IPv6 の管理

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 816–6184–10
2002 年 9 月

原典: *IPv6 Administration Guide*

Part No: 816-5250-10

Revision A

目次

はじめに　**7**

1　IPv6 (概要)　11
IPv6 の機能　11
IPv6 のヘッダーと拡張機能　12
ヘッダーフォーマット　12
拡張ヘッダー　13
IPv6 アドレス指定　14
ユニキャストアドレス　16
集約グローバルユニキャストアドレス　16
ローカルアドレス　17
組み込み IPv4 アドレスを伴った IPv6 アドレス　18
エニーキャストアドレス　19
マルチキャストアドレス　19
IPv6 のルーティング　20
IPv6 の近傍検索　21
ルーター通知　22
ルーター通知プレフィックス　22
ルーター通知メッセージ　23
近傍要請と不到達　23
IPv4 との比較　24
IPv6 ステートレスアドレス自動設定　25
ステートレス自動設定の条件　26
ステートフル自動設定モデル　26
ステートレス方式とステートフル方式をいつ使用するか　26
重複アドレスの検出アルゴリズム　27

IPv6 プロトコルの概要　27
IPv6 モビリティ (移動性) サポート　29
IPv6 サービス品質 (QoS) 機能　30
フローラベル　30
トラフィッククラス　31
IPv6 セキュリティの強化　32

2　IPv6 の管理 (手順)　33
IPv6 ノードを有効にする　33
IPv6 ノードを有効にする (作業マップ)　34
▼ ノード上の IPv6 を有効にする方法　34
▼ Solaris IPv6 ルーターの設定方法　35
▼ NIS と NIS+ に対する IPv6 アドレスの追加方法　36
▼ DNS に対する IPv6 アドレスの追加方法　37
IPv6 の監視　38
IPv6 の監視 (作業マップ)　38
▼ インタフェースアドレス割り当ての表示方法　39
▼ ネットワーク状態の表示方法　40
▼ IPv6 関連コマンドの出力表示の制御方法　43
▼ IPv6 ネットワークトラフィックの監視方法　44
▼ すべてのマルチホームホストアドレスの探査方法　45
▼ すべてのルーターのトレース方法　45
IP 内 IP トンネルの設定　46
IP 内 IP トンネルの設定 (作業マップ)　46
▼ IPv4 トンネルによる IPv6 の設定方法　46
▼ IPv6 トンネル経由の IPv6 の設定方法　47
▼ IPv6 トンネル経由の IPv4 の設定方法　48
▼ IPv4 トンネル経由の IPv4 の設定方法　49
▼ トンネルインタフェースで通知するためのルーターの設定方法　50
IPv6 ネームサービス情報の表示　50
IPv6 ネームサービス情報を表示する (作業マップ)　50
▼ IPv6 ネームサービス情報の表示方法　51
▼ DNS IPv6 PTR レコードの正確な更新の確認方法　52
▼ NIS による IPv6 情報の表示方法　52
▼ NIS+ による IPv6 情報の表示方法　53
▼ ネームサービスに依存しない IPv6 情報の表示方法　53

3 IPv6 のファイルおよびコマンド (リファレンス) 55
Solaris IPv6 実装の概要 55
IPv6 ネットワークインタフェース構成ファイル 56
IPv6 インタフェース構成ファイルのエントリ 57
`ifconfig` ユーティリティに対する IPv6 拡張機能 57
複数のネットワークインタフェースがあるノード 59
IPv4 の動作 59
IPv6 の動作 59
IPv6 デーモン 60
`in.ndpd` デーモン 60
`in.ripngd` デーモン 62
`inetd` インターネットサービスデーモン 63
既存のユーティリティに対する IPv6 拡張機能 64
`netstat(1M)` 65
`snoop(1M)` 65
`route(1M)` 65
`ping(1M)` 65
`traceroute(1M)` 66
表示出力の制御 66
IPv6 の Solaris トンネルインタフェース 67
Solaris ネームサービスに対する IPv6 拡張機能 69
`/etc/inet/ipnodes` ファイル 69
IPv6 の NIS 拡張機能 70
IPv6 の NIS+ 拡張機能 70
IPv6 の DNS 拡張機能 71
`nsswitch.conf` ファイルへの変更 71
ネームサービスコマンドの変更 72
NFS と RPC による IPv6 のサポート 72
ATM 経由の IPV6 サポート 73

4 IPv4 から IPv6 への移行 (リファレンス) 75
移行条件 75
標準移行ツール 76
デュアルスタックの実装 76
ネームサービスの設定 77
IPv4 互換アドレスフォーマットの使用 78
トンネル機構 78

アプリケーションとの対話　80
IPv4 と IPv6 の相互運用性　80
サイト移行のシナリオ　81
その他の移行機構　82

用語集　**85**

索引　**91**

はじめに

『*IPv6* の管理』では、Solaris™ オペレーティング環境にインストールされた IPv6 フレームワークの構成および管理について説明します。このマニュアルでは、SunOS™ 5.9 オペレーティングシステムがすでにインストールされているものとします。 さらに、使用するネットワークソフトウェアが設定されているものとします。SunOS 5.9 オペレーティングシステムは Solaris 製品ファミリの一部であり、Solaris 共通デスクトップ環境 (CDE) などが含まれます。また、SunOS 5.9 は、AT&T System V リリース 4 オペレーティングシステムに準拠しています。

注 – Solaris オペレーティング環境は、SPARC™ と IA の 2 種類のハードウェア (プラットフォーム) 上で動作します。また、Solaris オペレーティング環境は、64 ビットと 32 ビットの両方のアドレス空間でも動作します。このマニュアルで説明する情報は、章、節、注、箇条書き項目、図、表、例、コード例などで特に明記しないかぎり、両方のプラットフォーム、およびアドレス空間に適用されます。

対象読者

このマニュアルは、Solaris 9 リリースを実行するシステムの管理者を対象にしています。このマニュアルを活用するには、1、2 年程度の UNIX システムの管理経験が必要です。UNIX システム管理のトレーニングコースに参加することもこのマニュアルの理解に役立ちます。

内容の紹介

第 1 章では、IPv6 として知られる新しいインターネットプロトコルの概要について説明します。

第 2 章では、IPv6 や IPv6 ルーターを有効にする方法、IPv6 アドレスを DNS、NIS、NIS+ 用に設定する方法、ルーター間のトンネルの作成方法、IPv6 に対応したコマンドを使って診断する方法、IPv6 ネームサービス情報の表示方法について説明します。

第 3 章では、IPv6 の Solaris 実装に伴う概念について説明します。

第 4 章では、IPv4 から IPv6 への移行方法と、標準的な解決法の概要について説明します。

用語集では、主要な IP サービス用語の定義を提供します。

Sun のオンラインマニュアル

docs.sun.com[SM] では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、`http://docs.sun.com` です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`

表 **P–1** 表記上の規則　*(続き)*

字体または記号	意味	例
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system% su` `password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `  XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  `machine_name% command y|n [filename]`

- C シェルのスーパーユーザー

  `machine_name# command y|n [filename]`

- Bourne シェルおよび Korn シェル

  `$ command y|n [filename]`

- Bourne シェルおよび Korn シェルのスーパーユーザー

  `# command y|n [filename]`

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: Shift キーを押します)。ただし、キーボードによっては Enter キーが Return キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、Ctrl-D は Control キーを押したまま D キーを押すことを意味します。

一般規則

- このマニュアルでは、「IA」という用語は、Intel 32 ビットのプロセッサアーキテクチャを意味します。これには、Pentium、Pentium Pro、Pentium II、Pentium II Xeon、Celeron、Pentium III、Pentium III Xeon の各プロセッサ、および AMD、Cyrix が提供する互換マイクロプロセッサチップが含まれます。

第 1 章

IPv6 (概要)

インターネットプロトコル バージョン 6 (IPv6 : Internet Protocol version 6) はインターネットプロトコル (IP) の新しいバージョンです。 IPv6 は、現在の IPv4 から飛躍的に進歩しています。IPv4 から IPv6 には無理なく移行することができます。規定されている移行メカニズムを使用することにより、現在の運用に混乱を生じることなく IPv6 ネットワークを展開できます。IPv6 ではアドレス領域が増加しています。また、シンプルになったヘッダーフォーマット、認証とプライバシのサポート、アドレス割り当ての自動構成を採用し、サービス品質を一新してインターネット機能を強化しました。

この章では、以下の内容について説明します。

- 11 ページの「IPv6 の機能」
- 12 ページの「IPv6 のヘッダーと拡張機能」
- 14 ページの「IPv6 アドレス指定」
- 20 ページの「IPv6 のルーティング」
- 21 ページの「IPv6 の近傍検索」
- 25 ページの「IPv6 ステートレスアドレス自動設定」
- 29 ページの「IPv6 モビリティ (移動性) サポート」
- 30 ページの「IPv6 サービス品質 (QoS) 機能」
- 32 ページの「IPv6 セキュリティの強化」

IPv6 の機能

IPv4 から IPv6 への変更内容は、次のように分類できます。

- 拡張されたルーティングとアドレス指定機能 – IPv6 では IP アドレスサイズを 32 ビットから 128 ビットに拡大して、サポートするアドレス指定階層を広げています。IPv6 はアドレス可能なノード数を増やし、アドレスの自動設定も容易にしています。

スコープフィールドの追加により、マルチキャストアドレスに対するマルチキャストルーティングのスケーラビリティを強化しました。

エニーキャストアドレスという新しいタイプのアドレスを定義しました。エニーキャストアドレスは、ノードセットを識別します。エニーキャストアドレスに送信されたパケットは ノードの 1 つに配信されます。IPv6 ソースルートではエニーキャストアドレスを使用して、ノードでトラフィックフローのパスを制御できます。

- ヘッダーフォーマットの簡略化 – IPv4 ヘッダーフィールドが一部削除されたり、オプションになったりしました。この変更によってパケット処理の共通部分の処理コストが削減されます。また、アドレスのサイズは増えましたが、IPv6 ヘッダーの帯域幅コストは可能な限り少なくなりました。IPv6 アドレスの長さは、IPv4 アドレスの 4 倍ですが、IPv6 ヘッダーのサイズは IPv4 の 2 倍に抑えられています。
- オプションサポートの強化 – IP ヘッダーオプションのコード化の方法を変更したため、転送効率が改善されました。また、オプションの長さに関する制限が緩和されています。さらに、将来新しいオプションを導入する際の柔軟性が高くなりました。
- サービス品質の機能 – この機能は、送信側が特別な処理を必要とする特定のトラフィックフローに属するパケットのラベルを指定できます。たとえば、デフォルト以外の品質サービスやリアルタイムサービスなどです。
- 認証機能と機密機能 – IPv6 には認証、データの完全性、機密性をサポートする拡張機能の定義が組み込まれています。

IPv6 のヘッダーと拡張機能

IPv6 プロトコルは、基本 IPv6 ヘッダー、IPv6 拡張ヘッダーを含むヘッダーセットを定義します。

ヘッダーフォーマット

図 1–1 は、IPv6 ヘッダーに使用される要素とその順序を示します。

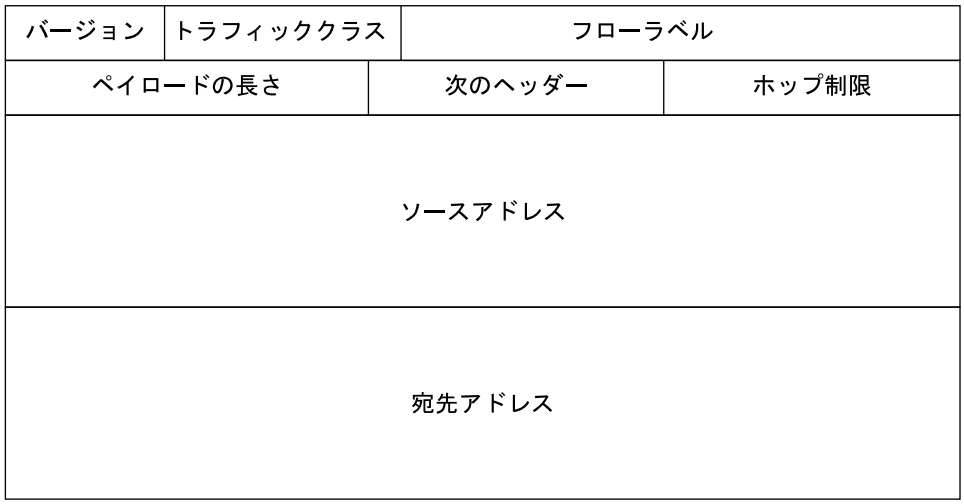

図 **1–1** IPv6 ヘッダーフォーマット

次に各ヘッダーフィールドの機能について説明します。

- バージョン – 4 ビットインターネットプロトコルバージョン番号。IPv6 では 6
- トラフィッククラス – 8ビットトラフィッククラスフィールド。31 ページの「トラフィッククラス」 を参照してください。
- フローラベル – 20 ビットフィールド。30 ページの「IPv6 サービス品質 (QoS) 機能」 を参照してください。
- ペイロードの長さ – オクテット単位で表す 16 ビット符号なし整数。IPv6 ヘッダーに続くパケットの残り
- 次のヘッダー – 8 ビットセレクタ。IPv6 ヘッダーのすぐ後ろに続くヘッダーのタイプを識別する。IPv4 プロトコルフィールドとして同じ値を使用する。13 ページの「拡張ヘッダー」 を参照してください。
- ホップ制限 – 8 ビット符号なし整数。パケットを送信するノードごとに値が 1 ずつ減る。ホップ制限がゼロになるとパケットが廃棄される
- ソースアドレス – 128 ビット。パケットの初期送信側のアドレス。14 ページの「IPv6 アドレス指定」 を参照してください。
- 宛先アドレス – 128 ビット。パケットの予定受信側のアドレス。オプションのルーティングヘッダーがある場合、必ずしも受信側とは限らない

拡張ヘッダー

IPv6 には、IPv4 から強化されたオプション機能があります。IPv6 オプションは、IPv6 ヘッダーとトランスポート層の間の独立した拡張ヘッダーにあります。パケットが最終的な宛先に到着するまで、その配送パスに存在するルーターは、ほとんどの場

合 IPv6 拡張ヘッダを確認または処理しません。そのため、オプションがあるパケットを処理するルーターの性能が大幅に改善されました。IPv4 では、オプションがある場合、ルーターですべてのオプションを調べる必要がありました。

IPv4 オプションとは異なり、IPv6 拡張機能ヘッダーの長さを任意に指定できます。またパケットに組み込むことのできるオプションの合計数が 40 バイト以内に限定されない点があります。この機能とその処理方法によって、IPv4 では非現実的であった機能を IPv6 オプションが使用できるようになりました。その良い例が IPv6 認証オプションとセキュリティカプセル化オプションです。

後続のオプションヘッダー (およびそのあとのトランスポートプロトコル) を処理する際の性能を強化するため、IPv6 オプションは常に 8 オクテットの整数倍の長さです。これにより、後続ヘッダーのバイト境界が維持されています。

次の IPv6 拡張ヘッダーが現在、定義されています。

- ルーティング – 拡張ルーティング (IPv4 ルーズソースルートにあたる)
- 断片化 – 断片化および再結合
- 認証 – 整合性および認証、セキュリティ
- カプセル化 – 機密性
- ホップバイホップオプション – ホップごとの処理が必要な特別なオプション
- 宛先オプション – 宛先ノードが判断するオプション情報

IPv6 アドレス指定

IPv6 アドレスは 128 ビット長であり、個々のインタフェースおよびインタフェースセットの識別子です。すべての IPv6 アドレスはインタフェースに割り当てられ、ホストやルーターには割り当てられません。各インタフェースの所属先は 1 つのノードだけなので、インタフェースのユニキャストアドレスは、そのノードの識別子として使用できます。1 つのインタフェースには、任意のタイプの複数の IPv6 アドレスを割り当てることができます。

IPv6 アドレスには、ユニキャスト、エニーキャスト、マルチキャストの 3 種類のタイプがあります。

- ユニキャストアドレスは、1 つのインタフェースを識別する
- エニーキャストアドレスは、インタフェースのセットを識別する。エニーキャストアドレスに送信されたパケットは そのセットのメンバーの 1 つに配信される
- マルチキャストアドレスは、インタフェースのグループを識別する。マルチキャストアドレスに送信されるパケットは、グループにあるすべてのインタフェースに配信される。

IPv6 では、ブロードキャストアドレスの代わりにマルチキャストアドレスが使われます。

IPv6 は IPv4 アドレスの 4 倍のビット数のアドレス、つまり 128 対 32 をサポートします。したがって、計算上はそのアドレス領域は IPv4 のアドレス領域の 40 億 x 40 億倍の大きさになります。実際にはアドレスの割り当てとルーティングでは階層を作成する必要があり、アドレス領域の利用効率が減少します。結果として、利用できるアドレス数は減少します。ただし当面は、IPv6 で提供するアドレス領域で十分です。

アドレスの先頭ビットでは IPv6 アドレスのタイプを指定します。この先頭ビットがある可変長フィールドをフォーマットプレフィックス (FP) といいます。 次の表は、これらのプレフィックス (接頭辞) の初期割り当てです。

表 **1–1** フォーマットプレフィックスの割り当て

割り当て	プレフィックス (バイナリ)	アドレス領域の端数
予約済み	0000 0000	1/256
割り当てなし	0000 0001	1/256
NSAP 割り当てに予約	0000 001	1/128
IPX 割り当てに予約	0000 010	1/128
割り当てなし	0000 011	1/128
割り当てなし	0000 1	1/32
割り当てなし	0001	1/16
集約グローバルユニキャストアドレス	001	1/8
割り当てなし	010	1/8
割り当てなし	011	1/8
ニュートラル相互接続ベースユニキャストアドレスに予約	100	1/8
割り当てなし	101	1/8
割り当てなし	110	1/8
割り当てなし	1110	1/16
割り当てなし	1111 0	1/32
割り当てなし	1111 10	1/64
割り当てなし	1111 110	1/128
割り当てなし	1111 1110 0	1/512
リンクローカル用アドレス	1111 1110 10	1/1024
サイトローカル用アドレス	1111 1110 11	1/1024
マルチキャストアドレス	1111 1111	1/256

割り当てには、集約グローバルユニキャストアドレス、ローカル用アドレス、マルチキャストアドレスの直接割り当てがサポートされています。NSAP (ネットワークサービスアクセスポイント) アドレス、IPX (相互ネットワークパケット交換プロトコル) アドレス、ニュートラル相互接続アドレスには空間が予約されています。残りのアドレス領域は将来用に割り当てなしになっています。残りのアドレス領域は、既存の領域の拡張部分に利用できます。たとえば、集約グローバルユニキャストアドレスへの追加として使用されます。残りの領域は、新規用としても使用されます。たとえば、個別のロケータと個別の識別子として使用されます。なお、エニーキャストアドレスはユニキャストアドレス領域の範囲外に割り当てられるため、ここには示していません。

初期設定で、アドレス領域の約 15 パーセントが割り当てられます。残りの 85 パーセントは将来用に予約されています。

ユニキャストアドレス

IPv6 ユニキャストアドレスの割り当て形式は、次のとおりです。

- 集約グローバルユニキャストアドレス
- ニュートラル相互接続ユニキャストアドレス
- NSAP アドレス
- IPX 階層アドレス
- サイトローカル用アドレス
- リンクローカル用アドレス
- IPv4 対応ホストアドレス

その他のアドレスタイプは、あとから定義できます。

集約グローバルユニキャストアドレス

集約グローバルユニキャストアドレスは、グローバル通信に使用するアドレスです。CIDR (クラスレス相互ドメインルーティング) における IPv4 アドレスに機能的に似ています。表 1–2 に、そのフォーマットをまとめます。

表 1–2 集約グローバルユニキャストアドレスのフォーマット

3 ビット	13 ビット	8 ビット	24 ビット	16 ビット	64 ビット
FP	TLA ID	RES	NLA ID	SLA ID	Interface ID

FP	フォーマットプレフィックス (001)
TLA ID	最上位集約識別子

RES	将来用に予約
NLA ID	次レベル集約識別子
SLA ID	サイトレベル集約識別子
INTERFACE ID	インタフェース識別子

最初の 48 ビットはパブリックトポロジを表します。次の 16 ビットは各サイトのトポロジを表します。

最初の 3 ビットは集約グローバルユニキャストアドレスとしてアドレスを識別します。次のフィールドである TLA ID はルーティング階層の最上位レベルです。その次の 8 ビットは将来用に予約されています。NLA ID フィールドは TLA ID を割り当てられた組織が、アドレス指定階層の作成と、サイトの識別に使用します。

SLA ID フィールドは、組織で各ローカルアドレス指定階層の作成とサブネットを識別するときに使用します。SLA ID フィールドの使い方は IPv4 のサブネットと似ていますが、組織別に割り当てることのできるサブネット数がはるかに多いところが異なります。16 ビット SLA ID フィールドがサポートするサブネットの数は 65,535 です。Interface ID は、リンク上のインタフェースを識別するために使用します。Interface ID はそのリンク上で一意である必要があります。また、より広い範囲で一意とすることができます。通常、インタフェース識別子はインタフェースのリンクのアドレスと同じです。または、インタフェースのリンク層のアドレスに基づいた値です。

ローカルアドレス

ローカル用アドレスは、ローカルにルーティング可能な範囲のみを対象とするユニキャストアドレスです。使用できるのは、サブネット内または加入者ネットワーク内に限定されます。ローカル用アドレスは、プラグアンドプレイのローカル通信を行うために、サイト内で使用します。また、グローバルアドレスを使用するためのブートストラップ操作を行うために使用されます。

ローカル用のユニキャストアドレスには、リンクローカルとサイトローカルの 2 種類があります。リンクローカル用は単一リンクで使用します。サイトローカル用は単一サイトで使用します。次の表は、リンクローカル用アドレスフォーマットを示したものです。

