6

KAMEプロジェクトによるIPv6の基本ソフトウェア開発

神明 達哉* 山本 和彦** 萩野 純一郎***
江崎 浩**** 村井 純*****

インターネットが急速に発展し，さまざまな応用が期待される一方で，アドレス不足が深刻化してきた．その抜本的な解決策としてのIPv6が注目を集めている．WIDEプロジェクトでは，IPv6の早期普及のために，実運用に耐え得る高品質な実装を広く公開することが重要であると考え，1998年4月にKAMEプロジェクトを結成した．KAMEプロジェクトが開発したIPv6基本ソフトウェアは，結成からの2年間でIPv6の参照実装としての地位を確立した．現在では，すべてのBSD系OSで正式に採用され，商用ルータの開発基盤としても利用されるなど，国際的に広く普及している．本稿では，KAMEプロジェクトによる実装の特徴的な部分を，特にカーネルの内部に焦点をあてて解説する．まずカーネルの全体像を概観する．ここでは，従来のBSD系OSの実装を基本的に踏襲しつつ，入出力処理や近隣探索機能を実現する際に，IPv6のプロトコル仕様に応じた改良を加えていることを示す．また，アドレススコープの概念がBSD系OSで自然に表現できていることを述べ，さらに，受信ヘッダ処理において仕様に準拠しつつ効率化を図ったことを説明する．

IPv6とKAMEプロジェクト

IPv6 (Internet Protocol version 6) [1]は，現行のインターネットプロトコルであるIPv4 (IP version 4) が抱える問題の解決，特にアドレス不足の解消を目的として，IETF (Internet Engineering Task Force) で標準化が進められてきた後継プロトコルである．

IPv4のアドレス空間は32ビットで，最大限に効率よく割り当てても世界人口程度が上限となる．インターネットの爆発的な普及を背景に，IPv4アドレスの不足は年々深刻化しており，現状でも個人が自由にアドレスを取得することは困難である．これに対し，IPv6では128ビットのアドレス空間が採用され，無限ともいえる広大な領域が提供される．

IPv6の基本的な仕様はこれまでにおおむね確定し，実験ネットワークによる試用期を経て，ネットワーク機器・OSベンダによる正式対応が発表され，ISP (Internet Service Provider) による商用サービスも始まるなど，実用段階に入ってきた．また，アドレス空間の拡張を背景として，携帯電話や家庭内ネットワークなど，新たにインターネットの技術が必要とされる領域でもIPv6は注目を集めている．

数年前までは，IPv6への移行が必須であることは誰もが認めつつも，その普及に関しては悪循環に陥っていた．IPv4アドレスが完全に枯渇するまでにはなお数年の余裕があることから，ユーザからの強い需要は発生しない．需要がないために，ベンダ側もOSや機器のIPv6対応には投資しにくく，対応機器がないためにユーザがIPv6を評価する環境も整わない，という状況であった．

早くからIPv6に注目して研究開発を進めてきたWIDEプロジェクトでは，高品質なIPv6対応ソフトウェアを自由に参照できる形で提供することによってこのような悪循環を断ち切ろうと考え，1998年4月にKAMEプロジェクト☆1を結成した．プロジェクトを始めるにあたっては，WIDEプロジェクトを中心として，それまでにIPv6の研究開発で実績を挙げていた技術者を複数の企業から募る形で開発グループを構成した．これにより，KAMEプロジェクトとしては，質の高いソフトウェアを短期間に開発するという目的がより現実的となる．また，派遣元の企業にとっては，次世代インターネットの最先端の知見が得られることになり，両者にとって理想的な関係のプロジェクトとなった．

* （株）東芝 研究開発センター jinmei@isl.rdc.toshiba.co.jp
** IIJ技術研究所 kazu@iijlab.net
*** IIJ技術研究所 itojun@iijlab.net
**** 東京大学情報基盤センタ hiroshi@wide.ad.jp
*****慶應義塾大学環境情報学部 jun@wide.ad.jp

