マルチメディア通信と分散処理 64−20
（1994. 3. 4）

協調分散処理のためのマルチキャスト機能をもつTCPの実現と性能評価

具志堅 博保 † 大西 淑雅 ‡ Bernady O. Apduhan † 末吉 敏則 †

†九州工業大学 情報工学部
‡九州工業大学 情報科学センター

概要

分散システムにおいて協調分散処理を行うには，信頼性のある効率的な通信を行う通信プロトコルが必要である．そこでその要求を満たすため，広く普及しているDARPA標準および業界標準の通信プロトコルTCPで一対多通信が可能なマルチキャスト機能を実現した．これにより協調分散処理において，ネットワーク・トラフィックの減少，プロトコル処理オーバヘッドの削減が行える．本稿では，マルチキャスト通信を行うためのグループの定義，そして信頼性を保証するために必要な主な制御および付加機能について述べる．さらに，従来のTCPとマルチキャスト機能をもつTCPを比較して得られた性能評価の結果について報告する．

Implementation and Evaluation of a TCP Multicast Facility for Cooperative Distributed Processing

Hiroyasu Gushiken † Yoshimasa Ohnishi ‡
Bernady O. Apduhan † Toshinori Sueyoshi †

†*Department of Artificial Intelligence, Kyushu Institute of Technology*
‡*Information Science Center, Kyushu Institute of Technology*

Abstract

A reliable and efficient network communication protocol is essential to execute efficient cooperative processing on a distributed environment. This paper describe the one-to-many multicast facility implementation on the standard TCP protocol in an effort to improve the network traffic and protocol processing overhead in cooperative distributed processing. Furthermore, this paper describes the definition of a group used in this study, the communication control, and the added function to guarantee reliability. Finally, the total efficiency evaluation results, i.e., the comparison of TCP Multicast with the standard TCP, is discussed.

1 はじめに

近年，計算機の高性能化および低価格化に伴い，各研究機関あるいは大学等の教育機関において，Ethernetのような伝送媒体で各計算機間を接続した分散システムが普及してきている．そのような環境の発達により，ネットワークを利用したさまざまなアプリケーション開発がなされており，また分散システムを一つの計算機とみなして処理を行う協調分散処理に関する研究も盛んに行われてきている．我々の研究室でも分散システムを利用し，分散共有メモリモデルに基づくハイパフォーマンス・コンピューティング環境，DSE(Distributed Supercomputing Environment)[1] の開発を行っている．

ここで定義する協調分散処理システムは，分散システム上で複数の計算機が全体で一つの処理を行い，各計算機が途中で故障を起さないこととする．このようなシステムでは，大量のデータを複数の計算機に送信したり，あるいは同期操作を伴う処理を行うためのデータ通信 (マルチキャスト機能) が必要不可欠であり，なおかつ信頼性を保証したデータ通信でなければならない．しかし，現存する通信プロトコル (TCP,UDP) では，協調分散処理に対応したものでないため，ネットワーク・トラフィックの増加，プロトコル処理のオーバヘッド，同期操作のタイムラグ等の問題が生じる．また，マルチキャスト機能をもつVMTP[2] およびXTP[3] 等の通信プロトコルは独自のOSにかなり依存しているため，一般的に使用するには扱いが難しいという問題点がある．

そこで本研究では，広く普及している分散システム上で協調分散処理を実現するため，マルチキャスト機能をもつTCPの設計を行った [4]．本稿では，この設計に基づき，実際に利用されている分散システム上にマルチキャスト機能をもつTCPを実装し性能評価を行った．具体的には，研究，開発，教育用としてソースが公開されているBSD系のソースを利用し，TCPの規格に逸れることなく実現を図った．

このDARPA標準であり業界標準でもある通信プロトコルTCPで一対多通信のマルチキャスト機能を実現することで，新しく通信プロトコルを作り出すことなく効率の良い一対多通信が可能となる．また，前述した問題点の解消もでき，従来同様ユーザ・インタフェースとしてUNIX標準のソケットを利用するためアプリケーションの生産性向上につながる．

以下，2章ではTCPでマルチキャスト機能を実現するために拡張を行った部分について述べ，3章では通常のTCPとマルチキャスト機能をもつTCPを比較し，評価実験を行った結果について報告する．そして，4章でその結果に関する考察，最後5章ではまとめと今後の課題について述べる．

