WIDE Technical-Report in 2008

USAGIプロジェクト活動の歴史と総括

wide-tr-usagi-history-00.pdf

WIDE Project : http://www.wide.ad.jp/

If you have any comments on this document, please contact to ad@wide.ad.jp

Title: USAGI プロジェクト活動の歴史と総括
Author(s): USAGI プロジェクトコアメンバ（usagi-core@ml.linux-ipv6.org）
Date: 2008/12/15

本文では、USAGI プロジェクト発足以前から USAGI プロジェクト完了までの活動の歴史 を、成果別の観点から分類し、まとめる。

USAGI プロジェクト発足以前

USAGI プロジェクトが発足する以前に、Linux での IPv6 に関するアクティビティ として、Linux IPv6 Users Group が存在した。その名前の通り、Linux で IPv6 を利用しようとしているユーザの相互互助を目的とした団 体だ。当時は、設定方法が不明なだけでなく、Linux そのものに実装上の問題もあり、ユーザとしての活動だけでは、IPv6 環境での Linux 利用に限界が見えていた。そのため、Linux IPv6 Users Group 内での議論、知見を元 に、USAGI プロジェクトの必要性が考えられるようになり、本プロジェクトの発足に繋がって行くことになる。

すなわち、Linux IPv6 Users Group がなければ、USAGI プロジェクト発足は大幅に 遅れるか、もしくはプロジェクト自体が存在しなかったことも考えられる。そこで本章では、USAGI プロジェクトの基盤となった Linux IPv6 Users Group につ いて触れる。

Linux IPv6 Users Group

Linux IPv6 Users Group では、Linux で IPv6 を使うための情報源として、Web ページを開設していた。http://www.v6.linux.or.jp/ である。Linux での IPv6 に関するインストール、各種設定、関連リンクなどがまとめられていた。さらに、Linux IPv6 JP ML という、メーリングリストも開設されていて、Linux での IPv6 に関する話題が話し合われていた。さらに、実験用 IPv6 ネットワー クの接続も提供しており、Linux を使っているユーザであれば、簡単な申請に よって、同ネットワークに接続した上で、実際に IPv6 を使ってみるといった ことも可能であった。また、当時はまだ、ウェブ、メール、FTP などの基本的なアプリケーションも 正式に IPv6 に対応していないものが多かったため、Linux IPv6 JP の活動の一環として、次に挙げるものの実験などを行なっており、黎明期における Linux での IPv6 普及に貢献した。

IPv6 による ML の配送

Linux IPv6 JP ML の配送を IPv6 で行うことを目的としていた。IPv6 対応の sendmail によるメールの配信実験は何回か行われており、成功している。

IPv6 での Web / anonymous FTP サービス

これは当時サービスを提供していた http://www.v6.linux.or.jp/ , ftp://ftp.v6.linux.or.jp/ ともに IPv6 によるアクセスが可能としていた。ともに 1 日に数件の IPv6 によるアクセスが記録されている。特に anonymous FTP サービスは、海外からの IPv6 アクセスが目立つのが特徴だった。

IPv6 対応アプリケーション開発

IPv6 対応アプリケーションを移植・開発していくことを目的とする。いくつか のアプリケーションが移植された。

インストール・設定文章の充実

IPv6 関連のインストールや設定の文書を充実させることを目的としていた。当時は、日本語での IPv6 自体の解説

記事もまだ極めて少なかったため、Linux 固有の話だけでなく、基礎的な IPv6 に関する情報も日本語にて提供していた。また、当時 IPv6 の接続方法として主流であったトンネリングによる設定方法 も記載していた。

USAGI プロジェクトの発足

USAGI プロジェクトは、Linux における IPv6 環境を向上させるために、2000 年 10 月に発足されたプロジェクトである。当時、すでに数多くの OS や商用ルータに IPv6 スタックが実装されていたが、Linux における IPv6 に関する状況は、 他の OS に比べて劣っていた。従っている仕様も古く、正確に動かない機能や、実装されていない機能が存在しており、実用に耐えないものとなっていた。そこで、この情報を打破すべく、USAGI プロジェクトによる改善が開始されるこ ととなった。

Linux における IPv6 プロトコルスタックの歴史

Linux における IPv6 の歴史は古く、linux-2.1.8 から IPv6 スタックが実装されている。これは他の OS に比べても早い時期に実装されており、Linux はいち早く IPv6 が利用できる OS であった。しかし、それ以来 IPv6 のコード はメンテナンスされておらず、現状では他の OS と比べ、かなり遅れた実装となっていた。

もともと、Linux の IPv6 スタックは、Pedro Roque という人物が、IPv6 の制定初期段階に実装したものである。この実装をもとに、A.N.Kuznetsov や Andi Kleen といったネットワークコードの管理者が改良やバグフィックスを 行なっていた。当時の安定版カーネルであった linux-2.2 系 や linux-2.4 系にも IPv6 スタックが組み込まれている。ただし、linux-2.2 系には、構造体 sockaddr_in6 に含まれていなくてはならない sin6_cope_id が存 在していない等、IPv6 を使う上で問題点が多かった

USAGI プロジェクト発足以降、linux-2.4 系のカーネルでは、徐々に USAGI プロジェクトによる改良が取り込まれつつあったが、完全には当時の RFC に追従しておらず、他の OS との相互接続性問題や機能の欠落といった問題が存在し ていた。

そこで、後述する TAHI プロジェクトの協力の元、自動テスト環境を構築し、仕様、 堅固性などを改善していくことになる

IPv6 コアプロトコルスタック改善に関する活動

プロジェクト開始時、Linux IPv6 スタックの品質の改善は大きな課題であった。品質上の問題は、TAHI テストの結果が悪いだけでなく、ちょっとしたきっかけでカーネルが動作を停止するなどの問題も抱えていた。

重篤な問題は発見するたびに修正を行ない、メインラインへの反映を行なった。一方、実装の理解が進むにつれ、ソケットオプションや近隣探索の処理において、正当性/健全性のチェックがかなり欠落しており、標準仕様に合致しない原因となっていることがわかった。様々な処理が場あたり的にされていることが 多く、処理の統合などの整理が進められた。また、カーネル内部で用いられていた独自の構造体定義などは理解を妨げるとして整理などを進めた。

これらの修正に並行して、ユーザ環境と管理機能としてアドレスやルータ選択の改善、プライバシ拡張(Privacy Extensions)、Node Information Query やインタフェース毎統計情報の対応、プログラミング環境の改善として、RFC2553 や RFC2292bis 機能の実装、IPv6 対応の拡大として、khttpd (カーネル内部 HTTP デーモン) や ARCnet の IPv6 対応などが進められた。特に RFC2553 (Basic Socket Interface Extensions for IPv6) 対応では、IPV6_V6ONLY ソケットオプションの対応に際して、bind(2) の挙動を網羅的に検討／再設計を行なわれた。

2001 年には、近隣探索やアドレス自動設定機能の処理に大改造を行ない、時間管理精度、状態管理の適正化とあいまって、TAHI 試験の結果が大幅に改善した。しかし、大改造の代償として、メインラインへのマージ作業が困難になった。

2002 年後半より、これまでに開発／改善してきたコードのメインラインへの統合作業が本格化し、アドレス自動設定が大幅に改善し、近隣探索も改善した。

2003 年には 2.6 に向けた改善が本格化した。その過程で発生したいくつかの 問題、例えば、パケットの出力処理の変更(ip6_append_data 化)に対応した。一方、USAGI 頒布物むけに大きな改造を加えた近隣探索については、再度整理・検討しつつ投稿し、メインライン頒布物に反映させていくなどして、状況を好転させるよう注力した。さらに、カーネル全般に関しても、関与をし始めた。これらの結果、2003 年 5 月、MAINTAINERS に加入することになった。

2003 年後半には、Linux 2.6 系列の USAGI 頒布物の品質は、IPv6 Ready Logo (Phase 1)をホストおよびルータの分野で取得できる程度にまで高められ、2004 年 3 月、1 月に行なわれた TAHI イベントの結果で正式にロゴを取得した。(2004 年夏には 2.4 系列でも取得している。)

2005 年には、経路の時間管理の改善などにより、ホスト機能の改善は一段落した。その後は、品質に関しては主にルータ機能の改善に注力した。特に、ルータ広告デーモンの改善などが課題であった。

2005 年末までの成果で、2006 年、2.6 ベースの USAGI 頒布物(*)が IPv6 Ready Logo (Phase-2; Host)を取得し、また、その成果を反映後、2006 年、メインライン(2.6.15)が IPv6 Ready Logo (Phase-2; Host, End-Node)を取得した。また、いくつかのルータ広告デーモンとカーネル内部パラメータとの相互作用について修正するなどして、2007 年、メインラインカーネル(2.6.20)とルータ広告デーモン(1.0)の組合せで、IPv6 Ready Logo (Phase-2; Router, SGW)を取得した。

