2003－OS－93 (10)
2003／5／8

IPv6 拡張ヘッダを用いた付加的なサービスの提供

奥山 航平‡ 新城 靖† 鈴木 真一* 板野 肯三†

†筑波大学電子・情報工学系
‡筑波大学理工学研究科
* 筑波大学システム情報工学研究科
URL: http://www.hlla.is.tsukuba.ac.jp/

要 旨

この論文は、IPv6 の拡張ヘッダの１つである終点オプションを用いて、付加的なサービスの提供を支援することについて述べている。終点オプションを用いる方法の利点は、既存のプロトコルを変更する必要がないことである。付加的なサービスとしては、アクセス制御、クッキー機能、および、論理クロックの提供があげられる。終点オプションを扱うために現在規定されている API (Application Program Interface) には、次のような問題がある。第１に、独立した複数のサービス提供モジュールを共存させられない。第２に、TCP/IP に対する API やカーネル・モジュールに対する API が存在しない。この論文では、新しい UDP/IP に対する API 、TCP/IP に対する API、および、カーネル内拡張モジュールに対する API について述べている。新しい API では、複数のモジュールが共存可能になっている。

Providing Additional Services with IPv6 Extension Headers

Kouhei Okuyama‡ Yasushi Shinjo† Shinichi Suzuki* Kozo Itano†

‡Master's Program in Science and Engineering, University of Tsukuba
†Institute of Information Sciences and Electronics, University of Tsukuba
* Doctoral Program in Systems and Information Engineering, University of Tsukuba
URL: http://www.hlla.is.tsukuba.ac.jp/

Abstract

This paper describes system support for providing additional services by using the destination option, one of the extension headers in IPv6. The advantage of using destination options is that additional services can be provided without modifications of existing protocols. The additional services include access control, cookie facilities, and logical clocks. The current API for destination options has the following problems. First, this API does not allow multiple independent service providers to work together. Second, the API does not include the facilities for TCP/IP and kernel modules. This paper proposes a new API for UDP/IP, TCP/IP, and kernel modules. The new API allows multiple service providers to work together.

1 はじめに

インターネットのアプリケーションにおいて、既存のサービスに対して補助的なサービスを付加したいという要求がある。たとえば、電子メールを転送するためのプロトコル SMTP (Simple Mail Transfer Protocol) では、従来は利用者認証の機能が存在しなかった。SMTP に対して新たに利用者認証の機能を規定したものが、SMTP-Auth である [8]。SMTP-Auth を利用するためには、従来の SMTP サーバ、および、クライアントを置き換える必要がある。しかしながら、多種多様なクライアントを SMTP-Auth 対応に書き換えることは大きな労力を必要とする。DNS (Domain Name System) のように、さらに多くのクライアントを持つプロトコルを変更することは、非常に難しい。

プロトコルを変更せずに付加的なサービスを提供する手段として、IPv6 拡張ヘッダを用いる方法がある。IPv6 拡張ヘッダ [1] とは、IPv4 の IP オプションに相当するものであり、少量のデータを IP パケットに入れて送ることを許すものである。IPv6 拡張ヘッダを利用することにより、既存のプロトコルを変更することなく付加的なサービスを提供するために必要なデータを送受信することができる。しかしながら、IPv6 拡張ヘッダの１つである終点オプションを操作するために現在利用可能な API (Application Program Interface) [10] やそれを実装した処理系 KAME[3] では、まず、ユーザレベルの API が複雑で使いにくいという問題がある。特に複数のサービスを共存させる枠組みが存在しない。また、TCP 用の API やカーネルレベルの API が存在しないという問題もある。

このような問題点を解決するために、我々は IPv6 拡張ヘッダの１つである終点オプションを用いた付加的なサービスの提供を支援する研究を行っている。この論文では、終点オプションを用いるプログラミングを支援するための次のような API について述べる。

- UDP/IP に対するより使いやすいユーザレベルの API。終点オプションを用いる複数のアプリケーションの共存を許す。
- カーネルレベルで利用可能な API
- TCP 用の API

