ネットワークキューにおける帯域制御方式*†

尾上裕子‡
NTT 情報通信網研究所

藤井敬三§
(株) システムコア

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

概要: 本研究では、マルチメディア処理を行うリアルタイム通信プロトコルの資源予約機構の一機能として、ネットワークキューにおける帯域予約制御について検討する。本方式は、優先度ベースキュー制御方式とレートベースキュー制御方式の混合方式により、予約したネットワーク上での VBR(Variable Bit Rate) 転送を可能とするとともに、マルチメディア処理技術に不可欠な QOS(Quality Of Service) の制御を提供する。さらに、1.5M の ISDN 上でネットワーク上に負荷を発生させ、本方式の帯域予約制御機能を取り入れた場合と通常の場合とで、UDP 通信のジッタ・スループットを測定した比較実験結果と、既存のリアルタイム通信プロトコルへの適用方法について述べる。

1　はじめに

近年、動画や音声などのマルチメディア情報を、広域分散環境で統合的に処理する技術への要求が高まっている。このマルチメディア情報を扱うリアルタイムシステムには、メディアのタイムクリティカルな性質のため、高速性・広帯域性に加えてタイミングの正当性が要求される。そこでリアルタイム OS では、処理サーバの周期やデッドラインの指定、サービスの優先順位制御やスケジューリングポリシによる時間制約の保証と処理の決定性、予測可能性といった項目を満足しなくてはならない [11][15]。さらに大規模広域分散環境では、ユーザレベルアプリケーションが要求する処理性能の特性を、通信ホスト間のエンドツーエンドで満足しなくてはならない。そこでリアルタイム通信サービスでは、メディア情報の特性と転送量に応じた帯域制御や優先度制御を、ネットワークワイドに提供する必要がある [5][8]。

2　リアルタイム通信における帯域制御

2.1　リアルタイム転送サービス

マルチメディア処理に対してリアルタイム転送サービスを提供するには、アプリケーションレベルが求めるサービス品質 (QOS) を、リアルタイム通信レベルで FlowSpec をはじめとしたネットワーク転送サービス性能に変換しなくてはならない [9][12]。このようなメディアフローに対して提供される転送性能保証サービスは、以下の 3 つに大別できる。

1. 絶対的な転送サービス保証型
ネットワークを共有する他の通信の転送量とは無関係に、あらかじめ設定された帯域量が使用可能であることを保証する。また、エンドツーエンドの遅延時間をある一定値内におさめる。

2. 予測可能な転送サービス保証型
エンドツーエンドの遅延時間の上限をあらかじめ予測することが可能となる。このとき、遅延時間の平均値を保証、又は全データのうち何％までがある遅延時間内に転送されるかを保証する。

3. 不完全な転送サービス保証 (ベストエフォート型)
転送制御無しで、あらゆるフローの転送パケットがサイズや優先度に無関係に、キューに入った順にサービスされる。

このうち、マルチメディア情報を扱うための通信プロトコルにおいて、IP プロトコルをはじめとするベストエフォート型の転送サービスモデルでは、パケット到着遅延時間のばらつきやネットワークオーバフローによりパケット損失が生じ、音声がとぎれたり映像が止まるといった問題が挙げられる。さらに、これが CPU の一時的な負荷やネットワーク輻輳を生じさせることになる。そこで XTP, ST-II, HeiTP などのリアルタイム通信プロトコルでは、あらかじめ必要な帯域幅・バッファ領域をあらかじめ予約しておく資源予約機能や、転送状況を常にモニタリングし、状況に応じてデータの転送量を動的に変更する転送レート制御機能などの帯域制御のためのフレームワークを提供している [3][10][13]。本研究では特に、これら上位レベルの帯域制御機構をブレークダウンした一機能として、ネットワークキューにおけるネットワーク帯域使用量の制御について検討する。

*この研究は、情報処理振興事業協会 (IPA) が実施している開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトのもとに行われました。

†A Bandwidth Control Method in Network Queues

‡Yuko Onoe, NTT Network Information Systems Laboratories
§Keizo Fujii, System Core Co. Ltd.
¶Hideyuki Tokuda, Institute of Environmental Information, Keio University/School of Computer Science, Carnegie-Mellon University

