# 第 VI 部

# Linux における IPv6/IPsec スタックの研究開発

# 第 6 部
# Linux における IPv6/IPsec スタックの研究開発

## 第 1 章　USAGI プロジェクトの概要と目的

USAGI プロジェクトは、Linux を中心により良い IPv6 環境を提供することを目的としたプロジェクトである。本プロジェクトは WIDE プロジェクトを中心として、有志によって構成されている。

USAGI プロジェクトの活動内容は、Linux の IPv6 スタックや IPv6 に関するライブラリ・アプリケーションの改良及び公開、成果物のコミュニティへの還元といった開発活動、また機会を捉えての啓蒙活動などである。これらの活動は IPv6 に関する検証技術の研究開発活動を行う TAHI プロジェクトと連携をとりつつ行っている。

USAGI プロジェクトでは開発成果を外部に公開し、誰でも参照可能なようにしている。以前はプロジェクトにおける開発成果を独自パッケージとして公開してきたが、活動の成果が実りコミュニティからの信頼を得ることができたことから、現在では Linux メインラインカーネルの開発に直接参加する形で成果を公開している。一方で現在でもより先進的な開発成果に関しては、開発ツリーを外部参照可能とすることで引き続き公開している。

2006 年度は 6 月に総務省より「平成 18 年度情報通信月間総務大臣表彰」を授与され、またメンバの一人が「2006 年度日本 OSS 貢献者賞」を受賞するなど、活動の成果が広く認められた年度となった。USAGI プロジェクトは、今後もグローバルな展開を続けていく。

なお、プロジェクトに関する最新の詳しい情報については <http://www.linux-ipv6.org> にて公開している。

## 第 2 章　2006 年の主な活動

### 2.1 IPv6 Mobility の設計と開発活動

#### 2.1.1 背景

USAGI プロジェクトでは、2003 年よりヘルシンキ工科大学と共同で Linux オペレーティングシステム上での Mobile IPv6 スタックの開発に取り組んできた。開発当初の Linux バージョン 2.4 に対する Mobile IPv6 スタック（MIPL）は、機能のほぼ全てをカーネル内に実装していた。その後、Linux カーネルツリーに大きな変更が加えられた Linux バージョン 2.6 への更新時に方針転換し、可能な限りユーザランドにてプロトコルスタックを実装し、Mobile IPv6 の仕様に最低限必要となる基本機能のみをカーネル内に実装する設計を新たに打ち出した（以降このプロトコルスタックを Linux バージョン 2.4 に対するスタックと区別するために MIPL2 と呼ぶ）。以降 2005 年末まで、Mobile IPv6 の基本仕様 [6, 112] の必要機能を MIPL2 で実現し、相互接続試験による動作検証や TAHI プロジェクトの提供するコンフォーマンス試験による機能検証に取り組み、成果を公開してきた。

#### 2.1.2 2006 年度の活動

2005 年度末には、MIPL2 の主要な機能は仕様を満たすまでに至り、安定した動作を確認することができた。2006 年はいよいよこれまでの成果をメインラインの Linux カーネルソースへマージする作業を開始した。マージに向けた主な作業は、MIPL に関わるカーネル内の機能を細分化する分類にはじまり、それぞれの設計方針が正しいかどうかの再検証を行い、カーネルメンテナと Linux カーネル開発のコミュニティへの説明、さらにそこでの議論を経て、メインラインへ受け入れられていった。マージに対しては、細分化した機能毎にパッチを公開し、個別に議論を行うことで明確に的を絞った議論をすること

ができている。さらに、機能によってはカーネルメンテナや Linux カーネル開発に関わる技術者の意見を取り入れて修正するなど柔軟に対応し、より品質のよい実装をメインラインに提供できている。また、機能毎に段階的にマージされていくメインラインの Linux カーネル上での Mobile IPv6 の動作検証は平行して継続し、動作状況など成果は公開している。

### 2.1.3 Linux-2.6 メインラインへのマージ

Linux-2.6 メインラインとは、Linus Torvalds 氏が管理する Linux ソースツリーの本流を意味する。Mobile IPv6[6, 112] は、IPv6 における事実上標準的なモビリティ・プロトコルであり、Linux カーネルのメインラインの一機能として利用可能となることが望まれている。

MIPL2 のカーネル機能は、様々な要素から構成されるためパッチは複数のカーネルソースの複数のディレクトリおよびファイルに分散している。また、メインラインへのマージは通常以下のプロセスを経て行われる。パッチセットとは、一つもしくは複数のパッチファイルの集合である。MIPL2 の場合もこの慣例に従い、以下のプロセスに従って、メインラインへのマージ作業を進めた。

(1) 機能毎にパーツ分けされたパッチセットをメーリングリスト上で公開
(2) コメントを受け付ける
(3) コメントに応じて必要な修正を加える
(4) カーネルメンテナが該当ソースツリー（ネットワーク機能の場合 net-2.6 ツリー）にパッチを取り込む（マージする）

#### 2.1.3.1 Mobile IPv6 カーネル機能概要

表 2.1 は、Linux カーネル内に必要な MIPL2 の機能を一覧にしたものである。機能は大別すると、XFRM、Mobile IPv6、IPsec、Neighbor Discovery（ND）、カーネル内のアドレス処理、ソースアドレス選択、経路、IPv6 コアの 8 つに分類される。最右の列はパッチの準備状況（ステータス）を示している。この欄が、「マージ済み」もしくは「他の開発者によって解決済み」となっている項目に関しては、パッチ機能がメインラインカーネルで利用可能な状況になっていることを意味している。「キャンセル」と書かれた項目に関しては、議論の結果不必要な機能との結論に達した項目である。その他のパッチに関しては、開発中であり今後パッチの公開、マージ作業が予定されている。なお、CN、HA、MN はそれぞれ Correspondent Node、Home Agent、Mobile Node の略である。

#### 2.1.3.2 XFRM への拡張

USAGI プロジェクトでは、XFRM という IPsec で使われている機構 [151] を拡張し、主に経路最適化機能の実現に用いた（表 2.1 項目 1 参照）。

**2.1.3.2.1 XFRM ベース機能への拡張**　XFRM には、policy・state という二種類のデータベースがある。

XFRM state は、従来から IPsec SAD（Security Association Database）として用いられている。

IPsec SAD の検索インタフェースでは、宛先アドレスリスト・SPI（Security Parameter Index）リストという二種類のリストが XFRM state に実装されており、主に送信時・受信時の暗号化・復号化のための検索にそれぞれ使っている。Mobile IPv6 の state 検索では、送信時においては宛先アドレスリストを IPsec と同様に利用することが可能だが、SPI あるいはそれに類する概念が Mobile IPv6 に無いため、従来のままでは受信時の検索が困難であったので、新たに送信アドレスリストを追加した。

XFRM policy は、従来から IPsec SPD（Security Policy Database）として用いられている。

Mobile IPv6 では、機能的には IPsec SPD として実現されている要素群で十分であったため、XFRM policy への機能追加はほぼ無い。一方、Mobile IPv6 のアプリケーションが XFRM policy のユーザとして新たに現れたことで複数のアプリケーションがそれぞれの XFRM policy を識別できる情報が必要となったため、XFRM policy の種別定義を追加した（2.1.3.3 項）。

**2.1.3.2.2 Mobile IPv6 専用拡張ヘッダのサポート**　XFRM では、state のプロトコルごとにヘッダ処理が分かれている。IPsec では、AH[119]・ESP[120]・IPComp[215] を XFRM プロトコルに定義し、それぞれの暗号化・復号化ヘッダ処理を実装している。Mobile IPv6 では、経路最適化で利用される経路ヘッダ・ホームアドレスオプションを XFRM プロトコルに追加し、パケット内のホームアドレスと気付けアドレスの交換処理を実装した。（表 2.1 項目 2-1, 2-2 参照）

