Linux における IPv6 基本ソフトウェアの研究開発

吉藤 英明[1], 神田 充[2], 高宮 紀明[3], 関谷 勇司[4], 江崎 浩[1], 村井 純[5]

[1] 東京大学大学院情報理工学系研究科 [2](株) 東芝 研究開発センター [3]NTT ソフトウェア (株)
[4] 東京大学情報基盤センター [5] 慶應義塾大学環境情報学部

1 はじめに

1990 年代前半から IETF において進められてきた IPv6 [1] の仕様はこれまでにほぼ確定し，種々のネットワーク機器や，オペレーティングシステム（OS）を含むソフトウェアのベンダが対応を推進している．また，ISP(Internet Service Provider) による商用サービスも始まるなど，いよいよ実用段階に入ってきている．

Linux における IPv6 の歴史は古く，1996 年終わりには Pedro Roque による実装が Linux の公式配布物の開発版に登場している．

1998 年，日本における Linux IPv6 ユーザが集まって発足した Linux IPv6 Users Group JP では，Linux IPv6 プロトコルスタックの実運用に即した種々の検証が行なわれ，次第に種々の相互運用性等の面で問題があることが明らかとなった．具体的には，

1. 基本ソケット API におけるスコープの概念の欠如
2. 近隣探索及び静的アドレス自動設定の不具合 [2].
3. 必須とされる IP セキュリティ，モビリティ機能の欠如

などの不都合であり，登場以降，積極的に開発が行われてこなかった故の，機能の未実装，準拠仕様の古さ，及び，仕様準拠度の低さなどによると考えられた．

このような状況を打開するため，2000 年 10 月より，産学共同の WIDE プロジェクト [3] の支援の下，Linux IPv6 Users Group JP の主要メンバである筆者らを中心に，USAGI(Universal Playground for IPv6) プロジェクト [4] が開始された．

本稿では，筆者らが Linux における実装上の問題点を解決し，高い品質，即ち，IETF 規格への高い整合性を実現するために行なった基本アーキテクチャの説明と評価を行なっている [5, 6].

2 時間管理と排他制御ポリシ

近隣探索（Neighbor Discovery）[7] 及び静的アドレス自動設定 [8] は，IPv6 の最も特徴的な機構の一つである．前者はリンク層アドレス解決とその不到達性検出，ルータ探索とリダイレクトの機能を担い，後者はアドレスの重複検知を含む静的アドレス自動設定の機能を担っている．これらの機能の実現にはネットワーク上に起こる種々のイベントとタイマによる，多様な状態の正確な管理が必要である．

従来の Linux の実装においては，近隣キャッシュエントリの状態遷移やアドレスの有効期間の管理が，それぞれ 30 秒や 120 秒という時間スケールを基準としたタイマによるポーリングで実現されていた．このため，数秒以下の精度でのタスク処理が要求される近隣キャッシュエントリの状態遷移やアドレスの有効期間の管理がきわめて不正確なものとなっていた．

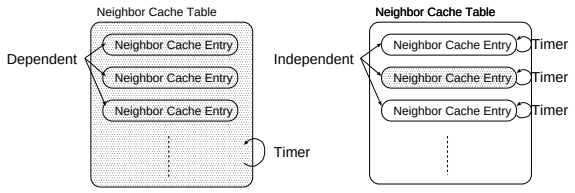

図 1: 近隣キャッシュの資源局所性（左:Linux, 右:USAGI）

この問題の解決方法としては，状態遷移処理を十分な精度で行なうよう，定期タイマ間隔を短くする方法がある．しかしながら，この手法では十分な精度を実現するためには常に十分な頻度で処理を繰り返さなければならず，また処理が空振りに終わる可能性が高いこと，個々の処理中にもそれらを保持するテーブル全体管轄する大域的な排他制御を行うことになること，などから，全体としても効率の低下を招くと考えられた．

USAGI 実装においては，近隣キャッシュエントリの状態遷移を整理し，エントリ毎にタイマを用意して，状態に応じたタイマを実行させることとした．これにより，個々のエントリ相互，およびその他の各種資源との資源独立性が高まり，大域的な排他制御を削減している（図 1）．