表 1–3 リンクローカル用アドレスフォーマット

10 ビット	54 ビット	64 ビット
1111111010	0	Interface ID

リンクローカル用アドレスは自動アドレス設定などの目的で 1 つのリンク上のアドレス指定に使用します。

表 1–4 は、サイトローカル用アドレスフォーマットです。

表 **1–4** サイトローカル用アドレス

10 ビット	38 ビット	16 ビット	64 ビット
1111111011	0	Subnet ID	Interface ID

どちらのタイプのローカルアドレスでも、インタフェース ID はそれを使用するドメインで一意な識別子である必要があります。通常は、識別子としてノードの IEEE-802 48 ビットアドレスを使用します。Subnet ID は、サイト内の特定のサブネットを識別します。Subnet ID と Interface ID を組み合わせてローカルアドレスを作成します。これで大規模なプライベートインターネットを構築することができ、その他のアドレス割り当てを行う必要はありません。

現在グローバルインターネットに接続していない組織はローカルアドレスを使用できます。ローカルアドレスを使うだけでグローバルインターネットアドレス領域からのアドレスプレフィックスを要求する必要はありません。この組織が将来インターネットに接続する場合、Subnet ID と Interface ID をグローバルプレフィックスと組み合わせてグローバルアドレスを作成することができます。たとえば、Registry ID、Provider ID、Subscriber ID の組み合わせでグローバルアドレスを作成できます。この拡張機能は IPv4 に対する大幅な改善点です。IPv4 では、プライベート (非グローバル) な IPv4 アドレスを使うサイトは、インターネットに接続する場合に手動で番号を指定し直す必要があります。IPv6 の場合、番号は自動的に指定し直されます。

組み込み IPv4 アドレスを伴った IPv6 アドレス

IPv6 移行機能では、ホストとルーターが IPv4 ルーティングインフラストラクチャのもとで IPv6 パケットを動的にトンネル処理できる方式を採用しています。この方式を利用した IPv6 ノードには、下位 32 ビットに IPv4 アドレスを保存した特別な IPv6 ユニキャストアドレスが割り当てられます。このタイプのアドレスを *IPv4* 互換 *IPv6* アドレスといいます。 次の表にそのフォーマットを示します。

表 **1–5** IPv4 互換 IPv6 アドレスフォーマット

80 ビット	16 ビット	32 ビット
0000.....0000	0000	IPv4 アドレス

組み込み IPv4 アドレスを保存する第 2 のタイプの IPv6 アドレスも定義されています。このアドレスは IPv6 アドレス領域内の IPv4 アドレスを表すときに使用します。このアドレスは主に、アプリケーション、API、オペレーティングシステムの実装内で使用します。このタイプのアドレスを *IPv4* マップ *IPv6* アドレスといいます。 次の表にそのフォーマットを示します。

表 **1–6** IPv4 マップ IPv6 アドレスフォーマット

80 ビット	16 ビット	32 ビット
0000.....0000	FFFF	IPv4 アドレス

エニーキャストアドレス

IPv6 エニーキャストアドレスは複数のインタフェースに割り当てるアドレスです。通常は、エニーキャストアドレスは異なるノードに所属しています。エニーキャストアドレスに送られたパケットは、そのアドレスを持つ最も近いインタフェースにルーティングされます。

エニーキャストアドレスはルートシーケンスの一部に使用できます。 したがって、ノードはトラフィックを搬送するインターネットサービスプロバイダを選択できます。この機能をソース選択ポリシーと呼ぶこともあります。この機能を実装するには、インターネットサービスプロバイダに所属するルーターセットを識別するようにエニーキャストアドレスを構成します。たとえば、インターネットサービスプロバイダごとに 1 つのエニーキャストを構成します。エニーキャストを、IPv6 ルーティングヘッダーで中間アドレスとして使用できます。これにより、特定のプロバイダによってパケットが配信されます。または、一連のプロバイダによって配信されます。また、エニーキャストアドレスは、特定のサブネットに接続されたルーターセットや、特定のルーティングドメインへのエントリを提供するルーターセットの識別にも使用できます。

定義済みのユニキャストアドレスフォーマットを利用すれば、ユニキャストアドレス領域からエニーキャストを指定できます。そのため、エニーキャストアドレスは、構文的にはユニキャストアドレスと区別がつきません。複数のインタフェースにユニキャストアドレスを割り当てる場合は、ユニキャストアドレスをエニーキャストアドレスに変換します。ただし、そのアドレスがエニーキャストアドレスであることがわかるように、アドレスを割り当てるノードを明示的に構成する必要があります。

マルチキャストアドレス

IPv6 マルチキャストアドレスは、インタフェースグループの識別子です。1 つのインタフェースが所属できるマルチキャストグループは複数設定できます。表 1–7 は、マルチキャストアドレスフォーマットを示します。

表 **1–7** マルチキャストアドレスフォーマット

8 ビット	4 ビット	4 ビット	112 ビット
11111111	FLGS	SCOP	グループ ID

アドレスの先頭の 11111111 は、アドレスがマルチキャストアドレスであることを表します。FLGS は、4 つのフラグ (0,0,0,T) のセットです。

上位 3 つのフラグは、予約されていて、これらのフラグは 0 に初期設定される必要があります。

- **T=0** – 固定的に割り当てられた (既知の) マルチキャストアドレスを識別する。このマルチキャストアドレスは、グローバルインターネット番号指定機関が割り当てます。
- **T=1** – 非固定的に割り当てられた (一時的な) マルチキャストアドレスを識別する

SCOP は、4 ビットのマルチキャストスコープの値であり、マルチキャストグループの有効範囲を表します。表 1–8 は、SCOP の値です。

表 **1–8** SCOP の値

0	予約済み	8	組織ローカルスコープ
1	ノードローカルスコープ	9	(割り当てなし)
2	リンクローカルスコープ	A	(割り当てなし)
3	(割り当てなし)	B	(割り当てなし)
4	(割り当てなし)	C	(割り当てなし)
5	サイトローカルスコープ	D	(割り当てなし)
6	(割り当てなし)	E	グローバルスコープ
7	(割り当てなし)	F	予約済み

グループ ID は、指定スコープ内で、固定または一時的のどちらかのマルチキャストグループを識別します。

IPv6 のルーティング

IPv6 のルーティングは、CIDR における IPv4 のルーティングとほぼ同じです。唯一の違いは、IPv4 では 32 ビットアドレスを使用しますが、IPv6 では 128 ビットアドレスを使用することです。非常に簡単な拡張で、IPv4 のルーティングアルゴリズム OSPF、RIP、IDRP、IS-IS などをすべて IPv6 のルーティングに使用できます。

IPv6 には、新たに強力なルーティング機能をサポートした簡単なルーティング拡張機能も組み込まれました。次のリストに、新しいルーティング機能を示します。

- プロバイダ選択 (ポリシー、性能、コストなどを基準に)
- ホストの移動性 (現在の場所までのルート)
- アドレスの自動的な再指定 (新しいアドレスへのルート)

新しいルーティング機能を利用するには、IPv6 ルーティングオプションを使用する IPv6 アドレスのシーケンスを作成します。IPv6 の送信元は、ルーティングオプションでを使用して、パケットが宛先に至るまでに経由する複数の中間ノード (またはトポロジカルグループ) をリストします。この中間ノードは、パケットの宛先の途中に通過します。この機能は、IPv4 での緩やかな経路制御と記録オプションによく似ています。

アドレスシーケンスを一般的に使用する場合、通常は、ホストが受信したパケットのルートを逆戻りする必要があります。このパケットは、IPv6 認証ヘッダーを使用して正常に認証される必要があります。パケットを発信者に戻すには、アドレスシーケンスがパケット内に格納されている必要があります。IPv6 ホストの実装では、この方式により始点経路の処理と逆引きをサポートしています。始点経路の処理と逆引きは、プロバイダが新機能を実装するホストを使用するためのポイントです。新機能には、プロバイダの選択や拡張アドレスが含まれます。

IPv6 の近傍検索

IPv6 では、同じリンクに接続されたノード間の対話に関連した問題をまとめて解決しました。そのため、次のような問題を個々に解決する仕組みを定義しています。

- ルーター検索 – 接続されたリンクにあるルーターをホストが探索する
- プレフィックス探索 – どの宛先がリンクに接続されているかを定義するアドレスプレフィックスのセットをホストが探索する (オンリンクということもある)。リンクにある宛先と、ルーターからだけアクセスできる宛先を、ノードではプレフィックスで区別します。
- パラメータ探索 – ノードは、リンク MTU (最大伝送単位) などのリンクパラメータを調べる。また、出力パケットに設定するホップ限界数などのインターネットパラメータを調べる
- アドレス自動設定 – インタフェースのアドレスをノードが自動的に設定する
- アドレス解決 – 宛先の IP アドレスだけを使用してノードが近傍のリンク層アドレスを判定する (オンリンク宛先)
- 次のホップの決定 – アルゴリズムは宛先へのトラフィックが送られるべき近傍の IP アドレスに IP 宛先アドレスのマッピングを決定する。次のホップはルーターまたは宛先になる
- 不到達検出 – 近傍に到達不可能であることをノードが判定する。ルーターに使用される近傍の場合、代替デフォルトルーターを試行できる。ルーターとホストの場合、アドレス解決を再試行できる
- 重複アドレス検出 – あるノードがアドレスを要求したところ、別のノードがそのアドレスを使用していないかを判別する
- リダイレクト – 特定の宛先へのアクセス手段として、最適な最初のホップノードをルーターからホストに知らせる

近傍検索では、ルーター要請メッセージとルーター通知メッセージのペア、近傍要請メッセージと近傍通知メッセージのペア、リダイレクトメッセージという 5 種類の ICMP (インターネット制御メッセージプロトコル) パケットタイプを定義します。これらのメッセージの目的は、次のとおりです。

- ルーター要請 – インタフェースが使用可能になると、ホストはルーター要請を送信できる。この要請は、次に予定されている時刻ではなく、ただちにルーター通知メッセージを送信するようにルーターに要求する
- ルーター通知 – ルーターはさまざまなリンクパラメータやインターネットパラメータとともにその存在を通知する。ルーターは定期的に、あるいはルーター要請メッセージに応じて通知する。ルーター通知には、オンリンク判別またはアドレス設定、あるいはホップ限界数の選択肢などに使用するプレフィックスが含まれる
- 近傍要請 – 近傍のリンク層アドレスを判定するため、および、近傍がキャッシュリンク層アドレスで到達可能かどうかを確認するためにノードによって送信される。近傍要請は重複アドレス検出にも使用する
- 近傍通知 – 近傍要請メッセージに対する応答として、ノードでは未要請の近傍通知も送信してリンク層アドレスの変更を伝える
- リダイレクト – 宛先までの最適な最初のホップ、または宛先がオンリンクであることをルーターからホストに知らせる

ルーター通知

マルチキャスト対応リンクとポイントツーポイントリンクでは、ルーターは定期的にルーター通知パケットをマルチキャストして利用できることを知らせます。ホストはすべてのルーターからルーター通知を受け取り、デフォルトルーターのリストを作成します。利用できるルーターをホストが短時間 (2 、3 分以内) に知ることができるように、ルーターは頻繁にルーター通知を生成します。ただし、通知がないからといってルーターエラーであると判断できるほどの頻度ではありません。エラー検出には、近傍到達不能性を判別する別の検出アルゴリズムを利用します。

ルーター通知プレフィックス

ルーター通知には、オンリンク判別に使用するプレフィックスリストが含まれます。このプレフィックスリストは、自動アドレス設定にも使用されます。プレフィックスに付属するフラグは特定のプレフィックスの使用目的を表します。ホストは、通知されたオンリンクプレフィックスからリストを作成し管理します。リストは、パケットの宛先がいつオンリンクになっているか、あるいはルーターを離れているかを知るために使用します。通知されたオンリンクプレフィックスになくても宛先がオンリンクの場合があります。この場合、ルーターはリダイレクトを送ることができます。リダイレクトは送信側に、宛先が近傍であることを知らせます。

ルーター通知 (およびプレフィックス別のフラグ) では、ルーターからホストにアドレスの自動設定の方法を伝えることができます。たとえば、ステートフル (DHCPv6) か自動 (ステートレス) のどちらのアドレス設定を使用するかなどがあります。

ルーター通知メッセージ

ルーター通知メッセージには、ホストが出力パケットで使用する必要があるホップ限界数などのインターネットパラメータも組み込むことができます。また、オプションでリンク MTU などのリンクパラメータも組み込むことができます。この機能により、重要なパラメータの集中管理が可能になります。パラメータは、ルーターに設定され、関連付けられたすべてのホストに自動的に伝達されます。

ノードでは、宛先ノードに対してそのリンク層アドレスを戻すよう要求する近傍要請をマルチキャストしてアドレス解決を行います。近傍要請メッセージは、宛先アドレスの要請先のノードマルチキャストアドレスにマルチキャストされます。宛先は、そのリンク層アドレスをユニキャスト近傍通知メッセージで戻します。発信元と宛先の両方に対して 1 つの要求応答パケットペアで互いのリンク層アドレスを処理できます。発信元は、近傍要請に発信元のリンク層アドレスを組み込みます。

近傍要請と不到達

近傍要請メッセージでは、複数のノードに同じユニキャストアドレスが割り当てられているかを確認することもできます。

近傍不到達検出では、近傍エラーや近傍への送信パスのエラーを検出します。近傍不到達検出では、近傍に送信されるパケットがその近傍に実際にアクセスして、その IP 層で正しく処理されたかどうかを確認する肯定確認が必要です。近傍不到達検出では、2 つのソースの確認を使用します。可能な場合、上位層のプロトコルでは、接続が送信を処理中であるという肯定確認を戻します。先に送信されたデータは正しく配信されたということが通知されます。たとえば、最も新しい TCP 肯定を受信したことが通知されます。肯定応答が得られない場合、ノードはユニキャスト近傍要請メッセージを送信します。 このメッセージは、次のホップからの到達可能確認として近傍通知を要請します。不要なネットワークトラフィックを避けるため、ノードからアクティブにパケットが送信されている近傍にだけ探査メッセージが送信されます。

上記の一般的な問題を解決する以外に、近傍検索では次のような状況にも対応します。

- リンク層アドレスの変更 – リンク層アドレスの変更を認識したノードは、非要請の近傍通知パケットをマルチキャストできる。ノードはすべてのノードにマルチキャストして、無効になったキャッシュリンク層アドレスを更新できる。非要請通知の送信は、性能強化が目的。近傍不到達検出アルゴリズムにより、すべてのノードが確実に新しいアドレスを探索できるが、遅延が多少伸びる可能性がある
- 入力負荷均衡 – インタフェースを複製したノードでは、同じリンク上の複数のネットワークインタフェース間の入力パケットの受信の負荷均衡ができる。このようなノード間では、同じインタフェースに複数のリンク層アドレスが割り当てられる。たとえば、1 つのネットワークドライバで、複数のネットワークインタフェースカードを、複数のリンク層アドレスを持つ 1 つの論理インタフェースとして表現できる

負荷均衡は、ルーターがソースリンク層アドレスをルーター通知パケットから省略することを可能にすることで処理する。この場合、近傍では近傍要請メッセージを使用してルーターのリンク層アドレスを確認する。近傍通知メッセージの戻りには、要請元によって異なるリンク層アドレスが組み込まれる

- エニーキャストアドレス – エニーキャストアドレスは、等価サービスを提供するノードセットの1 つを識別する。同じリンクの複数のノードは同じ任意を 認識するように設定できる。近傍検索では、ノードが同じ宛先に対する複数の近傍通知を受信するようにノードを設定してエニーキャストを処理する。エニーキャストアドレスの通知にはすべて、取り消しできない通知としてのタグが設定される。取り消しできない通知により、複数存在する可能性がある通知の中でどれを使用するかを判定する特定の規則が呼び出される
- プロキシ通知 – 宛先アドレスのかわりにパケットを受信するルーターは、取り消し無効の近傍通知を発行できる。ルーターは、近傍要請に応答できない宛先アドレスのかわりにパケットを受信する。現在はプロキシの使用方法は指定されていないが、オフリンクになった移動ノードをプロキシ通知で処理できる可能性がある。ただし、プロキシは、このプロトコルを実装していないノードを処理する一般的な機構として使用されることはない

IPv4 との比較

IPv6 近傍検索プロトコルは、IPv4 プロトコル ARP (アドレス解決プロトコル)、ICMP ルーター検索、ICMP リダイレクトを組み合わせたようなものです。IPv4 には近傍不到達検出に全般的に対応できるプロトコルや機構はありませんでした。ただし、ホスト条件ではデッドゲートウェイ検出に対応できるアルゴリズムがいくつか指定されています。デッドゲートウェイ検出は、近傍不到達検出の一部です。

近傍検索プロトコルでは、IPv4 プロトコルセットに対するさまざまな強化措置が施されています。

- ルーター検索はベースプロトコルセットの一部であり、ホストがルーティングプロトコルを *snoop* する必要はない
- ルーター通知ではリンク層アドレスが伝達される。ルーターのリンク層アドレスの解決に、これ以外のパケット交換は不要
- ルーター通知ではリンクのプレフィックスが伝達される。ネットマスクを設定する独立した機構は不要
- ルーター通知では、アドレス自動設定が使用可能になる
- ルーターは、ホストがリンクで使用するMTU を通知できる。したがって、MTU が定義されていないすべてのノードはリンク上の同じ MTU 値を使用する
- アドレス解決マルチキャストは、40 億 (2^32) マルチキャストアドレスに展開され、宛先以外のノードに対するアドレス解決関係の割り込みを大幅に削減した。さらに、IPv6 以外のマシンの割り込みをなくした
- リダイレクトには、新しい最初のホップのリンク層アドレスを保存する。独立したアドレス解決がなくてもリダイレクトを受信できる

- 同じリンクに複数のプレフィックスを関連付けられる。デフォルトで、ホストはルーター通知からすべてのオンリンクプレフィックスを受け取る。ただし、ルーター通知にあるプレフィックスをすべて、あるいは一部省略するようにルーターを設定できる。その場合、ホストは宛先がオフリンクであるとみなす。その結果、ホストはルーターにトラフィックを送信する。ルーターは適宜リダイレクトを発行する
- IPv4 と異なり、IPv6 リダイレクトメッセージの受信者は新しい次のホップがオンリンクであるとみなす。IPv4 では、ホストはリダイレクトメッセージを無視し、リンクのネットワークマスクに基づいて、リンクにない次ホップを指定する。IPv6 リダイレクト機構は XRedirect 機能に似ている。リダイレクト機構は、非ブロードキャストおよび共有メディアリンクで有効。これらのリンクでは、ノードはオンリンク宛先のすべてのプレフィックスを確認できない
- 近傍不能性検出により、障害ルーターがある場合のパケット伝送能力が改善される。また、この機能により、部分的に障害があるリンクやパーティション化されたリンクを経由するパケット伝送、あるいはリンク層アドレスが変更されたノードを経由するパケット伝送が改善される。たとえば、移動ノードは、頻繁に更新される ARP キャッシュのおかげでオフリンクになっても接続が切れない
- ARPとは異なり、近傍検索では、近傍不到達検出により、ハーフリンクエラーを検出する。近傍検索は、双方向接続がない近傍にトラフィックが送信されるのを防ぐ
- IPv4 ルーター検索と異なり、ルーター通知メッセージにはユーザー定義フィールドはない。安定性の異なるルーターの操作にユーザー定義フィールドは不要。近傍不能性検出で、デッドルーターを検出し、アクティブルーターに切り替えることができる
- リンクローカルアドレスでルーターを一意に識別しておけば、ホストでルーター関連付けを維持できる。ルーターを識別する機能は、ルーター通知で必要とされる。また、この機能はリダイレクトメッセージも必要とされる。サイトが新しいグローバルプレフィックスを使用しても、ホストはルーター関連付けを維持する必要がある。
- 近傍検索メッセージのホップ制限は受信時に 255 なので、プロトコルがオフリンクノードによるスプーフィングの被害を受けることがない。これに対し、IPv4 オフリンクノードは、ICMP (インターネット制御メッセージプロトコル) リダイレクトメッセージと、ルーター通知メッセージを送ることができる。
- ICMP 層にアドレス解決を配置すると、プロトコルが ARP よりも媒体に依存しなくなる。その結果、標準 IP 認証とセキュリティ機構が使用できるようになる

IPv6 ステートレスアドレス自動設定

ホストでは、IPv6 のインタフェースの自動設定を数ステップかけて実行します。自動設定プロセスでは、リンクローカルアドレスの作成、リンク上の一意性の検査、どのような情報を自動設定するか (アドレス、その他の情報、または両方)、 アドレスをス

テートフル機構またはステートフル機構、あるいはその両方で取得するかの決定が行われます。ここでは、リンクローカルアドレスの生成手順、ステートレスアドレス自動設定によるサイトローカルアドレスとグローバルアドレスの生成手順、そして重複アドレス検出手順について説明します。

ステートレス自動設定の条件

IPv6 では、ステートフルとステートレスのアドレス自動設定機構を定義しています。ステートレス自動設定では、手動によるホストの設定は不要です。ルーターは最小限の設定 (あれば) ですみ、サーバーの追加も不要です。ステートレス機構により、ホストは独自のアドレスを生成できます。ステートレス機構は、アドレスを生成するために、ローカルな情報とルーターが通知する情報を使用します。ルーターはリンクに関連付けられたサブネットを識別するプレフィックスを通知します。ホストはサブネット上で一意にインタフェースを識別するインタフェース識別子を生成します。アドレスはこれらのプレフィックスとインタフェース識別子を組み合わせて作ります。ルーターがない場合、ホストはリンクローカルアドレスだけを生成します。ただし、同じリンクに接続されたノード間の通信では、リンクローカルアドレスで十分です。

ステートフル自動設定モデル

ステートフル自動設定モデルでは、ホストはインタフェースアドレスや設定情報とパラメータをサーバーから取り込みます。サーバーでは、どのホストにどのアドレスが割り当てられたかを保存したデータベースを管理します。ホストは、ステートフル自動設定プロトコルを利用してアドレスやその他の設定情報をサーバーから取り込むことができます。ステートレス自動設定とステートフル自動設定は互いに補完し合います。たとえば、ホストでは、ステートレス自動設定でアドレスを設定し、ステートレス自動設定でその他の情報を取り込みます。

