UDPを用いたクラスタコンピューティング向け通信プロトコル設計のための予備実験

江口 真
九州工業大学 情報工学部

手塚 忠則
九州工業大学 情報工学部
松下電器産業株式会社
九州マルチメディアシステム研究所

末吉 敏則
熊本大学 工学部

有田 五次郎
九州工業大学 情報工学部

Abstract

今日，高速なネットワークで計算機間を接続した*LAN*環境が普及し，この*LAN*環境を利用して並列処理を行うクラスタコンピューティングの研究が盛んに行われている．我々の研究室では，分散共有メモリモデルに基づくクラスタコンピューティング環境*DSE(Distributed Supercomputing Environment)*の研究を行っている．従来，*DSE*における通信処理機構では，*TCP*を用いてきた．しかし，*TCP*はコネクション型のプロトコルなので通信ポートを大量に消費してしまう．*OS*によっては，通信ポートに制限があり，直接接続により利用可能な計算機数を制限するという問題がある．そこで，我々は，*1*つの通信ポートで複数の計算機と通信ができる*UDP*に着目し，*UDP*をベースとするクラスタコンピューティング向けのプロトコルの検討を行っている．本論文は，プロトコル設計に先立ち，*UDP*を利用して基本的なデータ転送を行った場合の*TCP*に対する性能比を報告するとともに，加えるべき機能について述べる．

1. はじめに

近年，高速なネットワークで計算機間を接続したLAN環境を利用して，仮想的な並列計算機を構成するクラスタコンピューティングの研究が盛んに行われている [1][2]．我々の研究室では，分散共有メモリに基づいたクラスタコンピューティング環境である，分散スーパーコンピューティング環境 DSE(Distributed Supercomputing Environment) の研究を行っている [3][4][5][6].

DSEでは，計算機間の通信機構としてTCPを用いている．しかし，TCPはコネクション型のプロトコルであるため，1つの接続に対し，1つのポートを必要とする．ポートはオペレーティングシステム (OS) 資源の1つであり，多くのOSではユーザプロセスが同時に利用できるポートの個数に制限が課せられている．DSEでは，1台の計算機で複数のプロセスを実行可能なので計算機台数より多いクラスタ環境が実現できる．また，ネットワークの高速化や高性能化により数100台規模のクラスタ環境も実現可能になりつつある．このような，大規模クラスタ環境においてTCPを用いるとポート数の制限により直接通信できない可能性がある．

そこで，従来のDSEでは，割り当て可能なポート数より多い大規模なクラスタ環境を利用して並列処理を行う場合には，LAN上に鎖網やトーラス網などの仮想的なネットワークを構築していた．このため，直接接続されていない計算機間で通信を行うためには他の計算機を経由することになり，この中継が大きな通信オーバヘッドとなっていた．

我々は，この計算機を経由する通信オーバヘッドを削減することを目的として，1つのポートで複数の計算機と通信が可能なUDPを利用してDSEでの通信を行うことを検討している．UDPはコネクションレス型のプロトコルであり，TCPのような信頼性のための機構を有していない．このため，UDPを利用する場合には，信頼性のための機構を上位のプログラムで実現する必要がある．

本稿では，UDPを用いたプロトコル設計に先立ち行った予備実験の結果について報告し，DSEを対象としたプロトコルに要求される機能や制御について考察する．

2. プロトコル概要

DSEでUDPを利用する場合，上位プロトコルには以下の2つの機能が求められる．

(1) 複数の計算機から送信したDSEメッセージを送信元別に送信された順序で受信できる機能．

(2) DSEメッセージが紛失した際に復元できる機能．

クラスタコンピューティング環境では，複数計算機間での到着順序を保証した信頼性のある通信が要求される．UDPは信頼性の機構を有していないため，上記の2つの機能を実現するには，上位プロトコルで次に示す2つの制御が最低限必要である．

(1) 順序制御
DSEメッセージを必要に応じて分割して転送し，受信側で再び組立てる制御である．分割はネットワークの最大データ長であるMTU(Maximum Transmission Unit)より大きいDSEメッセージを送信する場合に行う．

