第 7 部

次世代インターネットプロトコル

第 1 章

v6 分科会

v6 分科会では、IPv6、6bone、および、IPsec について研究を進めている。この章では、v6 分科会が 1997 年度に活動した内容について報告する。

IPv6 に関する研究成果として、まず始点アドレス選択を 1.1節で述べる。エニーキャストについては、1.2節で解説する。1.3節と 1.4節では、それぞれ IPv6 over ATM PVC と PPP について触れる。アプリケーションを IPv6 へ対応させる試みを 1.5節で説明するが、メールに関しては節を分け 1.6節で解説する。また、1.7節でトランスポート・リレー方式のトランスレータについて触れ、IPv6 対応の DNS については 1.8節で述べる。

6bone の運用に関する成果として、6bone の歴史と現状、および、登録局について 1.9節でまとめる。IPsec に関連する研究成果として、1.10節で IPsec の相互接続性実験について報告し、1.11節では鍵交換の相互接続性実験の結果を述べる。最後に、1.12節で v6 分科会の年間行事を時系列にまとめる。

1.1 始点アドレス選択

始点アドレスの選択は IPv6 を構成する技術の中で、重要な機能の 1 つであることが WIDE 6bone と呼ばれる IPv6 実験ネットワークの運用を通じて分かった。この節では、アドレス・スコープ、マルチホームでの運用、および、リナンバリングといった様々な IPv6 ネットワーク状況で発生する問題点を説明した後、それらの複合環境下で相互接続性を提供する始点アドレス選択アルゴリズムを提案する。

1.1.1 アドレス・スコープ

IPv6 では複数のスコープを用意し、異なるスコープのアドレスを複数個ネットワーク・インターフェイス (以下、IF と略記) に割り当てることを設計段階から考慮していた。

インターネット全域での一意性が保証され、パケットの到達可能性があるのはグローバル・アドレスである。この他に、サイト内にしか一意性、到達可能性がないサイトローカル・アドレスや、同一データリンク内にしか一意性、到達可能性がないリンクローカル・アドレスもある。

実際に通信する際には、始点と終点のアドレス・スコープが異なっている場合には、片方向の到達可能性しか得られないという問題が発生する。我々はこれに対して、終点アドレスと同じアドレス・スコープを持つ始点アドレスを選択するべきであると提案した [53]。終点アドレスより広いアドレス・スコープを持つ始点アドレスを選択すれば双方向の到達性は確保される。しかし現実の運用では、アドレスの安定性などを考慮すれば、無闇に広いアドレス・スコープのアドレスを選択することは問題がある。そのため、始点と終点のアドレス・スコープは同じにするべきである。

1.1.2 マルチホームでの運用

マルチホーム環境での IPv6 の経路制御は、IPv4 と同様の問題を抱えている。以下の図を例にとって説明する。図中、S、P、Q は組織、線は回線を意味する。

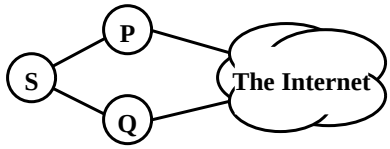

図 1.1: マルチホームの例

S という組織が P、Q という 2 つのプロバイダに接続し、アドレス P:S、Q:S が与えられているとする。マルチホームの組織 S は以下の要求がある。

1. 往復パケットは同じ回線を通過させたい — インターネット上のあるホストに、S のホストが通信を始めたとする。外向きのパケットが S–P の回線を通過するとすれば、内向きのパケットも S–P を通過させたい。そのためにはパケットの始点アドレスには Q:S ではなく、P:S を選択する必要がある。

2. 回線を有効活用したい — S–P と S–Q の回線を両方とも有効に利用したい。どちらか一方の帯域だけが専有されるのは避けたい。

この問題を部分的に解決する方法として、終点と最長一致する始点を選択することが挙げられる。最長一致は、始点と終点がアドレス的に近いことを意味する。現在のインターネットは、アドレスとトポロジーに強い束縛関係があるので、最長一致は始点と終点がトポロジー的に近いことを意味することが多い。トポロジー的に近い方を選ぶと、効率的な経路制御になるのは明らかである。たとえば上記の例で最長一致を利用すると、P へのホストに対する通信には始点 P:S が選択されることになる。

1.1.3　リナンバリング

ある組織をリナンバリングするときには、できるだけ既存の通信に影響を与えないように、緩やかに移行する必要がある。WIDE 6bone では度重なるリナンバリングの結果、緩やかなリナンバリングの運用経験を蓄積できた。これは、IPv6 では 1 つの IF に対して複数のアドレスを登録できるという特徴を利用したものである。リナンバリングの大まかな運用手順を以下に示す。

1. ルータに対して新アドレスを追加する。旧アドレスは消去してはならないが、始点アドレスとして選択しないようにする。
2. ホストに対して新しいルータ通知メッセージを送る。古い情報は送出せずにタイム・アウトして消えるのを待つ。
3. 新しい経路を組織外に対してアナウンスする。古い経路のアナウンスも継続する。
4. ネームサーバの設定を変更する。正引きに対しては旧アドレスを消去し、新アドレスだけを設定する。逆引きに関しては新旧とも登録しておく。
5. インターネットで名前情報が更新されるのを待つ。およそ数日あればよい。
6. 古い経路のアナウンスをやめる。
7. ルータに登録している旧アドレスを消す。
8. 古い逆引き情報を消す。

この手順をさらに考察して、可能な限り自動的にリナンバリングを進める方法についても提案していきたい。

1.1.4　提案する始点選択のアルゴリズム

以上の状況を組み合わせることにより、我々が提案する始点選択のアルゴリズムは以下の通りである。

1. 始点アドレスが確定しているときはそれを用いる
2. 終点アドレスに基づいて経路表を探索し、出力 IF を定める。その IF に設定されているアドレスの組が始点アドレスの候補である
3. エニーキャストとマルチキャストを候補からとり除く
4. リナンバリングのための旧アドレスを候補からとり除く

5. 終点アドレスとスコープの異なるアドレスを候補からとり除く
6. 候補にアドレスが 1 つ以上残っていればその中から始点アドレスを決定する。
7. もし、候補が 1 つも存在しなければ残りの IF に設定されているすべてのアドレスを候補として (3)-(6) のアルゴリズムを繰り返す。
8. 候補が 1 つもなければエラーを返す

現在最終的に複数の候補が残った場合は、終点との最長一致によって始点を決定している。

1.2 エニーキャスト

エニーキャスト通信とは、資源探索の方法の 1 つである。エニーキャスト通信を使えば、複数のサーバの中から最も近いサーバと自動的に通信可能となる。この節では、エニーキャストの実現に向けて通信方法とその問題点について議論する。

1.2.1 エニーキャストの利点

エニーキャストが実現されると、以下のような利点がある。

- サーバ群の抽象化 — 複数のサーバを 1 つのアドレスに抽象化する。
- 最短サーバの自動選択 — 複数のサーバの中から最も近いサーバを自動的に選択する。
- サーバの冗長化による信頼性の向上 — 複数のサーバを用意することで、1 つの場合に比べて信頼性が向上する。
- サーバの冗長化による負荷分散 — クライアントからの距離により選択されるサーバが変化するので、自動的にサーバへの負荷が分散される。

1.2.2 データグラム指向のエニーキャストの実現

エニーキャストでサービスを提供するサーバは、そのサービスに割り当てられたアドレスを受信できるよう設定される。つまり、あるアドレスを持つサーバが複数存在することになる。サーバは、このエニーキャスト・アドレスを送信時に始点として選択してはならない。

複数のサーバの内、どのサーバがパケットを受信するかは、経路制御と近隣探索プロトコルで決定する。異なるサブネットに属するサーバが存在した場合は、経路制御からみて最も近いサーバにパケットが届く。同一サブネット内に複数のサーバが存在する場合は、近隣探索プロトコルに早く答えた方にパケットが届けられる。

このように、エニーキャストにより選択されるサーバは経路制御に強く依存している。UDP などデータグラム指向のトランスポートを用いて、1 回限りのサービス要求/応答によりサービスを提供する場合は、この経路制御への依存性は特に問題とならない。ただし、NFS のように UDP を使って何度も要求/応答を繰り返す場合は、サーバの一意性欠如が問題となる。

1.2.3　コネクション指向のエニーキャストの実現

前述のように、エニーキャスト通信は経路制御に依存しており、通信時の経路制御によって届くサーバが異なる。このサーバの一意性の欠如は、コネクション指向の通信を実現する上で大きな問題となる。

TCP ではコネクションを開始するパケットが明確に特定できる。そこで、TCP ではコネクション開始時にサーバを特定することだけにエニーキャストを利用し、その後はユニキャストを用いて通信する方法が考えられる。これには以下の 2 つの実現方法が考えられている。

(1) Jim–Pedro 方式：サーバが、コネクション確立要求パケットの終点がエニーキャストであることを認識し、応答時にエニーキャスト・アドレスを IPv6 オプション・ヘッダに含めて返送する。クライアントは、相手からの応答の中に IPv6 オプション・ヘッダが存在し、そしてその中身として、確立中のコネクションの終点アドレスが含まれることを認識し、終点がエニーキャスト・アドレスであったことを知る。そこでコネクションの終点アドレスを、相手からの応答の始点ユニキャスト・アドレスへと変更する。

```
クライアント ----Syn-----> サーバ     始点=ユニキャスト、終点エニーキャスト
クライアント <--Syn,Ack--  サーバ     始点=ユニキャスト、終点ユニキャスト
                                      ＋オプション（エニーキャスト）
```

(2) Onoe 方式：クライアントは Syn を受信したときに限り、Ack を ISS と比較することによりセッションを決定する。そして対応するセッションの当初の終点であるエニーキャスト・アドレスと、受信した Syn の始点であるユニキャストが互いに対応関係にあるものと学習する。以降はエニーキャスト・アドレスに対応するユニキャスト・アドレスを終点アドレスとして用いる。

```
クライアント ----Syn-----> サーバ     始点=ユニキャスト、終点エニーキャスト
クライアント <--Syn,Ack--  サーバ     始点=ユニキャスト、終点ユニキャスト
```

実装としては、以下のようになる。

- 通常通り PCB を検索する

- みつからない場合
  anycast -> unicast の場合、みつからないはずである。

  そこでさらに、

  ```
  (th_flags & (Syn|Ack)) == (Syn|Ack)
  ```

  の場合、Syn_SENT の中から、通し番号が一致するものを探す。

- みつからなければ Rst

- みつかれば、PCB のリモートアドレスを始点で置き換える。

- Syn が再送された場合、PCB は書換え済なので、最初の検索でみつかる。

となる。

ただし、厳密には上記 2 項目目の処理で曖昧さをなくすためには、ISS を比較しなければならない。したがって動的なコネクション確立時に Syn + データを配送できなくなる。Jim–Pedro 案と比較すると、技巧的な手法といえる。

1.2.4 エニーキャスト通信の問題点

ここでは、エニーキャスト通信を実現する上で解決すべき問題点をまとめる。

- エニーキャスト・アドレスの認識— エニーキャストは通常のユニキャストと書式が同じであり、区別できない。そこで、サーバにエニーキャストを設定する場合は、このアドレスがエニーキャストだと知らせるなんらかの仕組みが必要になる。

- 経路制御通知— エニーキャスト・アドレスに対する経路情報を誰がどうやって流すのか検討する必要がある。

- 経路情報の急増— エニーキャスト・アドレスに対する経路情報は、ホスト経路となるので、むやみに流すと経路情報を急増させる。よって、組織内に閉じた経路情報にするなどの工夫が必要となる。

- 最適な通信相手選択— エニーキャストで選択されたサーバは、その時点で一番近いサーバであるが、経路の変更があった場合一番近くではなくなるかもしれない。このような効率の問題も検討する必要がある。

- エラー・ケースへの対処— エニーキャスト通信でどのようなエラーが生じるか検討し、対処していく必要がある。

1.3 IPv6 over ATM PVC

ATM ネットワークは広域接続の通信基盤として国際、国内を問わず急速に発展している。特に、国内では T1 より広い帯域で遠隔地を接続する場合、ATM 専用線の利用は一般的となってきている。しかしながら、IPv6 を ATM ネットワーク上で利用するための標準化は未だなされていない。このような状況において、IPv6 over ATM を標準化し相互接続性を確保することは重要である。本節では v6 分科会が IETF に提案した IPv6 over ATM 仕様の概略、および Hydrangea での実装と相互接続実験の結果について述べる。なお、v6 分科会が IETF IPng 分科会に対して提案した直後に、別のグループから IETF ION (IP over NBMA network) 分科会に同様の内容を含んだ提案がなされた。IPng 分科会の代表者を交えた話し合いの結果、IPv6 over ATM の標準化は ION 分科会が担当し、v6 分科会が提案した仕様すべてを ION 分科会の提案に含めることとなった。

1.3.1 仕様

この節では、v6 分科会から IETF に提案した IPv6 over ATM の仕様について説明する。この仕様は分科会のメーリング・リストと合宿 BOF での議論、およびそれにしたがってなされた実装経験を基に策定された。本仕様は 1997 年 11 月と 1998 年 2 月の 2 度にわたり IETF の IPng 分科会へインターネット・ドラフト (以下、ID と略記) として提案され、最終的に ION 分科会が作業を進めている IPv6 over NBMA network 標準化の一部として採り入れられることとなった。

本仕様では、適用範囲を PVC による Point-to-Point (以下、P2P と略記) 接続に制限することで、IPv6 マルチキャストへの対応など ATM 特有の問題を回避し、早期の相互接続性確保を目標としている。

本仕様の主な内容は、以下の 4 点である。

- パケットのカプセル化
- MTU
- IF ID
- 近隣探索

パケットのカプセル化は AAL5[54] にしたがい、プロトコル ID としては Ethernet と同様 0x86DD を用いる。したがって、カプセル化は図 1.2のようになる。

IPv6 のユニキャストとマルチキャストはカプセル化のレベルでは区別しない。この区別は IPv6 層でなされる。

本仕様では、原則としてヌル・カプセル化は認めない。ヌル・カプセル化は、ネットワーク層として IPv6 のみを用いる環境でなければ有効に働かない。しかし、現状では IPv4 と

