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し引き続き公開している。

2007 年度はメンバの 1 人が「NEA OSS Award」を受賞し、昨年に引き続き活動の成果が広く認められた年度となった。また 2008 年 3 月をめどとしたプロジェクトの完遂に向けての活動を進めていった。

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　2007 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（HUT）Go-Core プロジェクトと共同で 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 2006 年末まで、Mobile Ipv6 の基本仕様 [10, 78] 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 2007 年度の活動

MIPv6 に必要となる機能は、2006 年度までにほぼ Linux カーネルへのマージを完了したが、いくつか重要な機能が残っており、2007 年度はそれらのマージ作業を完了した。さらに、これまでマージされた機能は、Linux カーネルのバージョンアップに伴い、最新のカーネルで動作するようメンテナンスを実施した。

また、開発中のカーネルコードと、検証用のユーザランドの MIPv6 デーモンは随時 USAGI プロジェクトから公開した。

表 2.1. カーネルパッチ一覧

#	大項目	#	小項目	#	パッチ名	コード種別	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（2006）
				2	XFRM state use source address hash	new	required	CN HA MN	マージ済み（2006）
				3	XFRM bulk operation support	new	required	HA MN	キャンセル（2007）
				4	find IPsec header place to insert it with HAO/RT2	new	required	(CN) HA MN	マージ済み（2006）
				5	XFRM state inbound for mip6 exthdrs	new	required	(CN) HA MN	マージ済み（2006）
				6	update XFRM state last used timestamp	fix	optional	CN	マージ済み（2006）
				7	decide a device by source address of IPv6 header	new	optional	(HA) MN	キャンセル（2006）
				8	use mode as type even it was a flag	new	required	CN HA MN	マージ済み（2006）
				9	use acquire even inbound for RO trigger	new	optional	MN	キャンセル（2006）
				10	XFRM debug	new	optional		キャンセル（2006）
				11	Source address support for state id	fix	required	CN HA MN	マージ済み（2006）
				12	non-fragment protocol support	new	required	CN HA MN	マージ済み（2006）
				13	Sub policy support	new	required	CN HA MN	マージ済み（2006）
2	MIP6	1	HAO	1	HAO sending	new	required	MN	マージ済み（2006）
				2	HAO + ah6 sending	new	required	MN	マージ済み（2006）
				3	HAO receiving	new	required	CN HA	マージ済み（2006）
				4	BE report	new	required	CN HA	マージ済み（2006）
				5	TLV parser	new	optional	CN HA MN	マージ済み（2006）
		2	RT2	1	RT2 sending	new	required	CN HA	マージ済み（2006）
				2	RT2 receiving	new	required	MN	マージ済み（2006）
		3	MH	1	MH handling	new	required	CN HA MN	マージ済み（2006）
				2	MH sending	new	required	CN HA MN	マージ済み（2006）
				3	MH receiving	new	required	CN HA MN	マージ済み（2006）
		4	ICMP6	1	Swap HAO address before sending ICMP6 error	new	required	CN HA MN	マージ済み（2006）
				2	Swap RT2 address before sending ICMP6 error	new	required	MN	キャンセル（2006）
				3	Swap RT address with segment left field when receiving ICMP6 error	new	optional	CN	作業中
		5	—	1	mip6 debug	new	optional	CN HA MN	キャンセル（2006）
3	IPsec	1	MIGRATE	1	PF_KEY MIGRATE	new	required	HA MN	マージ済み（2007）
				2	MIGRATE extension (All upper layer protocol support, multiple bundle)	new	optional	HA MN	マージ済み（2007）
		2	Misc	1	SP selector ifindex	new	required	HA MN	キャンセル（2007）
				2	IPsec + TCP	fix	required	(CN) HA MN	マージ済み（2007）
				3	IPsec6 + RT sending	fix	required	(CN) HA	キャンセル（2006）
				4	IPComp and generic tunnel	fix	required		他の開発者によって解決済み（2006）
				5	Inbound block policy	fix	required		他の開発者によって解決済み（2006）
				6	allow to match wild-card user id at policy selector	fix	optional		キャンセル（2006）
				7	Decapsulated IPsec tunnel causes redirect	new	optional	HA	マージ済み（2007）
4	neighbor	1	proxy	1	proxy entry carries flag	new	required	HA	マージ済み（2006）
				2	don't do proxy forwarding when unicast ND	fix	required	HA	マージ済み（2006）
				3	don't do proxy forwarding to link-local destination	new	required	HA	マージ済み（2006）
				4	don't update neighbor cache when NA destined to proxied entry	fix	required	HA	マージ済み（2006）
				5	proxy NDP sysctl	new	optional	HA	マージ済み（2006）
		2	—	1	fix ndisc_flow_init to use ifindex	fix	unknown	(MN)	マージ済み（2006）
		3	—	1	fix fl6_merge_options	fix	required	CN HA	マージ済み（2006）
		4	RA	1	Use it as a default router in receiving RA	fix	required	MN	マージ済み（2006）
				2	Use it as a prefix in receiving RA	fix	required	MN	マージ済み（2006）
5	address	1	flag	1	HoA flag	new	required	MN	マージ済み（2006）
				2	never do DAD for HoA	new	required	MN	マージ済み（2006）
		2	lifetime	1	change address lifetime from user-land	new	required	MN	マージ済み（2006）
		3	prefix	1	add prefix info from user-land	new	required	MN	マージ済み（2007）
6	source address selection	1	—	1	source address selection: USAGI arch	new	required	MN	マージ済み（2006）
		2	—	1	source address selection: SUBTREE fix	new	required	MN	マージ済み（2007）
		3	HoA	1	source address selection: HoA support	new	required	MN	マージ済み（2006）
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（2006）
		2	SUBTREE	1	SUBTREE fix for policy routing	fix	unknown	HA MN	マージ済み（2007）
		3	—	1	removing netlink_skb_parms	fix	unknown		マージ済み（2006）
		4	anycast	1	anycast routing	fix	unknown	HA	他の開発者によって解決済み（2006）
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（2006）
		2	—	1	more optimized inet6_skb_parm	fix	optional		キャンセル（2006）
		3	ipv6 tunnel	1	ipv6 tunnel fix	fix	unknown	HA MN	マージ済み（2007）
9	NETFILTER	1	IPtables	1	MH match module	new	optional		マージ済み（2007）