ステートレス方式とステートフル方式をいつ使用するか

ホストが使用するアドレスを厳密に知る必要はない場合に、ステートレス方式を使用します。ただし、アドレスは一意である必要があります。また、アドレスは正しくルートできる必要もあります。正確なアドレス割り当てに対してサイトでさらに厳しく管理する必要がある場合に、ステートフル方式を使用します。ステートフルとステートレスのどちらのアドレス自動設定も同時に使用できます。サイト管理者は、ルーター通知メッセージのフィールドの設定を通じて、どの方式の自動設定を使用するかを指定します。

IPv6 アドレスは、一定の時間 (場合によっては無限に) インタフェースにリースされます。各アドレスには、アドレスがどれだけの時間、インタフェースに割り当てられるかを示す寿命があります。寿命が尽きると、結合とアドレスが無効になり、そのアドレスを別のインタフェースに割り当てることができます。アドレスの割り当ての終了

を正常に行うため、アドレスはインタフェースに割り当てられた状態で 2 つの別々のフェーズを経ます。最初、アドレスには優先権が与えられ、任意に通信ができます。次に、アドレスの現在のインタフェース割り当てが無効になるという前提から、優先順位が下がります。優先順位が低い状態で、アドレスを使用するのは避けるべきですが、使用できないわけではありません。新しい通信 (たとえば、新しい TCP 接続の開始など) ではできるだけ優先順位の高いアドレスを使用します。優先順位の低いアドレスを使用できるのは、そのアドレスを使用中のアプリケーションだけにする必要があります。サービスを打ち切らないと別のアドレスに切り替えるのが困難なアプリケーションは、優先順位の低いアドレスを使用できます。

重複アドレスの検出アルゴリズム

特定のリンク上ですべての設定済みアドレスが一意であることを保証するため、ノードは重複アドレスの検出アルゴリズムを実行します。この実行は、インタフェースにアドレスを割り当てる前に行われる必要があります。重複アドレスの検出アルゴリズムはすべてのアドレスを対象として実行されます。

このマニュアルで指定する自動設定プロセスは、ホストにだけ適用し、ルーターには適用しません。ホストの自動設定では、ルーターが通知した情報を使用するため、ルーターは別の手段で設定する必要があります。ただし、このマニュアルで説明した機構を使用して、ルーターによってリンクローカルアドレスが生成される場合があります。また、インタフェースに割り当てられる前に、すべてのアドレスにおいてルーターによる重複アドレスの検出処理が正常終了していることが望まれます。

IPv6 プロトコルの概要

ここでは、自動設定中にインタフェースが実行する通常の手順について概要を説明します。自動設定が行われるのはマルチキャスト対応リンクだけです。たとえばシステム起動時など、マルチキャスト対応インタフェースが使用可能な状態で開始します。ノード (ホストとルーターの両方) では、そのインタフェースのリンクローカルアドレスを生成して自動設定プロセスを開始します。リンクローカルアドレスは、インタフェースの識別子を既知のリンクローカルプレフィックスに追加して作成します。

ノードは、この仮リンクローカルアドレスがリンク上の別のノードで使用されていないことを確認する必要があります。この確認が終わったら、リンクローカルアドレスをインタフェースに割り当てることができます。特に、ノードは宛先が仮アドレスになっている近傍要請メッセージを送信します。別のノードがそのアドレスを使用中の場合、そのノードはそのことを伝える内容を含む近傍要請を返信します。別のノードがそのアドレスを使用しようと試みている場合、そのノードもその宛先に近傍要請を送信します。近傍要請送信や再送の数と、連続した要請間の遅延 はリンクによって異なります。これらのパラメータは、システム管理で設定できます。

ノードにおいて、仮リンクローカルアドレスが一意でないことがわかると自動設定が打ち切られるため、手動でインタフェースを設定する必要があります。この状態からの回復を簡単にするには、管理者が代替インタフェース識別子を提供してデフォルト

識別子を無効にします。これにより、新しい (一意であると考えられる) インタフェース識別子を利用して自動設定機構を実行できます。そうでなければ、リンクローカルアドレスとその他のアドレスは手動で設定します。

この仮リンクローカルアドレスが一意であると判断されると、ノードはインタフェースにそのアドレスを割り当てます。このとき、ノードは近傍ノードと IP レベルで接続されます。自動設定手順の残りは、ホストだけで実行されます。

ルーター通知の受信

自動設定の次の手順では、ルーター通知を受信するか、ルーターが存在しないことを確認します。ルーターがあれば、ホストが実行すべき自動設定の種類を指定したルーター通知が送信されます。ルーターがない場合、ステートフル自動設定が呼び出されます。

ルーターはルーター通知を定期的に送信します。ただし、連続した送信と送信の間の遅延は、自動設定を実行するホスト側の待機時間より通常は長くなります。通知を迅速に受信するため、すべてのルーターマルチキャストグループに 1 つまたは複数のルーター要請を送信します。ルーター通知には 2 つのフラグがあり、どのようなステートフル自動設定 (あれば) を実行すべきかを表します。管理アドレス設定フラグは、アドレスの取得時にホストがステートフル自動設定を使用するかどうかを表します。もう 1 つのステートフル設定フラグは、その他の情報 (アドレスを除く) の取得時にホストがステートフル自動設定を使用するかどうかを表します。

プレフィックス情報

ルーター通知にプレフィックス情報オプションがある場合、これらのオプションにはステートレスアドレス自動設定におけるサイトローカルアドレスとグローバルアドレスの生成に必要な情報を保存します。ルーター通知のステートレスアドレス自動設定フィールドとステートフルアドレス自動設定フィールドは個別に処理されます。ホストでは、ステートフルアドレス自動設定とステートレスアドレス自動設定を同時に使用できます。プレフィックス情報オプションフィールドの1 つである自動アドレス設定フラグは、オプションがステートレス自動設定にも適用されるかどうかを表します。適用される場合、補助オプションフィールドにサブネットプレフィッスと寿命値が保存されます。これらの値は、プレフィックスから作成されたアドレスがどれだけの時間優先権を持ち有効であるかを表します。

ルーターではルーター通知が定期的に生成されるので、ホストでは常に新しい通知を受信します。ホストは各通知に組み込まれた情報を上記の手順で処理し、情報を追加します。また、ホストは前の通知で受け取った情報を更新します。

アドレスの一意性

安全性確保のため、すべてのアドレスについて、インタフェースに対する割り当て前に一意かどうかが確認されます。ただし、ステートレス自動設定で作成したアドレスの場合は状況が異なります。アドレスの一意性は、インタフェース識別子から生成されるアドレスの一部で主に決まります。したがって、ノードにおいてリンクローカル

アドレスの一意性が確認されると、他のアドレスの個別の確認は不要になります。これらのアドレスが、同じインタフェース識別子から生成されているためです。ただし、手動で得られるアドレスはすべて、個別に一意であることを確認する必要があります。ステートフルアドレスの自動設定で得られるアドレスについても同じです。一部のサイトでは、重複アドレスの検出を実行するためのオーバーヘッドが大きく、それを実行することで得られる利益が帳消しになる場合があります。そのようなサイトでは、インタフェース別設定フラグの設定で重複アドレスの検出の使用を無効にできます。

自動設定処理を短時間で終了するために、ルーター通知の待機とリンクローカルアドレスの生成 (およびその一意性の確認) をホストで並列して実行できます。ルーターでは、ルーター要請に対する応答が数秒遅れる可能性があります。そのため、上記 2 つの手順を 1 つずつ実行すると、自動設定を完了するために必要な合計時間が大幅に長くなる可能性があります。

IPv6 モビリティ (移動性) サポート

ルーティングは、パケットの宛先 IP アドレスのサブネットプレフィックスに基づいて行われます。そのため、モバイルノードを宛先とするパケットは、ホームリンクに関連付けられていないノードには到達できません。ホームリンクは、ノードの IPv6 サブネットプレフィックスが存在するリンクです。通信を継続するために、ノードが新しいリンクに移動するたびに、モバイルノードがその IP アドレスを変更できます。ただし、モバイルノードの位置を変更すると、移動ノードではトランスポート層とその上位層の接続が失われます。以上のことから、将来、インターネットに接続するモバイルコンピュータが増加することを考えると、IPv6 モビリティサポートが大きな意味を持つことになります。

上記の問題に IPv6 モビリティサポートが対応します。IPv6 モビリティでは、モバイルノードがリンク間を移動してもその IP アドレスは変更されません。モバイルノードに対する IP アドレスの割り当ては、そのノードのホームリンク上のホームサブネットプレフィックスの範囲内で行われます。これをノードのホームアドレスといいます。

これにより、モバイルノードのホームアドレスにルートされたパケットは、宛先アクセスできます。モバイルノードが、現在インターネットのどこに接続していても問題はありません。モバイルノードが新しいリンクに移動しても他のノード (固定またはモバイル) との通信は途切れません。

ホームを離れたモバイルノードと送受信するパケットを透過的にルーティングする問題は IPv6 移動サポートで解決できます。しかし、モバイルコンピュータや無線ネットワークの使用に伴うすべての問題が解決されるわけではありません。特に次の問題には対処できません。

- 通常の無線ネットワークのようにアクセスできるときとできないときがあるリンクの処理。ただし、移動検出手順でいくつかの問題は処理できる
- モバイルノードが接続しているリンクのアクセス制御

IPv6 サービス品質 (QoS) 機能

ホストは 、IPv6 ヘッダーのフローラベルフィールドとトラフィッククラスフィールドを使用できます。ホストは、これらのフィールドを使用して、IPv6 ルーターによる特別処理を要求するパケットを識別します。特別処理の例としては、デフォルト以外のサービス品質やリアルタイムサービスがあります。この機能により、ある程度一貫したスループット、遅延、ジッターが必要なアプリケーションをサポートできます。この種のアプリケーションには、マルチメディアアプリケーションまたはリアルタイムアプリケーションがあります。

フローラベル

発信元では、IPv6 ヘッダーの 20 ビットのフローラベルフィールドを使用できます。送信元は、IPv6 ルーターによる特別処理を要求するパケットに、このフィールドを使用してラベルを付けます。特別処理の例としては、デフォルト以外のサービス品質やリアルタイムサービスがあります。この IPv6 の機能はまだ実験段階であり、インターネットのフローサポートの条件が確定すると変更される可能性があります。一部のホストまたはルーターではフローラベルフィールドの機能をサポートしていません。このようなホストまたはルーターでは、パケットの生成時にフローラベルフィールドをゼロに設定する必要があります。パケットが転送される場合は、フローラベルフィールドは変更されないまま転送されます。パケットを受信したホストやルーターはフローラベルフィールドを無視します。

フローとは

フローは特定の送信元から特定のユニキャストまたはマルチキャスト宛先に送信されるパケットのシーケンスです。ソースは、ルーターによる特別処理を必要とします。特別処理の特性は、制御プロトコルによってルーターに伝達される場合があります。制御プロトコルとして、リソース予約プロトコルを使用できます。 また、ホップバイホップオプションなど、フローのパケット内の情報によって伝達される場合もあります。

ソースから宛先までのアクティブフローが複数ある場合があります。また、どのフローとも関連していないトラフィックを含む場合もあります。フローの一意の識別はソースアドレスとゼロ以外のフローラベルの組み合わせによって行います。フローに所属しないパケットは、ゼロに設定されたフローラベルを運びます。

フローのソースノードでは、フローにフローラベルを割り当てます。新しいフローラベルは、擬似的な方法で、ランダムに選択します。16 進数で、1 から FFFFF の範囲から均等に選択します。ランダムに割り当てることにより、ルーターはフローラベルフィールド内の任意のビットセットをハッシュキーとして利用できます。ルーターは、ハッシュキーを使ってフローに関連付けられた状態を調べることができます。

同じフローに所属するパケット

同じフローに所属するパケットは、同じソースアドレス、同じ宛先アドレス、同じゼロ以外のフローラベルで送信します。これらのパケットのどれかにホップバイホップオプションヘッダーが含まれる場合、パケットを同じホップバイホップオプションヘッダーの内容で生成する必要があります。ただし、ホップバイホップオプションヘッダーの次のヘッダーフィールドは除かれます。これらのパケットのどれかにルーティングヘッダーが含まれる場合、パケットの拡張ヘッダーを同じ内容で生成する必要があります。この同じ内容には、ルーティングヘッダーより前のすべての拡張ヘッダーと、ルーティングヘッダーが含まれます。ただし、ルーティングヘッダーの次のヘッダーフィールドは除かれます。ルーターや宛先では、場合によってはこれらの条件が満たされているかを確認できます。違反を検出した場合、そのことを送信元に報告する必要があります。違反を報告するには、ICMP パラメータ問題メッセージ、コード 0 を使用します。違反は、フローラベルフィールドの上位オクテットで表されます。この上位オクテットは、IPv6 パケット内のオフセット 1 オクテットです。

ルーターは、任意のフローのフロー処理状態を自由にセットアップできます。この場合ルーターは、制御プロトコル、ホップバイホップオプション、その他の手段による、明示的なフロー確立情報を必要としません。たとえば、未知のゼロ以外に設定されたフローラベルを持つパケットを特定のソースから受信した場合、ルーターではその IPv6 ヘッダーを処理できます。ルーターは、フローラベルがゼロに設定されている拡張ヘッダーを処理する場合と同じ方法で、必要な拡張ヘッダーを処理できます。ルーターは、次中継点のインタフェースの判別を行います。場合によってはホップバイホップオプションの更新、ルーティングヘッダーのポインタとアドレスの加算、あるいはパケットのキューイングの方法の決定なども行います。パケットのキューイングの方法の決定は、パケットのトラフィッククラスフィールドに基づいて行われます。ルーターは、この処理手順の結果を記憶することを選択できます。そして、記憶した後でその情報をキャッシュに保存できます。始点アドレスとフローラベルがキャッシュキーとして使用されます。同じ始点アドレスとフローラベルを持つ後続のパケット については、キャッシュされた情報を参照することにより処理できます。これらのパケットの始点アドレスとフローラベルをすべて調べる必要はありません。ルーターは、フローの最初のパケットは確認しますが、その後はフィールドの内容は変更されないと仮定することができます。

トラフィッククラス

パケットを生成したノードは、IPv6 パケットの異なるクラスまたは優先順位を識別する必要があります。その場合、IPv6 ヘッダーのトラフィッククラスフィールドが使用されます。パケットを転送するルーターも同じ目的でトラフィッククラスフィールドを使用します。

トラフィッククラスフィールドには、以下の一般的な要件が適用されます。

- 1 つのノード内の IPv6 サービスへのサービスインタフェースは、上位層プロトコルに対して、トラフィッククラスビットの値を提供する必要があります。上位層プロトコルで生成されたパケットにはトラフィッククラスビットが必要です。デフォルト値は、8 ビットすべてが 0 です。

- 一部またはすべてのトラフィッククラスビットをサポートするノードは、ビットの値を変更することができます。変更できるのは、サポートする特定の使用方法に従ってそのノードが生成、転送、または受信するパケット内のビットの値です。ノードは、特定の使用方法をサポートしないすべてのトラフィッククラスフィールド内のビットを無視し、変更しないようにしなければなりません。
- 受信パケット内のトラフィッククラスビットは、そのパケットの発信元が送信した値とは異なる値である可能性があります。したがって、上位層プロトコルは、トラフィッククラスビットの値が同じであると仮定することはできません。

IPv6 セキュリティの強化

現在のインターネットには多くのセキュリティ問題があります。インターネットでは、アプリケーション層より下の層には有効な機密機構や認証機構がありません。この欠点に対し、IPv6 では、セキュリティサービスを提供する 2 つの統合オプションを設けて対応しています。この 2 つのオプションは、別々に、あるいはまとめて使用してさまざまなユーザーにさまざまなセキュリティレベルを提供できます。ユーザー通信が異なれば、セキュリティのニーズも異なります。

最初のオプションは IPv6 認証ヘッダー (AH) と呼ばれる拡張ヘッダーです。この拡張ヘッダーは、IPv6 データグラムに機密性を持たない認証と完全性を提供します。この拡張機能はアルゴリズムに依存せず、さまざまな認証方式をサポートします。認証ヘッダーは、ワールドワイドなインターネット内の相互運用性の保証を支援するために、それを使用することが提案されています。認証ヘッダーを使用することにより、ホストなりすまし攻撃など、主なネットワーク侵害を回避できます。IPv6 でソースルーティングを使用する場合、IP ソースルーティングに明らかな危険性があるので IPv6 認証ヘッダーが重要になります。上位層プロトコルおよび上位層サービスには、現在有効な保護策はありません。しかし、インターネット層に認証ヘッダーを使用することで、ホスト発信元認証を提供できます。

2 番めのオプションである、IPv6 カプセル化セキュリティペイロード (ESP) と呼ばれる拡張ヘッダーは、IPv6 データグラムに完全性と機密性を提供します。いくつかの同じようなのセキュリティプロトコル よりも単純ですが、ESP はフレキシブルなままで、独立したアルゴリズムです同様のセキュリティプロトコルには、SP3D とISO NLSP が含まれます。

IPv6 認証ヘッダーと IPv6 カプセル化セキュリティヘッダーは、新しいインターネットプロトコルセキュリティ (IPsec) の機能です。IPsec の概要については、『*Solaris* のシステム管理 *(IP* サービス*)*』の「IPsec (概要)」を参照してください。IP sec の実装の方法については、『*Solaris* のシステム管理 *(IP* サービス*)*』の「IPsec の管理 (手順)」を参照してください。

第 2 章

IPv6 の管理 (手順)

この章では、IPv6 や IPv6 ルーターを有効にする方法、IPv6 アドレスを DNS、NIS、NIS+ 用に設定する方法、ルーター間のトンネルの作成方法、診断情報を表示する IPv6 の追加コマンドを実行する方法、IPv6 ネームサービス情報の表示方法について説明します。

この章では、以下の内容について説明します。

- 33 ページの「IPv6 ノードを有効にする」
- 34 ページの「IPv6 ノードを有効にする (作業マップ)」
- 38 ページの「IPv6 の監視」
- 38 ページの「IPv6 の監視 (作業マップ)」
- 46 ページの「IP 内 IP トンネルの設定」
- 46 ページの「IP 内 IP トンネルの設定 (作業マップ)」
- 50 ページの「IPv6 ネームサービス情報の表示」
- 50 ページの「IPv6 ネームサービス情報を表示する (作業マップ)」

トピック	情報
IPv6 の概要	第 1 章
IPv4 から IPv6 への移行	第 4 章
この章で説明する手順に関する概念情報	第 3 章

IPv6 ノードを有効にする

この節では、IPv6 ノードをネットワークで設定するときに必要な手順について説明します。

注 – この節で「ノード」という用語は、Solaris サーバーまたはクライアントワークステーションを指します。

IPv6 ノードを有効にする (作業マップ)

表 **2–1** IPv6 ノードを有効にする (作業マップ)

作業	説明	参照個所
ノード上の IPv6 を有効にする	`hostname6.`*interface* ファイルの操作、アドレスの表示、`/etc/inet/ipnodes` ファイルへのアドレスの入力。この表の「注」参照。	34 ページの「ノード上の IPv6 を有効にする方法」
Solaris IPv6 ルーターの設定	`indp.conf` ファイルへのエントリの追加	35 ページの「Solaris IPv6 ルーターの設定方法」
IPv6 アドレスを NIS と NIS+ に追加	`/etc/ipnodes` ファイルへのエントリ追加	36 ページの「NIS と NIS+ に対する IPv6 アドレスの追加方法」
IPv6 アドレスを DNS に追加	DNS ゾーンと逆ゾーンファイルに対する AAAA レコードの追加	37 ページの「DNS に対する IPv6 アドレスの追加方法」

注 – IPv6 は、Solaris ソフトウェアをインストールするときにシステムで有効にできます。インストールプロセスで `yes` と応答して IPv6 を有効にすると、あとの IPv6 を有効にする手順を省略できます。

▼ ノード上の IPv6 を有効にする方法

1. **IPv6** を有効にしたいシステム上でスーパーユーザーになります。
2. コマンド行で、各インタフェースに対して次のように入力します。

   ```
   # touch /etc/hostname6.interface
   ```

   interface `le0`、`le1` などのインタフェース名

3. リブートします。

注 – リブートすると、ルーター検索パケットが送信されます。ルーターがプレフィックスを応答することにより、ノードが IP アドレスでインタフェースを設定できるようになります。リブートすると、主なネットワークデーモンも IPv6 モードで再起動します。

---

4. コマンド行で次のコマンドを入力して **IPv6** アドレスを表示します。

   ```
   # ifconfig -a6
   ```

   IPv4 アドレスと IPv6 アドレスを表示するには、`-a` オプションだけを使用します。

5. 適切なネームサービスに、**IPv6** アドレスを次のように追加します。

   - NIS と NIS+ については、36 ページの「NIS と NIS+ に対する IPv6 アドレスの追加方法」を参照してください。
   - DNS については、37 ページの「DNS に対する IPv6 アドレスの追加方法」を参照してください。

▼ Solaris IPv6 ルーターの設定方法

1. ルーターとして機能するシステム上で、スーパーユーザーになります。

2. `/etc/inet/ndpd.conf` ファイルを編集して、サブネットプレフィックスを使用して次のエントリを **1** つまたは複数追加します。

   変数と使用できる値のリストについては、`in.ndpd(1M)` のマニュアルページを参照してください。`ndpd.conf` ファイルについては、`ndpd.conf(4)` のマニュアルページを参照してください。

   a. すべてのインタフェースについて、ルーター動作を指定するエントリを追加します。

      `ifdefault` *variable value*

   b. プレフィックス通知のデフォルト動作を指定するエントリを追加します。

      `prefixdefault` *variable value*

   c. インタフェースパラメータごとのセットエントリを追加します。

      `if` *interface variable value*

   d. インタフェースプレフィックス情報ごとの通知エントリを追加します。

      `prefix` *prefix/length interface variable value*

3. リブートします。

注 – ホストのサブネットアドレスプレフィックスが、近傍検索 (in.ndpd) からホストにリレーされます。また、次世代 RIP ルーティングプロトコル (in.ripngd) が自動的に実行されます。

例 – ndpd.conf ルーター構成ファイル

```
# Send router advertisements out all NICs
ifdefault AdvSendAdvertisements on
# Advertise a global prefix and a
# site local prefix on three interfaces.
# 0x9255 = 146.85
prefix 2:0:0:9255::0/64          hme0
prefix fec0:0:0:9255::0/64      hme0
# 0x9256 = 146.86
prefix 2:0:0:9256::0/64      hme1
prefix fec0:0:0:9256::0/64     hme1
# 0x9259 = 146.89
prefix 2:0:0:9259::0/64         hme2
prefix fec0:0:0:9259::0/64     hme2
```

