社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2004－ＳＬＤＭ－114　(16)
2004／3／19

レスポンシブリンクを用いた実時間マルチキャスト機構

橘内　和也†　　山崎　信行†

† 慶應義塾大学大学院 理工学研究科
〒 223–8522 横浜市港北区日吉 3–14–1
E-mail: †{kitsunai,yamasaki}@ny.ics.keio.ac.jp

あらまし　複数のノードが協調しながら処理を行なう分散実時間システムでは，単一のノードが特定のグループに対してデータ送信を行なうマルチキャスト型の通信が必要となる場面が多く存在する．本研究では，並列分散制御用の実時間通信規格であるレスポンシブリンク上において受信側主導で資源予約を行なうマルチキャスト機構を実装した．受信側主導で行なうコネクション確立の際に，送信ノードに至るまでに存在する中継ノードを利用してコネクションを分岐させることで，資源予約の効率性を潜在的に高めることが可能である．さらに通信の実時間性に関しての基礎評価を行なった結果，トポロジに関わらず非常に低遅延かつジッタの少ない通信がマルチキャストにおいても可能であることがわかった．

キーワード　分散実時間システム，マルチキャスト，レスポンシブリンク

Developing Real-Time Multicasting Machanism on *Responsive Link*

Kazuya KITSUNAI† and Nobuyuki YAMASAKI†

† Keio University
3–14–1 Hiyoshi, Kouhoku-ku, Yokohama, Kanagawa 223–8522 Japan
E-mail: †{kitsunai,yamasaki}@ny.ics.keio.ac.jp

Abstract　Hard Real-Time Multicasting capability has emerged in distributed real-time systems, in which multiple nodes cooperate and process many system tasks together in real-time. We're now developing the real-time communication middleware called Real-Time Channel Manager(RCM) on *RT-Frontier* also developed in our laboratory. In this paper, we extends it with real-time multicasting capability on *Responsive Link* which is a real-time communication standard for parallel/distributed real-time processing. And also, evaluated some overhead and communication jitter. We believe that our multicasting mechanism satisfies many various QoS requirement for real-time applications in real world.

Key words　Distributed Real-Time System, Multicasting, Responsive Link

1. はじめに

複数のノードでネットワークを構成し並列/分散処理を行なう分散実時間システムにおいては，厳しい時間制約を持つハードリアルタイム通信と，比較的緩やかな制約のソフトリアルタイム通信が混在し，それらを満足する通信機構が必要となる．単一のノードが複数のノードに対して同一のデータを一括に送信するマルチキャストでは，主にマルチメディアコンテンツの配信などにターゲットが置かれ，これまでに IP Multicast や Application Level Multicast (ALM) などが提案されている．近年，より研究が盛んな ALM ではアプリケーションレイヤでグループ管理やルーティングを行なうことで，ユニキャストの機構のみでマルチキャストを実現することができるため，IP Multicast に対して，既存のネットワーク構成に手を加えることなくマルチキャストが実現できるという利点がある．しかし，ユニキャストを元に行なわれるため，場合によっては IP Multicast に比べて冗長なルーティングが行なわれてしまうという問題があり，それらに対処すべきプロトコルがいくつか開発されている．[1]～[6]

従来のマルチキャスト機構では，非常に精度の求められるセンサデータのようなハードリアルタイム通信を想定することはまれである．決められた一定の通信周期を持つようなハードリアルタイム通信の実時間性を保証するには，通信パスにおける資源予約を行なう手法が一般的である．そこで，ハードリアルタイム通信においてマルチキャストを行なうためには各受信ノードまでの通信パスの間で資源予約をする必要がある．

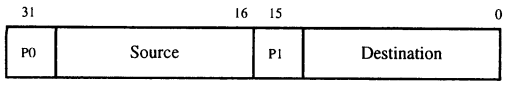


図 1　レスポンシブリンクのルーティングヘッダ

表 1　ルーティングテーブルの属性ビット

EE	イベントリンク有効ビット
DE	データテーブル有効ビット
PE	優先度付け替え有効ビット
P1, P0	PE ビットが有効なときの付け替え後の優先度
L0	パケットを CPU に取りこむ
L1, L2, L3, L4	リンクの 0〜3 に対応．各リンクに出力