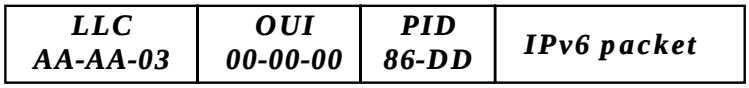

図 1.2: AAL5 による IPv6 パケットのカプセル化

IPv6 を併用することが多いと考えられるため、併用環境で不都合を生じる恐れのあるヌル・カプセル化を用いないこととした。

本仕様における MTU サイズは文献 [55] にしたがって 9180 バイトとする。

IF ID については、基本的には文献 [56] にしたがう。すなわち、ATM NIC の MAC アドレスを基にして 64 ビットの ID を生成する。1 枚の NIC によって複数の論理 IF を実現している場合には、それらの論理 IF 間で同一の IF ID を用いることになるが、IF ID はノード内で一意である必要はないため、これは問題にはならない。

ATM のドライバの実装によっては、MAC アドレスの取得が不可能、あるいは困難な場合もある。そのような場合を考慮し、本仕様では他の IF に MAC アドレスが存在する場合は、それを利用できるよう定めている。

近隣探索に関しては、相互接続実験の経験を踏まえて、以下のような若干特殊な仕様となっている。

- 近隣要請メッセージを受けたノードは応答として近隣通知メッセージを返さなければならない。
- いかなるメッセージにおいても、データリンク層アドレス・オプションは無視する。このオプションが含まれていない場合、または未知の書式でこのオプションを受けとった場合にも、メッセージそのものは受理しなければならない。

v6 分科会内で実施された相互接続実験では、近隣要請メッセージへの応答がないために通信を継続できないケースや、データリンク層アドレス・オプションが含まれていないために近隣要請あるいは近隣通知メッセージが破棄され、それによって通信が継続できなくなるケースなどがみられた。そこで、近隣探索プロトコルの仕様のうち、P2P リンクにおいて最低限必要と考えられる部分集合を定め、その中で特に相互接続性の観点から重要と思われるものを仕様として明記した。

1.3.2 Hydrangea での実装

v6 分科会ではワシントン大学の Charles D. Cranor によって開発、公開された FreeBSD 用 ATM デバイス・ドライバを基として、IPv6 over ATM PVC を Hydrangea に実装した [57, 58, 59]。

本実装の基となったオリジナルのドライバおよび Hydrangea がサポートしている ATM NIC を以下に示す。

- Adaptec 社製 ANA-5930、ANA-5940、ANA-5930A、ANA5940A
- Efficient Network 社製 ENI-155p-S、ENI-155p-C

オリジナルのドライバですでに実現されていた機能は以下の通りである。

- IPv4 over PVC ATM [54]
- Logical Internet Subnet [54]
- FreeBSD のサポート

Hydrangea で新たに追加された機能は以下の通りである。

- IPv6 over PVC ATM
- ATM セル・レベルでのシェィピング機能
- IF ID
- P2P IF への対応とマルチキャスト、ブロードキャストのサポート
- BSD/OS のサポート
- BPF による ATM セルの取り扱い (受信のみ)

1997 年 9 月 8 日〜 11 日に開催された WIDE 秋合宿中に実施された相互接続試験の結果、Hydrangea と以下の組織の IPv6 over ATM PVC 実装との間で相互接続性が確認された。

- 株式会社日立製作所
- 株式会社東芝
- 富士通株式会社

また、10 月に開催された北陸でのワークショップにおいて、同様の試験を実施した。

1.3.3 6bone における運用

WIDE 6bone は当初 WIDE バックボーン中で IPv4 用に使われている専用線の帯域を多重化装置によって分割し、IPv6 にも割り当てることで敷設されていた。現在、WIDE バックボーンの一部は専用線から ATM に移行している。そこで、WIDE 6bone の一部も ATM へ移行する必要があった。ATM は論理的な多重化装置であるので、ATM スイッチがすでに設置されている環境では新たに装置を導入することなしに IPv6 のための帯域を確保できる。

前節で述べた実験の後、10 月 8 日に奈良先端—北陸先端間、および、奈良先端—東大間に PVC を張り、IPv6 に割り当てて運用を開始した。実装としては Hydrangea が用いられている。これらの運用を通じて、Hydrangea での IPv6 over PVC の安定性が確かめられている。

1.4 PPP

この節では PPP の IPv6 対応の実装について述べる。対象としては、FreeBSD2.2.5 版、および、2.2.6 版のユーザ空間で実装されている IIJPPP と BSD/OS のカーネル空間で実装されている PPP である。

1.4.1 PPP の複数プロトコルへの対応

PPP では P2P リンク上で複数のプロトコルのパケットをカプセル化可能である。また、リンク固有の情報と、各プロトコル固有の情報を交換し設定する機能がある。これらは、それぞれ LCP(Link Control Protocol)、NCP(Network Control Protocol) と呼ばれ、NCP はさらにプロトコルごとに区別される。IPv4 と IPv6 の NCP はそれぞれ、IPCP、IPV6CP と呼ばれる。IPV6CP の仕様および、IPv6 パケットをカプセル化するための仕様は、IP Version 6 over PPP[60] で規定されている。

IIJPPP では、すべてのプロトコルに共通の処理と、IPv4 に依存した処理の切りわけが不十分だったため、モジュールを整理した。すべてのプロトコルに共通の処理を行なうモジュールと、IPv6 固有の処理を行なうモジュールを新規追加し、IPv4 固有の処理は IPv4 のモジュールのみにまとめるようにした。

IIJPPP に関しては、さらに重要な変更が必要であった。IIJPPP では、PPP のネゴシエーションは tty との間でパケットをやり取りすることにより、アプリケーション・レベルで処理している。しかしデータ・パケットに関しては、tun ドライバと tun IF という特殊な実装を用いてカーネルとの間で受け渡す仕組みとなっている。PPP プロセスが回線からデータ・パケットを受信するとそのパケットを tun ドライバに対して書き込む。これがカーネル内部的には tun IF から受信したパケットとして受け渡され、他の IF と同様に処理される。逆にカーネルが tun IF に送信したパケットは、PPP プロセスが tun ドライバ

から読み込み、tty に対して書き込まれ、回線へと送信される。

読み込みや書き込みの際には、プロトコル種別を受け渡す方法はなく、従来は IPv4 に決め打ちで処理されていた。しかし IPv6 の追加に伴い、プロトコルを識別できる機構の拡張が必要となった。そこで以下の方針にしたがってプロトコル識別のための仕組みを実装することとした。

- 既存実装との互換性 — 既存の IIJPPP のソース、できればバイナリが、拡張された tun ドライバと tun IF を備えたカーネル上で動作できること。拡張された IIJPPP のソース、できればバイナリが、既存の tun ドライバと tun IF を備えたカーネル上で動作できること。

- 将来的な拡張性、保守性 — 新たなプロトコルを追加する際も、tun ドライバと tun IF の修正が必要最小限になること。

- 拡張に伴うオーバーヘッド — 拡張の結果生じる通信処理のオーバーヘッドはできる限り小さくすること。

具体的な方法としては、tun ドライバを経由してアプリケーションとカーネルの間でやり取りするデータに対して、IPv4 以外のプロトコルの場合には 4 オクテットのフィールドを先頭に追加し、そこに sys/socket.h で定義されるアドレス・ファミリ (以下、AF と略記) の値を埋め込むことにした。IPv4 の場合は従来と同様、パケット・データをそのままやり取りする。

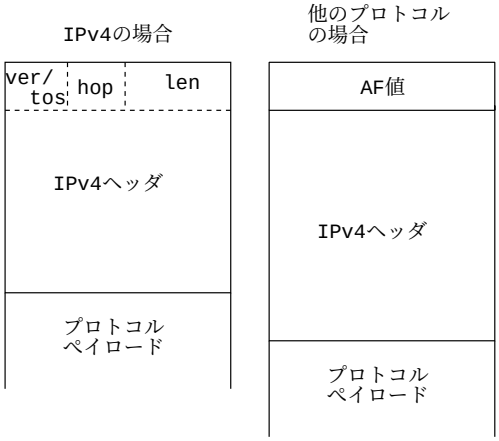

図 1.3: AF の追加

アプリケーションとカーネル間のデータのやり取りにおいて、受信側でのプロトコル種別の判別方法としては、受信データの先頭 4 バイトの値を符合無し整数とみなし、その値

で判断する。値が 255 を超えていれば IPv4 パケットとみなし、255 以下であれば先頭 4 バイトに AF 値が追加された IPv4 以外のパケットとみなすこととした。

この判別が正しく動作する理由は以下である。sys/socket.h で定義される AF の最大値は 255 である。AF は sockaddr 構造体の sa_family フィールドに代入されて使用されるが、この sa_family フィールドのサイズは 4.4BSD から 1 オクテットとなったため、今後も 255 を超えることはない。一方、IPv4 の場合はデータの先頭が IPv4 プロトコル・ヘッダの先頭 4 バイトとなる。この部分は先頭 1 オクテット部分にバージョン値、TOS 値として何らかの値が設定されており、末尾の 2 オクテット部分にも IP データ長として値が設定されている。仮にこの部分を 4 バイトの符合無し整数とみなした場合、現在一般的に使用されているビッグ・エンディアン、リトル・エンディアンのいずれのバイト順序においても決して 255 を下回ることはない。したがって、先頭 4 バイトが 255 を上回るか否かによって、IPv4 パケットか否かを判断することが可能である。

BSD/OS の PPP は、LCP、NCP はユーザ空間で、データ・パケットはカーネル空間で処理するように実装されている。カーネル部に関しては、複数のプロトコルのパケットを並列に処理できるように元から設計されているので、IPv6 への対応は容易であった。回線から受け取ったパケットを適切に IPv6 入力部、もしくはユーザ空間に送るコードを追加しただけである。

しかし、ユーザ空間の実装は単一プロトコルしか考慮されておらず、これを複数のプロトコルに対応する必要があった。具体的には IIJPPP とほぼ同様で、まず、各プロトコルに共通の処理と IPv4 に依存した処理に切りわけ、モジュールを整理した。次に IPv4 に依存した処理を拡張する形で、IPv6 の処理部を作成した。また、一旦 IPCP のネゴシエーションが成立した後は、リンクが切断するまで待ち状態に入る構造になっていたが、IPv4 と IPv6 の状態遷移が並列に動作するようにプログラム構造を変更した。

1.4.2 PPP における IPv6 アドレス割り振りに関する配慮

IIJPPP の IPv4 では、PPP リンクの確立と共に PPP リンク上の IPv4 アドレスが有効となり、リンク切断と共に無効とされる。また、PPP リンク上の IPv4 アドレスは、IPCP でのネゴシエーションの結果、割り振られる。

一方、IPv6 の仕様としては、リンク固有のリンクローカル・アドレスが定義されており、リンク内で一意の値を持つ IF ID を元にして自動的に設定される。

そして、Hydrangea の実装としては、すべての IF のリンクローカル・アドレスは、その回線種別に関わらずシステムの初期化時に一律で割り振られることとなっている。また、基本的にシステムの動作中に削除されることは想定されていない。IPv6 ホストのグローバル・アドレスはルータがホストに対して定期的に通知するルータ通知メッセージに含まれるプレフィックスと、IF ID を組み合わせ、やはり自動的に生成される。したがって、IIJPPP 従来のアドレス割り振り、削除動作とは異なる部分がある。

このため、IIJPPP の IPv6 アドレスに関する処理については、IPv4 と異なる拡張を施し

た。具体的には以下の通りである。

- PPP リンク切断後もリンクローカル・アドレスは削除しない。
- 静的に設定したアドレスを学習しておき、PPP リンク確立時には、静的設定されたアドレスのみを削除することとした。ルータ通知メッセージによって自動学習したアドレスに関しては、PPP リンク切断の結果、ルータ通知メッセージを受信しなくなれば、カーネル内のタイマ監視によって自動的に削除される。

1.5　アプリケーション

この節では、IPv6 に対応させたアプリケーションについて解説する。1.5.1 節では、World Wide Web(WWW) の代理サーバについて述べる。また、tcpd と qmail について、それぞれ 1.5.2 節と 1.5.3節で触れる。

1.5.1　HTTP 代理サーバ

6bone における IPv6 通信実験の進展に伴い、デュアル・スタック・ホストの数が増加している。今後 IPv6 がより普及していけば、IPv6 専用ネットワークに接続された IPv6 専用ホストが出現する。IPv6 専用ホストは IPv4 の接続性を持たないため、IPv4 ネットワーク上に提供されている種々の資源に直接利用できない。特に、WWW[61] は近年実社会に欠かせない存在となっており、IPv6 専用ホストから IPv4 ネットワーク上に提供されている WWW 上の資源をアクセスする手段を提供することは急務の課題である。このための手法としては、IPv4-v6 トランスレータを利用する方法など種々があるが、ここでは HTTP 代理サーバの利用について述べる。以下、サーバ、クライアント、代理サーバといった場合は、すべて HTTP 用を意味する。

代理サーバとは、HTTP[62] により通信するサーバとクライアント間の通信を仲介するためのサーバである。代理サーバは、サーバに対してクライアントのような振舞をし、クライアントに対してサーバのような振舞をするプログラムとして実装されている。HTTP は TCP 上のプロトコルとして実現されているが、TCP 接続はクライアントから代理サーバへの接続と、代理サーバからサーバへの接続の計 2 本が別々に設立される。

IPv6 専用ネットワークに接続されたホスト上のクライアントから、IPv4 ネットワーク上のサーバに HTTP により接続するためには、デュアル・スタック・ホスト上で動作する代理サーバを用意すればよい。この際、クライアントから代理サーバまでは IPv6 を利用した接続、代理サーバからサーバまでは IPv4 を利用した接続となる。

このために、我々は Apache[63] というサーバ／代理サーバ・サーバを改造し、IPv4 と IPv6 の両方を取り扱えるようにした。改造版 Apache は代理サーバとして動作する際、(1) クライアントからの接続を IPv4/v6 のいずれか一方で待ち受けることができ、(2) IPv4/v6 両方のサーバへの接続が可能である。