**表 2.1.** カーネルパッチ一覧

| # | 大項目 | # | 小項目 | # | パッチ名 | 機能の位置づけ | 必須 | 機能分類 | ステータス |
|---|---|---|---|---|---|---|---|---|---|
| 1 | XFRM | 1 | XFRM | 1 | XFRM state support coa and HAO/RT2 | new | required | CN HA MN | マージ済み |
| | | | | 2 | XFRM state use source address hash | new | required | CN HA MN | マージ済み |
| | | | | 3 | XFRM bulk operation support | new | required | HA MN | 開発中 |
| | | | | 4 | find IPsec header place to insert it with HAO/RT2 | new | required | (CN) HA MN | マージ済み |
| | | | | 5 | XFRM state inbound for mip6 exthdrs | new | required | (CN) HA MN | マージ済み |
| | | | | 6 | update XFRM state last used timestamp | fix | optional | CN | マージ済み |
| | | | | 7 | decide a device by source address of IPv6 header | new | optional | (HA) MN | キャンセル |
| | | | | 8 | use mode as type even it was a flag | new | required | CN HA MN | マージ済み |
| | | | | 9 | use acquire even inbound for RO trigger | new | optional | MN | キャンセル |
| | | | | 10 | XFRM debug | new | optional | | キャンセル |
| | | | | 11 | Source address support for state id | fix | required | CN HA MN | マージ済み |
| | | | | 12 | non-fragment protocol support | new | required | CN HA MN | マージ済み |
| | | | | 13 | Sub policy support | new | required | CN HA MN | マージ済み |
| 2 | MIP6 | 1 | HAO | 1 | HAO sending | new | required | MN | マージ済み |
| | | | | 2 | HAO + ah6 sending | new | required | MN | マージ済み |
| | | | | 3 | HAO receiving | new | required | CN HA | マージ済み |
| | | | | 4 | BE report | new | required | CN HA | マージ済み |
| | | | | 5 | TLV parser | new | optional | CN HA MN | マージ済み |
| | | 2 | RT2 | 1 | RT2 sending | new | required | CN HA | マージ済み |
| | | | | 2 | RT2 receiving | new | required | MN | マージ済み |
| | | 3 | MH | 1 | MH handling | new | required | CN HA MN | マージ済み |
| | | | | 2 | MH sending | new | required | CN HA MN | マージ済み |
| | | | | 3 | MH receiving | new | required | CN HA MN | マージ済み |
| | | 4 | ICMP6 | 1 | Swap HAO address before sending ICMP6 error | new | required | CN HA MN | マージ済み |
| | | | | 2 | Swap RT2 address before sending ICMP6 error | new | required | MN | キャンセル |
| | | | | 3 | Swap RT address with segment left field when receiving ICMP6 error | new | optional | CN | 不明 |
| | | 5 | — | 1 | mip6 debug | new | optional | CN HA MN | キャンセル |
| 3 | IPsec | 1 | MIGRATE | 1 | PF_KEY MIGRATE | new | required | HA MN | 開発中 |
| | | | | 2 | MIGRATE extension | new | optional | HA MN | 開発中 |
| | | 2 | Misc | 1 | SP selector ifindex | new | required | HA MN | 開発中 |
| | | | | 2 | IPsec + TCP | fix | required | (CN) HA MN | 他の開発者によって解決済み |
| | | | | 3 | UDP6 outbound flowi fix | fix | unknown | | 不明 |
| | | | | 4 | IPsec6 + RT sending | fix | required | (CN) HA | キャンセル |
| | | | | 5 | IPComp and generic tunnel | fix | required | | 他の開発者によって解決済み |
| | | | | 6 | Inbound block policy | fix | required | | 他の開発者によって解決済み |
| | | | | 7 | allow to match wild-card user id at policy selector | fix | optional | | キャンセル |
| 4 | neighbor | 1 | proxy | 1 | proxy entry carries flag | new | required | HA | マージ済み |
| | | | | 2 | dump all proxy entry | new | optional | HA | パッチレビュー中 |
| | | | | 3 | don't do proxy forwarding when unicast ND | fix | required | HA | マージ済み |
| | | | | 4 | don't do proxy forwarding to link-local destination | new | required | HA | マージ済み |
| | | | | 5 | don't update neighbor cache when NA destined to proxied entry | fix | required | HA | マージ済み |
| | | | | 6 | proxy NDP sysctl | new | optional | HA | マージ済み |
| | | 2 | — | 1 | fix ndisc_flow_init to use ifindex | fix | unknown | (MN) | マージ済み |
| | | 3 | — | 1 | fix fl6_merge_options | fix | required | CN HA | マージ済み |
| | | 4 | RA | 1 | Use it as a default router in receiving RA | fix | required | MN | マージ済み |
| | | | | 2 | Use it as a prefix in receiving RA | fix | required | MN | マージ済み |
| 5 | address | 1 | flag | 1 | HoA flag | new | required | MN | マージ済み |
| | | | | 2 | never do DAD for HoA | new | required | MN | マージ済み |
| | | 2 | lifetime | 1 | change address lifetime from user-land | new | required | MN | マージ済み |
| | | 3 | prefix | 1 | add prefix info from user-land | new | required | MN | パッチレビュー中 |
| 6 | source address selection | 1 | — | 1 | source address selection: USAGI arch | new | required | MN | マージ済み |
| | | 2 | — | 1 | source address selection: SUBTREE fix | new | required | MN | 他の開発者によって解決済み |
| | | 3 | HoA | 1 | source address selection: HoA support | new | required | MN | マージ済み |
| 7 | routing | 1 | multiple table | 1 | multiple table or rule for policy routing | new | required | HA MN | 他の開発者によって解決済み |
| | | 2 | SUBTREE | 1 | SUBTREE fix for policy routing | fix | unknown | HA MN | 他の開発者によって解決済み |
| | | 3 | — | 1 | removing netlink_skb_parms | fix | unknown | | マージ済み |
| | | 4 | anycast | 1 | anycast routing | fix | unknown | HA | 他の開発者によって解決済み |
| 8 | ipv6 | 1 | cmsg | 1 | ipv6 cmsg 2292 fix in receiving | fix | required | MN | 他の開発者によって解決済み |
| | | 2 | — | 1 | more optimized inet6_skb_parm | fix | optional | | キャンセル |
| | | 3 | ipv6 tunnel | 1 | ipv6 tunnel fix | fix | unknown | HA MN | パッチレビュー中 |

#### 2.1.3.3 マルチ・ポリシ

データベース識別用に XFRM policy エントリに種別フィールドを追加し、値として MAIN・SUB を指定可能とした。パケット処理では、最も相応しい XFRM policy エントリが MAIN・SUB からそれぞれ一つずつ選ばれ、仕様に従ったヘッダ順序で並び替えてから従来の処理を行う。また、カーネルの下位互換性のために、種別が未指定のエントリや PF_KEY ソケット [139] 経由でのデータベース操作は、MAIN 種別として扱うようにした。よって Mobile IPv6 を IPsec とともに使う場合は、IPsec 用を MAIN、Mobile IPv6 用を SUB で利用する。（表 2.1 項目 1-13 参照）

#### 2.1.3.4 Mobility Header 処理

Mobile IPv6 のシグナリングで用いる Mobility ヘッダ（MH）の処理はカーネルの RAW ソケット機能を利用する。ただし、処理のほとんどはアプリケーションで実装され、カーネルの処理は以下のみである。（表 2.1 項目 2-3 参照）

(a) 送受信 MH チェックサム
(b) 受信 MH ヘッダ長検証
(c) 受信 MH ペイロードプロトコル検証

(a) の機能はカーネルの従来の RAW ソケット処理に類して実装した。(b)(c) の機能は、いずれも検証失敗の場合に ICMP エラーを返信する仕様である。ICMP エラーの処理では、元の受信パケットをそのままの形で ICMP データ部にコピーする必要があり、アプリケーションでの実装が容易でないことから、(b) および (c) の機能はカーネル内で実装した。

#### 2.1.3.5 IPv6 Neighbor Discovery への拡張

Neighbor Discovery への拡張は、以下の 3 点である。（表 2.1 項目 4 参照）

(a) アドレス種別を表すフラグの追加
(b) RA 受信処理の細分化
(c) アドレスを手動で設定した場合のアドレスとプレフィックスルートの有効時間設定

(a) では、ホームアドレスを示すためのアドレス種別および重複アドレス検知（Duplicate Address Detection）を実行するか否かの情報を追加した。

(b) では、ルータからの RA を受信した際に、ホストが以下の 4 通りの処理を行うことを可能にした。

(b-1) 受信した RA を元に IPv6 アドレス、デフォルトルートを設定する。
(b-2) 受信した RA を元にデフォルトルートのみを設定する。
(b-3) 受信した RA を元に IPv6 アドレスのみを設定する。
(b-4) 受信した RA を無視する。

(c) では、Mobile Node（MN）が RA 受信時に、ユーザランドからカーネルへアドレスおよびデフォルトルータ情報を設定可能にするため、任意の IPv6 アドレスの valid lifetime および preferred lifetime、そしてプレフィックスルートの有効期限を設定する機能を実装した。

上記 (a) と (b) は、既に Linux カーネルソースのメインラインにマージされている機能である。(c) は実装が完了し、今後メーリングリスト上で公開し、メインラインへのマージを要請する予定である。

#### 2.1.3.6 ポリシ・ルーティング

Mobile IPv6 で定義されている MN は、ホームアドレスをソースアドレスとして送信する場合と、気付けアドレスをソースアドレスとして送信する場合がある。

ホームアドレスがソースアドレスとして選択されるか、気付けアドレスがソースアドレスとして選択されるかの判断は、MN が送信する Mobile IPv6 用のシグナルメッセージの種類や、MN 上で動作しているアプリケーションがソースアドレスを指定しているかどうかに依存するが、いずれの場合もパケットが正しくルーティングされることが必要である。

ポリシ・ルーティングは、このような機能を実現するためのルーティングに対する拡張である。ポリシ・ルーティングは、ソースアドレス・ルーティングとマルチ・ルーティング・テーブルで構成されており、以下ではそれぞれの機能について説明する。（表 2.1 項目 7 参照）