これらの API を用いることで、IPv6 を利用した付加的なサービスの提供が容易になる。たとえば、アクセス制御を実現するために、カーネル内で安全にユーザ識別子などの情報を付加することが可能になる。カーネル内で実現した場合、しばしばクライアントを変更する必要がなくなる。終点オプションに挿入されたユーザ識別子は、受信側のカーネルだけでなく、利用者プログラムにおいても利用可能になる。たとえば、DNS の名前サーバでは、要求メッセージ本体に含まれたドメイン名と終点オプションに含まれた利用者識別子の両方を利用してアクセス制御を行うことが可能になる。

2 関連研究

2.1 プロトコルの組み合わせによる付加的なサービスの提供

プロトコルを変更しないで付加的なサービスを提供する方法としては、複数のプロトコルを組み合わせて用いる方法がある。たとえば、電子メールに対して利用者認証機能を付加する方法としては、プロトコルを変更する方法の他に、POP before SMTP と呼ばれる方法も使われている。これは、SMTP に対して利用者認証のサービスを付加するために、電子メールを発信する前に、たとえ電子メールを受信する必要がなくても POP により認証を行うものである。一度認証を行ったクライアント（ＩＰアドレス）からは、一定時間の間は、認証機能がない SMTP により電子メールの発信を許す。

プロトコルを組み合わせて利用する方法のもう１つの形態は、トンネリングを使う方法である。 たとえば、SSH ポートフォワーディング [13] では、上位層 TCP/IP の通信と、下位層の SSH が持っている暗号化サービスを同時に利用することができる。PPPoE (Point to Point Protocol over Ethernet) [6] では、PPP による利用者認証付きのデータグラム転送サービスを、Ethernet というインタフェースを通じて利用することが可能となる。

このように複数のプロトコルを組み合わせる方法では、既にそれらのプロトコルに組み込まれているサービスを付加することはできるが、まったく新しいサービスを付加することはできない。本研究では、IPv6 拡張ヘッダの１つ終点オプションを用いることで、新たなサービスを付加することを支援する。

2.2 アクティブネットワーク

アクティブネットワーク (acive network) とは、本来ネットワーク層の処理しか行わないルータに対して、アプリケーション層までの処理を実行可能することで付加的なサービスを提供しようとするネットワークである [9]。アクティブネットワークの応用としては、ルータにおいてアプリケーションレベルの処理を必要とするファイアウォールやプロキシの実行があげられる [9]。

アクティブネットワークでは、基本的にルータにおける処理が必要であるため、アクティブネットワークを導入するにはルータを対応させる必要がある。本研究では、末端のホストにおいて解釈される拡張ヘッダである終点オプションを用いる。そのため、途中のルータは IPv6 に対応していれば変更する必要がない。

2.3 IPsec

IPsec とは、IP レベルでホスト単位の認証、暗号化を行うための技術である [4]。これにより IP パケットの完全性や機密性が保障される。IPsec で行うことは、通信相手の認証、暗号化、鍵の管理などである。IPsec の実現のために拡張ヘッダを用いている。具体的には通信相手を認証するための認証ヘッダ（AH)、トンネリングと暗号化を実現するためのカプセル化セキュリティペイロードヘッダ（ESP ヘッダ）というものがある。IPsec は、ホスト単位の認証機能を提供するが、ユーザを識別する機能はない。本研究では、IPv6 拡張ヘッダの 1 つである終点オプションを用いてユーザに関連した情報を送り、ユーザの認証やユーザを識別したアクセス制御を実現する。

3 IPv6 拡張ヘッダ

IPv6 拡張ヘッダは、IPv6 のパケットにおいて固定長の IPv6 基本ヘッダに続き可変長の付加的な情報を格納する部分である。拡張ヘッダは、IPv6 基本ヘッダと上位層 (TCP または UDP) のヘッダの間に挿入され、複数の拡張ヘッダをチェーン状に並べることができる。拡張ヘッダには、次のような種類がある。

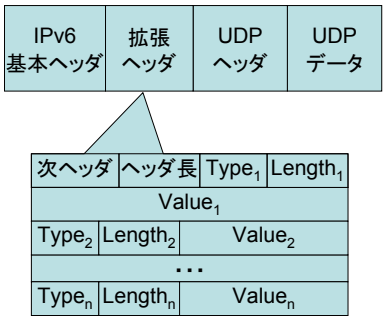

図 1: IPv6 拡張ヘッダ中の終点オプションにおける TLV エンコーディング.