☆1 http://www.kame.net/

プロジェクト名の「KAME」は「かめ」と発音する．この奇抜な名称の由来は，公式には，開発用オフィスの所在地である「刈込（かりごめ）」の省略形であるということになっているが，実際のところは発音そのままの「亀」をローマ字表記しただけである．その出どころは，KAMEプロジェクト発足以前に，WIDEプロジェクトのIPv6分科会が主催したワークショップ中の出来事であった．デバッグに行き詰まったある参加者が，会場にあった大きな亀のぬいぐるみに抱きついて「ああ，亀さん助けて」とつぶやいた．その姿があまりにも印象的であったため，それからほどなくして誕生したプロジェクトの名称も，ごく自然に「KAME」に決まったという次第である．その名付け親ともいうべき名台詞の主は，プロジェクト発足当初からの開発メンバとして活躍している．

KAMEプロジェクトでは，開発の基盤としてBSD（Berkeley Software Distribution）系のオペレーティングシステム（OS）を用いている．BSDにはIPv4の参照実装として長い実績があり，またそのネットワークコードについての文献も豊富で，新たにIPv6の参照実装を開発する際の基盤としてふさわしいOSである．開発の初期段階では，プロジェクトの成果物はこれらのOSに対する差分の形で提供していた．現在では，4つあるBSD系OSの最新版すべてに，IPv6対応用としてKAMEプロジェクトの実装が採用されている．

ネームサーバ「BIND」の開発元であるISC（Internet Software Consortium）や，経路制御ソフトウェアの開発団体として有名なgatedコンソーシアムにおいても，KAMEプロジェクトのソフトウェアがそれぞれのIPv6対応の開発基盤として利用されている．また，エンドユーザ向けOSとしてだけでなく，IPv6対応の商用ルータにも採用されており，その一部はすでに商業ネットワークで運用されている．

日本発のプロジェクトは，その活動が国内に閉じてしまい，世界的には無名のまま消滅するという結果を招くことがある．過去のそうした例における反省から，KAMEプロジェクトは，当初から成果を世界に問うという方針を採った．具体的には，プロジェクトのwebページ，開発物に付属するドキュメント，関連するメーリングリストへのアナウンスはすべて英語のみとし，試用版のユーザで構成するメーリングリストでも英語のみで議論してきた．開発の迅速さや成果物の品質に加えて，こうした姿勢も，BSD系OSや各種組織・ベンダでの採用につながっているといえる．

KAMEプロジェクトは，その性格上，IPv6プロトコルの実装を活動の中心としているが，開発および運用の過程から得られた経験をもとに，IETFにおける仕様の標準化にも積極的に関与してきた．また実際に，Internet Draftとよばれる仕様文書をIETFに提案し，その一部は標準化の対象として議論されている．

こうした成果への評価と，今後への期待とから，当初2年間の予定であったKAMEプロジェクトはさらに2年間延長されることとなった．後半の2年間では，実装活動を中心としつつも，IPv6の普及面にもより力を注ぐ予定である．

本稿では，KAMEプロジェクトで開発したIPv6の基本ソフトウェア（本稿では以後，単にKAMEソフトウェアと呼ぶ）について，その全体構成と，その中でも特に特徴的な部分について解説する．

KAMEプロジェクトによるIPv6の実装

前述の通り，KAMEソフトウェアはBSD系OSの上で開発されている．これらのOSでは，ネットワークのプログラムは大きく2つに分けられる．すなわち，カーネル空間のプロトコルスタックと，ユーザ空間におけるアプリケーションプログラムである．

KAMEプロジェクトの開発対象は，この両方にわたる．カーネル空間での開発は，BSD系OSカーネルに新しいプロトコルスタックを追加する形となる．ユーザ空間については，メール，WWW，FTP，遠隔ログインといった主要な既存ネットワークアプリケーションをIPv6に対応させるとともに，管理ツールや経路制御デーモンのようなIPv6専用アプリケーションを新規に開発した．これらのアプリケーションのうち，一部はすでにBSD系OSの配布物に含まれている．また，既存アプリケーションの一部については，開発元との協力のもと，正式リリースでのIPv6対応も進めている．

これ以後，本稿では，特にカーネル空間のプロトコルスタックを中心に解説する．IPv4とIPv6の間にはプロトコルとしての互換性はないが，基本的な設計思想には共通の部分も多い．したがって，その実装であるソフトウェアの開発に際しても，既存のIPv4プロトコルスタックと同様の手法を一部流用できる．特に，TCP（Transmission Control Protocol）やUDP（User Datagram Protocol）といったトランスポート層のプロトコル処理には，IPv4とIPv6の間にほとんど違いがない．実際，KAMEソフトウェアにおいても，こうしたトランスポート層の実装では，IPv4からの転用か，IPv4とIPv6を統合して1つのスタックにまとめるかのいずれかの形をとっている．