IPsec スタックに関する活動

IPv6 において、IP Security (IPsec)は必須機能として定義されている。2000 年、USAGI プロジェクトが発足した当時の Linux カーネルには、IPsec スタックが実装されていなかった。Linux 向けの IPsec スタックとして、FreeS/WAN プロジェクトによるスタックと鍵交換プロトコルが提供されていたが、メインラインカーネルへは取り込まれていなかった。

FreeS/WAN のスタックは、仮想ネットワークデバイスを用いた設計となっており、IPsec で言うところの bump-in-the-stack アーキテクチャであった。主に VPN として利用することをターゲットとし、そのスタックは、IPv4 のみをサポートし、IPv6 をサポートしていなかった。一方で、鍵交換プロトコル Internet Key Exchange (IKEv1) をサポートする FreeS/WAN プロジェクトの鍵交換デーモン (pluto) は、IPv6 を用いて 鍵交換可能であった。

IPv6 を実装する観点から FreeS/WAN のスタックを考察すると、その実装が仮想ネットワークデバイスを 用いているため

- トランスポートモードの実装が繁雑になること
- IPsec ポリシの設定と同時にルーティングテーブルの設定が必要であること
- 他の IPv6 拡張ヘッダと処理ルーチンが全く別になること
- 完成したパケットを処理するためにパフォーマンスが悪いこと

などがあり、このため FreeS/WAN をベースとして IPv6 IPsec を実装することは、効率が悪く、実装したとしても良い実装ができないと判断した。

また、2001 年には、ヨーロッパの IABG が、FreeS/WAN ベースの IPv6 IPsec サポートを提供していたが、IPv4 と IPv6 でアーキテクチャが異なり中途半端なものであった。そこで、USAGI プロジェクトでは、鍵設定を行うためのカーネルとユーザランド間インターフェースである PFKEYv2 の実装部分のみを FreeS/WAN から再利用し、他のカーネル内の処理部分については新規に実装を行うことにした。その際の実装方針は次の通りである。

- 重複した実装を避けるため、仮想ネットワークデバイスではなく、IP スタック内に実装する
- IPv6 だけではなく、IPv4 のサポートも行う
- 暗号通信が法的に制限される国もあるため暗号はモジュール化する
- PFKEYv2 やデータベース、パケット処理はモジュール化する
- 処理速度の優先度は高くなく、できるだけシンプルな実装とする

IPsec を用いた運用で必要となる鍵交換デーモンに関しては、既存実装(pluto)があることから 優先度を下げ、カーネルの実装後に取り組むこととした。

2001 年度は、単純に AH, ESP の処理が行えるようになることを目標に開発を行った。この過程で、Interop 2001 の shownet で展示を行い、Ottwa Linux Symposium (OLS2001)、IETF 51st、IPsec Back off Finland、IETF 52nd、IETF 53rd に参加した。OLS2001 では、FreeS/WAN、IABG のメンバと IPsec の設計に関する議論を行い、FreeS/WAN はスタックではなく、ユーザサポートに注目していることを確認した。また、IETF 52nd では IBM USA から協力の申し出を受けた。当時の IPsec の実装を行う上で、 Symmetric Multi Processor(SMP)のサポートは、一つの課題であった、現在と異なり、SMP の環境は一般的でなく、SMP に関する経験を豊富に持っていた IBM の協力は、カーネル内にデータベースを必要とする IPsec の実装において非常に有効であった。

2002 年度は、主にトンネルモードのサポートと IPv4 の実装を行うと共に、メインラインへ統合する準備を行った。2002 年 9 月にメインライン(Linux-2.5)に対するパッチを netdev へ投稿したが、最終的には、メンテナ自身が別途開発したアーキテクチャによる IPsec が採用され、Linux-2.6 に導入されることとなった。メンテナの実装で特徴的なことは、現在の Linux ネットワークスタックで採用されているゼロコピーの思想を導入していることである。しかし、この IPsec は IPv4 のみであり IPv6 をサポートしていなかった。そこで、メンテナが導入した IPsec アーキテクチャに基づいて IPv6 の実装を行うこととなった。

2002 年度後半から 2003 年度にかけて、メンテナが導入したアーキテクチャによる IPsec に IPv6 サポート を追加する作業を行った。メンテナが導入したアーキテクチャは IPv4 を前提として設計している部分があり、この部分のサポートに多くの時間を費すこととなった。

これまで、2001 年度から 2002 年度にかけて独自に IPsec を実装した経験があったため、IPsec のその他の 部分に関しては作業を迅速に行うことが可能であった。そして、USAGI プロジェクトの成果として、2003 年末にリリースされた新しい安定版シリーズとなる Linux-2.6.0 に IPv6 の IPesc を組み込むことで、Linux は IPsec をサポートしたことになる。2004 年度以降は、基本的にメンテナンスフェーズに以降する。2004 年度には、これまでの実装によって得た知識をコミュニティに還元するべく、Linux-2.6 シリーズで IPsec を実装するために新しく導入されたアーキテクチャに関する発表を OLS2004 で発表した。また、当時最新の鍵交換プロトコルであった IKEv2 を実装する作業を開始し、後の Racoon2 プロジェクトとなった。

最近の活動としては、RFC3566(AES-XCBC-MAC-96)の実装や、クロスプロトコルファミリ IPsec トンネル (IPv4 over IPv6 IPSec tunnel, IPv6 over IPv4 IPsec tunnel)などの実装を行った。

Mobile IPv6 スタックに関する活動

MIPv6 スタックへの当初の取り組み(MIPL1 の開発に参加)

USAGI プロジェクトの MIPv6 の取り組みは、MIPL1 が USAGI プロジェクトの提供するカーネルパッチと協調して動作することを確認することから始まった。2001 年には、エヌ・ティ・ティ・ソフトウェア株式会社の高宮が参加し、MIPv6 の 度重なる仕様変更に対する追従、および TAHI イベントにて他実装との相互接続試験を行った。

仕様変更への追従については、MIPv6 基本仕様の ID-13 からはじまり、RFC3775 として公開されるまでの間、MIPL1 をベースとして USAGI プロジェクトのカーネルにマージしながら改良を行い、公開を行った。マージの過程では動作確認を行い、検出された重要なバグについては修正パッチを作成して Go-Core プロジェ クトへ不具合を報告し、MIPv6 スタックの品質の確保に寄与した。MIPL1 は 2.4 系カーネル向けに公開されていたが、完全なものではなく、ベース となっている Linux カーネルの IPv6 スタックも不十分であった。IPv6 スタックの品質を高めている USAGI プロジェクトで開発されたカーネルパッチ上では MIPL1 はコンパイルできず、MIPv6 スタックとして安定しているものは 存在しなかった。そこで USAGI プロジェクトとして、以下の目的を設定し、MIPv6 スタックの開発を実施した。

- MIPv6 スタックのよりよいデザインの設計

- USAGI プロジェクトで開発される機能との連携

作業としては、まず MIPL1 を USAGI プロジェクトのカーネルパッチにマージする ことから始め、バグ修正、機能改善等を実施した。また、MIPL1 のデザインの問題点の 1 つとして、ホームアドレスが直接インタ フェースに設定される構造が挙げあられる。この構造は、IPv6 の近隣探索との親和性が悪く、インタフェースに透過的な移動機能の実現が難しかった。この問題については GO-Core プロジェクトと OLS にて議論を行い、仮想デバイ スを使用してホームアドレスを管理する方式を提案し、試験実装を行った。成果は安定したコードとして公開されるまでには至らなかったが、有効性は確認することができた。また、この仮想デバイスによる管理は、その後の 2.6 系 カーネルにおいて、トンネルデバイスによるホームアドレスの管理という形で実現されることとなる。

さらに特筆すべき機能改善は、IPsec との連携である。MIPL1 は、IPsec の認証 ヘッダを独自で実装していたが、暗号ペイロードは実装していなかった。MIPL1 の IPsec は汎用性がなく、USAGI プロジェクトの IPsec の機能と重複する機 能であった。さらに、MIPv6 基本仕様が RFC として公開されたときには ESP が必 須となり、MIPL1 のみでは仕様を満たせなくなっていた。

2.4 系カーネルで IPsec を使用するためには、 USAGI プロジェクト、FreeS/WAN プロジェクト等のパッチが必要であった。 USAGI プロジェクトが IPsec の開発を行っているというアドバンテージを活かし、IPsec 担当者と協力しながら、MN-HA 間の MIPv6 シグナリング保護のための IPsec 利用の仕様を実装した。 ここまでの実装における成果は下の通りである。