▼ NIS と NIS+ に対する IPv6 アドレスの追加方法

NIS+ 用に ipnodes.org_dir という新しいテーブルが追加されました。このテーブルには、ホスト用の IPv4 アドレスと IPv6 アドレスの両方が保存されています。既存の IPv4 情報だけを保持している hosts.org_dir テーブルは、既存のアプリケーションを円滑に実行するために残しておきます。hosts.org_dir テーブルと ipnodes.org_dir テーブルはどちらも IPv4 アドレスと整合させておく必要があります。概要については、69 ページの「Solaris ネームサービスに対する IPv6 拡張機能」を参照してください。

新しい ipnodes.org_dir テーブルの管理方法は、hosts.org_dir の管理方法と似ています。従来の NIS+ テーブルの管理に使用したのと同じツールとユーティリティは、ipnodes.org_dir にも有効です。NIS+ テーブルの操作についての詳細は、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』を参照してください。

次の手順では、/etc/inet/ipnodes のエントリを ipnodes.org_dir テーブルに詳細モードでマージします。NIS+ テーブルは、nistbladm(1)、nissetup(1M)、または nisserver(1M) のどれかで作成されたものとします。

- コマンド行で、次のコマンドを入力します。

  ```
  % nisaddent -mv -f /etc/inet/ipnodes ipnodes
  ```

  ipnodes.org_dir テーブルを表示するには、次のように操作します。

- コマンド行で、次のコマンドを入力します。

```
% nisaddent -d ipnodes
```

NIS 用に 2 つの新しいマップが追加されました。 `ipnodes.byname` と `ipnodes.byaddr` です。これらのマップは、いずれも IPv4 と IPv6 のホスト名とアドレスの関連付けを保存しています。`hosts.byname` マップと `hosts.byaddr` マップは、IPv4 のホスト名とアドレスの関連情報だけを保存していますが、既存のアプリケーションが動作できるように変更されていません。新しいマップの管理は、以前の `hosts.byname` マップと `hosts.byaddr` マップの管理方法と同様です。*hosts* マップを IPv4 アドレスで更新すると、新しい ipnode マップも同じ情報で更新されることに注意してください。

注 – IPv6 対応ツールは、新しい NIS マップと新しい NIS+ テーブルを使用します。

▼ DNS に対する IPv6 アドレスの追加方法

1. **DNS** があるシステム上でスーパーユーザーになります。
2. **DNS** ゾーンファイルに、**IPv6** 有効化ホストの **AAAA** レコードを次のフォーマットで追加して編集します。

   host-name `IN AAAA` *host-address*

3. **DNS** 逆ゾーンファイルを編集し、次のフォーマットで **PTR** レコードを追加します。

   host-address `IN PTR` *host-name*

   AAAA レコードと PTR レコードの詳細については、RFC 1886 を参照してください。

例 – DNS ゾーンファイル

```
vallejo        IN        AAAA   2::9256:a00:20ff:fe12
IN        AAAA   fec0::9256:a00:20ff:fe12:528
```

例 – DNS 逆ゾーンファイル

```
$ORIGIN    ip6.int.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.2.0.0.0 \
    IN        PTR        vallejo.Eng.apex.COM.
8.2.5.0.2.1.e.f.f.f.9.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.0.c.e.f \
    IN        PTR        vallejo.Eng.apex.COM.
```

IPv6 の監視

次のコマンドは IPv6 の Solaris 実装に対応するように変更されています。

- `ifconfig(1M)`
- `netstat(1M)`
- `snoop(1M)`
- `ping(1M)`
- `traceroute(1M)`

追加コマンドを使用すると診断を実行できます。これらのコマンドの考え方については、57 ページの「`ifconfig` ユーティリティに対する IPv6 拡張機能」と 64 ページの「既存のユーティリティに対する IPv6 拡張機能」を参照してください。

IPv6 の監視 (作業マップ)

表 **2–2** IPv6 の監視 (作業マップ)

作業	説明	参照個所
インタフェースアドレス割り当ての表示	`ifconfig` コマンドで、すべてのアドレス割り当て、または IPv4 か IPv6 アドレス割り当てだけを表示	39 ページの「インタフェースアドレス割り当ての表示方法」
ネットワーク状態の表示	すべてのソケットとルーティングテーブルエントリ、IPv4 用の inet アドレスファミリー、IPv6 用の inet6 アドレスファミリー、`netstat` コマンドによるインタフェースの IPv6 または ICMPv6 カウンタの統計を表示	40 ページの「ネットワーク状態の表示方法」
IPv6 関連コマンドの出力表示の制御	`ping` コマンド、`netstat` コマンド、 `ifconfig` コマンド、`traceroute` コマンドの出力の制御。`inet_type` という名前のファイルの作成と、そのファイル内の DEFAULT_IP 変数の設定	43 ページの「IPv6 関連コマンドの出力表示の制御方法」
IPv6 ネットワークトラフィックだけの監視	`snoop` コマンドによるすべての IPv6 パケットの表示	44 ページの「IPv6 ネットワークトラフィックの監視方法」
すべてのマルチホームホストアドレスの探査	`ping` コマンドによるすべてのアドレスの確認	45 ページの「すべてのマルチホームホストアドレスの探査方法」
すべてのルートのトレース	`traceroute` コマンドの使用	45 ページの「すべてのルーターのトレース方法」

▼ インタフェースアドレス割り当ての表示方法

IPv4 や IPv6 のアドレス割り当ての場合だけでなく、すべてのアドレス割り当てを表示する場合も `ifconfig` コマンドを使用します。

- コマンド行で次のコマンドを入力します。

```
% ifconfig [option]
```

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

例 – すべてのインタフェースについてアドレス指定情報を表示

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.146.86.255
        ether 8:0:73:56:a8
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

例 – すべての IPv4 インタフェースについてアドレス指定情報を表示

```
% ifconfig -a4
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

例 – すべての IPv6 インタフェースについてアドレス指定情報を表示

```
% ifconfig -a6
lo0: flags=2000849 mtu 8252 index 1
        inet6 ::1/128
```

```
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:20ff:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

▼ ネットワーク状態の表示方法

次の手順では、 `netstat` コマンドで、次に示すネットワークデータ構造フォーマットを表示できます。

- すべてのソケットとルーティングテーブルのエントリ
- IPv4 用の inet アドレスファミリ
- IPv6 用の inet6 アドレスファミリ
- インタフェース別統計 – IPv6/ICMPv6 カウンタ

● コマンド行で次のコマンドを入力します。

```
% netstat [option]
```

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

例 – すべてのソケットとルーティングテーブルエントリの表示

```
% netstat -a
UDP: IPv4
   Local Address         Remote Address     State
-------------------- -------------------- -------
      *.*                                  Unbound
      *.apexrpc                             Idle
      *.*                                  Unbound
            .
            .
UDP: IPv6
   Local Address                    Remote Address                   State
If
--------------------------------- ---------------------------------- -------
      *.*                                                           Unbound
      *.time                                                        Idle
      *.echo                                                        Idle
      *.discard                                                     Idle
      *.daytime                                                     Idle
      *.chargen                                                     Idle

TCP: IPv4
```

```
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
      *.*                  *.*                0      0     0      0 IDLE
      *.apexrpc            *.*                0      0     0      0 LISTEN
      *.*                  *.*                0      0     0      0 IDLE
      *.ftp                *.*                0      0     0      0 LISTEN
localhost.427              *.*                0      0     0      0 LISTEN
      *.telnet             *.*                0      0     0      0 LISTEN
tn.apex.COM.telnet is.Eng.apex.COM         8760      0   8760     0 ESTABLISHED
tn.apex.COM.33528 np.apex.COM.46637        8760      0   8760     0 TIME_WAIT
tn.apex.COM.33529 np.apex.COM.apexrpc      8760      0   8760     0 TIME_WAIT
TCP: IPv6
   Local Address     Remote Address   Swind Send-Q  Rwind Recv-Q   State  If
----------------- -----------------   ----- ------  ----- ------   -----
      *.*                  *.*             0      0      0      0   IDLE
      *.ftp                *.*             0      0      0      0   LISTEN
      *.telnet             *.*             0      0      0      0   LISTEN
      *.shell              *.*             0      0      0      0   LISTEN
      *.smtp               *.*             0      0      0      0   LISTEN
        .
        .
 2::56:8.login     something.1023    8640      0   8640      0 ESTABLISHED
 fe80::a:a8.echo   fe80::a:89        8640      0   8640      0 ESTABLISHED
 fe80::a:a8.ftp    fe80::a:90        8640      0   8640      0 ESTABLISHED
```

例 – IPv4 用の inet アドレスファミリを表示

```
% netstat -f inet
TCP: IPv4
   Local Address        Remote Address    Swind Send-Q Rwind Recv-Q  State
-------------------- -------------------- ----- ------ ----- ------ -------
tn.apex.COM.telnet    is.apex.COM.35388    8760      0  8760   0  ESTABLISHED
tn.apex.COM.1022      alive-v4.nfsd        8760      0  8760   0  ESTABLISHED
tn.apex.COM.1021      sl.apex.COM.nfsd     8760      0  8760   0  ESTABLISHED
.
.
tn.apex.COM.33539     np.apex.COM.apexrpc  8760      0  8760   0  TIME_WAIT
```

例 – IPv6 用の inet6 アドレスファミリを表示

```
% netstat -f inet6
TCP: IPv6
   Local Address        Remote Address     Swind Send-Q Rwind Recv-Q   State      If
-------------------- -------------------- ----- ------ ----- ------ ----------- -----
2::56:a8.login        something.1023       8640      0  8640      0 ESTABLISHED
fe80::a0:a8.echo      fe80::a0:de.35389    8640      0  8640      0 ESTABLISHED
.
.
fe80::a0:a8.ftp-data  fe80::a0:de.35394   25920      0 25920      0 TIME_WAIT
```

例 – インタフェース別統計を表示 - IPv6 / ICMPv6 カウンタ

```
% netstat -sa
RAWIP
        rawipInDatagrams    =  1407     rawipInErrors       =     0
        rawipInCksumErrs    =     0     rawipOutDatagrams   =     5
        rawipOutErrors      =     0

UDP
        udpInDatagrams      =  7900     udpInErrors         =     0
        udpOutDatagrams     =  7725     udpOutErrors        =     0

TCP     tcpRtoAlgorithm     =     4     tcpRtoMin           =   200
        tcpRtoMax           = 60000     tcpMaxConn          =    -1
                .
                .
IPv4    ipForwarding        =     2     ipDefaultTTL        =   255
        ipInReceives        =406345     ipInHdrErrors       =     0
        ipInAddrErrors      =     0     ipInCksumErrs       =     0
                .
                .
IPv6 for lo0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =     0
        ipv6InReceives      =     0     ipv6InHdrErrors     =     0
                .
                .
IPv6 for le0
        ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
                .
                .
IPv6    ipv6Forwarding      =     2     ipv6DefaultHopLimit =   255
        ipv6InReceives      =   885     ipv6InHdrErrors     =     0
                .
                .
ICMPv4  icmpInMsgs          =   618     icmpInErrors        =     0
        icmpInCksumErrs     =     0     icmpInUnknowns      =     0
        icmpInDestUnreachs  =     5     icmpInTimeExcds     =     0
                .
                .
ICMPv6 for lo0
        icmp6InMsgs         =     0     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
                .
                .
ICMPv6 for le0
        icmp6InMsgs         =   796     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
        icmp6InTimeExcds    =     0     icmp6InParmProblems =     0
                .
                .
ICMPv6  icmp6InMsgs         =   796     icmp6InErrors       =     0
        icmp6InDestUnreachs =     0     icmp6InAdminProhibs =     0
```

```
                .
                .
IGMP:
       2542 messages received
          0 messages received with too few bytes
          0 messages received with bad checksum
       2542 membership queries received
                .
                .
```

▼ IPv6 関連コマンドの出力表示の制御方法

netstat コマンドと ifconfig コマンドの出力は制御できます。まず、/etc/default ディレクトリで inet_type という名のファイルを作成します。次に、DEFAULT_IP 変数の値を指定します。inet_type 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。

1. /etc/default/inet_type ファイルを作成します。
2. 必要に応じて、次のいずれかのエントリを作成します。
   - IPv4 情報だけを表示するには、次のように入力します。

     ```
     DEFAULT_IP=IP_VERSION4
     ```
   - IPv4 情報と IPv6 情報を表示するには、次のいずれかを入力します。

     ```
     DEFAULT_IP=BOTH
     ```

     または

     ```
     DEFAULT_IP=IP_VERSION6
     ```

注 – ifconfig の -4 フラグと -6 フラグの設定は、inet_type ファイルに設定された値より優先します。また、netstat の -f フラグの設定も、inet_type ファイルに設定された値より優先します。

例 – IPv4 情報と IPv6 情報を選択する出力の制御

- DEFAULT_IP=BOTH または DEFAULT_IP=IP_VERSION6 変数を inet_type ファイルで設定する場合、次の結果が得られます。

```
% ifconfig -a
lo0: flags=1000849 mtu 8232 index 1
        inet 120.10.0.1 netmask ff000000
le0: flags=1000843 mtu 1500 index 2
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
lo0: flags=2000849 mtu 8252 index 1
```

```
        inet6 ::1/128
le0: flags=2000841 mtu 1500 index 2
        ether 8:0:20:56:a8
        inet6 fe80::a00:fe73:56a8/10
le0:1: flags=2080841 mtu 1500 index 2
        inet6 fec0::56:a00:fe73:56a8/64
le0:2: flags=2080841 mtu 1500 index 2
        inet6 2::56:a00:fe73:56a8/64
```

- DEFAULT_IP=IP_VERSION4 変数を `inet_type` ファイルで設定する場合、次の結果が得られます。

```
% ifconfig -a
lo0: flags=849 mtu 8232
        inet 120.10.0.1 netmask ff000000
le0: flags=843 mtu 1500
        inet 120.46.86.54 netmask ffffff00 broadcast 120.46.86.255
        ether 8:0:20:56:a8
```

▼ IPv6 ネットワークトラフィックの監視方法

すべての IPv6 パケットを表示するためには、次のように `snoop` コマンドを実行します。

1. スーパーユーザーになります。
2. コマンド行で次のコマンドを入力します。

```
# snoop ip6
```

`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。

例 – IPv6 ネットワークトラフィックだけの表示

```
# snoop ip6
Using device /dev/le (promiscuous mode)
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=104
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=104
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=152
fe80::a0:a1 -> ff02::9     IPv6  S=fe80::a0:a1 D=ff02::9 LEN=892
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:a8 -> fe80::a0:de IPv6  S=fe80::a0:a8 D=fe80::a0:de LEN=72
fe80::a0:de -> fe80::a0:a8 IPv6  S=fe80::a0:de D=fe80::a0:a8 LEN=72
```

▼ すべてのマルチホームホストアドレスの探査方法

この操作では、ping コマンドですべてのアドレスを調べます。

- コマンド行で次のコマンドを入力します。

```
% ping -a ipng11
ipng11 (2::102:a00:fe79:19b0) is alive
ipng11 (fec0::102:a00:fe79:19b0) is alive
ipng11 (190.68.10.75) is alive
```

ping コマンドについての詳細は、ping(1M) のマニュアルページを参照してください。

▼ すべてのルーターのトレース方法

この操作では、traceroute コマンドですべてのルーターを調べます。

- コマンド行で次のコマンドを入力します。

```
% traceroute -a <hostname>
```

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

例 – すべてのルーターのトレース

```
% traceroute -a ipng11
traceroute: Warning: Multiple interfaces found; using 2::56:a0:a8 @ le0:2
traceroute to ipng11 (2::102:a00:fe79:19b0),30 hops max, 60 byte packets
 1  ipng-rout86 (2::56:a00:fe1f:59a1)  35.534 ms  56.998 ms *
 2  2::255:0:c0a8:717  32.659 ms  39.444 ms *
 3  ipng61.Eng.apex.COM (2::103:a00:fe9a:ce7b)  401.518 ms  7.143 ms *
 4  ipng12-00 (2::100:a00:fe7c:cf35)  113.034 ms  7.949 ms *
 5  ipng11 (2::102:a00:fe79:19b0)  66.111 ms *  36.965 ms

traceroute: Warning: Multiple interfaces found; using fec0::56:a8 @ le0:1
traceroute to ipng11 (fec0::10:b0), 30 hops max, 60 byte packets
 1  ipng-rout86 (fec0::56:a00:fe1f:59a1) 96.342 ms  78.282 ms 88.327 ms
 2  ipng8-tun1 (fec0::25:0:0:c0a8:717)  268.614 ms  508.416 ms  438.774 ms
 3  ipng61.Eng.apex.COM (fec0::103:a00:fe9a:ce7b)  6.356 ms * 713.166 ms
 4  ipng12-00 (fec0::100:a00:fe7c:cf35)  7.409 ms * 122.094 ms
 5  ipng11 (fec0::102:a00:fe79:19b0)  10.620 ms * *

traceroute to ipng11.eng.apex.com (190.68.10.75),30 hops max,40 byte packets
 1  rmpj17c-086.Eng.apex.COM (120.46.86.1)  4.360 ms  3.452 ms  3.479 ms
 2  flrmpj17u.Eng.apex.COM (120.46.17.131)  4.062 ms  3.848 ms  3.505 ms
 3  ipng8.Eng.apex.COM (120.68.7.23)  4.773 ms *  4.294 ms
 4  ipng61.Eng.apex.COM (120.68.10.104)  5.128 ms  5.362 ms *
 5  ipng12-20.Eng.apex.COM (120.68.10.62)  7.298 ms  5.444 ms *
```

```
6  ipng11.Eng.apex.COM (120.68.10.75)  8.053 ms  6.394 ms *
```

IP 内 IP トンネルの設定

ここでは、IP 内 IP トンネルの設定方法について説明します。カプセル化には次の種類があります。

- IPv4 トンネル経由の IPv4
- IPv4 トンネル経由の IPv6
- IPv6 トンネル経由の IPv6
- IPv6 トンネル経由の IPv4

トンネルの概念については、67 ページの「IPv6 の Solaris トンネルインタフェース」と 78 ページの「トンネル機構」を参照してください。

IP 内 IP トンネルの設定 (作業マップ)

表 2–3 IP 内 IP トンネルを設定する (作業マップ)

作業	説明	手順の説明
IPv4 経由の IPv6 トンネルの設定	`hostname6.ip.tun` *n* ファイルに必要なエントリを表示する	46 ページの「IPv4 トンネルによる IPv6 の設定方法」
IPv6 経由の IPv6 トンネルの設定	`hostname6.ip6.tun` *n* ファイルに必要なエントリを表示する	47 ページの「IPv6 トンネル経由の IPv6 の設定方法」
IPv6 経由の IPv4 トンネルの設定	`hostname.ip6.tun` *n* ファイルに必要なエントリを表示する	48 ページの「IPv6 トンネル経由の IPv4 の設定方法」
IPv4 経由の IPv4 トンネルの設定	`hostname.ip.tun` *n* ファイルに必要なエントリを表示する	49 ページの「IPv4 トンネル経由の IPv4 の設定方法」
トンネルインタフェースに通知するルーターの設定	`/etc/inet/ndpd.conf` ファイルに必要なエントリを表示する	50 ページの「トンネルインタフェースで通知するためのルーターの設定方法」

▼ IPv4 トンネルによる IPv6 の設定方法

1. スーパーユーザーになります。

2. /etc/hostname6.ip.tun *n* ファイルを作成します。*n* には **0**、**1**、**2** などの値を使用します。次に、以下の手順に従って、エントリを追加します。

   a. トンネルソースアドレスとトンネル宛先アドレスを追加します。

      tsrc *IPv4-source-addr* tdst *IPv4-destination-addr* up

   b. **(**省略可能**)** ソース **IPv6** アドレスと宛先 **IPv6** アドレスの論理インタフェースを追加します。

      addif *IPv6-source-address IPv6-destination-address* up

      このインタフェースに対してアドレスを自動設定したい場合は、この手順を省きます。各トンネルに対するリンクローカルアドレスを設定する必要はありません。リンクローカルアドレスは自動的に設定されます。

トンネルを設定したあと、リブートしてください。

注 – 双方向通信を実現するには、トンネルのもう一方の端についても同じ手順を行う必要があります。

使用するシステムをルーターとして設定する場合、リブートする前にトンネルインタフェースに通知するようにルーターを設定する必要もあります。50 ページの「トンネルインタフェースで通知するためのルーターの設定方法」 を参照してください。

例 — IPv6 アドレスを自動設定するための IPv6 構成ファイルのエントリ

次に、すべての IPv6 アドレスが自動設定されるトンネルの例を示します。

```
tsrc 129.146.86.138 tdst 192.168.7.19 up
```

例 — 手動で設定されたアドレスの IPv6 構成ファイルのエントリ

次に、グローバルソースアドレスとグローバル宛先アドレスが手動で設定されるトンネルの例を示します。サイトローカルソースアドレスとサイトローカル宛先アドレスも手動で設定されます。

```
tsrc 120.46.86.138 tdst 190.68.7.19 up
addif fec0::1234:a00:fe12:528 fec0::5678:a00:20ff:fe12:1234 up
addif 2::1234:a00:fe12:528 2::5678:a00:20ff:fe12:1234 up
```

▼ IPv6 トンネル経由の IPv6 の設定方法

1. スーパーユーザーになります。

2. /etc/hostname6.ip6.tun *n* ファイルを作成します。*n* には **0**、**1**、**2** などの値を使用します。次に、以下の手順に従って、エントリを追加します。

   a. トンネルソースアドレスとトンネル宛先アドレスを追加します。

      `tsrc` *IPv6-source-address* `tdst` *IPv6-destination-address* `up`

   b. **(**省略可能**)** ソース **IPv6** アドレスと宛先 **IPv6** アドレスの論理インタフェースを追加します。

      `addif` *IPv6-source-address IPv6-destination-address* `up`

      このインタフェースに対してアドレスを自動設定したい場合は、この手順を省きます。各トンネルに対するリンクローカルアドレスを設定する必要はありません。リンクローカルアドレスは自動的に設定されます。