2.1.2.1 カーネルにマージされた機能のメンテナンス

今年度、新たにカーネルにマージされた主な機能は、IPsec と MIPv6 が連携するための、トンネルモードの IPsec のエンドポイントを書き換える Migrate 機能と、MIPv6 に対応したファイヤウォールなどのための、MIPv6 制御メッセージをフィルタリングする MH match module である。

既にマージされた機能群は、カーネルのバージョンアップに伴って正常に動作することを確認し、問題が生じた場合には随時修正をするメンテナンス作業を継続している。

表 2.1 に詳細なマージ状況を示す。

2.1.2.2 UMIP の公開

MIPL2 の開発は HUT Go-Core プロジェクトと USAGI プロジェクトの協調体制のもと進められて来たが、HUT Go-Core プロジェクトの終了とともに、協調体制の見直しが必要となった。USAGI プロ

ジェクトでは、これまでの Go-Core プロジェクトメンバとの密な連携体制を継続することが難しいと判断し、ソースツリーのメンテナンスを USAGI プロジェクト独自で行う方針を採る決断を下した。これを受けて、2005 年 12 月時点での MIPL2 のスナップショットをもとにしたリリース（umip-0.1）および、2006 年 6 月時点の MIPL2（2.0.2）に基づくリリース（umip-0.3、umip-0.4）を公開している。umip-0.2 に関しては、内部的なバージョンアップに留まり、公開されることは無かった。

umip-0.4 からは、IKEv2 への対応を見据え、RFC3776（[10]）に従った IPsec の設定方法をやめ、RFC4877（[32]）に準拠した設定方法を採用した。

umip-0.4 の動作状況としては、Correspondent Node（CN）、Home Agent（HA）に関しては、基本動作に問題はない。Mobile Node（MN）は、一部に問題を残すものの、概ね正常に動作する。MN における主な問題は、同一リンク上での MN 間において、経路最適化パケットをデフォルトルータの MAC アドレスへ送信し、デフォルトルータ経由での通信となることである。

2.1.3 今後の取り組み

今後も、最新の Linux カーネルで動作する安定した MIPv6 スタックを継続的に提供していく。MIPv6 スタックは、カーネル内の MIPv6 スタックユーザ空間のデーモンプログラムから構成されるが、前者はすでにメインラインカーネルに組み込まれているため、ここでは実質的に後者のみを指す。

UMIP のソースツリーは、2005 年 12 月時点の MIPL2 のスナップショットより派生した UMIP-0.1、そして 2006 年 6 月時点の MIPL2（MIPL2.0.2）より派生した UMIP-0.3 および UMIP-0.4 がある。今後は、MIPL2.0.2 より派生したソースツリー上でメンテナンスを行っていく。