一方，IPv6独自の機能を実装する際には，IPv4のプロトコルスタックで用いられた手法を単純には流用できない場合がある．そこで，KAMEソフトウェアでは，IPv4プロトコルスタックの実装とは異なる独自の手法を一

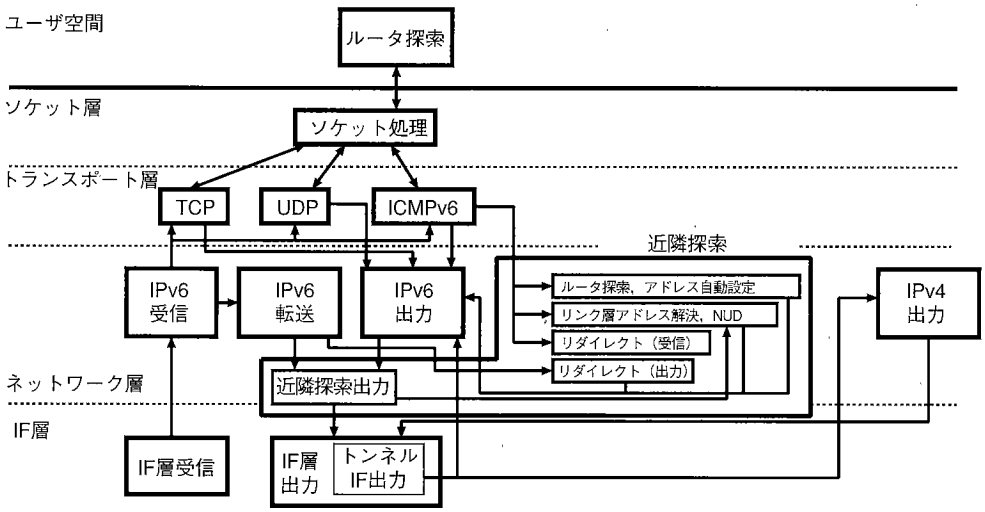

図-1　KAMEソフトウェアのアーキテクチャ

部導入している.

本章では，まず，KAMEソフトウェアにおけるIPv6プロトコル処理の全体構造を概観する．次に，KAMEソフトウェアにおいて独自に導入された手法に焦点をあてて，より詳細に解説する.

■KAMEソフトウェア全体のアーキテクチャ

図-1に，KAMEソフトウェア全体の構成を示す．図の中で，実線より下の部分がカーネルのプロトコルスタックを示しており，それをさらにソケット，トランスポート，ネットワーク，インタフェース（IF）の4層に分けて点線で区切ってある．図-1では，入力，出力，転送といった基本的な処理単位を1つのブロックとして表し，各ブロック間の処理の流れを矢印で示した．これはまた，おおむね入出力パケットの流れも意味している．また，実線の上部はユーザ空間であり，ここでは一般のアプリケーションプログラムが動作する.

層構造や基本的なパケット入出力処理の流れについては，BSDのIPv4プロトコルスタックとほぼ同じである．ただし，KAMEソフトウェアでは，ルータとしてパケットを転送する際にIPv6の送信処理を経由させていない．これは，IPv6では転送時でのパケットの分割（fragmentation）が禁止されたことを受けて，転送処理を簡略化するためである.

IPv6に特徴的な部分としては，近隣探索（Neighbor Discovery）[4]プロトコルの処理が挙げられる．近隣探索プロトコルには複数の機能が含まれるが，ここでは，それらを大きく以下の3つに分類した.