**2.1.3.6.1 ソースアドレス・ルーティング**　ソースアドレス・ルーティングは、端末がルーティングテーブルの検索キーとしてソースアドレスを使用するルーティングテーブルの検索方式である。この機能は、2.4 系カーネルを対象とした MIPL およびそのコードをマージした USAGI の 2.4 系カーネルのコード上で実装されていた。

しかし 2.4 系カーネル上でのソースアドレス・ルーティングは、宛先アドレスを検索キーとしてルーティ

ングテーブルを検索していた元々のルーティングテーブルの検索方式に対して、ソースアドレスを検索キーとして変更する実装で Mobile IPv6 に特化した実装となっていた。

2.6 系カーネルでは、上で実装する方針として、後述するマルチ・ルーティング・テーブルと組み合わせることにより、既存のルーティング機能との親和性を保ちつつ、Mobile IPv6 以外の用途にも利用可能な汎用機能として実装した。(表 2.1 項目 6 参照)

**2.1.3.6.2 マルチ・ルーティング・テーブル** マルチ・ルーティング・テーブルは、ルーティングテーブルを複数持つことを可能とし、それぞれのルーティングテーブルの選択条件(ポリシ)を、ユーザが設定可能とした機能である。(表 2.1 項目 7 参照)

ポリシとして選択できる要素は、ソースアドレスだけでなく、インタフェース、トラフィッククラスなども設定できるように拡張され、Mobile IPv6 だけでなくマルチホーム環境に対しても応用可能なように実装した。

### 2.1.3.7 IPsec への拡張

IPsec への拡張は、提案方式 [243](以降「MIGRATE 機能」と表記)に基づいている。MIGRATE 機能は、トンネル・モードの IPsec SA のエンドポイントを更新するための拡張機能である。

MIPL2 では、従来から MIGRATE 機能が実装されていたが、カーネルマージに向けたパッチ作成において、いくつかの機能上の修正点が認められたため、再度実装を行った。主な修正点は、SA バンドルの対処を考慮して MIGRATE メッセージのフォーマットを変更した点である。修正の目的は、一つの MIGRATE メッセージによって、複数の該当する SA(該当する SP と関連付いたバンドルされた SA)全てのエンドポイントを更新可能にするためである。これを実現するために、更新対象の SA を特定する情報(プロトコル、モード、REQID、古いエンドポイント)を複数指定可能とするようメッセージを変更した。その他、一度に送信するパッチのサイズを考慮して、最初に送付するパッチでは policy/state のバルク処理を省略することとした。(表 2.1 項目 3-1 参照)

現在、実装が進められておりテスト作業が終了した後にパッチの公開を予定している。

### 2.1.3.8 MIPL2 デーモン(mip6d)への影響

これまで説明してきた Linux カーネルに対する変更・修正は、マージの過程で若干の修正が加わったものが多い。こうしたカーネルパッチの変更は、既存の MIPL2 デーモンプログラム(mip6d)に影響を与える。具体的には、以下の修正が必要となる。

(a) 2.1.3.3 に伴う、マルチ・ポリシの対応
(b) 2.1.3.5 に伴う、プレフィックス管理の追加
(c) 経路最適化のトリガとなるポリシに含まれている、インタフェース識別子の削除
(d) カーネルから要求される XFRM state の扱いの変更(新規設定ではなく更新とする)
(e) MH を付与する state のキャッシュ更新処理の追加
(f) カーネルのヘッダ定義の修正
(g) プロキシ Neighbor Discovery 機能の有効・無効設定インターフェースの追加
(h) 2.1.3.7 に伴う、変更された MIGRATE メッセージの対応

(a) は mip6d が扱う XFRM policy および XFRM state を IPsec と分離してマルチ・ポリシの機能を使用する機能である。

(b) は、mip6d からの IP アドレス設定により、アドレス設定に加えてプレフィックスルートを設定することを可能にする機能である。

(c) は、カーネルから通知される state 要求にインタフェース識別子が入っていないが、MIPL2 の実装はインタフェース識別子が含まれることを期待している差分の吸収である。

(d) は、カーネルが state 要求を mip6d に通知する場合の修正である。カーネルが state 要求を行う場合、カーネルは事前に state の雛型を登録するため、mip6d はその雛型に対して更新する必要がある。MIPL2 は新規登録することを期待しているためこの修正を行った。

(e) は、カーネル内の state バンドルと呼ばれるパケット処理効率化のためのキャッシュ機構を明示的に無効にする処理である。MH 送信はバインディングと無関係に行うため、この無効化は必須の処理であるが、MIPL2 開発時は、カーネル内部でパケット送信のたびに強制的に更新していた。

(f) は、MIPL2 が動作するカーネルのバージョンに合わせた修正である。

(g) は、無効設定すると、非プロキシノードにお

**表 2.2.**　カーネル 2.6.19 の動作状況

| | 基本機能 | IPsec トランスポートモード | IPsec トンネルモード |
|---|---|---|---|
| CN | ○ | —*3 | — |
| HA | ○*1 | ○ | ×*4 |
| MN | ×*2 | N/A | ×*4 |

いて、転送パケットごとに生じる空のプロキシエントリを検索する無駄な負荷を回避できる機能である。Home Agent（HA）では、この設定を明示的に有効にする必要がある。

(h) は、変更された MIGRATE メッセージフォーマットに従って mip6d を修正する必要があることを示している。

#### 2.1.3.9 マージ状況

2007 年 12 月末での状況としては、IPsec の MIGRATE 機能（2.1.3.7）およびプレフィックス情報のコンフィギュレーション機能（2.1.3.5 の (c)）を除き、MIPL2 が機能する上で必要な全ての機能がメインラインのソースにマージ済みである。（表 2.1 参照）

### 2.1.4 動作状況

Linux カーネル 2.6.19（2006 年 11 月末リリース）と、ユーザランドに必要なパッチを組み合わせた動作実績は表 2.2 の通りである。

Correspondent Node（CN）に関しては、基本動作に問題は無い。また、マルチ・ポリシの実現によって、MN と CN との間でトランスポートモードの IPsec SA を確立した状況下でノード間の通信を IPsec によって保護することが可能となった（*3)。

HA に関しては、ポリシ・ルーティング機能に依存した一部異常系エラー処理を除いて、一連の正常系の処理が動作可能となっている（*1)。IPsec トンネルに関しては、MIGRATE 機能のパッチが開発中であるため利用不可能である（*4)。

MN に関しては、機能上ポリシ・ルーティング機能に依存度が高いため、正常系の動作に問題が生じている（*2)。IPsec トンネルに関しては、HA と同様に MIGRATE 機能のパッチ待ちの状態である（*4)。

### 2.1.5 今後の展開

今後は、まずメインラインにマージされていない必要な機能（MIGRATE 機能やプレフィックス情報のコンフィギュレーション等）のパッチを公開し、マージを済ませる必要がある。次に、最新の Linux カーネルソース上で問題なく動作する Mobile IPv6 プロトコルスタック（ユーザランド）の配布に向けて、各種バグやソフトウェアの修正を行っていく。これまでと同様にコンフォーマンステストと相互接続性テストを通じてソフトウェアに不具合がないかどうかのチェックを行っていく。ユーザランドへの変更は、バグフィックスであり、機能追加や設計上の変更などの大きな変更は予定していない。ヘルシンキ工科大学 GO-Core プロジェクトメンバとの緊密な連携が必要不可欠である。

## 2.2 パケットフィルタに関する開発活動

### 2.2.1 概要

USAGI プロジェクトは Linux の IPv6 スタックや IPv6 に関するライブラリ、アプリケーションの開発、改良を行っており、パケットフィルタ機能もその対象となっている。今年度、USAGI プロジェクトは昨年度に引き続き IPv4/IPv6 に対応した Connection Tracking のメンテナンスを行った。また、パケットフィルタの設定コマンド iptables、ip6tables のコード共通化を行った。以下では、これらの内容について報告する。

### 2.2.2 2006 年度の開発内容

#### 2.2.2.1 IPv4/IPv6に対応したConnection Tracking のメンテナンス

Connection Tracking は機器に入ってくる TCP や UDP のフローの状態変化を追跡する Linux カーネルの一機能である。Linux には IPv4 専用の Connection Tracking（以下 ip_conntrack と記す）が実装されていたが、USAGI プロジェクトが開発した IPv4/IPv6 対応 Connection Tracking（以下 nf_conntrack と記す）が 2005 年 11 月 Linux カーネルに採用された。

今年度、USAGI プロジェクトは nf_conntrack のコードメンテナンスを行った。その内容は、Linux

コミュニティのメーリングリストに投稿されるパッチのレビュー、それに伴う議論、および不具合修正である。今年度 USAGI プロジェクト以外の開発者からは主に以下のパッチが投稿された。

