マルチメディア通信と分散処理 85－8
（1997. 11. 6）

# TCPとの共存を目指した Reliable Multicastの輻輳制御方式の提案

佐野 哲央† 山内 長承‡ 城下 輝治† 高橋 修†
†NTT 情報通信研究所
‡IBM 東京基礎研究所 ‡東京都立大学工学部

あらまし バルク Reliable Multicast (RM) に対するフロー／輻輳制御は, RM トラヒックが TCP のような既存トラヒックを押し潰さずネットワーク帯域を共有するために不可欠である. 本報告では1対多のバルク RM に対するフロー／輻輳制御方式として Monitor-Based Flow Control(MBFC) を提案し, MBFC において RM と TCP との共存を可能とする輻輳制御方式を示す. また提案する方式の有効性を示すため我々の提案するバルク RM プロトコル RMTP に MBFC を実装し, RMTP と ftp による同時バルクデータ転送を行い, 共有リンクの利用帯域を測定する評価実験を行い, RM と TCP の両者間で効果的に帯域の共有が達成できることを確認した.

# Congestion Control for Reliable Multicast to realize coexistence with TCP

*Tetsuo Sano*† *Nagatsugu Yamanouchi*‡ *Teruji Shiroshita*† *Osamu Takahashi*†
†NTT Infomation and Communication Systems Laboratories
‡IBM Research, Tokyo Research Laboratory ‡Tokyo Metropolitan University

**Abstract** This paper proposes Monitor-based Flow Control (MBFC) for flow/congestion control for one-to-many bulk reliable multicast (RM) protocols. A bulk RM on top of IP multicast requires flow/congestion control, because it needs to share the link bandwidth with other legacy traffic such as TCP so that RM does not push them away aggressively. MBFC is a generic mechanism to implement such flow/congestion control. Applying TCP-like congestion control in the rate control policy, it realizes coexistence with TCP traffic. MBFC was implemented and evaluated in our RM protocol[1][2]. The experiments showed that our rate control policy effectively achieved bandwidth sharing between bulk RM and bulk TCP transfer traffic.

## 1 はじめに

マルチキャストを用いて多数のユーザに一斉に誤りなくデータ配送する Reliable Multicast(以降 RM) の研究が盛んに行われている. その利用方法としてはバルクデータ転送, 短周期のトランザクション配送, 協調作業インフラなどが挙げられる. 特にバルクデータ転送はプッシュ型情報分配サービスの基盤技術として注目されている.

RM の実用化に際しては二つの大きな課題がある. 信頼性のあるデータ転送の実現とフロー／輻輳制御の実現である. 前者の実現に関しては再送機構や Foward Error Correction(FEC) についての考察が数多く行われている. フロー／輻輳制御は RM 実用化の上で最も重要な問題である. マルチキャストは潜在的にネットワークおよびそのユーザに対しての影響が大きいため, 適切なフロー／輻輳制御により受信者のバッファ溢れを防ぎ, かつ RM トラヒックが他の既存トラヒックとネットワーク帯域を共有できるように, RM のネットワークへのフローを調節する必要がある. しかしながら, この分野における研究はまだ十分に行われていない. 報告者等は [3] においてスループットの向上を目標に RM のフロー制御に関する議論した. そこで本報告では RM の輻輳制御方式に焦点を当て, RM トラヒックと現在のインターネット上のトラヒックの大部分を占める TCP トラヒックの共存を実現する新たなフロー／輻輳制御方式 Monitor-based Flow Control (MBFC) を提案する.

以下に本報告の構成を示す. 2節ではウィンドウ制御とレート制御について比較し, レート制御がバルク RM プロトコルにより適していること結論付ける. 3節ではバル

ク RM に対するフロー／輻輳制御方式として Monitor-based Flow Control (MBFC) を提案する．4節では TCP-compatible なバルク RM 用の輻輳制御を実現する方法について述べる．5節では本提案方式の評価結果を示す．6節に本報告のまとめを示す．