- **ルータ探索とアドレスの自動設定**：ルータ探索（router discovery）は，同一リンク上に存在するルータを認識，管理する機能であり，これを通じて，ホストはデフォルトルータを自動的に設定する．また，アドレスの自動設定（address autoconfiguration）とは，ルータから与えられた情報を元にしてホストのIPv6アドレスを自動的に割り当てる機能である.
- **アドレス解決と到達不能性検出**：アドレス解決（address resolution）は，近隣ノードのIPv6アドレスに対応するリンク層アドレスを求める機能である．また，到達不能性検出（neighbor unreachability detecion，以後単にNUDと呼ぶ）は，近隣ノードへのIPv6パケットが正しく届くかどうかを確認する機能である.
- **リダイレクト**：ルータからホストに対し，同一内リンクでの冗長な転送が起きないように，最善のルータを指示するためのメッセージがリダイレクト（redirect）である．リダイレクトメッセージは，ルータの転送処理の際にルータからのみ送信され，逆にリダイレクトを受信して処理するのはホストのみである．そこで図-1では，リダイレクトを送信と受信の機能に分けて示してある.

図-1からも分かるように，KAMEソフトウェアでは，近隣探索のほぼすべての機能をカーネル内で実装している．これは，近隣探索の機能がIF層や経路表など，カーネルの内部情報と密接に関連していることによる．ただし，ルータ探索の一部の機能，特にルータからのメッセージ送信処理は，カーネルの内部情報から比較的独立しているため，開発効率や設定の簡易化を考慮してユーザ空間のアプリケーションとして実装している.

近隣探索プロトコルはICMPv6メッセージを利用するので，メッセージ処理という観点からは，ICMPv6ソケットを利用してすべてユーザ空間で処理するという方法も考えられる．実際，仏INRIAによるIPv6の実装では，すべての近隣探索機能をユーザ空間のアプリケーションとして実現している．一般に，ユーザ空間での開発は，カーネルでのそれと比べてデバッグが容易であり，開発効率の点では優れている．一方，カーネルの内部情報を参照または操作するためにカーネルとユーザ空間の

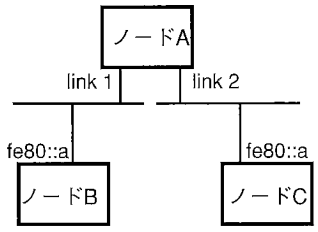

図-2　リンクローカルアドレスの一意性

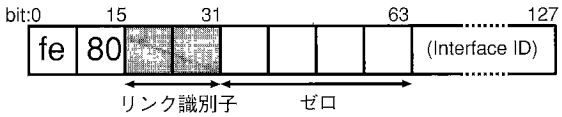

図-3　KAMEソフトウェアにおける
リンクローカルアドレスの内部表現

やりとりが必要になるため，実行効率の点では不利である．

送信および転送処理を経たパケットは，まず近隣探索としての送信処理を施される．ここで，必要があればリンク層アドレス解決やNUDのための処理を行う．IPv4の実装では，IPの出力処理から直接IF層の出力処理を呼び出し，そこからさらにリンク層アドレスの解決のためにARP (Address Resolution Protocol) の処理を呼び出しているが，KAMEソフトウェアでは，IF層での出力処理の前に近隣探索の処理を挟んでいることになる．近隣探索プロトコルそのものはIFに依存しないので，このように順序を変えてもプロトコル処理上は支障がない．また，近隣探索の処理を先に済ませることで，IFに障害が生じて出力処理が途中で打ち切られるような場合にもNUDが正常に動作するといった利点が得られる．

IPv4からIPv6への移行期に重要な技術として，IPv6パケットをIPv4パケットにカプセル化して送るトンネリング技術がある．トンネリングにより，IPv6バックボーンへの直接の接続性がない場合にも，IPv4ネットワークを介して複数のIPv6ネットワークを相互に接続できる．KAMEソフトウェアでは，トンネリングのためのカプセル化処理をネットワークIFとして抽象化[2]している．本稿では，このIFをトンネルIFと呼ぶ．

トンネリングを必要とする宛先アドレスに対しては，経路表の送信IFとしてトンネルIFが指定され，その宛先アドレス向けのパケットはトンネルIFの出力処理に渡される．その中で必要なカプセル化処理を行い，カプセル化後の外側のヘッダに対応する適切なIP層の出力処理を経て，再びIF層に全体のパケットが出力される．

IPv6への移行技術としては，専らIPv6パケットをIPv4パケット内にカプセル化するトンネリングが利用されるが，トンネルIF自体は汎用的に実装されており，同一プロトコルへのカプセル化を含め，どのような組合せのカプセル化も可能である．このため，設定によっては，カプセル化を繰り返してカーネルのスタック領域を使い尽くす一種のループが発生することがある．KAMEソフトウェアでは，これを防ぐために，カプセル化の繰り返し回数を記録し，一定の上限値に達した場合に処理を止める仕組みを導入している．