- フラグメントヘッダ（IPv4 では基本ヘッダに含まれていたが IPv6 では基本ヘッダに含まれない)。
- IPsec[4] の AH (Authentication Header) と ESP (Encapsulating Security Payload) ヘッダ
- ルーティングヘッダ。
- オプションヘッダ。ルータごとに解釈されるホップ毎オプションと終点でのみ解釈される**終点オプション** *(destination option)* の 2 つがある。

終点オプションには、仕様の詳細が定められておらず、ユーザが利用できる部分がある。本研究では、この終点オプションを用いて付加的なサービスを実現するための情報を送受信することを支援する。

終点オプションヘッダは、一般に、TLV（Type-Length-Value) エンコーディングと呼ばれる方法で格納する必要がある。これは、図 1 に示したように、タイプ（8 ビット)、オプションデータ部分の長さ（8 ビット)、および、実際のデータ部分（可変長）という 3 つのフィールドを持つ。タイプの上位 3 ビットは利用法が決められている[1]。

4 付加的なサービスの例

この章では、IPv6 終点オプションを利用して提供可能な付加的なサービスの例を示す。本研究では、これらのサービスが標準化され広く使われるようになるのではなく、むしろ、特定のグループに属している人々の間で、特定の応用に特化した使われ方をすることを想定している。したがって、本研究では

[1] 上位 1 ビット目と 2 ビット目はルータが認識できないオプションを受け取った時の動作、3 ビット目はオプション値が途中で変更することを許すかどうかを規定する。

このような多種多様なサービスの提供を、個別に支援することを目的とする。

4.1 付加的なアクセス制御

アクセス制御サービスを付加的に提供する方法としては、ルータによるパケットフィルタやアプリケーション・レベルのプロキシを利用する方法がある。パケットフィルタでは、IP アドレスやポート番号、プロキシサーバでは、さらに応用固有の要求の内容を用いてアクセス制御を行う。それらの情報に加えて、終点オプションに付加的な情報を格納することで、より高度なアクセス制御を行うことができる。付加的な情報としては、次のようなものが考えられる。

- 利用者を識別する情報。UID (User Identifier) や GID (Group Identifier) など。
- アクセス・トークンやケーパビリティ
- アプリケーションの識別子

UID や GID は、オペレーティング・システムのカーネルという比較的安全な場所に保存されており、その中でファイルやプロセスに対するアクセス制御を行うために使われている。我々は既に IPv4 の IP オプション機能を用いて TCP のクライアント側からサーバ側に利用者情報を送信する仕組みを提案し、実装した [11]。その研究をふまえ、今後は、IPv6 の終点オプションを用いて、TCP 以外のサービス (UDP を使うサービス) や、認証・アクセス制御以外のサービスを付加的に提供することを支援する。

UDP を使うサービスとして、最も重要なものが DNS である [7]。DNS の要求メッセージには、ドメイン名は含まれているが、UID などの利用者を識別する情報は含まれていない。

IPv6 終点オプションを用いて DNS の要求メッセージに利用者を識別する情報を付加することにより、まず管理者によるネットワーク利用の制限が可能になる。たとえば、親や教師は、子供が閲覧可能なページを制限することができる。それから、各利用者が自らネットワーク利用を制限することもある。これにより、たとえば意図していない WWW サイトへの接続を制限し、Web bug 等による個人情報の漏洩を抑制することが可能になる。

4.2 クッキー機能

クッキーは、RPC や HTTP のようなコネクションレスの通信において、状態を維持するための仕組みである。たとえば、HTTP では、クッキーはセッションの概念を作り出すため、および、異なる IP アドレスからアクセスしてくる利用者を識別するために使われている [5]。

このクッキーは、現在 HTTP や RPC でしか利用されていない。クッキーを IPv6 終点オプションに格納することにより、HTTP 以外のプロトコル、特にコネクションレスである UDP でも同じような仕組みを利用することが可能となる。

4.3 論理クロック

論理クロックとは、分散システムなどで用いられる、システム全体で一貫しているクロックである [12]。論理クロックが実現されると、以下のような分散システムにおける課題が容易に解決される [12]。

