実時間通信のための次世代インターネットプロトコルサーバ*

尾上裕子
NTT 情報通信研究所

木原誠司
NTT 情報通信研究所

舩渡大地
慶應義塾大学環境情報学部

松井康範
慶應義塾大学環境情報学部

徳田英幸
慶應義塾大学環境情報学部
カーネギーメロン大学計算機科学部

次世代インターネットプロトコル IPv6 では、24 ビットのフローラベルや優先度フィールドなど、実時間通信処理サービスへの要求に応じた QOS サポートが考慮されており、今後の課題として (1) フローラベル毎のフロー制御、(2) ATM の QOS コネクションとのマッピング、(3) ルーティングプロトコルや NDP (Neighbor Discovery Protocol) とのキャッシュ情報の一貫性管理、(4) キューでの優先度制御、(5) パスと資源予約制御が挙げられる。さらに、慶應大学を中心とした次世代マイクロカーネル (MKng) 研究プロジェクトにおける、RT- Mach (Realtime Mach) OS 上でのリアルタイムスレッドを用いた次世代インターネットプロトコルサーバの実装と，通常のファイル転送プロトコル (v6) による輻輳時のジッタの評価結果について述べる。

1　はじめに

現在、音声・画像など時間制約を持つ連続メディアデータを、インターネット上でリアルタイムに通信することへの要求が高まっている。MBONE(Multicast backBONE) や VOD(Video On Demand) による双方向多対多または一方向放送型通信サービスなど、リアルタイム通信サービスの利用が先攻しており、実際にはネットワークや中継ルータの CPU の負荷・輻輳による遅延時間やそのジッタ、パケットロスなどに悩まされている。特にネットワークにおいては、ATM やセルスイッチルータによる高速かつ転送処理性能保証可能なネットワークの利用と、エンドツーエンドで予測可能サービスを提供するためのリアルタイム通信プロトコルが望まれる [2][12][13][14]。

2　次世代インターネットプロトコルにおける実時間通信機構

次世代インターネットプロトコル IPv6[1] は、インターネットの急速な規模拡大に伴うスケーリング問題を解決し、トンネリング技術をはじめとした既存の IPv4 からの柔軟な移行の機構を提供しつつ、リアルタイムフロー通信、ホストアドレス自動設定 (プラグアンドプレイ) による移動体通信、セキュリティなどの新しい機能をサポートする。具体的機能としては、(1) ヘッダフォーマットの簡略化、(2) オプションのフォーマット化、(3) アドレス機構の拡張、(4) QOS(Quality Of Service) 制御のためのフローラベル導入、(5) セキュリティ、認証機能のサポートが挙げられる。

表 1,2に、次世代インターネットプロトコルと従来 (IPv4) との比較を示す。

(* は、リアルタイム通信向けに IPv6 に導入された機能)

表 1: IPv4 と IPv6 との比較

	IPv4	IPv6
ヘッダフィールド		
アドレス長	ソース / 宛先アドレス共に 32 ビット	ソース / 宛先アドレス共に 128 ビット
*フロー ID	宛て先アドレスとポートの組を上位プロトコルでのぞき見して判別。	送信ノードがランダムに決定するフロー ID を導入。QOS を指定する特定のストリームに割り当てられる。
*プライオリティフィールド	TOS(Type Of Service) フィールドパケットにセットされてはいたが、(ex.4.4BSD ftp のコントロールデータコネクション) 処理ルータはほとんど無かった。	0 - 7: ノンリアルタイム用。 8 - 16: リアルタイム用。
チェックサム	16 ビットのヘッダチェックサム。	ヘッダフィールドからなくなった。
ペイロード長	16 ビットの Total Length フィールドで IP ヘッダを含む。	16 ビットの Payload Length で IP ヘッダは含まない。
長さの制限	最大 64K バイトまで。	Jumbo Payload Option で 64K バイト以上が可能。
オプションヘッダ	ルータは全てのオプションをチェック。	オプションを柔軟にサポートするため、拡張ヘッダとした。デフォルトではルータは見ない。
オプションヘッダ		
ルーティング	ソースルーティングとして中継を指定。	ルーティングヘッダで Strict または Loose を指定。
フラグメンテーション	中継ノードで MTU に応じてフラグメント。	フラグメンテーションヘッダでソースノードのみでデータグラムをフラグメント。
セキュリティ	単なるエンカプスレーションとして実現。	セキュリティヘッダの導入。
ホップバイホップ	ルータは全てのオプションを見るので、オプションヘッダは意味的には全てホップバイホップオプション。	ホップバイホップヘッダにより、ルータが見るパケットと見ないパケットを区別。