UMIP ソースツリーに施される修正は、UMIP 開発者による修正の他に、ユーザから提供されたパッチがある。ユーザから提供されるパッチに関しては、メーリング・リストに送付されたパッチを、UMIP 開発者がレビューし、その有用性および無害であることが確認された上でソースツリーに取り込む。

基本的に、これまで USAGI プロジェクト内で MIPv6 の開発に携わってきた開発者が継続してメンテナンスを行っていく。しかしながら、今後は USAGI プロジェクト外から有志のボランティアを受け入れていく体制を取っていく予定である。ただし、新たな開発メンバは MIPv6 および Linux カーネルの知識、そしてソフトウェア開発能力を備えた人物に限る。

リリースは、バグフィックス、新機能の追加等が施された際、適宜行っていく。リリースの間隔は特に決めていないが、これまでの実績では数ヶ月に 1 度の頻度でバージョンアップを行ってきている。

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、ip6tables 用拡張モジュールの大幅な拡充を行った。以下では、これらの内容について報告する。

2.2.2 2007 年度の開発内容

2.2.2.1 IPv4/IPv6 に対応した Connection Tracking のメモリ管理方法改善

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 1 機能であり、パケットフィルタ機能や IPv4 NAT 機能に利用される機能である。元々 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 におけるメモリ管理方法を改善した。以前から、IPv4 NAT やアプリケーションレイヤプロトコルの追跡などの拡張機能を適用しない場合に、nf_conntrack が各コネクション用に割り当てるメモリスペースの一部が使用されず無駄になることがあるという意見が多くあった。そこで、各コネクションに適用される機能に応じて適切なメモリスペースを割り当てることができ、かつ拡張機能を開発しやすい API を備えた nf_conntrack 用のメモリ管理機構を実装した。

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

従来 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た（それぞれ iptables、ip6tables）。しかし、これらは以下を扱う拡張モジュール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

そこで昨年度より、上記モジュールの実装に必要な内部 API 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った。また、iptables と ip6tables で別々に実装されていた 22 個のモジュールをそれぞれ統合した。

2.2.2.3 ip6tables 用拡張モジュールの大幅な拡充

iptables、ip6tables 用拡張モジュールの内部 API を共通化したことで、これまで iptables のみがサポートしていた拡張モジュールを ip6tables に対応させることが容易になった。そこで、そのようなモジュール 14 個を ip6tables に対応するよう変更した。

2.2.3 開発体制

昨年度に引き続き、USAGI プロジェクトメンバの小堺が 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 プロジェクト（`http://www.netfilter.org`）のコアメンバとして積極的に開発を進めた。

2.2.4 現在の状況と今後の予定

2.2.2.1 節で述べた nf_conntrack に対する改善は Linux 2.6.23 に採用された。また、2.2.2.2 節および 2.2.2.3 節で述べたユーザランドコマンド iptables、ip6tables に対する改善は iptables-2.4.0 に採用した。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。

近年 USAGI プロジェクトの活動により、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。

2.3 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 カーネル 2.6.24-rc5 に準拠している。

2.3.1 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ティング機能は、現在の Linux カーネル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、メインラインカーネル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 つがある。

- Linux IPv6 Multicast Forwarding (`http://clarinet.u-strasbg.fr/~hoerdt/dev/linux_ipv6_mforwarding/`)
- MRD6 (`http://unix.freshmeat.net/projects/mrd6/`)

Linux IPv6 Multicast Forwarding 実装は、Linux カーネル 2.6.7 に対するパッチ形式となっている。この実装は既に開発が停止しているが、USAGI プロジェクトメンバーの手によって、USAGI プロジェクトの開発 git ツリーに取り込まれており、現在 Linux カーネル 2.6.24-rc5 にて動作するよう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 は一風変わった実装であり、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読み取り、書き込みを行ってマルチキャストルーティングを実現している。debian にはパッケージも用意されており、カーネルを入れ替えることなく手軽にインストールならびに利用することのできる実装となっている。また、更新の頻度は高くないものの開発は継続されている。しかし、USAGI プロジェクトでは MRD 実装の検証は行っていないため、動作に関しては不明である。