■アドレススコープの扱い

IPv6のアドレス体系[3]には，有効範囲（スコープ）の概念が導入されている．あるスコープに属する範囲をゾーンと呼び，アドレスの一意性はそのゾーン内でのみ保証される．また，各IPv6パケットは，その送信元または宛先アドレスがそれぞれ属するゾーンを越えて転送されることはない．

現在，ユニキャストアドレス用に，リンクローカル，サイトローカル，グローバルという3種類のスコープが定義されている．特に，リンクローカルアドレスは，近隣探索や経路制御プロトコルにおいて重要な役割を果たす．リンクローカルアドレスは，fe80::/10という固定のプレフィクスで識別される．

上で述べたアドレスの一意性に関する性質から，ゾーン境界にあたるノードでは，各ゾーンごとにアドレスを識別する必要がある．たとえば，**図-2**において，ノードAはlink 1およびlink 2のリンク境界に属している．このとき，link 1およびlink 2上に，同じリンクローカルアドレスfe80::aを持つノードBおよびCが存在し得るため，ノードAにおいてBまたはCを識別するためには，そのリンクローカルアドレスに加えて，どのリンクのアドレスであるかという情報が必要になる．

そこで，KAMEソフトウェアでは，128ビットのリンクローカルアドレスのうち，16-31ビットまでの16ビットにリンクの識別子を埋め込んだアドレスを内部表現として用いている（**図-3**）．この表現を用いると，図-2におけるノードBおよびCのリンクローカルアドレスは，それぞれfe80:1::aおよびfe80:2::aとなる．この内部表現はカーネルの中に閉じており，ノードの外およびユーザ空間のアプリケーションからは隠蔽される．

この方式の最大の利点は，128ビットのアドレス自体にリンクの識別子を埋め込むために，リンク境界のノードにおいても経路表を1つにまとめられることである．すなわち，単一の経路表のままで各リンクごとの経路を管理できるため，BSD系OSの既存の経路表処理部分との整合性がよい．ただし，この方式では，入出力パ

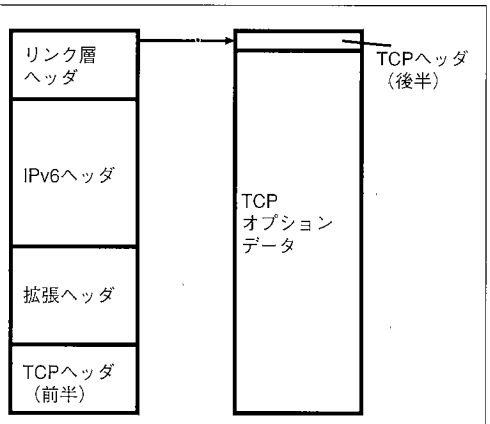

図-4　TCPヘッダが2つのmbufにまたがって格納される例

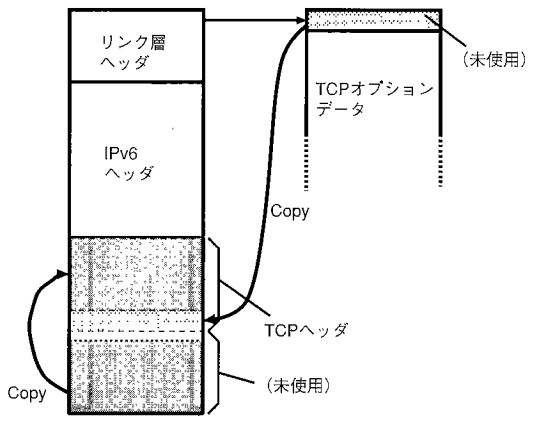

図-5　拡張ヘッダを削除し，TCPヘッダを連続領域にコピーした状態

ケットに含まれるアドレスを処理する際に，常にリンクローカルアドレスかどうかを確認し，必要があればリンクの識別子を埋め込む，または取り除くといった作業が必要となる．KAMEソフトウェアの実装は，後者の負荷よりも前者の効果をより重視した結果であるといえる．