改造版 Apache では、起動時オプションにより IPv4 による接続を待機するか IPv6 による接続を待機するかを切替え可能である。

```
# httpd          IPv4 による接続を待ち受ける (デフォルト)
# httpd -4       IPv4 による接続を待ち受ける
# httpd -6       IPv6 による接続を待ち受ける
```

これにより、IPv6 専用ホストに対するプロトコル変換サービスの提供と IPv4 専用ホストに対するそれとを切替え可能である。必要であれば、単一のデュアル・スタック・ホスト上で両方の設定の改造版 Apache を実行できる。

改造版 Apache はクライアントの指示にしたがい、IPv4 ホスト上で動作するサーバ、もしくは IPv6 ホスト上で動作するサーバのいずれにも接続できる。クライアントからの要求は、たとえば「`GET http://www.wide.ad.jp/`」などのように、URL を用いて代理サーバに送られる。この際、URL のホスト部 (上の例では `www.wide.ad.jp`) が DNS により IPv4 アドレスに解決されれば、改造版 Apache は IPv4 を用いてサーバへ接続する。ホスト部が IPv6 アドレスに解決されれば、サーバへの接続には IPv6 が用いられる。

基本的な動作は以上述べたように単純だが、実装上の難点がいくつか存在する。1 つは設定ファイルなどにおけるアドレス空間の曖昧性である。Apache の設定ファイル中の各所にはホスト名が出現する。あるホスト名に対し、DNS データベース上に

```
foo.wide.ad.jp. IN A    127.0.0.1
                IN AAAA ::1
```

などのように IPv4 アドレスと IPv6 アドレスの両方が存在した場合、これをどちらと解釈すべきかは単純には判断できない。現在のところは改造版 Apache の動作モードにしたがって曖昧性を解消しているが、これが常に適切かどうかは議論の残るところである。もう 1 つがフィルタの記述である。一般に代理サーバでは、特定の IP アドレスの範囲からの代理サーバ自体へのアクセスの禁止や、「特定の IP アドレスの範囲については次の代理サーバに接続する」といった設定ができる。これにより、代理サーバが必要ない場合には代理サーバが利用されないようにし、プロトコルによる通信経路を最適化できる。ここで、(1) どのようにすれば改造前の設定ファイル書式を変えずに IPv4/v6 の両方のフィルタを提供できるか、(2) 曖昧な場合はどうするのか、(3) 「IPv4 なら A、IPv6 なら B」のような記述はどうすべきか、などが問題となる。この点については検討中である。

今回の改良では、代理サーバとしてはあまり高機能でない Apache を改造したが、これは Apache のコードが比較的 IPv6 に対応しやすかったためである。代理サーバとしてより高機能なプログラムとして squid がある。しかし、squid のコードは`struct in_addr` が多様されているなど IPv4 に深く依存して記述されており、Apache に比べ IPv6 化しづらかったため今回対象としなかった。代理サーバは、WWW の閲覧に用いられるトラフィックを削減するために IPv4 と IPv6 の区別なく重要な技術である。このため、squid についても早い時期に IPv6 に対応させたい。

1.5.2 tcpd

現在 IPv6 では実験に重きが置かれているいるため、IPv4 よりもセキュリティ意識が薄い組織が多い。しかしながら、6bone が広まるにつれ、6bone につながる組織やホストの数は増加し、IPv4 インターネットと同様にセキュリティ保全が重要になってくる。そこで、より安全なシステムを構築するためには、これまで IPv4 で使ってきた技術も取り込む必要がある。

IPv4 インターネットでは、IPv4 アドレスやポート番号を基にアクセス制御を実現する tcp_wrappers や xinetd というツールがある。これらは、サービスを提供するサーバ側において inetd から起動されるツール、たとえば telnet/ftp/finger 等へのアクセスを制限し、記録を残すために用いられている。

今回は、よく使う機能を中心に tcp_wrappers_7.6 を IPv6 へ対応させた。IPv6 アドレスを使うために、定義ファイルは別としている。

実現した機能は以下の通りである。

- libwrap.a, tcpd.h
  tcp_wrappers_7.6 の代りにリンク可能
  - struct requestinfo *fromhost()
  - int hosts_access()
- /usr/libexec/tcpd
  tcp_wrappers_7.6 の tcpd の代りに使用可能
- /etc/hosts.access アクセス制御を記述

```
"DEFAULT" | FLAGS
SERVICE | FLAGS
allow | CLIENTS [ | FLAGS [ | OPTIONS ] ]
deny | CLIENTS [ | FLAGS ]
以上の繰り返し
```

  - SERVICE = service 名 {,@{{inet#,inet6#,}HOSTNAME,IPv4/IPv6 アドレス }}
  - CLIENTS = {user@,}{{inet4#,inet6#,}{HOSTNAME,”ALL”,”KNOWN”,”.”DOMAIN}, IPv4/IPv6 アドレス {,/マスク }}
  - FLAGS アクセス・チェックに関する制御
    * ident を引くかどうか

    - 標準動作の有効/無効
    - syslog 先の指定
    - DNS の正引きと逆引きが不一致だった場合の動作の指定
  - OPTIONS allow 時の動作指定
    - setenv,uid 指定,gid 指定,umask 指定,nice 値変更
    - 起動プログラムの指定

現在は Hydrangea の ports に含まれて配布されている。また、以下の URL にて公開している。

```
http://www.rcac.tdi.co.jp/fujiwara/tcpd-v6-0.0-980106.tar.gz
```

1.5.3 qmail

メールを高速に配送するプログラムとして qmail がある。この節では、qmail を IPv6 に対応させることについて述べる。

qmail は、キューの管理を介して小さいアプリケーションを連係しメールを配送する。qmail におけるメールの受信では、まず inetd から tcp-env が SMTP のコネクションに対し起動される。tcp-env はアドレスを解釈する。この情報を基に、qmail-smtpd がキューイングを担当する。qmail-smtpd では、アドレスに依存した操作はなく、単なる read/write を実行している。そこで、tcp-env を IPv6 にも対応させたところ、SMTP/IPv6 でメールを受信できるようになった。具体的な処理の流れは、inet6d → tcp-env(IPv6 対応) → qmail-smtpd(無変更) → qmail-send(無変更) となる。

以下に localhost の場合と、自分自身のアドレスへ SMTP したときの Received ヘッダを示す。

```
Received: from localhost6 (0000:0000:0000:0000:0000:0000:0000:0001)
  by localhost6 with SMTP; 23 Apr 1998 14:15:47 -0000
Received: from spacecraft.f.rcac.tdi.co.jp (3ffe:0501:200b:00a0:02a0:c9ff:fe3d:615b)
  by spacecraft.f.rcac.tdi.co.jp with SMTP; 27 Apr 1998 10:48:15 -0000
```

qmail におけるメールの送信は、qmail-remote が担当している。そこで、DNS を引くための関数を変更して IPv6 アドレスを扱えるようにした。MX レコードをアドレスに変換する部分では、対応する IPv4 アドレスと IPv6 アドレスの両方を調べ、全部返すこととした。実際に配送する部分では優先順位順で処理するが同じ送り先に対し IPv6 アドレスの次に IPv4 アドレスを引くようにしたため、どちらのアドレスも持っている場合は IPv6 から先に試みる。その他の変更点としては、エラー時の記録を取る部分を IPv6 アドレスにも対応させたぐらいである。今回の実装では、リゾルバについては全く考慮せず、Hydrangea のリゾルバを利用している。

今回の実装のパッチを以下の URL にて公開している。

```
http://www.rcac.tdi.co.jp/fujiwara/qmail-1.01-v6.02+.diff
```

この直後に sendmail が IPv6 に対応し、qmail と sendmail とで SMTP/IPv6 でメールが交換できることを確認した。

また qmail では、外へメールを出すときは、メーラからの接続と外からの接続の区別を qmail-smtpd に知らせる必要がある。自分の MUA からの接続の場合は、環境変数 RELAYCLIENT を設定してやる必要がある。そのため、tcp_wrappers_7.6 のようなツールが必要になったので、前述の tcpd を作成した。

inet6d.conf には以下のように記述する。

```
smtp    stream  tcp     nowait  root    /usr/libexec/tcpd   \
    /var/qmail/bin/tcp-env /var/qmail/bin/qmail-smtpd
```

/etc/hosts.access には以下のように記述する。

```
tcp-env syslog local1.notice,defaultallow
        ALLOW ::1 || setenv RELAYCLIENT ''
        ALLOW 127.0.0.1 || setenv RELAYCLIENT ''
```

その他は IPv4 での qmail と同じである。

1.6　メール

メールは、インターネットにおける基本サービスであり、インターネット・プロトコルが IPv4 から IPv6 への移行期間においても、従来から利用されていたシステムはしばらくの間そのまま利用できなければならない。IPv6 分科会では、IPv6 に対応したメールの配送システムを実装するとともに、IPv4 から IPv6 への移行の際に問題を起こさない方式や移行手順を検討し提案していこうと考えている。特に、メールの配送システムでは、DNS の MX レコードを用いるという点において、通信の際に必要となる処理が他のアプリケーションと大きく異なっている。したがって、IPv4 と IPv6 とが共存する場合の DNS の MX レコードの解釈や問題を起こさない設定方法について明確にしておかなければならない。

1.6.1　sendmail の IPv6 への対応

現在インターネット上で広く利用されているメールの配送システムの 1 つに sendmail がある。sendmail は最新のバージョン 8.8.8 においても IPv6 をサポートしておらず、バージョン 8.10 以降で対応予定とのことである。IPv6 分科会では、メールを IPv6 で配送するために、sendmail で IPv6 をサポートするためのパッチを作成した。このパッチを適用した sendmail は、Hydrangea や v6d といったプラットホーム上で利用可能である。この sendmail は、IPv4 と IPv6 のデュアル・スタック対応となり、DNS の MX レコードが示

すメール受信ホスト (mail exchanger) に対して、IPv4 アドレスが対応付けられていればIPv4 上で SMTP 接続を試み、IPv6 アドレスが対応付けられていれば IPv6 上で SMTP 接続を試みる。また、両方のアドレスが対応付けられている場合は、まず IPv6 による接続を試みて、接続できない場合は IPv4 による接続を試す。

SMTP を用いてメールを送信した場合、そのメールが IPv4/IPv6 のどちらのプロトコルを利用して送られてきたかの区別がつきにくい。IPv4 と IPv6 の混在環境におけるデバッグでは、どちらのプロトコルが利用されたかが容易に判別できることが重要である。そこで、sendmail の IPv6 拡張では、IPv6 上の SMTP を経由して送られてきたメールに対しては、以下に示すように Received: ヘッダの経由プロトコルを示す部分 (with) に ESMTP/IPv6 (または、SMTP/IPv6) という文字列を明示的に記録するようにしている。また、Received: ヘッダの発信元ホストを示す部分 (from) に現れる IP アドレスも、IPv6 のものとなる。

```
Received: from sh.wide.ad.jp (daemon@[3ffe:501:1000::1])
        by onoe2.sm.sony.co.jp (8.8.7+IPv6/Sony6.1MX)
        with ESMTP/IPv6 id AAA02786; Fri, 3 Apr 1998 00:08:09 GMT
```

1.6.2 smtpfeed の IPv6 対応

WIDE プロジェクトでは、メーリング・リストにおけるメールの配送を高速化するために、smtpfeed と呼ばれるプログラムを開発している。smtpfeed は sendmail から呼び出す外部メーラ型の実装で、sendmail のメール配送処理を高速化する。WIDE プロジェクトのメール・サーバでは、sendmail および smtpfeed を用いて、各分科会等のメーリング・リストのメール配送処理を高速化している。

当初、この smtpfeed のメール配送ルーチンは IPv4 のみに対応していたが、これを IPv6 に対応させ、IPv6 分科会のメールの配送を IPv6 を用いて高速に配送できるようにした。smtpfeed は GNU の autoconf で作成された configure を用いてコンパイル環境を調査する。IPv6 をサポートした smtpfeed を生成する場合は、configure の実行時に `--enable-ipv6` オプションを指定する。

1.6.3 IPv6 移行時のメール相互運用性に関する調査

IPv6 への移行は IPv4 と IPv6 の混在する環境を経ることになるが、両プロトコルが混在する環境下では、ネームサーバに IPv4 アドレスと IPv6 アドレスを混在させて登録する。このような状況においても、従来から利用されてきている IPv4 のみを対象としたメール配送システムの実装は引き続き問題なく利用できなければならない。しかしながら、先行して IPv4 と IPv6 の両アドレスをネームサーバに登録しているドメインに対してメールを送ろうとした場合に、エラーが発生してメールが届かなかったという報告が寄せられている。残念ながら実装の種類など詳細については情報不足のため不明である。

そこで IPv6 分科会では、このようなメールの配送の信頼性にかかわる問題について、問題を持つ実装の特定および、その実装がどの程度利用されているのか、ということについて調査した。調査期間は 1998 年 1 月 16 日から 1 月 31 日までとした。調査に協力頂いたのは、ip-connection メーリング・リストに参加している管理者の方々である。調査手順を以下に示す。

1. 準備

   (a) sh.v6.wide.ad.jp に対して、IPv4 のアドレスを示す A レコードと、IPv6 のアドレスを示す AAAA レコードの両方を定義する。

```
sh.v6.wide.ad.jp.       IN A    203.178.137.73
sh.v6.wide.ad.jp.       IN AAAA 3ffe:501:1000::1
```

   (b) sh-mx.v6.wide.ad.jp には、IPv6 のアドレスのみを持つ sh-v6.v6.wide.ad.jp を第 1 位 MX とし、sh.v6.wide.ad.jp を第 2 位 MX とする MX レコードを定義する。

```
sh-mx.v6.wide.ad.jp.    IN MX   10 sh-v6.v6.wide.ad.jp.
sh-mx.v6.wide.ad.jp.    IN MX   20 sh.v6.wide.ad.jp.

sh-v6.v6.wide.ad.jp.    IN AAAA 3ffe:501:1000::1

sh.v6.wide.ad.jp.       IN A    203.178.137.73
sh.v6.wide.ad.jp.       IN AAAA 3ffe:501:1000::1
```