また、IPv6 PIM ルーティングデーモンの実装は、次の 4 つである。

- pim6sd for Linux (`http://clarinet.u-strasbg.fr/~hoerdt/dev/pim6sd_linux/`)
- MRD6 (`http://artemis.av.it.pt/mrd6/`)
- XORP (`http://www.xorp.org/`)
- mcast-tools (`http://sourceforge.net/projects/mcast-tools/`)

pim6sd for Linux の実装は、前述の Linux IPv6 Multicast Forwarding カーネル実装に対応した pim6sd の実装である。BSD 用の pim6sd をもとに、Linux にて動作するよう改造を加えたものである。MLDv2/PIM-SSM にも対応している。しかし、ベースとなっている pim6sd は 2003 年当時の実装のままであり、本家の pim6sd の更新には追従しておらず、開発も止まっている。

MRD6 は前述の通りユーザランドで機能するマルチキャストルーティング実装であるため、PIM ルーティングデーモンもそのパッケージに含まれる。しかし動作は不明である。

XORP はルーティングプロトコルデーモンの集合パッケージであり、その中に PIM ルーティング機能も含まれる。しかし、XORP はあくまでもルーティングプロトコルの集合パッケージであるため、IPv6 マルチキャストルーティング機能の無い Linux カーネル上において動作させた場合には、PIM-SM ルーティングプロトコルをサポートするのみで、マルチキャストルーティングを行うことはできない。したがって、実質上 IPv6 PIM-SM ルータとして機能させることはできない。また、PIM-SSM もサポートされていない。

mcast-tools は主に BSD 用の IPv6 PIM-DM、PIM-SM デーモンを提供しているパッケージである。過去に Linux サポートも盛り込まれていたため、Linux 用コードも存在している。しかし、Linux サポートは 2006 年 7 月を最後に中断されており、現状そのままではコンパイルすることができない。

そこで USAGI プロジェクトでは、メインラインカーネルに IPv6 マルチキャストルーティング機能を統合すべく、前述の Linux IPv6 Multicast Forwarding 実装ならびに pim6sd for Linux、mcast-tools 実装をベースに、最新 Linux カーネルにて IPv6 マルチキャストルーティングを提供するための作業を 2006 年度から行っている。

2.3.2 IPv6 マルチキャストルーティング設定

本項では、Linux IPv6 Multicast Forwarding ならびに pim6sd for Linux および mcast-tools を用いた IPv6 PIM-SSM の設定例ならびに検証結果について示す。なお、最新 IPv6 Multicast Forwarding を含んだカーネルコードは、USAGI プロジェクトの git ツリーにて公開されている。また、このカーネルに対応した pim6sd は、その修正が完全に検証されていないため、未リリースである。修正が検証され次第、リリースする。

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
```

さらに、pim6sd.conf を以下のように設定し、pim6sd を起動する。

```
phyint eth1 mld_version any;
phyint eth2 mld_version any;
log all;
```

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。

図 2.1. 実験ネットワーク構成図

```
Multicast Interface Table
Mif   PhyIF   Local-Address/Prefixlen                    Scope   Flags
 0      eth0  fe80::213:72ff:fe3b:c1fb/64                2       DR PIM QRY
              2001:200:1b0:1000:213:72ff:fe3b:c1fb/64     0
              Timers: PIM hello = 0:15, MLD query = 1:50
              possible MLD version = 1 2
 1      eth1  fe80::213:72ff:fe3b:c1fc/64                3       DR QRY NO-NBR
              2001:200:1b0:fffe::2/64                     0
              Timers: PIM hello = 0:15, MLD query = 1:50
              possible MLD version = 1 2
 2        lo  ::1/128                                    0       DISABLED
              Timers: PIM hello = 0:00, MLD query = 0:00
              possible MLD version = 1
 3    regist  fe80::213:72ff:fe3b:c1fb/64                2       REGISTER
              Timers: PIM hello = 0:00, MLD query = 0:00
              possible MLD version = 1

PIM Neighbor List
Mif   PhyIF   Address                         Timer
 0      eth0  fe80::2000:1                    90
              2001:200:1b0:1000::2000:1

MLD Querier List
Mif   PhyIF   Address                         Timer      Last
 0      eth0  fe80::213:72ff:fe3b:c1fb             255       46s
 1      eth1  fe80::213:72ff:fe3b:c1fc             255       46s

Reported MLD Group
Mif   PhyIF   Group(Group-Timer,MLD-ver(Filter-Mode,Compat-Timer))/Source(TimerID)
 1      eth1  ff3e::4321:1234 (#0 (v2 IN #1024))
              2001:200:0:1c04:213:72ff:fe52:b05f (#19)