- 最大 1 回メッセージ配信
- クロックを用いたキャッシュの一貫性
- クロックを用いた分散トランザクションの制御

本研究では、IPv6 終点オプションを利用して論理クロックを実現し、既存のプロトコルに対してこのような機能を付加することを目指す。

5 IPv6 終点オプションを利用する上での問題点

現在、IPv6 終点オプションを用いたプログラミングは、それほど容易なことではない。その要因の１つは、終点オプションを扱うための API にある。現在の API には、次のような問題点がある。

- 終点オプションを設定できるアプリケーションはパケットごとに１つに限定されている。
- ユーザレベルの API では、UDP、および、raw socket しか規定されていない。TCP で扱うための API が規定されていない。
- カーネルレベルでの API が存在しない。
- 終点オプションの送受信には、特権利用者の権限が必要である。

現在の API では、ある場所で終点オプションを設定した場合、別の場所では終点オプションを設定できないようになっている。このため、複数の場所で終点オプションを使うことができない。たとえば、カーネルにおいて終点オプションに利用者情報を付加することにした場合、利用者レベルでクッキー等別の目的に使うことはできない。

また、終点オプションは、通常のデータと同じく終点に送り届けられ解釈されるものである。ルーティングヘッダなど高い安全性が求められるものとは異なり、本研究の目的では、本来はプロトコルを改変し通常のデータとして送信しても差し支えないようなものを送り届ける。したがって、一般ユーザの権限で送受信を許しても問題は生じない。ところが、現在の KAME の実装では、終点オプションの送受信にも、他の拡張ヘッダの送受信と同様に特権利用者の権限を必要としている。

UDP だけでなく、TCP においても終点オプションを利用することの有用性は、我々の先行研究 [11] において明らかにしている。しかしながら、TCP のストリームの概念と終点オプションの間のセマンティックギャップを埋める方法は、これまで知られていなかった [10]。

現在の API は、ユーザレベルの socket 関連のシステムコールとライブラリ関数を規定しているだけである。4.1 節で述べたように、アクセス制御を実現するためには、カーネル内で利用者情報を終点オプションに埋める必要がある。またカーネルを修正すれは、クライアント側のプログラムを一切修正することなく新たなサービスを提供することもできる [11]。しかしながら、現在は、そのようなカーネル内の API は提供されていない。

6 終点オプションを扱うためのフレームワーク

5 章で述べた問題を解決するために、我々は、IPv6 終点オプションを扱うためのフレームワークを設計した。その概要を、図 2 に示す。

本フレームワークでは終点オプションの付加を、利用者プログラムに加えてカーネルにおいても可能にする。同様に、終点オプションの解釈も、受信側の利用者プロセスだけでなくカーネルにおいても可能にする。このため、カーネルにおいて終点オプションを扱うための API を新たに設計した。またより使いやすい利用者レベルの API を設計した。新しい API の最大の特徴は、複数の場所で終点オプションを設定することを許している点にある。

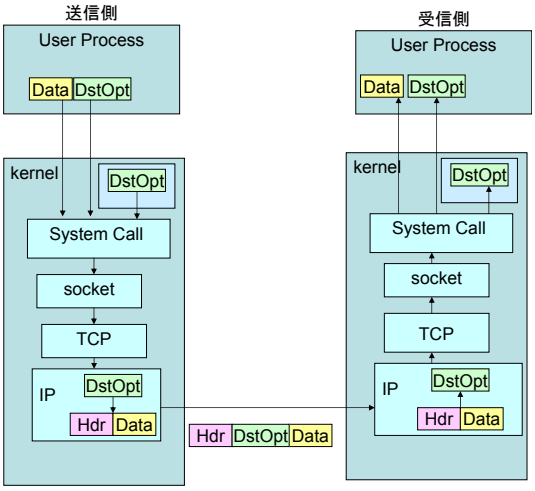

図 2: カーネルとユーザ・プロセスによる IPv6 終点オプションの付加と解釈

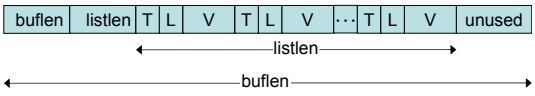

図 3: ip6_dest_tlvlist 構造体.

