TCP による高速データ転送のための通信処理軽減手法の提案と実装

寺井 達彦 † **松尾 孝広** ‡ **長谷川 剛** § **村田 正幸** †

† 大阪大学大学院基礎工学研究科情報数理系専攻
〒 560-8531 大阪府豊中市待兼山町 1-3
Phone: 06-6850-6616, Fax: 06-6850-6589
E-mail: {terai, murata}@ics.es.osaka-u.ac.jp

§ 大阪大学大学院経済学研究科
〒 560-0043 大阪府豊中市待兼山町 1-7
Phone: 06-6850-5233
E-mail: hasegawa@econ.osaka-u.ac.jp

‡ 関西電力株式会社
E-mail: t-matuo@ics.es.osaka-u.ac.jp

あらまし これまで、ネットワーク速度に比較して、エンドホストのデータ転送処理速度は高速であったため、エンドホストが処理ボトルネックとなる状況はさほど想定されておらず、エンドホストの高速化、高機能化に関してはあまり検討がなされていなかった。本稿では、まず TCP によるデータ転送を行う際のエンドホストにおける処理を分析し、その結果、CPU による TCP/IP プロトコルの処理時間と比較して、メモリアクセス処理に要する時間が大きいことを示した。そこで、メモリアクセス回数の削減によってデータ転送の高速化を行う、エンドホストにおけるデータ処理方式の提案を行う。また、実コンピュータ上に提案する処理方式を実装し、性能評価実験を行い、従来方式に比べて、データ転送速度を約 30–50%高速化できることを示す。

キーワード TCP (Transmission Control Protocol)、エンドホスト、ソケットバッファ、メモリコピー、プロトコル処理

Improvement of Processing Time at Endhosts for High-Speed Data Transfer by TCP

Tatsuhiko Terai† **Takahiro Matsuo**‡ **Go Hasegawa**§ **Masayuki Murata**†

†Department of Infomatics and Mathematical Science
Graduate School of Engineering Science, Osaka University
1-3, Machikaneyama, Toyonaka, Osaka 560-8531, Japan
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: {terai, murata}@ics.es.osaka-u.ac.jp

§Graduate School of Economics, Osaka University
1-7, Machikaneyama, Toyonaka, Osaka 560-0043, Japan
Phone: +81-6-6850-5233
E-mail: hasegawa@econ.osaka-u.ac.jp

‡Kansai Electric Power Co.
E-mail: t-matuo@ics.es.osaka-u.ac.jp

Abstract Most of the researches have been concentrated on how to avoid and dissolve the network congestion, and only a few discussions on the performance improvement of the endhosts are recently made. In this paper, we focus on data transfer using TCP and analyze the details of TCP data transfer at the endhost. Through the analysis, we have found that memory access is the largest obstacle for performance improvement in TCP data transfer. Then we propose a new mechanism which reduces the number of memory access in TCP data transfer, and compose some of new socket system calls to avoid the redundant memory copy process. We further implement the proposed mechanism to the actual computers running FreeBSD, and show that the proposed mechanism can improve 30–50% of the transfer speed compared with the original mechanism through some experiments.

Keyword TCP (Transmission Control Protocol), endhosts, socket buffer, memory copy, protocol processing

1 はじめに

近年のインターネットの急速な発展に伴い、現在では非常に多くの情報を WWW (World Wide Web) を介して得ることが可能となっている。WWW の普及によるインターネット利用者の爆発的な増加と、それに伴うネットワークトラヒックの増加に対しては、さまざまな解決策が提案、実装されている。

その一例として、物理層、データリンク層におけるネットワークの高速化、高機能化が挙げられる。例えば、WDM (Wavelength Division Multiplexing) や MAPOS (Multiple Access Protocol over SONET/SDH) [1] といった光通信技術を用いた高速なデータ転送技術をはじめとして、高帯域ネットワークの実現に向けて多くの研究が行われている。また、ネットワークの輻輳に対してもさまざまな解決案が検討されている。特に、現在のインターネットトラヒックの大部分を占めているトランスポート層プロトコルである TCP (Transmission Control Protocol) については、輻輳回避方式に関しての多くの研究が行われてきた (例えば [2])。