成果

- TAHI テストイベント

当初は TAHI テストイベントの相互接続試験において、他実装と相互接続で きない状況が続いた。しかし、仕様に関しては KAME プロジェクトとの意識合わせを行い、参加回数を重ねる毎に仕様の詳細を把握することで、相互接続性を確保することができた。

- N+I

2002 年にシャープ(株)が、Zaurus に MIPv6 のモバイルノード機能を搭載したいと打診があり、協力した。KAME のモバイルノードに搭載されている Web カメラサーバからの動画を受信するデモを実施した。

2003 年では、同じくモバイルノード機能を搭載した Zaurus から、mplayer で動画ストリーミングを再生するデモを実施した。

mipl1 での成果 (mipl1 に見切りをつける)

MIPL1 の成果は、MIPv6 の仕様の更新に追従して提供できるようになった。当時、MIPv6 の仕様は、セキュリティの対応を含め、RFC 策定までめまぐるしく更新されていった。MIPL1 は、これらに対応するとともに、相互接続イベントに積極的に参加し、動作の検証を行ってきた。

頻繁な仕様の変更に追従できたのには、新たなにメンバーが加入したことが大き い。2002 年の 10 月に、偶然にも同時期に 2 名のメンバーが USAGI プロジェクトに参画した。(株)日立製作所の中村は、前任の同社のメンバーとの交代でプロジェクトに参画したが、交代を機に MIPv6 のホームエージェントの機能を担当するこ とになった。以降、ホームエージェントの機能にとどまらず、MIPv6 に関わるカーネル全般に幅広く携わることになる。また、シャープ(株)の新本は、MIPv6 で最も複雑な機能をもつモバイルノードを高宮とともに担当することになった。 さらに、モバイルノードの機能を Zaurus へ移植し、動作の検証も行っていく。

MIPv6 の基本仕様は、2004 年 5 月に RFC が発行されるまで頻繁に更新された。2002 年以降、ドラフト 16 版からドラフトの最終版となる 24 版までの更新が 1 年半の間に行われた。さらに、これに平行してセキュリティ機能を定める、MN-HA 間の MIPv6 シグナリング保護のための IPsec 利用の仕様が定められた。MIPL1 では、こ れらに対応して、MIPL0.9.3(Linux2.4.18), MIPL0.9.5.1(Linux2.4.20)とリリースを行った。

当時の USAGI プロジェクトの MIPv6 に対する作業は、HUT のリリースコードのバグ フィックスであり、ヘビーユーザの域を出ていなかった。HUT へバグと修正案を提示し、HUT の管理コードに反映される。ところが、MIPv6 のセキュリティ対応を必要としたころを境に、USAGI プロジェクトは HUT とは分岐してコードの管理を行 うようになる。

MIPv6 のセキュリティへの対応は、MIPv6 スタックと IPSec スタックが連動して動作する必要があった。USAGI プロジェクトには、IPSec のスペシャリストがメンバーにいたため、その作業にはうってつけだった。この MN-HA 間の MIPv6 シグナリング保護のための IPsec 利用の仕様への対応は、USAGI プロジェクトが主導して実装を行った。コードの更新が大幅に必要だったため、HUT のリリースする MIPv6 スタックとは独立して作業が進め、十分に動作を検証できた後に、HUT へコードの開示し、MIPL へマージされるというスケジュールで作業が進められた。

そのため、この頃から積極的に相互接続イベントへ参加するようになった。国内で催される TAHI 相互接続テストイベントに加え、海外で開催される相互接続テストイベントにも積極的に参加するようになった。特に、ベルギー・ブリュッセルで行われた第 4 回 ETSI IPv6 Plugtest Event は、初めて海外で参加した相互接続 イベントであり、思い出深いものであった。

この相互接続イベントは、MIPv6 基本仕様は第 24 版、MN-HA 間の MIPv6 シグナリング保護のための IPSec 利用の仕様は第 6 版が発行され、各仕様は RFC 発行へ向けて基本合意された時期であった。このため、前後参加した相互接続イベントをみても、多数の団体が参加する盛況な相互接続イベントであった。MIPL1 は安定して動作しており、問題なく相互接続性を確認することができた。イベントにおいて焦点となった IPSec によるセキュリティ機能についても、位置登録メッセージ(BU/BA)のトランスポートモード IPSec や、ホームエージェントとモバイルノー ド間のトンネルモード IPSec について、他団体と比較しても安定した動作を確認することができた。さらに、モバイルノードについては、PC での検証だけでなく、Zaurus へ移植を行い、相互接続性を確認することができた。

MIPL1 は、TAHI テストイベントや ETSI IPv6 Plugtest Event などのそれぞれの相互接続試験において、当時の IPv6 の仕様に追従した実装を準備して参加し、安定した動作を確認することができた。

また、広く外部へ成果をアピールすることも精力的に行った。2002 年には、KAME プロジェクトと協力し、Netwold+Interop へ出展し、デモンストレーションを行った。Zaurus をモバイルノートとし、KAME プロジェクトのホームエージェント として、映像を配信するシステムを展示し、MIPv6 の有効性と MIPL1 の安定性を示すことができた。

最終的に、Linux2.4 系を基盤とする MIPL1 は、2004 年 1 月にリリースされた USAGI プロジェクトの Stable5 にその成果含んで公開することができた。以降、Linux2.6 系を基盤とする新実装に向けた作業を中心に作業を進めていくことになる。

mipl2 の開発(カーネルからユーザランドへ)

USAGI プロジェクトは、2.4 系カーネルでの開発では、MIPv6 スタックの改良を 独自の開発ツリー行うまでであったが、2.5 系カーネルの開発ツリーが現われた後は、MIPv6 スタックの設計を洗練し実績を向上させて、2.5/2.6 系カーネルにパッチが取り込まれることを開発目標に設定した。

一方、GO-Core プロジェクトも、MIPL1 の 2.5 系カーネル移植を開始し、カーネルパッチを Linux のネットワーク開発コミュニティ(netdev) に公開した。

- 2002/10/02 に、2.5.40 カーネルパッチ
  http://marc.info/?l=linux-kernel¥&m=103355091705888¥&w=2

- 2002/10/17 に、2.5.43 カーネルパッチ
  http://marc.info/?l=linux-kernel¥&m=103487225025475¥&w=2

これらのパッチは、MIPL1 の設計を多く引き継いでおり、MIPv6 スタックがカーネル空間で実現されるという特徴を持っていた。

netdev では、これらのパッチは、カーネルのソースコード修正量が 多過ぎるという見解となり、受け入れられなかった。netdev からの主な要望は以下である。

- ユーザ空間で実現可能な部分を切り離すこと
- カーネルで MIPv6 に特化する部分が少くなるように既存の機能を使いまわすこと

そこで、カーネル空間の修正を少なくするために MIPv6 スタックを 再設計する必要があった。
2003/04 頃、netdev の取りまとめである David S. Miller 氏は、以下の MIPv6 スタック開発関係者と設計案について整理を行った。

- GO-Core プロジェクト
- IBM の Venkata 氏
- USAGI プロジェクト

GO-Core プロジェクト、USAGI プロジェクトはそれぞれの設計案を提示したところ、基本的な部分は類似していた。違いは、経路最適化を実現するためのデータ管理方法であった。

- 経路表のキャッシュ機能で行う(GO-Core プロジェクト)
- XFRM 機能で行う(USAGI プロジェクト)

USAGI プロジェクトは、類似した設計を元に別々に実装するのは意味がないと判断し、協力して一つの実装を作る提案をした。そして、カーネルとユーザランドの基本設計と開発方針の意識合わせを行うため、2003/09 に GO-Core プロジェクトを短期的に日本に招待し、face-to-face で集中的な 議論を行った。主な決定事項は以下の通りである。

- カーネルで拡張するリソースの決定
  - 相違点だった経路最適化のデータ管理は XFRM 機能で行うことにした
- カーネルのインターフェースの決定
- ユーザランドのデーモンは MIPL2 と呼ぶことにする
- 開発ツリーは mobile-ipv6.org ドメインのサーバで GO-Core プロジェクトが 管理する

以降、USAGI プロジェクトは GO-Core プロジェクトと共同開発を進めていくことになる。しかし、共同開発を進めるにあたり、以下の問題があった。

- USAGI プロジェクトの活動に対して GO-Core プロジェクトの応答が遅く、USAGI プロジェクトが送付したパッチに対する議論や結論が無いまま、後になって別方式で GO-Core プロジェクトによって実現されるという事態が散見される
- GO-Core プロジェクトが大きな変更を行うときに、USAGI プロジェクトへの相談が無い
- 品質の考え方に差があり、USAGI プロジェクトからは、ある程度動作実績のあるコードを 提供しているが、GO-Core プロジェクトが取り込むコードは動作実績が無いことが多い