2 マルチキャスト機能をもつTCP

この章では，マルチキャスト機能をもつTCPの設計仕様について基本的な概念，付加機能および実装上変更を行った部分に関してのみ述べる．この機能の実現において下位層としては，マルチキャスト拡張IP[5] を利用した．設計詳細に関しては文献 [4] を参照されたい．

2.1 一対多のマルチキャスト通信

本稿で扱う一対多のマルチキャスト通信およびグループの定義は，以下の通りである．

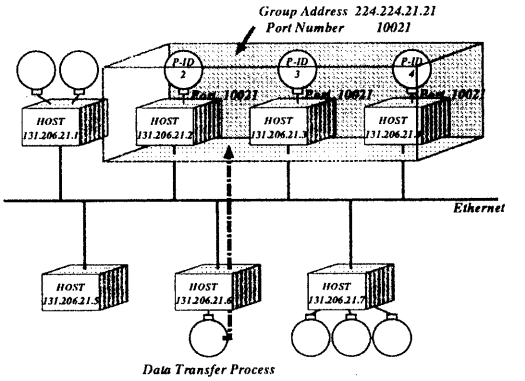

図 1: マルチキャスト通信およびグループの定義

マルチキャスト通信

ここで扱うマルチキャスト通信は，複数ホストで構成されたグループに対して任意の一つのホストがデータ通信を行う，一対多のマルチキャスト通信を指している．また，図1で示すようにデー

タ通信は矢印の方向，つまりあるホストからグループに対しての一方向データ通信のみである．

この制限は，TCP 上でマルチキャスト機能を実現するために，データ送信ホストでグループに属する全ホストの状態情報を管理しなければならないとうい制約からきている．

グループの定義

グループを構成するためには条件として，以下の四つを満たさなければならない．その理由については括弧内で説明する．

1. グループ構成ホストは，クラス D の IP アドレス (224.0.0.0 ～ 239.255.255.255) を保持しなければならない．(マルチキャスト拡張 IP を利用することで，グループ構成ホストが同じデータをほとんど同時期に受信可能となるからである)

2. グループ構成ホスト上のデータ受信プロセスは，全く共通のポート番号を保持しなければならない．(データ送信ホストから指定できるデータ受信側のポート番号は一つだけである．具体的に言えば，TCP ヘッダの受信側ポート番号領域は一つしか指定できない)

3. 各受信プロセスは，それぞれ異なった ID 番号を保持する必要がある．(データ送信ホストにおいて，グループ構成ホストから返される ACK パケットが，どのホストからのものであるか判断しなければならない．それを識別するためにオプションとして ACK パケットの TCP ヘッダ内に設定する)

4. マルチキャスト通信の間，グループへの途中参加・離脱はできない．(TCP を基にしてマルチキャスト機能を拡張しているため，データ通信はコネクション型ストリーム通信ある．そのため，一旦確立したコネクションは，データ送信完了まで開放できない)

2.2 TCP 制御ブロックの拡張

グループ構成ホストの状態情報を管理する tcp_gci 制御ブロックの構造体を図 2 に示す．この制御ブロックには，グループ構成ホストから返される ACK パケット内の情報が書き込まれる．この情報に基づき，グループとしての ACK パケットが決定されデータ通信が行われる．決定方法の詳細は 2.4 節の順序制御の部分で述べる．

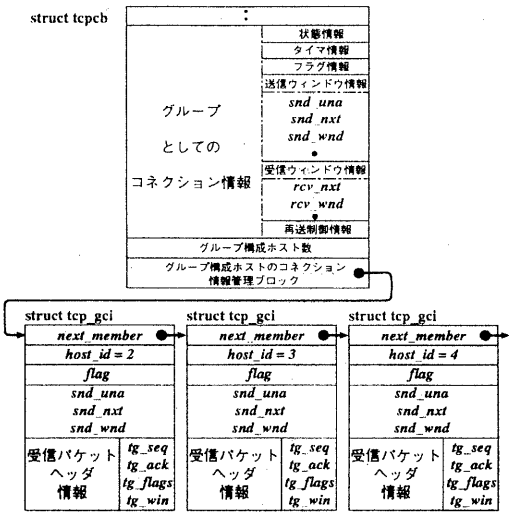

図 2: 拡張した TCP 制御ブロック

2.3 コネクションの開設および閉鎖

データ送信ホストが SYN パケットを送信し，グループを構成する全ホストから ACK & SYN パケットを受信，そしてその SYN パケットの ACK をグループに返すことでコネクション開設を行う．コネクションの閉鎖は開設のやりとりと同様，FIN パケットの送受信によって行う．