2.2 ネットワークキューにおける帯域制御

現在のネットワークキュー制御方式では、バッファをリンク式に接続したキューをインタフェース毎に保持する。上位レイヤからの転送要求があると、データリンクレイヤで転送パケットはキューの最後尾につながれ、転送中でなければ転送モードに入り、転送中ならそのままとなる。このとき、データリンクレイヤでは FIFO 式でキューからバッファを取り出し、転送サービスを行う。また、キュー内のパケット数があらかじめ指定した値を越えると、それ以後到着したパケットは廃棄される。

この FIFO 式では転送パケットの生存時間を考慮せずに転送制御されるため、古いパケットのキュー内遅延が新しく到着したパケットの転送を脅かすことになる。また、転送パケットには優先度付けがなされず、均等にキューを共有する。このためサイズの大きなパケットほど転送スケジュールの占有時間も長い。これに対し、以下のようなキュー制御方式が提案されている。

1.WFQ(Weighted Fair Queueing)[1]
トランスポートプロトコルのふるまいに対して公平性を保つために提案された、全てのホストに均等に帯域幅を割り当てる bit-by-bit round-robin service 方式に対して、帯域幅をキュー数より多い数 (m) の帯域スロットに分割し、このスロットを各ホストやフローに適当に割り当てることにより、帯域割当量の重み付けを提供する。

2. FIFO+[2]
ホップ毎に加算される遅延時間を制限することにより、最悪遅延時間を保証しようとする。フローをクラス分けし、各ホップでのキュー内遅延をクラス毎に計算する。そして、各パケット毎にクラスの平均値との差分を算出し、到着時間にこの差分をオフセット値として加算し、パケット転送のスケジューリングを行う。WFQ に比べて最悪遅延時間を小さくできるが、統計的なスケジューリング方法であり他のフローの転送に影響を受け、絶対的な保証とは言えない。

上述のキュー制御方式のうち WFQ では、アドミッションコントロールなどの別のルーチンによって、セッションの優先度に応じて帯域幅を適当に配分し、帯域スロットを各メディアフローに割り当てる。これに対し FIFO+ では、転送サービススケジューリングにおいて優先度は加味されるものの、優先度が等しい転送パケットで遅延時間の平均化を行っている。このため、互いに他のセッションの影響を受け、遅延時間や転送レートを絶対的に保証することはできない。そこで本研究では、WFQ を用いてセッション毎にスロット割当量の異なる絶対的なサービス保証を目指したうえで、マルチメディア処理に求められる要求条件を満足するキュー制御方式について検討する。

2.3 VBR の扱い

キュー制御方式として WFQ をそのまま適用した場合には、割り当てられたキューのレート値に等しい Constant Bit Rate(CBR) チェネルを転送することになる。一方、メディアデータは MPEG 圧縮技術により VBR データに圧縮されるため、転送データのレート値を一定にするなど、メディア情報のソース側に帯域使用量に制限を与えることになる。そこで本研究では、保証対象パラメータとして最低帯域幅のみを予約し、メディア情報のレートが可変的な VBR を想定して、予約時に申告した帯域量以上にデータを転送することは可能とする。転送サービス側では、ネットワークが空いているときにはこの VBR をネットワーク帯域限界値まで転送できる。しかし、ネットワーク混雑時には各フローに対して最低帯域幅までは保証するが、それ以上に関してはキューにおいてフロー毎に流量を切り詰め、転送量を制御する。

2.4 マルチメディア処理のためのキュー制御ポリシー

より効果的なマルチメディア処理を提供するためには、さらに以下に示すような、メディア情報が持つ特性を考慮した制御ポリシーの適用が望まれる。

1. 情報の優先度
マルチメディア処理においては、遅延時間やジッタ・パケットエラー率などの各 QOS パラメータに対し、音声データの方が画像データよりも影響を受けやすく、サービス品質に大きく影響する。このため、音声と動画を同時に転送する場合には、動画セッションに対してスケーリングダウンの処理を適用することで、音声の質を維持する。

2. 情報の新鮮度
マルチメディア情報を扱う場合には、情報の一貫性・正確性よりも、情報が持つ連続的な性質から情報の新鮮度が重視され、古い情報はすぐに無効となる。さらにこの古い情報によって新しい情報が遅らされたり、廃棄されることを避ける必要がある。

