

インターネットでの並列分散処理の実装検討

古賀久志、下國 治、新家正総、河合 純、
小林伸治、水野裕識、陣崎 明

新情報処理開発機構 並列分散システム富士通研究室
〒211-8588 川崎市中原区上小田中 4-1-1

並列計算にはバリア同期、DSM におけるページ共有などマルチキャストを使うと効率の良い処理が多いが、広域並列分散環境で従来の IP multicast を使うのは信頼性の点で問題がある。最近、信頼性のあるマルチキャストとして reliable multicast 技術が提案されており、それを使用する事を検討する。

本論文では、まず並列分散処理向けの reliable multicast の持つべき性質を紹介し、それをベースに実際のなプロトコルを設計する。このプロトコルでは ACK パケットの統合、無駄な再送の抑制を主にルータでのテーブル管理、探索で実現する。本プロトコルを我々が開発中のネットワークサーバシステム Comet で動作させると、1 パケットあたり 200ns の遅延で処理できるという予測が得られた。

広域ネットワーク、マルチキャスト、リライアブルマルチキャスト

An Implementation Design on the Internet Protocol for Parallel and Distributed Processing

Hisashi Koga, Osamu Shimokuni, Tadafusa Niinomi, Jun Kawai,
Shinji Kobayashi, Hironori Mizuno, Akira Jinzaki,

Parallel and Distributed Systems Fujitsu Laboratory, RWCP
1-1, Kamikodanaka 4-Chome, Nakahara-ku, Kawasaki, 211-8588 Japan
E-mail: comet@flab.fujitsu.co.jp

In parallel -distributed computing, many processings such as barrier synchronization and page sharing in DSM can be handled efficiently by multicasting. However, current IP multicast does never provide reliability. Recently, a technology named "reliable multicast" is proposed and we attempt to apply it to parallel-distributed computing. In this paper, the properties which reliable multicast must have in order to operate in a parallel-distributed computing environment are mentioned first. Then, a practical protocol is designed according to the properties. This protocol reduces redundant packet retransmissions and unifies several ACK packets into one ACK packet mainly by the table management and the table search.

We obtain the estimation result that one packet will be processed within 200ns delay if the protocol is implemented on the network-server Comet which we are currently developing.

Wide-area-network, multicast, reliable multicast

1. はじめに

近年 ATM 等の高速な広域網が利用できるようになるに連れ、広域分散した並列計算環境の実現が可能になってきた。従来のせいぜい数十 m しか物理的に離れていない PC クラスタベースの並列システムでは専用通信プロトコルを使うのが普通だが、世界的に分散した並列計算環境を構築するには IP プロトコルを採用するのが必要になる。

並列計算には分散共有メモリのページ無効化命令、バリア同期命令といったマルチキャスト通信を使った方が効率のよい処理が多い。広域環境でこれを実現しようとする IP マルチキャストを使うことになる。だが、既存の IP マルチキャストではパケットの到達性を保証しないため、アプリケーションレイヤで意識して信頼性を確保しなければならず煩雑である。

最近、インターネット研究者の間で注目されてる技術として reliable multicast がある[1][4]。これは一言で述べると、TCP-like な信頼性を保証するマルチキャスト通信技術であり、この技術を使えば、並列計算アプリケーションがパケットロスの可能性を意識しなくて良くなる事が期待される。reliable multicast については、現在様々なプロトコルが提案、発表されている段階である。

我々は、様々な reliable multicast の取り得る手法の中で、どのような特徴を持てば並列分散計算に適しているかを検討し別論文[2]で発表した。数万台規模の計算機を結合する事を想定し、それに耐えるだけのスケラビリティを持たせることを目標にした。

本論文では[2]の方式を詳細化して具体的なプロトコルを設計し、実用性の評価を行う。特に、我々が開発している Comet ネットワークサーバ[3]の上で本プロトコルを実装した時の処理時間を予測する。

Comet はプロトコル処理をネットワークアダプタにオフロードすることにより高速な通信処理を実現するシステムである。reliable multicast では信頼性を保証する分ネットワークレイヤでの処理が重いので、ネットワークアダプタへの処理オフロードにより通信性能の大きな向上が期待できる。

2. 並列計算向きの reliable multicast

本節では我々が [2]で提案した並列計算処理向きの

reliable multicast が持つべき性質を述べる。

2.1. 前提条件

インターネットに接続された数万台規模の計算機でマルチキャストグループを構成して並列計算を行う。提案されるプロトコルはそれに見合うだけのスケラビリティをもつ必要がある。

