NACK-based 高信頼マルチキャストのための RTT 見積方式

森谷優貴† 渥美幸雄†

同一ファイルを複数の受信者に配信する際には，高信頼マルチキャスト（Reliable Multicast：RM）が有効である．NACK に基づいて信頼性を保証する NACK-based RM では，NACK や再送データがロスした場合，受信者が NACK の再送を行う必要があり，受信者は NACK を送信すると NACK 再送タイマをかける．パケットロスの回復までの遅延時間を短くし，無駄な再送を回避するためには，NACK 再送タイマは送受信者間の RTT を基準として設定する必要がある．また，多くの RM における輻輳制御方式でも，制御パラメータとして RTT を使用している．このように，RTT は RM における重要なパラメータの 1 つであり，できるかぎり正確に見積もる必要がある．本論文では，NACK-based RM において，RTT を必要とする受信者ほど高頻度に，ネットワークに必要のない負荷をかけることなく，RTT が必要な時点で正確な RTT を見積もるために，NACK，NACK 確認および再送データを使用して階層的に RTT を見積もる方式を提案する．また，本提案方式を NACK-based RM の RTT 見積機構としてシミュレーション環境上に実装し，性能評価と RM 通信に与える効果の検証を行った．

RTT Estimation Method for NACK-based Reliable Multicast

Yuki Moritani† and Yukio Atsumi†

Reliable multicast is useful for distributing files to lots of receivers. In NACK-based reliable multicast communications that use NACK for guarantee of reliability, the multicast receivers need to retransmit NACK to repair the loss of NACK or retransmission data. Therefore, multicast receivers start NACK retransmission timer when they have transmitted NACK. It is necessary to set the retransmission timer based on the RTT between a receiver and a sender for shortening the recovery delay interval and avoiding useless retransmissions. And the RTT is also used as a coutrol parameter in lots of reliable multicast congestion control methods. Therefore, the RTT is an important parameter in reliable multicast communications and it is necessary to estimate as an accurate RTT as possible. In this paper, we propose a hierarchal RTT estimation method that uses NACK, NACK confirmation and retransmission data. By using our proposed method, a receiver who needs an accurate RTT can estimate it more frequently at the moment when it is needed without applying unnecessary load on the network. We implemented our proposed method as an RTT estimation function of NACK-based reliable multicast in the simulation environment, and evaluated the performance and the impact to the reliable multicast communications.

1. はじめに

インターネットを介した音楽ファイルの配信や，プログラムのアップデート等が一般的に行われるようになり，大容量のファイルを，ネットワークを通して送受信する機会が増加している．複数の受信者に同一ファイルを配信する場合は，高信頼マルチキャスト（Reliable Multicast：RM）が有効であり，IETF（Internet Engineering Task Force）においてもさかんに標準化活動がなされている[1)~5)]．

† 株式会社 NTT ドコモマルチメディア研究所
Multimedia Laboratories, NTT DoCoMo Inc.

文献 1) で述べられているように，RM に対する要求はアプリケーションごとに様々であり，唯一の RM プロトコルにより，すべてのアプリケーションの要求を満たすことは実現性に乏しい．このため，IETF では複数のプロトコルの標準化を同時に進めており，信頼性保証機構の違いにより，以下の 3 つのプロトコル（Protocol Instantiation）が提案されている．

- NACK-based protocol
- Tree-based ACK protocol
- Asynchronous Layered Coding protocol using FEC（ALC+FEC）

また，RM プロトコルを構成する各要素を Building Block に分解し，Building Block 単位での標準化も進

められている[6]．

ノード間の往復遅延時間（Round Trip Time：RTT）はRMプロトコルにとって重要なパラメータの1つであり，特にNACK-basedなRMプロトコルでは，Building Blockの1つとしてRTT Collectionが定義されている[7]．NACK-based RMでは，パケットをロスした受信者が，再送要求であるNACKパケットのロスや再送データパケットのロスの発生に備えて，NACK再送のためのタイマをかける．このタイマ基準値として送受信者間のRTTが利用される．また，NACK-basedなRMプロトコルに限らずRMの輻輳制御機構では，インターネットトラヒックの90%を占めるTCPと親和性を保つことが重要であるため，受信者のパケットロス率とRTTからTCPの近似スループット関数を用いて，送信者が送信レートの制御等を行う必要があり，全受信者中スループットが最低となる受信者（最悪受信者）のパケットロス率とRTTを利用する．このように，NACK-basedなRMでは，RTTが必要とされるのはパケットロス発生時が主であり，また最悪受信者は主としてパケットロス率が高い受信者であるため，パケットロス率が高い受信者ほど正確なRTTを見積もることが必要である．

しかしながら，既存のRMにおけるRTT見積方式は，定期的にノード間でRTT見積パケットをやりとりすることでRTTを見積もる方式である．このため，受信者のRTTの見積頻度は全受信者で一定となり，誤ったRTTをNACK再送タイマ基準値としてしまうことがあり，ネットワーク中に無駄なNACKや再送データが送信されたり，パケットロスが長時間回復しなかったりするという問題点がある．

これを解決するため，本論文では，NACK-based RMにおいて，ネットワークに不要な負荷をかけず，パケットロス率が高い受信者ほど測定頻度が高く，またRTTを必要とするパケットロス発生時に正確なRTTを見積もることができるように，パケットロス発生時にやりとりされるNACK，NACK確認，再送データにより階層的にRTTを見積もる方式を提案する．また，計算機シミュレーションを用いて従来方式と比較評価することにより，提案方式の有効性を検証する．

本論文は，以下のように構成されている．2章では，NACK-basedなRMの概要，および使用される制御機構について説明し，NACK-based RMプロトコルにおけるRTT見積りの重要性について詳細に記述する．次に3章において，従来のRTT見積方式について説明し，4章では，従来方式の問題点を解決することができるRTT見積方式を提案する．5章では，計算機シミュレーションにより，提案方式の性能評価と，RM通信に与える効果の検証を行い，最後に6章でまとめを行う．

2. NACK-based 高信頼マルチキャスト（NACK-based RM）

2.1 プロトコル概要および制御機構

ユニキャストの代表的な信頼性プロトコルであるTCPでは，信頼性保証機構としてACKを使用し，受信者はパケットを正常に受信することができた場合に，送信者にACKを送信することでパケットの信頼性を確保している．これをマルチキャストで行うと，送信者が1パケット送信するたびに，全受信者からACKが届くことになるので，送信者の負荷が増加する（ACK implosion）ため，スケーラビリティが得られないという問題がある．このため信頼性保証機構としてNACKを使用するNACK-basedなRMプロトコルでは，受信者がパケットロスを検出した際にNACKを送信者に送信することでパケットロスを通知し，送信者が再送を行うことで信頼性を確保している．NACKを送信する受信者はパケットをロスした受信者のみであるため，信頼性保証機構としてACKを使用する場合と比較して，スケーラビリティを得ることができる．