2.5 上位レイヤの通信形態の混在

分散環境において、ユーザはリアルタイム処理とノンリアルタイム処理を頻繁に切り替えたり、同時処理を行ったりしている。このため、分散マルチメディアシステムのような時間制約を持つ通信は、ftp といったファイル転送の通信や telnet といったインタラクティブな通信と、ネットワーク帯域を共有している。このようなリアルタイム通信とノンリアルタイム通信とが混在した分散環境では、リアルタイム通信が予約量以上に帯域を使用するにあたり、初期申告量以上の通信がノンリアルタイム通信に影響を与えないことが望ま

れる。そこで、ノンリアルタイム送信期間中にノンリアルタイム通信が無い時に限り、優先度に従ってリアルタイム通信を行うこととする。

さらにノンリアルタイムな通信のうち、telnet をはじめとするインタラクティブな通信、またはルーティング制御情報などの制御用の通信などの緊急を要する通信は、最高優先度で送られる必要がある。このため、上位レベルの通信をネットワークレベルではリアルタイム通信、制御・インタラクティブ通信、ノンリアルタイム通信の3 タイプに分け、別々に扱う必要がある。ここでは、このうちリアルタイム通信とノンリアルタイム通信について扱うこととする。

3 ネットワークキューにおける帯域予約制御

これまで提案されているネットワークキュー制御方式は、以下の 2 タイプに大別される。

- 優先度毎にキューを設け、同一優先度のセッションはキューを共有するという、優先度ベースキュー制御方式
- セッション毎の最大遅延時間などからレート要求値を決定し、別ルーチンのアドミッションコントロールによりレート値を割り当てられるレートベースキュー制御方式

優先度ベースキュー制御方式では、高優先度アプリケーション要求の競合が起こり、潜在的な高負荷を検知および回避するのが困難となる。一方マルチメディアシステムでは、セッション毎に異なるサービス品質が要求されるため、FlowSpec などの転送サービス性能の各パラメータ値も、セッション ID をキーとしてセッション毎に指定される。そこで本研究では、ネットワークキューはアプリケーションにより要求される FlowSpec 毎、つまりセッション毎とし、各セッションが指定する FlowSpec に応じたキュー制御方式を決定する。さらに本研究のキュー制御方式として、優先度ベースとレートベースの両者の混合方式により、キューを制御する。つまり、マルチメディア情報が持つ連続的な性質により、基本的にはレートベースキュー制御方式に従ってタイミングの正当性を図り、スループットの最低限度値を保証する。さらに、空き領域に関しては、ノンリアルタイムキューを最高優先度とし、セッションの優先度に応じた優先度キュー制御方式を適用する。

3.1 帯域予約制御方式

ここでの帯域予約の仕組みは、以下の項目をベースとし、RSVP[14] をはじめとする上位レベルの資源予約機構の一機能として働く。

1. メディアフローが帯域幅要求を申告する枠組みを提供
2. 新しいフローからの要求を許可するかどうかを判断
3. 許可された帯域幅の定期的なフローへの割り当て
4. 帯域幅の空き領域に関してはフローの優先度に応じて帯域を提供
5. オーバーロード状態時に予約量まで帯域使用量を制限するにあたり、フローが指定するキュー制御ポリシーを適用

3.2 キューの定義

まず、キューは以下に示す属性により定義される。

・ **sessionID(host_addr, port_num)**
リアルタイム通信において、メディアフローを識別するためにフローに一意の sessionID を指定し、この sessionID 毎にリアルタイムキューを生成する。sessionID は、ホストアドレスとポート番号の組で構成される。

・ **queue_length(max, limit)**
キューの長さは limit と max で示され、limit を越えた場合に queue handling policy に従ってキュー内に溜まっているバッファは廃棄される。このとき設定されるキューの長さと転送パケット廃棄率は反比例し、キューを長くするほどパケット廃棄は避けられるが、その分キュー内遅延時間が大きくなる。さらに、制御ポリシー適用のタイミングを規定する limit では、長期にわたるリアルタイム通信では短く設定し、後続のパケット転送にひびかないようにする。これに対し短期間の通信の場合には、少しくらい許容量を越えても、短期のトラヒックバーストがスループット低下を招かない程度に長くとることが望まれる。

・ **queue_type(RT/non RT)**
リアルタイムキューの場合はさらに queue_sync の属性を保持し、あらかじめアドミッションコントロールなどにより、周期的な帯域使用の保証がなされる。これに対し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)**
スケジュールされるための周期 (cycle) と期間 (term) を指定する。cycle には転送開始の周期を、term には転送期間を指定する。

・ **queue_priority**
セッション自体の優先度はリアルタイムキュー生成時にアドミッションコントロールにより使用されるが、こちらは互いに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な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た