2.2. ネットワークに対する仮定

Gbps 程度の速度をもつ回線で結ばれたインターネットを仮定する。ネットワークの性質は、高帯域 (数 Gbps)、高遅延 (~数十 msec)。エラー率は高くないが、信頼性を保証していないネットワークを仮定する。

2.3. reliable multicast の持つべき性質

reliable multicast では、まず一般のマルチキャスト配送機能である

- マルチキャストグループの経路設定機能
- マルチキャストパケットの配送機能

を持つ必要がある。

これに加えて reliable multicast では TCP プロトコルのような到達性を保証するため、以下の機能を提供する必要があり、ここが議論の対象となる。

- 到達確認機能
- パケットロス時の再送機能

これらの機能について我々が提案した方式を整理する。

2.3.1. 到達確認の強さ

- 到達順序保証をするかどうかをアプリケーションの要求に応じて adaptive に変更できる。

並列計算への応用であるという点を考慮すると、アプリケーションから見てパケットがシーケンス番号通りの順序で到着するかは非常に重要である。アプリケーションの制約の強さによって、到達順を保証するか否かを adaptive に選択できるのが望ましい。

2.3.2. 到達確認方式

- ACK パケットによる到着確認

到達確認は TCP と同じく ACK パケットによって行う。ACK パケットの中身は受信側ノードのバッファサイズ (window サイズ) 及び、どこまでパケットデータ列を受け取ったかを示すシーケンス番号からなる。送信側ノードは TCP のようなスライディングウィンド

ウを持つ。

- ACK パケットをルータで統合する

数万台もある receiver それぞれから sender にユニキャストで1個ずつ ACK パケットを送っていたのでは ACK パケットが爆発する。スケーラビリティを持たせる為にルータで ACK パケットを統合して負荷を減らす(図1)。

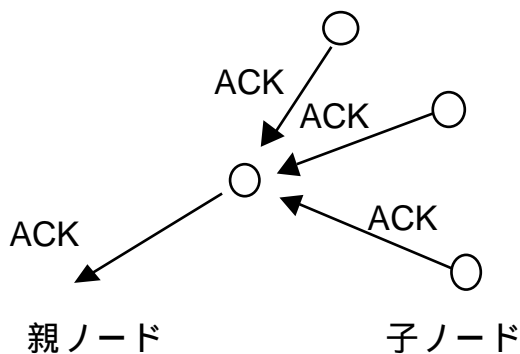


図1: ACK パケットのルータでの統合

2.3.3. 再送パケットの送信元

- パケットを再送時には送信元から再送する

パケット不到達時にどこからパケットを再送させるかについて考える。中間ノード(ルータ)にデータを保持させるのは、沢山のマルチキャストグループを扱うことを考えると非現実である。パケットを受信できた最寄りのノードから再送してもらうのはよく見られる手法であるが、インターネットのようにトポロジーが明確に定められないネットワークにおいては最寄りのノードを見付けるのが困難である。結局、パケット再送は送信元から実施するのが良い。

2.3.4. 再送範囲の制御

- ACK パケットのシーケンス番号を見て、パケットが届かなかったノードにのみ再送を実施する。

ネットワークへの負荷を減らすため届かなかったノードへのみ再送を行う。ルータに子ノードから返ってきた ACK パケットの情報を登録しておき、その情報を利用して再送範囲の制限を実施する。

3. プロトコルの詳細

この節では、第2節で議論した並列分散計算に向けた reliable multicast の性質を元に、具体的にプロトコルを設計する。マルチキャストの経路設定、グループ管理についてはここでは議論せず、信頼性(到達確認、再送)に絞って議論をすすめる。

まず、1つの IP マルチキャストグループの中では sender が1つしかいない単方向通信しか許さない。複数の方向への通信は単方向通信を複数個組み合わせて実現する。これは、単方向にすることでマルチキャストツリーの中での親子関係が自明に定まり、ACK パケットの統合、再送範囲の制御を単純に実現することを狙っている、

本稿で述べるプロトコルの大きなポイントは

- TCP のような信頼性を持たせるため新しいトランスポート層プロトコルを定義し、そのヘッダ内情報を利用して通信する。
- ACK の統合、再送範囲の制御をルータで行うためルータがセッション情報を持つ。その情報をマルチキャスト IP アドレスに対する経路表を拡張する形で定義した。

という2点である。

3.1. トランスポート層プロトコル RM

TCP のようにシーケンス番号と ACK ベースで到達確認を行うため、新しいトランスポート層プロトコル RM(Reliable Multicast)を定義した。IP ヘッダの上位プロトコルフィールドに RM プロトコルを示す値が書かれる。