トンネルを設定したあと、リブートしてください。

注 – 双方向通信を実現するには、トンネルのもう一方の端についても同じ手順を行う必要があります。

使用するシステムをルーターとして設定する場合、リブートする前にトンネルインタフェースに通知するようにルーターを設定する必要もあります。50 ページの「トンネルインタフェースで通知するためのルーターの設定方法」 を参照してください。

例 — IPv6 トンネル経由で IPv6 を作成するための IPv6 構成ファイルのエントリ

次に、IPv6 トンネルによる IPv6 のエントリの例を示します。

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3 up
```

▼ IPv6 トンネル経由の IPv4 の設定方法

1. スーパーユーザーになります。

2. /etc/hostname.ip6.tun *n* ファイルを作成します。*n* には **0**、**1**、**2** などの値を使用します。次に、以下の手順に従って、エントリを追加します。

   a. トンネルソースアドレスとトンネル宛先アドレスを追加します。

      `tsrc` *IPv6-source-address* `tdst` *IPv6-destination-address*
      tunnel-IPv4-source-address tunnel-IPv4-destination-address `up`

   b. **(**省略可能**)** ソース **IPv6** アドレスと宛先 **IPv6** アドレスの論理インタフェースを追加します。

```
addif IPv6-source-address  IPv6-destination-address up
```

トンネルを設定したあと、リブートしてください。

注 – 双方向通信を実現するには、トンネルのもう一方の端についても同じ手順を行う必要があります。

使用するシステムをルーターとして設定する場合、リブートする前にトンネルインタフェースに通知するようにルーターを設定する必要もあります。50 ページの「トンネルインタフェースで通知するためのルーターの設定方法」 を参照してください。

例 — IPv6 トンネル経由で IPv4 を作成するための IPv4 構成ファイルのエントリ

次に、IPv6 トンネル経由の IPv4 のエントリの例を示します。

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

▼ IPv4 トンネル経由の IPv4 の設定方法

1. スーパーユーザーになります。
2. `/etc/hostname.ip.tun`*n* ファイルを作成します。*n* には **0**、**1**、**2** などの値を使用します。次に、以下の手順に従って、エントリを追加します。

   a. トンネルソースアドレスとトンネル宛先アドレスを追加します。

   ```
   tsrc IPv4-source-address tdst IPv4-destination-address
   tunnel-IPv4-source-address  tunnel-IPv4-destination-address up
   ```

   b. (省略可能**)** ソース **IPv4** アドレスと宛先 **IPv4** アドレスの論理インタフェースを追加します。

   ```
   addif IPv4-source-address  IPv4-destination-address up
   ```

トンネルを設定したあと、リブートしてください。

注 – 双方向通信を実現するには、トンネルのもう一方の端についても同じ手順を行う必要があります。

使用するシステムをルーターとして設定する場合、リブートする前にトンネルインタフェースに通知するようにルーターを設定する必要もあります。50 ページの「トンネルインタフェースで通知するためのルーターの設定方法」 を参照してください。

例 — IPv4 トンネル経由で IPv4 を作成するための IPv4 構成ファイルのエントリ

次に、IPv4 トンネル経由の IPv4 のエントリの例を示します。

```
tsrc 120.46.86.158 tdst 120.46.86.122
10.0.0.4 10.0.0.61 up
```

▼ トンネルインタフェースで通知するためのルーターの設定方法

トンネルごとに次の操作をします。

1. スーパーユーザーになります。
2. `/etc/inet/ndpd.conf` ファイルを編集します。次の手順に従って、エントリを追加します。
   a. トンネルインタフェース経由のルーター通知を有効にします。

      ```
      if ip.tun n AdvSendAdvertisements 1
      ```

   b. 必要に応じてプレフィックスを追加します。

      ```
      prefix interface-address ip.tun n
      ```

3. リブートします。

IPv6 ネームサービス情報の表示

ここでは、IPv6 ネームサービス情報を表示する手順について説明します。

IPv6 ネームサービス情報を表示する (作業マップ)

表 **2–4** IPv6 ネームサービス情報を表示する (作業マップ)

作業	説明	参照個所
IPv6 ネームサービス情報の表示	`nslookup` コマンドで、IPv6 ネームサービス情報を表示する	51 ページの「IPv6 ネームサービス情報の表示方法」

表 **2–4** IPv6 ネームサービス情報を表示する (作業マップ)　(続き)

作業	説明	参照個所
DNS IPv6 PTR レコードの正確な更新の確認	nslookup コマンドで DNS IPv6 PTR レコードを表示する。また、set q=PTR パラメータを使用する	52 ページの「DNS IPv6 PTR レコードの正確な更新の確認方法」
NIS+ による IPv6 情報の表示	ypmatch コマンドで、IPv6 情報を NIS から表示する	52 ページの「NIS による IPv6 情報の表示方法」
NIS+ による IPv6 情報の表示	nismatch コマンドを実行して NIS+ で IPv6 情報を表示する	53 ページの「NIS+ による IPv6 情報の表示方法」
ネームサービスに依存しない IPv6 情報の表示	getent コマンドで IPv6 情報を表示する	53 ページの「ネームサービスに依存しない IPv6 情報の表示方法」

▼ IPv6 ネームサービス情報の表示方法

nslookup コマンドで IPv6 ネームサービス情報を表示するには、次のように操作します。

1. コマンド行で次のコマンドを入力します。

   % /usr/sbin/nslookup

   デフォルトサーバー名とアドレスが表示され、nslookup コマンドの山括弧 (>) プロンプトが表示されます。

2. 特定のホストの情報を表示するには、山括弧プロンプトに次のコマンドを入力します。

   >**set q=any**
   >*host-name*

3. **AAAA** レコードだけを表示するには、山括弧プロンプトに次のコマンドを入力します。

   >**set q=AAAA**

4. **exit** を入力して、コマンドを終了します。

例 – nslookup による IPv6 情報の表示

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=any
> vallejo
Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
```

```
vallejo.ipv6.eng.apex.com      IPv6 address = fec0::9256:a00:fe12:528
vallejo.ipv6.eng.apex.com      IPv6 address = 2::9256:a00:fe12:528
> exit
```

▼ DNS IPv6 PTR レコードの正確な更新の確認方法

nslookup コマンドを使用して DNS IPv6 PTR レコードを表示します。

1. コマンド行で次のコマンドを入力します。

   % /usr/sbin/nslookup

   デフォルトサーバー名とアドレスが表示され、nslookup コマンドの山括弧プロンプトが表示されます。

2. **PTR** レコードを表示するには、山括弧プロンプトに次のコマンドを入力します。

   >set q=PTR

3. **exit** を入力して、コマンドを終了します。

例 – nslookup による PTR レコードの表示

```
%  /usr/sbin/nslookup
Default Server:  space1999.Eng.apex.COM
Address:  120.46.168.78
> set q=PTR
> 8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.2.0.0.0.ip6.int

8.2.5.0.2.1.e.f.f.f.0.2.0.0.a.0.6.5.2.9.0.0.0.0.0.0.0.2.0.0.0.ip6.int name =
vallejo.ipv6.Eng.apex.COM
ip6.int nameserver = space1999.Eng.apex.COM
> exit
```

▼ NIS による IPv6 情報の表示方法

ypmatch コマンドを実行して NIS で IPv6 情報を表示するには、次のように操作します。

- コマンド行で次のコマンドを入力します。

  % ypmatch *host-name* **ipnodes.***byname*

  host-name に関する情報が表示されます。

  例 **2–1** 例 — ypmatch を使用して NIS で IPv6 情報を表示する

  ```
  % ypmatch vallejo ipnodes.byname
  fec0::9256:a00:20ff:fe12:528    vallejo
  2::9256:a00:20ff:fe12:528       vallejo
  ```

▼ NIS+ による IPv6 情報の表示方法

nismatch コマンドを実行して NIS で IPv6 情報を表示するには、次のように操作します。

- コマンド行で次のコマンドを入力します。

  % **nismatch** *host-name* **ipnodes.***org-dir*

  host-name に関する情報が表示されます。

  例 **2–2** 例 – nismatch を使用して NIS+ で IPv6 情報を表示する

```
% nismatch vallejo ipnodes.org_dir
vallejo vallejo fec0::9256:a00:20ff:fe12:528
vallejo vallejo 2::9256:a00:20ff:fe12:528
```

▼ ネームサービスに依存しない IPv6 情報の表示方法

- コマンド行で次のコマンドを入力します。

  % **getent ipnodes** *host-name*

  host-name に関する情報が表示されます。

  例 **2–3** 例 – getent を使用したネームサービスに依存しない IPv6 情報の表示

```
% getent ipnodes vallejo
2::56:a00:fe87:9aba        vallejo vallejo
fec0::56:a00:fe87:9aba     vallejo vallejo
```

第 3 章

IPv6 のファイルおよびコマンド (リファレンス)

Solaris の IPv6 の実装は、主にカーネルレベルとユーザーレベルの両方の TCP/IP スタックへの変更から構成されます。新しい IPv6 モジュールにより、トンネル、ルーター検索、ステートレスアドレス自動設定を使用できます。この章では、IPv6 の Solaris 実装に伴う概念について説明します。

この章では、以下の内容について説明します。

- 55 ページの「Solaris IPv6 実装の概要」
- 56 ページの「IPv6 ネットワークインタフェース構成ファイル」
- 59 ページの「複数のネットワークインタフェースがあるノード」
- 60 ページの「IPv6 デーモン」
- 64 ページの「既存のユーティリティに対する IPv6 拡張機能」
- 66 ページの「表示出力の制御」
- 67 ページの「IPv6 の Solaris トンネルインタフェース」
- 69 ページの「Solaris ネームサービスに対する IPv6 拡張機能」
- 72 ページの「NFS と RPC による IPv6 のサポート」
- 73 ページの「ATM 経由の IPV6 サポート」

Solaris IPv6 実装の概要

IPv4 から IPv6 への移行の一部として、IPv6 では IPv6 パケットを IPv4 パケット内にカプセル化する方式が指定されます。また、IPv6 では、IPv6 パケット内にカプセル化された IPv6 パケットも指定します。その結果、パケットのカプセル化を行う新しいモジュール `tun(7M)` が追加されました。このモジュールは、トンネルモジュールと呼ばれ、結合されます。このモジュールは、物理インタフェースと同様に `ifconfig` ユーティリティを使用して設定されます。 このモジュールによってトンネルモジュールが IP デバイスと IP モジュール間に配置されます。トンネルデバイスにもシステムインタフェースリストにエントリがあります。

ifconfig(1M) ユーティリティも変更されました。このユーティリティは、IPv6 スタックを作成し、新しいパラメータをサポートします。これらについてはこの章で、あとから説明します。

ルーター検索とステートレスアドレス自動設定を行うため、in.ndpd(1M) デーモンが追加されました。

IPv6 ネットワークインタフェース構成ファイル

IPv4 では起動時に /etc/hostname.*interface* を使用しましたが、IPv6 でも起動時にファイル /etc/hostname6.*interface* を使用してネットワークインタフェースを自動的に定義します。このとき、少なくとも /etc/hostname.* ファイルまたは、/etc/hostname6.* ファイルがローカルマシンに存在している必要があります。これらのファイルは、Solaris インストールプログラムで生成されます。ファイル名の *interface* は、プライマリネットワークインタフェースのデバイス名に置き換えられます。

ファイル名の構文は、次のとおりです。

hostname.*interface*
hostname6.*interface*

interface の構文は、次のとおりです。

dev[.*Module*[.*Module* ...]]*PPA*

Dev	ネットワークインタフェースデバイス。デバイスは le、qe など物理ネットワークインタフェースか、トンネルなどの論理インタフェース。詳細については、67 ページの「IPv6 の Solaris トンネルインタフェース」を参照してください。
Module	結合される際にデバイスにプッシュされるストリームモジュールのリスト
PPA	物理的な接続ポイント

構文 [.[.]] も可能です。

有効なファイル名は、次のとおりです。

```
hostname6.le0
hostname.ip.tun0
hostname.ip6.tun0
hostname6.ip.tun0
hostname6.ip6.tun0
```

IPv6 インタフェース構成ファイルのエントリ

IPv6 におけるインタフェースの自動設定では、その所属するリンク層アドレスに基づいてリンクローカルアドレスをノード側で計算できます。そのため、IPv6 インタフェース構成ファイルにはエントリがないことがあります。その場合、起動スクリプトによってインタフェースが設定されます。ノードは近傍検索デーモン `in.ndpd` で他のアドレスやプレフィックスの情報を取り出します。インタフェースに静的アドレスが必要な場合、`ifconfig` ユーティリティのコマンドインタフェースを使用します。その結果、アドレスまたはホスト名が `/etc/hostname6.`*interface* (または `/etc/hostname.`*interface*) に保存されます。*interface* (または `/etc/hostname.`*interface* を使用してネットワークインタフェースを定義します。インタフェースが構成されるときに、その内容が `ifconfig` に渡されます。

この場合、ファイルに含まれるエントリは 1 つだけです。このエントリは、ネットワークインタフェースに関連付けられたホスト名または IP アドレスです。たとえば、`ahaggar` というマシンの一次ネットワークインタフェースが `smc0` であるとします。その `/etc/hostname6.*` ファイル名は `/etc/hostname6.smc0` となります。そのエントリは `ahaggar` です。

ネットワーキングの起動スクリプトでは、ルーティングデーモンとパケット転送を開始するために、インタフェース数と、`/etc/inet/ndpd.conf` ファイルの有無を調べます。35 ページの「Solaris IPv6 ルーターの設定方法」を参照してください。

`ifconfig` ユーティリティに対する IPv6 拡張機能

`ifconfig` ユーティリティにより、IPv6 インタフェースとトンネルモジュールを結合できるようになりました。`ifconfig(1M)` ユーティリティでは、ioctl の拡張セットで IPv4 ネットワークインタフェースと IPv6 ネットワークインタフェースの両方を設定します。表 3–1 は、このユーティリティに追加されたオプションセットです。このユーティリティによる診断手順については、39 ページの「インタフェースアドレス割り当ての表示方法」を参照してください。

表 3–1 新しい `ifconfig` ユーティリティオプション

オプション	説明
`index`	インタフェースインデックスを設定する
`tsrc/ tdst`	トンネルソース / 宛先を設定する
`addif`	論理インタフェースの次の候補を作成する
`removeif`	指定された IP アドレスの論理インタフェースを削除する
`destination`	インタフェースにポイントツーポイント宛先アドレスを設定する

表 **3–1** 新しい ifconfig ユーティリティオプション　(続き)

オプション	説明
set	インタフェースにアドレスとネットマスクのどちらか、または両方を設定する
subnet	インタフェースのサブネットアドレスを設定する
xmit/ -xmit	インタフェースにおけるパケット伝送を使用可能または使用不能する

IPv6 設定手順については、33 ページの「IPv6 ノードを有効にする」を参照してください。

例 – 新しい ifconfig ユーティリティオプション

次に示す ifconfig コマンドは、まず hme0:3 論理インタフェースを 1234::5678/64 IPv6 アドレスに作成します。次に up オプションでインタフェースを使用可能にし、状態を報告し、インタフェースを使用不可にします。最後に、インタフェースを削除します。

例 **3–1** 例 – addif と removeif の使用

```
# ifconfig hme0 inet6 addif 1234::5678/64 up
Created new logical interface hme0:3

# ifconfig hme0:3 inet6
hme0:3: flags=2000841<UP,RUNNING,MULTICAST,IPv6> mtu 1500 index 2
        inet6 1234::5678/64

# ifconfig hme0:3 inet6 down

# ifconfig hme0 inet6 removeif 1234::5678
```

次に示す ifconfig コマンドは、まず物理インタフェース名に関連付けられたデバイスを開きます。次に TCP/IP がデバイスを使用するために必要なストリームを構成し、デバイスの状態を報告し、トンネルのソースアドレスと宛先アドレスを構成します。最後に、構成後のデバイスの最新状態を報告します。

例 **3–2** 例 – tsrc/tdst と index

```
# ifconfig ip.tun0 inet6 plumb index 13

# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,
IPv6> mtu 1480 index 13
        inet tunnel src 0.0.0.0
        inet6 fe80::/10 --> ::

# ifconfig ip.tun0 inet6 tsrc 120.46.86.158 tdst 120.46.86.122
```

例 **3–2** 例 – tsrc/tdst と index　(続き)

```
# ifconfig ip.tun0 inet6
ip.tun0: flags=2200850<POINTOPOINT,RUNNING,MULTICAST,NONUD,
IPv6> mtu 1480 index 13
        inet tunnel src 120.46.86.158  tunnel dst 120.46.86.122
        inet6 fe80::8192:569e/10 --> fe80::8192:567a
```

複数のネットワークインタフェースがあるノード

ノードに複数のネットワークインタフェースがある場合、追加インタフェース用に /etc/hostname.*interface* ファイルを作成する必要があります。

IPv4 の動作

たとえば、マシン timbuktu について考えてみます。このシステムは、『*Solaris* のシステム管理 *(IP*サービス*)*』の「サンプルネットワーク内のホスト」に示されています。このシステムには、2 つのネットワークインタフェースがあり、ルーターとして機能します。プライマリネットワークインタフェース le0 は、ネットワーク 192.9.200 に接続されています。システムの IP アドレスは 192.9.200.70、ホスト名は timbuktu です。Solaris インストールプログラムによって、一次ネットワークインタフェースにファイル /etc/hostname.le0 が作成され、ホスト名 timbuktu がファイルに入力されます。

2 番めのネットワークインタフェースは le1 で、192.9.201 に接続されています。このインタフェースは timbuktu に物理的にインストールされていますが、独自の IP アドレスが必要です。そのため、このインタフェースに対して /etc/hostname.le1 ファイルを手動で作成する必要があります。このファイルのエントリはルーター名 timbuktu-201 です。

IPv6 の動作

IPv6 を設定する場合、/etc/hostname6.le0 と /etc/hostname6.le1 のインタフェースだけが必要です。各インタフェースアドレスは、システムの起動時に自動的に設定されます。

IPv6 デーモン

ここでは、次の IPv6 デーモンについて説明します。

- in.ndpd – IPv6 自動設定用のデーモン
- in.ripngd – IPv6 のネットワークルーティングデーモン
- inetd – インターネットサービスデーモン

in.ndpd デーモン

このデーモンはルーター発見を実装します。 デーモンは、IPv6 の自動アドレス設定も実装します。 表 3–4 は、サポートされているオプションを示します。

表 **3–2** in.ndpd デーモンのオプション

オプション	説明
-d	すべてのイベントのデバッグをオンにする
-D	特定のイベントのデバッグをオンにする
-f	設定を読み出す元のファイル (デフォルトファイルのかわり)
-I	インタフェースごとに関連情報を印刷する
-n	ルーター通知をループバックしない
-r	受信パケットを無視する
-v	冗長モード (さまざまな種類の診断メッセージを報告する)
-t	パケット追跡をオンにする

パラメータは、in.ndpd の動作を制御します。これらのパラメータは /etc/inet/ndpd.conf 構成ファイルと /var/inet/ndpd_state.*interface* 起動ファイル (存在する場合) に設定されます。

/etc/inet/ndpd.conf が存在すると構文解析され、ノードをルーターとして使用するための設定が行われます。表 3–3 に、このファイルに出現する可能性がある各種キーワードをまとめます。ホストを起動しても、ルーターがすぐには使用できない場合があります。ルーターによって通知されたパケットがドロップしたり、また、通知されたパケットがホストに届かない場合もあります。
/var/inet/ndpd_state.*interface* ファイルは状態ファイルです。このファイルはノードごとに定期的に更新されます。ノードに障害が発生し再起動した場合、ルーターがなくてもノードはインタフェースを設定できます。このファイルにはインタフェースアドレス、更新時間、有効期間などの情報が保存されています。また、先のルーター通知で得られた情報も保存されています。

注 – 状態ファイルの内容を変更する必要はありません。このファイルは、in.ndpd デーモンが自動的に管理します。

表 **3–3** /etc/inet/ndpd.conf キーワード

キーワード	説明
ifdefault	すべてのインタフェースのルーターの動作を指定する。次の構文を使用してルーターパラメータと対応する値を設定する ifdefault [*variable value*]
prefixdefault	プレフィックス通知のデフォルトの動作を指定する。次の構文を使用してルーターパラメータと対応する値を設定する prefixdefault [*variable value*]
if	インタフェース別パラメータを設定する。構文は次のとおり if *interface* [*variable value*]
prefix	インタフェース別プレフィックス情報を通知する。構文は次のとおり prefix *prefix /length interface* [*variable value*]

注 – ifdefault と prefixdefault エントリは、構成ファイルの if エントリと prefix エントリの前に置く必要があります。

設定変数と設定できる値については、in.ndpd(1M) と ndpd.conf(4) のマニュアルページを参照してください。

例 – /etc/inet/ndpd.conf ファイル

次の例は、コメント行のテンプレートと、キーワードと設定変数の使用方法を示します。

```
# ifdefault       [variable value]*
# prefixdefault [variable value]*
# if ifname     [variable value]*
# prefix prefix/length ifname
#
#  Per interface configuration variables
#
#DupAddrDetectTransmits
#AdvSendAdvertisements
#MaxRtrAdvInterval
#MinRtrAdvInterval
```

```
#AdvManagedFlag
#AdvOtherConfigFlag
#AdvLinkMTU
#AdvReachableTime
#AdvRetransTimer
#AdvCurHopLimit
#AdvDefaultLifetime
#
# Per Prefix:  AdvPrefixList configuration variables
#
#
#AdvValidLifetime
#AdvOnLinkFlag
#AdvPreferredLifetime
#AdvAutonomousFlag
#AdvValidExpiration
#AdvPreferredExpiration

ifdefault AdvReachableTime 30000 AdvRetransTimer 2000
prefixdefault AdvValidLifetime 240m AdvPreferredLifetime 120m

if qe0 AdvSendAdvertisements 1
prefix 2:0:0:56::/64 qe0
prefix fec0:0:0:56::/64 qe0

if qe1 AdvSendAdvertisements 1
prefix 2:0:0:55::/64 qe1
prefix fec0:0:0:56::/64 qe1

if qe2 AdvSendAdvertisements 1
prefix 2:0:0:54::/64 qe2
prefix fec0:0:0:54::/64 qe2
```

in.ripngd デーモン

in.ripngd デーモンは、IPv6 ルーターの RIP 次世代ルーティングプロトコルを実装します。このプロトコルは、IPv6 用の RIP に相当する内容を定義します。RIP は、広く使用されている IPv4 ルーティングプロトコルで、Bellman-Ford 距離ベクトルアルゴリズムに基づいています。表 3–4 は、サポートされているオプションを示します。