   これら 2 つのアドレスに対してネームサーバに問い合わせると、どちらについても AAAA レコードを含む応答がネームサーバから送り返されてくる。IPv4 を前提にした実装の中には、この AAAA レコードのような実装当時に予期できなかったレコードが送り返されてきた場合に、正常にメールが送れなくなるものがあるかもしれない。また、後者のアドレスについては、第 1 位の MX が AAAA レコードのみを持つホストであるため、IPv4 のみで検索すると MX の指すホストがアドレスを持たないことになり、配信不能として扱われる可能性がある。

   特に、後者の sh-mx.v6.wide.ad.jp の定義形態は、IPv6 への移行が進んだ状況下において、IPv6 のみをサポートしているメール・サーバをプライマリのメール・サーバとして利用し、IPv6 と IPv4 の両方をサポートしたメール・サーバを IPv4 と IPv6 のプロトコル間中継用のセカンダリ・メール・サーバとして利用するような状況を想定しており、このような設定は、IPv6 への移行の進行にしたがって広く用いられるようになることが予想される。

2. 調査

(a) 各組織のメール・サーバから、以下に示す 2 つのメールアドレスにそれぞれメールを送ってもらう。

```
v6-dns-test@sh.v6.wide.ad.jp
v6-dns-test@sh-mx.v6.wide.ad.jp
```

(b) 送信したメールに対して、Host Unknown 等の配送エラーが発生した場合は、エラーの詳細について報告してもらう。報告の内容は以下の通り。

- エラー・メールの内容
- エラー・メールを発信したホストのメール配送プログラムの名前およびバージョン
- そのプログラムにリンクされている BIND ライブラリのバージョン
- そのホストが直接参照しているネームサーバのバージョン

3. 結果

調査では、1256 サブドメインの協力を得て、1380 通のメールが届いた。問題なくメールが配送された実装を表 1.1に示す。届いたメールのうち、発信者アドレスの設定が不適切で自動応答メールが返送できなかったものは 33 通あった。管理者からのエラー・レポートは 3 通送られてきたが、1 通はネットワークの一時的な不調によりメールがすぐに送られなかったことを指摘するものであり、異常な動作ではなかった。残り 2 通は、以下のそれぞれの実装からメールを送信した場合のエラーを報告するものであった。

- Post.Office version 1.9.3/bind 4.9.5
- sendmail 8.8.5/bind 4.9.5b1

前者のエラー・メッセージは “MX loopback to myself” であり、後者のエラー・メッセージは “Too many hops” であったが、後者も正しい設定がされていれば “MX loopback to myself” となるべきものであった。しかし同一の実装を用いる他の組織では問題は発生しておらず、また後の追試では問題が再現しなかった。これらの実装は確率的な問題を持っているようにも思われるが、さらなる調査の協力が得られなかったので、これ以上の詳細については不明である。

4. 結論

調査の結果、多くの組織では実際に問題なくメールを配送できることが確認できた。しかしながら、2 箇所からエラーが報告されていることもあり、完全に安心できるというわけではない。ただ、古い sendmail (5.61 以降) や bind (4.8.3 以降) であっても、特にエラーを招くような実装上の問題はみあたらなかったため、とりあえず問題はないものとして、今後の IPv6 への移行のための方針を検討することにした。

表 1.1: 問題なくメールが配信できた実装のリスト

実装	ホスト数
sendmail	997
smap/fwtk	57
VirusWall	15
qmail	12
Post.Office	11
Borderware	10
BlackHole	7
Generic SMTP handler	6
IMS	5
MS Exchange Internet Mail	5
Eudra Internet Server	5
Netscape Mail Server	4
Apple Internet Server	3
NT SMTP Server	3
Firewall-1	2
PMDF	2
Lotus SMTP MTA	1

配送エラーは、メールの発信側で発生するため、IPv6 アドレスをネームサーバに定義した組織側でエラーを検出することは不可能である。したがって、今後発生するかもしれない IPv6 への移行に関連するトラブルに備えて、各組織に問題の発生の可能性について十分な広報をしておく必要がある。

一方、無用なトラブルを避け、安全に SMTP/IPv6 の実験をする場合には、user@v6.domain といった IPv4 の世界で利用していないサブドメインを用意して、そこに IPv6 関係のアドレスを定義するとよいだろう。しかしながら、これはあくまでも実験的利用にしか対応できない方法である。

1.6.4　IPv6 分科会メーリング・リストの IPv6 による配送

前項の調査結果を受けて、WIDE プロジェクトのメール・サーバである sh.wide.ad.jp のメール配送システムを IPv4 と IPv6 の両方をサポートするものに移行させた。まず、IPv6 分科会メーリング・リストの配送については、前述の IPv6 に対応した sendmail および smtpfeed を起動することにより、配送先ホストが IPv6 アドレスを持っていれば IPv6 上の

SMTP によって配送し、IPv4 アドレスしか持っていなければ IPv4 上の SMTP によって配送するようにした。また、メールの受信についても、sh.wide.ad.jp に関するアドレスとして、IPv4 アドレスである 203.178.137.73 に加えて IPv6 アドレスである 3ffe:501:1000::1 を追加設定し、IPv4/IPv6 のどちらで送られてくるメールも受信可能とした。

sh.wide.ad.jp での IPv6 への対応は v6d を用いているが、v6d による実装では IPv4 と IPv6 の TCP ポート空間が独立しているため、sendmail プロセスはそれぞれのプロトコル用に 1 つずつ動作させておく必要がある。したがって、プロセス毎に生成される sendmail.pid ファイルはプロトコル毎に異なる名前あるいはパスにしておくことが望ましい。また、syslog への出力や ps の際の表示においてもどちらのプロトコルを扱うプロセスであるかが容易に判別できることが望ましい。さらに、メールの受信時に両デーモン・プロセスで異なる処理を施したい場合には、sendmail.cf や mqueue は分離しておくべきである。なお、いずれの場合でも、IPv4/IPv6 の両方のアドレスを持つホストでは、IPv4/IPv6 アドレスに対応付けられているホスト名や MX のドメイン名をすべて解釈できるように、忘れずに必要な名前をすべて定義しておくことが重要である。

1.7 FAITH

IPv4 から IPv6 へ緩やかに移行するためには、IPv4 と IPv6 の変換し直接の通信を可能にするトランスレータが役割を果たす。これまでのトランスレータの主流は、ヘッダ変換方式であった。

IPv6 と IPv4 間のヘッダ変換技術である SIIT は、IPv4 と IPv4 間のヘッダ変換技術である NAT と同様の問題点を含んでいる。たとえば、FTP のように IPv4 アドレスを直接交換するプロトコルでは、この IPv4 アドレスを IPv6 アドレスに変換する必要があるが、これは困難である。また、IPsec は本質的に変換不可能である。

さらに SIIT は、NAT にはなかった問題点を抱えている。たとえば、ICMP と ICMPv6 のそれぞれのセマンティクスは、完全な相互変換が不可能である。また、IPv6 ヘッダは IPv4 ヘッダよりも通常 20 バイト大きいため、IPv4 のバルク・パケットを IPv6 ヘッダに変換すると、分割せざるを得ない。

上位の層でアドレスの違いを吸収してやると、これらの問題は解決できる。たとえば代理サーバ方式は、アプリケーション・レベルでのリレーであり、確立するコネクションはそれぞれのネットワーク・プロトコルで閉じることになる。代理サーバは TELNET などの対話的なプロトコルに不向きなので、我々はトランスポート・リレー方式を採用した。方式自体を FAITH、実装したデーモンを faithd と呼ぶ。

現在 faithd は、IPv6 から IPv4 方向への変換が可能である。すなわち、TCP/IPv6 コネクションを終点の代わりに受け取り、新たに TCP/IPv4 コネクションを実際の終点に確立する。そして、この 2 つのコネクションを流れるデータの橋渡しをする。

faithd は、Hydrangea 上で動作するように設計されている。Hydrangea では、指定され

た IPv6 アドレス空間に合致する TCP/IPv6 パケットを、それが自分宛ではなくても上位層に渡すように改造されている。よって、faithd は実際には自分宛でない TCP/IPv6 コネクションを終点の代わりに受理できる。

faithd の機能を利用するには、IPv6 のクライアント側で IPv4 アドレスを IPv6 アドレスに変換する必要がある。これには、リゾルバで対応する方法と、ネームサーバで処理する方法の 2 つがある。

リゾルバで対応する方式では、指定された名前に対しアドレスを検索するが、A レコードしかなかった場合、あらかじめ指定された IPv6 のプレフィックスを連結し AAAA レコードとして返す。このプレフィックスは、faithd が動くデュアル・スタック・ルータで指定された IPv6 アドレス空間と一致している必要がある。このおかげで IPv6 クライアントは、実際には IPv4 ホストである終点をあたかも IPv6 ホストと思い TCP/IPv6 コネクションを開こうとする。このコネクションは、faithd が受け取り IPv6 アドレスに埋め込まれた IPv4 アドレスを取り出して、実際の終点に新たに IPv6/TCP コネクションを張る。

ネームサーバで処理する方法では、ある名前に対し AAAA レコードの解決が要求され、実際には A レコードしか存在しない場合に、上記と同様に AAAA レコードを作成し返答する。

リゾルバ方式では、利用する IPv6 クライアントすべてに拡張したリゾルバをリンクする必要がある。しかし、粒度が小さい分プロセス毎に動作を決定できる。ネームサーバ方式ではクライアントを変更する必要はないが、粒度が大きいためサイトの全体あるいは一部内のクライアントの動作を決定してしまう。

Hydrangea ではリゾルバ方式が getaddrinfo() を改良する形で提供されている。また、ネームサーバ方式は後述の newbie を導入すれば利用できる。

1.8　v6 DNS

この節では、IPv6 環境での名前サービスの意味、WIDE 内の実装、および WIDE 6bone での運用を述べる。

1.8.1　名前サービス技術開発の意味

IPv6 では、アドレスの直接利用は非現実的である。技術者などの熟練した技術を持つ利用者ならば、アドレスを直接取り扱うことができる。しかし実際、128 ビットのアドレスは利用しやすい形式ではない。したがって、IPv6 での名前サービス技術は IPv6 技術の利用しやすさと密接に関係している。

従来の名前サービス技術は、名前の問い合わせとその解決が中心である。しかし IPv6 での運用には、名前解決だけでは不十分な場合がある。名前データベースの動的更新 (Dynamic Updates in Domain Name System[64]) などの付加的なサービスが利用できると、管理の利便性は飛躍的に向上する。

1.8.2 WIDE での実装

newbie は、Hydrangea を念頭に置いた DNS の実装である。newbie は IPv6 のみではなく、名前データベースの動的な更新や、ラップトップなど動的に接続状態が変わるシステム上でのスタブ・リゾルバとしての動作などを考慮して実装した。

主な特徴は以下の通り。

- 動的なデータベース操作 — DNS UPDATE[64] に沿って外部からデータベースを動的に更新できる。このサービスを利用して、自動設定されたアドレスなどを名前データベースへ反映できる。また、セキュリティ上の問題から外部からの動的更新を利用できない場合も、サーバを直接操作することでサービスに影響を与えずにデータベースを動的に更新できる。

- セキュリティ機構 (プロトタイプ) — 外部からの動的更新を認証するための簡単なセキュリティ機構のプロトタイプを実装した。この認証方式はプロトタイプであり、Secure Domain Name System Dynamic Update[65] で規定されているプロトコルとの互換性はない。

- IPv6 の利用 — IPv4、IPv6 両方を利用した通信が可能である。また、IPv6 のみをデータ交換に利用し、既存の IPv4 を利用した名前サーバに影響を与えずに動作できる。これにより、実験的な利用や既存の環境を残したままでの FAITH との協調が可能である。

- FAITH との協調 — A レコードを FAITH で利用する方式で AAAA レコードに変換する。つまり、FAITH と協調利用する場合、管理者が FAITH で利用するプレフィックスを名前サーバに指定する。名前サーバは、IPv6 アドレスが求められたドメイン名に対して IPv4 アドレスのみを発見した場合、IPv4 アドレスに指定されたプレフィックスを付加して、IPv6 アドレスのようにみせかける。

- ネットワークが利用できない場合の動作 — 可搬計算機などの環境では、ネットワークに接続されていない場合、明示的に指定することで、動作を最適化できる。つまりキャッシュされていないデータに関して、接続性のないネットワーク上の名前サーバに問い合わせず、即座に接続拒否のエラーを返答する。

- 動的な forwarder(問い合わせ転送先) 設定 — 可搬計算機など、ネットワークへの接続状態が頻繁に変化する環境に応じるために、forwarder を動的に設定できる。DHCP で得られた名前サーバを forwarder に設定することで、キャッシュを効率的に利用できる。

newbie は、Hydrangea が導入された FreeBSD を想定して実装されている。ただし動的なデータベースを実現するために、Berekeley DB 1.86 の存在を前提としている。

newbie は、IMC(Internet Mail Consortium) 主催の DNSCONNECT で相互接続性を検証している。ここでは DNS UPDATE と DHCP との協調動作を試験し、1 つを除いたすべての実装と協調動作を確認し良好な結果を得た。

また、第 41 回の IETF での DNSIND 相互接続実験で、DNS UPDATE に関して検証し、他の実装との協調動作を確認した。

IPv6 での通信が可能なリゾルバは、Hydrangea や v6d に付属している。もともと v6d 用に開発されたライブラリを、Hydrangea での利用のために改良が加えられた。

IPv6 名前サーバと IPv6 リゾルバの組み合わせにより、IPv6 しか利用できないネットワーク上での名前解決が可能になる。

1.8.3　動作モデルの検討

ネットワーク層では IPv4 と IPv6 の空間は分離されているが、両者のアドレス情報、すなわちリソース・レコードは同じ名前空間に属する。しかし、アドレス情報の交換にはネットワーク層を利用した通信が不可欠である。ここで、ネットワーク層は 2 つの空間に分離されている。両方の空間に到達できる場合は、ネットワーク上のすべてのデータベースを利用でき、問題は発生しない。しかし、片方の空間にしか到達できない場合、つまり IPv4 と IPv6 の片方のプロトコルのみしか利用できないと、到達不可能な名前サーバに必要な情報が存在する場合がありうる。

