情報の取得

6to4relay コマンドを使用すると、6to4 リレールーターのサポートが有効になっているかどうかを確認できます。次の例は、6to4 リレールーターのサポートを無効にした場合 (これが Oracle Solaris のデフォルト) の出力です。

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

6to4 リレールーターのサポートを有効にすると、次の出力が表示されます。

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

IP トンネル構成の変更

トンネルの構成を変更するには、次のコマンド構文を使用します。

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

既存のトンネルのタイプは変更できません。したがって、-T *type* オプションはこのコマンドでは許可されません。次のトンネルパラメータだけが変更できます。

-a [local|remote]=*address,...*
: ローカルアドレスとリモートトンネルアドレスに対応するリテラル IP アドレスまたはホスト名を指定します。トンネルのタイプに応じて、アドレスを 1 つだけ指定するか、ローカルアドレスとリモートアドレスの両方を指定します。ローカルアドレスとリモートアドレスの両方を指定する場合は、それらのアドレスをコンマで区切ります。

  - IPv4 トンネルが機能するためには、ローカルとリモートの IPv4 アドレスが必要です。
  - IPv6 トンネルが機能するためには、ローカルとリモートの IPv6 アドレスが必要です。
  - 6to4 トンネルが機能するためには、ローカル IPv4 アドレスが必要です。

  永続的な IP トンネルデータリンクの構成でホスト名をアドレスとして使用した場合、それらのホスト名が構成ストレージに保存されます。次回以降のシステムブート時に、トンネル作成時に使用された IP アドレスとは異なる IP アドレスに名前が解決された場合、トンネルは新しい構成を取得します。

  トンネルのローカルとリモートのアドレスを変更する場合には、変更しているトンネルのタイプとこれらのアドレスが矛盾しないことを確認してください。

- トンネルリンクの名前を変更するには、modify-iptun コマンドではなく、次のように dladm rename-link コマンドを使用します。

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

- hoplimit や encaplimit などのトンネルプロパティーを変更するには、modify-iptun コマンドではなく dladm set-linkprop コマンドを使用します。

例 5-6 トンネルのアドレスとプロパティーの変更

次の例は 2 つの手順から構成されています。まず、IPv4 トンネル vpn0 のローカルとリモートのアドレスが一時的に変更されます。あとでシステムがリブートされるときに、このトンネルはまた元のアドレスを使用するようになります。2 番目のコマンドは、vpn0 の hoplimit を 60 に変更する方法を示しています。

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

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

IP トンネルの構成の表示

IP トンネルの構成を表示するには、次のコマンド構文を使用します。

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

-p　　マシンによる解析が可能な形式で情報を表示します。この引数はオプションです。

-o *fields*　　特定のトンネル情報を提供するフィールドのうち、選択されたフィールドを表示します。

tunnel-link　　構成情報を表示するトンネルを指定します。この引数はオプションです。トンネル名を省略すると、このコマンドはシステム上のすべてのトンネルの情報を表示します。

例 5-7　　すべてのトンネルに関する情報の表示

次の例では、システム上にトンネルが 1 つしか存在しません。

```
# 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　　選択されたフィールドをマシンによる解析が可能な形式で表示します。

次の例では、トンネル情報を含む特定のフィールドのみが表示されます。

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

IP トンネルのプロパティーの表示

トンネルリンクのプロパティーを表示するには、次のコマンド構文を使用します。

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

-c　　マシンによる解析が可能な形式で情報を表示します。この引数はオプションです。

-o *fields*　　リンクのプロパティーに関する特定の情報を提供する、選択されたフィールドを表示します。

tunnel-link プロパティーに関する情報を表示するトンネルを指定します。この引数はオプションです。トンネル名を省略すると、このコマンドはシステム上のすべてのトンネルの情報を表示します。

例 5-9 トンネルのプロパティーの表示

次の例は、トンネルのすべてのリンクプロパティーを表示する方法を示しています。

```
# 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            --        --
```

▼ IP トンネルを削除する方法

1. **トンネル上に構成されている IP インタフェースを、インタフェースのタイプに応じて unplumb します。**

   # ipadm delete-ip *tunnel-link*

   注記 - トンネルを正常に削除するには、そのトンネル上で既存の IP インタフェースが plumb されていてはいけません。

2. **IP トンネルを削除します。**

   # dladm delete-iptun *tunnel-link*

このコマンドのオプションは -t だけです。これにより、トンネルは一時的に削除されます。システムをリブートするときに、トンネルが復元されます。

例 5-10 IPv6 インタフェースが構成された IPv6 トンネルの削除

次の例では、永続的なトンネルが恒久的に削除されます。

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

索引

た

な

は

ら

D

F

I

M

N

P

R

S

T