表 3–4 in.ripngd デーモンのオプション

オプション	説明
-p *n*	*n* は RIPNG パケットの送受信に使用する代替ポート番号を指定する
-q	ルーティング情報を打ち切る
-s	デーモンがルーターとして動作している場合でもルーティング情報の提供を強制する

表 **3–4** in.ripngd デーモンのオプション　(続き)

-P	ポイズンリバースを打ち切る
-S	in.ripngd がルーターとして機能しない場合、各ルーターにはデフォルトのルートだけが指定される

inetd インターネットサービスデーモン

IPv6 有効化サーバーは、IPv4 アドレスか IPv6 アドレスを処理できるサーバーです。IPv6 有効化サーバーは、対応するクライアントで使用しているプロトコルと同じプロトコルを使用します。/etc/inet/inetd.conf ファイルには、inetd(1M) がソケット経由でインターネット要求を受信したときに呼び出すサーバーリストが保存されています。ソケットベースのインターネットサーバーエントリはそれぞれ、次の構文を使用する 1 行です。

service_name socket_type proto flags user server_pathname args

各フィールドに指定できる値については、inetd.conf(4) のマニュアルページを参照してください。Solaris オペレーティング環境の場合、IPv6 有効化としてサービスを /etc/inet/inetd.conf ファイルに指定するには、*proto* フィールドに tcp6 または udp6 を指定します。サービスが IPv4 専用の場合、*proto* フィールドは tcp または udp として指定します。サービスの *proto* 値に tcp6 または udp6 を指定すると、inetd は所定のデーモンに AF_INET6 ソケットを渡します。

inetd.conf ファイルの次のエントリは、IPv4 クライアントアプリケーションと IPv6 クライアントアプリケーションの両方と通信できる udp サーバー (myserver) を表します。

例 **3–3** IPv4 クライアントアプリケーションと IPv6 クライアントアプリケーションの両方と通信するサーバー

```
myserver   dgram   udp6    wait    root    /usr/sbin/myserver    myserver
```

IPv6 有効化サーバーは、AF_INET (IPv4 専用) ソケットまたは AF_INET6 (IPv6 と IPv4) ソケットを inetd から継承できます。サービスに対して *proto* 値は、tcp6、udp6、tcp または udp として指定されています。この種のサーバーでは、2 つの inetd.conf エントリを指定できます。1 つは *proto* を tcp として、もう1 つは *proto* を tcp6 として指定できます。

注 – AF_INET6 ソケットは、IPv4 プロトコルと IPv6 プロトコルのどちらでも使用できるため、*proto* 値 tcp6 (udp6) を指定すれば充分です。

各種 IPv6 有効化サーバーの記述方法については、『プログラミングインタフェース』を参照してください。

Solaris ソフトウェアとともに提供されるサーバーはすべて、*proto* 値を tcp6 または udp6 と指定する inetd エントリが1 つあれば十分です。ただし、リモートシェルサーバー (shell) とリモート実行サーバー (exec) のエントリには、tcp と tcp6 の両方の *proto* 値を指定する必要があります。例 3–4 は、rlogin、telnet、shell、exec 用の inetd エントリです。

例 **3–4** Solaris ソフトウェアで提供されるサーバー用の inetd.conf エントリ

```
login stream    tcp6 nowait root  /usr/sbin/in.rlogind  in.rlogind
telnet stream    tcp6 nowait root /usr/sbin/in.telnetd in.telnetd
shell    stream    tcp    nowait    root    /usr/sbin/in.rshd    in.rshd
shell    stream    tcp6    nowait    root    /usr/sbin/in.rshd    in.rshd
exec    stream    tcp    nowait    root    /usr/sbin/in.rexecd    in.rexecd
exec    stream    tcp6    nowait    root    /usr/sbin/in.rexecd    in.rexecd
```

TCP ラッパーは、telnet などさまざまなネットワークサービスで入力要求を監視、フィルタ処理するためのパブリックドメインユーティリティです。以上のユーティリティの *server_pathname* として TCP ラッパーを指定する場合、*TCP* ラッパーが IPv6 対応である必要があります。対応していない場合、TCP ラッパーで使用するサービスの *proto* を tcp か udp に指定する必要があります。

また、Solaris ユーティリティを別の実装と入れ替える場合、そのサービスの実装が IPv6 をサポートしていることを確認する必要があります。サポートしていない場合、その実装の *proto* を tcp か udp に指定する必要があります。

注 – *proto* 値を tcp か udp のどちらか一方に指定すると、サービスでは IPv4 だけが使用されます。IPv4 接続と IPv6 接続の両方を有効にするには、proto 値を tcp6 か udp6 に指定する必要があります。サービスで IPv6 をサポートしていない場合、tcp6 や udp6 は指定しないでください。

ソケットを使用する IPv6 有効化サーバーについては、『プログラミングインタフェース』のソケット API への IPv6 拡張機能についての説明を参照してください。

既存のユーティリティに対する IPv6 拡張機能

ユーザーレベルインタフェースでは、次のユーティリティの組み込み拡張機能も変更されました。

- netstat(1M)
- snoop(1M)
- route(1M)

- ping(1M)
- traceroute(1M)

ifconfig(1M) ユーティリティも変更されました。詳細については、57 ページの「ifconfig ユーティリティに対する IPv6 拡張機能」を参照してください。

netstat(1M)

IPv4 ネットワーク状態の表示の他、netstat では IPv6 ネットワーク状態も表示できます。/etc/default/inet_type ファイルと -f コマンド行オプションで DEFAULT_IP 値を設定して、表示するプロトコル情報を選択できます。DEFAULT_IP のパラメータ設定では、netstat に IPv4 情報だけが表示されていることを確認できます。この設定は、-f オプションで無効にできます。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。

新しい -p オプションでは、net-to-media テーブルが表示されます。これは、IPv4 用の ARP テーブルであり、IPv6 用の近傍キャッシュです。詳細については、netstat(1M) のマニュアルページを参照してください。このコマンドの使用方法については、40 ページの「ネットワーク状態の表示方法」を参照してください。

snoop(1M)

snoop コマンドは、IPv4 パケットと IPv6 パケットの両方を取り込んで、IPv6 ヘッダー、IPv6 拡張ヘッダー、ICMPv6 ヘッダー、近傍検索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 プロトコルキーワードか ip6 プロトコルキーワードを指定すると、snoop コマンドは IPv4 パケットか IPv6 パケットのどちらかだけを表示します。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。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。このコマンドの使用方法については、44 ページの「IPv6 ネットワークトラフィックの監視方法」を参照してください。

route(1M)

このユーティリティは、IPv4 ルーターと IPv6 ルーターの両方で実行できます。デフォルトで、route は IPv4 ルートで実行します。コマンド行で route コマンドの直後にオプション -inet6 を指定すると、操作が IPv6 ルートで実行されます。詳細については、route(1M) のマニュアルページを参照してください。

ping(1M)

ping コマンドは、IPv4 プロトコルと IPv6 プロトコルの両方で、宛先ホストを調べることができます。プロトコル選択は、指定の宛先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が、宛先ホストの IPv6 アドレスを

戻すと、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、IPv4 プロトコルを使用します。-A コマンド行オプションで使用するプロトコルを指定すれば、この動作を無効にできます。

その他、-a コマンド行オプションを指定すれば、マルチホーム宛先ホストのアドレスをすべて ping できます。詳細については、ping(1M) のマニュアルページを参照してください。このコマンドの使用方法については、45 ページの「すべてのマルチホームホストアドレスの探査方法」を参照してください。

traceroute(1M)

traceroute コマンドを使用して、指定ホストまでの IPv4 ルートと IPv6 ルートの両方をトレースできます。使用するプロトコルの選択について、traceroute では、ping と同じアルゴリズムを使用します。選択を無効にするには、-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すべてのアドレスまでの各ルートは -a コマンド行オプションでトレースできます。traceroute(1M) のマニュアルページを参照してください。

表示出力の制御

netstat コマンドと ifconfig コマンドによる出力表示の方法を制御できます。

- コマンド行に追加したキーワードで、inet アドレスまたは inet6 アドレスを指定する
- /etc/default/inet_type ファイルの設定変数 DEFAULT_IP を設定する

DEFAULT_IP の値は、IP_VERSION4、IP_VERSION6、BOTH のどれかに設定できます。DEFAULT_IP を指定してこのファイルを作成しない場合、netstat と ifconfig では、両方のバージョンが表示されます。

注 – inet キーワードオプションと inet6 キーワードオプションは、netstat コマンドと ifconfig コマンドの使用時に inet_type ファイルで設定した値を無効にします。

操作については、43 ページの「IPv6 関連コマンドの出力表示の制御方法」を参照してください。

IPv6 の Solaris トンネルインタフェース

トンネルインタフェースのフォーマットは次のとおりです。

`ip.tun` *ppa*

ppa は物理的な接続ポイントです。

システム起動時に、トンネルモジュール (tun) は、(`ifconfig` によって) IP の最上位にプッシュされ、仮想インタフェースが作成されます。このプッシュは、`hostname6.*` ファイルを作成することによって行われます。

たとえば、IPv4 ネットワーク経由で IPv6 パケットをカプセル化するためのトンネルを作成するには、次のファイルを作成します。

`/etc/hostname6.ip.tun0`

このファイルの内容は、インタフェースが結合された後に `ifconfig`(1M) に渡されます。ポイントツーポイントトンネルの設定に必要なパラメータになります。

次のリストは、`hostname6.ip.tun0` ファイルのエントリの例です。

例 **3–5** `hostname6.`*interface* エントリ

```
tsrc 120.68.100.23 tdst 120.68.7.19 up
addif 1234:1234::1 5678:5678::2 up
```

この例では、IPv4 ソースと宛先アドレスは、IPv6 リンクローカルアドレスを自動設定するためのトークンとして使用されます。これらのアドレスは、 `ip.tun0` インタフェースのソースと宛先アドレスです。次の 2 つのインタフェース `ip.tun0` インタフェースと、論理インタフェース `ip.tun0:1` が設定されます。 論理インタフェースには、`addif` コマンドによって指定されたソースと宛先 IPv6 アドレスがあります。

すでに述べたとおり、システムをマルチユーザーとして起動すると、これらの構成ファイルの内容が変更されずに `ifconfig` に渡されます。上の例は次の内容と同じです。

```
# ifconfig ip.tun0 inet6 plumb
# ifconfig ip.tun0 inet6 tsrc 120.68.100.23 tdst 120.68.7.19 up
# ifconfig ip.tun0 inet6 addif 1234:1234::1 5678:5678::2 up
```

このトンネルにおける `ifconfig -a` の出力は次のとおりです。

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,
NONUD,IPv6> mtu 1480 index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::c0a8:6417/10 --> fe80::c0a8:713
ip.tun0:1: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,NONUD,
IPv6> mtu 1480 index 5
        inet6 1234:1234::1/128 --> 5678:5678::2
```

次の構文で構成ファイルに行を追加すれば、さらに論理インタフェースを設定できます。

```
addif IPv6-source IPv6-destination up
```

注 – トンネルのどちらかの端がトンネル経由で 1 つ以上のプレフィックスを通知する IPv6 ルーターである場合、トンネル構成ファイルには `addif` コマンドは必要ありません。 他のアドレスは自動設定されるため、必要とされる可能性があるのは `tsrc` と `tdst` だけです。

場合によっては、特定のトンネルについて、固有のソースリンクローカルアドレスと宛先リンクローカルアドレスを手動で設定する必要があることもあります。その場合、構成ファイルの最初の行を変更して、これらのリンクローカルアドレスを組み込みます。次に例を示します。

```
tsrc 120.68.100.23 tdst 120.68.7.19 fe80::1/10 fe80::2 up
```

ソースリンクローカルアドレスには、長さが 10 のプレフィックスがあります。この例では、`ip.tun0` インタフェースは次のようになります。

```
ip.tun0: flags=2200850<UP,POINTOPOINT,RUNNING,MULTICAST,
NONUD,IPv6> mtu 1480 index 6
        inet tunnel src 120.68.100.23  tunnel dst 120.68.7.19
        inet6 fe80::1/10 --> fe80::2
```

IPv6 ネットワーク (IPv6 経由の IPv6) 経由で IPv6 パケットをカプセル化するためのトンネルを作成するには、次のファイルを作成します。

```
/etc/hostname6.ip6.tun0
```

次のリストは、`hostname6.ip6.tun0` ファイルのエントリの例です。

例 **3–6** `hostname6.`*interface* エントリ

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3 up
```

IPv6 ネットワーク (IPv6 経由の IPv4) 経由で IPv4 パケットをカプセル化するためのトンネルを作成するには、次のファイルを作成します。

```
/etc/hostname.ip6.tun0
```

次のリストは、`hostname.ip6.tun0` ファイルのエントリの一例です。

例 **3–7** `hostname.`*interface* エントリ

```
tsrc 2000::114:a00:20ff:fe72:668c tdst 2000::103:a00:20ff:fe9b:a1c3
10.0.0.4 10.0.0.61 up
```

IPv4 ネットワーク (IPv4 経由の IPv4) 経由で IPv4 パケットをカプセル化するためのトンネルを作成するには、次のファイルを作成します。

/etc/hostname.ip.tun0

次のリストは、hostname.ip.tun0 ファイルのエントリの一例です。

例 **3–8** hostname.*interface* エントリ

```
tsrc 120.46.86.158 tdst 120.46.86.122
10.0.0.4 10.0.0.61 up
```

tun の固有の情報については、tun(7M) のマニュアルページを参照してください。IPv6 への移行時のトンネルの概念の一般的な説明については、78 ページの「トンネル機構」を参照してください。トンネルの設定方法については、46 ページの「IPv4 トンネルによる IPv6 の設定方法」を参照してください。

Solaris ネームサービスに対する IPv6 拡張機能

ここでは、Solaris 8 リリースで IPv6 の実装により導入されたネーミングの変更について説明します。 IPv6 アドレスは Solaris ネームサービス (NIS、NIS+、DNS およびファイル) のどれでも保存できます。また、IPv6 RPC トランスポートで NIS と NIS+ を使用して NIS データまたは NIS+ データを検出することもできます。

/etc/inet/ipnodes ファイル

/etc/inet/ipnodes ファイルには、IPv4 と IPv6 のアドレスが格納されています。このファイルはローカルデータベースとして、ホスト名を IPv4 アドレスや IPv6 アドレスに関連付けます。ホスト名やそのアドレスは、/etc/inet/ipnodes などの静的ファイルには保存しないでください。ただし、テスト目的の場合は IPv4 アドレスを /etc/inet/hosts に保存するのと同じ方法で IPv6 アドレスを保存します。ipnodes ファイルでは、hosts ファイルと同じフォーマット変換を使用します。ipnodes ファイルについては、ipnodes(4) のマニュアルページを参照してください。

IPv6-aware (IPv6 が利用可能な) ユーティリティでは、新しい /etc/inet/ipnodes データベースを使用します。既存の /etc/hosts データベースには、IPv4 アドレスだけを保存していますが、既存のアプリケーションの便宜上、このデータベースは変更されません。ipnodes データベースがない場合、IPv6-aware ユーティリティでは既存の hosts データベースを使用します。

注 – アドレスを追加する必要がある場合、IPv4 アドレスは hosts ファイルと ipnodes ファイルの両方に追加しなければなりません。IPv6 アドレスは ipnodes ファイルにだけ追加します。

例 – /etc/inet/ipnodes ファイル

```
#
# Internet IPv6 host table
# with both IPv4 and IPv6 addresses
#
::1     localhost
2::9255:a00:20ff:fe78:f37c   fripp.guitars.com fripp fripp-v6
fe80::a00:20ff:fe78:f37c     fripp-11.guitars.com frippll
120.46.85.87                    fripp.guitars.com fripp fripp-v4
2::9255:a00:20ff:fe87:9aba   strat.guitars.com strat strat-v6
fe80::a00:20ff:fe87:9aba     strat-11.guitars.com stratll
120.46.85.177                   strat.guitars.com strat strat-v4 loghost
```

注 – 上記の例のように、ホスト名アドレスは、ホスト名でグループにまとめる必要があります。

IPv6 の NIS 拡張機能

NIS 用に 2 つの新しいマップが追加されました。 ipnodes.byname と ipnodes.byaddr です。/etc/inet/ipnodes と同様に、これらのマップには、IPv4 情報と IPv6 情報の両方が保存されます。既存の hosts.byname と hosts.byaddr マップは、IPv4 情報だけを保存しています。既存のアプリケーションの便宜上変更されていません。

IPv6 の NIS+ 拡張機能

NIS+ 用に ipnodes.org_dir という新しいテーブルが追加されました。このテーブルには、ホスト用の IPv4 アドレスと IPv6 アドレスの両方が保存されています。既存の hosts.org_dir テーブルは IPv4 アドレス情報だけを保存しています。このテーブルは、既存のアプリケーションの便宜上変更されていません。

IPv6 の DNS 拡張機能

AAAA レコードとして定義された新しいリソースレコードが、RFC 1886 で定義されています。この AAAA レコードは、ホスト名を 128 ビット IPv6 アドレスにマップします。PTR レコードは IPv6 でも、IP アドレスをホスト名にマップするときに使用されています。128 ビットアドレスの 32 の 4 ビットニブルは、IPv6 アドレス用に反転されています。各ニブルは対応する 16 進 ASCII 値に変換されます。変換後、`ip6.int` が追加されます。

`nsswitch.conf` ファイルへの変更

`/etc/inet/ipnodes` で IPv6 アドレスを調べる機能に加え、IPv6 サポートは、NIS ネームサービス、NIS+ ネームサービス、DNS ネームサービスに追加されています。その結果、`nssiwtch.conf(4)` ファイルは IPv6 検索をサポートするように変更されました。ipnodes 行が `/etc/nsswitch.conf` ファイルに追加されました。この追加により、Solaris ネームサービス (NIS、NIS+、DNS、ファイル) の新しいデータベースで検索が可能になりました。次の太字で示された行は、ipnodes エントリの例です。

```
hosts:  files dns nisplus [NOTFOUND=return]
ipnodes: files dns nisplus [NOTFOUND=return]
```

注 – IPv4 アドレスと IPv6 アドレスでこれらの `ipnodes` データベースを生成してから、複数のネームサービスで `ipnodes` を探すように `/etc/nsswitch.conf` ファイルを変更してください。ホストアドレスの解決時に不要な遅延が発生してしまうからです (起動タイミングの遅れが発生することもあります)。

図 3–1 は、`gethostbyname()` コマンドと `getipnodebyname()` コマンドを使用するアプリケーションにおける、`nsswitch.conf` ファイルと新しいネームサービスデータベースの新しい関係を示します。斜体の項目は新規です。`gethostbyname()` コマンドは、`/etc/inet/hosts` に保存されている IPv4 アドレスだけを調べます。`getipnodebyname()` コマンドは、`nsswitch.conf` ファイルの `ipnodes` エントリで指定したデータベースを調べます。検索に失敗すると、`nsswitch.conf` ファイルの `hosts` エントリで指定したデータベースを調べます。

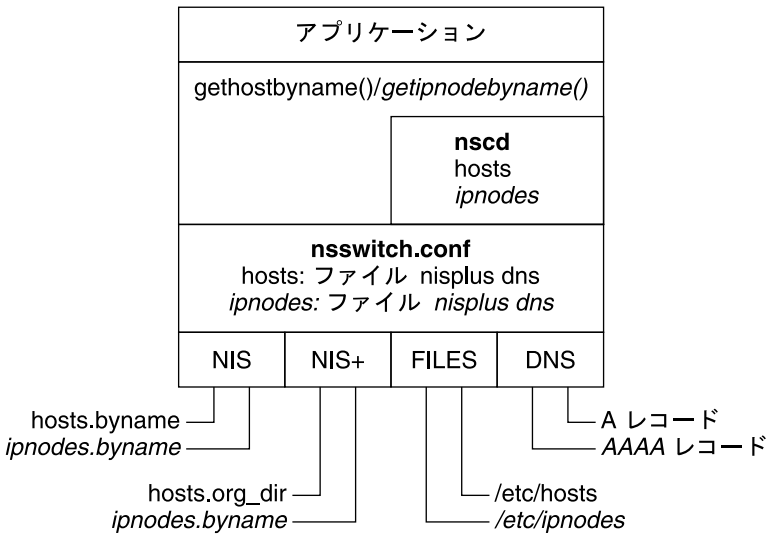

図 **3–1** `nsswitch.conf` とネームサービスの関係

ネームサービスの詳細は、『*Solaris* のシステム管理 *(*ネーミングとディレクトリサービス *: DNS*、*NIS*、*LDAP* 編*)*』を参照してください。

ネームサービスコマンドの変更

IPv6 をサポートできるように、既存のネームサービスコマンドで IPv6 アドレスを調べることができます。たとえば、`ypmatch` コマンドは、新しい NIS マップに使用できます。`nismatch` コマンドは、新しい NIS+ テーブルに使用できます。`nslookup` コマンドでは、DNS の新しい AAAA レコードを調べることができます。ネームサービスの変更については、70 ページの「IPv6 の NIS 拡張機能」、70 ページの「IPv6 の NIS+ 拡張機能」、および 71 ページの「IPv6 の DNS 拡張機能」を参照してください。

これらのコマンドの使用手順については、50 ページの「IPv6 ネームサービス情報の表示」を参照してください。

NFS と RPC による IPv6 のサポート

NFS と RPC ソフトウェアは、シームレスに IPv6 をサポートします。NFS サービスに関連のある既存のコマンドは変更されていません。ほとんどの RPC アプリケーションが、変更なしで IPv6 で実行できます。トランスポート機能のある一部の高度 RPC アプリケーションに更新が必要な場合があります。