## 2 バルク RM に対するフロー／輻輳制御

フロー制御は通信のエンド-エンドで働き受信バッファを最大限に利用することを目的としている．一方輻輳制御は送信者がネットワークに過大なフローを流さないようにすることを目的としており，エンド-エンド [4] や，ゲートウェイ (ルータ)[5] 上で実現される．RM サービスをインターネット上で実用化することを考えた場合，フロー／輻輳制御は特に重要な問題であり，かつ困難な問題でもある．つまり送出フローをネットワーク上に広範囲に渡って存在する複数の受信者およびサブネットワークに合わせて調節しなければならないからである．本報告では既存環境において容易に実現できるエンド-エンドでの RM の輻輳制御の実現について考察する．以下，最も一般的なフロー制御方式としてウィンドウ制御とレート制御を取り上げ，バルク RM データ転送への適合性についてその比較検討を行う．

### 2.1 ウィンドウ制御 v.s. レート制御

RM 環境でウィンドウ制御を適用する場合，3 つの問題: 1) データ受信毎に多数受信者からの ACK が生成・集中し輻輳を起こす feedback implosion, 2) 複数の受信者が広告する異なるウィンドウに対し共通かつ適切に働く送信ウィンドウの定義, 3) 複数受信者に対する round trip time(RTT) 見積り，に対処する必要がある．1) の対処法として，backoff algorithm[6][7], aggregation[8][9] があるが，その導入より応答遅延が大きくなる．また 2) に対する有効な解は知られていない．3) の RTT 見積りは，ウィンドウ制御においてはパケット紛失を効率的に検出する上で重要である．短過ぎる見積りは不要な再送を起こし，長過ぎると紛失検出に時間を要しスループットが低下する．加えてマルチキャスト環境下では対象となる受信者が多数かつ動的に変化するので，その正確な見積りは困難となる．結果的にこれらの問題はウィンドウ制御のバルク RM プロトコルへの適用を困難にしている．

一方レート制御は送出パケット間隔を調整することでフロー制御を実現する．そのためバルク RM プロトコルへの適用も容易である．しかしレート制御では受信者またはネットワークの状態を把握し，送信レートを調節するための外部モニタ機構が別途必要となる．モニタ駆動者により送信者駆動型 (sender-initiated probe) と受信者駆動型 (receiver-initiated probe) に分類できる．

受信者駆動型においては受信者が自発的にモニタを行う．例えば NACK に基づく再送方式 [10] では，受信者がパケットロスを検出して NACK を返す．これを受けた送信者は，受信者またはネットワークが過負荷であると解釈し，送信レートを下げる．また NACK が一定時間来ない場合には送信レートを上げる．この方法の欠点は輻輳等により NACK が紛失した場合でも送信レートを上げることである．また上述とは異なる方法としてマルチキャストの join/leave によるルーティングトポロジを利用した redundant layered multicasting[11] が挙げられる．

一方送信者駆動型においては送信者が受信者にモニタを指示する．つまり送信者がモニタの頻度を制御する．RM 環境では送信者がモニタを頻繁に指示する場合，多数の受信者からモニタ結果が送り返されるため feedback implosion と同様の問題が起こり易い．またモニタ頻度が稀過ぎると，モニタ時の結果が現状を反映しなくなる．送信者は必要に応じてその頻度を調節する必要がある．

両者の併用により，前述のような NACK 紛失時の好ましくない挙動を回避できる．ここで報告者等は，1) 受信者側にほとんど手を加える必要がないこと，2) 実際に RM サービスを行う場合，参加メンバを定期的に監視する必要があること，から送信者駆動型での実現を方針とする．

## 3 Monitor-based Flow Control

本節ではバルク RM のフロー／輻輳制御方式として Monitor-based Flow Control (MBFC) を提案する．

### 3.1 MBFC の概要

MBFC はレート制御に基づいており，送信者駆動型モニタにより送信レートを調整する．モニタ部とフィードバックポリシ部の二部から構成され，モニタ部では送信者は定期的にデータパケットの受信数成功率をモニタし，フィードバックポリシ部ではモニタ結果を受けて事前に定義された関数に基づき送信レートを調整する．送信者はこの過程をデータ転送中に繰り返す．以下各部の動作を詳しく説明する．表記の簡単化のためデータパケットを DT で，DT のシーケンス番号を SN(DT) で表す．

### 3.2 MBFC の構成

[モニタ部]

モニタ部において送信者は受信者に対してモニタ開始及び終了を以下のような手順で指示する．

1. 送信者は DT 受信数のカウント開始を受信者に指示する．
2. 各受信者は上記の指示を受信後，DT 受信数のカウントを開始する．
3. 送信者は事前に定めた個数の DT を送出後，受信者