2.4 信頼性を保証したデータ通信

ここでは，マルチキャスト機能を TCP で実現する際に必要となる付加機能について説明する．説明に先立ち，以下で必要となる用語についてここで定義しておく．

SND.UNA	送達確認されていない最小送信順序番号．ここで，グループに対しては SND.UND$_{GROUP}$ と記述し，グループを構成する各ホストに対しては SND.UNA$_{HOST[i]}$ と記述することにする．以下の変数についても同様である．
SND.NXT	次に送信可能なデータの順序番号．
SND.WND	相手が受信できる，SND.UNA からのデータのバイト数．
SND.MAX	送信済みデータの SEQ 番号．
RCV.NXT	期待されている次の ACK 番号．

RCV.WND 受信されたはずのRCV.NXTからのデータのバイト数.

PKT.SEQ パケット内に示されるSEQ番号.

PKT.ACK パケット内に示されるACK番号.

PKT.WIN データ受信側で受信可能な受信バッファの空き領域のバイト数.

順序制御

順序番号に関してはTCPと同様に，パケット内のデータに対してバイト(オクテット)毎に番号をつける．この値が更新されるのは，データ送信ホストからグループに向けた順序番号のみである．これは，データ送信方向が一方向であるという制限のためである．また，送達確認に関しては，グループ構成ホストから全ACKを受信した時点で，データ送信ホストはグループのACKとして送達確認を行う．これは，データ送信ホストが如何にも，グループ構成ホスト全体を一つのホストとしてイメージし，データ通信を行っているようにするためである．グループ構成ホストからのACKパケット内の情報は，常にtcp_gci制御ブロックにを書き込まれる．その書き込む条件と，グループのACKパケットの計算式を以下に示す．

if ($SND.UNA_{HOST[i]} < PKT.ACK_{HOST[i]} \leq SND.MAX_{GROUP}$)
{ tcp_gciへACK情報を書き込む }
if (全ホストからACKパケットを受信) {
$PKT.SEQ_{GROUP} = \min(PKT.SEQ_{HOST[1]}, \cdots, PKT.SEQ_{HOST[n]})$
$PKT.ACK_{GROUP} = \min(PKT.ACK_{HOST[1]}, \cdots, PKT.ACK_{HOST[n]})$
$PKT.WIN_{GROUP} = \min(PKT.WIN_{HOST[1]}, \cdots, PKT.WIN_{HOST[n]})$
}

フロー制御

グループを構成する各ホストから返されるACKパケットには，そのホストがもつ受信バッファの空き領域を示す$PKT.WIN_{HOST[i]}$がある．この情報により各ホストの$RCV.NXT_{HOST[i]}$および$RCV.WND_{HOST[i]}$が決まる．それを基に，以下の計算式でグループに対する情報を更新する．

$RCV.NXT_{GROUP} = \min(RCV.NXT_{HOST[1]}, \cdots, RCV.NXT_{HOST[n]})$
$RCV.WND_{GROUP} = \min(RCV.WND_{HOST[1]}, \cdots, RCV.WND_{HOST[n]})$

再送制御

再送に関しても，TCP同様，送達確認されないデータに対して行う．異なる部分は，再送タイマの制御に必要なRTT(Round Trip Time)値の決定が，グループのACKパケットに対してだけKarnのアルゴリズムを適用している点である．

2.5 ソケットを使ったネットワーク通信

ソケットを利用してマルチキャスト機能をもつTCPを利用するために，最低必要となる手続きを図3 に示す．以下ではその手続きについて，データ送信ホスト側とグループ構成ホスト側に分け，手順を追って説明する．