ATM 経由の IPV6 サポート

Solaris オペレーティング環境では、IPv6 経由の ATM、固定仮想回路 (PVC)、静的な交換仮想回路 (SVC) をサポートするようになりました。

第 4 章

IPv4 から IPv6 への移行 (リファレンス)

ホストとルーターを IPv6 にアップグレードする場合、これらのホストおよびルーターを、IPv4 ホストおよび IPv4 ルーターで相互運送できるようにする必要があります。 この章では、IPv4 から IPv6 への移行と、標準的な解決法の概要について説明します。RFC 1933 でも、移行問題の詳しい解決法を示しています。

この章では、以下の内容について説明します。

- 75 ページの「移行条件」
- 76 ページの「標準移行ツール」
- 80 ページの「IPv4 と IPv6 の相互運用性」
- 81 ページの「サイト移行のシナリオ」
- 82 ページの「その他の移行機構」

移行条件

移行時のグローバルな調整は不要です。サイトとインターネットサービスプロバイダ (ISP) はそれぞれのスケジュールで移行できます。また、移行時の依存条件も最小限に抑えました。たとえば、ホストのアップグレード前にルーターを IPv6 にアップグレードしなくても移行できます。

サイトが異なれば、移行時にはそれぞれの制約が課されます。また、IPv6 の初期アダプタには、IPv6 の製品版ユーザーの場合とは異なる問題があります。RFC 1933 は現在利用できる移行ツールを定義しています。移行の必然性としては、IPv4 アドレス領域の不足または IPv6 の新機能を使用する必要性のどちらか、または両方が考えられます。IPv6 仕様では、移行時には、既存のプロトコルの完全な互換性が求められます。 また既存のアプリケーションとの互換性も求められます。

移行方式を理解できるように、次の用語を定義します。

- IPv4 専用ノード – IPv4 だけを実装したホストやルーター。IPv4 専用ノードでは IPv6 は認識できない。移行以前に既存の IPv4 ホストとルーターのインストール可能ベースは IPv4 専用ノード
- IPv6/IPv4 ノード – IPv4 と IPv6 の両方を実装するホストとルーター。デュアルスタックとも呼ぶ
- IPv6 専用ノード – IPv6 を実装するホストまたはルーター。IPv4 を実装しない
- IPv6 ノード – IPv6 を実装するホストまたはルーター。IPv6/IPv4 ノードと IPv6 専用ノードは、どちらも IPv6 ノード
- IPv4 ノード – IPv4 を実装するホストまたはルーター。IPv6/IPv4 ノードと IPv4 専用ノードは、どちらも IPv4 ノード
- サイト – インターネットのプライベートトポロジの 1 つ。すなわちあらゆるユーザーを対象としたトラフィック伝送を行わないトポロジ。サイトが物理的に広範囲に展開されることがある。たとえば、多国籍企業のプライベートネットワークは、1 つのサイト

標準移行ツール

RFC 1933 は、次の移行方式を定義しています。

- ホストとルーターを IPv6 にアップグレードするとき、それらの IPv4 の機能を残す。したがって、すべての IPv4 プロトコルおよびアプリケーションとの互換性が確保される。このようなホストおよびルーターをデュアルスタックと呼ぶ
- IPv6 対応ノードに関する情報は、DNS などのネームサービスを利用して伝送する
- IPv6 アドレス形式には、IPv4 アドレスを保存する
- IPv4 パケットで IPv6 パケットをトンネル処理して、IPv6 にアップグレードされていないルーターを通過できる

デュアルスタックの実装

デュアルスタックとは、アプリケーションからネットワーク層に至るプロトコルスタックのすべてのレベルの完全な複製をいいます。デュアルスタックの例として、同じマシンで実行する OSI プロトコルと TCP/IP プロトコルがあります。ただし、 IPv6 移行の観点からは、プロトコルスタックに IPv4 と IPv6 の両方を組み込むことを表します。残りスタックは同一となります。この場合、同じ伝送プロトコル (TCP、UDPなど) が IPv4 と IPv6 の両方で実行します。また、同じアプリケーションも IPv4 と IPv6 の両方で実行します。

次の図は、OSI 層全体にわたるデュアルスタックプロトコルを表します。

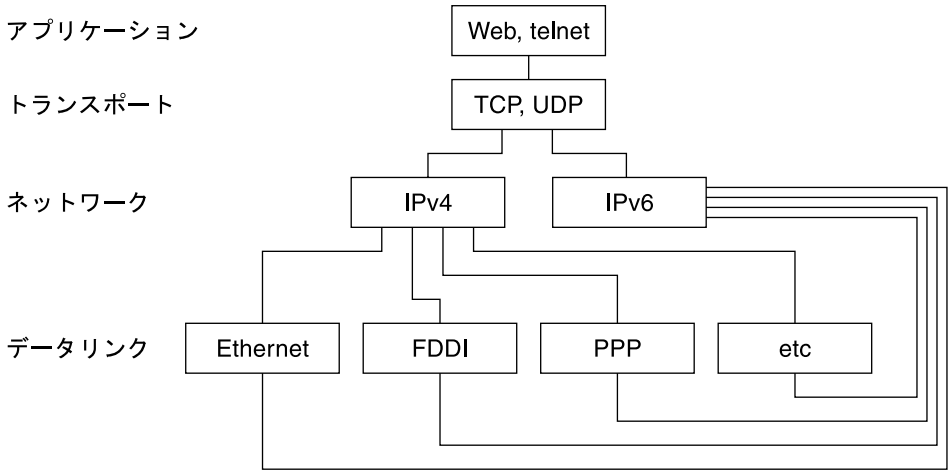

図 **4–1** デュアルスタックプロトコル

デュアルスタック方式では、ホストとルーター両方のサブセットをアップグレードして、IPv4 に加えて IPv6 をサポートします。この方法では、アップグレードされた後のノードからも IPv4 で常に IPv4 専用ノードと相互運用できます。

ネームサービスの設定

デュアルノードでは、ピアが IPv6 と IPv4 のどちらをサポートしているか明確でないと、伝送時にどちらの IP バージョンを使用するのかが決まりません。そこで、ネームサービスでどんな情報を伝達するかを制御すると、デュアルノードで使用する IP バージョンを決定できます。さらに、ネームサービスで IPv4 ノードの IP アドレスと IPv6 ノードの IP アドレスを定義します。それによって、デュアルノードでは、両方のアドレスをネームサービスで使用できます。

IPv6 アドレスをネームサービスに指定した場合も、IPv6 でノードにアクセスできます。ただし、ノードにアクセスできるのは、ネームサービスから情報を得たノードだけです。たとえば、NIS に IPv6 アドレスを指定すると、その IPv6 ホストは IPv6 からアクセスできます。ただし、IPv6 ホストにアクセスできるのは、NIS ドメインに所属する IPv6 とデュアルノードだけです。 グローバル DNS に IPv6 アドレスを指定するには、そのノードがインターネット IPv6 バックボーンからアクセスできることが条件です。これは、IPv4 の場合も同様です。たとえば、メール配信の操作は、IPv4 でアクセスできるノードの IPv4 アドレスがあるかどうかに依存します。 これは、HTTP プロキシの操作の場合も同様です。 たとえば、ファイアウォールなどの理由で IPv4 でアクセスできない場合、ネームサービスは内部ファイアウォールと 外部ファイアウォールのデータベースに分けます。 これにより、IPv4 アドレスがアクセスできる範囲だけで認識できるようになります。

ネームサービスのアクセスに使用するプロトコルは、ネームサービスで検索できるアドレスタイプに依存しません。このネームサービスサポートとデュアルスタックでは、デュアルノードが IPv4 専用ノードと通信するときに、IPv4 を使用できます。 さらに、 IPv6 ノードと通信するときに、このネームサービスでは IPv6 を使用することができます。 ただし、宛先が IPv6 経路で到達できる必要があります。

IPv4 互換アドレスフォーマットの使用

通常 32 ビット IPv4 アドレスは、128 ビット IPv6 アドレスで表現できます。移行機能では、次の 2 つの形式を定義しています。

- IPv4 互換アドレス

000 ... 000	IPv4 アドレス

- IPv4 マップアドレス

000 ... 000	0xffff	IPv4 アドレス

IPv6 ノードは互換フォーマットで表現します。このフォーマットでは、実際の IPv6 アドレスがなくても IPv6 ノードを使用できます。また、IPv4 専用ルーターで自動トンネルを使用できるため、このアドレスフォーマットではさまざまな IPv6 設定の試用が可能です。ただし、IPv6 ステートレスアドレス自動設定機構では、このアドレスは設定できません。IPv6 ステートレスアドレス自動設定機構には、DHCPv4 など既存の IPv4 機構や静的構成ファイルが必要なためです。

マップアドレスフォーマットでは、IPv4 ノードを表現します。現在ソケット API の一部でだけ、このアドレスフォーマットの使用方法が定義されています。アプリケーションでは、IPv6 アドレスと IPv4 アドレスの両方に共通のアドレスフォーマットを使用できます。 共通のアドレスフォーマットは、IPv4 アド レスを 128 ビットマップアドレスで表現します。ただし、IPv4 プロトコルトランスレータと IPv6 プロトコルトランスレータがないと、これらのアドレスは使用できません。

トンネル機構

移行時の依存状態を最小限に抑える目的から、2 つの IPv6 ノード間にあるすべてのルーターで IPv6 をサポートする必要がありません。この機構をトンネルといいます。基本的に IPv6 パケットは IPv4 パケット内部に組み込まれ、IPv4 ルーター間を転送されます。以下の図は、IPv4 ルーター (R) 間のトンネル機構を示します。

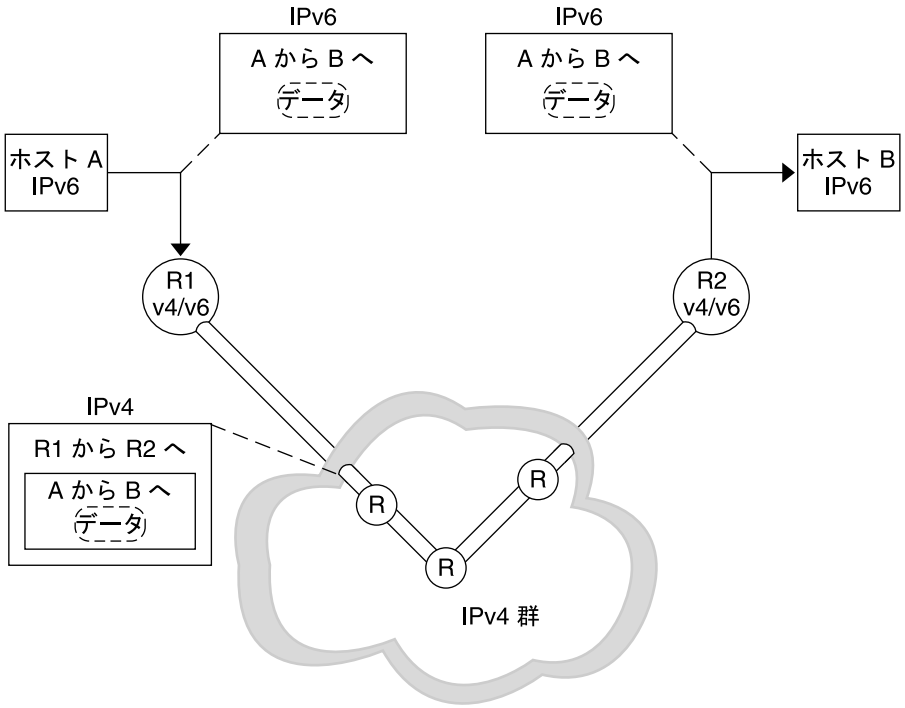

図 **4–2** トンネル機構

その他、移行時には次のようなトンネル機構の使用方法があります。

- 2 つのルーター間で設定したトンネル (上記の図を参照)
- デュアルホストで終了する自動トンネル

設定トンネルは、MBONE (IPv4 マルチキャストバックボーン) など現在はインターネットで他の目的に使用します。設定トンネルの作成手順からいうと、2 つのルーターを設定して、その間に IPv4 ネットワーク経由の仮想ポイントツーポイントリンクを作成します。 近い将来インターネットのさまざまな局面にこの種のトンネルが利用されるでしょう。

自動トンネル

初期の実験的配置では、自動トンネルの使用可能範囲は限定されています。自動トンネルは、IPv4 – 互換アドレスを要求します。 自動トンネルは、IPv6 ルーターが使用できない場合に IPv6 ノードを接続するために使用できます。自動トンネルネットワークインタフェースを設定すれば、トンネルの発信元はデュアルホストとデュアルルーターのどちらが発信元でも使用できます。終点は必ずデュアルホストになります。トンネルのはたらきにより、宛先 IPv4 アドレス (トンネルの終点) が IPv4 互換宛先アドレスから抽出されて動的に指定されます。

アプリケーションとの対話

IPv6 にアップグレードしたノードでも、IPv6 を使用できるかどうかはアプリケーション次第です。アプリケーションで、IPv6 アドレスのネームサービスを要求するネットワーキング API を使用しない場合があります。また、アプリケーション側で変更が必要な API (ソケットなど) を使用する場合もあります。さらに、API のプロバイダ (`java.net` クラスなどの実装) が IPv6 アドレスをサポートしていない場合もあります。 どの場合も、ノードが送受信するのは IPv4 ノードのように IPv4 パケットだけです。

次の用語は、インターネットの世界では標準用語として使用されています。

- IPv6-unaware (非認識) – IPv6 アドレスを処理できないアプリケーション。IPv4 アドレスのないノードとは通信できない
- IPv6-aware (認識) – IPv4 アドレスがないノードとも通信できるアプリケーション。長い IPv6 アドレスも処理できる。アプリケーションに透過な場合がある。たとえば実際のアドレスの内容や形式が API によって非表示になる場合など
- IPv6-enabled (有効化) – IPv6-aware であるだけでなく、フローラベルなど IPv6 固有の機能が利用できる。有効化アプリケーションは低下モードで IPv4 も処理できる
- **IPv6–required (要求) —** このアプリケーションは、IPv6– 特定機能をいくつか要求します。このアプリケーションは IPv4 では作動できません。

IPv4 と IPv6 の相互運用性

IPv4 から IPv6 に段階的に移行する場合、新しく導入する IPv6 有効化アプリケーションと共に既存の IPv4 アプリケーションも使用しなければなりません。最初の段階で、ベンダーは、デュアルスタックを実行しているホストとルーターのプラットホームを提供します。 デュアルスタックとは、IPv4 プロトコルスタックと IPv6 プロトコルスタックのことです。 IPv4 アプリケーションは、少なくとも 1 つの IPv6 インタフェースで IPv6 有効化になっているデュアルスタックでも実行できます。アプリケーションの変更や移植は不要です。

デュアルスタックで実行する IPv6 アプリケーションも、IPv4 プロトコルを使用できます。IPv6アプリケーションは、IPv4 マップIPv6 アドレスを使用します。 IPv6 は設計上、IPv4 と IPv6 で別々のアプリケーションは不要です。たとえば、デュアルホストの IPv4 クライアントがなくても IPv4 専用ホストのサーバーと「通信」できます。また独立した IPv6 クライアントがなくても IPv6 サーバーと通信できます。 IPv4 クライアントアプリケーションを新しい IPv6 API に移植するだけです。クライアントは、IPv4 専用サーバーと通信できます。 また、デュアルホストまたは IPv6 専用ホストで実行中の IPv6 サーバーとも通信できます。

ネームサーバーからクライアントが取り出すアドレスで、IPv6 や IPv4 を使用するかどうかが決まります。たとえば、ネームサーバーにそのサーバーの IPv6 アドレスが指定されている場合、サーバーは IPv6 を処理できます。

表 4–1 に IPv4 と IPv6 のクライアントとサーバー間の相互運用性をまとめます。表 4–1 では、デュアルスタックホストに、IPv4 と IPv6 両方のアドレスがそれぞれのネームサービスデータベースに存在するものとします。

表 **4–1** クライアントサーバーアプリケーション: IPv4 と IPv6 の相互運用性

アプリケーションの種類 (ノードの種類)	**IPv6-unaware (非認識) サーバー (IPv4 専用ノード)**	**IPv6-unaware (非認識) サーバー (IPv6 有効化ノード)**	**IPv6-aware (認識) サーバー (IPv6 専用ノード)**	**IPv6-aware (認識) サーバー (IPv6 有効化ノード)**
IPv6-unaware (非認識) クライアント (IPv4 専用ノード)	IPv4	IPv4	X	IPv4
IPv6-unaware (非認識) クライアント (IPv6 有効化ノード)	IPv4	IPv4	X	IPv4
IPv6-aware (認識) クライアント (IPv6 専用ノード)	X	X	IPv6	IPv6
IPv6-aware (認識) クライアント (IPv6 有効化ノード)	IPv4	(IPv4)	IPv6	IPv6

X は、それぞれのサーバーとクライアント間の通信ができないことを表します。

(IPv4) は、クライアントの選択するアドレスによって相互運用性が決まることを表します。IPv6 アドレスを選択すると、クライアントの処理はエラーになります。 ただし、IPv4 アドレスを選択すると、IPv4 マップ IPv6 アドレスとしてクライアントに戻り、IPv4 データグラム が送信されて処理が成功します。

IPv6 配置の初期段階では、IPv6 のほとんどの実装がデュアルスタックノードで処理されます。一般ベンダーではほとんど、初期状態では IPv6 専用実装をリリースしません。

サイト移行のシナリオ

サイトや ISP では、それぞれ事情が異なり、移行段階の手順が異なります。 ここでは、サイト移行のシナリオの例をいくつか紹介します。

IPv6 へのサイトの移行では、最初に IPv6 アドレスをサポートするためのネームサービスをアップグレードします。DNS の場合、BIND 4.9.4 以降などの新しい AAAA (クアドA) レコードをサポートする DNS サーバーにアップグレードします。2 つの新しい NIS マップと NIS+ テーブルを Solaris システムで作成、管理できます。新しいデータベースの詳細については、69 ページの「Solaris ネームサービスに対する IPv6 拡張機能」 を参照してください。

ネームサービスで IPv6 アドレスを処理できるようになったら、ホストの移行を開始します。ホストは、次の手順で移行します。

- ホストを 1 つずつアップグレードします。IPv4 互換アドレスと自動トンネルを使用します。ルーターのアップグレードは不要です。この方法は最初の試験的な移行に適しています。IPv6 の機能のすべてが利用できるわけではありません。したがって、ステートレスアドレス自動設定や IP マルチキャストは利用できません。このシナリオはアプリケーションが IPv6 で実行できるかどうかを確認するときに使用します。また、アプリケーションが IPv6 IP 層セキュリティを利用できるかどうかを確認するときも使用します。
- サブネットを 1 つずつアップグレードします。ルーター間に設定したトンネルを使用します。このシナリオでは、サブネットごとに少なくとも 1 つのルーターをデュアルにアップグレードします。 サイト内のデュアルルーターは設定したトンネルで結合します。 これで、サブネット上のホストでは、IPv6 の全機能を利用できます。このように段階的にアップグレードしていく中で徐々にアップグレードされるルーターが増加するとともに、設定済みのトンネルは削除できます。
- ホストをアップグレードする前にすべてのルーターをデュアルにアップグレードします。この方法は逐次行われるように思えますが、すべてのルーターがアップグレードされるまでは IPv6 の機能を利用できません。このシナリオでは、段階的な配置方式は制約されます。

その他の移行機構

先に説明した方法では、デュアルノードと IPv4 ノード間で相互運用をします。その場合、デュアルノードには IPv4 アドレスがあります。ただし、その方法では、IPv6 専用ノードと IPv4 専用ノードの間で相互運用しません。また、IPv4 アドレスのないデュアルノードと IPv4 専用ノード間でも相互運用しません。 ほとんどの実装ではデュアルにできます。ただし、デュアル実装には、IPv4 専用ノードとの相互運用が必要なすべてのノードごとに 1 つのアドレスを割り当てるのに充分な IPv4 アドレス領域が必要です。

次に、新しい移行機構がなくても相互運用を実現できる方法を示します。

- IPv6 専用ノードとインターネットの他の要素との間にアプリケーション層ゲートウェイ (ALG) を配置する。現在使用されている ALG としては、HTTP プロキ シとメールリレーがある
- IPv4 用の、NAT ボックス (ネットワークアドレストランスレータ) をすでに売り出している会社もある。 これは、内部のプライベート IP アドレス (ネットワーク 10 など。RFC 1918 参照) と、外部 の IP アドレスの間の変換を行う。 このような会社では、IPv6 から IPv4 アドレスへの変換もサポートするように、NAT ボックス をアップグレードする可能性が高い

残念ながら、ALG と NAT のどちらの方法も、弱点があります。これらの方法を使用すると、インターネットの基盤がかなり弱まります。 IETF では、IPv6 専用ノードと IPv4 専用ノードとのより良い相互運用性のために努力しています。1 つの提案としては、必要に応じて IPv4 互換アドレスを割り当てる方法でヘッダートランスレータを使用する方法があります。別の方法としては、必要に応じて IPv4 互換アドレスを割り当て、IPv6 トンネルで IPv4 を利用して IPv6 専用ルーターをブリッジできます。

ステートレスヘッダートランスレータでは、使用中の IPv6 アドレスを IPv4 アドレスとして表現できれば、IPv4 ヘッ ダーフォーマットと IPv6 ヘッダーフォーマットの間の変換が可能です。つまり、アドレスは、IPv4 互換アドレスである必要があります。もしくは、IPv4マップアドレスである必要があります。 これらのトランスレータのサポートは、IPv6 プロトコ ルに組み込まれています。 暗号化されているパケットを除いて、変換時に情報は失われません。ソースルーティングなどの使用頻度の低い機能は、情報が失われてしまうことがあります。

用語集

この用語集には、このマニュアルで新たに使用した、「*Global Glossary*」にはない用語の説明だけが記載されています。その他の用語の説明については、「*Sun Global Glossary*」を参照してください。