しかしながら，NACK-basedなプロトコルであっても，送信者に近いリンク上でパケットロスが発生すると，多くの受信者においてパケットロスとなる．このため，送信者にNACKが集中（NACK implosion）してしまい，スケーラビリティが得られないという問題がある．この問題に対処するため，SRM（Scalable Reliable Multicast）[8]において，NACK Suppression機構が提案されている．NACK Suppressionは，受信者がパケットロスを検出した際に，ランダム時間待った後，NACKをマルチキャストで送信するものである．同じパケットに対するNACKはただ1つ送信者に届けばよいため，NACK送信を待っている受信者が，他の受信者が送信したNACKを受信した場合，NACKの送信を中止することで，送信者へのNACKの集中を避けることができる機構である．

また，ARM（Active Reliable Multicast）[9]では，ネットワーク中のルータがRM通信のサポートを行い，複数の受信者から送信されたNACKを受信した場合に，最初のNACKのみを送信者に転送し，重複して届いたNACKは廃棄するNACK Fusion機構（NACK Aggregationとも呼ばれる）が提案されている．NACK Fusionを使用すると，ルータでのNACK処理量は増加するが，送信者に届くNACK数が抑え

られるため，NACK Suppression を使用した場合よりも，スケーラビリティが得られることが報告されている[10]．また，ARM ではルータが送信者から送信されたパケットをキャッシュしておき，受信者からの NACK に対して，送信者の代理で再送を行う Local Recovery も提案されている．

さらに，PGM（Pragmatic General Multicast）[4] や，AER（Active Error Recovery）/ NCA（Nominee Congestion Avoidance）[11] では，NACK Suppression と NACK Fusion を組み合わせた機構が提案されている．これらのプロトコルでは，受信者は NACK を送信する前にランダム時間待ち，送信者へ向けてユニキャストで NACK を送信する．ルータが NACK を受信すると，ルータは NACK を受信したことを通知する NACK 確認パケットを受信者にマルチキャストするとともに，その NACK が対応データに対する最初の NACK であれば，送信者に転送する．NACK 送信を待っている受信者が，NACK 確認を受信した場合は NACK 送信を中止する．この NACK Suppression と NACK Fusion を組み合わせた機構を使用することで，送信者での NACK 処理量のみならず，ルータでの NACK 処理量も減らすことができるため，さらに高いスケーラビリティを得ることが可能である．

また，一般のインターネットで RM を利用する場合には，従来の TCP トラヒックを圧迫しないように，TCP-Friendly な輻輳制御機構が必要となる[12]．TCP-Friendly な輻輳制御を実現するためには，TCP と同様の輻輳制御アルゴリズムを取り入れたフロー制御が必要となるが，全受信者が TCP の ACK に相当するパケットを送信すると，ACK Implosion となるため現実的でない．このため，AER/NCA や PGMCC [13] において，RM の受信者中で経路が最も輻輳しており，受信スループットが最も小さい受信者（最悪受信者）と送信者との間で，TCP と同じ輻輳制御機構を用いる方式が提案されており，これらの機構を利用することにより TCP-Friendly な輻輳制御を実現することが可能である．

2.2 RTT 見積りの重要性

NACK-based な RM では，受信者がパケットロスを検出し，NACK を送信したとしても，NACK パケットが送信者へ届く前にロスしたり，送信者が NACK に応じて送信した再送データが受信者への配信経路上で再度ロスすることがある．このため，受信者は NACK を送信した際には，NACK 再送のためのタイマをかけ，これらロスが発生により再送データが届かず，タイマが切れた場合には，NACK を再送する必要がある．ここで使用するタイマ値が短すぎる場合，送信者からデータが再送されているにもかかわらず再度 NACK パケットを送信してしまうことになり，ネットワークに無駄なパケットが流れることになる．また，タイマ値が長すぎると NACK や再送データがロスした場合，再送までの待ち時間が長くなり，パケットロスが長時間回復しない．このため，受信者はタイマ値を適切に設定する必要がある．NACK を送信してから，再送パケットが届くまでの時間は送受信者間の RTT となるので，このタイマは送受信者間の RTT を基準とするのが望ましく，このため受信者は NACK を送信するときに自身から送信者までの正確な RTT 値を知っていなければならない．なお，本論文では，この送受信者間の RTT 値を受信者の srcRTT と定義する．

また，NACK Suppression を使用する場合，受信者は NACK 送信までランダム時間待つ必要があり，このため NACK 送信までのタイマをかける．上記の NACK 再送の場合と同じように，ここで使用するタイマ基準値が短すぎる場合は，他の受信者が NACK を送信しているにもかかわらず NACK を送信してしまうことになり，また長すぎるとほかにロスした受信者が存在しない場合には，データ回復までの待ち時間が長くなってしまう．効率的に NACK Suppression を働かせるためには，タイマ基準値として全受信者中で，最も srcRTT が大きい受信者の srcRTT を基準とし，ランダム数を乗じてタイマ値とするのが適切であることが知られている[14]．なお，AER/NCA や PGM のようにルータが NACK 確認を送信する場合は，ルータ配下の受信者中でルータとの RTT が最も大きい受信者の RTT を NACK Suppression の際のタイマ基準値とする必要がある．なお，本論文では，この NACK Suppression の際のタイマ基準値となる RTT 値を supRTT と定義する．

また，AER/NCA や PGMCC 等の輻輳制御機構を使用するためには，最悪受信者を適切に選択する必要がある．最悪受信者は，TCP の近似スループット関数[15]により，パケットロス率と srcRTT を用いて，最も帯域が小さいと想定される受信者を，送信者が選択する．パケットロス率は受信パケットを受信者自身が測定できるため，正確な srcRTT を見積もることができれば，自身の状態を上記文献中に記載されている方法や，文献 16) に記載されている通知方法を用いて，送信者に通知することにより，TCP-Friendly な輻輳制御が実現できる．

このように NACK-based な RM では，srcRTT や supRTT は重要なパラメータであり，受信者がこれら

を正確に見積もることができなければ，無駄なNACKや再送データの送信によるネットワーク負荷の増大，誤った輻輳制御によるスループットの低下や，他トラヒックの圧迫等を引き起こす恐れがある．したがって，RTT見積機構は非常に重要な機構であるといえる．また，ネットワークの状態は時々刻々と変化するため，変化に追従して正確な値を見積もることが必要である．

3. 従来のRTT見積方式

3.1 従来方式の概要

インターネットのような広範囲なネットワークにおいて，RMを行う場合，マルチキャストの送受信を行う全ノードの時計を同期することは現実的には不可能である．このため，RMのRTT見積方式としては，ノード間の時刻同期を必要としない方式が必要となる．