一方，アドレス管理に関しては，全アドレスで最も直近に評価すべき時刻を推定して次のタイマを決定することにより，アドレスの非優先化および無効化を確実に実現している．処理タスクの実行頻度は最頻でも 0.5 秒毎とすることにより，アドレス管理においてしばしば発生する，ほぼ同時に処理すべきアドレスが複数あった場合の負荷を軽減することができるようなシステムとした．

3 プロトコル処理におけるモジュールおよびインタフェースの再定義

近隣のノードの探索とその到達性の管理の概念は IPv6 固有ではないため，Linux においては，IPv6 の近隣キャッシュエントリ (NCE) と，IPv4 の ARP エントリ，DECnet の近隣エントリが，核となる単一の処理部によって扱われている．しかし，この統合は画一的で個々の詳細を考慮されていないため，近隣通知メッセージ（Neighbor

Advertisement）のように，メッセージに含まれるフラグや近隣の状態に依存した複雑な処理ができなかった．

USAGI 実装においては，NCE 更新処理を，その原理と前章の時間管理・排他制御ポリシにしたがった枠組として再設計した上で，処理関数の引数の意味を拡張して IPv6 近隣通知メッセージの形式などの詳細の特殊処理を派生させ，適切な近隣探索の状態遷移を実現した．

また，近隣探索メッセージ処理については，従来，処理方法が統一的でなく，安定運用に不可欠である正当性試験の多くが不十分であった．USAGI 実装においては，関係する種々の処理を共通部品化するとともに，受信処理部の構造を統一化して可読性を向上させ，正確で堅牢な正当性試験を実現している．

4 IPv6 への円滑な移行性の実現

IPv6 では，IPv4 からの移行技術の一つとして，IPv4 射影アドレスと呼ばれる機構を用意している．これは，IPv4 空間を広大な IPv6 空間の一部に射影するもので，アプリケーションは，IPv4 の通信相手を IPv6 のそれと同様に扱うことができる [9]．これにより，アプリケーションは IPv6 に対応するだけで IPv4 も簡単にサービスできる利点がある．しかし，単純なアドレスによるアクセス制限では従来の制御ファイルをそのまま流用できなかったり，アプリケーションやカーネルの処理が繁雑になってセキュリティ上の弱点が生じやすいという欠点がある．

従来，Linux は射影アドレスをサポートし，IPv6 と IPv4 でポート空間を完全に共有していた．このことは仕様上問題はなく，IPv4 パケットの配送が IPv4 ソケットの存在に左右される脆弱性もないが，射影アドレスの取り扱い自体が弱点となる可能性がある．実用上も，射影アドレスをサポートしない OpenBSD などを考慮して IPv4 と IPv6 のそれぞれにソケットを作成するようなアプリケーションや全く別のプロセスでサービスするようなアプリケーションを必ずしもうまく動作させることができなかった．

USAGI プロジェクトではセキュリティ上および利便性，従来の挙動との互換性の観点から，この挙動を設計した．具体的には，IPv4 射影アドレスを無効にするための IPV6_V6ONLY ソケットオプション [10] を導入し，IPv6 ソケットに対してこれが設定されている場合に限り IPv4 ソケットとの共存を許すものとするように拡張した．これにより，当該ソケットオプションを設定しない従来の Linux アプリケーションのなどとの互換性を維持しつつ，特にアクセス制御などを重視するアプリケーションもこのソケットオプションを利用することで実現可能となった．

5 IP セキュリティの実現

IP セキュリティ（IPsec）[11] の機能は，IPv6 仕様上必須とされているにもかかわらず，カーネル配布物の中に IPsec の実装は含まれていなかった．一方，IPv4 については FreeS/WAN プロジェクト [12] がその実装を独自に提供し，IPv6 についても，この FreeS/WAN プロジェクトのソースコードを一部利用していたものが存在していただけであった[1]．