*Next Generation Internet Protocol Servers for Realtime Communications, Yuko ONOE, Seiji KIHARA (NTT), Daichi FUNATO, Yasunori MATSUI (Keio University), Hideyuki TOKUDA (Carnegie Mellon University / Keio University)

表 2: IPv4 と IPv6 との比較

	IPv4	IPv6
アドレス		
マルチキャスト	ホスト内ブロードキャスト 127.0.0.1 を使用。	anycast アドレスを使用し、ホスト内マルチキャストとなる。scope の概念導入により node/ link/ site/ organization/ global を区別する。
リンクアドレス	アドレス解決のため別プロトコル ARP (Address Resolution Protocol) を用意し、IP アドレスをネットワークアドレスへマッピング。ICMP Router Discovery/ ICMP Redirect を使用。	リンクローカル IP アドレスの導入により、NDP として IP プロトコル内でアドレス解決。
割り当て	DHCP などステートフルアドレス設定。	ステートレスアドレス自動設定で、サーバがプレフィックスを宣言する。
ICMP		
グループ管理	IGMP(Internet Group Management Protocol) として別プロトコル 。	ICMPv6 に吸収され、query/ report/ reduction を定義している。
チェックサム	IP ヘッダのチェックサムとは別に ICMP のチェックサム。	ICMPv6 のチェックサムフィールドの計算に IP の pseudo-header を含む。
MTU 検知	途中の中継ノードでフラグメント。	Path MTU Discovery による。
その他		
RSVP	UDP/IP 上。	IP データグラムのペイロード内へ、NextHdr=46 として

2.1 リアルタイム通信支援機能とフローの扱い

フローラベルおよびプライオリティクラスは v6 ヘッダフィールドとして定義され、インターネットにおいて品質の高いマルチメディア通信を提供するためリアルタイムフローを認識する [7]。ここでフローとは、特定のソースから特定の宛先へ転送されるパケットの連続で、同一フローラベルのパケットは、以下の性質を合わせ持つ。

- 宛先アドレス、次ホップノードが同一。
- 同一な資源予約グループおよびキューグループに属す。
- 同一のホップバイホップオプション、ルーティングヘッダを持つ。

2.1.1 フロー ID

これまでは、TCP/UDP など上位プロトコルのポート番号と宛先 IP アドレス、IP の IDENT フィールドを盗み見することにより、ネットワークレイヤレベルでフローを認識していた。しかし、IPv6 のセキュリティヘッダを用いてパケットが暗号化されると、ネットワークレイヤレベルでポートをのぞき見することができなくなる。

さらに従来の IP ヘッダ情報だけではわからないことを判別するとき、つまり宛先ノードのアドレスが同じでもネットワークプロトコルレベルでの処理結果が異なるとき、宛先アドレスによるキャッシュだけでは不充分となる。例えば、同一ホスト間の複数の TV 会議システムフローがそれぞれ必要とする帯域を確保しようとするとき、さらにネットワーク負荷分散のため特定のフローに対してソースルーティングを行い、フローそれぞれ経路が異なる場合もある。

このように、同一ホスト上の複数のコネクション毎の資源管理、経路制御、フロー制御、優先度制御、アクセス制御のためフロー ID が必要となり、この場合にはフロー制御用プロトコルなどを新たに組み込み、明示的にフロー ID を管理する機構が必要となる。さらに、実装においてフロー ID による処理の効率化、高速化などが望める。