本研究では通信パスにおける資源予約を前提としたマルチキャスト機構の実装を行い，通信の実時間性に関わる基礎評価を示す．

以下では，まず 2 章で並列分散処理用実時間通信規格であるレスポンシブリンクについて述べる．続く 3 章では当研究室で開発中の分散実時間 OS である RT-Frontier の通信機構について紹介する．さらに 4 章で本機構の設計方針を述べ，5 章では現段階の実装状況を，6 章では評価について述べる．

2. レスポンシブリンクの概要

レスポンシブリンク [7] は，情報処理学会の試行標準化 ISO/IEC での標準化作業が行なわれている，並列分散制御用のリアルタイム通信規格である．レスポンシブリンクでは，レイテンシとスループットのトレードオフのためにソフトリアルタイム用のデータ (データリンク) とハードリアルタイム用のイベント (イベントリンク) が分離されているという特徴がある．また，パケットの衝突による予測性の低下を避けるために結合形態は Point-to-Point となっている．さらにパケットに優先度を付加することができ，通信ノード内でのパケットの追い越し機能や優先度別のルーティング，ノード毎での優先度の付け替え機能といった特徴を持つ．これらの特徴によって実時間通信を分散管理することが可能となる．

レスポンシブリンクのルーティングテーブルは 4 レベルの優先度 2 ビットを含む 32 ビットで構成され図 1 で示す形となる．また，その属性ビットとして表 1 が設定可能である．

レスポンシブリンクでは，優先度によるパケット制御を行うが，低優先度のパケットほどそのバンド幅に対する影響が大きくなる．そこで，低優先度の通信においてもある一定のバンド幅を保証できるようにネットワーク資源を管理するソフトウェア機構も同時に必要となると考えられる．

3. RT-Frontier の通信機構

本研究室で開発している分散実時間オペレーティングシステムの RT-Frontier はマイクロカーネル型のサーバベースドアーキテクチャで設計されている．その通信機構はレスポンシブリンク上に設計されており，基本的には送信側のノードとスレッド，受信側のノードとスレッド，さらに優先度を組み合わせて

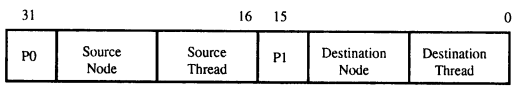


図 2　RT-Frontier のユニキャスト用ヘッダ

指定することで図 2 の形でヘッダを形成し，レスポンシブリンクのヘッダとして用いる．このため，通常のユニキャストを行なうためには，送受信ノード間でそれぞれの ID を何らかの方法で伝え，ルーティングテーブルを送受信ノード両方において設定する必要がある．

RT-Frontier ではさらに，リモートからの要求を送受信するためのヘッダと，後述するリアルタイムチャネル機構に用いるヘッダが，通常のユニキャスト用のヘッダと分離されている．リモートからの要求を送受信するためのヘッダはシステム内のどのノードからも要求を受けつけることができるように，あらかじめ起動時に各ノードで受信するように設定が行なわれており，送信側は受信側のノード ID とスレッド ID がわかれば送信することが可能である．そのため，主に各ノードに常駐するサービススレッドに要求を送信するために用いられている．

RT-Frontier では，レスポンシブリンクのデータリンクとイベントリンクをそれぞれ管理するために，データリンクマネージャーとイベントリンクマネージャーがカーネルスレッドとして存在する．Responsive Processor ではパケットの送受信は Dual Port Memory(DPM) を用いて行なわれるため，DPM が他のメッセージで上書きされる前に各スレッドが保持する受信バッファに到着したメッセージを退避するといった処理を行なう役割を持つ．

3.1 リアルタイムチャネル機構

リアルタイムチャネル [8] とは Ferrari らによって提案された，実時間通信に必要な資源予約を伴なう片方向の通信路のことである．RT-Frontier においても，実時間性の高い通信路を提供するために通信ミドルウェアとしてその機構が実装されている [9]．RT-Frontier のリアルタイムチャネル機構ではバンド幅，許容遅延，許容ジッタを受信側で指定することでコネクションの確立を行なう．実装されたミドルウェアとして，リアルタイムチャネルマネージャー (以下 RCM) がカーネルスレッドとして各ノードに常駐する．RCM はユーザからの要求を受けて，要求を処理するエージェントスレッドを生成する．生成されたエージェントスレッドはユーザが指定した QoS パラメータに従って，CPU 資源の予約，ネットワーク資源の予約，ルーティングを受信側から送信ノードに向かって 2 パスで行なう．その様子を図 3 に示す．