RM ヘッダフォーマットを図2に示す。

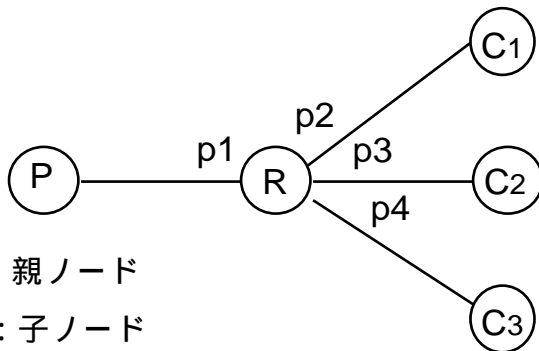
- 制御ビット(16bit) :
最上位 bit がパケットが ACK パケットかどうかを示す。その他のビットは未定義。
- マルチキャスト IP アドレスフィールド(32bit) :
後述のように本プロトコルにおける ACK パケットはマルチキャスト IP アドレスを IP ヘッダに含めない。どのマルチキャストに対する ACK パケットかを見分けるのに使う。
- シーケンス番号フィールド(32bit) :
データ送信時には単にそのデータのシーケンス番号を示し、ACK パケット内ではどこまでパケットを受け取れているかを示す。
- WINDOW フィールド(16bit) :
ACK パケット内でのみ valid であり、受信ノードでの余りバッファサイズを示す。
この他にエラーチェックの為にトレイラ - として、
- CRC コード(32bit)
を持つ。

Multicast IP アドレス	
シーケンス番号	
制御ビット	WINDOW サイズ

図 2 : RM ヘッダフォーマット

3.2. ルータの持つ情報

ルータで ACK の統合、再送制御を行うため、ルータはマルチキャスト IP アドレス毎にセッション情報を持つ。図 3のネットワークでルータ R が持つセッション情報のフォーマットを図 4に示す。



- P : 親ノード
- C_i : 子ノード
- R : ルータ
- p_i : ポート番号

図 3 : マルチキャストツリー例

図 4に示されるようにルータはマルチキャスト IP アドレスをキーとするテーブルを持ち、各エントリ内で

- 親ノードの IP アドレス。マルチキャストツリー構成時に取得して登録しておく。
- 各子ノードに対して、
 1. どこまで子ノードがパケットを受け取っているかを示すシーケンス番号
 2. 子ノードの現在の window サイズ
 を情報として持つ。

このエントリは、子ノードから ACK パケットを受け取った時のみにルータが以下の手順で更新する。

- RM ヘッダの制御ビットを見て ACK パケットであると判定。
- window サイズのエントリは ACK パケットを受け取る度に書き換え。
- シーケンス番号のエントリは、テーブル上の記録値より ACK パケット内のシーケンス番号が大きい時に更新。そうでない時は無視。

Multicast address	p1	親	親ノードの IP アドレス	
	p2	子	シーケンス番号 1	Window サイズ 1
	p3	子	シーケンス番号 2	Window サイズ 2
	p4	子	シーケンス番号 3	Window サイズ 3

図 4 : セッション情報フォーマット

3.3. ルータでの処理詳細

ルータがパケットを受け取った時の処理について説明する。ルータがパケットを受け取ると、

1. IP ヘッダの上位プロトコルフィールドを見て reliable multicast パケットであることを確認
2. RM ヘッダの制御ビットから ACK パケットかどうかを判定
3. RM ヘッダのマルチキャスト IP アドレスの値をキーに3.2で述べたテーブル検索。

データパケット（往路）であれば 3.3.1 へ進み、ACK パケットであれば3.3.2へ進む。

3.3.1. データパケット処理

1. RM ヘッダ内のシーケンス番号フィールド A と各子ポートに対応するテーブルエントリのシーケンス番号フィールド値 B を比較。
 2. A > B となったポートにのみパケットを出力する
- 2で再送範囲の制御を行っているが、ルータは単にシーケンス番号を比較するだけであり、どのシーケンスが再送状態であるかの状態を管理する必要はない。

3.3.2. ACK パケットの処理

1. 3.2で述べた手順でテーブルエントリを更新
2. 1の結果、
 - シーケンス番号エントリの min 値が増加した
 - あるいは
 - window サイズの min 値があるしきい値より小さくなった

時のみに、親ノードの IP アドレスをテーブルから取得して、ACK パケットを親ノードに返す。

この時、ACK パケットの RM ヘッダのシーケンス番号、window フィールドには、それぞれテーブルのシーケンス番号エントリ、window サイズエントリの最小値を埋め込む。さらに IP ヘッダのソースアドレス