SRMではシーケンス番号等の制御情報のやりとりを行うSession Messageを使用してRTT見積りを行っている．ノード A が時刻 t_1 にSession Message P_a に t_1 の情報を含めて送信し，ノード B が時刻 t_2 に P_a を受信したとすると，ノード B は時刻 t_3 にSession Message P_b を送信する際，$\Delta = t_3 - t_2$ を計算し，P_b に t_1 と Δ の情報を含めて送信する．ノード A が時刻 t_4 に P_b を受信すると，ノード A はAB間のRTT T_{ab} を $T_{ab} = t_4 - t_1 - \Delta$ として見積もる．SRMではSession Messageは定期的にマルチキャストされ，Session Messageを送信する際には，他の全ノードからのSession Messageの送信時刻のタイムスタンプ値と受信から送信までの時間差をリストとして送信することで，Session Messageを受信した全ノードがSession Messageを送信したノードまでのRTTを見積もることができる．

しかしながら，SRMのRTT見積方式はSession Message中に全ノードのタイムスタンプ値と時間差情報を含める必要があり，また，全ノードがSession Messageの送信を行う必要があるため，ノード数の2乗に比例した情報量がRTT見積りに使用され，スケーラビリティに乏しい．このためSharmaらは，いくつかのノードよりなるグループから代表ノードを選出し，代表ノードと代表でないノード間，および代表ノードどうしの2階層に分けてRTT見積りを行い，代表ノードがグループの他のノードに対して他の代表ノードとのRTTをリストとして通知することにより，SRMのRTT見積りを階層的に行う方式を提案している[17]．この方式により，RTT見積りのためのパケット量を削減することができ，ある程度スケーラビリティを向上させることができる．しかしながら，依然として情報量はノード数の2乗に比例するため，スケーラビリティには限界があり，また代表ノードが測定するのは他の代表ノードとのRTTであるため，代表でないノードどうしのRTT等は正確に見積もることができないという問題点がある．

スケーラビリティの問題を解決するため，Ozdemirらはノード間のRTTが一方向遅延の2倍であるという仮定の下で，ノード数に比例した情報量でRTTを見積もる方式を提案している[18]．この方式では，最初にReference Pointと呼ばれるノード S が送信時刻のタイムスタンプを含んだRTT見積パケットをマルチキャストし，受信した全ノードはそのタイムスタンプを含むパケットを S に返信することで，S が他の全ノードとのRTTを見積もる．その後，S は見積もった全ノードと S とのRTTをリストとして，全ノードにマルチキャスト送信することで，全ノードが S と他のノードのRTTを知る．次に，あるノード Q がプローブパケット m_q をマルチキャストで送信し，S が m_q を受信すると S は通知メッセージ M_q をマルチキャストで送信する．あるノード R が m_q，M_q を受信した時刻をそれぞれ t_m，t_M とし，S と Q，S と R，R と Q のRTTをそれぞれ d_{sq}，d_{sr}，d_{rq} とすると $d_{rq}/2 = d_{sq}/2 + d_{sr}/2 - (t_M - t_m)$ となり，d_{sq}，d_{sr} は上記通知メッセージから既知であるので，d_{rq} を見積もることができる．なお，この方式は S を中継受信ノードとし，階層的にRTT見積りを行うことで，RMTP-II[19]等のTree-based ACKプロトコルに適用することも可能である．

また，AER/NCAではNACK Fusion等を行うルータ（Active Router：AR）がRTT見積りをサポートすることで，階層的にRTTを見積もる方式が提案されている．ARが定期的にRTT見積パケットを送信し，送信者が返信することで，ARは送信者と自身とのRTTを見積もる．各受信ノードが同様にARとのRTTを測定する際に，ARが送信者とAR間のRTTを返信パケットに含めることで，受信者はARと自身のRTTと送信者とARとのRTTとの和から，自身と送信者とのRTTを見積もることができる．この方式では，一方向遅延からRTTを見積もるのではなく，実測RTTの和を使用しているため，送信者から受信者へのデータ配信経路と，受信者から送信者へのNACK送信経路が異なる場合であっても，正確なRTT見積りを行うことができる．なお，AER/NCAやPGMでは，SPM（Source Path Message）を使用して，ARのアドレスを受信者に通知することで，データの配信時およびNACK送信時には，必ず同一

の AR を経由することが保証されているため，このような RTT 見積方式を使用することができる．

3.2 従来方式の問題点

2.2 節で述べたように，srcRTT や supRTT という RTT 値は RM の重要なパラメータであり，NACK を送信する際のタイマ基準値や，輻輳制御のための最悪受信者選択に使用される．受信者が NACK を送信するのは，パケットロスを検出した場合であり，その際，送受信者間の経路で輻輳が発生し，RTT は大きくなっていると想定される．また，最悪受信者はパケットロス率，RTT が大きい受信者である．これらのことから，経路の輻輳により，パケットロス率，RTT がともに高くなっている受信者ほど頻繁に RTT 見積りを行い，正確な RTT を知る必要があることが分かる．しかしながら，従来の RTT 見積り方式では，各受信者が定期的に見積りを行うため，各受信者の RTT 見積頻度は一定となる．このため，頻繁な RTT 見積りを必要とする受信者の RTT 見積頻度を高くするといったことができないという問題点がある．また，RTT 見積パケットを使用して RTT を見積もるため，頻繁に RTT 見積りを行う場合には，RTT 見積パケットがネットワークの負荷となるという問題がある．また，ネットワーク規模や輻輳頻度，受信者数等によって適切な RTT 見積間隔があると考えられるが，最適値を決定することは非常に困難であるという問題がある．

4. 提案 RTT 見積方式

4.1 提案方式の説明

3.2 節で述べた問題点を解決するため，本論文では，受信者がパケットロス検出時に送信する NACK，Active Router（AR）が NACK を受信した際に受信者にマルチキャストする NACK 確認，送信者が NACK に応じて送信する再送データを用いて，階層的に srcRTT や supRTT の見積りを行う RTT 見積方式を提案する．

4.1.1 用語の定義

本提案方式で使用する用語を以下のように定義する．upRTT はそれを保持するノード（受信者もしくは AR）と，そのノードからみて送信者方向（上流）のノード（AR もしくは送信者）間の RTT 値である．supRTT(Up) は，上流ノードの配下で上流ノードとの upRTT が最大となるノードの upRTT であり，受信者もしくは AR が上流ノードに NACK を送信する際に NACK Suppression の基準値として使用する supRTT である．一方，supRTT(Down) は，上流ノードが配下ノードに通知するために保持する supRTT であり，上流ノードは，配下ノードからの upRTT 通知をもとに，その最大値を supRTT(Down) として選択する．したがって，上流ノードの supRTT(Down) と配下ノードの supRTT(Up) は，同じ値となる．

表 1 各ノードの保持パラメータ
Table 1 Parameters of each node.