現在リアルタイムチャネルを利用するためのユーザに対するインタフェースとして API が定義されている．以下の 3 つはリアルタイムチャネルの確立・切断に必要となるものである．リアルタイムチャネルの確立・切断は受信側から行なわれるため，送信側でそれを検知するための接続待機ルーチンも定義されている．

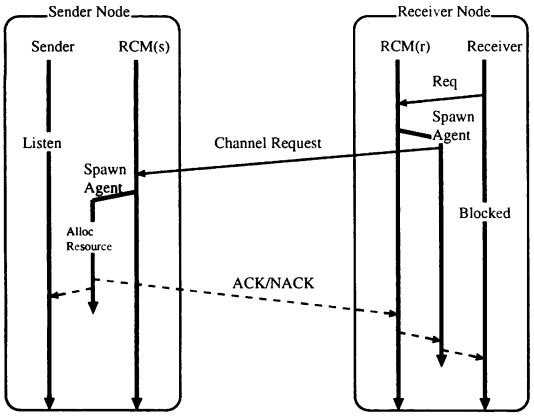
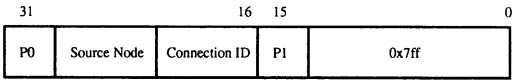


図 3　リアルタイムチャネルの確立手順



P0,P1: Priority

図 4　リアルタイムチャネルヘッダ

```
/* リアルタイムチャネル確立要求 (受信側)*/
int rtc_establish_req(nodeid_t src_nid,
                      thid_t, src_thid,
                      struct qos_attr qos)

/* リアルタイムチャネル切断要求 (受信側)*/
int rtc_teardown(hdr_t rtc_hdr)

/* リアルタイムチャネル接続待機 (送信側)*/
void rtc_listen(struct rtc_attr *attr)
```

4. 設計方針

本研究のマルチキャスト機構は前述のリアルタイムチャネルマネージャーを拡張することで実装する．以降ではマルチキャストにおいて問題となる部分について述べる．

4.1 グループアドレスの割り当て

マルチキャストのグループアドレスとして，RT-Frontier で予約されているリアルタイムチャネル用のヘッダ (図 4) を用いる．このヘッダは，リアルタイムチャネル確立要求時に送信ノード側で，送信ノード ID と送信ノード内で一意に割り当てられるコネクション ID を組み合わせて生成され，送信スレッドはそのヘッダを用いて受信グループに対してマルチキャストを行なうことが可能となる．

4.2 コネクション確立・切断

受信グループの形成は，各受信ノードがその送信ノードに対してリアルタイムチャネル確立要求を出すことによって，暗黙のうちに形成されることになる．つまり，複数の受信ノードが出すリアルタイムチャネル確立要求によってできた複数のパスによって，マルチキャストパスが形成され受信グループが形成されることになる．

同じグループアドレスで受信しようとするスレッドはリアルタイムチャネルの確立において同じヘッダを用いるため，送信ノードまでのパスを共有する場面がしばしば存在する．同一のリアルタイムチャネルヘッダを通信パス確立までの中間ノードで発見した場合に既存のコネクションからフローを分岐させることによりマルチキャストを実現する．

フローの分岐とはハードウェア的にはレスポンシブリンクのルーティングテーブルのリンクビット ($L0$〜$L4$) を変更すればよい．RT-Frontier ではリンクが接続されている先の隣接ノードの ID を識別することが可能なので，対応するリンクビットを有効にすればそのリンクにもパケットが複製されて出力されることになる．$L0$ ビットを有効にすれば，自ノードに取りこむことも可能であるため，今までリアルタイムチャネルの中継ノードでしかなかったノードからも取りこむことが可能である．この際リモートノードへリクエストパケットを送信する必要がないため，受信グループへの参加は非常に低いオーバヘッドで実現できる．