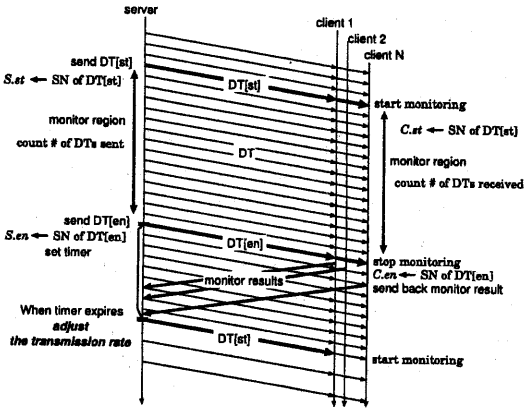


図 1: MBFC におけるモニタ部の動作概要

に対して DT カウント終了を指示し, 応答待ちタイマをセットする. ここで測定開始・終了の指示は, DT ヘッダ部の特定ビットを立てて通知する. 表記の簡単化のためモニタ開始を指示する DT を DT[st] で, モニタ終了を指示する DT を DT[en] で表す.

4. 受信者は DT[en] を受信するとモニタ結果を送信者に送信する. このモニタ結果には DT[st], DT[en] のシーケンス番号及びモニタ期間に受信した DT 総数が含まれる[1].
5. 送信者はモニタ結果を収集しフィードバック部に渡す. タイムアウトも一つの輻輳情報としてフィードバック部に利用される.

SN(DT[en]) − SN(DT[st]) + 1 とモニタ期間中に送出されたパケット総数とは必ずしも一致しないことに注意が必要である. よって送信者と受信者で送受信した DT 数および DT[st], SN(DT[en]) のシーケンス番号を記録する必要がある. 図 1 はモニタ部の一連の動作を示している.

[フィードバックポリシ部]

以下の 3 つのプロセスから構成される.

1. モニタ結果の要約

   このプロセスでは受信者からのモニタ結果を受信し, 要約し, それをマッピングプロセスに渡す.

   ここでは 2 つの閾値 $\mathrm{TH}_{up}$, $\mathrm{TH}_{down}$ および 4 つの変数 *up*, *down*, *resp*, *act_n* を使用する:

   $\mathrm{TH}_{up}$ : 受信結果により変数 "up" をカウントするかどうか決める閾値

   $\mathrm{TH}_{down}$ : 受信結果により変数 "down" をカウントするかどうか決める閾値

   *up* : DT の成功受信率が閾値 $\mathrm{TH}_{up}$ 以上である受信者の総数

   *down* : DT の成功受信率が閾値 $\mathrm{TH}_{down}$ 未満である受信者の総数

   *invalid* : 送信者が指示したモニタ区間と異なる値を返した受信者の総数

   *resp* : モニタ結果を返した受信者の総数

   *act_n* : 送信者が現在管理している受信者の総数

   (ここで $0 < \mathrm{TH}_{down} \leq \mathrm{TH}_{up} \leq 1$)

   各モニタ結果は閾値 $\mathrm{TH}_{up}$ and $\mathrm{TH}_{down}$ に基づき *up* の増加, *down* の増加, 送信レートの保持に関連付けられる. ここで $(act\_n - resp)$ はモニタ結果の得られなかった受信者数を表す. 例えば $((down + invalid + (act\_n - resp))/act\_n)$ が 1 に近い場合, ほとんど全ての受信者が過負荷状態であるか, ネットワークが輻輳している状態であると考えられ, 送信者は送信レートを下げる必要がある.

   タイマ満了前にモニタ結果を受信する毎に, 送信者は以下のコードを実行する.

```
resp++;  /* 応答のあった受信者数を数える */
/* モニタ区間が送信者と同一かどうかチェック */
if ((S.st==C.st) && (S.en==C.en)) {
    /* 受信成功率が閾値 TH_up 以上の受信者
       を数える */
    if (C.cnt/S.cnt ≥ TH_up) up++;
    /* 受信成功率が閾値 TH_down 未満の受信
       者数を数える */
    if (C.cnt/S.cnt < TH_down) down++;
},
```

   ここで *S.st* および *S.en* は送信者によって記録された DT[st] および DT[en] のシーケンス番号を, *C.st* および *C.en* は受信者が報告したモニタ結果に含まれる自身が受信した DT[st] および DT[en] のシーケンス番号を表す. *S.cnt* および *C.cnt* は送信者がモニタ期間中に送出した DT パケット数および受信者が受信した DT パケット数をそれぞれ表す. タイムアウト後, 変数 *up*, *down*, *invalid*, *resp*, *act_n* は次プロセスに渡される.