ノード	保持パラメータ
送信者	supRTT(Down)
AR	srcRTT, upRTT, supRTT(Up, Down)
受信者	srcRTT, upRTT, supRTT(Up)

表 2 各パケットの付加データ
Table 2 Additional data of each packet.

パケット	付加データ
NACK	Sequence No., ID, timestamp, upRTT
NACK 確認	Sequence No., ID, timestamp, supRTT
再送データ	ID, timestamp, srcRTT, supRTT

4.1.2 各ノードの保持パラメータ

表 1 に，本提案方式で各ノードが保持するパラメータをそれぞれ示す．受信者は upRTT，srcRTT，supRTT(Up) をパラメータとして保持し．AR も同様のパラメータを保持するが，supRTT については，自身が NACK Suppression を行う際に，タイマ基準値として使用するための上流への supRTT(Up) と，配下受信者へ通知するための supRTT(Down) の 2 つを保持する．また，送信者は supRTT(Down) のみをパラメータとして保持している．なお，これらのパラメータの初期値は AER/NCA で使用されている RTT 見積パケットを用いた方式により見積もる．

次に，提案 RTT 見積方式のアルゴリズムについて説明するが，srcRTT，supRTT の見積りはそれぞれ独立に行われるため，両者をわけて説明することとする．また，**表 2** は，本提案方式において，これら RTT を見積もるために NACK，NACK 確認，再送データの各パケットに付加するデータである．

4.1.3 srcRTT の見積り

図 1 は提案 RTT 見積方式における srcRTT 見積りの概要図である．

受信者はパケットロスを検知すると，NACK パケットを上流ノードに送信する．この際，NACK パケットに自身の ID（IP アドレス等）と送信時刻のタイムスタンプ値を付加する．また，AR は NACK を受信すると配下ノードに NACK 確認をマルチキャストで送信する．この際，NACK 確認パケットに NACK パケット中の ID とタイムスタンプ値を付加する．NACK 確認パケットを受信した受信者は，NACK 確認受信時刻と NACK 確認パケット中のタイムスタンプ値の差

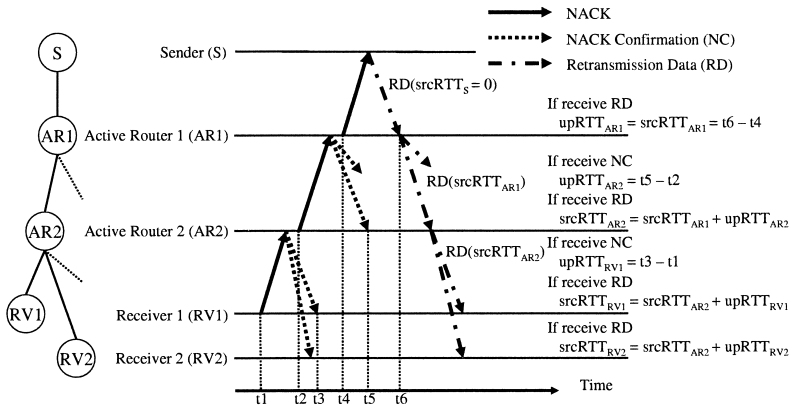

図 1 提案 RTT 見積方式における srcRTT 見積り
Fig. 1 srcRTT estimation in proposed RTT estimation method.

分をとることで，自身の upRTT を見積もる．なお，upRTT を見積もるのは，自信の ID と NACK 確認パケット中の ID が一致する場合のみである．同様にして，送信者直下でない AR（図中の AR1）も NACK と NACK 確認により，自身の upRTT を見積もる．NACK が送信者に到達すると，送信者はパケットをロスした受信者が存在することを検知し，再送データを送信する．この際，AR が NACK 確認パケットを送信するのと同様に，NACK 中の ID とタイムスタンプ値を付加する．また，自身の srcRTT（送信者の場合は 0）も付加して送信する．再送データを受信した送信者直下の AR は，自身の upRTT を見積もるとともに，再送データ中の srcRTT（この場合は 0）と自身の upRTT の和をとることにより，自身の srcRTT を見積もる．AR は配下ノードに再送データを送信する際，再送データに自身の srcRTT を付加して送信する．配下ノードは，再送データを受信すると，自身の upRTT と再送データ中の srcRTT（上流ノードの srcRTT）の和をとることにより自身の srcRTT を見積もる．受信者まで再送データが到達すると，受信者も AR と同様に自身の srcRTT を見積もる．各ノードが上記アルゴリズムに従うことで，NACK，NACK 確認，再送データを使用して正確な srcRTT を見積もることができる．また，図 1 中で RV2 が srcRTT の見積りもを行っているように，本提案方式では，srcRTT を自身の upRTT と上流ノードの srcRTT の和で見積もるため，NACK と NACK 確認パケットにより upRTT を見積もった受信者のみならず，再送データを受信した全受信者が srcRTT を見積もることができる．

4.1.4 supRTT の見積り

図 2 は提案 RTT 見積方式における supRTT 見積りの概要図である．

受信者は NACK パケットを送信する際，自身が前回見積もった upRTT を付加して送信する．NACK パケットを受信した AR は，NACK 中の upRTT と自身が保持している supRTT(Down) を比較し，前者の方が大きければ supRTT(Down) を NACK 中の upRTT で更新する．また，AR は NACK 確認パケットを送信する際には，先ほど更新した supRTT(Down) を付加して送信する．NACK 確認パケットを受信した受信者は，NACK 確認パケット中の supRTT で自身の supRTT(Up) を更新する．AR が上流ノードに NACK パケットを送信する際も同様に，NACK パケットに自身の upRTT を付加して送信し，NACK パケットを受信した上流ノードは supRTT(Down) を更新する．NACK パケットが送信者に到達した場合は，送信者は supRTT(Donw) を更新した後，再送データに supRTT(Down) を付加して送信する．再送データを受信した送信者直下のノードは，supRTT(Up) を更新する．また AR は再送データを配下ノードに配信する際にも，現在の supRTT(Down) を付加して送信する．受信者は再送データを受信した場合も supRTT(Up) を更新する．NACK 確認のみでなく，再送データでも supRTT(Down) を通知することで，NACK 確認送信後に更新された supRTT(Down) についても通知でき，配下ノードの supRTT(Up) と上流ノードの supRTT(Down) にずれが生じている時間を短くすることができる．各ノードがこのようなアルゴリズムに従って処理を行うことにより，AR および受信者は正確な supRTT を見積もることができる．なお，更新された supRTT の通知はマルチキャストで送信される NACK 確認や再送データによって行われるため，srcRTT の場合と同様に NACK を送信していないノードも supRTT を見積もることができる．