- 肥大化した nf_conntrack のコードを複数ファイルに分割するパッチ
- ip_conntrack に実装されたアプリケーションプロトコル用モジュールを nf_conntrack へ移植するパッチ
- ip_conntrack を利用した IPv4 NAT 機能を、nf_conntrack を利用したものへ移植するパッチ

このように、今年度は USAGI プロジェクト以外の開発者による nf_conntrack 関連の開発も多く行われ、ip_conntrack から nf_conntrack への移行が着実に進んだ。

nf_conntrack に関連する開発が進む一方で新たな課題も出現している。その一つは、コネクション毎に確保するメモリスペースを管理しづらいというものである。nf_conntrack が追跡する情報はコネクションに適用する機能により異なるため、従来の nf_conntrack はそれぞれに適した大きさのメモリスペースを確保していた。しかし IPv4 NAT 機能の対応に伴い、追跡すべき情報が動的に変更される可能性が出た。そこで現在の nf_conntrack は、IPv4 NAT 機能がカーネルにロードされている場合常に最大のメモリスペースを確保するよう変更されている。これは一時的な処置であり、使用メモリ量の抑制とメンテナンスの容易性を兼ね備えた方法が必要とされている。

#### 2.2.2.2 パケットフィルタの設定コマンド iptables、ip6tables のコード共通化

現在 Linux では IPv4、IPv6 それぞれのパケットフィルタを設定するユーザコマンドが別々に実装されている（それぞれ iptables、ip6tables）。しかし、これらは以下を扱う処理部で類似部分を多く含むため、長らく共通化が望まれていた。

- 操作対象のパケットを選択するマッチルール用モジュール
  ex.: TCP/UDP ポート番号によるマッチングなど
- パケットに適用する操作を指定するターゲットルール用モジュール
  ex.: 破棄、通過許可、ロギングなど

そこで、これらの実装に必要な API やこれらのモジュールを扱う処理部を iptables、ip6tables で共通化した。これにより新たなモジュールを開発する場合、共通化された API で 1 つのモジュールを実装すれば iptables、ip6tables 両方に対応できるようになる。

### 2.2.3 開発体制

昨年度に引き続き、nf_conntrack の開発に際しては、Linux におけるパケットフィルタや NAT 機能の開発、メンテナンスを行っている Netfilter Project（`http://www.netfilter.org`）と密に連携しながら進めた。また昨年度 USAGI プロジェクトのメンバの 1 人が Netfilter Project のコアメンバとなったことで、iptables のコードコミット権を得たなど、今年度の開発がさらにスムーズになった。

### 2.2.4 現在の状況と今後の予定

2.2.2.1 で述べた nf_conntrack に対する全ての変更は Linux 2.6.20 に取り込まれる予定であり、Linux 2.6.20 に向けこれらの変更に伴う不具合修正等を引き続き行う予定である。また、メモリスペースの割り当て方法について、使用メモリ量の抑制とメンテナンスの容易性を兼ね備えた方法を開発者間で議論していく予定である。なお、現在 nf_conntrack は ip_conntrack と同レベルの機能全てを実現しているため、十分な nf_conntrack の安定化作業の後 ip_conntrack は Linux カーネルから削除される予定である。但し、ユーザから見た使用方法に変更はない。

iptables、ip6tables のコード共通化については、2006 年 11 月、Netfilter Project のメーリングリスト上にそのパッチを投稿済みである。他の開発者からの同意を得たため、iptables 1.4.0 に取り込む予定である。なお、この変更に伴うコマンドの使用方法についても変更はなく、ユーザはこれまでどおり iptables、ip6tables を利用可能である。

ここ 2 年で nf_conntrack の導入、カーネルとユーザコマンドにおける IPv4/IPv6 処理部の共通化など、パケットフィルタ関連の API が大きく変化している。それに伴い開発者向けドキュメントが陳腐化しているため、今後これらドキュメントの整備を行いたいと考えている。

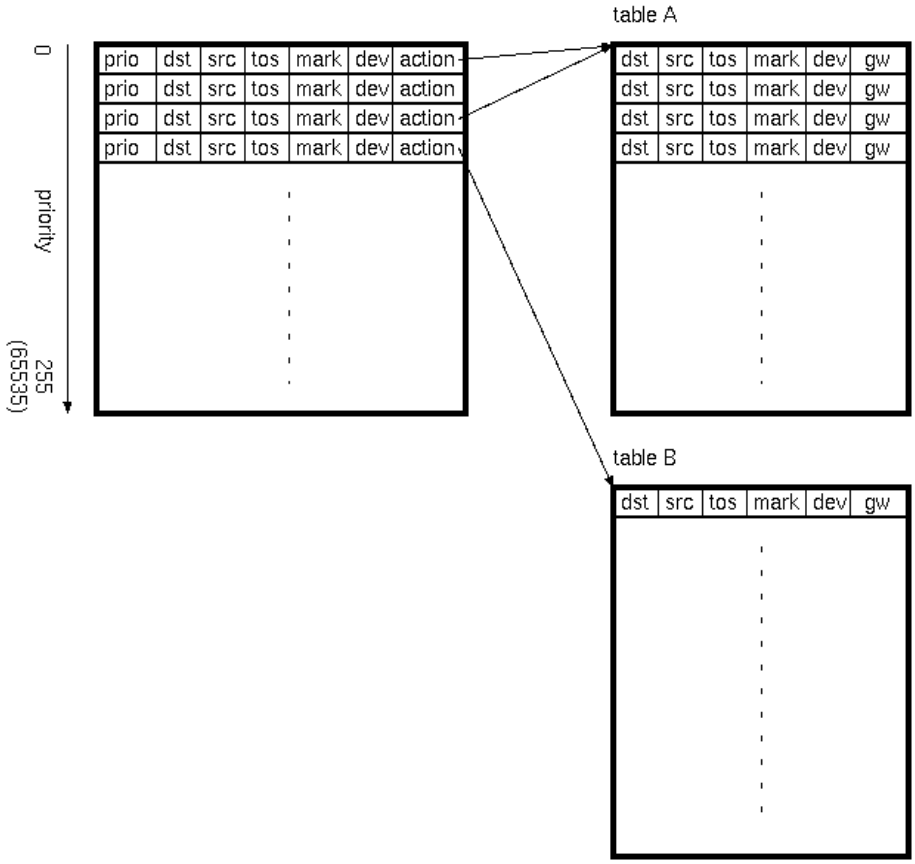



**図 2.1.** IPv4 Policy Routing Table

## 2.3 送信元アドレスに基づく経路選択の設計

### 2.3.1 ポリシ・ルーティングとその Linux 実装

インターネット上で次ホップ（Next Hop）の選択を行う経路選択（Routing）は、宛先アドレス（および、あまり使われていないが、サービス型（Type of Service; ToS）フィールドに基づいて行われることを原則としている。しかし、場合によっては、それ以外の要素、たとえば、パケットの送信元アドレスなどによって特定の経路を選択したいことがあり、ポリシ・ルーティング（Policy Routing）と呼ばれる。

Linux におけるポリシ・ルーティングは、IPv4 に関してはかなり古くから存在しており、ip rule コマンドにより優先度付で設定されるルーティング・ポリシ・データベース（Routing Policy Database; RPDB）に、アクション並びに経路表を関連づける方法で実装されている（図 2.1）。

個々の経路ルールに設定できる項目には次のものがある：

- 送信元アドレス（from）
- 宛先アドレス（to）
- サービスタイプ（tos）
- Netfilter モジュールなどによるマーキング（mark）
- 入力デバイス（iif）

この概念そのものは IPv4 に特化したものではなく、IPv6 にも援用できるものであるが、実際には、機能の断片だけが実装された状態で長い時間が経過していた。しかし、Linux における Mobile IPv6 機能がこの機能を利用することから、ポリシ・ルーティング機能の整理と汎用的実装が重要な課題となった。

### 2.3.2 パケット処理における送信元アドレス選択と経路選択の相互作用

Mobile IPv6 における移動ノード（Mobile Node; MN）がポリシ・ルーティングを活用することを例として、IPv6 のポリシ・ルーティングは、ホストでの利用も積極的にされることが想定される。この際、問題となるのが、ポリシ・ルーティングを行うノード自らが送信を行うパケットである所謂自発パケッ

トに関係する、送信元アドレス選択と経路選択との相互作用である。

送信元アドレスは、アプリケーションなどの上位層から指定されないことがしばしばであり、その際には、カーネルが適切なものを選択する。IPv6 におけるその方法は、RFC3484 Default Address Selection for Internet Protocol version 6（IPv6）として規定されている（注：IPv4 でも同様の議論は可能であるが、同文書の範囲外である）。この中で、経路選択との相互作用に関しては、経路選択、特に、出力インタフェース選択が、送信元アドレス選択に先だって行われることを前提としているとされており、実際、従来の Linux IPv6 実装でも、送信元アドレスの選択は、経路選択が完了したのちに行われるように実装されている。