2.2 他の実時間通信プロトコルの IPv6 対応

2.2.1 RSVPv6

RSVP(Resource reservation protocol[11]) は、現在リアルタイム通信用資源予約プロトコルの一つとして注目されており、通信の分散性および動的変化への追従性に向け、受信側主導の予約方式を採用している。この受信者主導型の予約を実現するため、RSVP では、受信ノードからマルチキャストグループへの ICMP の join メッセージに続き、送信ノードから Path メッセージを送信し (フロー ID によるパスの予約)、受信ノードから Resv メッセージを送信し (フロー ID による資源の予約)、送信ノードからメディアフローを転送する (フロー ID によるフォワード処理の予約)。

ここでネットワークパスを予約するにあたり、予約パスを決定する Path メッセージに対し、マルチメディア通信のような継続的な大容量データは、ATM や FDDI 同期モードのような予約可能なデータリンクを含む特定のパスや、資源予約デーモンが作動しているルータをつなげる様に通ることなどの要求条件が挙げられる。そこで、各受信ノードに対し途中のルータをある程度指定するため、IPv6 のルーティングヘッダを使用することが可能である。

2.2.2 RTPv6

RTP(Realtime Transport Protocol[8]) RSVP による予約資源との対応付けや、トランスレータおよびミクサーを通るための経路指定、ルータにおけるトラヒックモニタリングやシェーピングなど、ネットワークレベルでの制御が望まれる場合に、この送信セッションを一意に対応するフロー ID を使用することになる。

2.2.3 IPv6 over ATM

ルータと ATM を統合した IP スイッチでは、ルータ部からパケット中継処理を切り離し、ATM スイッチ部でカットスルーして転送速度の高速化を図っている [4][5]。このとき、RSVP の資源予約要求メッセージにより、ホストと IP スイッチ間で新しい QOS チャネルを接続し、IP スイッチの ATM スイッチ部で接続ホスト間のチャネルを直結させ、フロー ID を割り当て、ホスト間はフロー ID により通信する。このとき、トラヒック量の多くないコネクションレス型の通信では、新しいコネクション確立や処理の切り離しによるオーバーヘッドの割合が大きくなり、結果的に中継処理が遅れてしまう。一方、マルチメディア通信のような持続的なコネクションを要求するアプリケーションの場合に有効となる。そのためフロー分類として、(1) パケットの TCP/IP ヘッダの送信元 / 送信先アドレスと TCP/UDP ポートフィールドが同じなら同一フローとみなすタイプと、(2)TCP/IP ヘッダの送信元 / 送信先アドレスが同じであるパケットが 60 秒間に

13回以上確認されたら同一フローとみなすタイプがある。この分類により、ファイル転送(ftp)、ファイル共有(NFS)、Web のブラウジング(http)、マルチメディア通信などの継続的なパケット転送の場合に、IP スイッチコントローラが IFMP(Ipsilon Flow Management Protocol) パケットを上流ノードに転送し、特定フローのそれ以降のトラヒックは指定された VC を用いてスイッチングされる。

ここで、IPv6 でフローの分類を行う場合、まず ftp-data, NFS, http などフローを認識する必要のある well-known な宛先ポート番号に対し、あらかじめ確保しておいた well-known なフロー ID を割り当て、そのフローに属するパケットは必ずスイッチングする。また、マルチメディア通信のようなフローについては、コントローラ部分の実測に基づき決定する。

1. 特定の well-known なフロー ID のパケットに対し、送信アドレスとそのフロー ID でスイッチングを行う。(well-known type)
2. それ以外のフローに対し、以下の2種類が考えられる。
   - RSVP などの明示的な IFMP 処理要求 (static type)
   - 同一な送信アドレスとフロー ID のパケットがある程度継続 (dynamic type)

このように、IP スイッチのスイッチング処理をはじめとして、ルータにおける特定フローの高速フォワード処理を促したいとき、例えば well-known なポート番号のサービスに対し送信ノードでフロー ID を割り当てることとする。

2.3 フロー ID の利用

2.3.1 フロー ID 毎の保持情報

中継ノードでは、ネットワーク資源管理や経路制御情報など、宛先アドレスは同じでも異なる振る舞いを要求するフローにおいて、フロー内では同じであるべき以下の情報をフロー ID 毎に保持する。