めに使用する。

・queue_handling policy(TIMEOUT/QOS_LEVEL DROP/QOS_LEVEL SHUTOUT)

キューの長さが limit 値を越えた場合に、queue handling policy に従ってキューの切り詰めを行っていく。例えば、TIMEOUT が指定された場合には、一定時間以上にキュー内にいた古いバッファの廃棄する。また、QOS_LEVEL が指定された場合には、パケットの QOS フィールドに従い、時間的解像度・空間的解像度などの QOS 値の低いものの間引きをする。

3.3 キュー制御方式

ここでは、ネットワークキューにおける帯域予約制御方式として、アプリケーションレベルの要求に応じたネットワーク資源の割当方法と、割り当てられた資源上での転送制御方式の二点に分けて述べる。

3.3.1 ネットワーク資源の割当て

データリンクレイヤにおいては、以下の手順に従ってネットワーク資源の割り当てを行う。

1. メディアフローは、sessionID、帯域使用量や制御ポリシーを指定することにより、ネットワークレベルの帯域予約を行う。
2. 帯域要求量に対し、総予約量が帯域幅を越えない範囲であれば新しい要求を許可し、なければ拒否する。
3. sessionID 毎に指定された要求量に応じた構造を持つキューを生成する。

アドミッションコントロールとして、ここでは参考文献 [6] で示される

$$1 \geq \sum_{i=1}^{s} \frac{Term_i}{Cycle_i}$$

を用いるが、コンテキストスイッチやインタラプト処理などのオーバーヘッドを測定するような、より正確なメカニズムが必要である。

さらにここでは、$Cycle_i$ はキュースケジューリングのサイクル (固定値) とする。例えば、Cycle = 100 msec のうち $Term_i$ に指定された期間が、そのセッションへの割当量となる。

3.3.2 転送制御方式

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を転送する。

A. 転送手順

1 資源予約フラグを確認する。

2 フラグが立っていたら、送信しようとする転送パケットのデスティネーションアドレスとポートが、予約されているものと一致しているかどうかを確認する。

2.a 予約されているものならリアルタイム用のキューの長さを確認する。

2.a' キューの長さが limit で指定されている値よりも短ければ、そのままリアルタイムキューに入れる。長ければ、キュー制御ポリシーに従ってキューの長さを規定以内に収まるようにキュー内バッファの間引きを行う、または転送パケットの QOS フィールドを調べ、QOS 値が指定値以下ならパケットを廃棄する。

2.b 予約されているものと異なれば、ノンリアルタイムキューに入れる。

3 転送中でなければキューからパケットを取り出して転送し、転送中であればそのままリターン。

B. キュースケジューリング手順

次に、キュー制御の基本動作を示す。

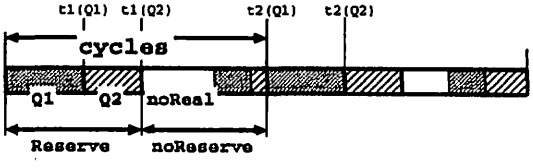

図 1: キュー制御

1 リアルタイムキュー送信期間であることを確認し、次にリアルタイムキューから送信する時間 t2 を計算する。

2 リアルタイム送信終了時刻 t1 まで、リアルタイムキューから転送バッファを取り出し送信。

3 ノンリアルタイム送信期間中は、ノンリアルタイムキューから転送バッファを取り出し、送信する。このとき、送ろうとするパケットのサイズを計算し、ノンリアルタイム転送期間中に転送が終了するかどうかを計算する。

4 ノンリアルタイム転送期間が終了し、リアルタイム転送が開始される (t2) までに送信が終了する場合は送信し、そうでなければキューに戻す。

5 t2 になったらリアルタイムキューからの送信を開始する。

6 どちらの場合も、キューが空なら相手のキューから送信する。

6において、リアルタイムキューが複数存在する場合には、キューの優先度に従って転送バッファを決定する。この制御方式により、キューに入ってくるパケット量が予測できない VBR を扱うリアルタイムチャネルにおいて、キュー送出レートの予約値よりキュー入力レートが大きくても、他のキューが空いていれば予約量以上の転送レートを確保できる。ただし、ノンリアルタイム転送期間中にノンリアルタイムな転送パケットがある場合にはそちらが優先されるので、リアルタイム側はネットワーク混雑時には予約量まで帯域使用が制限され、それ以上はキューに溜まることになる。このときキューがあらかじめ決定した長さ (limit) 以上になると、キューの制御ポリシーに従ってキュー内バッファの切り詰めが行われるため、パケットの優先度を無視した無差別な転送パケットの廃棄が回避される。