4.3 資源予約・解放

中間ノードでのフローの分岐が可能であるために，リアルタイムチャネルの共有が可能となり，資源予約を効率化できる．通常のユニキャスト用途のリアルタイムチャネルでは受信ノードから送信ノードに至るまで全てのパスにおいて資源予約を行なうが (図 5)，マルチキャストを考慮した本機構の場合 (図 6) にはすでに存在する通信パス間の資源予約を行なう必要がないためすでに存在するパスの資源予約を省略できる．マルチキャストではしばしば単一の送信ノードが非常に多くの受信クライアントをかかえる状況が多く，送信ノードの周りのネットワーク資源は枯渇しがちとなるため，この利点は大きい．

また，資源解放においても，まだ他のノードで受信が行なわれている場合があるため，分岐点となる中間ノードでは分岐しているパスを記録しておき，実際の下流のパスが全て解放された後に上流のパスの資源解放を行なう必要がある．

問題点としては，完全に中継ノードとなったノードは，他ノードによって確立されたリアルタイムチャネルのために，自ノードの周りのネットワーク資源が予約されてしまうことにある．現在の実装ではこの問題に対処できていないが，今後の方針として，ネットワーク資源予約のアドミッションコントロールを残余バンド幅や許容遅延だけでなく，各ノードごとにその周りのネットワーク資源予約の可否を設定できるようにし，それを考慮に入れたアドミッションコントロールを行なっていきたいと考えている．

4.4 グループ内での異なる QoS 要求

動画像フレームのマルチキャストでは，受信側がその処理能力によってそのフレームレートを調整したい場面が存在する．図 7 のように同じコンテンツを受信するクライアント同士で異なる QoS 要求を実現するためには，ハードウェアによるスイッチングだけでフローを分岐してしまうと，各通信パスで同じパ

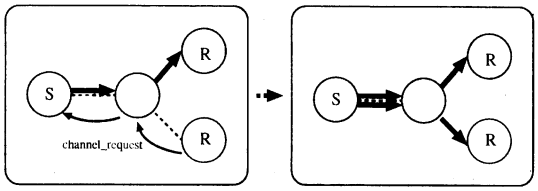


図 5　ユニキャストでのフローの生成

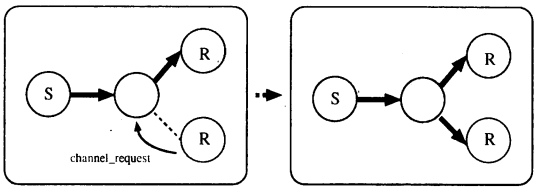


図 6　中間ノードからのフローの分岐

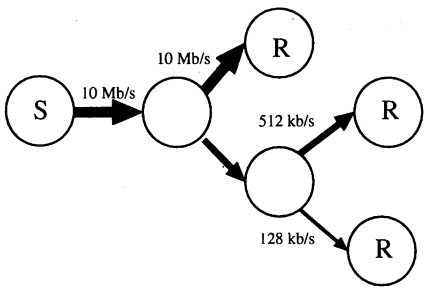


図 7　グループ内での異なる QoS 要求

ンド幅が必要となるため望ましくない．

このような異なる QoS 要求を解消する手法として，中継ノードでいったん送信データを取りこんで各クライアントが要求する QoS に従って再度送信を行なうプロキシサーバを用意する方法と，フローを複数に分割し受信するフローの数によってバンド幅をコントロールする Receiver-Driven Layered Multicast (RLM) [10] が考えられる．

プロキシサーバを用いて QoS をコントロールする方法では送信を行なうスレッドの実行周期と受信側のデータリンクマネージャの実行周期のタイミングによって送受信の通信遅延が大きく変動してしまう問題がある [11]．

一方，RLM を本機構で実現するには，同一スレッドに対して複数のリアルタイムチャネルを確立すればよいのでそのまま実現可能である．しかし，各フローが異なるパスを経由することが考えられるため，受信側か中継ノード側で順序制御が必要になると考えられ，この手法については現在検討中である．