USAGI プロジェクトの成果のうち、カーネルのパッチは開発ツリーへ取り込まれるもののユーザランドのパッチはほとんど取り込まれない状態が継続した。
USAGI プロジェクトは、GO-Core プロジェクトが共同開発ツリーを事実上管理できていないと判断し、相互接続イベント参加など外部への公開時には、一時的に独自のツリーで開発していた。イベント終了後には、成果をパッチで GO-Core プロジェクトに提出していたが、ユーザランドのパッチが取り込まれにくい事情は変わらなかった。
一方、2004 年に、Linux の利用環境が益々増えていく中で、同プラットフォーム上で 動作する MIPv6 の参照プロトコルスタックの重要性が高いと判断した日本エリクソンより、杉本が USAGI プロジェクトに加入する。

杉本は、加入前からMIPv6の仕様に造詣が深いため、即戦力として活躍し、USAGIプロジェクトの開発を加速させた。また、MIPv6スタック開発者とIPsecスタック開発者がそれぞれ重要と思いながらも相互の仕様と実装を理解するハードルが高いため検討と実装がなされていなかったMIPv6-IPsec連携機能を主に担当し、他実装に先行した。
成果はドラフト(PF_KEY Extension as an Interface between Mobile IPv6 and IPsec/IKE)としてIETFより公開している。

新メンバー加入で活動がより活性化したUSAGIプロジェクトは、GO-Coreプロジェクトとの共同開発が難航し続けたものの、各種相互接続イベントに参加してMIPL2を検証していった。
カーネルが落ちない程度に動作し出すのは、ホームエージェントは2004年2月(Connectathon 2004)、モバイルノード/コレスポンデントノードは 2004年11月(5th ETSI IPv6 Plugtests)である。カーネルバージョンが2.6.8だったこの時期、モバイルノードの成果をZaurus SL6000へ移植し、検証を始めている。2005年2月(5th TAHI Event)には、カーネルとユーザランドの基本機能が一通り揃った。2005年10月にリリースしたMIPL-2.0-rc3では、カーネルのIPv6およびMIPv6の不具合、MIPL2の不具合が修正されホームエージェント/コレスポンデントノードでTAHIテストイベントの仕様適合性試験に全PASS相当の品質に達した。2006年1月(6th TAHI Event)でモバイルノード(PC/Zaurus両方)もTAHIテストイベントの仕様適合性試験を全PASS相当に達した。また、モバイルノード-ホームエージェント間のIKEの相互接続について基本動作を確認し、今後の問題点を洗い出すことに成功した。

Linuxメインラインへのマージとその後

Linuxメインラインへのマージ作業

メインラインカーネルへのマージ作業は主にユーザ空間の開発がほぼ収束した後、2006年中に集中的に行われた。マージ作業は、従来の慣習に従い段階的に進められた。まずは機能毎にパッチを分類し、それぞれをnetdevに送信し、public reviewのフィードバックを待つ。何らかの修正が必要な場合はそれを施した後に再度パッチを公開し、特に問題がなければマージの候補となる。こうしたパッチはDavid S. Miller氏のツリー、続いてメインラインツリーへとマージされる。MIPv6に必要なカーネルパッチは上記のサイクルを経て、全てがメインラインカーネルにマージされた。
MIPv6のために必要なパッチは、機能別に分類すると以下のものがある。

- MIPv6特有の拡張ヘッダ処理
- NDP拡張
- MHメッセージのカーネル内処理
- Policy Routing
- Home Address対応
- MIGRATE

MIPv6特有の拡張ヘッダ、具体的にはHome address destination optionとRouting header type 2を処理する機能をカーネルに追加した。これは、XFRMフレームワークを拡張することによって実現した。前述の二つの拡張ヘッダはXFRMで取り扱われる新たなプロトコルとして追加され、カーネル内での当該拡張ヘッダの送受信処理が可能となった。パッチの多くは2.6.19に取り込まれた。また、MIPv6特有のポリシと従来のIPsecのポリシと区別するため、サブポリシという概念を導入した。これにより、IPsecとMIPv6の管理を独立して行うことが可能となり、ユーザビリティが向上した。この他、XFRMに関連した機能としてMIPv6機能のモジュール化が2.6.23にて取り込まれた。
IPv6 Neighbor Discovery Protocol(NDP)に関しては、既存のProxy NDPのコードの不具合を修正することでホームエージェントに必要な機能を充足させた。具体的には、ユニキャストアドレス宛のNeighbor Solicitationメッセージが正しく処理されるようプログラムを修正した。関連したパッチは2.6.19に取り込まれた。
MHメッセージのカーネル内処理は、チェックサムの計算、ヘッダ長の確認等の基本的なチェックが含まれる。また、エラーが見つかった際に必要に応じてユーザ空間(mip6d)がエラーメッセージ(BE)を送信するために、エラー通知を

行う機能をカーネル内に実装した。関連したパッチは 2.6.19 にて取り込まれた。

Policy Routing に関しては、HUT Go-Core プロジェクトと USAGI プロジェクトが伴に開発を続けてきたが、Thomas Graf 氏によって同等機能のパッチがメインライン(2.6.19)に先に取り込まれた。これにより、Policy Routing の一般的な機能は実現されたが、モバイルノードに必要な機能の点ではまだ不備があった。具体的には、送信アドレス指定ルールと送信アドレス選択機能の間にジレンマがあり、それを解決する必要があった。USAGI プロジェクトは、解決方法を議論し、パッチをメインラインに提供した。パッチは無事にメインラインカーネル(2.6.22)にマージされた。

ホームアドレスの拡張は、モバイルノードに必要な機能である。カーネルは、 ホームアドレスをその他の IP アドレスと区別し、いくつかの例外的な処理を行う必要がある。モバイルノードがホームリンクに戻った際、重複アドレス検知は行われない。また、送信元 IP アドレスを指定しないアプリケーションに対し て、カーネルはホームアドレスを優先的に選択する。関連したパッチは 2.6.19 もしくはそれ以前に取り込まれた。

最後に、MIGRATE と呼ばれる拡張機能である。これは MIPv6 の移動に伴って IPsec SPD/SAD のトンネルのエンドポイントのアドレスを更新するための機能である。本機能は、MIPL2 の開発を通じて必要な機能として認識されたもので、IETF に向けた技術提案も行った。具体的には、PF_KEY フレームワークに MIGRATE と呼ばれる新たなメッセージを定義し、このメッセージによって MIPv6 が IPsec/IKE に対して移動に伴う IP アドレスの更新を通知することが可能となった。関連したパッチは 2.6.21 で取り込まれた。

UMIP

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下 進められて来たが、HUT Go-Core プロジェクトの終了と伴に、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下にしたリリース(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)をリリースしている。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。

今後の取り組み

ここでは、USAGI プロジェクト完遂後の UMIP のメンテナンス方針について説明 する。

目的・ゴール

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するこ と。

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が、前者は既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。

ソースツリーとメンテナンス方針

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。

開発メンバ

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。

リリース頻度

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一度の頻度でバージョンアップを行ってきている。

ソフトウェア格納場所

現在議論中である。いずれにせよ、一般ユーザが簡単にソフトウェアをダウン ロード可能な環境を提供する予定である。

Netfilter に関する活動

パケットフィルタの開発

USAGI プロジェクトは 2003 年度から本格的に Linux における IPv6 パケットフィルタ 機能の拡充および品質改善を開始した。当時から Linux のパケットフィルタ機能はオープンな Linux コミュニティである Netfilter プロジェクトにより開発、保守されており、既に IPv6 パケットフィルタは彼らによって実装されていた。しかし、IPv6 パケットフィルタは IPv4 パケットフィルタに比べ多くの機能が不足しており、バグを多く抱えていた。さらに、開発者不足から IPv4 機能に対する改善が IPv6 機能にも適用されるまで長い時間が費やされる状況が続いていた。そこで、USAGI プロジェクトは IPv4 パケットフィルタと同程度の機能と品質を備えた IPv6 パケットフィルタの提供を目的として以下の開発活動を行った。

- IPv6 に対応した Stateful filter の開発
- IPv6 パケットフィルタの品質向上とルールの拡充

以下ではこれら 2 つの活動内容について報告する。

IPv6 に対応した Stateful filter の開発

Stateful filter は通信コネクションの状態を追跡し、それを基にパケットの破棄や 通過の許可を判断する機能である。この機能は Linux において IPv6 パケットフィルタの中でも最も大きな不足機能であった。そこで、2003 年 6 月、東芝から提供された実装を最新カーネルに導入する形で IPv6 版 Stateful filter の実装を開始した。この 実装は、2003 年当時既に存在していた IPv4 版 Stateful filter のコードをベースにしたものであった。2003 年 12 月、Netfilter プロジェクトおよび Linux カーネル開発者のメーリングリストに投稿した。この投稿を契機に、Stateful filter の コードデザインに関する議論をパケットフィルタの開発者との間で活発に行い、IPv4 版と IPv6 版の Stateful filter の統合する方向で意見が一致した(以下では、統合した Stateful filter を単に Stateful filter と記す)。この後、本格的な Stateful filter の実装が始まるが、基本的にコードを USAGI プロジェクトが実装し、それを Netfilter プロジェクト内のメーリングリスト投稿、意見を集約し、さらなる改善を繰り返す体制で開発を行った。最終的に、Stateful filter には以下の要件を採用した。