一方、ポリシ・ルーティングでは、検索すべき経路表の選択基準である RPDB の検索において、宛先アドレスに加え、送信元アドレスが考慮の対象となる。ここで、Mobile IPv6 では、送信元アドレスとしてはホーム・アドレス（Home Address; HoA）を優先し、また、Linux 実装では HoA を送信元アドレスとするパケットを特別扱いしてトンネル経由でホーム・エージェント（Home Agent; HA）に送るためにこの機能を用いている。しかし、従来の手順では、経路選択を行う段階では、送信元アドレスは未だ選択されておらず、上位層から指定されていない限り、未指定アドレスのままである。アプリケーションが、パケットの送信時に送信元アドレスを適切に選択していることは期待できないため、カーネルは未指定な送信元アドレスに関しても適切に HoA を選択し、それに基づいて経路を選択するように構成する必要がある。

### 2.3.3 Linux における IPv6 経路表の実装と送信元アドレスに基づく経路選択の設計

Linux における IPv6 経路表の実装は図 2.2 のような構成である（注: 点線は、経路表を検索する際にメモリを参照する流れの概略を示す）。経路表は宛先アドレス用と送信元アドレス用の 2 段 Radish Tree で構成されており、rt6_info 構造体のリスト構造により、同一な宛先アドレスと送信元アドレスの組で複数の異なる経路を保持することができる。

上位層によって指定されていない場合における、送信元アドレスの選択については、現時点で、図の (1)–(5) の候補がある。

(1) RPDB を検索する前
(2) 宛先アドレスに対応する Main Tree を検索する前
(3) 宛先アドレスが検索された後、送信元アドレスに対応する Sub Tree を検索する前
(4) 宛先及び送信元アドレスの検索を終了し、対応する複数の経路の比較ごと
(5) 経路選択が完了した後

これらの組み合わせにより、いくつかの方式が検討された。

(a)(1): RPDB の前で行う方法。

この場合、送信元アドレス選択は出力インタフェース選択の結果を反映できない。

(b)(2) 及び (5): RPDB 検索において、個々のルールのアクションが経路表を検索するものである場合に限って、一旦送信元アドレスを無視した上、(2) において、ルールに送信元アドレスが含まれていればそのルールに制限された送信元アドレス選択を行い、そうでなければ、(5) において送信元アドレスを補完し、最終的に RPDB との整合性を確認する方法。

(a) と同様、送信元アドレス選択は出力インタフェース選択の結果を反映できない。

(c)(3) 及び (5): (b) と同様だが、(2) のかわりに (3) において、Sub Tree が存在する場合を追加的条件とするもの。

送信元アドレスが RPDB や経路に関係していなければ要求を満たすことができるが、そうでない場合は、選択の順序が規定と合致しておらず、出力インタフェースの選択の結果を反映できない。

(d)(3) および (4): (c) と同様だが、(3) において全ての関連する出力インタフェースの可能性を網羅して検索し、もっとも送信元アドレスとの一致長が長く、送信元アドレスの評価の高いものを選択する方式。

送信元アドレスが RPDB や経路に関係していなくても要求を満たすことができる可能性が高いが、実装がかなり複雑で実装速度が遅くなる。また、複数の選択肢があった場合にどう選択するかも課題である。

(e)(5): RPDB 選択時、未指定の送信元アドレスは、送信元アドレスに関連しない RPDB エントリのみを有効として扱う方法。

結果として出力されるパケットにおいて、送信元アドレスが RPDP のルールと、一見、矛盾する可能

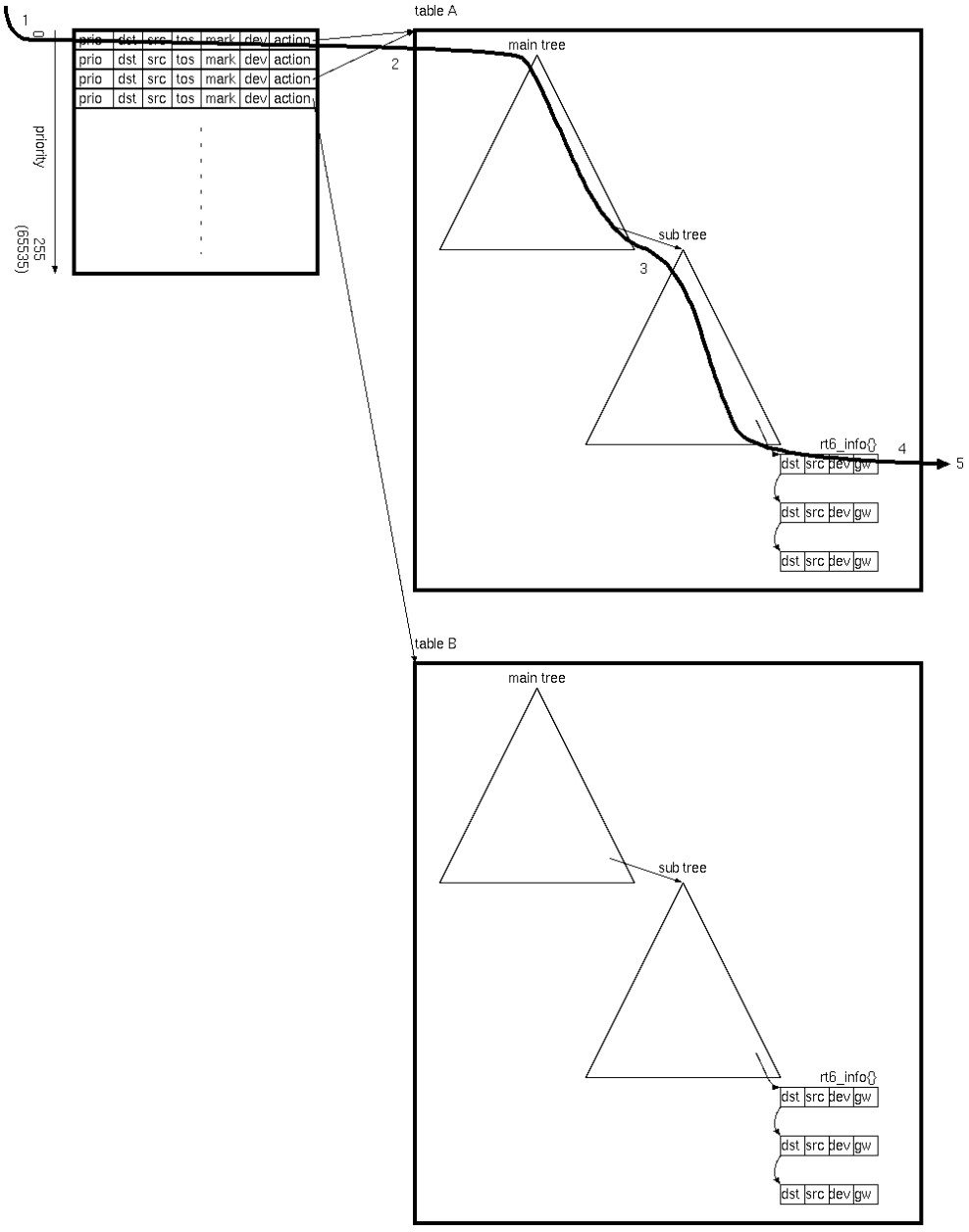



**図 2.2.**　IPv6 Routing Table

性がある。この方法で Mobile IPv6 実装が動作可能か検討する余地がある。

(f) その他

なお、RPDB 検索で一旦送信元アドレスを無視する場合を、個々のルールのアクションが経路表を検索するものである場合に限る理由は、そもそも、経路選択とはパケットを終点にまで正しく届けることが目的であるから、アクションとしてパケットの破棄を指定することは必ずしも正統的な方法でなく、Netfilter を用いるべきであるが、もしパケットの破棄が指定される場合にも送信元アドレスを無視すると、送信元アドレスが指定されていないとほぼ常にそのルールに合致してしまい、事実上パケットの送信が困難になるためである。

### 2.3.4 今後の展開

前節で掲げたいずれの方式も少なからず問題があると予想されているが、未だ絶対的な結論には達していない。今後は、いくつかの実装を通じてさらなる検討を行う予定である。

## 2.4 IPv6 Multicast の設計と開発活動

### Linux におけるマルチキャストの実装

本節では、Linux におけるマルチキャスト実装の現状について述べる。IPv4 と IPv6、ならびにクライアントとしての機能とルータとしての機能に分類して述べる。なお、本報告書で述べる現状に関しては、Linux kernel 2.6.19 に準拠している。

#### 2.4.1 IPv4 マルチキャストの実装

IPv4 マルチキャストクライアントの機能としては、IGMPv2 と IGMPv3 をサポートしている。通信相手により判断して、IGMPv3 をサポートしていないマルチキャストルータが存在する場合には、自動的に IGMPv2 による制御に切り替える仕組みとなっている。

また、マルチキャストルータ機能は、PIM-SM の version 1 と version 2 をサポートしている。マルチキャスト経路表は、BSD 系の実装と同じく、マルチキャスト経路をフォワーディングキャッシュとして保持する実装となっている。API も BSD 系と同じものになっている。