3.4 キュー制御ポリシー

ネットワーク輻輳時には、帯域使用量を予約量まで抑えなくてはならない。キューの長さが limit を越えている状態が一定期間続くことにより、輻輳発生を知ることが出来る。このとき、以下の手段によりネットワークキューにおいて転送パケットの送出レートの抑制を行う。

・**QOS_LEVEL**
転送パケットの QOS フィールドに指定される値をここでは QOS レベルと定義する。この QOS レベルに従って転送制御が行われる。

1. SHUTOUT
   QOS レベルの低い転送パケットは、キューに入れずに廃棄される。

2. DROP
   QOS レベルの高い転送パケットをキューに入れるときには、QOS レベルの低いパケットをキューから取り出し、キューの長さが規定量になるようにする。

・**TIMEOUT**
キュー内生存時間の長い転送パケットをキューから取り出し、キューの長さが規定量になるようにする。

4 実験

4.1 実験方法

帯域予約制御の効果を図るために、本研究で提案する制御方式を 1.5M の ISDN のデバイスドライバ (ICCB-H1) に組み込み、適用実験を行った。実験方法としては、1.5M ISDN につながる 2 台の Sun WS (SS2, SunOS 4.1.3) 間で、一定時間間隔で UDP パケットを送信した。この実験では、上位レベルのアプリケーションにより、3 Kbytes のパケットを 100 msec 間隔で 2 パケットずつ送信する。このパケットにはパケットシーケンス番号のフィールドを設け、受信側ではパケットを受信後、受信時刻とシーケンス番号から得られる受信予定時刻との差分を測定した。この受信予定時刻は、パケット送信時間と輻輳発生前の遅延時間の平均値との和とした。

この UDP の通信と同時に、ネットワーク負荷として TCP のコネクションを張った。この TCP では、ファイル転送プロトコル (ftp) のセッションを想定し、ISDN 1.5M の最大 MTU である 4Kbytes のパケットを 1000 個待ち無しで送り続けた。このとき、デフォルトウィンドウサイズは 24Kbytes に設定している。

帯域予約に関しては、ISDN 1.5M の最大スループット値 100 Kbytes のうち、UDP 通信の使用帯域幅は (3000 + header 長 (4[シーケンス番号] + 20[IP ヘッダ長] + 14[Ether ヘッダ長])) * 2 * 10 sec = 61 Kbytes であるが、CPU の内部タイマの粒度が 10 msec であるために、サイクルを 100 msec のうち 70 msec をリアルタイムキュー転送時間と設定した。そのため、スループット値 100 Kbytes のうちの 70 Kbytes が帯域予約量となる。

4.2 実験結果

はじめに、100 msec 中 70 msec の帯域予約を行った場合と行わなかった場合とで、UDP の片道伝送遅延時間の変動 (ジッタ) を測定した (図 2)。その結果、予約を行なった場合には遅延時間の微妙な変動はあるものの、ほとんどがサイクルの 100 msec 以内におさまっていると見ることが出来る。これに対し予約を行わなかった場合には、遅延時間は 650 msec から 1 sec の間で大きく変動している。両者共、パケットロスは無かった。さらにこれと同時に、TCP コネクション側の片道伝送遅延時間の変動を測定した。その結果、予約を行わなかった場合の UDP 側の変動と、TCP コネクション側の遅延時間の変動の波が一致して表れることが示された (図 3)。つまり、予約を行わない場合 UDP の遅延時間の変動は、同時にセッションを張った TCP コネクションのウィンドウサイズの動的変化による輻輳制御に依存して発生している。