UDPではパケットの到着順序が保証されていないため，シーケンス番号と呼ばれる連番をパケットに割り振って送信し，受信側ではこれを利用してDSEメッセージを組み立てる必要がある．また，複数のプロセスから送られて来たパケットを受信する必要があるため，シーケンス番号以外に送信元のプロセスを区別する番号も同時に付加する．送信元とシーケンス番号を見ることにより複数の計算機から送られて来たメッセージを送信元別に，かつ送信し順序で正確に組み立てることができる．

(2) 再送制御
パケット消失が発生した場合に，パケットを再送する制御である．この制御にも，前述のシーケンス番号を用いる．また，同一ポートで複数のプロセスからのDSEメッセージを受信するため，各プロセスに対して個別に再送制御を行う必要がある．

以上の制御をUDPの上位プロトコルに付加することで，DSEメッセージのやり取りが可能となる．次の予備実験では，UDPとTCPの性能比較を行い，プロトコル設計の基本となる制御について検討している．

3. 予備実験

3.1. 実験目的

本実験では，UDPを用いてピンポン転送およびバースト転送の2つの実験を行った．

- ピンポン転送
TCPとUDPを速度の面において比較するものである．TCPは現行のDSEにおいて利用しているプロトコルであり，プロトコルをUDPに変更した場合でも通信処理のパフォーマンスの低下を避ける必要がある．つまり，DSEの通信パフォーマンスを低下させない条件下において，上位プロトコル処理に割り当て可能なおおよその処理時間の調査がこの実験の目的である．
- バースト転送
UDPを用いて，連続でデータを転送した場合にデータ消失の割合を調査するものである．つまり，UDPを用いた場合のデータ受信能力を調べ，前述した再送制御のみで十分であるかを検討する．

3.2. 実験環境

表1に実験環境を示した．なお，実験はネットワークトラフィック量が少なく，計算機自身の負荷も少ない状態で行った．また，表1において送信および受信ソケットバッファのサイズを示しているが，これは，ソケットバッファのサイズを調べるシステムコール(getsockopt())により調査した値である．

表 1. 実験環境

計算機	SparcStation20(model50)
OS	SunOS 4.1.4
ネットワーク	Ethernet 10Mbps 10Base5
ソケット送信バッファサイズ	9,000Bytes
ソケット受信バッファサイズ	18,032Bytes

3.3. 実験内容

3.3.1 ピンポン転送

TCP，UDPそれぞれのプロトコルについてピンポン転送による実験を行った．図1に示すように，本実験では，まず，計算機Aから，計算機Bにデータを送信する．計算機Bでは，計算機Aからのデータの受信処理を終えると，直ちに計算機Aに対してデータを送信する．

DSEでは，比較的小さいデータサイズのメッセージが頻繁にやりとりされるために計算機Aから計算機Bへ一度に送るデータサイズは16, 32, 64, 128, 256, 512, 1,024Bytesとした．

実験では，このピンポン転送を10,000回行うのに要する時間を測定した．往復の時間ではなく10,000回の転送時間にした理由は，測定に利用したタイマーの精度の問題および，タイマー関数(gettimeofday())を呼び出すオーバヘッドの影響を少なくするためである．また，TCPについては送信遅延を少なくするためのオプション(NO_DELAY)をシステムコール(setsockopt())により指定した．このオプションを指定することで，ピンポン転送の処理速度が速くなる．このことは，DSEに関する実験においても確認している[5].

3.3.2 バースト転送

バースト転送による実験では，図2に示すように，計算機Aは連続的にデータの送信を行い，計算機Bでそのデータを受信する．この実験では，10,000回送信を行い

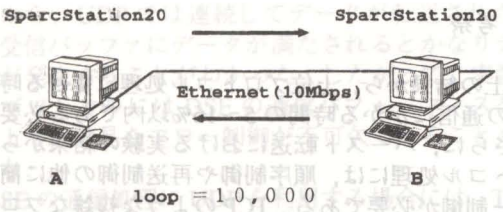

図 1. ピンポン転送

受信できた個数を調べた．また，消失したパケット (受信側で受け取れなかったパケット) の分布を見るために各パケットには 0 から 9,999 までの連番を付けて送信した．なお，一度に送信するデータサイズはピンポン転送と同様に 16, 32, 64, 128, 256, 512, 1,024Bytes の 7 つとした．