現在 Linux にて利用できる IPv4 PIM ルーティングデーモンの実装は、次の 2 つである。

- USC pimd (`http://netweb.usc.edu/pim/`)
- XORP (`http://www.xorp.org/`)

#### 2.4.2 IPv6 マルチキャストの実装

Ipv6 マルチキャストクライアントの機能としては、MLDv1 と MLDv2 をサポートしている。IPv4 と同じく、マルチキャストルータがサポートしている機能によって判断して自動的に切り替える仕組みとなっている。

マルチキャストルータ機能は、現在の Linux kernel には実装されていない。過去に、いくつかのプロジェクトによる実装があったが、本流 kernel には取り入れられることはなかった。現在利用できるマルチキャストルーティング実装としては、次の 2 つである。

- Linux IPv6 Multicast Forwarding (`http://clarinet.u-strasbg.fr/~hoerdt/dev/linux_ipv6_mforwarding/`)
- MRD6 (`http://artemis.av.it.pt/mrd6/`)

Linux IPv6 Multicast Forwarding 実装は、Linux kernel 2.6.7 に対するパッチ形式となっている。この実装は USAGI プロジェクトの開発ツリーに取り込んであり、現在 Linux kernel 2.6.19 にて動作するよう多少の修正が加えられている。この実装は IPv4 のマルチキャストルーティング実装、ならびに BSD 系 IPv6 マルチキャストルーティング実装と類似のものであり、マルチキャスト経路をフォワーディングキャッシュとして保持する実装となっている。

一方、MRD6 は特殊な実装であり、パケットフォワーディングならびにパケットコピーをユーザランドのデーモンで行ってしまう実装である。PACKET Socket を利用してユーザランドからパケットの監視を行い、操作を行っている。

また、IPv6 PIM ルーティングデーモンの実装は、次の 3 つである。

- pimd6sd for Linux and ∗BSD (`http://clarinet.u-strasbg.fr/~hoerdt/dev/pim6sd_linux/`)
- MRD6 (`http://artemis.av.it.pt/mrd6/`)
- XORP (`http://www.xorp.org/`)

#### 2.4.3 IPv6 マルチキャストルーティング設定

本報告書では、Linux IPv6 Multicast Forwarding を Linux kernel 2.6.19 に適用したものと、pim6sd for Linux and ∗BSD を利用したマルチキャストルータの構築例を示す。

まず、カーネルコンパイル時のオプションとして、以下を有効にする。

- IPV6_MROUTE
- IPV6_PIMSM_V2

次に、カーネル起動後、以下の設定を sysctl もしくは proc filesystem を用いて行う。

- net.ipv6.conf.all.mc_forwarding = 1
- net.ipv6.conf.all.forwarding = 1

さらに、pim6sd.conf の設定を以下のように行い、pim6sd を起動する。

```
phyint eth1 mld_version any;
phyint eth2 mld_version any;
phyint eth3 mld_version 1;
cand_rp
cand_bootstrap_router;
log all;
group_prefix ff4e::20/128;
```

phyint にてマルチキャストルーティングを有効にする物理インタフェースを指定し、MLD バージョ

ンを指定する。group_prefix を指定した場合には PIM-SSM として動作する。

pim6sd 起動後は、pim6stat コマンドを用いて pim6sd の状態を見ることができる。pim6stat コマンドを実行すると、/var/run/pim6sd.dump というファイルが生成され、このファイルに状態が記録されている。

```
Multicast Interface Table
 Mif   PhyIF Local-Address/Prefixlen                    Scope Flags
  0     eth1 fe80::250:4ff:feb7:481d/64                 3     DR PIM QRY
             2001:200:1b0:1000:250:4ff:feb7:481d/64     0
             Timers: PIM hello = 0:25, MLD query = 0:25
             possible MLD version = 1 2
  1     eth2 fe80::2c0:9fff:feed:358c/64                4     DR QRY NO-NBR
             2001:200:ffff::2/64                        0
             Timers: PIM hello = 0:25, MLD query = 0:25
             possible MLD version = 1 2
  2     eth3 fe80::213:ceff:fe45:afa2/64                5     PIM QRY
             2001:200:0:8410:213:ceff:fe45:afa2/64      0
             Timers: PIM hello = 0:25, MLD query = 0:25
             possible MLD version = 1
  3       lo ::1/128                                    0     DISABLED
             Timers: PIM hello = 0:00, MLD query = 0:00
             possible MLD version = 1
  4   regist fe80::250:4ff:feb7:481d/64                 3     REGISTER
             Timers: PIM hello = 0:00, MLD query = 0:00
             possible MLD version = 1

Multicast Routing Table
 Source          Group           RP-addr         Flags
-------------------------(*,*,RP)-------------------------
Number of Groups: 0
Number of Cache MIRRORs: 0

----------------------------RP-Set----------------------------
Current BSR address: 2001:200:ffff::2 Prio: 0 Timeout: 2
RP-address(Upstream)/Group prefix             Prio Hold Age
```

#### 2.4.4 IPv6 マルチキャスト経路表の設計

Linux IPv6 Multicast Forwarding によるマルチキャスト経路表の実装は、BSD 系と同じく、フォワーディングキャッシュを利用した実装となっている。通常の IPv6 ユニキャスト経路表において、ff00::/8 宛の経路の next-hop を :: とし、そのパケット処理関数の ip6_mc_input 内にてマルチキャストルーティングの処理を追加することで実現している。

USAGI プロジェクトでは、このマルチキャスト経路情報を、フォワーディングキャッシュとして持たせるのではなく、従来の IPv6 ユニキャスト経路表の枠組みを使って管理できるよう再設計しようと試みている。マルチキャスト経路の設定、削除も netlink socket を利用したものに変更する。しかし、この場合にも従来の setsockopt API も受け付けるよう実装することにより、従来の pim6sd といった経路制御デーモンを変更することなく利用できるよう設計する。

具体的には、経路表にマルチキャストグループを宛先とし、その next-hop を :: とした経路を生成する。この経路に対する input 関数を別途用意しその関数の中でグループに対する oif の管理を行うよう設計する。これによって、別途フォワーディングキャッシュを生成することなく、ユニキャスト経路とマルチキャスト経路を一元的にカーネル内で管理することが可能となる。

### 2.5 品質向上活動

#### 2.5.1 品質向上活動について

USAGI プロジェクトでは、Linux においてより良い品質の IPv6・IPsec プロトコルスタックおよび IPv6・IPsec に関するライブラリ・アプリケーションを提供する活動を続けている。本プロジェクトの活動の成果は Linux コミュニティに広く受け入れられている。例えば IPv6・IPsec のプロトコルスタックの改良コードは、メインラインカーネルに採り入れられており、また 2006 年度は Mobile IPv6 においても多くのコードが取り入れられた。

広く Linux コミュニティにコードが受け入れられるようになった現在、更なる開発活動に加え、既に開発したコードの品質維持・向上も重要である。この品質維持・向上に対する USAGI プロジェクトの活動をここに報告する。

#### 2.5.2 TAHI Automatic Running System

##### 2.5.2.1 TAHI Conformance Test を利用した Regression Test システムの開発

USAGI プロジェクトが貢献している Linux カーネルは、開発活動が活発に行われており、ネットワーク周りのコードに関しても日々数多くの新機能の追加、改良がなされている。この日々変更されるカーネルコードに対し、メンテナ及び多くの開発者はバグが混入しないよう目を光らせているが、変更によって副作用が生じる可能性は常に付きまとう。

そこで、USAGI プロジェクトでは毎日リリースされている Linux カーネルのスナップショットに対し、IPv6 プロトコルスタックにバグが混入していないか確認するための Regression Test システムを開発した。IPv6 プロトコルスタックのテスト自身には、TAHI プロジェクトの TAHI IPv6 Conformance Test Suite（`http://www.tahi.org/`）を利用している。この自動テスト実行システムにより、IPv6 プロトコルスタックにバグが生じた際に速やかに発見お