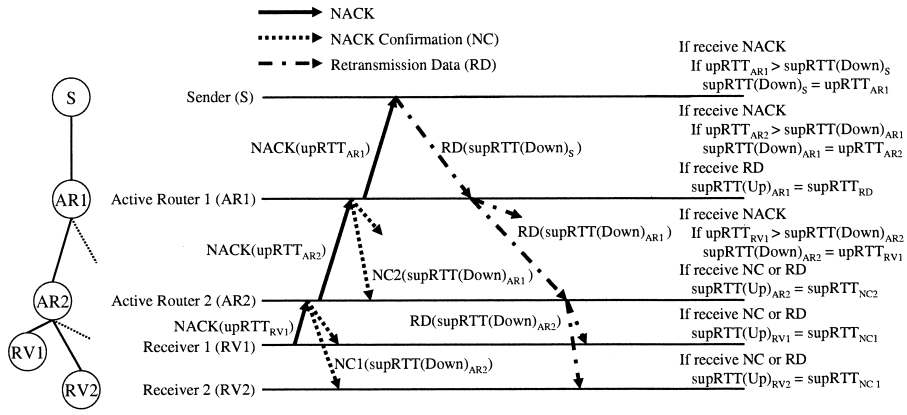

図 2 提案 RTT 見積方式における supRTT 見積り
Fig. 2 supRTT estimation in proposed RTT estimation method.

また，**図 3** は，提案方式における各ノードの擬似コードを示したものである．各ノードがこれらの処理に従うことで，上記説明のように，NACK，NACK 確認，再送データを用いて階層的に srcRTT，supRTT を見積もることができる．なお，これらの擬似コードからも分かるように，提案方式では，パケットの送信先等を変更する必要はなく，AER/NCA や PGM 等の一般的な NACK-based な RM プロトコルで使用されているパケットに，RTT 見積用のデータを付加し，受信時の処理を追加することで RTT 見積りを行うことができる．

4.2 提案方式の特徴

本提案方式を使用することにより，正確な RTT を必要とするパケットロス発生時に RTT 見積りを行うことができる．加えて，輻輳により RTT が増加し，パケットロス率が高くなっている受信者ほど，NACK を送信する回数が多くなるため，状態が悪く，正確な RTT を必要とする受信者の RTT 見積頻度を高くすることが可能である．また，NACK，NACK 確認，再送データを用いて RTT 見積りを行っているため，RTT 見積りのための特別なパケットを必要とせず，ネットワークに不要な負荷をかけることなく正確な RTT 見積りを行うことができる．また，階層的見積りを使用しており，NACK Suppression や NACK Fusion により upRTT 見積りを行わなかった受信者であっても，AR が見積もった輻輳経路の RTT を再送データにより知ることができるため，輻輳経路を共有する全受信者が正確な RTT 見積りを行うことができる．また本方式は，状態が悪くなった受信者が自律的に見積りを行うため，受信者数等により RTT 見積り間隔を変更する必要がなく，汎用性に優れているといえる．**表 3** に提案方式と従来方式の特徴比較表を示す．

```
Sender's protocol
  If received NACK:
    if (NACK.upRTT > supRTT(Down))
      supRTT(Down) = NACK.upRTT;
      Judge whether retransmit data or not;
        // In case of duplicated NACK in short period, no need to send
  On sending RD: // RD is Retransmission Data
    RD.srcRTT = 0;
    RD.supRTT = supRTT(Down);
    RD.ID = NACK.ID;
    RD.timestamp = NACK.timestamp;
    send RD;

AR's protocol  // AR is Active Router
  If received NACK:
    if (NACK.upRTT > supRTT(Down))
      supRTT(Down) = NACK.upRTT;
      Judge whether send NACK confirmation or not;
        // In case of duplicated NACK, no need to send by NACK Fusion
  On sending NC: // NC is NACK Confirmation
    NC.supRTT = supRTT(Down);
    NC.ID = NACK.ID;
    NC.timestamp = NACK.timestamp;
    send NC;
    start NACK suppression timer;
  On sending NACK:  // NACK suppression timer expired
    NACK.ID = myID; //myID is this Node's ID
    NACK.timestamp = gettimeofday();
    NACK.upRTT = upRTT;
    send NACK;
  If received RD:
    if (RD.ID == myID)
      upRTT = gettimeofday() − RD.timestamp;
    supRTT(Up) = RD.supRTT;
    srcRTT = RD.srcRTT + upRTT;
    RD.supRTT = supRTT(Down);
    RD.srcRTT = srcRTT;
    send RD;

Receiver's protocol
  On sending NACK:  // When detect packet loss
    NACK.ID = myID;
    NACK.timestamp = gettimeofday();
    NACK.upRTT = upRTT;
    send NACK;
    start NACK retransmission timer;
  If received NC:
    if (RD.ID == myID)
      upRTT = gettimeofday() − RD.timestamp;
    supRTT(Up) = RD.supRTT;
  If received RD:
    if (RD.ID == myID)
      upRTT = gettimeofday() − RD.timestamp;
    supRTT(Up) = RD.supRTT;
    srcRTT = RD.srcRTT + upRTT;
```

図 3 提案 RTT 見積り方式の擬似コード
Fig. 3 Pseudocode in proposed RTT estimation method.

表 3 提案方式と従来方式の特徴比較
Table 3 Features comparison of proposed and conventional methods.

方式	見積方法	見積周期および頻度
SRM	End-End で双方向	周期的で一定
Sharma et al.	部分階層的で双方向	周期的で一定
Ozdemir et al.	階層的で一方向	周期的で一定
AER/NCA	階層的で双方向	周期的で一定
提案方式	階層的で双方向	非周期的で可変（高ロス率なら高頻度）

表 4 シミュレーションパラメータ
Table 4 Simulation parameters.

パラメータ	値
Reliable Multicast Protocol	AER/NCA
Local Recovery	なし
ルータキュー形式	Drop-tail
ルータキューサイズ	20 [packets]
データパケットサイズ	1,000 [bytes]
制御パケットサイズ	30 [bytes]

なお，図 1 および図 2 では，説明を簡単にするため，AR は 2 段構成としたが，AR を多段構成とした場合も，AR の上流ノードが他の AR になるだけであるので，まったく問題なく動作する．また，ARM や AER/NCA で提案されている Local Recovery を使用する場合であっても，AR が Local Recovery による再送を行う際に，再送データに自身の srcRTT を付加するので，送信者が再送を行う場合と同様に，受信者は正確な srcRTT を見積もることができる．

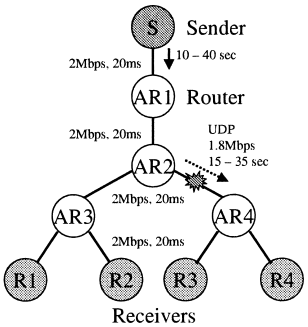

図 4 シミュレーショントポロジ 1
Fig. 4 Simulation topology 1.