Multicast Routing Table
Source           Group           RP-addr          Flags
-----------------------------(S,G)-----------------------------
2001:200:0:1c04:213:72ff:fe52:b05f ff3e::4321:1234 NULL      CACHE SG
Joined        oifs:   ....
Pruned        oifs:   ....
Leaves        oifs:   .l..
Asserted      oifs:   ....
Outgoing      oifs:   .o..
Incoming         :    I...
Upstream    nbr:   fe80::2000:1

TIMERS: Entry=0 JP=60 RS=0 Assert=0
  MIF   0   1   2   3   4   5   6   7   8   9
    0   0   0   0   0
---------------------------(*,*,RP)--------------------------
Number of Groups: 1
Number of Cache MIRRORs: 1

----------------------------RP-Set----------------------------
Current BSR address: 2001:200:1b0:fffe::2 Prio: 0 Timeout: 25
RP-address(Upstream)/Group prefix            Prio Hold Age
2001:200:1b0:fffe::2(myself)
    ff00::/8                          0   150   145

--------------------CallOut Timer Queue-----------------
TimerID   Expiry-Time[s]
#20       5
#19       254
```

図 2.2.　pim6stat の結果

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。

動作検証は図 2.1 の構成にて行った。

ルーティングプロトコルは IPv6 PIM-SSM を用い、ssmpingd/ssmping ツールを用いて動作検証を行った。この際の Multicast Router（Linux）における pim6stat の結果を図 2.2 に示す。

図2.2の通り、mcast receiverは`ff3e::4321:1234`というマルチキャストグループにjoinし、そのマルチキャストグループのmcast senderは`2001:200:0:1c04:213:72ff:fe52:b05f`というホストであるということが認識されている。この状態でmcast receiverはマルチキャストパケットを受信できており、1台のLinuxルータを含む、3台のマルチキャストルータを経由したIPv6マルチキャストルーティングを検証することができた。

2.3.3 今後の方針

Linux IPv6 Multicast Forwarding ならびにmcast-toolsの動作検証ならびに改造を進め、安定した動作が行えるレベルまで達した段階で、一度メインラインカーネルへの統合を進める方針である。

メインラインカーネルに統合した後には、Linuxのカーネル構造に適したマルチキャストルーティング機構への改造を計画している。現在のLinuxにおけるIPv6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、改造することを目標とする。

2.4 可変的なアドレス選択ポリシの設計

2.4.1 ソケットAPIと自動端点選択

通信アプリケーションの作成に広く使われているソケットAPIでは、上位層が端点（アドレス、ポート番号）を詳細に指定することなく、接続を確立したり、メッセージを送信したりすることができる。

表 2.2. アドレス空間

プレフィックス	ラベル	注釈
`::1/128`	0	ループバックアドレス
`::/0`	1	デフォルト
`2002::/16`	2	6to4アドレス
`::/96`	3	IPv4互換アドレス
`::ffff:0:0/96`	4	IPv4写像アドレス
`fc00::/7`	5	ULA（RFC4193）アドレス（Linux拡張）
`2001::/32`	6	Teredo（RFC4380）アドレス（Linux拡張）