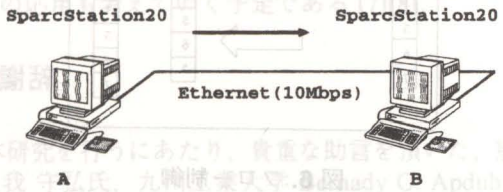

図 2. バースト転送

4. 実験結果及び考察

4.1. ピンポン転送の結果

TCP, UDP それぞれにおいて，ピンポン転送を行った結果を図3に示す．すべてのデータサイズにおいて，UDP は TCP より短い時間でデータの転送を行えることがわかる．また，UDP の実行時間はデータサイズが 16Bytes の時は，TCP の実行時間の 84 %であるのに対し，データサイズが 1,024Bytes の時は，TCP の実行時間の 95% であった．よって，UDP に付加する信頼性の機構に要するプロトコル処理時間は，TCP 送信時間の 5～16%以内に抑える必要があることがわかる．データサイズが小さい方が，TCP と UDP の比率の差が大きい理由の詳細を以下に述べることにする．

TCP に要する時間は，信頼性のためのプロトコル処理時間とデータ転送時間に分けられる．一方，UDP に要する時間は，主にデータ転送時間である．信頼性の機構としてヘッダにシーケンス番号，ACK 番号，ウィンドウサイズなどを付加するプロトコル処理はデータサイズによらず一定である．しかし，データ転送に要する時間はデータサイズが大きくなるにつれ増加する．このため，データサイズが小さい方の TCP と UDP のピンポン転送での時間における比率の差が大きくなったと考えられる．

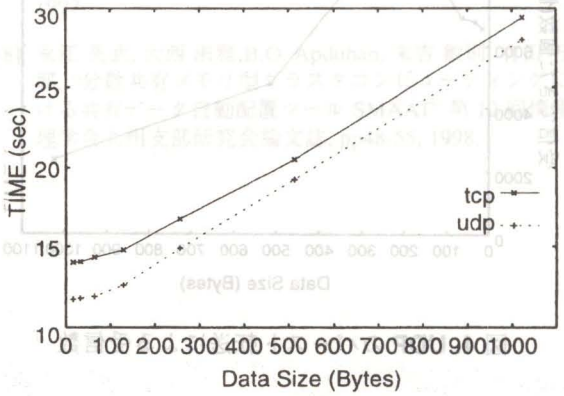

図 3. TCP と UDP のピンポン転送による比較

4.2 バースト転送の結果

連続にデータを転送するバースト転送を行った結果を図 4 に示す．データサイズが大きい 512, 1,024Bytes のときは，データサイズが小さい 16, 32, 64, 128, 256Bytes に比べて受信できた個数が少なかった．これは，データサイズが大きい場合には受信バッファが早くあふれるためと考えられる．一方，データサイズが 16, 32, 64Bytes よりも 128, 256Bytes の方が受信できた個数が多かった．この理由は，データサイズが小さい 16, 32, 64Bytes の場合，送信側でのデータの送信にかかる時間が非常に短いため，データが到着するスピードが速すぎて受信処理が間に合わないためだと考えられる．128, 256Bytes の場合，送信側でデータの送信にある程度の時間がかかるため，16, 32, 64Bytes のときに比べると受信処理が間に合っていると考えられる．これらの結果から，バースト転送ではデータサイズ 256Bytes が最適と考えられる．

図 5 は，横軸にパケットの番号をとり，受信できなかったパケットに対して縦線を引いたものである．なお，データサイズが 16, 32, 64Bytes と 128, 256Bytes それぞれについては，ほぼ同様な結果が得られたので，16Bytes のものと 128Bytes を代表として示した．また，パケット番号 1000 番から 9,999 番までは，600 番～1,000 番までと同様の状態が続いているためここでは省略した．