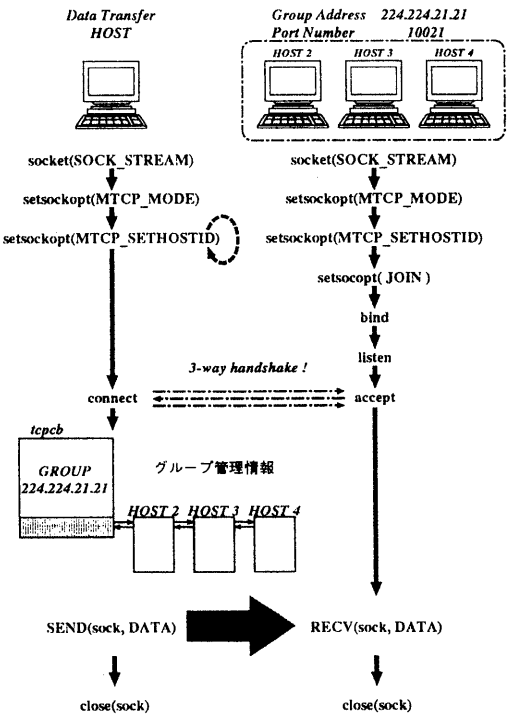

図 3: ソケットを利用したプロセス間通信

データ送信ホスト側

1. socket(SOCK_STREM)システムコールを使いソケット開く．
2. マルチキャスト通信を行うためsetsockopt(M-TCP_MODE)システムコールによりモード設定を行う．

3. グループを構成するホスト数だけ setsockopt(SET_HOSTID) システムコールを実行する．(グループ構成ホストの管理情報ブロックが作成される)
4. グループがもつインターネット・アドレス (この例では 224.224.21.21) に対して connect システムコールを実行することでコネクションが開設される．
5. send システムコールによりマルチキャストによるデータ送信を行う．このとき開いたソケットに対してユニキャスト通信はできず，常にマルチキャスト通信だけとなる．
6. データ送信が終了後，close システムコールによりソケットを閉じる．

グループ構成ホスト側

1. データ送信ホスト側と同様，ストリーム型のプロセス間通信を行うためにソケット開く．
2. データ送信ホスト側と同様，マルチキャスト通信を行うために setsockopt(MTC_PMODE) システムコールでモード設定を行う．
3. 自ホストの ID を setsockopt(SET_HOSTID) システムコールにより設定する．
4. グループを構成する各ホストは setsockopt(JOIN) システムコールを利用することで，それぞれがグループのインターネット・アドレス (この例では 224.224.21.21) を保持する．
5. bind システムコールにより，グループのインターネット・アドレスの名前づけを行う．
6. listen システムコールにより状態遷移 (LISTEN) を行い，データ送信側からの開設要求の待ち状態に入る．
7. accept システムコールによりデータ送信側の connect システムコールとの間で 3-way handshake が行われる．
8. コネクション開設後，データ送信ホストからマルチキャストされたデータは，各ホスト上の recv システムコールで受信される．
9. データ受信が完了後，グループを構成する各ホストそれぞれが close システムコールでソケットを閉じる．

3 評価実験

ここでは，実装したマルチキャスト機能をもつ TCP の通信プロトコル処理に関して，評価実験を行った結果について報告する．実験は，表 1に示す条件で行った．実験は，一つのホストが複数

表 1: 実験環境の構成

ワークステーション	SPARC station 2 : 10 台 Sun Microsystems 社 (処理能力 28.5 MIPS 主記憶 48 Mbytes)
OS	Sun OS 4.1.3
伝送媒体	Ethernet (10 Mbps)
送信量	16 bytes, 4096 bytes

のホストへ送信を行う一方向通信を，TCP とマルチキャスト機能をもつ TCP で比較した．測定条件は協調分散処理を考慮して，送信遅延が起きないようにソケットに TCP_NODELAY オプションを設定した．