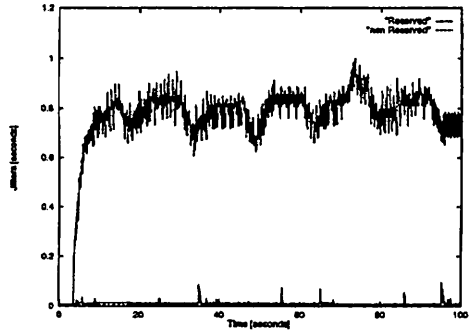

図 2: 0.07/0.1 sec の帯域予約による UDP のジッタ

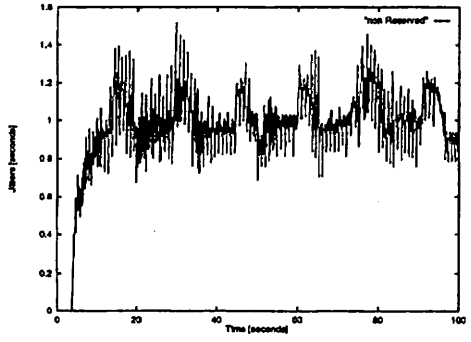

図 3: TCP コネクションのジッタ

この実験の TCP コネクションは例えば ftp の様なファイル転送プロトコルを用いた通信、UDP コネクションは MBONE(Multicast Backbone) のような通信にそれぞれ置き換えることが出来る。帯域予約を行わない場合には、これらがネットワークキューを取り合って、ジッタ・キュー内遅延およびパケットロスの要因となる。これに対し、リアルタイム通信に対して適切な容量で帯域予約を行った場合には、ジッタを小さく抑えることが出来る。さらにこのとき、UDP と TCP のセッションのスループットを測定した (図 4)。このうち予約を行った場合には、UDP のセッションはほぼ 60 KBytessec で安定しているが、TCP セッションは大きく変動している。これに対し、予約を行わなかった場合には、TCP セッションで同じサイズのファイルを転送しているにもかかわらず、スループット値が増加し転送時間は小さくなっている。一方、UDP 側のスループットは頻繁に変動している。このように、予約機構を用いないとネットワーク輻輳時にメディアフローの遅延時間・ジッタが増加し、スループットは変動する。

同様の実験を、スウェーデン国立コンピュータ科学研究所から PDS として提供されるコネクション型のプロトコル ST-II の実装を用いて行った。ST-II で

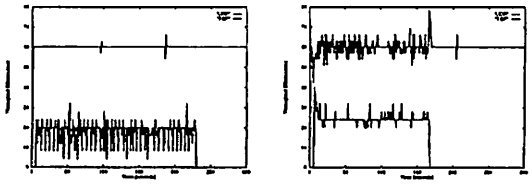

図 4: 帯域予約を行った場合と行わなかった場合の UDP および TCP のスループット値

は、IP パケットフォーマットのバージョン番号 = 5 を使用することで、IP と区別される。この実装では、STREAM 型のソケットと同様のアプリケーションインタフェースを持ち、名前付きソケットによりコネクションを確立し、そのソケットを通じてパケットを転送する。この ST-II の通信と同時に TCP のコネクションを張り、帯域予約を行った場合と行わなかった場合とで、ジッタおよびスループットを測定した。その結果、ST-II の場合も UDP/IP と同様の実験結果が得られた。

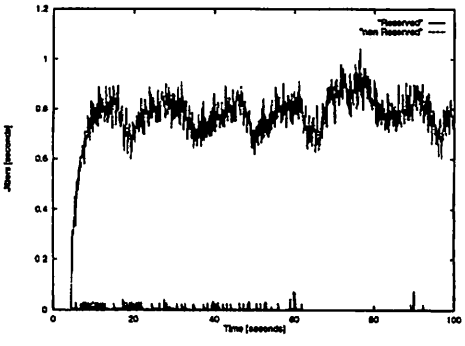

図 5: 0.07/0.1 sec の帯域予約による ST-II のジッタ

5 リアルタイム通信プロトコルにおけるキュー制御方式

次に、本方式が既存の通信プロトコルの資源制御機構に対し、どのように機能するかについて述べる。

5.1 適用効果

各セッションの QOS を決定する場合に、HeiTP などの既存のリアルタイム通信プロトコルでは、エンドノードで提供されるモニタリング機能により、エンドツーエンドでのネットワークトラヒックの状態を把握して判断する。ここで、ネットワーク媒体の固定帯域幅などの潜在的要因による QOS の決定は変化が少ない

と考えられるため、セッション確立時にエンドツーエンド型で決定することができる(end-to-end QOS)。これに対し、ネットワークや中継となるゲートウェイなどの輻輳時の対処となるQOSレベル低下は、ポイントツーポイント型により輻輳発生個所で処理を行うほうが、輻輳に対して迅速に対処することが可能となる(point-to-point QOS)。さらに、輻輳時のキュー内のバッファ占有率・エンドツーエンドのスループット・ネットワーク遅延・ボトルネックポイントの帯域使用率などの項目において、後者の方式がより有効であることが示されている[7]。