図 5 から，パケット番号の若い方ではデータを受信していることがわかる．データサイズが大きくなるにつれ，パケットロスし始めるパケット番号が 500 番前後から 300 番，100 番，50 番としだいに若くなっている．データサイズの大きい方がパケット番号の若いものからパケットロスしている理由は，データサイズが大きくなるとソケット受信バッファが早くいっぱいになるためであると考えられる．受信バッファサイズが 18,032Bytes であるため，落ち始めるシーケンス番号は，それぞれの

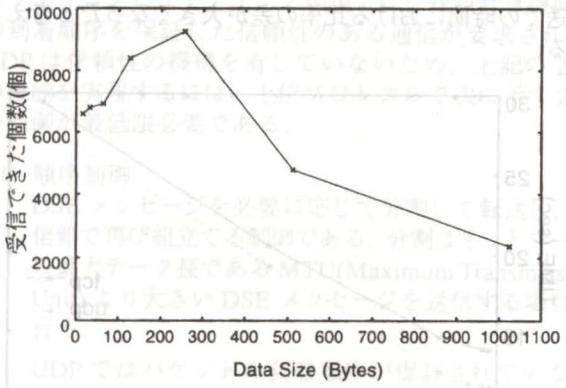

図 4. UDP のバースト転送による受信数

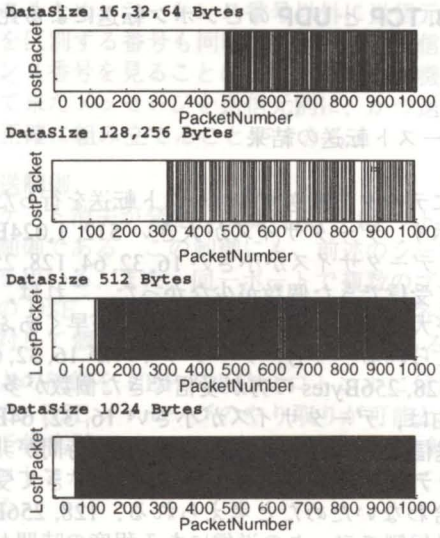

図 5. 受信できなかったパケット番号

データサイズ (16, 32, 64, 128, 256, 512, 1,024Bytes) で割って理論値はそれぞれ 1,127 番目, 564 番目, 281 番目, 140 番目, 70 番目, 35 番目, 17 番目となる．データサイズが 16, 32Bytes のとき理論値よりも先にパケットロスするのは UDP のヘッダやネットワークによる影響と思われる．一方，データサイズが 64～ 1,024Bytes のとき理論値よりも後にパケットロスするのは，受信処理が行なわれて受信バッファに空きができるからである．バースト転送の実験からわかるように，連続でデータを送信し，完全に受信できるのは，受信バッファにデータが満たされるまでのパケット番号の若いものに限られる．このため，再送制御のみによるデータの復元では，再送が膨大に発生し再送にかかるオーバヘッドの増大が予想される．したがって，フロー制御が不可欠であることがわかる．

4.3 考察

以上の結果から，上位プロトコル処理に費せる時間は TCP の通信にかかる時間の 5～16%以内である必要がある．さらに，バースト転送における実験の結果から上位プロトコル処理には，順序制御や再送制御の他に簡易なフロー制御が必要である．TCP のような複雑なフロー制御を用いる方法もあるが，上位プロトコルの処理時間を短くすることが重要である．