1. 資源予約情報
   RSVP をはじめとする資源予約プロトコルにより、バッファ長・遅延時間・ジッタ・スループットなどの FlowSpec 予約値をフロー ID 毎に保持する。
2. ルーティング情報、オプション処理結果
   経路制御プロトコルにより、宛先アドレスから次ノードアドレスをキャッシュする。さらに、ルーティング拡張ヘッダオプションで指定された中継ノードから得られる次ノードアドレスを、ルーティング情報より優先して使用する。
3. ICMP 処理情報、ローカルネットワーク (NDP) 情報
   次ノードアドレスから得られる次ノードリンクアドレス・次ノードアップ状況をキャッシュする。
4. フロー毎のキュー管理情報
   キューの属性として、キューのリミット値・タイプ・周期と転送期間・優先度とキュー制御ポリシを保持する。
5. フロー毎のトラヒックモニタリング情報、スケジューリング情報
   フローモニタ機能による平均スループット・平均遅延などの、ネットワーク資源の利用量。

2.3.2 フロー毎のキャッシュの利用

自ノード受信処理、マルチキャストフォワード処理、フォワード処理、ICMP 返答処理と、各処理結果をフロー ID 毎にキャッシュする。このとき、ルーティングプロトコルの管理テーブル内の情報と重複または関連しているため、ルーティングテーブル、NDP テーブル、フロー ID テーブルの一貫性確保の方法として、(1)3つを1テーブルで共通管理、(2)3テーブルの一貫性管理、(3) ステートフル型のフロー管理が挙げられる。

そもそもフロー ID は、複数の送信ノードが独立に ID を割り当てそれを検索キーに用いるときの衝突による検索遅延を少なくするために、送信ノードでランダムに割り当てることになっている。そのため、個別のデータベースとしてキーを作成するが、フロー ID のテーブルの内容は各プロトコルへのポインタとして参照のみとし、変更はそれぞれのプロトコルが行う。例えば、MAC アドレスの変更時には、NDP テーブルだけを変更することになる。

2.3.3 処理の予約, NDP の予約

マルチメディア情報の持つ連続性や新鮮度などの特徴から、中継での遅延を最小にとどめるという要求がある。現在の NDP のリンクアドレス確保の方法では、NA 受信後一定時間経過すると、元のリンクアドレスで取りあえずパケットを送信しておき、さらに NS (Neighbor Solicitation) を送信する。また、一定期間 NDP によるリンクアドレスを確保するため、定期的な NA を依頼し、最後の NA からある時間過ぎても受信されなかったら NS を送信する。このように予約プロトコルにより最低限必要な資源をフロー ID で予約すると同時に、NDP の予約により必要なリンクアドレスをあらかじめ確保しておくことも可能である。

2.3.4 フロー毎のコントロールメッセージ処理

ルーティング消滅や NDP の Neighbor Unreachability Detection (NUD) 処理による次ノードダウン検出など、ネットワークトポロジー変化をフロー ID のキャッシュテーブルに反映させるため、キャッシュは NDP キャッシュの状態変化に対応させる必要があり、NDP の NDU 処理、またはフロー ID が指定された ICMP によりキャッシュ変更またはクリアする。

さらに、マルチキャストグループ変化に対しては、フロー ID を持つパケットの宛先がマルチキャストグループに対する場合、ICMP Group Join, Group Leave, Group Term によりあらかじめフロー ID のキャッシュ情報を保持、追加、削除を行う。

2.3.5 フロー ID, プライオリティ毎のパケットフィルタ処理

IPv6 上では送信側でフローの階層毎にフロー ID を振り、送信ノードアドレスとの組でフローフィルタ設定要求およびルータにおけるフロー制御を行い、フロー内優先度の調整をする。これにより、ルータにおいて IPv6 のヘッダフィールドでフォワードを決定することが可能となり、オプションヘッダ長が任意であることによる転送処理効率低下が避けられる。