5. シミュレーション評価

提案方式の有効性を検証するため，提案方式による RTT 見積りの性能の評価，および提案方式が RM 通信に与える効果について，計算機シミュレーションを用いて評価する．シミュレータとして，VINT プロジェクトのネットワークシミュレータ ns-2☆を使用し，全シミュレーション共通で**表 4** に示すパラメータを用いた．ファイル等のデータを送信するため，データパケットサイズは 1,000 Byte とし，NACK や RTT 見積り等の制御パケットのサイズは制御情報を付加することを考慮して 30 Byte であるとした．なお，これらパケットサイズは IP ヘッダ等を除いたパケットサイズである．NACK-based な RM プロトコルとして AER/NCA を使用し，提案方式と比較する従来方式として，AER/NCA で使用されている RTT 見積り方式を使用した．また，4.2 節で述べたように，Local Recovery は本提案方式と同時に使用することができるが，RTT 見積りの影響を明確に評価するために，本シミュレーションでは Local Recovery を行わないこととした．

5.1 RTT 見積性能の検証

まず，提案方式による RTT 見積りの性能を評価するため，**図 4** に示す比較的単純なツリー構造のトポロジを用いてシミュレーションを行った．4 受信者がそれぞれツリーの葉に位置し，送信者から受信者までは 4 ホップであり，すべてのリンクのリンク帯域，遅延はそれぞれ 2 Mbps，20 ms である．シミュレーション開始後 10 秒から 40 秒までの間，送信者からマルチキャストパケットを送信し，輻輳を発生させるために，15 秒から 35 秒までの間 AR2–4 間のリンクに外乱トラヒックを 1.8 Mbps の固定レートで送信した．従来方式で，定期的 RTT 見積間隔をそれぞれ 0.5 秒，1 秒，5 秒とした場合，および本提案方式を使用した場合の 4 通りについてシミュレーションを行った．

図 5〜図 8 に，従来方式を用い，見積間隔を 0.5 秒，1 秒，5 秒として RTT 見積りを行った場合，および提案方式を用いた場合の，受信者 3 における実際の RTT と見積 RTT の時間変化を示す．なお，タイマをかける時点で正確な RTT が見積もれているかどうかを分かりやすくするため，図中では，ロス検出時刻と，その時刻において受信者が保持している見積 RTT を “○” で，プロットしている．

最も輻輳している時間帯は，外乱トラヒックを送信し始めた 15 秒過ぎであり，グラフからこの時刻において最もロスが発生していることが分かる．従来方式で RTT 見積間隔を 0.5 秒とした場合，および提案方式においてはこの輻輳時においてもほぼ正確な RTT を見積もることができているが，従来方式で見積間隔を 1 秒，5 秒とした場合には正確な RTT が見積もれていないことが分かる．図 8 から分かるように，提案方式を用いた場合は，全時間を通して RTT を必要と

☆ http://www.isi.edu/nsnam/ns/

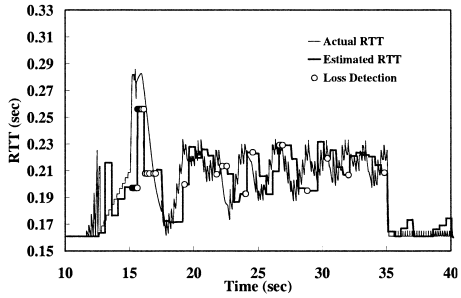

図 5 従来方式による srcRTT の時間変化（見積間隔 0.5 秒）
Fig. 5 Time-srcRTT change by using the conventional method (estimation period: 0.5 sec.).

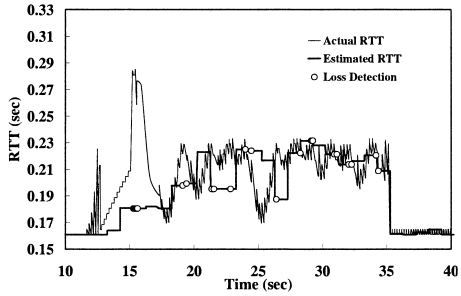

図 6 従来方式による srcRTT の時間変化（見積間隔 1 秒）
Fig. 6 Time-srcRTT change by using the conventional method (estimation period: 1 sec.).

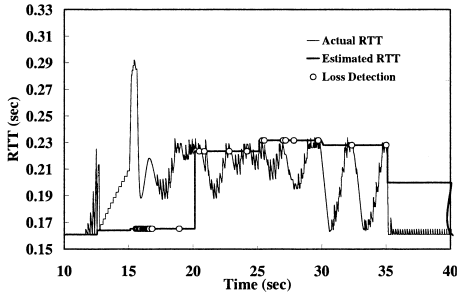

図 7 従来方式による srcRTT の時間変化（見積間隔 5 秒）
Fig. 7 Time-srcRTT change by using the conventional method (estimation period: 5 sec.).

するロス検出時点で，ほぼ正確な RTT を見積もれていることが分かる．なお，提案方式を用いた場合，外乱トラヒックがなくなった 35 秒より後の時間帯での見積 RTT が実際の RTT と大きく異なるが，このときはロスが発生せず，RTT を必要としないため問題とならない．また，その後他の外乱トラヒック等により輻輳が発生した場合には，その時点で正確な RTT を見積もることが可能である．以上のことから，提案方式を使用することで，RTT を必要とするパケットロス時に正確な RTT を見積もることができるということが分かる．

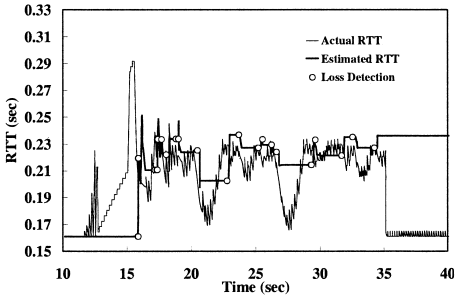

図 8 提案方式による srcRTT の時間変化
Fig. 8 Time-srcRTT change by using proposed method.

表 5 シミュレーション結果 1
Table 5 Simulation results 1.