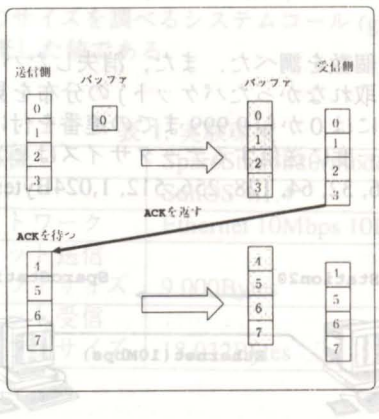

図 6. フロー制御

例えば，図 6 のように送信側から一度に送れるデータの個数を制限する (図では，一度に送れるパケット数を 4 個としている)．送信側は送りたいデータが多量にあっても一定のパケット数だけ送信し，ACK が受信側から送られてくるのを待つ．受信側は一定のパケット数を受信すると，ACK により受信したことを送信側に知らせる．ACK を受信した送信側は次のデータ (4,5,6,7) を受信側に向けて送信を開始する．クラスタコンピューティングでは複数の計算機間で通信を行うため，それぞれの計算機ごとに受信数を数え ACK を返す機構が必要である．このような簡易なフロー制御を導入できれば，ソケット受信バッファの溢れを抑制することができる．なお，実際に通信機構として用いる際には，適切なデータサイズ，1 度に送信できる適切なパケット数を十分考慮しなければならない．これに関しては，実際にプロトコルを実装し，評価を行う予定である．

5 おわりに

本稿では，UDP を用いたプロトコル設計に先立ち行った予備実験の結果について考察した．

ピンポン転送実験では，データサイズが 16 から 1024Bytes の範囲において，UDP は TCP で通信処理を行う場合に比べて，約 5～16%程度短い時間で処理を行えることがわかった．よって，DSE の処理速度を低下させないためには，この処理時間の差に収まる程度の軽いプロトコルでなければならない．バースト転送実験の

結果から，UDPでは連続してデータが転送された場合に，受信バッファにデータが満たされるとかなりデータ消失が発生することがわかった．したがって，実装する上位プロトコルにはDSEの通信パフォーマンスを考慮した上で，簡易なフロー制御が不可欠であることも確認できた．

DSEの通信処理にUDPを利用する場合には，TCPに用いられているウィンドウ制御のように複雑なフロー制御は用いずに簡単なフロー制御順序制御，再送制御の3つを備えた単純で処理の軽いプロトコルを用意する必要がある．これらの制御を実現するために，上位プロトコルに用いるヘッダの作成を行い，それぞれの制御を実装していく予定である．さらに，並列アプリケーションによる評価も行う予定である．

また，今後の課題として，UDPのブロードキャストの利用も検討している．具体的には，初期データや計算結果などの同じデータを複数の計算機に送信する場合を考えている．さらには，キャッシュのコヒーレンス制御等への応用も考えていく予定である [7][8].

6 謝辞

本研究を行うにあたり，貴重な助言を頂いた，熊本大学久我 守弘氏，九州工業大学 Bernady O. Apduhan 氏，及び大西 淑雅氏に深く感謝いたします．

References

[1] 小口 正人，新谷 隆彦，田村 孝之，喜連 川優: "ATM 結合 PC クラスタの TCP 再送機構の解析と並列データマイニングの性能向上", 情報処理学会研究報告 DPS87-37,OS77-37,pp.215-220,1998

[2] 手塚 宏史，堀 敦史，Francis O'Carroll 石川 裕: "RWC PC Cluster II の構築と性能評価", ARC128-5,HPC70-5,1998

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

[4] 大西 淑雅，了戒 清，末吉 敏則: "分散共有メモリモデルに基づく HPC 環境の高性能実装と性能評価," 情報処理学会論文誌, Vol.36, No.7, pp1729-1737, 1995.

[5] B.O. Apduhan, T. Sueyoshi, T. Tezuka and I. Arita:"The Effect of Communication Processing in Network Supercomputing Environment",Proc. of 7th International Joint Workshop on Computer Communications,pp.373-380,1992

[6] 手塚 忠則，了戒 清，末吉 敏則: "分散システムを利用した並列処理環境における通信処理の影響," 情報処理学会「マルチメディア通信と分散処理」ワークショップ論文集, pp.249-256, 1993.

[7] 宮原 敦夫，大西 淑雅，末吉 敏則: "クラスタコンピューティングにおけるソフトウェア・キャッシュ機構の評価," 第 11 回情報処理学会九州支部研究会論文誌, pp250-259, 1997.

[8] 永江 英武，大西 淑雅,B.O. Apduhan，末吉 敏則，有田 五次郎: "分散共有メモリ型クラスタコンピューティングにおける共有データ自動配置ツール SMAAT" 第 12 回情報処理学会九州支部研究会論文誌, pp48-55, 1998.