2. 要約結果を送信レートの上下, 保持にマッピング

   モニタ要約結果を受け, 送信レートの増加, 減少, あ

[1] さらに送信者が RTT を計測するために受信者が応答を送らせるのに用いられたバックオフ時間も含まれる

るいは保持のいずれか一つの結論を出力する．以下にマッピングを行うコードの一例を示す:

```
if ((up/act_n) ≥ 0.98) { /* increase */
        new_rate = f_inc(cur_rate);
} else
if (((down/act_n) ≥ 0.10) || ((invalid+(act_n-resp))/act_n ≥ 0.20))
{ /* decrease */
        new_rate = f_dec(cur_rate);
} else {
        /* remain unchanged */
}
(where TH_up and TH_down are set to 1)
```

最初の式は全受信者の98%以上が *up* 対象の場合，このプロセスは送信レートを ($f_{inc}$により) 上げる決定をすることを表している．次式では全受信者の10%以上が *down* 対象あるいは，収集したモニタ結果が全受信者数の80% $= (1-0.20) \times 100$ に満たない場合，このプロセスは送信レートを ($f_{inc}$により) 増やす決定をすることを表す．それ以外の場合は送信レートは保持される．送信レートの上げ下げの割合はそれぞれ $f_{inc}$および $f_{dec}$によって規定される．

3. 送信レートの上げ下げの割合の調節

   マッピングプロセスが送信レートを上 (下) げる決定をした際に，その加速 (減速) 度を決定する関数 $f_{inc}(f_{dec})$ が起動される．$f_{inc}$, $f_{dec}$は RM 以外のトラヒックとのネットワークでの帯域共有ポリシを反映するものとなる．例えばバルク RM プロトコル上のサービスが高い優先度を有する場合には，$f_{inc}$を積極的にに加速するように設計し，$f_{dec}$は消極的な設定にする．

次節では特にTCPトラヒックとの共存を目的とした $f_{inc}$, $f_{dec}$の設計方法について議論する．

## 4 バルク RM と TCP の共存

インターネットのような広域ネットワークにおいてその特性は時間とともに激しくに変化する．通信プロトコルはこうした変化に追従できる輻輳制御が必要となる．特にマルチキャストは潜在的に輻輳を誘発し易いため，バルク RM サービスを実現する上で既存トラヒックすなわち現在のインターネット上の大部分を占める TCP との“共存”を可能とするような輻輳制御の実現が鍵となる．ここで“共存”とは，RM(TCP) トラヒックが TCP(RM) トラヒックを押し潰さないことを指すものとする．その実現に当たっての我々の方針は，TCP の輻輳制御とコンパチブルな MBFC の輻輳制御を設計することである．以下ではその方法について詳しく述べる．

### 4.1 TCP における輻輳制御と回避

TCP における輻輳制御および回避機構は“slow start”，“additive increase/multiplicative decrease”，“fast retransmit and fast recovery” の 3 つの独立したアルゴリズムから構成される [4][12][13][14]．ここでは各アルゴリズムの説明は省略する．Additive increase/multiplicative decrease および slow start は独立したアルゴリズムだが実装時には一つのコードで実現される．結合されたアルゴリズムでは輻輳ウィンドウ (*cwnd*) およびスロースタート閾値 (*ssthresh*) の二つの変数が使用される．

### 4.2 TCP コンパチブルな輻輳制御の実現

TCP コンパチブルな $f_{inc}$および $f_{dec}$のポリシを導入する．MBFC において送信レートは

$$T_{mon} = \frac{\mathrm{MSS} \cdot m\_reg}{rate} + \mathrm{RTT} + backoff \qquad (1)$$

毎に調節される．ここで *backoff* は implosion 問題を回避するために受信者側で応答を返す際にランダム時間応答を抑制するバックオフタイム機構 [6] で使用される最大応答抑制時間を表す．応答抑制による遅延を含まない真の RTT を測定するために受信者はモニタ結果に自身が使用した応答抑制時間を含めて報告する．*m_reg* はモニタ期間中にマルチキャスト送信されたデータ総数を表す．