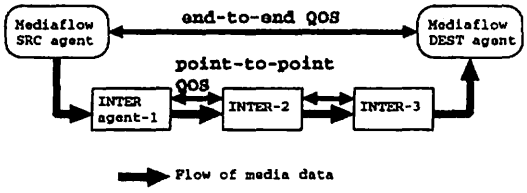

図 6: end-to-end QOS と point-to-point QOS

さらに、エンドツーエンド型でQOSレベルの低下を判断する場合、中継となるゲートウェイやネットワークパスの一部を共有する複数のセッションが、同時に輻輳を検知し並行してQOSレベル低下を指示するために、必要以上に転送量を低下させかえって帯域を余らせることになる。そのため、中継エージェントでのセッションの優先度を加味したポイントツーポイント型のQOS制御が望まれる。

以上の二点により、一次的な輻輳に対処するためのQOSレベル低下の判断は、輻輳発生個所においてなされることが望ましいと考える。本研究の制御方式では、輻輳個所となるネットワークキューにおいて輻輳検知および対処がなされる。そこで、本方式をリアルタイム通信プロトコルに適用することにより、効果的なポイントツーポイント型のQOS制御が行われ、エンドツーエンド型で決定されたQOSレベルに対し、輻輳時にはポイントツーポイント型でQOSレベルの低下を判断し、前者のQOSレベルに対して短期的に上塗りを行う。

5.2 資源制御機構への適用例

次に、既存の資源制御機構への本方式の適用例を示す。

ここでの資源制御機構は、資源予約とメディアフィルタリング機能の組合せで構成される。メディアフィルタリング機能は、予約を行ったネットワーク上でVBRデータを転送する場合に有効な方式である。

・SCENE-1

まずはじめにアプリケーションから帯域予約量の要求を受けると、RSVP をはじめとする既存の帯域予約機構を用いて、Session ID 毎に資源予約が行われる。この例では、ネットワークキューにおいて IF-1 に対する帯域は Q3 レベルで、IF-2 は Q1 レベルで予約される。

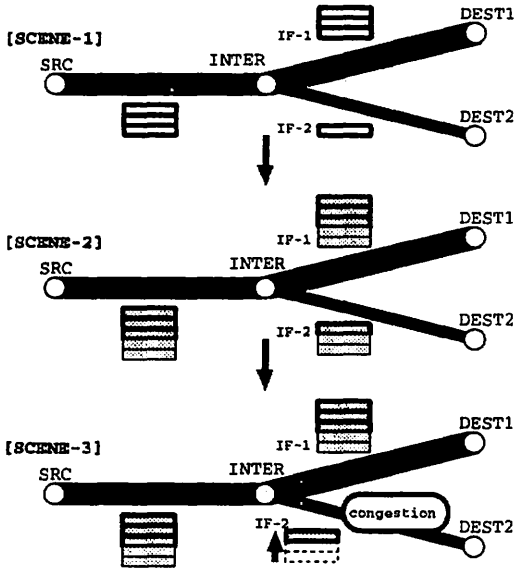

図 7: フィルタリング手続き

・SCENE-2

実際にメディア転送を行う上では VBR を想定し、帯域幅の総使用量が限界量に達するまでは、予約量以上の帯域を使用可能とする。このときフィルタリング機能を用いて、予約を行ったネットワーク上でのデータ転送制御が行われる[4][8]。このフィルタリング機能では、受信エンドノードでセッション毎のモニタリングを行い、フローと逆向きに中継エージェントにおいてインタフェース毎に以下のようなフィルタ(表1)を設定していく。この例では、IF-1 に対する QOS レベルは Q5、IF-2 に対しては Q3 が設定される。そのため実際のメディア転送時には、パケットの QOS フィールド値が Q3 以内のパケットは、IF-1 と IF-2 の両ネットワークにフォワードされるが、Q4 と Q5 のパケットは IF-1 へのみ転送される。ネットワークキューにおいては、IF-1 へは予約期間中に Q3 以下のパケットの転送量が確保され、それ以外の期間に Q4 と Q5 の分の