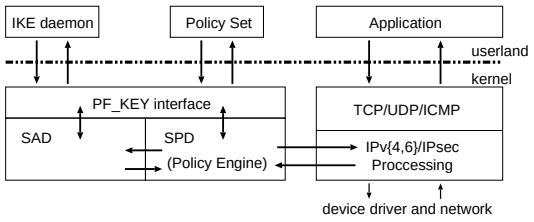

図 2: USAGI IPsec 実装の構造

FreeS/WAN プロジェクトの実装では IPsec のアーキテクチャーの中で IP バージョンに関わらず共通化できる部分である SAD(Secutirty Association Database) や SPD(Security Policy Database) などが強く IPv4 に依存している．さらに，パケット処理部は，IP 層（ネットワーク層）とデータリンク層の間に BITS(Bump in the Stack) と呼ばれる手法を適用した，仮想デバイスによるトンネリングとして実装されている．この手法では，IP 層に関する重複した実装が必要となるほか，たとえば TCP のポート番号といった，より詳細な制御ポリシ実現が困難であり，またパケットの転送効率が低下が発生してしまう．このため，USAGI プロジェクトでは本章で提案している構造での実装を行った．図 2 にその構成を示す．

5.1 IP バージョン非依存なデータベースの実現

IPsec 機能の中で SAD と SPD は IP バージョンには依存しない．このため，USAGI 実装においては，汎用ソケットアドレス構造体 **sockaddr_storage{}** を用いた，IPv4 と IPv6 で共用可能な SAD と SPD を実現している．

5.2 暗号/認証アルゴリズム

IPsec で使用する暗号化や認証のためアルゴリズムは IPsec 固有のものでなく一般的なものである．そのため，USAGI プロジェクトでは暗号化/復号化関数等のインタフェースを抽象化する cryptoapi と呼ばれる実装 [13] をアルゴリズムの実装に利用している[2]．これにより，様々なアルゴリズムを容易に追加削除することができ，また，種々の暗号輸出規制などが存在する国においても，アルゴリズムだけを除外して配布することが容易に実現できる．

5.3 カーネルインタフェース仕様

PF_KEY は，各種の秘密鍵及び認証鍵を設定，管理するために用いられるユーザ空間のアプリケーションとカーネル間のインタフェースであり，その仕様（バージョン 2）は RFC2367 [14] に定義されている．しかし，この文書に定義されている仕様は基本的なものだけで十分といえ

[1] これは、米国の暗号輸出規制が影響していたことは否めない．

[2] USAGI では des-cbc，3des-cbc 及び aes-cbc の各暗号アルゴリズム，hmac-md5 と hmac-sha1 の認証アルゴリズムが利用可能．

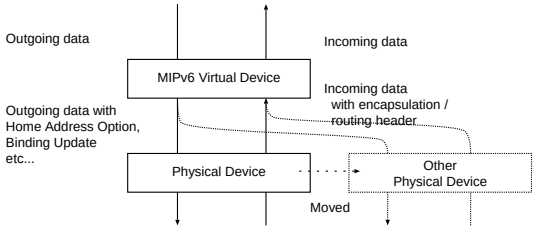

図 3: 仮想デバイスを用いたモバイル IP の実装

ず，実装ごとに独自の拡張がなされている．一方，IPsec において実際に各種のセキュリティポリシなどを設定，管理するためのインタフェースも必要であり，FreeS/WAN プロジェクトの実装では PF_KEY を拡張することでこれを実現している．

USAGI 実装では FreeS/WAN プロジェクトによる拡張部分の定義を利用することで，ユーザ空間のアプリケーションの互換性を保つようにしている．

6 モビリティ機能の実現

モバイル IPv6 は，IPv6 で注目されている機能の 1 つであり，IPv6 の機能を活かし，外部エージェント（Foreign Agent）を省略した通信が可能である．

IPv6 におけるモビリティの実現は，USAGI 実装では，ヘルシンキ工科大学（HUT）によるモバイル IPv6 の実装を取り入れ，USAGI 実装が基礎をおく最新カーネルでの動作が可能となっている．一方で，HUT の実装を取り入れた結果，以下の問題項目が顕在化した．

1. パケット送信時におけるソースアドレス選択法
2. 移動ノード上のホームアドレス（Home Address）設定
3. 移動ノードの位置管理等を司るホームエージェントを自動探索する機能である，ダイナミックホームエージェント探索（Dynamic Home Agent Discovery）
4. Netfilter および IP セキュリティ機能との相互運用性
5. モバイル IPv6 起動時の初期化

移動ノードがパケット送信する際は，特にアドレスのスコープに注意して適切なソースアドレスを選択しなければならない．USAGI 実装ではスコープとホームアドレスを共に意識した新しいソースアドレス選択手法 [15] を実装して運用互換性を高めている．