以下の節では、これらの新たな API について述べる。

6.1 UDP/IP に対するユーザレベル API の改良

今回設計した UDP/IP により終点オプションを送受信するためのシステムコールを表 1 に示す。表 1 に示したシステムコールは、全て図 3 に示した構造体 `ip6_dest_tlvlist` の番地を引数として取る。これは、3 章で述べた TLV エンコードされた値の並びを保持するための構造体である。先頭に、構造体全体の長さ、次に、実際に利用しているバイト数がある。それに続き、個々の TLV エンコードされた値が続く。この構造体には、通常のリストと同様に、要素の追加や削除のためのライブラリ関数も提供する。

従来の API と比較して、最大の特徴は、オプショ

表 1: 継続的に終点オプションを送受信するためのシステムコール

送/受	システムコール	optname	説明
送	setsockopt	IPV6_DSTOPTS_INSERT	終点オプションを送信するように設定する。
送	setsockopt	IPV6_DSTOPTS_DELETE	終点オプションを送信しないように設定する。
送	getsockopt	IPV6_DSTOPTS_LIST	現在送信されるように設定された終点オプションの一覧を返す。
受	setsockopt	IPV6_RECVDSTOPTS_ON	受け取りたい終点オプションを指定する。
受	setsockopt	IPV6_RECVDSTOPTS_OFF	もう受け取る必要がない終点オプションを指定する。
受	getsockopt	IPV6_RECVDSTOPTS_LIST	現在受信するように設定された終点オプションの一覧を返す。

```
ssize_t
inet6_opt_sendto(int s, void *buf, size_t len,
  int flags, struct sockaddr_in6 *from,
  socklen_t *flen, struct ip6_dest_tlvlist *l);

ssize_t
inet6_opt_recvfrom(int s, void *buf, size_t len,
  int flags, struct sockaddr_in6 *from,
  socklen_t *fromlen, struct ip6_dest_tlvlist *l);
```

図 4: UDP/IP に対して終点オプション付きのデータを送受信するライブラリ関数

```
list = tlvlist_alloc(100);
tlvlist_add( tlvlist, IP6OPT_USER_UID,
  sizeof(uid), &uid );
tlvlist_add( tlvlist, IP6OPT_USER_COOKIE,
  sizeof(cookie), &cookie );
s = socket(PF_INET6,SOCK_DGRAM,0);
setsockopt(s, IPPROTO_IPV6, IPV6_DSTOPTS_INSERT,
   tlvlist, tlvlist->buflen);
```

図 5: API の利用例 (UDP/IP、送信側、ソケット生成時にオプションを設定)

ンバッファの代りに 3 章で述べた TLV を扱うことにした点にある。従来の API におけるオプションバッファとは、IP データグラムに保存する形態とまったく同じ形で終点オプションを保存するものである。オプションバッファを使うと、IP データグラムを組み立てる際の処理が簡単になる。しかしながら、そのために複数のアプリケーションが終点オプションを設定できないという制約が生じていた。今回、我々の API では、TLV のリストという形式を用いたので、プログラマに対するインタフェースとしては抽象度が上がり、分かりやすくなった。さらに、実際の終点オプションの構成は、カーネルにより行われるので、複数の場所で TLV のリストを与えられたとしても、それらは自動的にカーネルにより構成される。

図 4 は、今回設計した利用者レベルのライブラリ関数である。これらのライブラリ関数は、それぞれシステムコール `sendto()` と `recvfrom()` と似せて設計した。ただし、終点オプションを送受信するための引数として構造体 `ip6_dest_tlvlist` を取る。システムコール `sendto()` と `recvfrom()` は、UDP/IP を用いるプログラムにおいて広く使われている。従来の API では、`sendmsg()` と `recvmsg()` システムコールにおける制御データ (`msg_control`) を用いる。この制御データを用いるプログラミングは非常に煩雑である。図 4 に示したライブラリ関

```
int
inet6_opt_listen(int s, int backlog,
  struct ip6_dest_tlvlist *tlvlist);

int
inet6_opt_connect(int s, const struct sockaddr *name,
  int namelen, struct ip6_dest_tlvlist *tlvlist);

ssize_t
inet6_opt_writev(int  fildes,  struct  iovec  *iov,
  int iovcnt, struct iovec *tlvlistvec);

ssize_t
inet6_opt_readv(int  fildes,  struct  iovec  *iov,
  int iovcnt, struct iovec *tlvlistvec);
```