しかし、エンドホストにおけるデータ転送処理の高速化、高機能化に関してはこれまであまり検討がなされていない。これは、これまではエンドホストの処理速度に比べてネットワークは低速であったため、エンドホストの処理がデータ転送時のボトルネックとなるような状況はさほど想定されなかったためである。しかし、現在では上述のようにネットワークにおけるデータ転送技術が発展し、ネットワーク帯域が飛躍的に増大したため、エンドホストがデータ転送処理のボトルネックとなりつつある。実際、現在のインターネットにおいても、人気のある Web Server ではピーク時に毎秒数百のリクエストを処理しなければならない状況にあり、現実にエンドホストの処理がボトルネックとなる状況が発生している。

そこで本稿では、まず、TCP によるデータ転送時におけるエンドホストの様々な処理にかかる時間の分析を行う。その結果、CPU によるプロトコル処理時間等と比較して、メモリアクセスの処理に大きな時間を要することを明らかにする。次に、TCP によるデータ転送時に必要なメモリアクセス回数を削減するデータ転送処理方式の提案を行う。また、提案したデータ転送処理方式を実コンピュータ上へ実装し、従来の処理方式との性能比較実験を行うことにより、その有効性の検証を行う。

2 エンドホストにおけるプロトコル処理機構

本稿では、API (Application Program Interface) として、多くの UNIX システムが用いているソケットインターフェース [3] を実装しているエンドホストを対象とする。API とはユーザアプリケーションによるプロトコルに依存しない命令要求を、TCP や UDP (User Datagram Protocol) 等のプロトコルに依存した処理へと変換するインターフェースであり、TCP や UDP 等によるデータ転送速度はこのインターフェースの実装方式に大きく依存する。本稿で対象とするソケットインターフェースは、1983 年に 4.2BSD において初めて実装され、現在では多くの BSD 系以外の UNIX システムや UNIX 以外のシステムにおいても現在広く実装されている API である (詳細については [4] 参照)。