ホームアドレスの設定は，現在ネットワークに接続している物理デバイスに直接設定するため，デバイスをまたがった移動透過性に欠ける．USAGI ではこの問題点を仮想デバイス（Virtual Device）を用いた解決を検討している．例えば，仮想デバイスを導入することによりホームアドレスの管理を物理的なデバイスから分離し，デバイス間で透過的な移動が可能となると考えられる（図 3）．

ダイナミックホームエージェント探索では移動ノードが送信する探索要求メッセージの宛先アドレスとしてエニキャストアドレスが利用される．現在の Linux ではエニキャストをサポートしていないため，HUT 実装では通常のユニキャストアドレスのようにエニキャストアドレスを割り当てることで対応している．USAGI 実装ではエニキャストをサポートし，その枠組の中でダイナミックホームエージェント探索を実現する．

HUT 実装は Netfilter の機能を利用してパケットの転送の制御などを行っているが，処理の流れが複雑で紛らわしい問題がある．また，通常のパケットフィルタリング等との機能と併用したときの動作に疑問がある．USAGI プロジェクトではこれらの機能との共存ができるよう，Netfilter との依存性を排して処理を簡素化し，また IP セキュリティ機能との相互作用も考慮したモバイル IPv6 の機能の実装を行なっている．

7 スコープの概念の実現

基本ソケット API 上におけるスコープの概念への対応は，Linux IPv6 Users Group JP の成果が Linux 実装に採用されている．この実装では，互換性に配慮して，新旧のソケットアドレス構造体に対応しつつ，リンクローカルアドレスのスコープが指定可能となった．

しかしながら、この実装はリンクローカルアドレスに限られており、パケット転送もサイトローカルの概念に対応していなかった．USAGI 実装では，経路表を用いたより完全なスコープの概念への対応を図るとともに，アプリケーション層にまでわたる総合的なスコープの概念への対応法を設計している．

8 USAGI プロトコルスタックの評価

USAGI プロジェクトによる IPv6 基本ソフトウェアの評価を，試験対象となる実装が仕様に照らして正当に動作するものであるかどうかを評価するシステムである TAHI 仕様適合性評価システムを用いて行なった．

表 1 に，本稿の論点である近隣探索（Neighbor Discovery），およびルータ探索・アドレス自動設定（Stateless Address Autoconfiguration）についての試験結果を示す．

試験は，USAGI プロジェクトによる IPv6 基本ソフトウェア（“USAGI”）のほか，オリジナルの Linux 配布物（“Linux”）を対象として行い，結果は，その分野に属する試験項目に対する合格率によって示した．また，参考のため，KAME プロジェクト [16] による実装（“KAME”）に対して行われた試験結果 [17] も示した．

この結果から，USAGI プロジェクトによる IPv6 基本ソフトウェアは，その仕様適合性の面でオリジナルの Linux 配布物に比べて顕著な改善が実現され，高品質として知られる KAME プロジェクトによる実装により近づいたものになっていることがわかる．

また，USAGI プロジェクトの成果は次第にオリジナルの Linux 配布物にも次第に反映されており，新しい版のオリジナル Linux 配布物では仕様適合性が改善されている．

表 1: TAHI 仕様適合性試験結果（合格率，抜粋）

試験	Linux		USAGI	KAME
分野	2.2.15	2.4.18	2.4 系	FBSD4.4 系
ND	34%	39%	89%	98%
SAA	4%	77%	98%	98%

表 2: Flood Ping による性能比較結果（単位：ミリ秒）

実装	最小	平均	最大
Linux-2.4.18	0.376	1.025	12.497
USAGI 2.4 系 (2002/2)	0.389	0.960	13.65

表 2 に，USAGI プロジェクトによる IPv6 基本ソフトウェアとオリジナルの Linux 実装の性能を示す．値は，Pentium II 400MHz プロセッサを搭載した測定対象機に対し，**ping6** コマンドに **flood** オプションを付加して 10 万パケットを連続して送信した際の，応答時間（3 回測定平均）である．この結果から，USAGI プロジェクトによる IPv6 基本ソフトウェアの実装では，時間精度を大幅に向上させたにもかかわらず，ほぼ同等の性能を示すことがわかる．また，第三者による試験では，処理能力の低い CPU の環境において，USAGI による IPv6 基本ソフトウェアが，オリジナルの Linux 実装に比べて約 60% 程度の負荷で動作することが報告されている [18]．以上のことから，USAGI プロジェクトによる IPv6 基本ソフトウェアは総じてより効率のよいものであるといえる．