DES	Data Encryption Standard。 1975 年に開発され、1981 年に ANSI X.3.92 として ANSI で標準化された対称鍵の暗号化方式。DES では 56 ビットの鍵を使用する。
IPsec	IP データグラムを保護するためのセキュリティアーキテクチャ。
IPv4	インターネットプロトコルバージョン 4。IPv4 は、しばしば IP として参照される。 このバージョンは 32 ビットのアドレス領域を提供する。
IPv6	インターネットプロトコルバージョン 6。このバージョンは 128 ビットのアドレス領域を提供する。
IP 内 **IP** カプセル化	IP パケット内の IP パケットをトンネリングするための機構
IP リンク	リンク層でノード間の通信に使用される通信設備や通信媒体。リンク層は、IPv4/IPv6 のすぐ下にある。 例としては、Ethernet (単一の、またはブリッジされた) または ATM ネットワークがある。1 つまたは複数の IPv4 サブネット番号またはネットワーク接頭辞が IP リンクに割り当てられる。同じサブネット番号またはネットワーク接頭辞を複数の IP リンクに割り当てることはできない。ATM LANE では、IP リンクは 1 つのエミュレートされた LAN である。ARP を使用する場合、ARP プロトコルの有効範囲は単一の IP リンクである。
MD5	デジタル署名などのメッセージ認証に使用する繰り返し暗号化のハッシュ関数。1991 年に Rivest 氏によって開発された。
MTU	最大転送単位。 リンクに転送できるサイズ (オクテット単位)。たとえば、Ethernet の MTU は 1500 オクテット。
PKI	Public Key Infrastructure。インターネットトランザクションに関係する各関係者の有効性を確認および承認する、デジタル署名、認証局、他の登録機関のシステム。

RSA	デジタル署名と公開鍵暗号化システムを取得するための方法。 1978 年に最初に公開され、 Rivest 氏、Shamir 氏、Adleman 氏によって開発された。
SADB	セキュリティアソシエーションデータベース。暗号化鍵とアルゴリズムを指定するテーブル。 鍵とアルゴリズムは、安全なデータ転送で使用される。
SHA-1 アルゴリズム	セキュリティ保護されたハッシュアルゴリズム。メッセージ要約を作成するために 2^{64} 文字以下の長さを入力するときに操作する。これは DSA への入力となる。
SPI	セキュリティパラメータインデックス。受信したパケットの暗号解除に受信側が使用する SADB の行を指定する整数
Triple-DES	Triple-Data Encryption Standard。 168 ビットの鍵を提供する対称鍵暗号化方法。
回復検出	障害の発生後、NIC や NIC から第 3 層の装置への経路が、正しく動作し始めたことを検出する処理
鍵管理	セキュリティアソシエーションを管理するための手法。
カプセル化	ヘッダーとペイロードを 1 番目のパケット内に配置し、そのパケットを 2 番目のパケットのペイロード内に配置すること。
カプセル化セキュリティヘッダー	IPv6 データグラムに対して認証と完全性を提供する拡張ヘッダー。
逆方向トンネル	モバイルノードの気付アドレスで始まり、ホームエージェントで終わるトンネル
近傍検索	接続されているリンク上にある他のホストをホストが特定できるようにするための IP メカニズム。
近傍通知	近傍要請メッセージに対する応答、またはデータリンク層アドレスの変更を通知するために、ノードが自発的に近傍通知メッセージを送ること。
近傍要請	近傍のリンク層アドレスを決定するために、ノードによって送信される要請。また、キャッシュされたリンク層アドレスによって近傍が到達可能であるかを確認する。
公開鍵暗号化	次の 2 つの鍵を使用する暗号化システム。公開鍵は相互に認識している。 非公開鍵は、メッセージの受信者だけが認識している。 IKE により、IPsec の公開鍵が提供される。
最小カプセル化	ホームエージェント、外来エージェント、およびモバイルノードによってサポートされる任意の形態の IPv4 内 IPv4 トンネリング。最小カプセル化は、IP 内 IP カプセル化よりも 8 ないし 12 バイト少ないオーバヘッドしか持たない。
サイトローカルアドレス	単一サイト上でアドレスを指定するために使用する。

自動設定	IPv6 において、ホストが自身のインタフェースを自動的に設定すること。
順方向トンネル	ホームエージェントから始まり、モバイルノードの気付アドレスで終わるトンネル
セキュリティアソシエーション **(SA)**	1 つのホストから 2 番目のホストに、セキュリティ属性を指定するアソシエーション。
セキュリティパラメータインデックス **(SPI)**	受信者が受信したパケットを復号化するために使用する必要がある SADB (セキュリティアソシエーションデータベース) 内の行を特定する整数値
ステートフル自動設定	ホストが、インタフェースアドレスや設定情報、およびパラメータをサーバーから取得すること。
ステートレス自動設定	ホストが、ローカルに入手可能な情報と、ルーターが通知した情報を組み合わせて自身のアドレスを生成すること。
専用アドレス	インターネット経由で経路指定ができない IP アドレス
双方向トンネル	双方向にデータグラムを送信するトンネル
待機	グループ内の他の物理インタフェースに障害が発生するまでデータの伝送には使用されない物理インタフェース
登録	モバイルノードが、ホームにないときに自分の気付アドレスを自分のホームエージェントおよび外来エージェントに登録すること。
任意キャストアドレス	(一般的に別のノードに属す) 複数のインタフェースに割り当てられる IP アドレス。任意キャストアドレスに送られたパケットは、そのアドレスを持つ、プロトコルに基づき「最も近い」インタフェースに配送される。パケットの経路指定は、経路指定プロトコルの距離測定に応じて決定される。
デュアルスタック	IPv6 への移行に使用する、IPv4 と IPv6 の両機能を併せ持つプロトコルスタックで、スタックの残り部分は同じ。
対称鍵暗号化	送信側と受信側が単一の共通鍵を使用する暗号化システム。この共通鍵は、メッセージの暗号解除に使用する。 対称鍵は、IPsec での大量データ転送の暗号化に使用する。対称鍵システムの一例として DES がある。
トンネル	カプセル化される間データグラムが通過するパス
認証ヘッダー	IP データグラムに対し認証と完全性を提供する拡張ヘッダー。機密性は提供されない。
認証局 **(CA)**	デジタル署名および公開鍵と非公開鍵のペアの作成に使用するデジタル証明書を発行する、公証された第三者機関または企業。CA は、一意の証明書を付与された個人の ID を保証する。
ネットワークアクセス識別子 **(NAI)**	user@domain 形式でモバイルノードを一意に特定するために使用する。

ネットワークインタフェースカード **(NIC)**	リンクとのインタフェースになる、内部ネットワークアダプタおよび独立したネットワークアダプタカード
ノード	ホストまたはルーター
汎用経路指定カプセル化 **(GRE)**	ホームエージェント、外来エージェント、およびモバイルノードによってサポートされる任意の形態のトンネリング。他の任意の (または同じ) ネットワーク層プロトコルの配信パケット内で任意のネットワーク層プロトコルのパケットをカプセル化できるようにする。
パケット	通信回線上で、1 単位として送られる情報の集合。ヘッダーとペイロードで構成される。
物理インタフェース	リンクに対するノードの接続。この接続は通常、デバイスドライバとネットワークアダプタとして実装される。ネットワークアダプタによっては、qfe のように複数の接続点をもつ場合もある。このマニュアルでは、「ネットワークアダプタ」は「単一接続点」を示す。
物理インタフェースグループ	同じリンクに接続されている、システムの物理インタフェース群。グループ内のすべての物理インタフェースには、識別のための空文字列でない同じ名前が割り当てられる。
物理インタフェースグループ名	グループを識別する、物理インタフェースに割り当てられる名前。この名前の有効範囲は 1 つのシステム。同じグループ名を共有する複数の物理インタフェースは、物理インタフェースグループを構成する。
ファイアウォール	組織内の私的ネットワークまたはイントラネットを、インターネットなどの外部ネットワークからの侵入に対して保護する装置またはソフトウェア。
ホップ	2 つのホストを分離するルーターの数を判別するための手段。たとえば、始点ホストと終点ホストが 3 つのルーターで分離されている場合、ホストは互いに 4 ホップ離れている、という。
マルチキャストアドレス	特定の方法でインタフェースのグループを特定する IP アドレス。マルチキャストアドレスに送信されるパケットは、グループにあるすべてのインタフェースに配信される。
ユニキャストアドレス	単一のインタフェースを指示する IP アドレス。
リダイレクト	特定の終点に到達するために、ホストに対して最適な最初のホップノードを、ルーターが通知すること。
リンクローカル使用アドレス	自動アドレス設定などのために、単一リンク上でアドレスを指定するために使用する。
ローカル使用アドレス	ローカルの経路指定可能な範囲だけを対象とするユニキャストアドレス (サブネット内またはネットワーク内)。 また、ローカルまたはグローバルな一意の範囲を対象とすることもできます。
ルーター通知	ルーターが、各種のリンクパラメータおよびインターネットパラメータと共に、その存在を定期的にあるいはルーター要請メッセージに応じて通知すること。

ルーター発見	ホストが、接続されているリンク上にあるルーターを特定すること。
ルーター要請	ホストがルーターに対し、次に予定されている時刻ではなく、ただちにルーター通知メッセージを送信するように要求すること。

索引

A

AAAA レコード, 37, 51, 71, 81
ATM サポート, IPv6 over, 73

D

DEFAULT_IP 変数, 43
DNS
  AAAA レコード, 37, 71, 81
  IPv6 アドレスを追加, 37
  IPv6 拡張機能, 71
  PTR レコード, 52
  逆ゾーンファイル, 37
  ゾーンファイル, 37

E

/etc/default/inet_type ファイル, 43
  DEFAULT_IP 値, 65, 66
/etc/hostname6.interface ファイル, 34, 46
  IPv6 トンネリング, 67
  複数のネットワークインタフェース, 56, 57
/etc/hosts ファイル, 69
/etc/inet/inetd.conf ファイル, 63
/etc/inet/ipnodes ファイル, 36, 69, 70, 71
/etc/inet/ndpd.conf ファイル, 35, 50, 57, 60
  キーワード, 61
/etc/nsswitch.conf ファイル, 71

G

getent コマンド, ipnodes オプション, 53
gethostbyname コマンド, 71
getipnodebyname コマンド, 71

H

hostname.interface ファイル
  複数のネットワークインタフェース, 56, 57
hosts.byaddr マップ, 37, 70
hosts.byname マップ, 37, 70
hosts.org_dir テーブル, 36, 70

I

ifconfig コマンド, 39, 55, 66, 67
  -a オプション, 35
  IPv6 拡張機能, 57
  アドレスの追加, 57
in.ndpd デーモン, 56, 57
  オプション, 60
in.ripngd デーモン, IPv6 オプション, 62
inet6 オプション, route コマンド, 65
inetd デーモン, 63
ipnodes.byaddr マップ, 37
ipnodes.byname マップ, 37
ipnodes.org_dir テーブル, 36, 70
ipnodes オプション, getent コマンド, 53
IPsec, 32
  IPv6 カプセル化セキュリティヘッダー, 32
  IPv6 認証ヘッダー, 32

IPv6
ATM サポート， 73
DNS AAAA レコード， 51, 81
DNS 拡張機能， 71
DNS にアドレスを追加， 37
/etc/hostname6.interface ファイル，
46
/etc/inet/inetd.conf ファイル， 63
/etc/inet/ipnodes ファイル， 69, 70
/etc/inet/ndpd.conf ファイル， 50
getent コマンド， 53
ifconfig コマンド， 39
ifconfig コマンドの拡張機能， 57
in.ndpd デーモン， 60
in.ripngd デーモン， 62
IPv4, IPv6との相互運用性， 80
IPv6
IPv4 との相互運用性， 80
IPv4 との比較， 24
IPv4 有効化ホストアドレス， 16
netstat コマンド， 40, 65
NFS と RPC のサポート， 72
NIS+ 拡張機能， 70
NIS+ テーブル， 81
NIS+ にアドレスを追加， 36
NIS 拡張機能， 70
NIS にアドレスを追加， 36
NIS マップ， 81
nslookup コマンド， 51, 52
ping コマンド， 45, 65
route コマンド， 65
snoop コマンド， 44, 65
traceroute コマンド， 45, 66
アドレス， 26
アドレス解決， 21
アドレス指定， 14
プレフィックスフォーマット割り当て，
15
アドレス自動設定， 21, 27, 60
アドレス領域， 15
アドレス割り当てを表示， 39
アプリケーションとの相互作用， 80
移行， 75
IPv4 互換アドレス， 78
移行シナリオ， 81
移行ツール， 75, 76
移行要求， 75
エニーキャストアドレス， 14, 19, 24
IPv6 (続き)
拡張ヘッダー， 14
拡張ヘッダーフィールド， 14
宛先オプション， 14
カプセル化， 14, 32
断片化， 14
認証， 14
ホップバイホップオプション， 14, 30, 31
ルーティング， 14
監視， 38
機能， 11
近傍検索， 21, 24, 25, 57
近傍不到達検出， 21, 25
近傍要請， 22
近傍要請と不到達， 23
サービス品質機能， 30
フローラベル， 30
サイトローカルアドレス， 26
サイトローカル使用アドレス， 16, 17
自動トンネル， 79
重複アドレス検出， 21
情報を NIS+ で表示， 53
情報を NIS で表示， 52
ステートフルアドレス自動設定， 26, 28
ステートレスアドレス自動設定， 25, 28, 29,
82
セキュリティの改善， 32
次のホップの決定， 21
デュアルスタック， 76, 80
動作， 59
トンネリング， 67, 76
トンネリング機構， 78
トンネルの設定， 46
認証ヘッダー， 21, 32
ネームサービス情報の表示， 50, 51
ネームサービスの設定， 77
ネットワークステータスを表示， 40
ネットワークトラフィックの監視， 44
ノード使用可能， 33
パケットのカプセル化， 55
パラメータ探索， 21
表示出力を制御， 43
プレフィックス探索， 21
プロトコル概要， 27
ヘッダー
トラフィッククラスフィールド， 13, 31
ヘッダーオプション， 14
ヘッダーと拡張機能， 12

IPv6 (続き)
ヘッダーフィールド
宛先アドレス, 13
ソースアドレス, 13
次のヘッダー, 13
トラフィッククラス, 30, 31
フローラベル, 13
ペイロードの長さ, 13
ホップ制限, 13
ヘッダーフォーマット, 12
マルチキャストアドレス, 14, 16, 19, 24
マルチホームホストの探査, 45
モビリティサポート
ホームアドレス, 29
ユーティリティの拡張機能, 64
ユニキャストアドレス, 14, 16
リダイレクト, 21, 22, 24
リンクローカルアドレス, 25, 26, 27, 29
リンクローカル使用アドレス, 16, 17
ルーター通知, 22, 23, 24, 25, 28
ルーターの設定, 35, 50
ルーター発見, 60
ルーター要請, 22, 28
ルーティング, 20
ルートのトレース, 45
ローカル使用アドレス, 16, 17
IPv6 アドレス
一意性, 28
組み込み IPv4 アドレス, 18
IPv4 互換 IPv6 アドレス, 18
IPv6 パケットのカプセル化, 55
IPv4 マップ IPv6 アドレス, 18
IPv4 有効化ホストアドレス, 16
IPX アドレス, 16
IP アドレス, IPv6, 14

M

MTU, 24

N

netstat コマンド, 40, 66
-a オプション, 40
-f オプション, 40, 65
inet6 オプション, 40
netstat コマンド (続き)
inet オプション, 40
IPv6, 65
-p オプション, 65
NFS のサポート, IPv6, 72
NIS
IPv6 アドレスの追加, 36
IPv6 拡張機能, 70
NIS+
IPv6 アドレスの追加, 36
IPv6 拡張機能, 70
NIS+ テーブル、IPv6, 81
nisaddent コマンド, 36
nisserver コマンド, 36
nissetup コマンド, 36
nistbladm コマンド, 36
NIS マップ、IPv6, 81
NSAP アドレス, 16
nslookup コマンド, 72
nslookup コマンド
IPv6, 51, 52

P

ping コマンド
-A オプション, 66
-a オプション, 45
-a コマンド, 66
IPv6, 45, 65
PTR レコード、DNS, 52

R

route コマンド
inet6 オプション, 65
IPv6, 65
RPC のサポート, IPv6, 72

S

snoop コマンド
ip6 オプション, 44
ip6 プロトコルキーワード, 65
IPv6, 65

T

TCP/IP ネットワーク
  構成ファイル
    /etc/hostname6.interface, 56, 57
traceroute コマンド, 66
  -a オプション, 45, 66
  IPv6, 66
tun モジュール, 55, 67

V

/var/inet/ndpd_state.interface ファイル, 60

あ

宛先アドレスフィールド, IPv6 ヘッダー, 13
宛先オプションフィールド, IPv6 拡張ヘッダー, 14
アドレス
  IPv6, 26
  IPv4 可能ホスト, 16
  IPX, 16
  NSAP, 16
  エニーキャスト, 14
  サイトローカル使用, 16, 17
  集約グローバルユニキャストアドレス, 16
  ニュートラル相互接続, 16
  マルチキャスト, 14
  ユニキャスト, 14, 16
    集約グローバル, 16
  リンクローカル使用, 16, 17
  ローカル使用, 17
  ローカル使用アドレス, 16
アドレス解決, IPv6, 21
アドレス指定, IPv6, 14
アドレス自動設定
  IPv6, 21, 27, 60
アドレス領域, IPv6, 15
アプリケーション層, ゲートウェイ, 82

い

移行シナリオ, IPv6, 81
インターネットプロトコルセキュリティ, 32
インタフェース ID
  IPv6 サイトローカルアドレス, 18
  IPv6 リンクローカル使用アドレス, 17
インタフェースアドレス, IPv6, 14

え

エニーキャストアドレス
  IPv6, 19, 24

か

拡張ヘッダー, IPv6, 14
カプセル化フィールド
  IPv6 拡張ヘッダー, 14, 32
管理対象アドレス設定フラグ, ルーター通知, 28

き

逆ゾーンファイル, 37
近傍検索
  IPv6, 21, 24, 25
近傍検索デーモン, 57
近傍不到達検出
  IPv6, 21, 25
近傍要請, IPv6, 22
近傍要請と不到達, 23

く

グループ ID, マルチキャストアドレス, 19

こ

構成
  TCP/IP 構成ファイル
    /etc/hostname6.interface, 56, 57
構成ファイル
  TCP/IP ネットワーク
    /etc/hostname6.interface, 56, 57

さ
サービス品質
IPv6, 30
IPv6 フローラベルフィールド, 30
サイトローカルアドレス
IPv6, 26
インタフェース ID, 18
サブネット ID, 18
サイトローカル使用アドレス, 16, 17
サブネット ID, IPv6 サイトローカルアドレス, 18

し
自動アドレス設定フラグ, ルーター通知プレフィックスフィールド, 28
自動トンネル, IPv6, 79
次ホップ, 25
重複アドレス検出, IPv6, 21
重複アドレスの検出, アルゴリズム, 27
集約グローバルユニキャストアドレス, 16

す
スコープの値, マルチキャストアドレス, 20
ステートフルアドレス自動設定, 26, 28
ステートレスアドレス自動設定, 25, 26, 28, 29
IPv6, 82

せ
セキュリティ, IPv6, 32

そ
ソースアドレスフィールド, IPv6 ヘッダー, 13
ゾーンファイル, 37
その他のステートフル設定フラグ, ルーター通知, 28

た
断片化フィールド, IPv6 拡張ヘッダー, 14

つ
次のヘッダーフィールド, IPv6 ヘッダー, 13
次のホップの決定, IPv6, 21

て
デーモン
in.ndpd, 60
in.ripngd, 62
inetd インターネットサービス, 63
IPv6, 60
デュアルスタック
IPv6, 76, 80

と
トラフィッククラスフィールド
IPv6 ヘッダー, 13, 31
トラフィックフィールド, IPv6 ヘッダー, 30
トンネリング, 76
IPv6, 67, 78
ルーターの設定, 50
トンネル, IPv6 設定, 46

に
ニュートラル相互接続アドレス, 16
入力負荷均衡, 23
認証フィールド, IPv6 拡張ヘッダー, 14
認証ヘッダー
IPv6, 21, 32

ね
ネームサービス
IPv6 拡張機能, 69
IPv6 情報の表示, 50, 51

ネットワークインタフェース
  複数のネットワークインタフェース
    /etc/hostname6.interface ファイル, 56, 57
ネットワークマスク, 25

は
パケット
  同じフローに属する, 31
  フロー, 30
パラメータ探索, IPv6, 21

ふ
フォーマットプレフィックス, IPv6, 15
負荷均衡, 入力, 23
複数のネットワークインタフェース
  /etc/hostname6.interface ファイル, 56, 57
プレフィックス
  ルーター通知, 22, 24
    自動アドレス設定フラグ, 28
プレフィックス探索, IPv6, 21
プレフィックスフォーマット割り当て, IPv6 アドレス, 15
フロー, パケット, 30
フローラベルフィールド
  IPv6 サービス品質, 30
  IPv6 ヘッダー, 13
プロキシ通知, 24

へ
ペイロードの長さフィールド, IPv6 ヘッダー, 13
ヘッダーフィールド, IPv6, 13

ほ
ホップ制限フィールド, IPv6 ヘッダー, 13
ホップバイホップオプションフィールド
  IPv6 拡張ヘッダー, 14, 30, 31

ま
マルチキャストアドレス, 16
  IPv6, 19, 24
  グループ ID, 19
  スコープの値, 20

め
メッセージ, ルーター通知, 23

も
モビリティサポート
  IPv6, 29
  ホームアドレス, 29

ゆ
ユニキャストアドレス, 16
  集約グローバル, 16
  フォーマットプレフィックス, 16

り
リダイレクト
  IPv6, 21, 22, 24
リンク層アドレス, 23
リンクローカルアドレス
  IPv6, 25, 26, 27, 29, 67
リンクローカル使用アドレス, 16, 17
  インタフェース ID, 17

る
ルーター, パケットのフロー, 31
ルーター設定, IPv6, 35
ルーター通知
  IPv6, 22, 23, 24, 25, 28
  プレフィックス
    自動アドレス設定フラグ, 28
ルーター発見, IPv6, 60

ルーター要請
IPv6, 22, 28
ルーティング, IPv6, 20
ルーティングフィールド, IPv6 拡張ヘッダー, 14
ルートのトレース, IPv6, 45

ろ

ローカル使用, 16
ローカル使用アドレス, 17