すでに述べたように，埋め込まれた識別子は，原則としてユーザ空間のアプリケーションからは隠蔽される．しかし，カーネル内のメモリを直接読み取るようなアプリケーションや，経路表操作のための特別なソケット（ルーティングソケット）を利用するアプリケーションは，埋め込まれた識別子を直接扱う必要がある．前者についてはこれは自明である．後者については，ルーティングソケットのカーネル内実装がネットワークプロトコル非依存の一般的な構造になっており，その中でIPv6に特有の操作を行うことが不自然であったという事情による．

このような点に対処するため，リンクの識別子をアドレスに埋め込む代わりに，IPv6用のソケットアドレス構造体内に定義されているスコープ識別子メンバを利用する方式も検討中である．この方法では，単一の経路表で複数のリンクの経路を管理するという利点を残しつつ，アプリケーションに対する完全な透過性も実現できる．

なお，本節でこれまでに述べた事項は，理論上はサイトローカルアドレスについても適用できる．ただし，現状ではサイトローカルアドレスは実運用上さほど広く使用されていないため，サイト境界のノードにおいてサイトの識別子を考慮する機能は実装していない．

■受信パケットのヘッダ処理

BSD系のOSでは，ネットワークのパケットを保持するためにmbufと呼ばれるデータ構造を用いる．Mbufは，データを格納するためのデータ領域と，その領域を管理するためのヘッダから構成され，必要に応じて複数のmbufを鎖状につなげて利用する．

リンク層のデバイスドライバは，受信したパケットをmbufに格納してネットワーク層の入力処理へ渡す．その際，パケットが1つのmbufのデータ領域に収まらない場合には，以下のいずれかの方法をとる．

(1) 別のmbufのデータ領域に残りのデータを収容し，2つのmbufを鎖状につなぐ．

(2) 外部領域とよばれる空間を別に割り当て，データをその空間内に収容する．この場合，mbuf内では外部領域へのポインタのみ保持し，mbuf自身のデータ領域は利用しない．

このうち，特に前者の方法では，IPv6ヘッダやトランスポート層のヘッダ，さらにもし含まれていればIPv6の拡張ヘッダが2つのmbufのデータ領域にまたがって格納される可能性がある（**図-4**の例参照）．同様に，IPv4でもトランスポート層ヘッダが2つのmbufにまたがる場合がある．

個々のヘッダの入力処理においてこのような不連続なデータを扱うのは不便であるため，従来のIPv4の実装では，IPヘッダとトランスポート層ヘッダの中間のオプションを削除したうえで，IPヘッダとトランスポート層のヘッダが連続した領域に存在するようにデータをコピーしている．それと同様の手法を図-4のパケットにそのまま適用したものが**図-5**である．すなわち，IPv6ヘッダとTCPヘッダの間の拡張ヘッダを削除し，それによって生まれた空間にTCPヘッダをコピーしている．

こうした細かいバッファ操作は，IPv4がBSD系のOSに実装された当初のように，メモリが貴重な資源である場合には有効である．しかし，今日ではメモリのビット単価が下がる一方でネットワークの処理性能に対す

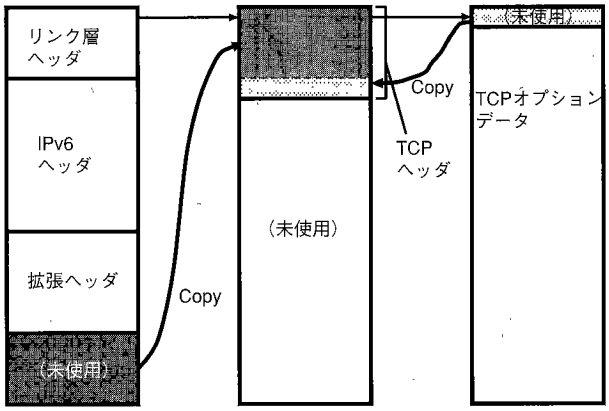

図-6　新規のmbufを確保してTCPヘッダのみをコピーする方式

る要求が厳しくなっており，むしろコピーによる性能低下を問題視すべきである．

また，IPv6では，パケット処理の途中で生じたエラーによりICMPエラーメッセージを送信する場合には，そのデータ部分に受信パケットをそのままの形で可能な限り長く含めるよう定められている．これを実現するためには，中間のヘッダを削除する方法は不適切である．