IPv6時代では、利用可能な端点の組合せが複数存在することが一般的であり、そのための基礎的な送信元及び宛先アドレスの決定[1]の方法がRFC3484 Default Address Selection for Internet Protocol version 6（IPv6）として規定されている[2]。

アドレス選択は、宛先アドレス選択（destination address selection）と送信元アドレス選択（source address selection）に大別される。

2.4.2 従来実装

Linux実装では、送信元アドレス選択はカーネルに実装されている。一方、宛先アドレス選択はC言語ライブラリであるglibcの`getaddrinfo(3)`関数に実装されており、必要な送信元アドレスに関する情報をカーネルから取得して行われる。

2.4.2.1 送信元アドレス選択

Linuxの送信元アドレス選択は、RFC3484に規定されたすべてのルールを実装している。アドレス選択ポリシテーブルはRFC3484で強く推奨されるような可変的なものとはなっていないが、RFC3484刊行後に定義されたアドレス空間をうまく扱えるように拡張されている（表2.2参照）。

実装では、効率化を特に重視している。送信元アドレス選択では、そもそも対象となるアドレスの種別やスコープの判定のためにアドレス種別判定関数である`__ipv6_addr_type()`を利用しているが、ラベル判定でもその結果を再利用している。

2.4.2.2 宛先アドレス選択

宛先アドレス選択に関しては、glibc-2.5以降が対応しており、ポリシテーブルの設定は、`/etc/gai.conf`

1　アドレス選択という。
2　IPv4でも同様の議論は可能であるが、基本的には同文書の範囲外である。

表 2.3. デフォルトの設定

プレフィクス	ラベル	注釈
::1/128	0	ループバックアドレス
::/0	1	デフォルト
2002::/16	2	6to4 アドレス
::/96	3	IPv4 互換アドレス
::ffff:0:0	4	IPv4 写像アドレス
fec0::/10	5	サイトローカルアドレス（拡張）
fc00::/7	6	ULA アドレス（拡張）
2001::/32	7	Teredo アドレス(拡張)

表 2.4. デフォルトの優先度

プレフィクス	優先度	注釈
::1/128	50	ループバックアドレス
::/0	40	デフォルト
2002::/16	30	6to4 アドレス
::/96	20	IPv4 互換アドレス
::ffff:0:0/96	10	IPv4 写像アドレス

ファイルによって行うことができる。カーネルからラベルに関する情報を得ることができないため、この設定ファイルには優先度に加え、ラベルに関する設定も記述される。また、設定ファイルは頻繁に変更されることを想定はしておらず、デフォルトでは最初の実行時に一度だけ読み込まれるだけである[3]。

デフォルトでの設定は表 2.3、表 2.4 の通りである。

2.4.3 Linux における可変的選択ポリシの設計と実装

従来の Linux 実装は、RFC3484 の最小限度のもので、強く推奨されている可変ポリシが実現されていなかった。このため、マルチプレフィックスでマルチホームの環境では問題が発生する場合があると指摘されていた [103]。

固定的な環境であれば、カーネルをコンパイルし直し、ユーザランド側の設定ファイルを設定することで原理的には対応可能であるが、すべての利用者にそれを求めることは現実的でなく、特に、DHCP によるポリシ配布を行なう場合などは致命的である。このため、Linux においても可変ポリシを実装することとした。

2.4.3.1 ポリシテーブル定義

アドレス選択ポリシテーブル検索のためのキーとしては、従来のプレフィクスに加え、出力インタフェースも使えるようにする。

得られるラベルは 32 ビットの非負整数値[4]、優先度は整数値とする。

優先度とラベルの機能分離を意識し、優先度はユーザ側で管理し、また、ラベルはカーネルで管理する。

2.4.3.2 カーネル実装

カーネル内において、ラベル情報はプレフィクスの長い順にソートされたリスト構造で管理される。これは、選択ポリシは現状ではそれほど大きくないと想定されるためであるが、仮に大きなポリシが必要となった場合には、より効率のよいデータ構造を導入することもできる。

リストの読み取りは効率よく行なえるようにすることが重要である。このため、RCU（Read-Copy-Update）による排他制御を採用した。また、従来の Linux 実装と同様、アドレス種別判定結果をできるだけ活用することとした。

また、更新回数を管理してユーザ空間に提供することで、ポリシテーブル解析の効率化を図っている。

IPv4 写像アドレスに対するラベル設定は、本質的に IPv4 アドレスに対するラベル管理として扱うべきものであるため、カーネル側では無効である。

2.4.3.3 インタフェース