- 下位互換性の保持
  新規実装の Stateful filter は従来の IPv4 版 Stateful filter を置き換える機能である。そのため、これまで後者と連携動作していたカーネルモジュールやユーザツールに対し下位互換性を提供した。また、開発者やユーザが緩やかに移行できるよう、移行期間中は両方の機能が Linux カーネルのソースツリーに並存できるようにした。

- メモリ使用量の低減

Stateful filter は通信コネクションを識別するため、各コネクションの両端点を表す IP アドレスやポート番号といった情報を保持する。Stateful filter を IPv6 に対応させるためには 32bit の IPv4 アドレスだけでなく 128bit のアドレスをコネクション毎に保持できるようにする必要があり、コネクション数が多いほどメモリ使用量が増大する 結果となった。この増大は Stateful filter を IPv6 に対応させるために不可欠であるため、その代わりに他の箇所で使用メモリ量を低減した。具体的には、各コネクション毎に割り当てられるメモリ領域のうち、基本機能用の領域、IPv4, IPv6 に共通な拡張機能用の領域、IPv4 NAT 機能が利用する領域に分割し、それぞれの領域を必要な場合のみ動的に割り当て可能とした。

- フラグメント化された IPv6 パケットの処理
  Stateful filter はパケット中の IP ヘッダやトランスポートプロトコルヘッダだけでなく、サポートしているアプリケーションプロトコルのデータも走査する。そのため State filter は IPv6 ホストにより複数のパケットに分割されたパケットを再構築するが、解析後再び分割する際にパケットサイズを変更しないよう工夫する 必要があった。

Stateful filter はこのほかにも多くの開発者の意見を取り入れコードの品質を向上させた。また、開発の過程で従来の IPv4 版 Stateful filter に対する機能改善や不具合修正をその都度新たな Stateful filter にも適用し、いつでも最新の Linux カーネルに導入する準備ができている状態にした。その結果、Stateful filter は 2005 年 11 月 Linux カーネル 2.6.15 に採用された。

IPv6 パケットフィルタの品質向上とルールの拡充

Stateful filter の開発と並行して、USAGI プロジェクトは 2004 年 8 月より IPv6 パケットフィルタの品質向上活動およびフィルタルールの拡充を開始した。2004 年当時、Linux の IPv6 パケットフィルタは TCP, UDP, ICMPv6 パケットの破棄、通過許可といった基本的な機能に加え、以下の拡張モジュールをサポートしていた。

- パケットに対してマッチングを行う 15 個の拡張マッチモジュール。多くは IPv6 拡張ヘッダに対するマッチングモジュールである。
- パケットに対して操作を加える 2 個の拡張ターゲットモジュール。

一方、IPv4 パケットフィルタは 22 個のマッチルール用拡張モジュールと、14 個のターゲットルール用拡張モジュールをサポートしていた。このうち、IPv6 パケットフィルタにも転用可能にも関わらず移植されていないモジュールは それぞれ 6 個、5 個存在していた。また、IPv6 パケットフィルタの拡張モジュールのうち 9 モジュールがバグを含んでおり、IPv4 パケットフィルタと比べて低品質な状態にあった。そこで、USAGI プロジェクトは 以下の活動を行った。

- 全モジュールの試験およびバグフィクス
  この作業により他のパケットフィルタ開発者から大いに信頼を得ることができ、その後 Stateful filter など他の開発をスムーズに行うことができるようになった。なお、この作業を行うためには様々なタイプのパケットを生成し、実際にそれらをパケットフィルタに処理させる必要があったため、パケットの生成には TAHI プロジェクトが開発している IPv6 Conformance Test Tool を利用した。

- ルールの拡充
  IPv6 に不足している機能を追加する最も容易な方法は IPv4 用のモジュールを IPv6 パケットフィルタに移植することであるが、メンテナンスコストが倍になる。そこでまず、カーネルおよびパケットフィルタ設定ツール両方に対して, IPv4, IPv6 に依存しない共通の処理を統合した。これにより、ひとつのフィルタルール用モジュールを実装すれば, IPv4、IPv6 どちらのパケットフィルタでも利用可能とすることができた。この上で, IPv4 のみをサポートしていたルール用モジュールを IPv6 にも可能な限り対応するよう変更した。この結果、2007 年 10 月時点で Linux2.6.23 は IPv6 に対応したマッチ用拡張モジュールを 34 個、ターゲット用拡張モジュールを 14 個と することができた。なおこの作業のうち、フィルタルール設定ツールにおける統合を USAGI プロジェクトが

行い，Linux カーネルにおける統合を Netfilter プロジェクトの他の開発者が中心となって行った。

まとめ

USAGI プロジェクトが Linux における IPv6 パケットフィルタの開発に貢献した結果， Linux の IPv6 パケットフィルタは IPv4 パケットフィルタと遜色ない機能と品質を備えるようになった。また、この貢献が認められ，2005 年には USAGI プロジェクトのパケット フィルタ開発担当者が Netfilter プロジェクトのコアメンバに就任し、引き続き IPv6 関連機能のメンテナンスを行っている。

IPv6 プロトコルスタック品質向上に関する活動

プロジェクトの目的として「製品として耐えうる品質の IPv6,IPsec プロトコ ルスタックを Linux に提供すること」を挙げている通り、USAGI プロジェクトでは常に品質に気を配り活動を行ってきた。我々の活動により品質が向上していったことはメンテナに認められている。現に 2004 年 10 月 Linux のネットワークレイヤーのメンテナ David S. Miller 氏が来日した際に品質について協議し、Linux の IPv6 サポートを EXPERIMENTAL（実験的）より解除することの同意を得ている。カーネルバージョンで言えば 2.6.12 のリリース前の時期である。

品質の向上に関する初期の活動は、パッチ作成ごとの TAHI Conformance Test の手動実行、IPv6 相互接続性テストイベントへの参加であった。この後，IPv6 Ready Logo Program の開始に伴い同プログラムへの参加，メインラインカーネルへのコードの統合が十分に進んだことに伴い自動レグレッションテ ストシステムの構築と活動を広げていった。

本節では我々が行ってきた品質向上活動として，IPv6 相互接続性テストイベントへの参加の歴史，IPv6 Ready Logo Program への参加の歴史、そしてメインラインコードのレグレッションを早期発見するために開発した TAHI Automatic Running System の開発の動機とシステムの解説を行う。最後に Linux の IPv6 プロ トコルスタックがどのように向上していったのか、その経緯をグラフ化し示す。

IPv6 相互接続性検証イベントへの参加

IPv6 に限らずネットワークプロトコルスタックの品質の評価項目として相互 接続性は重要な位置を占める。しかし、開発中のプロトコルスタックと同種のプロトコルスタックを持つ製品を手に入れるのは、ことそのプロトコルが新しいほど難しい。そこで我々は各地で催されている IPv6 相互接続性検証イベントに参加し、随時相互接続性の確認を行ってきた。

以下に過去に参加したイベントを挙げる。

- 2001/08: IPsec Bakeoff
- 2002/01: 3rd TAHI IPv6 Interoperability Test Event
- 2003/01: 4th TAHI IPv6 Interoperability Test Event
- 2003/09: 4th ETSI IPv6 Plugtests
- 2004/01: 5th TAHI IPv6 Interoperability Test Event
- 2004/02: Connectathon 2004
- 2004/11: 5th ETSI IPv6 Plugtest
- 2005/01: 6th TAHI IPv6 Interoperability Test Event
- 2006/01: 8th TAHI IPv6 Interoperability Test Event
- 2007/05: 9th TAHI IPv6 Interoperability Test Event

IPv6 Ready Logo Program への参加

IPv6 Ready Logo Program は国際認証機関である IPv6 Ready Logo Committee が相互運用性を認めた製品に認

証ロゴを発行するプログラムである。本プログラムにおいて、2003 年 9 月に「IPv6 機器同士が通信できること」に着目した"IPv6 Ready Logo Program Phase-1"が、2005 年 2 月に「実際のネットワー ク環境での使用に耐えること」に着目した"IPv6 Ready Logo Program Phase-2"が開始された。USAGI プロジェクトもこのプログラムに参加し、第三者機関による品質の証明を得るよう努めてきた。