5. 実装状況

本機構は 3.1 節で述べたリアルタイムチャネルマネージャー上に実装を行なっている．現在は，コネクション確立/切断の際に中継ノードから分岐を行なう機構が実装されている．また，ユーザに対して提供されている API は本機構が組みこまれた後も変更なしで利用可能である．送信ノードまでのルーティングに関しては，現段階では各ノードへ静的にルーティングされた最短パスを用いて行なわれており，ルーティングの部分については資源効率，耐故障性などの面を考慮して検討中である．

既存のリアルタイムチャネル機構から変更を加えた個所は以下に示す．

- リアルタイムチャネルを確立する際に各中継ノードにおいて要求するコネクションがすでに確立されていないかを確認する
- リアルタイムチャネル切断の際にも各中継ノードは切断するコネクションを使用しているクライアントがいないかを確認する

これらの変更はミドルウェア側の変更であるため，ユーザ側から見える部分の変更点はない．本機構のコネクション確立要求は，従来通り受信側主導で行なわれるため，リアルタイムチャネル用ヘッダと QoS パラメータを指定する従来のユニキャストのコネクション確立と同じである．ミドルウェア側でソースノードまでの通信パスをたどる中で既存のフローを検出した場合に，そこから受信側に折り返して資源予約を行ないパスを確立する．

6. 評価及び考察

6.1 評価環境

評価は，*Responsive Processor* PCI 評価ボード [12] を用いて行なった．Responsive Processor ではプロセッシングコアである SPARClite の動作速度及びバスクロックの速度を動的に変更することが可能である．そこで，評価に際しては，表 2 に示す設定で行なった．

表 2　Responsive Processor に関する設定

SPARClite の動作速度	$120MHz$
バスクロックの速度	$40MHz$
Responsive Link の通信速度	$40MBaud$

6.2 測定方法

オーバヘッドの測定には Responsive Processor の内部タイマを用いて行なった．このタイマは $0.05\mu sec$ ごとにカウントダウンしていく 16bit のタイマである．通信遅延については，Responsive Processor の PIO をトリガとしてロジックアナライザを用いて測定した．

6.3 評価に用いたトポロジ

評価には図 8 に示す 2 種類のトポロジで測定を行なった．ホップ数を変化させる場合には直列型の，同ホップ数での通信遅延を測定するときにはスター型のトポロジを使用した．

6.4 コネクション確立までのオーバヘッド

まず本機構の基礎評価として，マルチキャスト型のコネクション確立にかかるオーバヘッドを測定した．測定の際にはコ

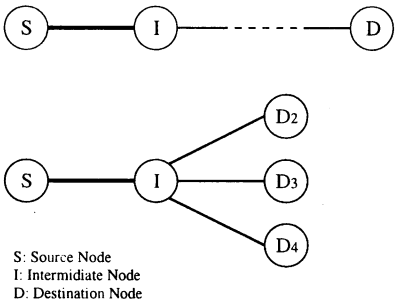


図 8　評価に用いたトポロジ

表 3　コネクション確立までのオーバヘッド

ホップ数	最小値 [μsec]	最大値 [μsec]	平均値 [μsec]
0	176	178	177
1	625	627	626
2	1048	1054	1050
3	1456	1461	1458
4	1870	1876	1872

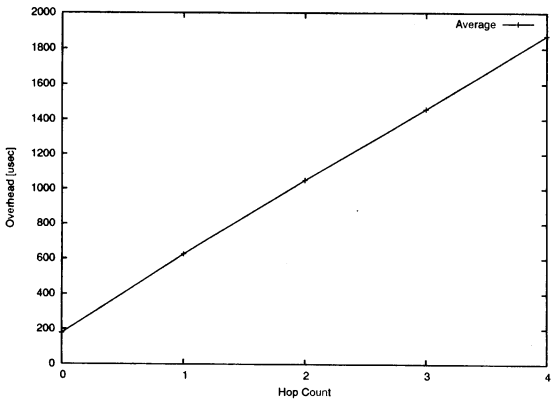


図 9　オーバヘッドの平均値

ネクション確立を連続して 10 回行ない，オーバヘッドについてその最大，最小，平均値を求めた．

RCM がチャネル確立のために生成するエージェントスレッドは非周期スレッドであり優先度が相対的に低くノード内の負荷が高いときには実行に影響が生じてしまうため，今回は負荷が無い状況でのオーバヘッドを測定した．