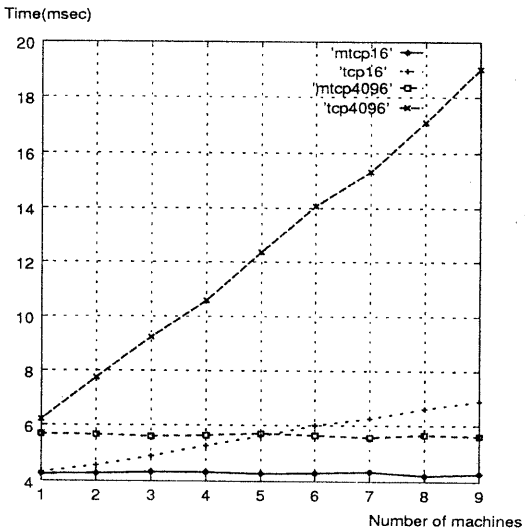

図 4: 実験結果

図 4は，DSE の同期操作等に必要とされるデータ量 16 bytes，そして TCP の最大送受信バッファの大きさである 4096 bytes のデータを，複数の計算機に send システムコールで送信したときにかかる時間を示している．これは，DSE が協調分散処理の一モデルであること，大量のデー

タ転送によるACKパケットの影響を観察するためである．測定は，他のバイト数についても行ったが，その結果からほとんど台数分に比例して送信効率が向上していることが確認できた．以下で考察を述べる．

4 考察

● **送信バイト数が小さい場合** (16 bytes)

同期操作等の小さなパケットのデータ送信を考慮して実験を行った．その結果，マルチキャスト機能をもつTCPを利用した場合，台数を増やしても通信プロトコル処理の負荷がほとんど変わらないことが分かった．この条件では，TCPと比べて効率的な処理が行えるといえる．

● **送信バイト数が大きい場合** (4096 bytes)

大量のデータを複数の計算機に送信することを考慮して実験を行った．TCPでは送信データが多くなると，より少ないデータ・サイズで送信を行うため，一ホストに対して行われる通信プロトコル処理が多くなる．そのため，ホスト数が多くなるにつれ，送信に必要な時間も多くなり，マルチキャスト機能をもつTCPとの差がよりいっそう顕著になってくる．このマルチキャスト機能をもつTCPを利用すれば，一ホストにかかる処理をグループに対して一度行うだけで良いため，効率的な処理が行えるといえる．

● **ACKパケットの影響について**

今回の実験では最大9台でグループを構成し評価を行ったが，ACKパケットによる影響はほとんど出現せず台数分の処理効率が得られた．しかし，これが32台，64台，そして将来的に256台とかなり多くの計算機を利用して協調分散処理を行う場合を想定すると，ACKパケットによる影響が出てくると予想される．これについては，シミュレーション評価の必要がある．

5 まとめ

本稿では，協調分散処理の通信処理向上を目指し，実際に使われている分散システム上で，TCPの規格を逸れることなくマルチキャスト機能を実装できることを示した．また，実験結果からその効果を確認し，通信プロトコル処理のオーバヘッドおよびネットワーク・トラフィックに関して性能向上が図れることが分かった．

このマルチキャスト機能をもつTCPでは，ユーザ・インタフェースとしてUNIX標準のソケットを利用でき，通信処理に要する時間もホスト数に関係なく一定に見積もることができるので，協調分散処理アプリケーションにおけるプログラム開発が容易になる効果もある．

今後の課題としては，マルチキャスト機能をもつTCPをハイパフォーマンス・コンピューティング環境DSEで利用できるように組み込み，実際の協調分散処理アプリケーションへ与える効果を総合的に調べる必要がある．

参考文献

[1] T. Tezuka, K. Ryokai, B. O. Apduhan and T. Sueyoshi："Implementation and Evaluation of a Distributed Supercomputing Environment on a Cluster of Workstation, *Proceedings of 1992 International Conference on Parallel and Distributed Systems*, pp.58-65, December 1992.

[2] David R. Cheriton, Carey L. Williamson："VMTP as the Transport Layer for High-Performance Distributed Systems", *IEEE Communications Magazine*, pp.37-44, June 1989.

[3] Greg Chesson："XTP/PE Overview", *Proceedings of 13th Conference on Local Computer Networks*, pp.292-296, Oct 1988.

[4] 高村 尚彦，大西 淑雅，B. O. Apduhan，末吉 敏則，有田 五次郎："TCPプロトコルにおけるマルチキャスト機能の実現法"，マルチメディア通信と分散処理ワークショップ，pp.49-56，March 1993.

[5] S. E. Deering："Host Extensions for IP Multicasting", *RFC-1112, SRI Network Information Center, Menlo Park, CA*, August 1989.