RTT 毎に *cwnd* が更新される TCP に対して，MBFC は RTT に比べてより大きな $T_{mon}$毎に送信レートの調整を行うことになる．よって MBFC は $T_{mon}$時間内に TCP が送信するセグメント数を見積もり同等量のデータを送出可能とする必要がある．

通信開始時またはタイムアウト直後，TCP 送信者は slow start の状態にある．この状態において，*ssthresh* に達するまで RTT 毎に *cwnd* は倍々に増やされる．ここで $n$ を以下のように定義する．

$$n \stackrel{\mathrm{def}}{=} \left\lfloor \log_2 \left( \frac{ssthresh}{\mathrm{MSS}_{tcp}} \right) \right\rfloor$$

*ssthresh* に到達するまでに，TCP 送信者は $\sum_{k=0}^{n} 2^k = 2^{n+1} - 1$ セグメントを送信する．よってその間の平均送信レートは

$$\frac{\mathrm{MSS}_{tcp} \cdot (2^{n+1} - 1)}{n \cdot \mathrm{RTT}} \doteq \frac{2 \cdot ssthresh - \mathrm{MSS}_{tcp}}{n \cdot \mathrm{RTT}}$$

$n \cdot$ RTT 時間後，タイムアウトが起きなければ *cwnd* は *ssthresh* に達し，TCP 送信者の状態は slow start から RTT 毎に 1 MSS ずつ *cwnd* を増やす additive increase

に移る．この間の平均送信レートは以下で計算される．

$$\frac{\mathrm{MSS}_{tcp}\cdot(2^{n+1}-1)+\sum_{k=1}^{N-n}(ssthresh+\mathrm{MSS}_{tcp}\cdot k)}{N}$$
$$\doteq \frac{1}{N}\left(ssthresh\cdot(N-n+2)+\mathrm{MSS}_{tcp}\cdot(N-n-2)\frac{(N-n+1)}{2}\right),\quad \left(\text{where } N=\left\lfloor\frac{T_{mon}}{\mathrm{RTT}}\right\rfloor\right)$$

コネクション確立時に two-way あるいは three-way handshake を行う RM プロトコルの場合には，MBFC はレート制御を開始時に既に適当な RTT 値を得ている．その場合は初期送信レートを以下の値に設定する．

$$R_{ini}=\begin{cases}\frac{2\cdot ssthresh-\mathrm{MSS}_{tcp}}{n\cdot\mathrm{RTT}}, & (\text{if } T_{mon}\le n\cdot\mathrm{RTT})\\ \frac{1}{N}\left(ssthresh\cdot(N-n+2)+\mathrm{MSS}_{tcp}\cdot\frac{(N-n-2)(N-n+1)}{2}\right), & (\text{otherwise})\end{cases}$$

RTT が MBFC 開始時に分からない場合には，$R_{ini}$は控え目な値に設定しておく必要がある．

一旦送信レートが $R_{ini}$にセットされると，送信者は TCP の additive increase と同様に線形にレートを増加させていく．TCP の場合 $T_{mon}$毎に $cwnd$は

$$\mathrm{MSS}_{tcp}\cdot\frac{T_{mon}}{\mathrm{RTT}}$$

増加する．よって MBFC は $T_{mon}$毎に

$$\alpha\cdot\frac{T_{mon}}{\mathrm{RTT}}\quad(\text{where } \alpha>0,\text{constant})\qquad(2)$$

ずつ増加するように設定する．輻輳時には送信レート $rate$ を以下のように計算し直す．

$$rate=\mathrm{MIN}\left(R_{ini},\frac{rate}{2}\right)\qquad(3)$$

初期送信レートが高過ぎた場合には，式 (3) によって TCP の multiplicative decrease 同様半分に下げられる．

## 5 評価

本節では TCP トラヒックとの共存性に関する MBFC の輻輳制御方式の有効性を確認するために行った評価実験ついて述べる．実験に際してはバルク RM プロトコルとして MBFC を実装した RMTP[1][2] を，バルク TCP として ftp を用いた．