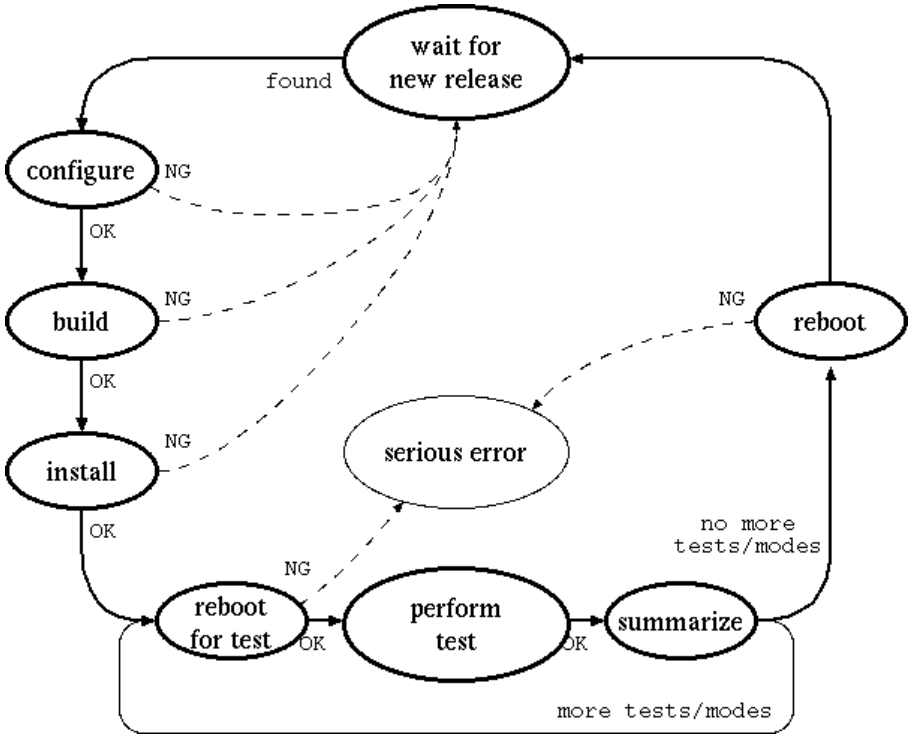



**図 2.3.** TAHI Automatic Running System の流れ

よび対処・修正ができる。

本システムは一般公開されており、http://testlab.linux-ipv6.org/および http://altair.linux-ipv6.org/ にて閲覧可能である。

#### 2.5.2.2 システムの流れ

システムは新しいカーネルリリースの待ち受け、ビルド、テストの工程を繰り返す。本システムではTAHI プロジェクトがリリースしている各 Conformance Test を使用することが可能である。また、一度ビルドしたカーネルに対し複数の Conformance Test を順次実行することも可能である。

システムの流れを図 2.3 に示す。システムはテストを行っていない間、新たなカーネルリリースを待ち受ける。待ち受けるリリース対象は、stable バージョンのほかに stable バージョンの準備段階である rc バージョン、および毎夜開発コードレポジトリより切り出してリリースされる git バージョンである。

新たなリリースを発見すると、システムは発見したカーネルをダウンロードし、ビルドを開始する。カーネルの configure、build、install は順次行われ、各過程のログは保管される。ログを参照することにより、ビルドエラーなどが生じた際にも解析することができる。各過程の処理が失敗に終わった際は、ソースに問題があったとし次のリリースを待ち受ける。

カーネルのビルドが終わると、システムはテストの実行に先立ち被テスト対象を再起動する。この再起動はビルドしたカーネルが正しく起動できるのか確認するために行われる。再起動時のログもテスト結果にあわせて保管される。再起動に失敗した場合、システムは一旦停止し、マニュアルでの確認を待つ。

ビルドが終了し再起動に成功すると、システムは TAHI Conformance Test を実行する。テスト終了後に結果を保管する際、システムは以前に行った実行結果との比較一覧を作成し別途保管する。この比較一覧の確認により、バグの混入が起こっていないか確認することができる。

TAHI Conformance Test が一つ終了した後、別の Conformance Test を実行するよう設定されていれば、システムは被テスト対象を再起動し、次に設定された Conformance Test を実行する。一旦システムを再起動するのは、前回のテスト時にカーネルが異常停止していないか検出するためである。再起動時のログは保管される。

カーネルへのテストが全て終わると、再び新たなカーネルリリースの待ち受け状態に戻る。この待ち

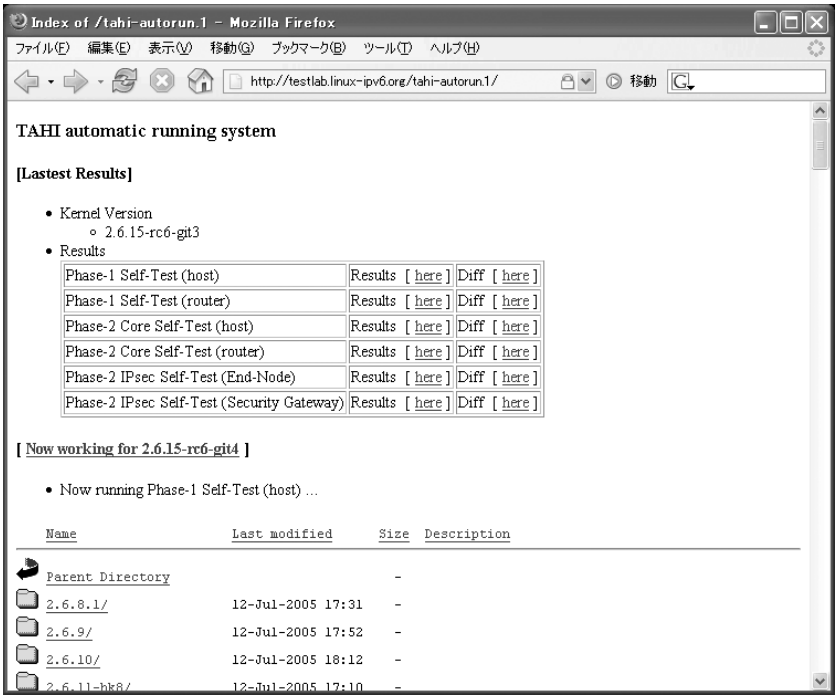



**図 2.4.**　ウェブブラウザよりのアクセス例

受け状態への移行に先立ち、被テスト対象は再起動し安定していると確認されているカーネルへと戻される。この再起動のログも保管される。

#### 2.5.2.3 データの収集および収集データの確認方法

システムは、テストの結果、前結果との比較一覧、テストしたカーネルのソースとそのコンパイル後のバイナリ、カーネルビルド等テストの各過程におけるログといったデータを収集する。収集したデータの総サイズはコンパイル時のカーネルオプション、実行した Conformance Test に依存するが、おおよそ一回当たり 450 MB となる。内訳はカーネルソースとバイナリで 300 MB、Conformance Test のログで 150 MB である。収集したデータはウェブブラウザから確認することができる。ウェブブラウザからのアクセス例を図 2.4 に示す。また結果比較の表示例を図 2.5 に示す。

#### 2.5.2.4 2006 年度の成果

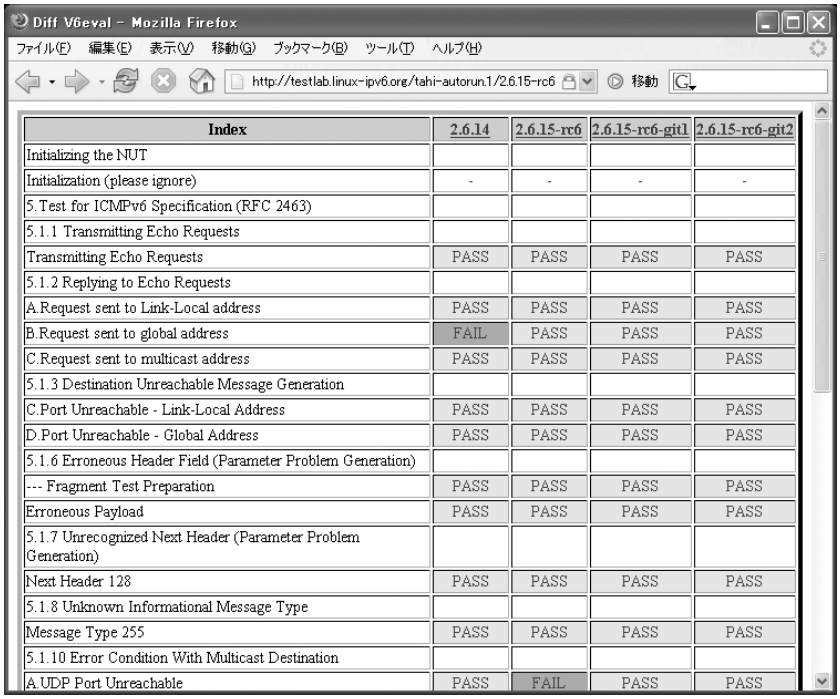
2006 年は IPsec において TAHI プロジェクトがリリースしている IPv6 Conformance Test Program Package のうちの IPsec に関するテストを実行対象に含めた。これは Authentication Header、そして IPv4 におけるバグの混入の監視依頼がコミュニティよりあったことに基づく。

また、Mobile IPv6 コードのカーネルへのマージを期に新たに Mobile IPv6 の Correspondent Node、Home Agent のテストを走らせることにした。しかし、既に走らせているテストの実行時間の総計が 24 時間を越えていたため Mobile IPv6 用に新たに 1 セット自動実行環境を用意した。

さらに、IPv6 Ready Logo Committee がこれまで分かれていた Phase-1 と Phase-2 の Core Protocols テスト仕様を統一したことに伴い IPv6 Ready Logo Phase-1 Conformance Test を実行対象より外した。