さらに、中継ルータが自発的に輻輳処理をするためにパケットを廃棄する場合には、フロー間優先度を示すプライオリティフィールドを使用する。例えば音声通信のパケットは

高優先度、画像通信はより低優先度に設定し、中継ルータにおける輻輳時には両通信が競合しメディア品質が乱れることから音声通信を保護する。

このようにメディアフローの階層毎にフローIDを振り、フロー毎に資源制御、キュー管理、キャッシュ使用など、フロー内の優先制御を行う[6]。また、プライオリティフィールドでフロー間に優先順位付けを行い、中継ルータにおける輻輳回避のためのパケット廃棄を行い、例えば音声通信を画像通信から保護する。

2.3.6 フローID毎のキュー制御

ネットワークドライバとサーバ間で、フローID毎に異なるリアルタイムキューを生成し、許可された帯域幅の定期的な割り当てなどの枠組みを提供する。まず、キューは表3に示す属性により定義される。

表3: リアルタイムキューの属性

queue_ID	送信ノードアドレスとフローIDの組により、メディアフローを識別する。
queue_len (max, limit)	limit 経過後は、キュー制御ポリシによりキュー内に溜まっているバッファは廃棄される。
queue_type (RT/nRT)	RT/nRTタイムキューでは queue_sync の属性に従い、周期的な帯域使用の保証がなされる。ノンリアルタイムキューからの転送は、上記以外の期間に最高優先度でスケジュールされる。
queue_sync (cycle, term)	転送開始の周期と転送期間を指定する。
queue_pri	別のキューの転送期間に、そのキューから転送されるべきバッファが無い場合、または空き時間に優先度の高いキューから順にバッファ転送する。
queue_policy	TIMEOUT(一定期間以上にキュー内にいた古いバッファを廃棄)、PRI DROP(低優先度のバッファをキューから間引く)、PRI SHUTOUT(低優先度のバッファをキューに入れない) を指定する。

データリンクレイヤにおいては、以下の手順に従ってパケット転送を行う。

- イベントドリブン型による start 起動
  1. RT キューへ ENQUEUE
  2. nonRT キューへ ENQUEUE
  3. ネットワークボードの transmit done のインタラプト
- 周期的に start 起動
  1. 開始時間 から period 毎に tflag をセット
  2. 開始時間 + term から period 毎に tflag をアンセット
- start 処理
  1. ボードが transmit 中ならリターン
  2. tflag がセットされいたら、RT キューから DEQUEUE
  3. tflag がセットされていない、かつ転送しようとするパケットが次の tflag をセットする時間までに送りきれるとき、nonRT キューから DEQUEUE

3 次世代インターネットプロトコルサーバの設計

3.1 RT-Mach の利用

本研究の基本ソフトウェアプラットフォームとして、米国カーネギーメロン大学と慶應義塾大学とで共同開発されている Real-Time Mach RMK95[9][10] を適用し、次世代ネットワークプロトコルサーバを実装中である。この RT-Mach は、周期やデッドライン、優先度などを指定できるリアルタイムスレッドモデル、プライオリティインバージョンを回避したリアルタイム同期メカニズム、各ポリシ選択するリアルタイムスケジューリングなどを特徴とする。

3.2 サーバ・ライブラリ構成

プロトコル実装に際し、OS カーネル内実装と比較してプロトコルサーバ実装の方が、柔軟性や変更および最適化の容易性の面から望まれる。しかしサーバ実装では、バッファコピーのオーバヘッドが大きいため、パフォーマンスの低下が予想される。そのため、アプリケーションアドレス空間内にリンクされたライブラリ内にネットワークプロトコルを実装する試みがなされている[3]。次世代ネットワークプロトコルサーバにおいて、表4にライブラリ・サーバ間で提供される機能を示す。

表4: サーバ・ライブラリ内の処理

ライブラリ内	BSD ソケットインタフェースの提供 通常のシステムコールとの振り分け ポート番号とサーバが管理する ID とのマッピングおよびサーバとの通信 セッションのファイルディスクリプタ状態管理 RT データ用ネットワークインタフェースの提供 RT データ用ヘッダ情報の取得 RT データ送受信
サーバ内	TCP/IP, UDP/IP のプロトコル処理 ICMP プロトコル処理 NDP プロトコル処理 ルーティング制御情報の保持 ネットワークコネクションの確立、解除 TCP セッションの状態管理 ファイルのファイルディスクリプタ状態管理 non RT データ送受信