プログラム参加当初は USAGI プロトコルスタックによるロゴ取得を目指した。最初にロゴを取得したのは 2004 年 4 月で、Phase-1 Logo を Host および Router において取得している。この後は USAGI プロトコルスタックでロゴの取得／更新を行う一方で，独立行政法人産業技術総合研究所、アルファシステムズと協業を行 い，USAGI プロトコルスタックを搭載したディストリビューション「KNOPPIX/IPv6」で Linux ディストリビューションとして世界で初めて Phase-1 Logo の取得を行った。しかし、USAGI プロトコルスタック独自でロゴを取得する意義は、USAGI プロトコルスタックのメインラインカーネルへの統合が進むことで薄れていった。USAGI プロトコルスタックでの本プログラムへの参加は STABLE RELEASE 6 での Phase-1 Logo 取得を最後に終えている。

USAGI プロトコルスタックのメインラインカーネルへの統合が充分に進んだ後は、メインラインカーネルにおいてロゴ取得を目指した。メインラインカーネルでの参加においては参加団体名を kernel.org とし、コンタクト先を Linux カーネルのネットワーク機能の共同メンテナである吉藤英明としている。USAGI プロ トコルスタックと同様、最初は Phase-1 で参加していたが，Phase-2 の開始と共にターゲットを Phase-2 に変えている。以下に取得したロゴと認定日、および取得時のバージョンを挙げる。

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

USAGI プロトコルスタック - 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

KNOPPIX IPv6 Edition における協業として

- 2.4 系 [Phase-1, Host]
  - 2004/08/29: knoppix_v3.4_ja20040629_ipv6_20040705

- 2.4 系 [Phase-1, Router]
  - 2004/08/29: knoppix_v3.4_ja20040629_ipv6_20040705

- 2.6 系 [Phase-1, Host]
  - 2004/08/29: knoppix_v3.4_ja20040629_ipv6_20040705

- 2.6 系 [Phase-1, Router]
  - 2004/08/29: knoppix_v3.4_ja20040629_ipv6_20040705

TAHI Automatic Running System

USAGI プロトコルスタックのメインラインカーネルへの統合が十分に進んだ後 は，大きな新機能以外のコードは手元に貯めることなく直接メインラインカーネルに提供する開発形態に自然切り替わった。メインライン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おいても日々数多くの 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っていた。

そこで、我々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レグレッションテストシステムの開発に着手した。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 を利用することにした。

本システムの開発に着手したのは 2004 年 7 月の後半である。同年 10 月に Linux のネットワークレイヤーのメンテナ David S. Miller 氏が来日した際にこのシステムを紹介し，その後公開に至った。

公開当初の 2004 年は Phase-1 向けの(host,router)の IPv6 Ready Logo Conformance Test を自動実行していた。2005 年には TAHI プロジェクトによる v6eval テストフレームワークを利用したテストスイートであれば、複数の テストスイートを順次実行できるよう機能拡張を行った。これにより Phase-2 Core(host,router)および Phase-2 IPsec(end-node, security gateway)のテストを実行対象に含めた。2006 年には MIPv6(CN,HA)および IPsec において IPv4 および Authentication Header のテストを実行対象に含めた。またこの時 期 Phase-1 のテストスイートが Phase-2 のテストスイートに統合されたのに伴い，Phase-1 のテストスイートを実行対象から外している。

現在以下のテストを自動実行している。

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

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。ウェブブラウザにおける表示例を図 1: ブラウザにおける表示例に示す。また結果比較の表示例を図 2: テスト結果の比較に示す。

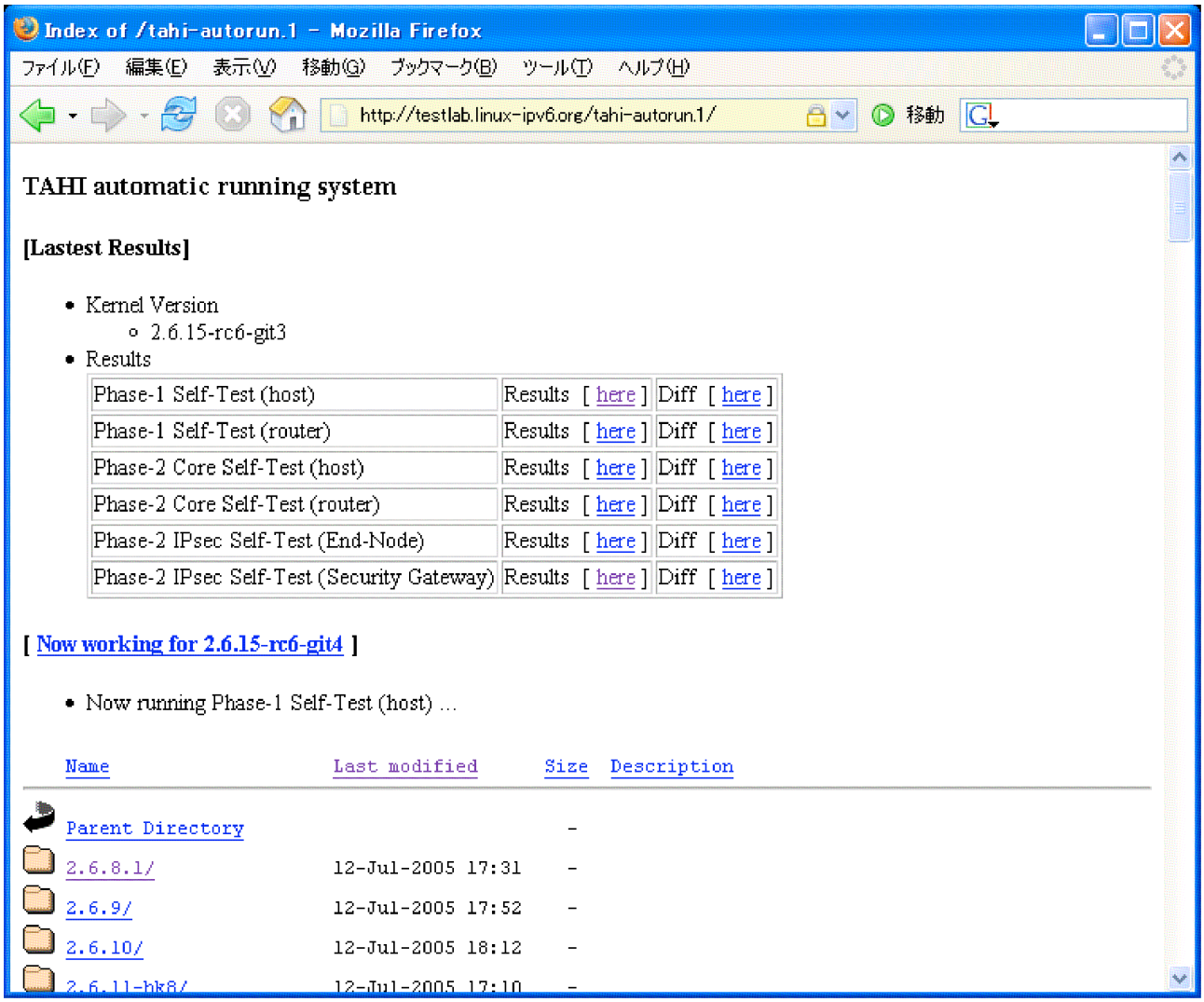
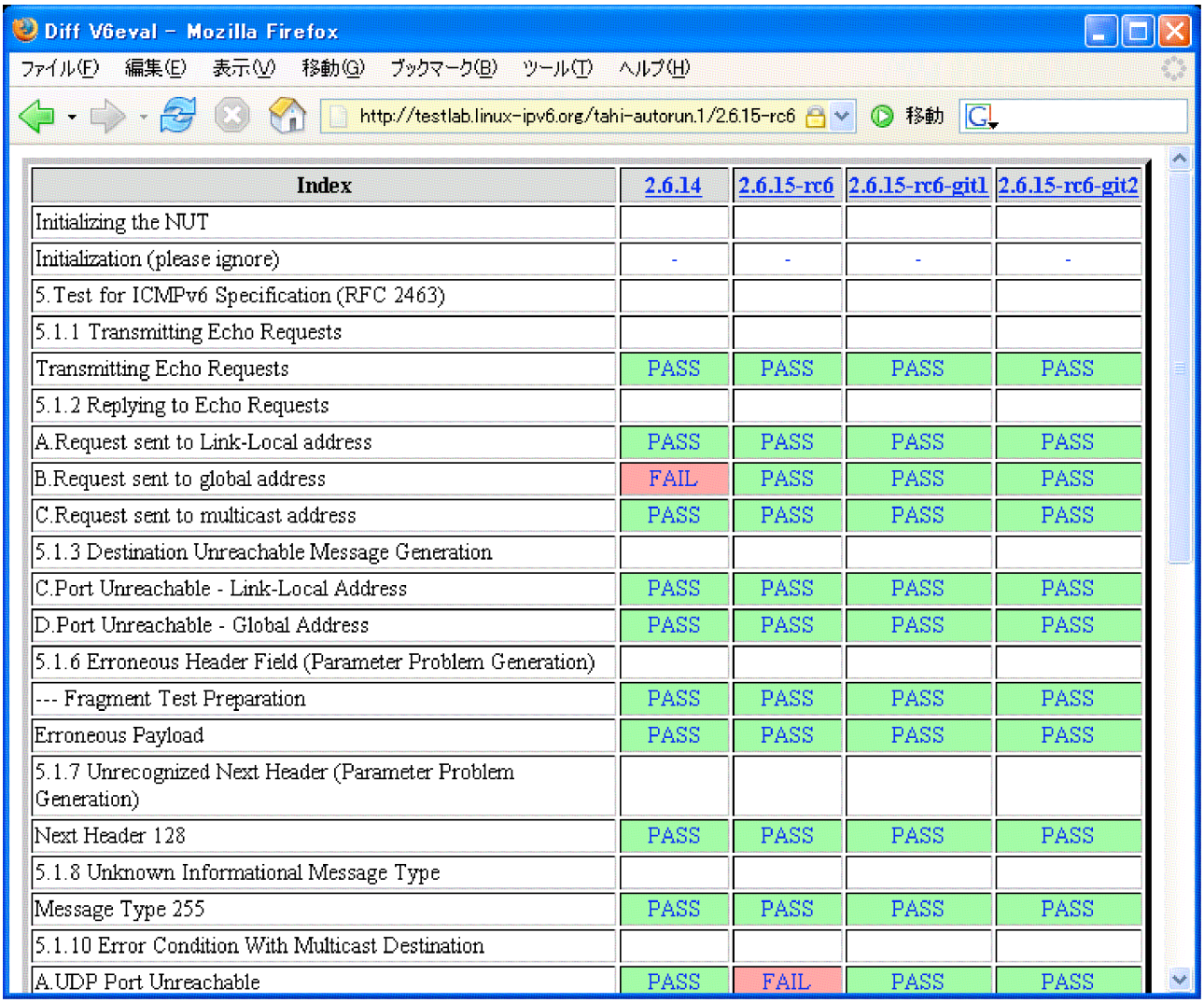