以下、TCP によるデータ転送を行う際のエンドホストのプロトコルスタック処理に関して、図 1 を用いて各層ごとに順に説明を行う。

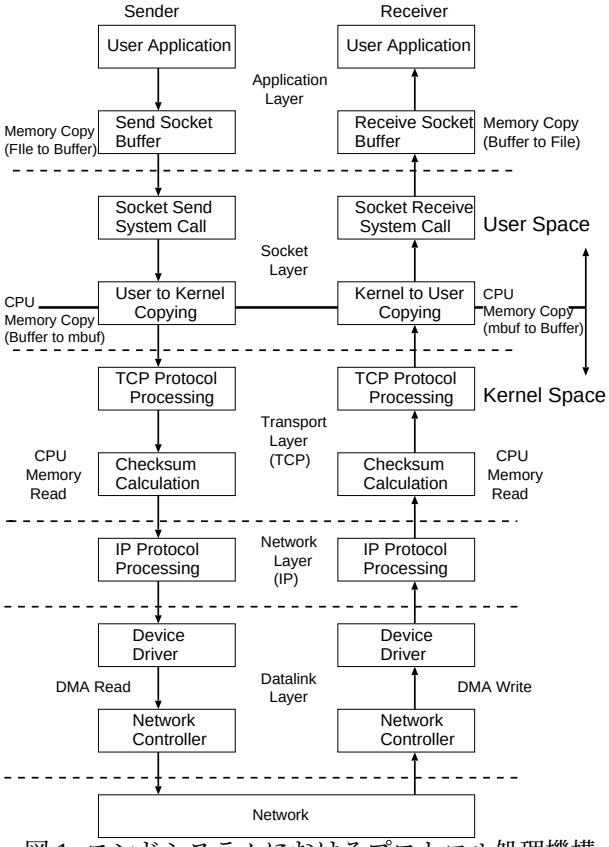

図 1: エンドシステムにおけるプロトコル処理機構

データ送信の際の処理は以下のようになる ([5] 参照)。

- **アプリケーション層**
  データ転送を行う際にはまず転送するデータをアプリケーションバッファに格納する。次に、ファイルのデータがディスク上に存在する場合にはディスクからメモリ上に存在するバッファへのコピーを行う (disk read、memory write)。また、メモリ上にあらかじめデータが存在する場合にはメモリ間でコピー (memory copy) を行う。
- **ソケット層**
  アプリケーションバッファへのデータ格納後、ソケットインターフェースのシステムコールを用いてソケットレイヤの送信バッファ (以下、送信ソケットバッファ) にアプリケーションバッファ内のデータを格納する (memory copy)。
- **トランスポート層**
  CPU によって TCP のプロトコルプロセッシングを行う（図中の “TCP Protocol Processing”）とともに、送信ソケットバッファ内の送信データから TCP セグメントを構成し、セグメント全体のチェックサムの計算を行うためにメモリにアクセスする (memory read)。
- **ネットワーク層**
  CPU によって IP のプロトコルプロセッシング（図中の “IP Protocol Processing”）を行い、TCP セグメントを用いて IP データグラムを構成する。
- **データリンク層**
  DMA (Direct Memory Access) によって IP データグラムをネットワークインターフェースのバッファに格納し (DMA READ)、ネットワークインターフェースのデバイスドライバ固有の処理によってパケットをネットワークリンク (物理層) に送出する。

図 1 に示すように UNIX におけるメモリ管理システム

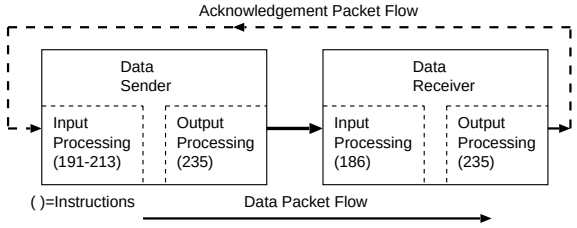

図 2: TCP の処理行程 (文献 [6])

の実装においては、ユーザアプリケーションが使用するメモリ領域とカーネルが使用するメモリ領域の分割を明確に行い、ユーザアプリケーションからのカーネル領域のデータに対するアクセスを禁止し、カーネル領域内にあるデータのユーザアプリケーションからの保護を実現している。

受信側のプロトコル処理については送信側と同様の処理を逆の手順で行うのでここでは省略する。

3 TCP/IP 通信におけるエンドホストの処理オーバーヘッドの分析

3.1 TCP/IP のプロセッシング命令数

文献 [6] では、TCP/IP プロトコルの処理に要する処理命令数を、実装されている TCP/IP ソースコードを解析することによって導いている。解析に用いられた TCP/IP は、現在広く使われている BSD 系 OS に実装された TCP/IP である。本稿では、文献 [6] の結果を用いて、TCP/IP の処理に必要な時間を算出する。図 2 は、TCP の処理行程の様子を示した図である。TCP プロセッシングの命令数は、その処理内容の違いにより送信側ホストと受信側ホストで異なる。データパケット、ACK パケットを送出する際に実行する命令数は、ともに 235 命令で等しいが、送信側ホストで ACK パケットを受信する際に行う命令数は、送信側では 191 から 213 命令であり (処理内容によって異なる)、受信側ホストでデータパケットを受信する際に行う命令数は 186 命令である。また、IP 処理を行う命令数は、パケット受信時には 57 命令、パケット送出時には 61 命令を要する。