以上により現在、次のテストをシステム上で走らせ、結果を公開している。

- IPv6 Ready Logo Phase-2 Conformance Test (Host)
- IPv6 Ready Logo Phase-2 Conformance Test (Router)
- IPsec for IPv6 Ready Logo Phase-2 Conformance Test (End-node)
- IPsec for IPv6 Ready Logo Phase-2 Conformance Test (Security Gateway)
- IPv6 Conformance Test For IPv6 IPsec
- Conformance Test For IPv4 IPsec
- Test Suite for IPv6 Ready Logo Phase-2




| Index | 2.6.14 | 2.6.15-rc6 | 2.6.15-rc6-git1 | 2.6.15-rc6-git2 |
|---|---|---|---|---|
| Initializing the NUT | | | | |
| Initialization (please ignore) | - | - | - | - |
| 5. Test for ICMPv6 Specification (RFC 2463) | | | | |
| 5.1.1 Transmitting Echo Requests | | | | |
| Transmitting Echo Requests | PASS | PASS | PASS | PASS |
| 5.1.2 Replying to Echo Requests | | | | |
| A.Request sent to Link-Local address | PASS | PASS | PASS | PASS |
| B.Request sent to global address | FAIL | PASS | PASS | PASS |
| C.Request sent to multicast address | PASS | PASS | PASS | PASS |
| 5.1.3 Destination Unreachable Message Generation | | | | |
| C.Port Unreachable - Link-Local Address | PASS | PASS | PASS | PASS |
| D.Port Unreachable - Global Address | PASS | PASS | PASS | PASS |
| 5.1.6 Erroneous Header Field (Parameter Problem Generation) | | | | |
| --- Fragment Test Preparation | PASS | PASS | PASS | PASS |
| Erroneous Payload | PASS | PASS | PASS | PASS |
| 5.1.7 Unrecognized Next Header (Parameter Problem Generation) | | | | |
| Next Header 128 | PASS | PASS | PASS | PASS |
| 5.1.8 Unknown Informational Message Type | | | | |
| Message Type 255 | PASS | PASS | PASS | PASS |
| 5.1.10 Error Condition With Multicast Destination | | | | |
| A.UDP Port Unreachable | PASS | FAIL | PASS | PASS |

**図 2.5.** 結果比較の表示例

MIPv6 (Correspondent Node)

- Test Suite for IPv6 Ready Logo Phase-2 MIPv6 (Home Agent)

#### 2.5.2.5 今後の予定

今後、更なる改善・利便性向上において以下の項目を検討している。

- Mobile IPv6 において Mobile Node 試験のサポート
- よりよいユーザーインターフェースの検討
- ウェブページのみならずメールによる結果の通知

### 2.5.3 IPv6 Ready Logo

#### 2.5.3.1 IPv6 Ready Logo Program 参加の目的

IPv6 Ready Logo Program とは、国際認証機関である IPv6 Ready Logo Committee（`http://www.IPv6ready.org/`）により行われている国際的接続認証活動である。2006 年 12 月現在、IPv6 実装の基礎的な相互接続性を確認対象とした Phase-1 認証、実運用性を主眼としより高度な機能を確認対象とした Phase-2 認証が行われている。Phase-2 認証においては IPv6 の中心機能に加えて IPsec、MIPv6 などの認証も行われている。

USAGI プロジェクトにおいても、提供する成果物の信頼性の高さを示すため、この IPv6 Ready Logo Program に参加し、国際的接続認証の取得に努めている。

#### 2.5.3.2 相互接続性テスト自動化ツール実行環境

IPv6 Ready Logo Program におけるテスト内容には、RFC に準拠しているか確認するための Conformance Test と他のデバイスとの相互接続性を確認するための相互接続性テストがある。後者は長らく手動で行われていたが 2005 年 7 月 TAHI プロジェクトにより vel と呼ばれる相互接続性テスト自動化ツールがリリースされた。USAGI プロジェクトはこの相互接続性テスト自動化ツールを実行する環境を構築した。この環境のトポロジを図 2.6 に示す。

#### 2.5.3.3 2006 年度の成果

2006 年度 USAGI プロジェクトは前節の相互接続性テスト自動化ツール実行環境を構築した。この構築に伴い、多くの修正を相互接続性テスト自動化ツールに施した。いくつかの例を以下に示す。これら修正は TAHI プロジェクトにフィードバックした。

- シナリオコンパイルの高速化
- 対向機器における Solaris のサポート
- シナリオの不明瞭な点の修正

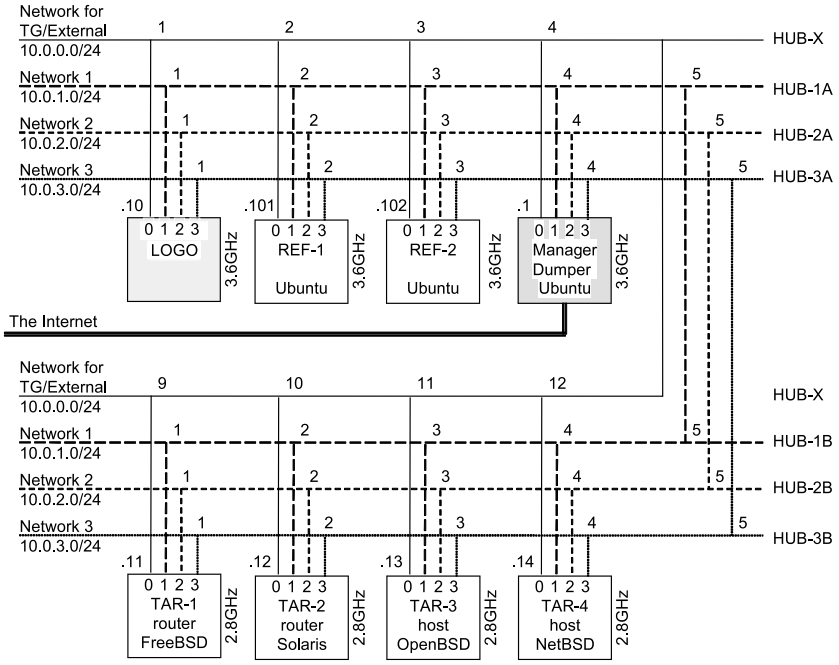



**図 2.6.**　相互接続性テスト自動化ツール実行環境のトポロジ

また USAGI プロジェクトは 2006 年 5 月、Linux のメインラインカーネルバージョン 2.6.15 において、IPv6 Ready Logo Phase-2 Core Protocol、Host 及び IPsec、End-node 認証を取得した。開発していた Mobile IP コードにおいてもロゴの認証を目指したが、欠陥が見つかり取得できなかった。こちらは今後の課題となる。

#### 2.5.3.4 今後の予定

今後、USAGI プロジェクトは品質を確たるものとするため更なるロゴの取得を目指す。

- IPv6 Ready Logo Phase-2 Core Protocol, Router
- IPv6 Ready Logo Phase-2 IPsec, Security Gateway
- IPv6 Ready Logo Phase-2 MIPv6, Correspondent Node
- IPv6 Ready Logo Phase-2 MIPv6, Home Agent
- IPv6 Ready Logo Phase-2 MIPv6, Mobile Node.

## 第 3 章　論文リスト

本年度の USAGI プロジェクトの論文リストを以下に示す。

(1) Shinta Sugimoto, Francis Dupont, Ryoji Kato, “Interactions between Mobile IPv6 and IPsec/IKE”, 情報処理学会論文誌, Vol. 47, No. 11, pp. 2967–2975, 2006.

(2) 吉藤英明, 関谷勇司, 高宮紀明, 宮澤和紀, 小堺康之, 知念充, 國武功一, “USAGI: IPv6 on Linux”, (株) アスキー, UNIX magazine 2006 年 7 月号, pp. 57–64, 2006/07.

(3) H. Yoshifuji, “Deploy A Real, Seamless IPv6 Ready Platform”, Linux Symposium 2006, Ottawa, July 2006.

(4) H. Yoshifuji, et al., “IPv6 Current Status and Next Steps”, Linux Kernel Developers' Netconf 2006, Tokyo, Japan, September 2006.

(5) H. Yoshifuji, “Current IPv6 Implementations and Future Perspective”, Taiwan and Japan Joint Advanced Networking Workshop on

IPv6, Hshinchu, Taiwan, November, 2006.
(6) 吉藤英明，小堺康之，中村雅英，“Linux Is IPv6 Ready——IPv6 はそこにある——”， Linux Conference 2006, 東京，2006.
(7) H. Yoshifuji, “IPv6 Stack Overview —Linux is IPv6 Ready—”, OSDL-Japan Linux Symposium, 東京，2006/06/13–14.
(8) H. Yoshifuji, “USAGI Project Experience”, OSDL-Japan Linux Symposium, 東京，2006/06/13–14.
(9) 吉藤英明，“Linux IPv6 スタックの研究開発”，2006 年度日本 OSS 貢献者賞（独立行政法人情報処理推進機構（IPA）主催）記念講演，IPA フォーラム 2006，東京，2006/10.