カーネルとユーザ空間とのインタフェースとしては、`netlink(7)` を用いる。変更が加えられた場合にのみ結果を再評価すればよいようにした。

2.4.3.4 ユーザ空間実装

カーネル内の設定を操作する基礎的ツールとして、iproute2 を改変して用いる。

2.4.4 今後の展開

Linux における可変的な送信元アドレス選択の実現に関しては、カーネル部分は 2.6.25 でサポートされる見込みである。また、glibc での宛先アドレス選択における可変的な送信元アドレス選択との連携、`/etc/gai.conf` の書き換え中の問題回避策については、現在検討・実装中である。

3　設定ファイルに reload yes と記述することによって、実行毎に設定ファイルを再解析するようにできる。ただし、glibc-2.7 より前のバージョンには不具合があり、正しく動作しない。

4　0xffffffff は予約されており、利用できない。

これらの機能を活用することにより、DHCPv6 を用いたアドレス選択ポリシの配布 [49] などにも対応することができるようになる。

2.4.5 付録

2.4.5.1 KAME 実装

KAME 実装においては、プレフィクスに対するラベル及び優先度をカーネルに保持する。ラベルおよび優先度はともに整数である。この情報は `sysctl(2)` および `ioctl(2)` によって設定、取得することができ、操作のためのツールとして `ip6addrctl(8)` が提供されている。

`getaddrinfo(3)` は、カーネルからその都度ポリシを取得して動作する。

カーネルには RFC3484 のデフォルトポリシを保持しておらず、起動時に起動スクリプトによってユーザスペースから設定される。

2.4.5.2 Solaris

Solaris においては、プレフィクスに対するラベル及び優先度を設定することができる。ラベルは文字列（最大 15 文字）であり、優先度は非負整数である。操作のためのツールとして `ipaddrsel(1M)` が提供されている。

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 IPv6、IPsec、Mobile IPv6 のプロトコルスタックおよびライブラリ、アプリケーションの開発そして改善する活動を行ってきた。USAGI 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。

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後は、更なる開発活動に加え、すでに開発したコードの品質維持・向上も重要となった。そこで、USAGI プロジェクトでは毎日のスナップショットリリースに対し自動でテストを実行する TAHI Automatic Running System の開発や、第三者機関による品質評価を得るための IPv6 Ready Logo Program へ参加、そして IPv6 Ready Logo 相互接続性テスト自動化ツール実行環境構築など品質向上に対する活動も行ってきた。

本稿では USAGI プロジェクトの品質向上活動のまとめとして、IPv6 Ready Logo Program への取り組み、そして IPv6 Ready Logo Core Phase-2 Test Suite を用いて Linux メインラインカーネルの品質状況の推移を示す。

2.5.2 IPv6 Ready Logo

2.5.2.1 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。2007 年 12 月現在、IPv6 実装の基礎的な相互接続性を検証対象とした Phase-1 認証、実運用性を主眼としより高度な機能を検証対象とした 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、Mobile IPv6 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きた。

2.5.2.2 成果

USAGI プロジェクトは IPv6 Ready Logo Phase-2 のうち、Core Protocol の Router 認証を 2007 年 9 月に、IPsec の Security Gateway 認証を 2007 年 10 月にメインラインカーネルのバージョン 2.6.20 において取得した。

過去取得した IPv6 Ready Logo は以下の通りである。

- USAGI プロトコルスタック
  - 2.4 系
    [Phase-1, Host]
    2004/09/13: s20040705a-linux24
    2005/03/17: sV6READYP1-20050121_20050124-linux24
    [Phase-1, Router]