このような問題を解決する一般的なモデルは提案されていない。現在は、IPv6 ネットワーク中で、IPv4 と IPv6 の両方のネットワークが利用可能なマシン上に名前サーバを置き、これをアプリケーション・レベルの変換・中継サーバとして利用している。newbie は、IPv4 と IPv6 を適宜使い分け可能なので、両方のネットワークへの到達性があれば必要な情報をすべて入手できる。したがって、IPv6 への到達性だけを持つクライアントは newbie を中継サーバとして利用して IPv4 空間にある情報を入手できる。

1.8.4　WIDE 6bone での名前サーバ運用

WIDE 6bone 上での名前サーバ運用には 2 つの側面がある。1 つは 6bone の名前空間の管理であり、もう 1 つは IPv6 を用いた名前問い合わせと解決である。

WIDE 6bone の名前空間の管理は、IPv4 を利用した既存の名前サーバ (ISC bind) を利用している。名前空間の構成は DNS Extensions to support IP version 6[66] にしたがっている。また、すべてのデータはゾーン・ファイルの編集によって変更される。つまり、現段階では外部からの動的な更新は、セキュリティ上の理由もあり利用していない。

また、IPv6 を用いた名前問い合わせに関しては、試験的に運用されている。WIDE 6bone 上の複数のサーバで、IPv6 TCP/UDP に対応する名前サーバが稼動しており、IPv6 での名前問い合わせに対応する。それぞれのサーバは IPv4 と IPv6 の両方の接続性を持つマシン上に設置され、中継サーバとしても機能する。また必要に応じて、FAITH と協調的に動

作する。

1.9 6bone と登録局

この節では、WIDE 6bone、6bone JP、6bone での統計、および、登録局について述べる。

1.9.1 WIDE 6bone

WIDE プロジェクトは、1996 年 7 月から世界的な IPv6 テストベッドである 6bone に参加している。6bone 内の WIDE プロジェクトの IPv6 ネットワークを WIDE 6bone と呼ぶ。WIDE 6bone 内の経路制御は、当初は静的な経路設定により実現されていたが、1997 年 2 月に RIPng を利用する経路制御デーモンを開発し、WIDE 6bone は動的な経路制御に移行した。しかし、WIDE 6bone の拡大にともなって、半年後にはノード間のホップ数が 10 ホップを超えるような場所も出現し、RIPng の運用限界が近付いていることが報告された。この時期には、6bone 全体の経路数は 300 近くに増加していた。

1997 年 10 月には IETF により新しく提唱された「経路集約型アドレス」に基づいて、6bone のリナンバリングを実施した。この際に、WIDE プロジェクトは、アジアで唯一 pTLA 割り当てを受けた。これ以降、WIDE バックボーンと WIDE 参加組織で構成された 3ffe:501::/32 のネットワークを WIDE 6bone と呼ぶようになる。WIDE 6bone は緩やかなリナンバリングの経験を積むために、9 月下旬から約 1ヵ月かけて移行した。

WIDE プロジェクト内では、このリナンバリング時に接続点単位でアドレスを割り当て、アドレスを図 1.4のように定めた。同時に、WIDE 6bone 登録局を立ち上げアドレス申請の窓口とし、接続組織のデータを WWW を通じて管理可能とした。

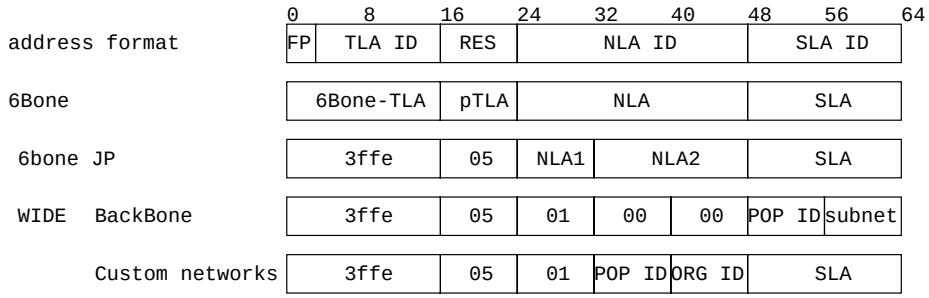

図 1.4: アドレス構造

1.9.2 6bone JP

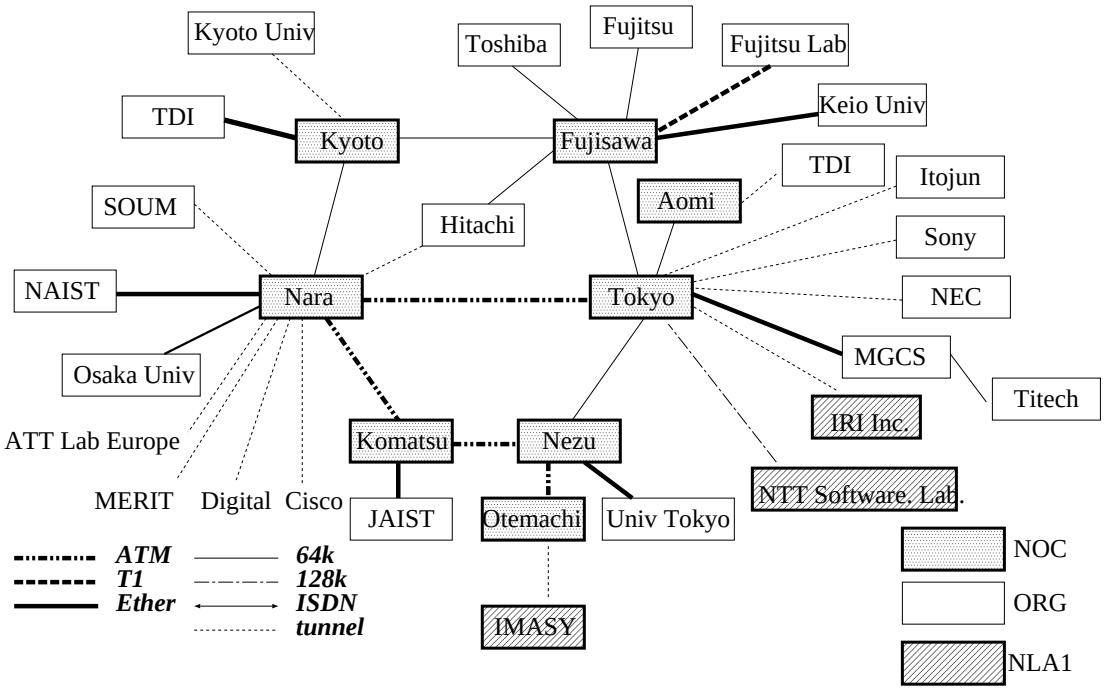

図 1.5: 6bone JP

WIDE プロジェクトでは、接続の受付を WIDE 参加組織に限っていたが、1998 年 1 月 1 日から WIDE 参加組織以外の接続受付を開始した。このネットワーク全体を 6bone JP と呼ぶ。このときから、WIDE 6bone は、6bone JP 内のネットワークという位置づけとなった。

6bone JP は順調な発展をとげ、1998 年 3 月現在、参加組織数 21(うち WIDE 参加組織 19)、NOC 数 9(表 1.2) と、世界でも最大規模の IPv6 ネットワークとなっている (図 1.5)。

6bone JP は、6bone に 4 つの海外組織へのトンネルを通して接続されている。当初は RIPng で経路を交換していたが、1998 年 1 月末に BGP4+への移行を始めた。1998 年 4 月末現在では、CISCO 社、Digital Equipment 社、ヨーロッパ AT&T 研究所、MERIT 社と接続している。

BGP4+は、バックボーン用の経路制御プロトコルである BGP-4 (Border Gateway Protocol 4) を複数のプロトコル用に拡張したもので [67]、これによって BGP-4 による IPv6 経路制御情報が交換できる。BGP4+では、パス・ベクトル型の経路制御プロトコルであること、ピアリングの概念、状態管理など、基本的な仕様は BGP-4 のものを受け継いでおり、オプション属性を追加することで複数のプロトコルに対応している。具体的にはたとえば、パス属性 (Path Attributes) フィールドのオプション属性として、マルチプロトコル到達可能情報 (`MP_UNREACH_NLRI`) 属性が定義されている。この属性には、AF の識別子、アドレス長などが含まれるため、IPv6 の経路情報を格納できる。

BGP4+の実装は gated を拡張したものを日立製作所の NR60 上で動作させている。アド

レスを集約したため RIPng での経路情報の混乱も収束し、一時期は 300 あった経路数も半分に減少している。

本来ならば経路数は 6bone のコア・サイト数と WIDE 6bone 内部の経路数の和である 60 程度に経路数は収まるはずだが、一部で古い経路や適切に集約されていない経路がアナウンスされている。

6bone は現在でもその大部分がトンネリングで構築されているのに対し、6bone JP は図 1.5で示したようにトンネリング以外の多様なリンクも用いられて構成されている。IPv6 のために専用線を用意したのは WIDE プロジェクトが世界初であり、今でも希少である。

現在 6bone JP で使用されているデータリンクの種類は次の通りである。

- トンネリング
- シリアル・リンク
- ATM
- ISDN
- T1

1.9.3 統計

6bone JP では 1 時間に 1 回、接続性に関する情報を収集して、WWW 上でその集計を公開している[1]。情報を視覚化することはトラブルの早期発見、修復に役立つ。また、MRTG を IPv6 に移植、運用して 6bone を流れるパケットの特徴を解析している[2]。

1.9.4 WIDE 6bone 登録局

6bone JP の拡大に伴って、参加組織が増大していくことが予想される。従来は手動で運営してきた接続情報等のデータベース管理、トポロジー図の更新等をできる限り自動化すべきである。我々は、1997 年 10 月から WIDE 6bone 登録局のホームページを用意し、WWW 上でデータベースを管理している[3]。また、これと連動して 6bone 登録局で管理されているデータを更新している。

IPv4 とは異なり、IPv6 ではアドレスが多段の階層構造になっているので、アドレスの割り当てやその管理が分散できるという利点がある。この利点を利用して、分散型の登録システムに関しても実験を開始している。

WIDE 6bone 登録局では、認証に PGP 署名を利用している。WIDE 6bone 登録局のデータを更新する人は、組織と自分の PGP 公開鍵を登録しておく。データの更新は入力には WWW を利用している。入力された内容は PGP で署名されたメールで WIDE 6bone 登録局に送られる。登録局では、事前に登録された管理者であるか認証しデータを更新する。

[1] http://www.v6.wide.ad.jp/Connectivity/ping/

[2] http://mango.itojun.org/mrtg/

[3] http://www.v6.sfc.wide.ad.jp/6bone/

表 1.2: 6bone JP 参加組織

6bone JP 接続ポイント	東京 奈良 京都 藤沢 小松 大手町 根津 青海
6bone JP 参加組織	WIDE プロジェクト APAN NTT ソフトウェア研究所 IRI Inc. インターネット互助会横浜
WIDE 6bone 参加組織	東京大学 Sony 京都大学 Matsushita Graphic Communication Systems,Inc 東京工業大学 NEC IRI 奈良先端科学技術大学院大学 大阪大学 創夢 TDI 慶應大学 富士通研究所 富士通 東芝 itojun.org 日立 NTT ソフトウェア研究所 北陸先端科学技術大学院大学

1.10 IPsec

IETF IPsec 分科会では、暗号/認証プロトコルである IPsec の標準化と鍵交換プロトコルである IKE(Internet Key Exchange) の標準化が進められている。ここでは、1997 年および 1998 年初頭の IETF IPsec 分科会における IPsec の標準化活動について述べる。IKE に関しては 1.11節を参照されたい。以下では ID のファイル名について、先頭の「draft-ietf-ipsec」および末尾の「.txt」を略す場合がある。

IPsec の仕様には大きく分けて RFC1825[68] から RFC1829[69] で定義される「旧版」と、現在 IETF IPsec 分科会で標準化が進んでいる「新版」の 2 つがある。旧版については現在標準化活動は停止しており、もっぱら新版に関して議論されている。旧版と新版の違いは以下のようにまとめられる。

- リプレイ攻撃防止のため、各ヘッダに通し番号 (sequence number) フィールドが追加されている。

  新版では、鍵交換アルゴリズムが利用可能な場合、同一の鍵を用いて 2^{32} 個以上のパケットを送信してはならない。鍵交換アルゴリズムを利用しない場合にはリプレイ攻撃防御を利用してはならない。

- 暗号ペイロード (ESP) に、ペイロードの認証機構が追加された。

  これは当初リプレイ攻撃防止のために追加されたが、ペイロードのみを認証したい場合にも利用される。

- 旧版で曖昧であった記述がより明解になった。

認証ヘッダ (AH) の ID の最新版は「IP Authentication Header (auth-header-05)」である。この ID では、HMAC-MD5-96 と HMAC-SHA-1-96 を「実装義務のある認証アルゴリズム」としている。暗号ペイロード (ESP) の ID の最新版は「IP Encapsulating Security Payload (ESP) (esp-v2-04)」である。この ID では、暗号アルゴリズムとして DES CBC モードと「暗号化なし」の 2 種、認証アルゴリズムとして HMAC-MD5-96、HMAC-SHA-1-96 と「認証なし」の 3 種の実装を義務づけている。「暗号化/認証なし」アルゴリズムは esp-v2-04 ではじめて導入された。これは、「ペイロードの認証はしたいがペイロードの暗号化はしない」「ペイロードの暗号化はしたいがペイロードの認証はしない」ような場合にも ESP が利用できるよう追加されている。以上挙げたアルゴリズム以外にも、いくつかの暗号/認証アルゴリズムが ID として提出されている。

IPsec および IKE に関する ID は、以下が 1998 年 3 月 20 日に IETF IPsec 分科会から IESG(Internet Engineering Steering Group) へ Proposed Standard 候補として提出された。

- Security Architecture for the Internet Protocol
  (draft-ietf-ipsec-arch-sec-04.txt)

- IP Authentication Header
  (draft-ietf-ipsec-auth-header-05.txt)

- The Use of HMAC-MD5-96 within ESP and AH
  (draft-ietf-ipsec-auth-hmac-md5-96-03.txt)