連続メディア用プロトコル処理においては、セッション確立後はデータ転送処理が定期的かつ単調で、制御処理のためのサーバとのインタラクションは、他の通信より少ないと考えられる。さらに、処理の高速性およびジッタを小さく抑えることが重要となる点から、転送処理はライブラリ内実装とした。

3.3 次世代インターネットプロトコルサーバの構成

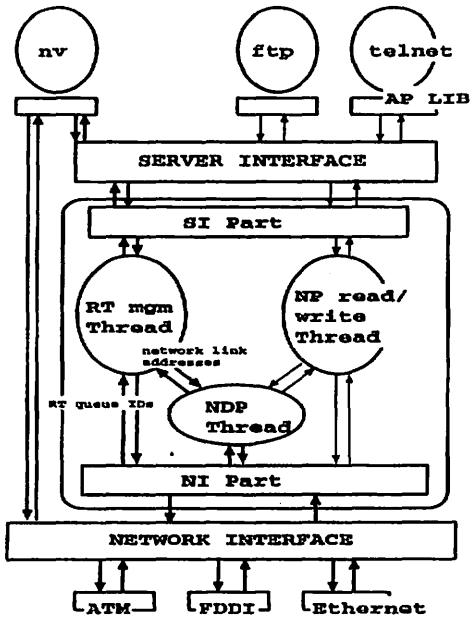

図 1: サーバ構成図

3.3.1 アプリケーションライブラリ部

トランスポートレイヤでは上位アプリケーションにより v4, v6 を使い分ける必要があり、このときネットワークレイヤ分離型 (TCP/IPv4 または TCP/IPv6) と、トランスポート分離型 (TCP/IPv4, TCP'/IPv6) の 2 種類がある。例えば IPv4 のソケットインタフェースを使用してコネクション接続要求待ちしているアプリケーションに IPv6 アドレスで接続してきた場合、前者は受信するが後者は受信しないことになり、ここでは後者としている。

ソケットが使用するデータ構造として、アドレスファミリー、プロトコルのポート番号、ネットワークアドレスがあるが、現在のデータ構造体は固定長のためそのまま使用することができず、別の構造体を新たに追加した。これに対し TLI(Transport Level Interface) では、アプリケーションにはアドレス長とアドレスへのポインタが返るので、変更無しで IPv6 アドレスを取り込むことが出来る。

またリアルタイムフローに対しては、サーバにヘッダ情報取得要求を依頼し、ヘッダ情報とリアルタイムキュー ID を受け取り、アプリケーションからの転送データにヘッダを追加してリアルタイムキューに直接転送パケットを入れる。リアルタイムフローの受信側では、フロー ID を指定したネットワークパケットフィルタをサーバと別にセットし、フロー毎に個別にパケットを受信する。

以下に、通常の BSD ソケットインタフェースに新しく追加した関数のインタフェースを示す。

- get_header(int s, char *bufp, size_t datalen, size_t &headlen, int flags, struct sockaddr_in6 *to, int tolen)
- change_header(int s, char *bufp, size_t datalen, size_t &headlen)
- get_flowid(u_int32_t &flowid)
- get_ds_port(mach_port_t ds_port)
- set_ds_rt_queue(mach_port_t ds_port, u_int32_t flowid, struct queue_attr *q_attr)
- set_ds_packet_filter(mach_port_t ds_port, mach_port_t rport, u_int32_t flowid)
- rt_sendto(mach_port_t ds_port, u_int32_t flowid, char *bufp, size_t buflen)
- rt_recvfrom(int s, mach_port_t rport, char *bufp, size_t buflen, int flags, struct sockaddr_in6 *from, int *fromlen)

3.3.2 アプリケーションインタフェース部

トランスポートプロトコルへのメッセージの振り分けを行い、アプリケーションライブラリからのメッセージを受信し、TCP, UDP, Flow マネージメントなど各トランスポートプロトコルのキューにバッファを入れ、それぞれのメッセージ処理を起動する。ライブラリにおけるソケット番号を元に生成される ID により、処理プロトコルやそのコントロールブロックへのポインタ、Mach ポート番号、排他制御用ロックなどを管理する。