図 6: UDP/IP に対して終点オプション付きのデータを送受信するライブラリ関数

数を用いることでプログラミングがはるかに容易になる。

図 5 に、定義した API の利用法を示す。複数の終点オプション (この例では uid と cookie) が融合できるようになっている。この例では、1 回のシステムコールで設定しているが、複数回のシステムコールで設定してもよい。

6.2 TCP に対するユーザレベル API

TCP/IP において終点オプションを扱うには、次の 2 つの方法が考えられる。

- コネクションを確立する時に送る
- データを送受信するたびに送る

前者は、コネクションを確立する時の SYN パケットに入れて送信する。我々が IPv4 において、ユーザ情報をクライアント側からサーバ側に送ったのは、この方法に相当する [11]。一般的には終点オプションは相互に交換される。この方法で、終点オプションを送信するには、6.1 章で述べた表 1 と同じ方法で設定し、コネクションの確立後に解除する。あるいは、図 6 に示したライブラリ関数 `inet6_opt_listen()` と `inet6_opt_connect()` を用いる。

TCP/IP においてデータを送受するたびに終点オプションを変えるのは、論理クロックの実現で必要になると思われる。この場合は、UDP/IP と同じく、`inet6_opt_sendto()` を用いる方法が考えられるが、それはうまく動作しない。それは、TCP/IP では、データの切目が保存されないからである。送信側で複数回に分けて送り出したデータが、受信側では結合されこともある。

この問題を解決するために、図 6 の 2 つのライブラリ関数 `inet6_opt_writev()` と `inet6_opt_readv()` を用いる。これは、それぞれ通常の `writev()` と `readv()` と同様にデータを送受信するが、データの切目が保存される。さらに、もう 1 つの `struct iovec` を用いて、終点オプションを送受信する。この API を実現するには、TCP 層を変更し、終点オプションのキューとデータの区切りを保存する必要がある。

6.3 カーネル API

カーネル API としては、我々が開発しているシステムコールに対するラッパ/リファレンスモニタ SysGuard [2] と類似の機能を提供する。SysGuard では、ガードと呼ばれる、デバイスドライバと同様にカーネルに組み込むモジュールを利用する。システムコールが実行されると、そのプロセスに対してそのシステムコールに設定されたガードが順に呼び出される。全てのガードが許可を返した時にのみシステムコール本体が実行される。

SysGuard では、ガードが適用されるスコープをプロセス単位で設定することができる。たとえば、利用者識別子、グループ識別子、および、プロセス木に対してガードを設定することができる。

```
after_socket(struct sockext_specific *sesd,
  int fd, int domain, int type, int protocol)
{
  struct ip6_dest_tlvlist *tlvlist ;
  char buf[12]; uid = getuid();
  tlvlist = tlvlist_init( buf,12 );
  lvlist_add( tlvlist, IP6OPT_USER_UID,
    sizeof(uid), &uid );
  setsockopt(fd, IPPROTO_IPV6, IPV6_DSTOPTS_INSERT,
   tlvlist, tlvlist->buflen);
}
```

図 7: UID を自動的に送信するような拡張モジュールの主要部分

SysGuard と類似の仕組みを、ソケット機能を拡張するために提供する。ただし、ソケットの機能を強化するために、次の 2 点を強化する。

- 寿命という概念を導入する。ファイル記述子と同じ寿命を持つ拡張機能モジュールのインスタンスを生成することを可能にする。
- マスタという概念を導入する。マスタは、他の拡張機能モジュールのインスタンスを生成し設定する。たとえば、システムコール `socket()` に対して設定されたマスタは、新たにインスタンスを生成し、そのファイル記述子に対応したソケットに張り付ける。

図 7 において関数 `after_socket()` は、関数 `socket()` システムコールの実行後に呼ばれる。引数は、`socket()` システムコールと同じものに加えて、結果となるファイル記述子と個々のインスタンスに固有のデータを保存するための領域を取る。この関数は、ユーザレベルの API と同じく `setsockopt()` により終点オプションを送り出すように設定している。

6.4 一般利用者による終点オプションの利用

我々の目的では、終点オプションは、本来はプロトコルを拡張し、通常のデータとして送るべきものを入れて送るものである。したがって、一般ユーザに対しても、終点オプションの送受信を許すことにする。ただし、次の 2 つの理由により、特権利用者にのみ送受信が許された領域を残す。