    2004/09/13: s20040705a-linux24
    2005/03/17: sV6READYP1-20050121_20050124-linux24
  - 2.6 系
    [Phase-1, Host]
    2004/02/26: s20040119-linux26
    2004/09/13: s20040705a-linux26
    2005/03/17: USAGI Stable Kit 6

[Phase-1, Router]
2004/02/26: s20040119-linux26
2004/09/13: s20040705a-linux26
2005/03/22: USAGI Stable Kit 6

- メインラインカーネル
  - 2.6 系

[Phase-1, Host]
2005/05/09: 2.6.11-rc2
[Phase-1, Router]
2005/05/09: 2.6.11-rc2
[Phase-2, Core, Host]
2006/05/30: 2.6.15
[Phase-2, IPsec, End Node]
2006/06/30: 2.6.15
[Phase-2, Core, Router]
2007/09/26: 2.6.20
[Phase-2, IPsec, Security Gateway]
2007/10/04: 2.6.20

2.5.3 IPv6 Ready Logo Core Phase-2 Test Suite で見る品質の推移

本項では TAHI プロジェクト（`http://www.tahi.org`）の提供する IPv6 仕様準拠テストスイート「Test Suite for IPv6 Ready Logo Phase-2 Core」を用い、2.4 系および 2.6 系カーネルの品質が向上していった様子を示す。使用したテストスイートのバージョンは実験時点（2007 年 12 月）において最新であった 1.4.9 である。

図 2.3 にホスト機能の品質推移を示す。続いて、

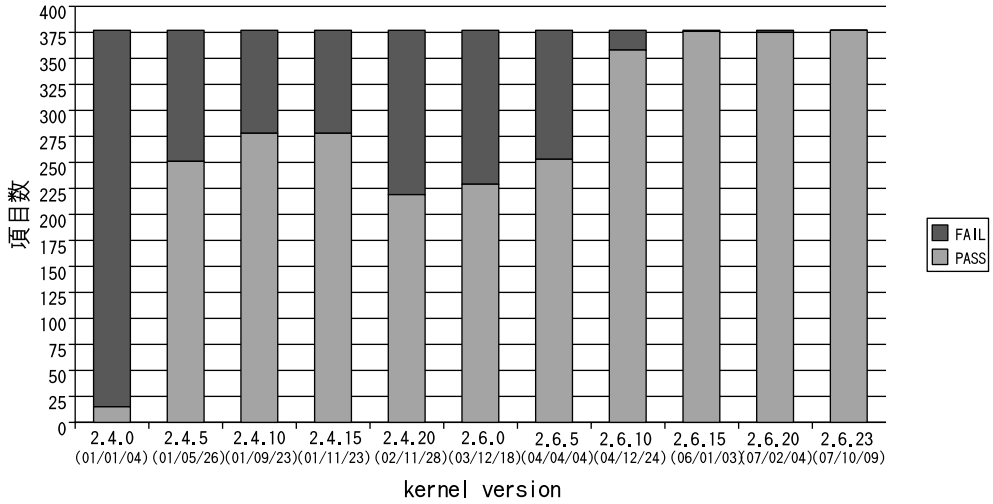

図 2.3.　ホスト機能の品質推移

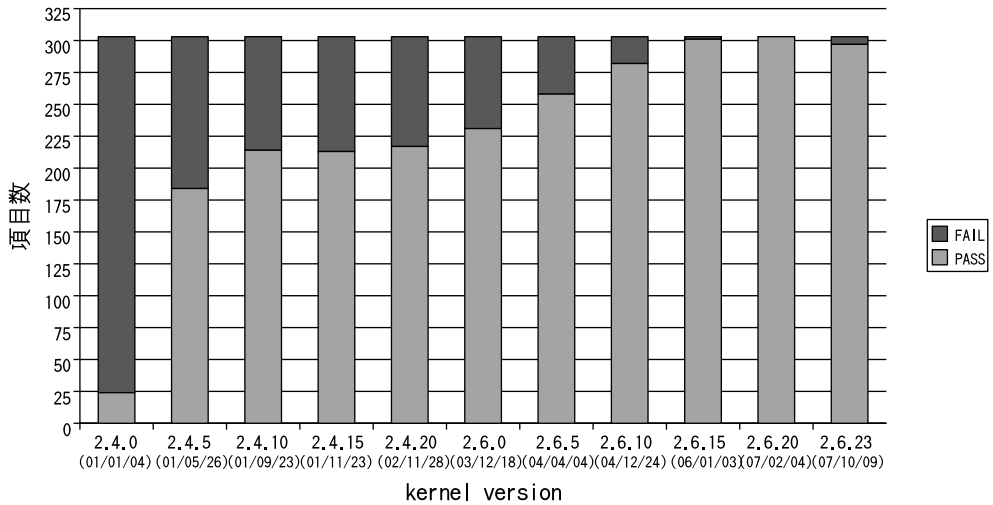

図 2.4.　ルータ機能の品質推移

図 2.4 にルータ機能の品質推移を示す。

なお、2.6 系カーネルにおいて、比較的最近まで FAIL が存在しているが、これはテストスイート更新に伴い新たに発見された FAIL であり、2.6.12 以降はその時点での最新のテストスイートの項目において FAIL はなかった。また、最新カーネルである 2.6.23 のルータ機能で FAIL が存在するのは type 0 routing header のセキュリティ問題にテストスイートよりも先に対応したためである。

2.5.4 まとめ

本報告書では USAGI プロジェクトにおける品質向上活動のまとめとして、IPv6 Ready Logo Program への取り組み、そして Linux の IPv6 プロトコルスタックにおける品質の推移を示した。活動を通して Linux の IPv6 プロトコルスタックは極めて良好となり、USAGI プロジェクトが目標として掲げていた “to deliver the production quality IPv6 and IPsec protocol stack for the Linux system” を充分に達成できた。

第 3 章　2007 年度対外発表

本年度の USAGI プロジェクトにおける対外発表を以下に示す。

【国際会議】

- Yasuyuki Kozakai, Hideaki Yoshifuji, Hiroshi Esaki, Jun Murai, “IPv6 Specific Issues to Track States of Network Flows”, ACM SIGCOMM 2007 Workshop IPv6 ’07, Kyoto, Japan, August 2007.

【講演】

- “Linux IPv6 Stack Development” H. Yoshifuji; Primera Cumbre Peruana de IPv6, Lima, Peru, May, 2007.
- “IPv6 Standard Application Programming” H. Yoshifuji; Primera Cumbre Peruana de IPv6, Lima, Peru, May, 2007.
- “Research and Development of Linux IPv6 Stack” H. Yoshifuji; The 6th North East Asia Open Source Software Promotion Forum, September, 2007.