表 1: フィルタテーブルの例 1

Session ID	Net IF	QOS level	Filter level
SID-1	IF-1	Q5	DEST
	IF-2	Q3	DEST

転送がなされることになる。

・SCENE-3

ネットワーク輻輳発生にともない、リアルタイムキューにおいてキューが溜まりはじめると、QOS レベルを低下させるようにキュー制御を行う。この例では、IF-2 に関してエンドツーエンド型で QOS レベルが Q3 と決定されており、短期的なトラヒックバーストが生じると、ネットワークキューにおいてポイントツーポイント型で QOS レベルが Q1 に下げられる (表 2)。同一インタフェースでは、ポイントツーポイント型の判断がエンドツーエンド型に優先されるが、ポイントツーポイント型の判断は一定時間でタイムアウトする。

表 2: フィルタテーブルの例 2

Session ID	Net IF	QOS level	Filter level
SID-1	IF-1	Q5	DEST
	IF-2	Q3	DEST
	IF-2	Q1	INTER

6 今後の課題

今回は、リアルタイム通信とノンリアルタイム通信の 2 タイプの通信形態のみに対して検討を行ったが、今後は制御情報交換やインタラクティブ通信などの緊急度の高いパケットを、ネットワークキューで優先的に処理する仕組みについて検討する。

また、今回はリアルタイム通信プロトコルへの適用例について述べたが、今後は FlowSpec 中の転送サービス性能パラメータとのマッピングについて、詳しく検討していく必要がある。

7 まとめ

本研究では、リアルタイム通信プロトコルの資源予約機構の一機能として、ネットワークキューにおける帯域予約制御について検討した。本方式は、優先度ベースキュー制御方式とレートベースキュー制御方式の混合方式とすることにより、予約したネットワーク上での VBR(Variable Bit Rate) 転送を可能とするとともに、マルチメディア処理技術に不可欠な QOS の制御を提供する。さらに、1.5M の ISDN 上での帯域予約制御の実験結果と、リアルタイム通信プロトコルへの適用方法について述べた。

8 謝辞

本研究を行うにあたりご協力頂いた斎藤信男教授、荻野達也助教授、開放型基盤ソフトウェア研究開発評価事業「マルチメディア統合環境基盤ソフトウェア」プロジェクトの皆様に感謝いたします。また、NTT 情報通信網研究所の和佐野哲男部長、大町雄一グループリーダ、グループメンバの皆様に感謝いたします。

参考文献

[1] C.Partridge. Gigabit Network. *Addison Wesley*, 1993.

[2] D.D.Clark, S.Shenker, and L.Zhang. Supporting Real-Time Applications in an Integrated Services Packet Network: Architecture and Mechanism. In *Proceedings of SIGCOMM92, ACM Press*, 1992.

[3] D. Hehmann, R.G. Herrtwich, and R. Steinmetz. Creating HeiTS: Objectives of the Heidelberg High-Speed Transport System. Technical report, 1992.

[4] D. Hoffman, M. Speer, and G. Fernando. Network Support for Dynamically Scaled Multimedia Data Streams. In *Proceedings of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1993.

[5] S. Kihara, Y. Onoe, S. Moriai, A. Mituzawa, A Nambu, and H. Tokuda. An implementation of ST-II Protocol as a user-level server on RT-Mach. In *Abstracts of the 4th International Workshop on Network and Operating System Support for Digital Audio and Video*, 1993.

[6] C.W. Mercer, S. Savage, and H. Tokuda. Processor Capacity Reserves: An Abstraction for Managing Processor Usage. In *IEEE Multimedia94*, 1994.

[7] P.P. Mishra and H. Kanakia. A Hop by Hop Rate-based Congestion Control Scheme. In *Proceedings of SIGCOMM92, ACM Press*, 1992.

[8] Y. Onoe. QOS-based Multicast Communication. In *Abstracts of the 2nd Workshop on Advanced Teleservices and High-Speed Communication Architectures*, 1994.

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

[10] W.T. Strayer, B.J. Demsey, and A.C. Weaver. *XTP: The Xpress Transfer Protocol*. Addison Wesley, 1992.

[11] 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.

[12] 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.

[13] C. Topolcic, S. Casner, C. Lynn, P. Park, and K. Schroder. Experimental internet stream protocol, version 2(ST-II), 1988. RFC1190.

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

[15] 西尾信彦, 追川修一, 緒方正陽, 尾上裕子, 河内谷清久仁, 塩野崎敦, 南部明, 持田茂人, 和田英彦, and 徳田英幸. マイクロカーネルによる連続メディア処理の基盤技術. 第5回コンピュータシステムシンポジウム論文集, October 1993.