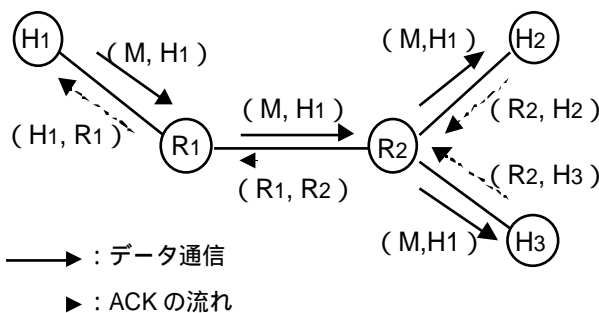
フィールドに自分の IP アドレス、宛先アドレスフィールドに親ノードの IP アドレスをセットする（図 5 の点線）

3. 親ノードに ACK を返す時は CRC を再計算する。
 図 5 で(H1, R1)という文字組は IP パケットのソースアドレスフィールドが R1 で宛先アドレスフィールドが H1 であることを示す。ACK パケットを 1 ホップ毎の IP ユニキャストで飛ばすのは、ACK パケットが往路と同じ経路を通ることを保証するためである。ACK パケットが往路と違う経路を通ると ACK を統合するのは非常に難しくなる。

また、

- 手続き 2 の 1 つ目の条件判定が ACK パケットの統合を行う。子ノード数を n とすると、平均的には子ノードからの ACK n 個に対して、親ノードへの ACK 1 回に抑制する。
- 2 つ目の条件判定は受信側のバッファに余裕がないという緊急事態に対しては、ACK をすぐ返してその情報を伝えるという事を意味する。

以上のように、ルータにおける ACK 統合は子ノードからの ACK のシーケンス番号、ウィンドウサイズのそれぞれ最小値を親ノードへの ACK の情報とする。



H_i : エンドノード
 R_i : ルータ
 M : マルチキャスト IP アドレス

図 5 : ルータによる IP アドレスフィールド変換

3.4. 通信端点の計算機での処理

送信側の計算機では通常の TCP のように以下の動作をする。

- 送信の際に ACK の応答時間管理
- ACK タイムアウト時の再送信
- エラーチェックコード (CRC) の計算
- ウィンドウサイズを基にした流量制御

受信側の計算機では、TCP と同様な

- 受信パケットの並べ替え、重複パケットのフィルタリング。エラーチェックコードの計算といった処理の他に以下の処理を行う。

- 3.3.2 で記述したようなユニキャストで ACK パケットを返す機構を実現するためのテーブル管理

4. プロトコルの評価

4.1. テーブルサイズ

我々のプロトコルでは、ACK パケットの統合、無駄な再送の抑制を、主にルータでのテーブル管理、探索で実現しており、テーブルサイズはプロトコルを評価する上での大きな要素である。

テーブルのエントリ数は、並列計算を協調して作業をしているノードの数と等しく、1 アプリケーションにつき数万エントリになる。1 エントリのサイズは子ノードの数に依存するが、図 4 から 50byte あれば十分である¹。仮に 5 万エントリあったとすると 1 アプリケーションあたりのテーブルサイズは

$$50\text{byte} \times 50000 = 2.5\text{MB}$$

くらいで大きな量ではない。但し、マルチキャスト IP アドレスが不連続だと、効率のよい探索実現が難しいので、アプリケーション毎に連続したマルチキャスト IP アドレスを使うことが望まれる。

4.2. Comet での実装

4.2.1. Comet

Comet[3]はプロトコル処理をネットワークアダプタ上のプロトコル処理エンジンで行うことにより通信の高速化を目指したシステムである（図 6）。

Comet のプロトコル処理エンジン（図 7）はプロトコルの変更に対応し、かつ複雑な処理を行うため、専用ハードウェア方式ではなく、プログラマブルハードウェアから構成される。プログラマブルハードウェアは構成メモリ（状態遷移表）を設定することによって動作を変更可能な有限状態機械である。以下に代表されるパケット処理に必要な機能ユニットを備え、パケットを伝送速度で実時間処理する。

- チェックサム計算
- CRC 計算

¹ 子が 6 個の場合、ポートに INDEX に 4 bit 使うとすると $4 + 0.5 \times 4 + 0.5 + 4 + (4 \times 2) + 6 = 46.5\text{byte}$