3.3.3 UDP, TCP プロトコル処理部

ネットワークプロトコルと連動し、アプリケーションインタフェース部から起動される送信スレッドとネットワークインタフェース部から起動される受信スレッドで構成される。受信処理に関しては、宛先ポート番号からアプリケーションインタフェース部の ID を得て、アプリケーションへの返答を依頼する。

3.3.4 IPv6 プロトコル処理部

上位プロトコルに引き続き出力インタフェースを得て、NDP プロトコル処理スレッドを起動する。またネットワークインタフェーススレッドから起動されると、上位プロトコルおよび ICMP コントロール処理に入る。

3.3.5 NDP プロトコル処理部

IPv4 の ARP(Address Resolution Protocol) に対応する IPv6 のネットワークアドレス検出機能としての NDP(Neighbor Discovery Protocol) プロトコルの処理を行う。IPv6 のパケットにネットワークアドレスを含むデータリンク用ヘッダを付加し、次ホップノードアドレス毎に到達可能性の状態やその時刻を次ノードキャッシュとして保持する。キャッシュ情報の欠如、到達不可能、到達可能時刻後の時間超過の場合には ICMP メッセージを送信する。到達可能性確認後、送信データをネットワークインタフェース部へのキューに入れる。

3.3.6 フロー制御部

資源予約プロトコルにより、特定フローのリアルタイム転送処理が要求されると、指定された宛先アドレスやプロトコル番号から、転送パケットヘッダを生成する。さらにフロー毎にネットワークインタフェース部を通してリアルタイムキューを作成する。

3.3.7 ネットワークインタフェース部

ネットワークサーバ内におけるフローの優先度制御のため、ネットワークドライバ内にフロー毎に複数のキューを作り、ネットワークインタフェース部ではキューIDを指定してデバイスへの書き込みを要求する。受信側では、フローID毎に異なる複数のネットワークパケットフィルタを使用し、各処理スレッドがフィルタからのイベントによりパケット受信を行う。

4 評価

実験環境としては、PC 互換機 A (32 ビット i486DX2, 66MHz, メモリ 16MB) と B (32 ビット Pentium, 95MHz, メモリ 16MB) を 3Com Elite16 Ethernet Interface を使用し、10 Mb/s Ethernet で接続した。PC 互換機では、RT-Mach RMK95 上の IPv6 サーバ内 UDP/IP 実装、IPv6 サーバとライブラリを併用したリアルタイムスレッドでの実装と Lites.1.1.u2 サーバ内実装 (IPv4) のパフォーマンスを比較した。サーバ・ライブラリ併用実装では、Ethernet driver にリアルタイムキューの実装も追加されている。512, 1024, 1460 バイトそれぞれのメッセージサイズのパケットを 100 msec 毎に送信し、受信ホストの受信アプリケーションでの受信カウントを測定した。測定中、IPv6 サーバを用いた実装では、サーバを経由するファイル転送プロトコル (ftp6)、Lites サーバではサーバ内の ftp4 を同時に走行させ、サーバに疑似的に輻輳を発生させた。ftp を走らせる前の遅延時間の平均値からの各パケットの受信カウントの差をジッタとし、ftp 走行中のジッタ平均値を測定した結果を表 5に示す。測定には、プロセッサイベントを測定する Intel Pentium プロセッサのハードウェアカウンタを利用した性能測定用プログラムを使用した。

表 5: ftp 走行中の受信カウンタのジッタ平均値

	メッセージサイズ (バイト)		
RT-Mach RMK95	512	1024	1460
IPv6 サーバ内実装	544322	573489	674024
IPv6 サーバ＋ライブラリ実装	2444	2697	3638
Lites1.1.u2 サーバ (IPv4)	186704	214978	217555

実験結果から、IPv6 サーバ＋ライブラリ実装は、他の実装よりかなりジッタが小さく抑えられている。これに対し、IPv6 サーバ内実装はデータが複数スレッドを渡り歩くことによるオーバヘッドのため、Lites サーバ実装よりもジッタが大きくなってしまっており、今後の課題となる。