こうしたことから，KAMEソフトウェアでは，先頭のmbufに受信パケットが収納できない場合は，常に外部領域を確保するという方針をとった．このため，(1) の型のデバイスドライバを書き換えて，常に (2) の方法を用いるようにしている．

外部領域のバッファ長はi386系のアーキテクチャでは2048バイトであり，現在のLAN環境の主流であるイーサネットの最大伝送単位 (MTU) が1500バイトであることを考えれば，このバッファはすべてのヘッダを格納するのに十分な大きさである．

ただし，この方法では，(1) 型のデバイスドライバが現れるたびに (2) 型に書き換える必要がある．おそらくはメモリ容量に対する要求の変化から，今日では (2) 型のデバイスドライバが増えているが，すべての実装者に対して (2) 型を要求することは不可能である．

また，たとえばループバックIFでは，送信時のmbufの鎖を保持したまま受信処理へ送ることで，mbufの解放，割り当てやデータのコピーを防いでいる．ところが，KAMEソフトウェアの出力パケット処理では，基本的に拡張ヘッダおよびプロトコルヘッダごとに1つmbufを割り当てるため，出力されるすべてのデータが1つのmbufに収まるとは限らない．このため，ループバックIFの送信処理において，2つ以上のmbufからなる出力パケットについては，外部領域を新たに確保してコピーし直すという処理を加えており，ここではむしろ性能を低下させる形となっている．

そこで，第3の方法として，デバイスドライバを変更せずに，また中間の拡張ヘッダも保持したまま，データコピーの回数をなるべく減らす方式の実装も進めている．この方法では，必要なデータが複数のmbufにまたがっている場合，新規のmbufを1つ用意し，そのデータ領域に，前後のmbufからそれぞれ必要なデータをコピーする．そのうえで，新規の1つを加えた3つのmbufを鎖状につなぐ．図-4に示した例をこの方法で処理した場合の最終的な形が**図-6**である．

この方法では，既存のデータ領域を上書きすることはないため，拡張ヘッダを含むすべてのヘッダをそのままの形で保持できる．また，必要に応じてコピーをすることで，デバイスドライバに対する要求を緩和し，ループバックIFでのデータコピーも防いでいる．さらに，パケットが (2) 型のデバイスドライバによって格納されている場合には，通常，まったくコピーを必要としない．すでに述べたように，(2) 型のデバイスドライバが増えてきている現状から，実際にコピーが必要となる場面は少ないと予想できる．

今後の予定

ここまで，特にIPv4とIPv6の相違点に由来する，KAMEソフトウェアの特徴的な部分をいくつか取り上げて説明してきた．KAMEソフトウェアは，基本構造としては，成熟した実装であるBSD系OSのプロトコルスタックを利用している．そのうえで，近隣探索プロトコル，アドレススコープ，拡張ヘッダといったIPv6独自の機能を自然に，あるいは効率よく実現するよう改良した実装だといえる．

最後に，KAMEプロジェクトにおける今後の開発項目を述べる．本文でも触れたように，すでにひと通りの実装が完了している機能の中にも，さらに改良，または整理を要する項目がある．たとえば，アドレススコープやmbufの扱いがそれにあたる．

また，本稿では割愛したが，IPsec (IP security) やモバイルIPといった応用機能もIPv6の特徴の1つとして注目されている．KAMEプロジェクトではこうした機能の実装も進めているが，今後これらの実装の完成度を上げるとともに，可用性の確認，仕様自体への検討を通じて，普及面にも力を注ぐ予定である．

参考文献

1) Deering, S. and Hinden, R: Internet Protocol, Version 6 (IPv6) Specification, RFC 2460 (1998).
2) Demizu, N. and Yamaguchi, S: DDT — A Versatile Tunneling Technology, In Proceedings of INET'94/JENC5, pp.661-1-661-9 (1994).
3) Hinden, R. and Deering, S: IP Version 6 Addressing Architecture, RFC 2373 (1998).
4) Narten, T., Nordmark, S. and Simpson, W. A.: Neighbor Discovery for IP Version 6 (IPv6), RFC 2461 (1998).

(平成12年11月1日受付)