- テーブル検索
- パケットヘッダ解析

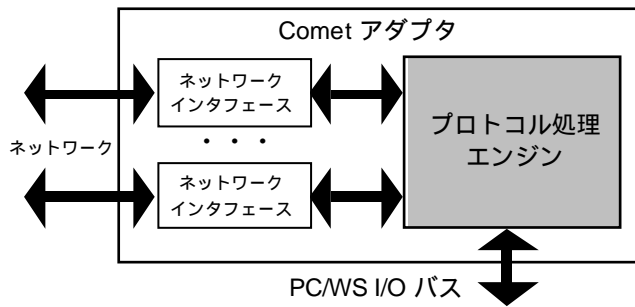


図 6: Comet ネットワークアダプタ

機能ユニットを多段接続すれば複雑な処理ができる。データメモリにはテーブルやパケットバッファが置かれる。Comet ネットワークアダプタは複数のネットワークインタフェースを搭載し、プログラマブルハードウェアのデータメモリに経路情報表を置いてルーティング処理もできる。

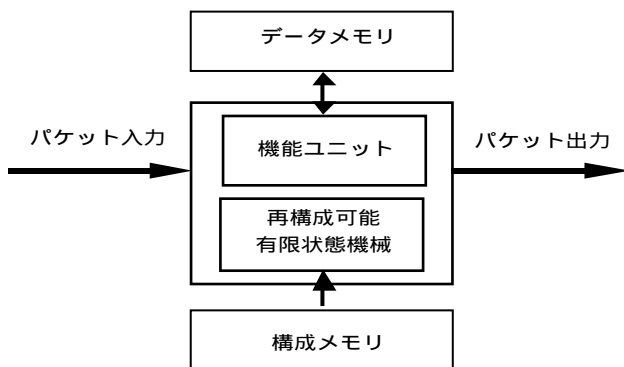


図 7: プロトコル処理エンジン

4.2.2. 処理時間

Comet で3.3節でのルータでの処理をさせたときの delay について考える。1ホップでの delay はデータパケットフォワーディング時

- RM ヘッダ解析
- IP ヘッダチェックサム (パケット受信時, 送出時)
- テーブル探索

ACK パケットフォワーディング時

- RM ヘッダ解析
- IP ヘッダチェックサム (パケット受信時, 送出時)
- テーブル探索と更新
- パケットヘッダ書き換え
- データ CRC (パケット受信時, 送出時)

の処理遅延からなる。

Comet のプロトコル処理エンジンは、パケットをオン

ザフライ処理できるので、RM ヘッダ解析とパケット受信時のエラーチェック(CRC,チェックサム)はデータ受信と並行して行える。ここで、Comet のチェックサ及びCRC 計算ユニットは 1Gbps 以上の速度を達成しネットワーク速度より早いので、1クロック分だけの遅延が発生したとして 16ns かかる(66MHz 動作時)。

ACK パケット送出時のチェックサム計算はパケット送出と並行にできないが、IP ヘッダ長は 20byte なので $20 \times 8 / 1G$ より 160ns かかる。一方、データCRC はトレーラーなので、パケット送出と並列処理可能。

テーブル検索は、ヘッダ解析してマルチキャストアドレスを取り出した後、データ受信とオーバーラップさせられる。テーブル検索を 8bit (256 エントリ) x 4 段で行うと4クロックで検索完了し、データ受信の遅延に隠蔽される。結局、200ns の遅延で1個のパケットを処理可能である。

5. まとめ

並列分散計算に適した reliable multicast が持つべき特徴[2]をベースに具体的なプロトコルを設計した。本プロトコルでは ACK パケットの統合、無駄な再送の抑制を主にルータでのテーブル管理、探索で実現する。

そして、Comet 上でプロトコルを実現した場合の処理時間の予測を行った結果、200ns の遅延で1パケットを処理できるという見積もりが得られ、プロトコルの実用性が確認できた。今後は Comet 上で実装を進め、見積もり通りの性能を出せるかを検証する。

参考文献

- [1] S. Floyd, et al.: A Reliable Multicast Framework for Light-weight Sessions and Application Level Framing, Proc. of the 1995 ACM SIGCOMM, pp. 342-356, Aug., 1995, MA.
- [2] 下國、古賀、新家、河合、小林、水野、陣崎：インターネットでの並列分散処理の実装検討、本研究会予稿, 1999
- [3] 陣崎、中村、村井：ギガビットルータ Comet のアーキテクチャとその評価、インターネットコンファレンス'98 論文集, pp. 89-96, 1998
- [4] 山内、石川、高橋：マルチキャストセキュリティにおける配送制御の効果と暗号の分担、インターネットコンファレンス '98 論文集, pp. 3-10, 1998