図 1: ブラウザにおける表示例



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: テスト結果の比較

品質向上状況の推移

ここでは TAHI プロジェクトの提供する 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 系カーネルのホスト機能，ルーター機能の品質が向上していった様子を示す。仕様したテストスイートのバージョンは本報告書執筆時に最新であった 1.4.9 である。

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 以降はその時点で最新のテストスイートの項目において FAIL はない。また、最新カーネル.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。

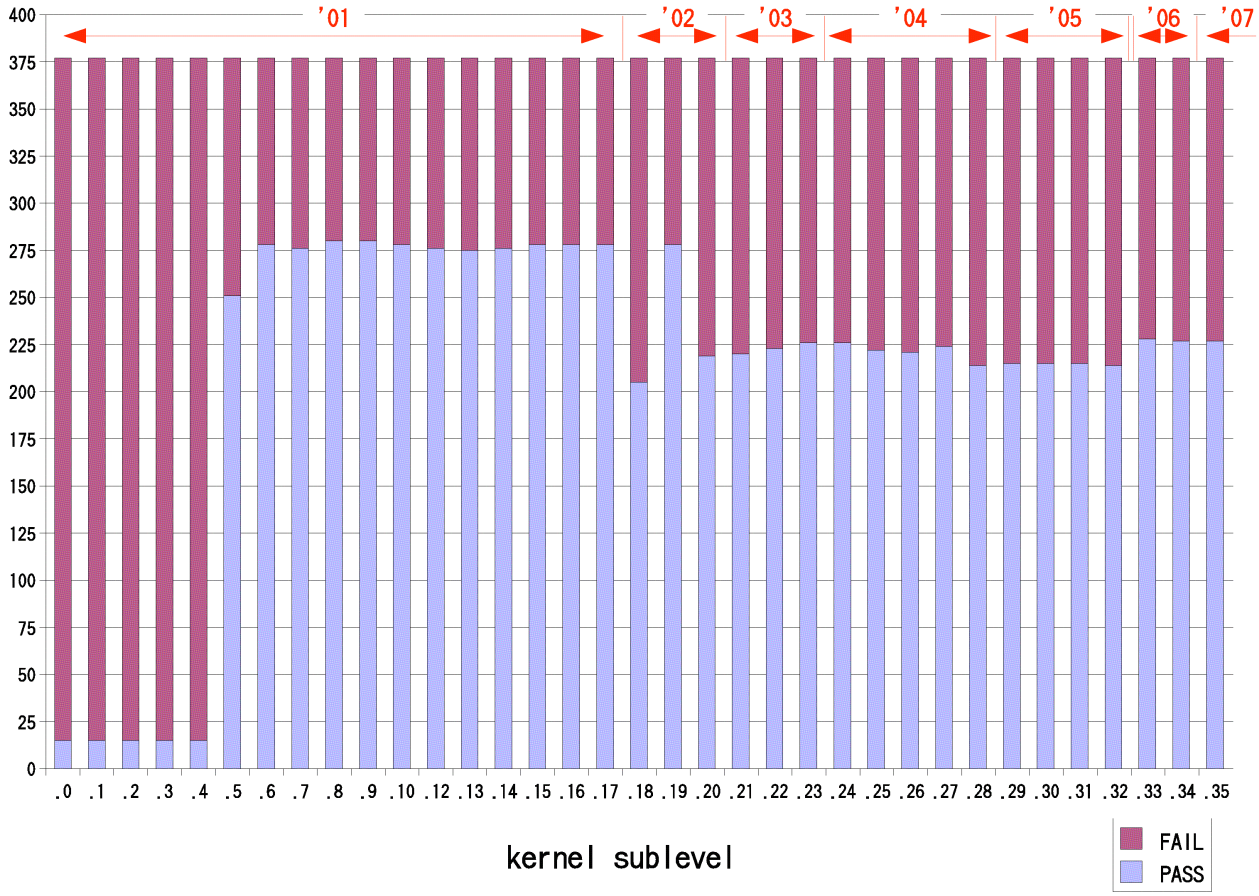


図 3: 2.4 系カーネル IPv6 ホストテスト結果

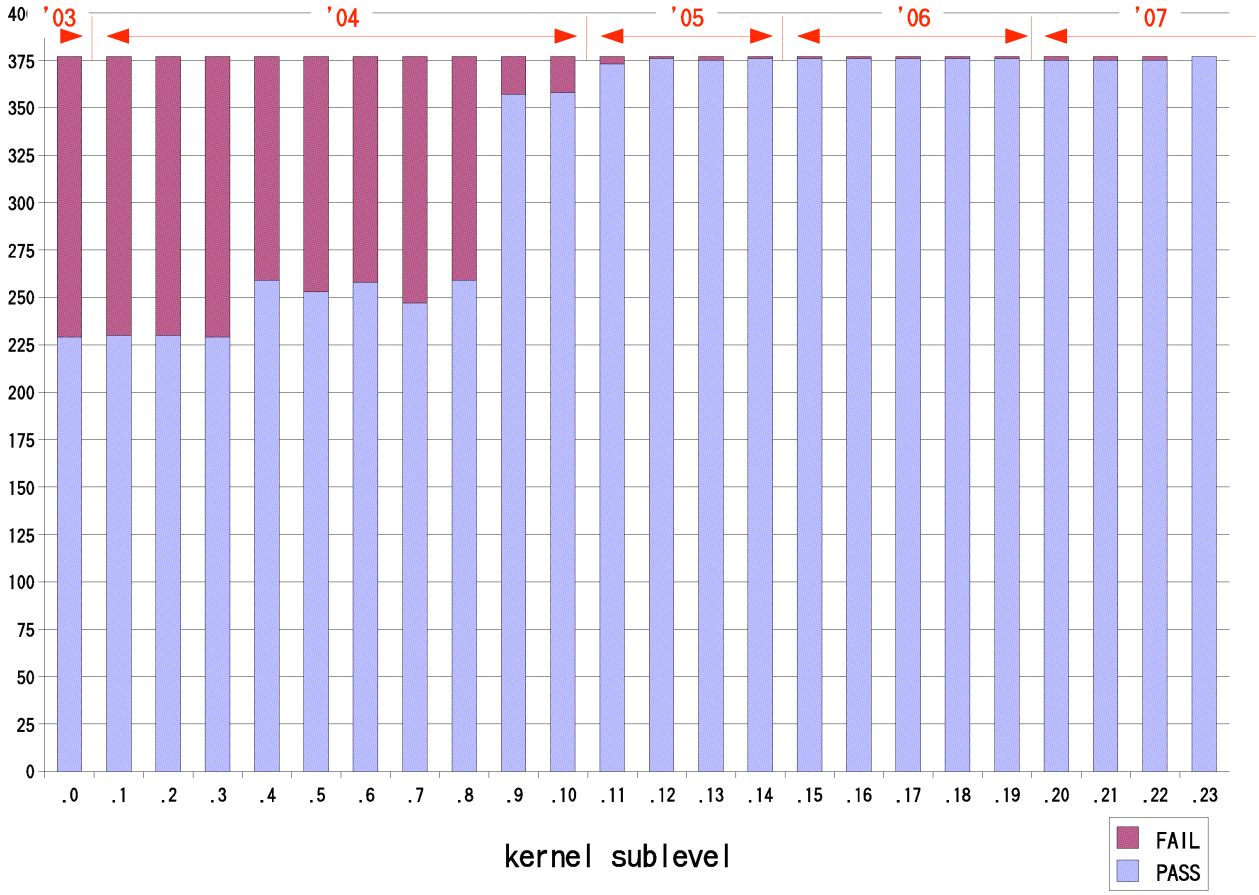


図 4: 2.6 系カーネル IPv6 ホストテスト結果

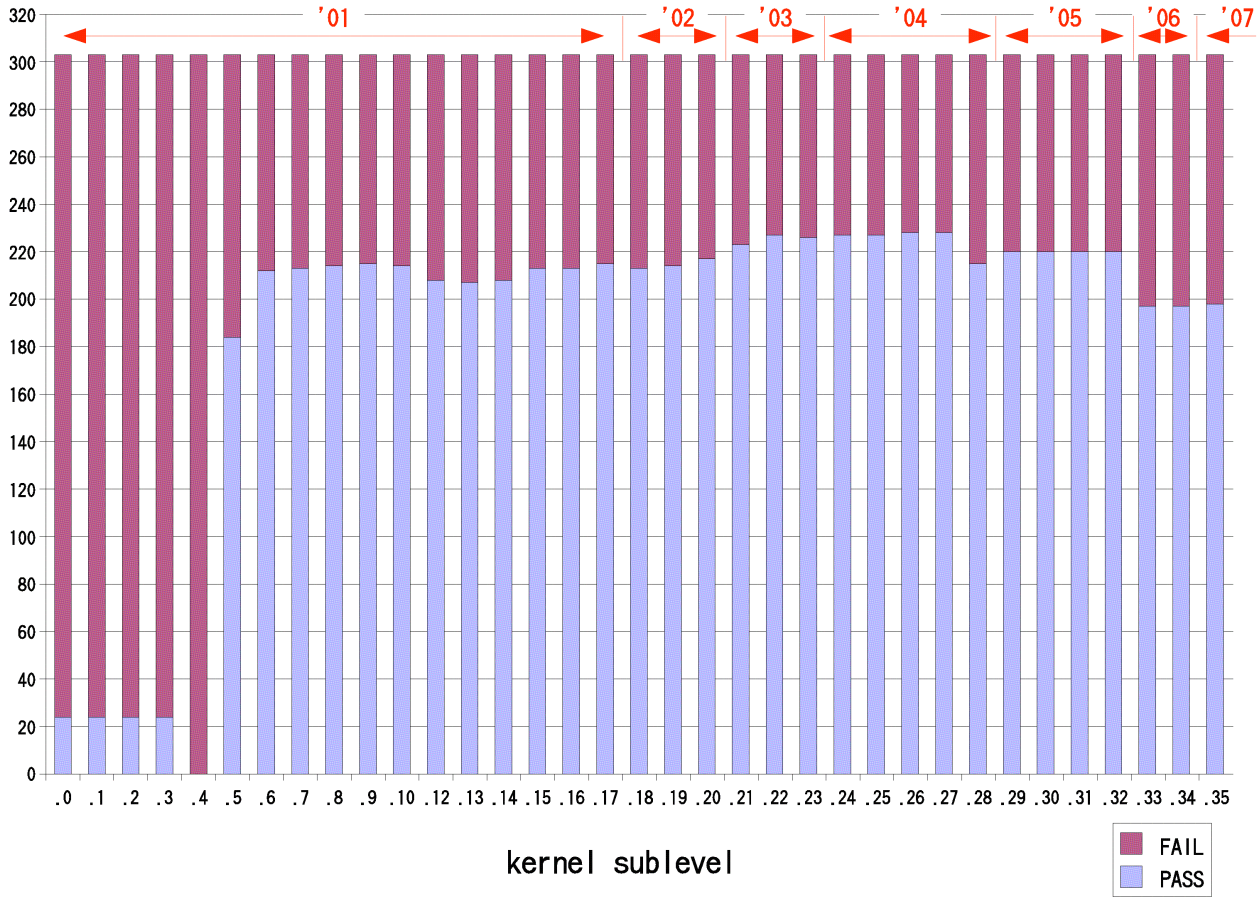


図 5: 2.4 系カーネル IPv6 ルータテスト結果

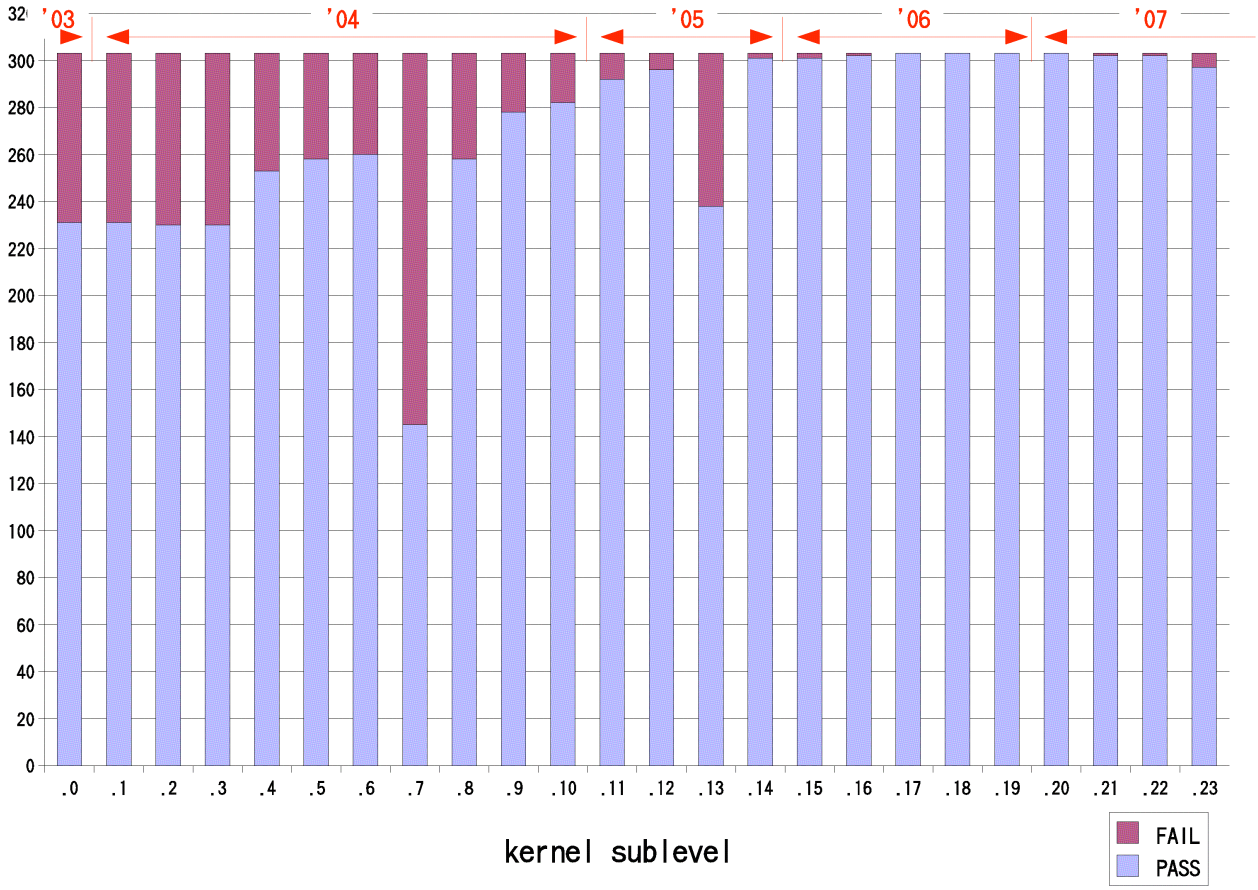


図 6: 2.6 系カーネル IPv6 ルータテスト結果