5 まとめ

次世代インターネットプロトコル IPv6 では、24 ビットのフローラベルや優先度フィールドなど、実時間通信処理サービスへの要求に応じた QOS サポートが考慮されており、今後の課題として (1) フローラベル毎のフロー制御、(2) ATM の QOS コネクションとのマッピング、(3) ルーティングプロトコルや NDP (Neighbor Discovery Protocol) とのキャッシュ情報の一貫性管理、(4) キューでの優先度制御、(5) パスと資源予約制御が挙げられる。さらに、慶應大学を中心とした次世代マイクロカーネル (MKng) 研究プロジェクトにおける、RT- Mach (Realtime Mach) OS 上でのリアルタイムスレッドを用いて、次世代インターネットプロトコルサーバを実装した結果、通常のファイル転送プロトコル (v6) による輻輳時のジッタ評価は、Lites サーバ内 IPv4 実装と比較して優位であった。

6 今後の課題

CMU および 慶應大 MKng プロジェクトで開発中の、リアルタイム制御機能を持つ X サーバ、ビデオ画像表示アプリケーションの Qt-play、ビデオ会議用アプリケーション nv へ適用し、ファイル転送プロトコルと同時走行して QOS 評価を行う。

謝辞

本研究を行うにあたりご協力頂いた次世代マイクロカーネル (MKng) 研究プロジェクトの皆様に感謝いたします。さらに、ご指導頂いている慶応義塾大学環境情報学部の斎藤信男学部長、荻野達也助教授に感謝いたします。また、いつも温かい励ましのお言葉を頂いている NTT 情報通信研究所第5プロジェクトの皆様に感謝いたします。

参考文献

[1] S. Deering and R. Hinden. Internet Protocol, Version 6 (IPv6) Specification, 1995. RFC1883.

[2] D. Funato and H. Tokuda. A Rate Based ATM Communication Mechanism in Real-Time Mach. In *IEEE Youth Forum in Computer Science and Engineering*, October 1995.

[3] C. Maeda and B.N. Bershad. Protocol Service Decomposition for High-Performance Networking. *SOSP*, 1992.

[4] P. Newman, W.L. Edwards, R. Hinden, E. Hoffman, F. Ching Liaw, T. Lyon, and G Minshall. Ipsilon Flow Management Protocol Specification for IPv4 Version 1.0, 1996. RFC1953.

[5] P. Newman, T. Lyon, and G. Minshall. Flow Labelled IP: A Connectionless Approach to ATM. In *Proceedings of IEEE Infocom*, 1996.

[6] Y. Onoe and H. Tokuda. Multicast Protocol for Continuous media. In *Abstracts of the 7th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1996.

[7] C. Partridge. A Proposed Flow Specification, 1992. RFC1363.

[8] H. Schulzrinne, S. Casner, R. Frederick, and V. Jacobson. RTP: A Transport Protocol for Real-Time Applications, 1996. RFC1889.

[9] H. Tokuda, T. Nakajima, and P. Rao. Real-Time Mach: Towards a Predictable Real-Time System. In *Proceedings of Mach Workshop, USENIX Association*, October 1990.

[10] H. Tokuda, Y. Tobe, S.T.-C. Chou, and J.M.F. Moura. Continuous Media Communication with Dynamic QOS Control Using ARTS with an FDDI Network. In *Proceedings of SIGCOMM92, ACM Press*, 1992.

[11] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala. RSVP: A New Resource ReSerVation Protocol. *IEEE Network*, 1993.

[12] 松井康範, 内海秀介, 松渡大地, 中村嘉志, 成田多良, 細川達巳, and 徳田英幸. ATM-LAN 環境と ATM-WAN 環境の転送特性の比較. 情処研報 *95-OS-71, 95-DPS-73, P.99-104*, November 1995.

[13] 徳田英幸, 迫川修一, 西尾信彦, 荻野達也, and 斎藤信男. MKng: 次世代マイクロカーネル研究プロジェクト. 第 *53* 回情処全国大会論文集, *5B-4*, July 1996.

[14] 木原誠司, 藤合敏, and 南部明. ST-II プロトコルサーバの Real-Time Mach への実装と評価. 情処研報 *94-OS-65, P.81-88*, July 1994.