- The Use of HMAC-SHA-1-96 within ESP and AH
  (draft-ietf-ipsec-auth-hmac-sha196-03.txt)

- The ESP DES-CBC Cipher Algorithm With Explicit IV
  (draft-ietf-ipsec-ciph-des-expiv-02.txt)

- IP Encapsulating Security Payload (ESP)
  (draft-ietf-ipsec-esp-v2-04.txt)

- The Internet IP Security Domain of Interpretation for ISAKMP
  (draft-ietf-ipsec-ipsec-doi-08.txt)

- Internet Security Association and Key Management Protocol (ISAKMP)
  (draft-ietf-ipsec-isakmp-09.txt)

- The Internet Key Exchange (IKE)
  (draft-ietf-ipsec-isakmp-oakley-07.txt)

- The NULL Encryption Algorithm and Its Use With IPsec
  (draft-ietf-ipsec-ciph-null-00.txt)

また、以下が informational RFC 候補として提出された。

- IP Security Document Roadmap
  (draft-ietf-ipsec-doc-roadmap-02.txt)

- The OAKLEY Key Determination Protocol
  (draft-ietf-ipsec-oakley-02.txt)

IPsec を実装する際に実装義務のある暗号/認証アルゴリズムは、すべて Proposed Standard 候補として提出されている。今後とも IPsec や IKE に関する議論は継続されるが、基本的には以上挙げた文書とは大きく変わらないものが RFC として発行され Standard トラックに乗るであろう。IPsec に関する API(Application Programming Interface) などに関しては、IPsec 分科会として標準化していない。

1.10.1 1997 年 WIDE 春合宿での相互接続実験

1997 年 3 月 16 日〜 3 月 19 日に、浜松市の遠鉄ホテル・エンパイアにて開催された 1997 年度 WIDE 春合宿において、各組織がそれぞれの実装を持ち寄り、IPsec の相互接続性を検証した。実験は IPv6/IPsec 共同ハッキング・ルーム内の合宿実験ネットワークの 1 セグメント上を使用して実験に取り組んだ。

また、本来は IPsec の通信で使用されるアルゴリズム、SPI、鍵などの必要なパラメータは、IKE の鍵交換デーモンにより設定されることになっている。だが、今回の実験はまだその前段階であるため、必要なパラメータは、あらかじめ取り決められた値を手動で設定して実験した。

なお、IPsec の主な機能である AH と ESP には、現在、RFC によって提案されている旧版と、ID によって提案されている新版が存在するが、下記の実験結果中の RFC 版、ID 版という語句がそれぞれに対応する。

1.10.2 IPv4 での相互接続実験

IPv4 での IPsec 相互接続実験には、(株) インターネットイニシアティブ、国際電信電話 (株) & (株) 創夢、(株) 東芝、日本電気 (株)、横河ディジタルコンピュータ (株)、龍谷大学の 6 実装が参加した。実験は ping によって接続を確認するという方法で進められた。相互接続実験の結果と、そのときに用いられたアルゴリズム、モード等を以下に示す。

「○」は相互接続性が確認できたこと、「△」は一部条件付きで相互接続性が確認できたこと、そして「×」は相互接続性が確認できなかったことを表す。

IIJ — 東芝

○ AH(ID 版)、HMAC-MD5、トランスポート/トンネル・モード

○ ESP(ID 版)、DES-CBC、IV 長=8、トランスポート/トンネル・モード

× ESP(ID 版)、3DES-CBC、トランスポート・モード
IIJ、東芝両側で ICMP チェックサム・エラーを検出した。復号化には成功しているが、IV などの部分の整合が取れていない可能性が高い。

× AH(ID 版)、HMAC-SHA1、トランスポート・モード
IIJ、東芝両側で認証エラーを検出した。

× AH(ID 版)、断片化されたパケット

× ESP(ID 版)、断片化されたパケット
IIJ 側で正しく分割されていない可能性がある。2000 バイトのデータに対して 3 つの断片が送信されていて、オフセット 1480 のものが 2 つ存在した。東芝 → IIJ には正しい応答が返る。

IIJ — 龍谷大

○ AH(ID 版)、HMAC-MD5、トンネル・モード

× ESP(ID 版)、CAST128-CBC、トランスポート・モード
IIJ、龍谷大両側で復号化に失敗した。

× ESP(ID 版)、DES-CBC、IV 長 = 8、トンネル・モード
龍谷大側で復号化に失敗した。IIJ 側では復号化には成功した。しかし、IIJ — 東芝、龍谷大 — 東芝間では通信に成功した。

IIJ — KDD & 創夢

× AH(RFC 版)、HMAC-MD5、トランスポート・モード
KDD & 創夢側の設定は HMAC-MD5 ではなく、KEYED-MD5 だった。

△ ESP(RFC 版)、DES-CBC、トランスポート・モード
KDD & 創夢側は内向き、外向きで同じ SPI の場合のみサポート。KDD & 創夢側で SPI のチェックを外すと通信に成功したので、暗号化/復号化部分は正常に動作しているといえる。

IIJ — YDC

○ AH(ID 版)、HMAC-MD5、トランスポート/トンネル・モード

△ ESP(ID 版)、DES-CBC、IV 長=8、トランスポート/トンネル・モード
YDC 側のパディング内容のチェックに引っかかった。

× ESP(ID 版)、3DES-CBC、トランスポート・モード
YDC 側で ICMP チェックサム・エラーを検出した。

東芝 — 龍谷大

○ AH(ID 版)、HMAC-MD5、トンネル・モード

○ ESP(ID 版)、DES-CBC + HMAC-MD5、明示的 IV、IV 長 = 4、トンネル・モード

東芝 — KDD & 創夢

× AH(RFC 版)、KEYED-MD5、トランスポート・モード

○ ESP(RFC 版)、DES-CBC、IV 長 = 8、トランスポート・モード

東芝 — YDC

○ ESP(ID 版)、3DES-CBC + HMAC-MD5、IV 長 = 8、トランスポート・モード

YDC — 龍谷大

○ AH(ID 版)、HMAC-MD5、トンネル/トランスポート・モード

× AH(ID 版)、HMAC-SHA1、トランスポート・モード
龍谷大 → YDC で HMAC-SHA1 の認証エラーを検出した。YDC の実装に問題がある可能性がある。YDC → 龍谷大は実施しなかった。

○ ESP(ID 版)、DES-CBC derived、トンネル/トランスポート・モード
龍谷大とは下の構成のパケットでの通信にも成功した。
[IP][AH トンネル][IP][ESP トンネル][IP][AH トランスポート][ESP トランスポート][ICMP]

YDC — KDD & 創夢

× AH(RFC 版)、KEYED-MD5、トランスポート・モード

○ ESP(RFC 版)、DES-CBC、IV 長 = 8、トランスポート・モード

1.10.3 IPv6 での相互接続実験

IPv6 での IPsec 相互接続実験には、(株) 東芝、日本電気 (株) の 2 実装が参加した。実験は IPv4 の場合と同様に、ping によって接続を確認するという方法で進められた。相互接続実験の結果と、そのときに用いられたアルゴリズム、モード等を以下に示す。

NEC — 東芝

○ AH(ID 版)、HMAC-MD5、トランスポート・モード

○ AH(ID 版)、HMAC-SHA1、トランスポート・モード

○ ESP(ID 版)、DES-CBC (明示的 IV) + HMAC-MD5、トランスポート・モード

○ ESP(ID 版)、DES-CBC (明示的 IV) + HMAC-SHA1、トランスポート・モード

○ AH (HMAC-MD5) + ESP (DES-CBC + HMAC-MD5)、トランスポート・モード

× AH(ID 版)、HMAC-MD5、トランスポート・モード、断片ヘッダ

× ESP(ID 版)、DES-CBC (明示的 IV) + HMAC-MD5、トランスポート・モード、断片ヘッダ

△ ESP(ID 版)、DES-CBC (明示的 IV) + HMAC-MD5、トンネル・モード
NEC → 東芝は成功。東芝 → NEC は失敗。
ICMP param prob unknown header(ptr = 6) が返る。IPv6 ヘッダの ip6_nxt の値がおかしいか、うまく処理できていない (原因究明できず)。

1.10.4　今後の相互接続実験について

今回の合宿での相互接続実験によって、実装中の不具合を洗いだし、合宿中のデバックによって問題点を解決できた。

今までの実験においては、広域 IPsec 運用のために必要不可欠となる IKE の鍵交換デーモンの独自実装を持ち込む組織はまだそれほど多くはなく、IPsec に重点を置いて実験してきた。今後 IKE の実装が増えるにつれて、鍵交換デーモンである IKE と連携した IPsec の相互接続性を検証していきたい。

また、合宿中は時間が限られており、実験、ハック、および、デバッグが繰り返されるため、希望の実験をすべて実施できたことは少なかった。今後は、現在着々と準備が進められている IPsec のテストベッドである WIDE IPsec Pingbone を利用して、合宿期間以外でも日常的に IPsec の相互接続性を検証できる環境を構築していきたいと考えている。

1.11　自動鍵管理プロトコル

ESP や AH 等のようにセキュリティ・アソシエーション (以下、SA と略記) を確立して通信する場合、鍵と呼ばれる秘密の情報を共有しなければならない。IPsec アーキテクチャ [70] では、管理者が能動的に秘密の共有情報を交換し管理する手動鍵管理の方法を実装必須としている。しかし、ソケット単位の SA を必要としている計算機の場合、新たなセッションが開始するたびに鍵を交換しなけれならないので、手動鍵管理では限界がある。また IP アドレス単位の SA でも、通信する計算機が増加すれば、それだけ管理する手間もかかる。このため、より広域的な拡張性のある自動的な鍵管理プロトコルが必要とされる。

1.11.1　IKE

IETF の IPsec 分科会では、鍵情報を自動的に交換する仕組みとして、IKE[71] を提案している。

IKE は Internet Security Association and Key Management Protocol (ISAKMP)[72] を基に定義されている。ISAKMP は、使用するアルゴリズムやその鍵、有効期限等の SA に関係するパラメータを、認証付きで交換するための枠組を定義したプロトコルである。交換される個々のパラメータは IPsec Domain Of Interpretation(DOI) [73] により定義されている。また、実際に交換される情報の基本的なアイデアを、OAKLEY[74] や SKEME[75] から得ている。IKE は UDP を使って通信する。ポート番号は 500 が予約されており、IPv4 と IPv6 の両方で利用でき、ユーザ空間に実装可能である。

図 1.6に IKE の位置付けを示す。アプリケーションからの依頼によるカーネルの通信開始時に、対応した SA が存在しない場合、ユーザ空間に実装された IKE がカーネルからの鍵要求を受けて鍵交換を始動したり、交換した鍵をカーネルへ設定できる。

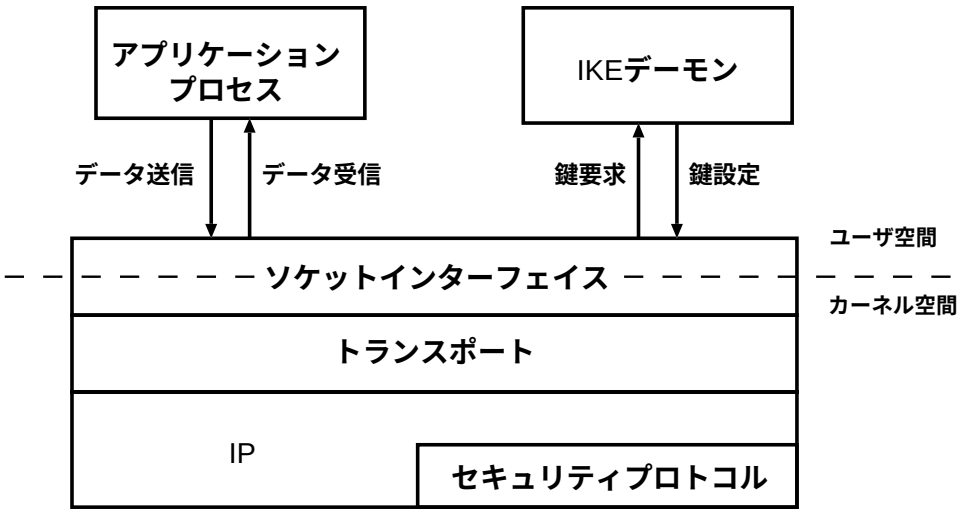

図 1.6: IKE の位置付け

IKE パケットは、IKE ヘッダ (図 1.7) と IKE ペイロード (図 1.8) から構成され、IKE ペイロードは、ヘッダ部とデータ部に分けられる。IKE ヘッダの始動者用クッキーと応答者用クッキーは、通信している当事者同士のセッションを識別する重要なフィールドである。IKE ヘッダと IKE ペイロードのヘッダ部には、次ペイロード・フィールドが存在し、各ペイロードを数珠つなぎにできる。次ペイロード・フィールドに 0 が指定されていれば、そのペイロードが最後であることを意味する。

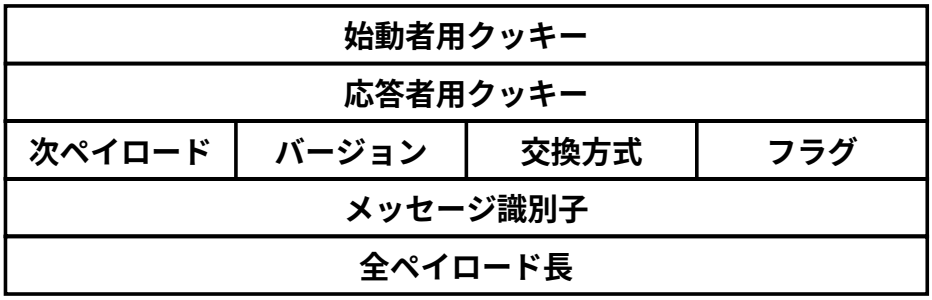

図 1.7: IKE ヘッダ

重要なペイロードに、SA のパラメータを含む SA ペイロード、鍵交換のための情報を含む鍵交換 (Key Exchange) ペイロード、乱数情報 (Nonce) ペイロード、自己情報 (Identification) ペイロード、認証情報 (Hash) ペイロードがある。