3.2 メモリアクセス機構

TCP/IP 通信を行う際のメモリアクセスに関しては、通常のメモリアクセス以外に DMA が存在する。DMA は、CPU による処理を介さずにネットワークデバイスがホストのメインメモリに直接アクセスを行う。従って、システムは CPU の処理と並列して大容量のデータ転送をネットワークデバイスとメモリの間で行うことが可能である。つまり、Device Driver と Network Controller 間の転送データの移動に DMA を用いているため (図 1 参照)、ネットワーク–マシン間のパケットの送受信と CPU 処理はオーバーラップされて行われている。

3.3 エンドホストの処理時間の推定

次に、実コンピュータの処理能力を基に、実際の TCP/IP 通信における送受信ホストでの処理時間の推定を行う。まず、推定に必要となるマシン性能の測定実験を lmbench [7] を用いて行う。lmbench は、UNIX 系 OS の詳細な性能を計測するためのベンチマークツールであり、マシンの基本的な構成要素についての性能を測定することができる。表 1 に本稿で用いた 2 種類のマシン (マシン 1、マシン 2) のベンチマークテストの結果を示す。表より、CPU 速度が低下するとメモリアクセスに関わる速度も低下していることが分かる。

表 1: 実験用マシンの性能評価の結果

	マシン 1	マシン 2
CPU	PIII Xeon 550MHz	PII 266MHz
Main Memory [MB]	512	128
Memory Copy [MB/sec]	180.32	51.98
Memory Read [MB/sec]	383.12	203.54
Memory Write [MB/sec]	181.46	71.88

次に、ベンチマークテストで得られた結果と 3.1 節の TCP/IP プロセッシング命令数の分析に基づいて、TCP/IP 通信における送受信ホストでの処理時間の推定を行う。推定を行うモデルとして、表 1 のマシン 1 を 2 台用いて、マシン間を 622 [Mbps] の帯域を持つリンクで接続し、4 [MB] のデータを転送する場合を考える。またパケットサイズを 16 [KB] とする。従って、転送パケット数は 258 パケットとなる。表 2 にマシン 1、マシン 2 をそれぞれ送信側ホストにした場合の処理時間の推定結果を示す。なお、DMA は CPU 処理とオーバーラップされて処理されるので、ここでは考慮していない。この結果から、TCP によるデータ転送時におけるエンドホストの行う処理においては、メモリアクセスの処理に要する時間が最も大きく、TCP/IP のプロトコルプロセッシングに要する時間に比較して、メモリコピーに要する時間は、約 200 倍であることが分かる。さらに、表 2 よりマシン速度が低い場合は、処理時間全体に対するメモリアクセスの処理にかかる時間がさらに大きくなっていることがわかる。また、技術の進歩による CPU 速度の向上と比較すると、メモリアクセス速度の向上スピードは遅いため、エンドホストの TCP/IP によるデータ転送処理速度を向上させるためには、メモリへのアクセス回数を減少させることが重要であると考えられる。

4 メモリコピー回避方式の提案と実装

2 章で述べたように、TCP によるデータ転送においては 2 回のメモリコピー (ファイルからアプリケーションバッファ、及びアプリケーションバッファから送信ソケットバッファ (mbuf と呼ばれる)) を必要とする。また、3 章で述べたようにメモリコピーがエンドホストの処理においての最も大きなオーバーヘッドとなるため、メモリコピー回数の削減がデータ転送速度の向上のための重要な要素である。本章ではまず従来方式におけるメモリコピー処理の説明を行い、その冗長性を指摘するとともに、メモリコピー回数の削減を行う方式を提案し、実装実験による提案方式の評価を行う。

4.1 従来方式による冗長なメモリコピー

図 3 に従来方式における TCP によるデータ転送の際の処理の様子を示す。従来方式においてはユーザアプリケーションが TCP によるファイル転送を行う際に、まず転送を行うファイルのデータをユーザ領域内のアプリケーションバッファに格納するためにメモリコピーを行う。次にソケットインターフェースにより提供されているシステムコールによって、アプリケーションバッファ内のデータをカーネル領域内のソケットバッファ (mbuf) に格納する。この際にもメモリコピーを行う。従って合計 2 回のメモリコピーが行われる (文献 [8] 参照)。この場合、例えばファイルシステム内に存在するファイルデータをユーザ領域のアプリケーションバッファを経由してから、カー