### 5.1 実験シナリオ

図 2に本評価で使用した実験環境の構成を示す．各ホスト H1〜H6 の OS は Solaris 2.5.1 を使用した．ルータにおいて Fair Queueing, Priority Queueing, Random Early Detection は動作していない．ルータ間を除く全てのリンクは 10Mbps の Ethernet で接続され，ルータ間のみ T1 回線でレームリレーシミュレータを介して接続されている．つまりルータ間リンクを容易に輻輳が起こるネットワーク上のボトルネックとして設定している．実験に際してはホスト H1 から H3 への ftp による 20MBytes のデータ転送とホスト H2 から H4, H5, H6 への RMTP による同サイズのデータ転送を同時に行い，受信側ホスト H3〜H6 のサブネット上で TCP(ftp), UDP(RMTP) のパケット到着時間を測定により単位時間当たりの受信レートを測定しボトルネックリンクのでの両トラヒックの占有率を評価した．表 1に本実験の MBFC でのパラメータ設定を示す．backoff time 2.5sec は 1000 台程度の受信者に対して feedback implosion を起こさない条件設定である [2]．RMTP のデータパケットサイズを 1024bytes とした．測定によると TCP の MSS は 1500bytes であった．

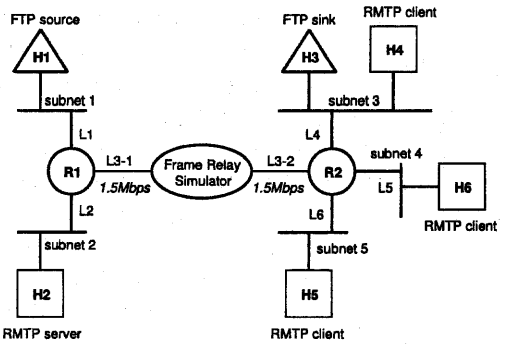


図 2: 実験環境

| | | | | | |
|---|---|---|---|---|---|
| $\mathrm{TH}_{up}$ | : | 100% | モニタ区間 | : | 10DTs |
| $\mathrm{TH}_{down}$ | : | 100% | backoff time | : | 2.5sec |
| ssthresh | : | 64KB | 係数 $\alpha$ | : | 50 |

表 1: 本実験においての MBFC のパラメータ設定

### 5.2 実験結果

図 3に 20MBytes 同時データ転送時の H3(ftp:sink) および H4(RMTP:client) での受信レートの測定結果を示す．ここで H5,H6 での受信レート結果は H4 のものとほぼ同一だったため省略した．これはボトルネックリンクを通過後に個別のサブネットあるいはホスト内部でのデータ紛失はほとんどなかったことを意味している．転送開始直後は TCP は slow start により十分早く高スループットに到達し利用帯域をほぼ占有している．一方 RMTP はコネクション確立の手続後にデータ転送を開始するため H4 での受信レートが一旦下がっている．その後 H4 の受信レートは徐々に上がり，送信開始後 10〜50 秒程度まで

はH3/H4ともに同程度の受信レートを示しており，理想適な共有状態を示している．ログによるとこのときH2はadditive increase状態にあり，緩やかに送信レートを上昇させている．その後50秒付近で輻輳を起こし一旦レートを下げ，再び送信レートを上昇させている．一方TCPにおいてもRMTPがレートを下げた数秒後に20KB/sec程度受信レートが落ちている．この時間差はルータR1において最大長となった待ち行列に入れられたパケットが送信されるまでの遅延時間によるものと考えられる．このことはH4での受信レートのグラフはx軸方向に数十秒程度並行移動した場合に両トラヒックの合計が1.5Mbpsに近い値になることからも理解できる．以降こうした上下動を繰り返し結果的にリンク容量を共有しながら，ftpは約280秒，RMTPは約340秒でデータ転送を完了している．RMTPにおいてデータ転送終了直前に大きく受信レートが落ち込んでいるのは，サーバによる再送処理のために一時的にデータが流れてこなかったためである．