9 まとめ

本稿では，USAGI プロジェクトによる IPv6 基本ソフトウェアについて，その特徴的なアーキテクチャをいくつか取りあげた．USAGI ソフトウェアは，近隣探索プロトコル，IP セキュリティ，モビリティといった実現において，IPv6 の機能をより自然，広範かつ安定に動作するための新しいソフトウェアアーキテクチャを提案し実装している．これらの成果は汎用な形態で公開されており，種々のシステム構成上で利用可能である．

USAGI プロジェクトの成果は Linux 配布物にも開発版を中心に次第に反映されている．IPsec の実装も，提案を契機に議論が開始され，開発版に IPv4 用の実装が含まれている．

最後に，今後の課題を挙げる．

1. Linux 実装の議論を反映した IPv6 用 IP セキュリティ機能の実装と提案．
2. 特に IP セキュリティ，モビリティ等の機能の統合いっそう進めた，概念的に自然かつ効率的なソフトウェア構造の実現
3. サイトの概念を含む，スコープの扱いの一層の改善．
4. セキュリティ機構，モビリティ機構とフィルタリング機構との協調．
5. マルチキャスト経路制御への対応

以上のような基礎・応用両面にわたる検討をもとに，より洗練され完成度の高い実装を広く提供して IPv6 の普及を図り，また標準化活動にも貢献する予定である．

参考文献

[1] S. Deering and R. Hinden, “Internet Protocol, Version 6 (IPv6) Specification,” RFC2460, 1998.

[2] 福原一朗, 国武功一, 吉藤英明, 岡村耕二, 楠根雄志, 関谷勇司, “Linux でつなぐ IPv6 の世界,” Linux Conference ’99, 1999.

[3] WIDE Project, <http://www.wide.ad.jp>.

[4] USAGI Project, <http://www.linux-ipv6.org>.

[5] 吉藤英明, 神田充, 高宮紀明, 関谷勇司, 江崎浩, 村井純, “USAGI プロジェクトによる IPv6 基本ソフトウェアの開発,” 電子情報通信学会論文誌 B, Vol. J85-B, No.8, pp. 1139-1346, 2002.

[6] Y. Sekiya, H. Yoshifuji, M. Kanda and K. Miyazawa, “Evaluation and Improvement of IPv6 Protocol Stack by USAGI Project,” Proceedings of Ottawa Linux Symposium 2002, pp.496-504, 2002.

[7] T. Narten, E. Nordmark and W. Simpson, “Neighbor Discovery for IP Version 6 (IPv6),” RFC2461, 1998.

[8] S. Thomson and T. Narten, “IPv6 Stateless Address Autoconfiguration,” RFC2462, 1998.

[9] R. Gilligan, S. Thomson, J. Bound and W. Stevens, “Basic Socket Interface Extensions for IPv6,” RFC2553, 1999.

[10] R. Gilligan, S. Thomson, J. Bound and W. Stevens, “Basic Socket Interface Extensions for IPv6,” draft-ietf-ipngwg-rfc2553bis-10.txt, 2002.

[11] S. Kent, R. Atkinson, “Security Architecture for the Internet Protocol,” RFC2401, 1998.

[12] FreeS/WAN Project, <http://www.freeswan.org>

[13] International Kernel Crypto API for GNU/Linux, <http://sourceforge.net/projects/cryptoapi/>.

[14] D. McDonald, C. Metz, B. Phan, “PF_KEY key Management API, Version 2,” RFC2367, 1998.

[15] R. Draves, “Default Address Selection for IPv6,” draft-ietf-ipngwg-default-addr-select-06.txt, 2001.

[16] KAME Project, <http://www.kame.net>.

[17] TAHI Project, “TAHI Conformance Test Report: kame-20011105-freebsd44-snap,” <http://www.tahi. org/report/KAME/freebsd44-snap-20011105/>, 2001.

[18] J. Koskelainen, “Performance of Usagi vs. Linux IPv6?,” <http://www.linux-ipv6.org/ml/usagi-users/ msg01134.html>, 2002.