IKE はフェーズ 1 とフェーズ 2 に分けられる。フェーズ 1 では、通信を安全にするために IKE 用の SA を確立する。フェーズ 2 では、確立された SA の下で ESP や AH の

次ペイロード	予約	ペイロード長
データ部		

図 1.8: IKE ペイロード

ための鍵情報を交換する。

1.11.2　フェーズ 1

フェーズ 1 は、IKE 同士の通信を安全にするために IKE 用の SA を確立するフェーズである。SA を確立するためには、メイン・モードとアグレッシブ・モードの 2 つが定義されている。それぞれ、ISAKMP で定義される自己情報保護交換、アグレッシブ交換を使用する。自己情報は、メイン・モードでは SA 確立後に交換されるので保護されるが、アグレッシブ・モードでは、SA が確立される前に交換するので保護されない。ただし、認証方式に公開鍵暗号を使用したときのみアグレッシブ・モードでも自己情報の保護が可能となる。またアグレッシブ・モードは、メイン・モードに比べて交換する回数が少ないため時間を短縮するためにも使用される。鍵を計算するための元となる情報の交換には Diffie-Hellman 鍵交換方式 (以下、DH と略記) を採用している。認証方法として電子署名、公開鍵暗号、既知共有鍵の 3 つが定義されている。通信する IKE 同士がどれか 1 つの使用を合意し使用する。

ここでは実装必須となっているメイン・モードの既知共有鍵認証方式 (図 1.9) を説明する。既知共有鍵認証方式は予め共有する秘密の鍵を使って相互に認証する方式であり、共有鍵は予め安全に交換され、共有できていなければならないという制約がある。これは他の認証方式でも同じことがいえる。これを図 1.9に示す。

1. 始動者は、IKE ヘッダの始動者用クッキー (Ci) に適切な値を設定し、応答者用クッキー (Cr) を 0 で埋める。そして SA ペイロード (SA*) に自分が使用可能な鍵交換方式や認証方式、ハッシュ・アルゴリズム、暗号アルゴリズムを指定する。この指定は複数可能である。

2. 応答者は、許容できる鍵交換方式や認証方式のうちから 1 つを選択し、それを SA ペイロード (SA) に設定する。また IKE ヘッダの応答者用クッキー (Cr) を適切な値に設定する。このとき始動者用クッキー (Ci) は変更しない。以降のすべての交換では、この 2 つのクッキーがセッションの識別子となる。

3. DH 鍵交換のための情報 (KEi) と、任意の乱数値 (Ni) を送信する。

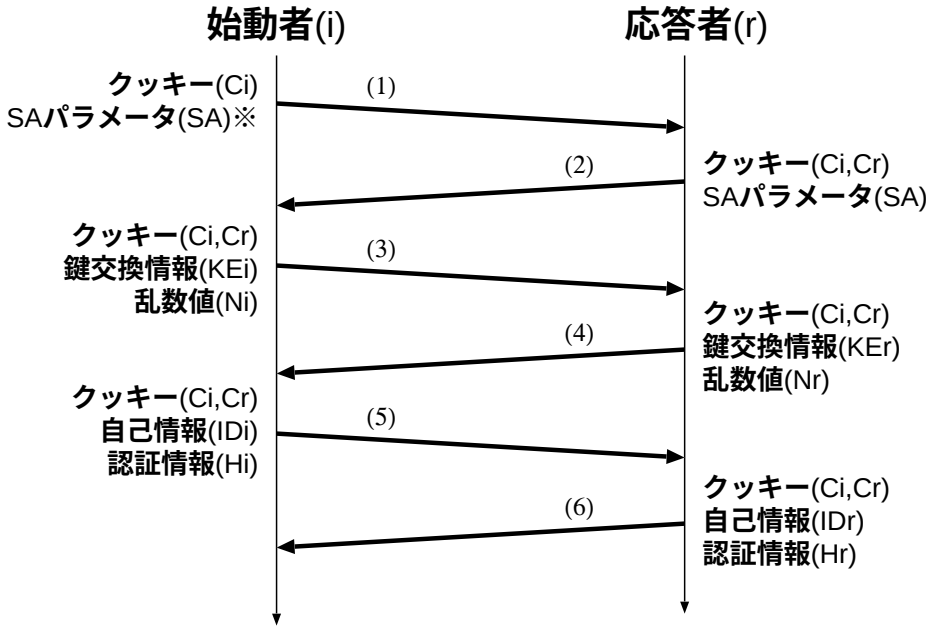

図 1.9: フェーズ 1 メイン・モード 既知共有鍵認証方式の交換

4. DH 鍵交換のための情報 (KEr) と、任意の乱数値 (Nr) を送信する。

   この時点で、KEi、KEr より DH 共有鍵情報 (KEir) を計算する。そして以下に示す計算式により、

```
SKEYID = prf(既知共有鍵、Ni | Nr)
SKEYID_d = prf(SKEYID、KEir | Ci | Cr | 0)
SKEYID_a = prf(SKEYID、SKEYID_d | KEir | Ci | Cr | 1)
SKEYID_e = prf(SKEYID、SKEYID_a | KEir | Ci | Cr | 2)
```

   prf(key、data) は交換された HMAC アルゴリズム
   ‘‘|’’ は結合を表す。0、1、2 は 1 バイトの数値。

   最終的に確立される SA のための鍵 (SKEYID_d)、IKE 用認証鍵 (SKEYID_a)、IKE 用暗号鍵 (SKEYID_e) が求められる。この時点で IKE 用の SA が確立したことになる。

5. DH 鍵交換情報 (KEi、KEr)、クッキー (Ci、Cr)、使用された SA ペイロードのデータ部 (SAp)、そして自己情報 (IDi) をハッシュ(Hi) し暗号化して送信する。ここで、暗号化に必要な最初の初期化ベクトル (IV0) は、

```
IV0 = hash(KEr | KEi)
```

として求め、以降の交換で使う初期化ベクトル (IVn) は

```
Vn =（前回暗号化または復号化したデータの後ろ 8 バイト）
```

とする。

6. ハッシュ(Hi) を再計算し始動者を認証する。応答者も同様に、DH 鍵交換情報 (KEi、KEr)、クッキー (Ci、Cr)、使用された SA ペイロードのデータ部 (SAp)、そして自己情報 (IDr) をハッシュ(Hr) し暗号化して送信する。

   始動者はハッシュ(Hr) を再計算し応答者を認証する。これで互いに認証済の SA が確立できたことになる。

1.11.3　フェーズ 2

フェーズ 2 は、フェーズ 1 で確立された SA の下で、ESP や AH 等の SA の情報を安全に交換するフェーズである。交換方式としてクィック・モードが定義されている (図 1.10)。フェーズ 2 では、複数の SA の情報がやりとりされるため、それらを識別するために IKE ヘッダに 4 バイトのメッセージ識別子 (Mid) がある。フェーズ 2 での暗号化に用いられる最初の初期化ベクトル (IV0) は

```
IV0 = hash(フェーズ 1 の暗号化または復号化したデータの後ろ 8 バイト | Mid)
```

として求められる、そして以降の初期化ベクトル (IVn) は

```
IVn =（前回暗号化または復号化したデータの後ろ 8 バイト）
```

とする。

1. 始動者は、IKE ヘッダのメッセージ識別子に適切な値を設定し、使用する自分向きの SA に関係するパラメータを SA ペイロード (SAi) に設定する。そして任意の乱数値 (Ni) を設定し、それらのハッシュ(H1) を計算する。

   ```
   H1 = prf(SKEYID_a、Mid | SA | Ni)
   ```

2. 応答者は、受信したデータとハッシュ(H1) を再計算し検証する。検証後、自分向きの SA に関係するなパラメータを SA ペイロード (SAr) に設定する。そして任意の乱数値 (Nr) を設定し、それらのハッシュ(H2) を計算する。

   ```
   H2 = prf(SKEYID_a、Mid | Ni | SA | Nr)
   ```

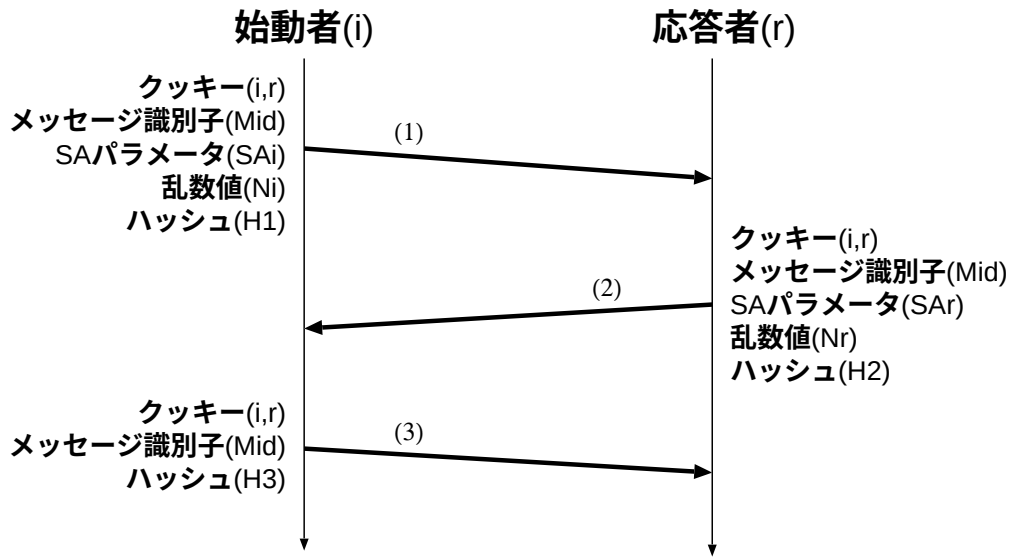

図 1.10: フェーズ 2 クィック・モード

3. 受信したデータとハッシュ(H2) を再計算し検証する。検証後、ハッシュ(H3) を計算し送信する。

```
H3 = prf(SKEYID_a、0 | Mid | Ni | Nr)
```

これにより SA が安全に交換されたことになる。

1.11.4 ヤマハの実装

ヤマハ (株) は、ISDN リモート・ルータ RT80i に IPv4 用の IKE を実装して相互接続試験に参加した。

この節ではヤマハの実装を概観していく。まずはじめに、IKE を中心とするシステムの構成を示し、システムを構成する個々の処理モジュールの役割について説明する。その後に、本実装の特徴と今後の課題について述べる。

1.11.5 システムの構成

本実装はリモート・ルータ RT80i のファームウェアの一部であり、IKE に関係するモジュールは大きく 4 つの部分からなる。

第 1 の部分は IKE デーモンであり鍵交換の主要な処理を担う。第 2 の部分は SA の管理モジュールである。第 3 の部分は鍵交換始動モジュールであり、IKE に鍵交換の開始を要求する。第 4 の部分は、IKE を設定するためのユーザ・インターフェイスである。

1.11.6　IKE デーモン

IKE デーモンは、UDP/500 に宛てられたメッセージを処理し鍵交換を進行させる。また、鍵交換始動モジュールの要求を受けた場合には、始動者となって鍵交換を開始する。以下に IKE の仕様を示す。

鍵交換モード (IKE フェーズ)	メイン・モード、アグレッシブ・モード
鍵交換モード (IPsec フェーズ)	クイック・モード
認証モード	既知共有鍵
グループ	modp768、modp1024
ID タイプ	IPv4 アドレス
暗号アルゴリズム	DES-CBC、3DES-CBC
ハッシュ・アルゴリズム	MD5
疑似乱数生成関数 (prf)	HMAC-MD5
状態	始点識別情報提示

表 1.3: IKE の仕様

本実装では、同時に複数の鍵交換セッションを扱うことができ、コマンド設定により、鍵交換の相手、および、セッションの数を制限できる。これは、リプレイ攻撃に対する防御措置として働くことを意図して実装したものである。

また、SA の新規登録、削除などの管理機構は独立させ、次に述べる SA 管理モジュールに委譲している。この理由は、IKE のみでなく AH や ESP に関する処理においても、SA の管理が必要なためである。

なお、実装に必要なアルゴリズムのうち、MD5 と DH は PGP に付属するフリーのコードを、DES-CBC と 3DES-CBC は LSI ジャパンの森公一郎氏によるフリーのコードを改造して用いている。また、HMAC-MD5 は、RFC2104 に掲載されているソースを改造して用いている。

1.11.7　SA 管理モジュール

SA 管理モジュールは、SA の新規登録、削除などの管理機能を担う。また、適時 SA の残り寿命を減じ、残り寿命の尽きた SA を消滅させる。加えて、鍵交換の途中においてメッセージの再送が必要であれば、IKE デーモンに対して再送の要求を送出する。IKE のメッセージ交換がタイム・アウトした場合には関係する SA を削除する。

1.11.8 鍵交換始動モジュール

鍵交換始動モジュールは、AH や ESP で用いる SA を、利用可能な状態に維持するために働く。すなわち、ルータの設定と SA の残り寿命に基づいて、必要な鍵交換を始動させ、必要な SA を確立させる。具体的な処理を以下に示す。

1. ルータの設定より、必要な SA の属性を求める。一般には複数の SA が利用されるため、属性の集合 S を求める。

2. すべての $s(\in S)$ について、s と同じ属性の SA がすでに管理されており、その残り寿命が十分に長ければ、s を S から取り除く。

3. $S' = \phi$ とする。すべての $s(\in S)$ について、s の相手先ホストが、管理されている IKE SA のいずれかの相手先ホストに一致すれば、s を S から S' に移す。

4. S の属性に基づいて SA ペイロードを構成し、IKE フェーズを始動する。

5. IKE フェーズが終了したら、$S + S'$ の属性に基づいて SA ペイロードを構成し、IPsec フェーズを始動する。

1.11.9 ユーザ・インターフェイス

既知共有鍵、SA の属性等を設定し、鍵交換を始動するために対話型ユーザ・インターフェイスを用意している。これは、従来から RT80i のファームフェアに具備されているものを用いる。ただし、telnet 等の遠隔操作により鍵交換を始動する場合には、生成された鍵を表示しないなど、セキュリティ面での工夫を設けている。