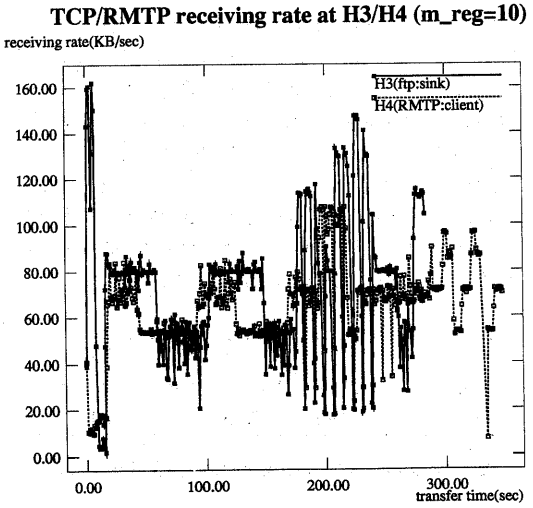


図 3: H3(ftp:sink), H4(rmtp:client)の受信レート

## 6 まとめ

本報告ではbulk reliable multicast(RM)に対するフロー／輻輳制御機構としてウィンドウ制御とフロー制御の比較検討を踏まえ，Monitor-based Flow Control (MBFC)およびMBFCにおいて現在のインターネット上の大部分を占めるTCPトラヒックとの共存を可能にする輻輳制御方式を提案し，その評価を行った．MBFCは送信者駆動型のモニタ機構を持ち，受信者に対して定期的にデータの受信状況を報告させることによりフロー／輻輳制御を行う．MBFCはモニタの開始・終了を指示するモニタ部とモニタ結果のまとめとそれに対応した送信レートの調整を行うフィードバック部から構成される．本報告においてはフィードバック部内のレート制御ポリシについて，RMトラヒックとTCPトラヒックとの共存を可能にするという観点から，TCPにおける輻輳制御(slow start, additive increase/multiplicative decrease)に適切に反応するバルクRM用の輻輳制御方式について考察した．

提案する輻輳制御方式を実装したバルクRMプロトコルRMTP[1][2]とftpによるバルクデータ転送において共有リンクの利用帯域を測定する評価実験を行った．実験結果はRMとTCPの両者間で効果的に帯域の共有が達成できること示している．

## 参考文献


[1] T Shiroshita, et. al., "Reliable Multicast Transport Protocol and its Applicability to Emerging Networks", *Technical Report of IEICE*, IN95-140, Mar. 1996.

[2] T. Shiroshita, et. al., "Performance evaluation of reliable multicast transport protocol for large-scale delivery", *IFIP PfHSN '96*, 1996.

[3] T. Sano, et. al., "Monitor-based Flow Control for Reliable Multicast Protocol and Its Evaluation", *Proc. of IEEE IPCCC '97*, pp.403-409, Feb. 1997

[4] Van Jacobson, "Congestion Avoidance and Control", *Proc. of ACM SIGCOMM '88*, pp.314-329, 1988

[5] Sally Floyd, et. al., "Random Early Detection Gateways for Congestion Avoidance", *IEEE/ACM Transactions on Networks*, Vol. 1, No. 4, pp.397-413, Aug. 1993

[6] P. B., Danzig, "Flow Control for Limited Buffer Multicast", *IEEE transaction on software enginieering*, Vol.20, No.1, pp1-12, Jan.1994.

[7] S. Floyd, et. al., "A realiable multicast framework for light-weight sessions and application level framing", *Proc. of ACM SIGCOMM '95*, pp.342-356, 1995

[8] H. Holbrook, et. al., "Log-Based Receiver-Reliable Multicast for Distributed Interactive Simulation", *Proc. of ACM SIGCOMM '95*, Aug. 1995

[9] J.C. Lin, et. al.,"RMTP: A Reliable Mutlicast Transport Protocol", *IEEE INFOCOMM '96*, pp.1414-1424, 1996

[10] J.R. Cooperstock, et. al., "Why Use a Fishing Line When You Have a Net? An Adaptive Multicast Data Distriution Protocol", Proc. of 1996 USENIX Technical Conference, Jan. 1996.

[11] L. Vicisano, et. al., "One to Many Reliable Bulk-Data Transfer in the MBone", *HIPPARCH 1997*, http://www.cs.ucl.ac.uk/staff/l.vicisano/hipparch97.ps.gz, Jun. 1997.

[12] R. Braden, "Requirements for Internet Hosts – Communication Layers", *RFC1122*, Oct. 1989

[13] W. Stevens, "TCP Slow Start, Congestion Avoidance, Fast Retransmit, and Fast Recovery Algorithm", *RFC2001*, Jan. 1997

[14] W. Stevens, "TCP/IP Illustrated, Volume 1: The Protocols", *Addison-Wesley Publishing Company*, 1994