	従来方式			提案方式
見積間隔	0.5 秒	1 秒	5 秒	
平均スループット [kbps]	468.8	511.7	503.2	517.6
総 NACK 数	1042	1076	776	278
送信 NACK 数/ロス数	1.715	0.995	1.833	0.913
不要パケット数/ロス数	0.05	0.061	0.097	0.019
正規化再送遅延時間	3.352	3.082	2.851	1.558
RTT 見積パケット総数	1328	704	224	48
RTT 見積回数（受信者 1）	83	44	14	3
RTT 見積回数（受信者 2）	83	44	14	3
RTT 見積回数（受信者 3）	83	44	14	55
RTT 見積回数（受信者 4）	83	44	14	55

5.2 RM 通信に与える効果の検証

表 5 は 5.1 節で行ったシミュレーション結果をまとめた表である．なお，ここで示す正規化再送遅延時間は，受信者がロスを検出してから再送データを受信するまでにかかる時間を RTT で割ったものである．

表から，スループットには RTT 見積りによる影響がほとんど現れていないことが分かる．輻輳制御機構を有する RM では，最悪受信者に合わせて TCP と同じ輻輳制御が行われるため，スループットは輻輳制御機構に依存する．本シミュレーションでは，RM プロトコルとして AER/NCA を使用しており，この輻輳制御における RTT 値の利用としては，srcRTT を最悪受信者選択のパラメータとして使用するだけである．スループットは，最悪受信者のパケットロス頻度等により数%は容易に変動するため，この差は RTT 見積りによる影響ではない．しかしながら，提案方式を使用した場合は，正確な RTT を見積もることができたため，再送や NACK Suppression のためのタイマを正確にかけることができ，全体としての NACK の送信量や，ロスに対する NACK 送信量，無駄な NACK 送信にともなう不要パケットの配信を防止することができていることが分かる．

一方，従来方式で RTT 見積間隔を 0.5 秒とした場合は，正確な RTT を見積もることができていたにもかかわらず，RTT 見積間隔を 1 秒とした場合と比較して，結果が悪くなっている．これは，輻輳時にも RTT 見積パケットを頻繁に送信したために，さらなる輻輳を発生させてしまったためであると考えられる．これに対し提案方式では，輻輳時に再送データの要求に使用される NACK，NACK の送信を抑圧するために送信される NACK 確認，NACK に応じて送信される再送データという再送制御機構に必要なパケットを使用して RTT 見積りを行っているため，ネットワークに不要な負荷をかけずに RTT を見積りが行え，従来方式よりも良い結果となっている．これらの結果から，提案方式を用いることにより，輻輳時にネットワークに不要な負荷をかけずに，正確な RTT 見積りが行えるということが分かる．

また，従来方式では，全受信者の見積頻度が同一であり，正確な RTT を見積もるために見積間隔を短くすればするほど，RTT 見積パケットの送信量が増加していることが分かる．これに対し，提案方式では RTT 見積パケットは初期フェーズでしか使用しないため，ほとんど送信されていない．また，輻輳が発生し，NACK を送信する必要がある受信者 3，4 の方が他の受信者と比較して，RTT 見積回数が多くなっており，受信者が自身の状態によって RTT 見積回数を自律的に調整できていることが分かる．ネットワークの状態は時々刻々と変化するため，提案方式のように受信者の状態によって RTT 見積頻度を自動的に変更できることは，非常に有効であると考えられる．

しかしながら，図 4 のトポロジでは，リンク帯域，遅延時間がすべてのリンクで同一であり，送受信者間のホップ数も全受信者で等しい単純なツリー構造であり，このトポロジにおける結果のみでは，提案方式の有効性が十分に検証できたと結論づけることはできないと考えられる．

提案方式は RM 通信において，正確な RTT を階層的に見積もる方式であり，RTT に影響を与える要因は，リンク遅延時間，ホップ数，輻輳等がある．このため，輻輳要因が存在し，送受信者間のホップ数や RTT が，受信者ごとに異なる比較的大規模なネットワーク環境において，提案方式の有効性を示すことができれば，ある程度複雑な環境においても提案方式が有効であると結論づけることができると考えられる．このため，実際のインターネットほど複雑ではないが，比較的大規模であり，帯域や遅延時間が異なるリンクから構成され，送信者から受信者へのホップ数や遅延時間が異なるネットワーク環境として，**図 9** に示すトポロジを使用し，提案方式の有効性を検証するためのシミュレーションを行った．

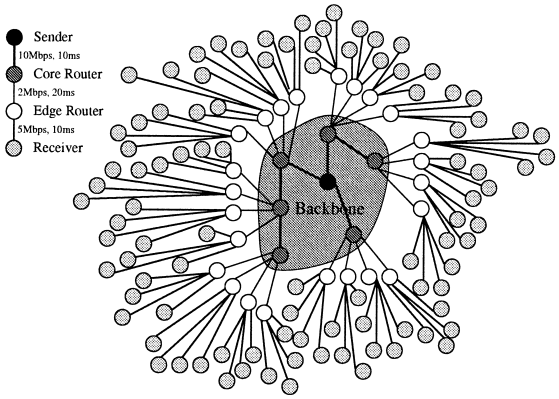

図 9 シミュレーショントポロジ 2
Fig. 9 Simulation topology 2.

送信者と直接および間接的に接続された 6 つの Core ルータがバックボーンネットワークに存在し，各 Core ルータに 4 つの Edge ルータが接続しており，各 Edge ルータには 4 受信者が接続している構成で，末端ノードはすべて受信者であり，合計 96 の受信者が送信者からのマルチキャストパケットを受信するとした．また，リンク帯域，遅延は送信者と Core ルータ間，および Core ルータどうしは 10 Mbps，10 ms とし，ボトルネックと想定される Core ルータと Edge ルータ間では，それぞれ 2 Mbps，20 ms，Edge ルータと受信者間はそれぞれ 5 Mbps，10 ms とした．シミュレーション時間は 50 秒間とし，その間ランダムにボトルネックリンクに輻輳を発生させるため，外乱として，1.5 Mbps の UDP トラヒックを各 Core ルータと Edge ルータ間のリンクごとに独立に，On/Off がそれぞれ平均 2 秒周期となる指数分布に従うように発生させた．なお，その他のパラメータについては，最初のシミュレーションと同様に表 4 のパラメータを用いてシミュレーションを行った．また，従来方式として同様に AER/NCA の RTT 見積方式を使用し，外乱トラヒックの平均時間が 2 秒であり，それを検出可能であり，かつネットワークに不要な負荷をなるだけ与えないようにするため，RTT 見積間隔は 1 秒とした．**表 6** にシミュレーション結果を示す．

この結果では提案方式において，従来方式より少しスループットが悪くなっているが，前述のように，スループットは輻輳制御機構に依存しており，この差は RTT 見積りによる影響ではない．その他の結果についても，ほとんど差が現れていない．しかしながら，

表 6 シミュレーション結果 2
Table 6 Simulation results 2.