1.11.10 ヤマハの実装の課題

ヤマハの実装は、認証方式として既知共有鍵のみを実装するなど、最低限の機能に限定した実装である。今後は、認証方式に加え、アルゴリズム、交換方式などの充実を図って相互接続試験に臨みたいと考えている。

AH と ESP、および IKE の連携については、前者の実装が進み検討が必要な段階にある。本実装では SA の残り寿命が閾値以下になったときに IKE を始動しているが、これ以外のアプローチも考えられる。たとえば、AH や ESP によって SA が必要になったときに IKE を始動するなどの方法が挙げられる。

加えて、より詳細な評価が必要である。従来は、簡単なネットワーク環境でしか運用していないが、今後は様々なネットワーク環境で運用し評価していくことが望まれる。

1.11.11　YDC の実装

YDC(株) では、IPv4 の IPsec 実装の次段階として自動鍵管理プロトコルである IKE の一部をアプリケーションとして実装した。このアプリケーションを以下 iked と呼ぶ。iked は FreeBSD 2.2.5 で開発し、同 OS 上と BSD/OS 2.1 で動作を確認した。アプリケーションなので他の OS への移植も容易かと思われる。

1.11.12　基本的な仕様

IKE とそれに関する他の仕様は、IETF の IPsec 分科会により標準化に向けて現時点でも改定が続いている。よって ID も逐次改版されている状態である。iked では IKE として仕様 [76] を採用し、ISAKMP は仕様 [77] を、DOI は仕様 [78] を採用し実装した。なお現バージョンの iked は鍵と SA の情報の交換だけが可能であり、カーネルに対しての直接の API はない。

以下に iked でサポートする機能を示す。

鍵交換方式 — メイン・モード、クィック・モード

DH 鍵交換のグループ — MODP768、MODP1024

認証方式 — 既知共有鍵

暗号アルゴリズム — DES-CBC、3DES-CBC

ハッシュ・アルゴリズム — MD5、SHA1

認証アルゴリズム — HMAC-MD5、HMAC-SHA1

自己情報 — IPv4 アドレス、ポート番号

状態 — 始点識別情報提示

暗号モジュールは、すべて SSLeay-0.8.0 に含まれる libcrypto.a を使用した。SSLeay はオーストラリアの Eric Young 氏が作った SSL パッケージであり、libcrypto.a はこれの暗号ライブラリである。

iked に与える設定は、起動時の引数と設定ファイル iked.conf により行う。可能な限り多くの設定を可能にするために iked.conf には多種なキーワードが存在する。以下に主なキーワードとその機能を挙げる。

`listen` IKE が待機する IP アドレスとポート番号の組を設定。

`remote` 交換に必要なパラメータを IP アドレス毎に設定。

`exchange_mode` 交換方式を設定。

`id_type` 自己情報のタイプと値を設定。

`phase` フェーズ毎に必要なパラメータを設定。

`proposal` SA に必要なパラメータを設定。

`transform` proposal で必要な変換を設定。

`try_to_send` 再送回数。

`send_timer` 再送のためのタイマ (秒)。

`padding` 暗号化に必要なパディングの設定。

`log` ログのレベル。

`filter` 接続の可否を IP アドレス毎に設定。

1.11.13 処理概要

以下に iked の基本的な動作を述べる。図 1.11に動作の流れ図を示す。

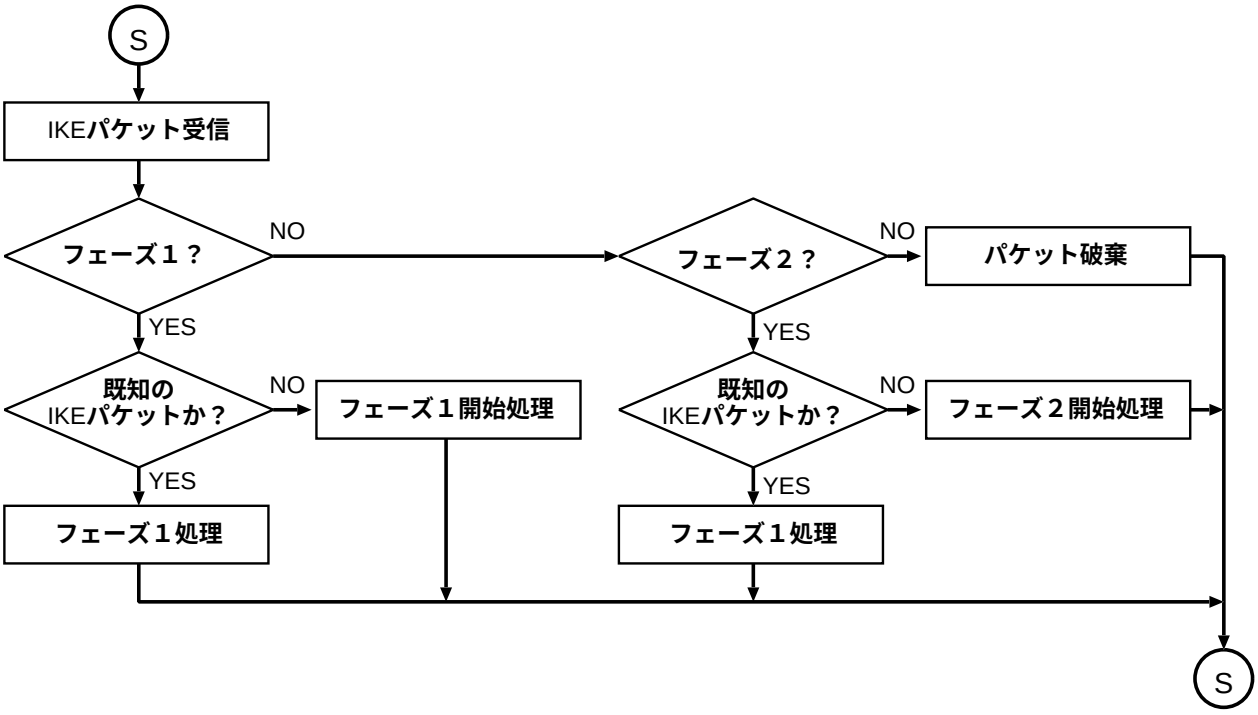

図 1.11: IKE の基本処理概要

iked は起動時に引数と iked.conf を解析し指定された状態で立ち上がる。そして iked.conf に指定されたポートでパケット待ち状態に入る。

パケットを受信すると、まず IKE ヘッダのメッセージ識別子をみてフェーズ 1 のパケットか、フェーズ 2 のパケットかを判断する。メッセージ識別子が 0 であればフェーズ 1 な

ので、次にすでに交換が始まっている相手からのパケットかを判断する。これにはパケットに含まれるクッキーを用いる。既存の交換のパケットであれば、対応した段階の処理をする。未知の相手からのパケットであれば、フェーズ 1 の開始処理をする。

受信したパケットのメッセージ識別子が 0 以外であれば、クッキーが既知であるかを判断する。もし未知のクッキーであればパケットを破棄し然るべき警告を出力する。クッキーが既知であるがメッセージ識別子が未知であれば、フェーズ 2 の開始処理をする。メッセージ識別子も既知であれば既存のフェーズ 2 の対応した段階の処理をする。

そしてそれぞれの処理が終った後、パケット待ち状態に入る。

1.11.14　iked の今後の課題

IKE とそれに関する仕様は今後も改版されていくことが予想されるが、今後も基本的にはそれに追従して実装を進めていく。最優先事項として、カーネルに自動鍵管理のための API [79] ををを組み込むことが挙げられる。これにより、より自動化されたシステムを構築できると考えている。そしてさらに次段階として IPv6 に対応することにより、より完全な IPsec アーキテクチャを目指していきたい。

1.11.15　1997 年 WIDE 春合宿での IKE 相互接続試験

1997 年 WIDE 春合宿において、ヤマハ (株) と YDC(株) の 2 社は、それぞれ RT80i + 特別仕様ファームウェア、ノート PC + FreeBSD2.2.5 に IKE の実装を用意して相互接続性を検証した。

実験に使用する各パラメータに関して、直前に交わされた合意は以下の通りである。

鍵交換モード (IKE フェーズ)	メイン・モード
鍵交換モード (IPsec フェーズ)	クイック・モード
認証モード	既知共有鍵
グループ	modp768
ID タイプ	IPv4 アドレス
暗号アルゴリズム	DES-CBC
ハッシュ・アルゴリズム	MD5
疑似乱数生成関数 (prf)	HMAC-MD5
状態	始点識別情報提示

表 1.4: 実験のための合意

相互接続試験は、ヤマハが始動者、YDC が応答者となって進められた。コードの変更を繰り返してメイン・モードの完了を確認したが、時間の関係でクィック・モードの相互接続

性を確認することはできなかった。

ヤマハの実装には、IKE の仕様に対する解釈違いに起因する不具合があったため、それらを合宿中に修正した。また、鍵計算のための数学ライブラリとして SSLeay を導入し (後、PGP に変更)、鍵計算の所用時間を大幅に改善した。

合宿前に不明確だった点については、BOF と相互接続試験を通じて明確化し、実装に反映させた。

たとえば、ESP の認証機能を用いる場合には、IKE が生成した鍵を暗号鍵と認証鍵に分割する必要があり、その方法について明確化する必要があった。ID には、アルゴリズムによって規定されるという記述があるが、現時点の ID にそれらの規定はなく、今後、何らかの言及が加わるものと推察される。

メッセージの再送時における初期ベクトルの扱いについては、再送時に初期ベクトルを変化させないことが BOF の中で合意された。

今後の課題としては、クィック・モードにおける相互接続性の検証、メッセージの再送処理などが挙げられる。

1.12 年間を通じての活動

この節では v6 分科会の活動を時系列に列挙する。1.12.1節では、主に v6 分科会が主催、参加したイベントを示す。また、IETF での活動内容について 1.12.2節で触れる。

1.12.1 v6 分科会の活動

v6 分科会が 1997 年度を通じて活動した内容を、下記にまとめる。

5/12∼15 N+I'97 ホットステージ

- 世界初の異なる実装間での IPv6 over ATM による接続が確認される。

5/17 WIDE 研究会

6/4∼6 N+I'97 SSD IPv6

- IPv6 over ATM によるバックボーンの構築と、WIDE 6bone を通じて 6bone へ接続される。

7/7∼10 v6 ワークショップ (SFC)

7/12 WIDE 研究会

7/28∼8/2 IOL 相互接続実験

8/10∼15 第 39 回 IETF(ミュンヘン)

- ngtrans 分科会にて、下記が決定。
  1. 経路集約型アドレスへの移行
  2. RIPng から BGP4+ へ移行
- WIDE プロジェクトは pTLA = 5 (3ffe:500::/24) を割り当ててもらう

9 月 ID “IPv6 over Point-to-Point ATM Link” の version 00 を提出

9/1∼5 WIDE 合宿 (浜松)

10/2∼5 v6 ワークショップ (北陸先端大)

- IPv6 over ATM の相互接続性テスト。(Hydrangea、東芝、富士通)

10/8 奈良先端 – 北陸先端、奈良先端 – 東大が IPv6 over ATM で接続

10/30 Bob Hinden SFC 訪問

- IPv6 関係者と議論

10/31 Steve Deering SFC 訪問

11 月 旧テスト・アドレスから新テスト・アドレス (経路集約型アドレス) へリナンバリング

11/15 WIDE 研究会 (九州大学)

11/16∼19 v6 ワークショップ (九州)

- P2P リンク上での近隣探索プロトコルの議論など
- IETF での発表準備

12/1∼5 IOL 相互接続実験

- 日立、東芝が参加

12/8∼12 第 40 回 IETF(ワシントン DC)

- 3 つのトランスレータの発表をする

12/18 Internet week’97 6bone-jp BOF

- 以下の条件を満たせば NLA1 を割り当てることを決定
  - 公序良俗に反しない

  - 1 年後に報告書を書く
  - 接続条件の明示
  - 逆ひきの設定
  - WIDE 登録局の更新義務
- 現在の NLA1
  - WIDE プロジェクト
  - APAN
  - NTT Lab.
  - IRI
  - IMASY

1998 年 1 月 対外経路制御を RIPng から BGP4+ へ

2 月 ID “IPv6 over Point-to-Point ATM Link” の version 01 を提出。

2/5 分散システム運用管理シンポジウム ’98 (東工大)

- WIDE v6 分科会のこれまでの活動を発表

3 月 ID “IPv6 over Point-to-Point ATM Link” IPng 分科会 で last call

3/16∼19 WIDE 合宿 (浜松)

3/30∼4/3 第 41 回 IETF(ロサンゼルス)

- IPv6 でのメール配送の発表

1.12.2 IETF での活動

1.12.1節で示したように 1997 年度には、以下の 4 つの IETF の会合が開催された。

- 1997 年 4 月 メンフィス
- 1997 年 8 月 ミニュンヘン
- 1997 年 12 月 ワシントン
- 1997 年 3 月 ロサンゼルス

8 月のミュンヘンの時点で、IPv6 パケットを PVC ATM 上で配送する規格がなかった。ミュンヘンで個人的に諸外国の実装者に尋ねたところ、この仕様があると有益であるとの回答を得た。そこで、1.3節で述べた IPv6 over PVC ATM の仕様を ID に記述し 9 月に提出した [80]。

また、11 月にはヘッダ変換ルータである NR60 とトランスポート・リレーである FAITH に関して、ネームサーバとの連動に焦点を絞って説明した ID[81] を提出した。

12 月の IETF では、土屋が NR60、新家が SOCK64、山本が FAITH について ngtrans 分科会で発表した。また、山本は WIDE プロジェクトのこれまでの活動内容や成果について同分科会で紹介した。

1.6.3節で述べたように 1 月に中村が中心となってデュアル・スタック環境での MTA の振るまいについて実験した。3 月の IETF では、この結果を山本が ngtrans 分科会で報告した。

1.13　v6 分科会のまとめ

v6 分科会にはさまざまなベンダーや大学の方が参加しており、IPv6 や IPsec の実装も複数提供されている。また、日頃から 6bone の運営に努力を傾けている。1997 年度において、v6 分科会は活発に活動したくさんの成果をあげたといってよい。