- Unix では、1024 以下のポート番号を特権利用

者に専有にしている。この仕組みを利用して、接続先が特権利用者の権限を持っていることを識別することができる。同様に、終点オプションを設定した主体が特権利用者の権限を持っていたかを知ることが重要なサービスがある。たとえば、アクセス制御の目的では、この機能を使うべきである。

- 既にいくつかの終点オプションが定義されている。それらの中で一般利用者が用いると問題を引き起こすものが含まれている。

6.3 節で述べたカーネル・モジュールは、特権利用者の権限の制約を緩和する。カーネル内のモジュールは、基本的には特権利用者の権限で動作する。一般利用者は、独自のカーネル・モジュールをインストールすることはできないが、既にインストールされたモジュールから拡張モジュールのインスタンスを生成することは許される。この性質により、一般ユーザの権限でも自らアクセス制御関連のインスタンスを活性化して利用することが可能になる。

6.5 実現

現在までに、UDP/IP において送受信するためのライブラリを部分的に実現した。そして、それを用いて、DNS 名前サーバに対するプロキシを実現した。実現環境は、KAME プロジェクト [3] が提供している FreeBSD 4.7 カーネルである。

このプロキシ・サーバは、終点オプションに含まれた UID を参照し、アクセスが許可されていれば、実際に DNS 名前サーバに要求を中継する。そして、名前サーバから送り返されてきた応答をクライアントに返す。ただし、現在の環境では、クライアント側も終点オプションを送信するためには、特権利用者の権限が必要になっている。

なお、このような UID の添付は、将来的には、6.3 節で述べたカーネル・モジュールで実現すべきものである。

7 おわりに

本研究では、IPv6 の拡張ヘッダの 1 つである終点オプションを用いた付加的なサービスの提供を支援する。付加的なサービスとして、アクセス制御、クッキー機能、および、論理クロックの提供を示した。現在の IPv6 終点オプションの API には、複数のサービス提供プログラムが同時に利用できないという問題点がある。また、カーネル内の API や TCP/IP に対する API が提供されていない。この論文では、UDP/IP のための新たな API を規定した。この API は、複数のサービス提供プログラムが共存可能になっている。それから、TCP/IP に対する API を示した。カーネル内 API は、システムコールに対するラッパ／リファレンスモニタ SysGuard の機能を参考にして設計した。

今後の課題は、今回示した API を全て提供することである。さらに、これらの API を利用して、提案した 3 つのサービスを提供する。

参考文献

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

[2] 榮樂恒太郎, 新城靖, 板野肯三: システム・コールに対するラッパ/リファレンス・モニタ SysGuard の設計と実現, 情報処理学会論文誌, Vol. 43, No. 6, pp. 1690–1701 (2002).

[3] KAME Project: *http://www.kame.net* (2003).

[4] Kent, S. and Atkinson, R.: *Security Architecture for the Internet Protocol*, RFC2401 (1998).

[5] Kristol, D. and Montulli, L.: *HTTP State Management Mechanism*, RFC2965 (2000).

[6] Mamakos, L., Lidl, K., Evarts, J., Carrel, D., Simone, D. and Wheeler, R.: *A Method for Transmitting PPP Over Ethernet (PPPoE)*, RFC2516 (1999).

[7] Mockapetris, P.: *Domain names - implementation and specification*, RFC1035 (1987).

[8] Myers, J.: *SMTP Service Extension for Authentication*, RFC2554 (1999).

[9] Psounis, K.: Active Networks: Applications, Security, Safety, and Architectures, *IEEE Communications Surveys*, pp. 1–16 (1999).

[10] Stevens, W. R., Thomas, M., Nordmark, E. and Jinmei, T.: *Advanced Sockets API for IPv6*, Internet-Draft (March 2003).

[11] 鈴木真一, 新城靖, 光来健一, 千葉滋, 板野肯三: ユーザ権限変更機構を利用した安全なイントラネットサーバの実現, 情報処理学会論文誌:コンピューティングシステム (ACS 2 号) (2003). (印刷中).

[12] Tanenbaum, A. S.: *Distributed Operating Systems*, Prentice Hall,Inc (1995).

[13] Ylonen, T., Kivinen, T., Saarinen, M., Rinne, T. and Lehtinen, S.: *SSH Transport Layer Protocol*, Internet-Draft (September 2002).