	従来方式（見積間隔 1 秒）	提案方式
平均スループット [kbps]	456.3	435.3
総 NACK 数	27,164	18,323
送信 NACK 数/ロス数	1.33	1.29
不要パケット数/ロス数	0.108	0.075
正規化再送遅延時間	3.179	2.911
RTT 見積パケット総数	23,892	786

NACK 送信量に関しては，提案方式を用いることにより，約 2/3 に抑えることができていることが分かる．NACK 数を抑えることにより，送信者やルータに届く NACK 量を減らすことができるため，NACK Implosion を回避し，同時に配信可能な受信者数を増加させることができる．したがって，この結果から提案方式により RM 通信のスケーラビリティを高くすることが可能となる．

なお，本シミュレーションでは RM プロトコルとして AER/NCA を使用したが，PGM 等でも本提案方式を使用することができる．

6. ま と め

本論文では，NACK-based な高信頼マルチキャストにおいて，重要なパラメータの 1 つであるノード間の RTT を，必要な時点で正確に見積もるために，NACK，NACK 確認，再送データを用いて階層的に RTT を見積もる方式を提案した．提案方式は，階層的見積りであるため，NACK を送信しない受信者も輻輳リンクの RTT を知ることができ，RTT を必要とする輻輳時に正確な RTT を素早く見積もることが可能である．また，輻輳時に受信者から送信される NACK をトリガとして見積りを行っているため，ネットワークに余分な負荷をかけることがないうえ，輻輳が発生し RTT が高くなっている受信者は自動的に RTT 見積頻度が高くなり，従来方式のように，受信者数や輻輳頻度に応じて適切な RTT 見積間隔を手動で調整する必要がない．

ネットワークシミュレータを使用して計算機シミュレーションを行い，提案方式を用いることにより，輻輳時に不要なネットワーク負荷をかけずに，正確な RTT を見積もることが可能であることを示した．また，従来方式と比較し，提案方式を用いることで，無駄な NACK 送信や無駄な再送データの配信を抑えることができ，スケーラビリティを向上させることが可能であることを示した．また，提案方式では，輻輳が発生し RTT 見積りが必要である受信者ほど，自動的に RTT 見積頻度が高くなることを示した．

提案方式は，受信者が自身の状態によって，自動的に RTT 見積頻度を調整するため，インターネットのように帯域等が非常に不均質な受信者が存在する環境に適した方式であると考えられる．このため，今後の検討課題としては，そのような非常に不均質な環境での適用性の検証ということが第 1 にあげられる．また，実際のネットワークに本方式を適用した場合の，機能付加や処理コストと，性能とのトレードオフについての検討も行う必要があると考えられる．

参 考 文 献

1) Mankin, A., et al.: IETF Criteria for Evaluating Reliable Multicast Transport and Application Protocols, IETF RFC2357 (Jun. 1998).
2) Handly, M., et al.: The Reliable Multicast Design Space for Bulk Data Transfer, IETF RFC2887 (Aug. 2000).
3) Whetten, B., et al.: Reliable Multicast Transport Building Blocks for One-to-Many Bulk-Data Transfer, IETF RFC3048 (Jan. 2001).
4) Speakman, T., et al.: PGM Reliable Transport Protocol Specification, IETF RFC3208 (Dec. 2001).
5) Kermode, R., et al.: Author Guidelines for Reliable Multicast Transport (RMT) Building Blocks and Protocol Instantiation documents, IETF RFC3269 (Apr. 2002).
6) 木下真吾：リライアブルマルチキャストの現状と最新動向，信学技報，IN2000-20 (May 2000).
7) Adamson, B., et al.: NACK-Oriented Reliable Multicast (NORM) Protocol Building Blocks, IETF Internet Draft, draft-ietf-rmt-bb-norm-03.txt, Work in Progress (Nov. 2001).
8) Floyd, S., et al.: A Reliable Multicast Framework for Light-Weight Sessioins and Application Level Framing, *IEEE/ACM Trans. Networking*, Vol.5, No.6, pp.784–367 (Dec. 1997).
9) Lehman, L.H., et al.: Active Reliable Multicast, *IEEE INFOCOM '98*, pp.581–589 (Apr. 1998).
10) 山口 誠ほか：Active Network 技術を適用した信頼性マルチキャストプロトコルの性能評価，電子情報通信学会論文誌，Vol.J-84B, No.3, pp.334–343 (Mar. 2001).
11) Kasera, S.K., et al.: Scalable Fair Reliable Multicast Using Active Services, *IEEE Network*, pp.48–57 (Jan./Feb. 2000).
12) Widmer J., et al.: TCP-Friendly Multicast Congestion Control (TFMCC), IETF Internet Draft, draft-ietf-rmt-bb-tfmcc-00.txt, Work in Progress (Nov. 2001).

13) Rizzo, L.: pgmcc: A TCP-friendly single-rate multicast congestion control scheme, *ACM SIGCOMM '00*, pp.17–28 (Aug. 2000).
14) Aamson, B., et al.: NACK-Oriented Reliable Multicast Protocol (NORM), IETF Internet Draft, draft-ietf-rmt-pi-norm-04.txt, Work in Progress (Mar. 2002).
15) Padhye, J., et al.: Modeling TCP Throughput: A Simple Model and its Empirical Validation, *ACM SIGCOMM '98* (Sep. 1998).
16) 森谷優貴ほか：信頼性マルチキャストにおける輻輳制御の検討，信学技報，IN2001-46 (Jul. 2001).
17) Sharma, P., et al.: Scalable Session Messages in SRM using Self-configuration, USC Technical Report (Jul. 1998).
18) Ozdemir, V., et al.: Scalable, Low-Overhead Network Delay Estimation, *IEEE INFOCOM '00*, pp.1343–1350 (Mar. 2000).
19) Whetten, B., et al.: An Overview of Reliable Multicast Transport Protocol II, *IEEE Network*, pp.37–47 (Jan./Feb. 2000).

(平成 14 年 6 月 21 日受付)
(平成 14 年 12 月 3 日採録)

森谷 優貴

昭和 51 年生．平成 10 年京都大学工学部電子工学科卒業．平成 12 年同大学大学院情報学研究科通信情報システム専攻修士課程修了．同年（株）NTT ドコモ入社．現在,（株）NTT ドコモマルチメディア研究所勤務．高信頼マルチキャスト，無線マルチメディア通信の研究開発に従事．電子情報通信学会会員．

渥美 幸雄（正会員）

昭和 27 年生．昭和 50 年慶應義塾大学工学部電子工学科卒業．昭和 52 年同大学大学院修士課程修了．同年電電公社（現 NTT）横須賀電気通信研究所入社．主に通信プロトコル，通信制御ソフトウェアの研究開発に従事．平成 6 年（株）超高速ネットワーク・コンピュータ技術研究所．プロトコルアーキテクチャの研究に従事．平成 11 年より（株）NTT ドコモマルチメディア研究所勤務．次世代のモバイルインターネット方式の研究開発に従事．電子情報通信学会会員．博士（情報工学）．