また，評価に利用したトポロジは図 8 における直列に接続されたもので，送信ノードの隣に中継ノードを置き，マルチキャストの場合には実際はここからフローの分岐が行われるようにした．

表 3 では中継ノードまでの各ホップ数によるコネクション確立オーバヘッドを示している．マルチキャストを行なう本機構の場合はすでにフローが受信ノードを通っている場合 (表 3 の 0 ホップ) があるため，リモートノードとの通信を行なうことなく既存のフローを取りこむことが可能な場合がある．コネクション確立は前述のように 2 回のパスで行なわれるため，そ

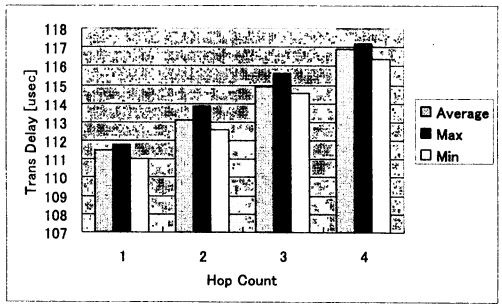


図 10　64 バイトデータの通信遅延

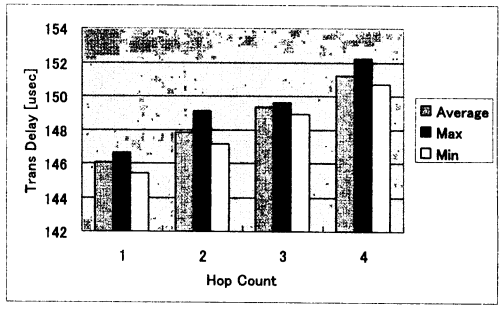


図 11　128 バイトデータの通信遅延

のオーバヘッドは要求パケットの通信遅延と，資源管理などのノード内処理の実行時間となり，図 9 の結果のようにおおよそホップ数に比例することになる．

6.5　ホップ数と通信遅延に関する評価

次に，構築されたマルチキャストツリーにおける通信遅延を測定し，本機構のマルチキャスト通信の実時間性能の評価を行なった．使用したトポロジは直列型である．送信データサイズは，64 バイトと 128 バイトで測定を行なった．

ハードウェアによるスイッチングによってマルチキャストを実現しているため，ジッタが非常に小さいことがわかる．64 バイトのデータでは 1μsec 以内，128 バイトのデータでは 2μsec 以内のジッタで抑えることが可能であることがわかった．

6.6　同階層の受信ノード間の通信遅延に関する評価

最後に図 8 のスター型のようにマルチキャストツリーにおける同一階層のノード D_2, D_3, D_4 の間での通信遅延の差について測定を行なった．測定には先ほどと同じく 64 バイト，128 バイトのメッセージを用いて行なった変動の様子を見るために 10 回の送受信時の各ノードの遅延を測定し，変動の様子をグラフにしたものを図 12 と図 13 に示す．図 8 の受信ノードと送信ノードとの距離は全て 2 ホップであるため，通信データが到着するタイミングは同じであると考えられる．

図 12 と図 13 から，各ノード間の通信遅延の差は 64 バイトのデータでは約 1μsec 以内，128 バイトのデータではおおむね 2μsec 以内であることがわかった．

以上のように，直列型，スター型においてもマルチキャストによる通信遅延のジッタは非常に小さいことが確認できた．こ

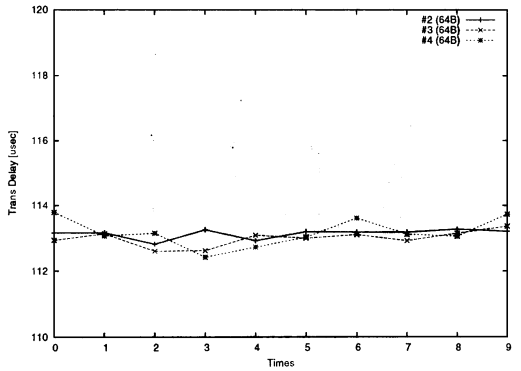


図 12　64 バイトデータでの通信遅延の変動

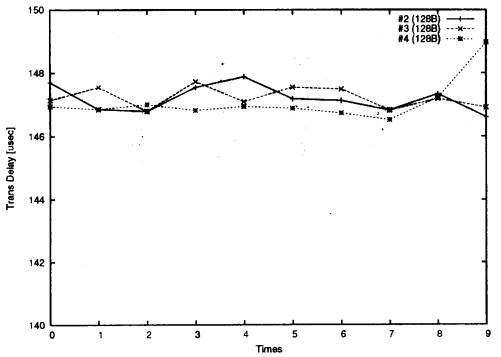


図 13　128 バイトデータでの通信遅延の変動

の性質をもとに，今後は厳しい時間制約を持つようなハードリアルタイム通信でのマルチキャスト機構についてさらに検討を重ねていきたい．

7. 終わりに

本研究では，多種多様な QoS 要求が混在する分散実時間システムのための実時間マルチキャスト機構の実装及び基礎評価を示した．資源予約を伴なうマルチキャストにおいて，受信側からコネクションを確立し，中継ノードからの分岐を行なうことによって資源予約の効率性を保ち，コネクション確立までのオーバヘッドを潜在的に削減することができることを示した．

本機構では扱わなかったルーティング，トラヒック制御や異なる QoS 要求を扱うためのセッション管理といった問題についても今後は取り組んでいきたいと考えている．

謝辞　本論文の研究は，文部科学省の科学技術振興調整費の支援による．また本研究の一部は，科学技術振興機構 CREST の支援による．

文　献

[1] Y. hua Chu, S. G. Rao and H. Zhang: "A Case for End System Multicast", Proceedings of the 2000 ACM SIGMETRICS international conference on Measurement and modeling of computer systems, pp. 1–12 (2000).

[2] J. Jannotti, D. K. Gifford, K. L. Johnson, M. F. Kaashoek and J. James W. O'Toole: "Overcast: Reliable Multicasting with an Overlay Network", Proceedings of USENIX Symposium on Operating Systems Design and Implementation (OSDI), pp. 197–212 (2000).

[3] D. A. Helder and S. Jamin: "Banana tree protocol, an end-host multicast protocol", Technical report, TR-429-00 (2000).

[4] D. Pendarakis, S. Shi, D. Verma and M. Waldvogel: "ALMI: An application level multicast infrastructure", 3rd USNIX Symposium on Internet Technologies and Systems (USITS '01), San Francisco, CA, USA, pp. 49–60 (2001).

[5] D. A. Helder and S. Jamin: "End-host multicast communication using switch-trees protocols", Proceedings of the Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems (GP2PC) (2002).

[6] N. Mimura, K. Nakauchi, H. Morikawa and T. Aoyama: "RelayCast: A Middleware for Application-level Multicast Services", Proceedings of 3rd International Workshop on Global and Peer-to-Peer Computing on Large Scale Distributed Systems(GP2PC 2003), pp. 434–441 (2003).

[7] "Responsive Link (IPSJ-TS 0006:2003)", http://www.itscj.ipsj.or.jp/ipsj-ts/02-06/toc.htm.

[8] D. Ferrari and D. C. Verma: "A Scheme for Real-Time Channel Establishment in Wide-Area Networks", IEEE Journal on Selected Areas in Communications, **8**, 3, pp. 368–379 (1990).

[9] K. Katoh, H. Kobayashi, N. Yamasaki and Y. Anzai: "Elastic network management for flexible and efficient real-time communication", Proceedings of 18th International Conference on Computers and Their Applications, pp. 80–83 (2003).

[10] S. McCanne, V. Jacobson and M. Vetterli: "Receiver-driven layered multicast", Proceedings of ACM SIGCOMM, pp. 117–130 (1996).

[11] K. Kitsunai, H. Kobayashi, N. Yamasaki and Y. Anzai: "Extensible real-time data dissemination on channel-based reflective memory", Proceedings of the 18th International Conference on Computers and Their Applications, pp. 236–239 (2003).

[12] N. Yamasaki: "Design and Implementation of Responsive Processor for Parallel/Distributed Control and Its Development Environments", Journal of Robotics and Mechatronics, **13**, 2, pp. 125–133 (2001).