表 2: 送信側ホストにおける処理時間の推定結果

	マシン 1 (PIII-550MHz,Memory 512 [MB])		マシン 2 (PII-266MHz,Memory 128 [MB])	
処理内容	導出過程	処理時間 (μs)	導出過程	処理時間 (μs)
Memory Copy (file to buffer)	4 [MB]/180.32 [MB/sec]	22182	4 [MB]/51.98 [MB/sec]	76953
Memory Copy (buffer to mbuf)	4 [MB]/180.32 [MB/sec]	22182	4 [MB]/51.98 [MB/sec]	76953
TCP output (data pkt)	235 [ins] × 258 [pkt]/$550M$	105	235 [ins] × 258 [pkt]/$266M$	217
Memory Read (for checksum)	4 [MB]/383.12 [MB/sec]	10441	4 [MB]/203.54 [MB/sec]	19652
IP output (data pkt)	61 [ins] × 258 [pkt]/$550M$	27	61 [ins] × 258 [pkt]/$266M$	56
IP input (ack pkt)	57 [ins] × 258 [pkt]/$550M$	25	57 [ins] × 258 [pkt]/$266M$	53
TCP input (ack pkt)	213 [ins] × 258 [pkt]/$550M$	91	213 [ins] × 258 [pkt]/$266M$	197
合計		55053		174081

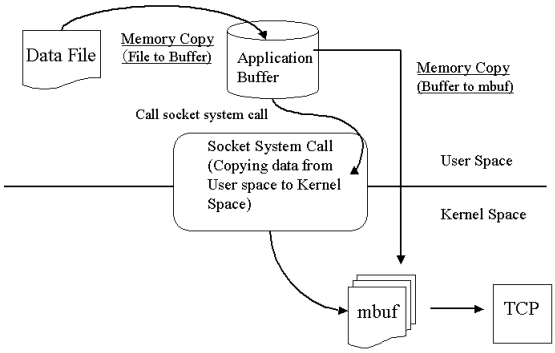

図 3: 従来方式における TCP データ転送の際の処理方式

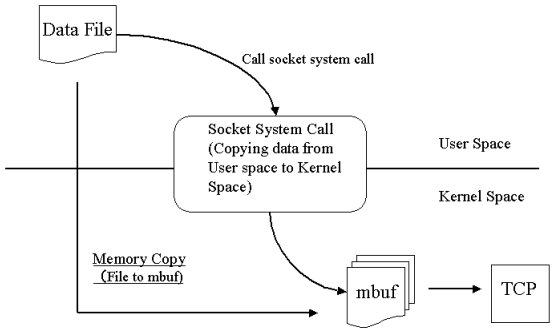

図 4: 提案方式における TCP データ転送の際の処理方式

ネル領域内のソケットバッファに格納する事は冗長であり、ファイルシステムから直接カーネル領域内のソケットバッファに格納すれば効率のよいデータ転送が行うことができると考えられる。2 章で述べたように、UNIX におけるメモリ管理システムの実装においては、ユーザ領域、カーネル領域及びファイルシステムは、使用するメモリ領域を明確に分割している。従来方式においてはデータをアプリケーションバッファにメモリコピーを行う必要があるが、データ転送の際には一度目のメモリコピーは冗長であるといえる。

4.2 提案方式によるメモリコピー回避の方法

3.3 節で述べたように、エンドホストの TCP/IP によるデータ転送処理速度を向上させるためには、メモリコピーの回数を減少させることが重要であると考えられる。メモリコピーの回数を減少させる手法としては、例えば文献 [9] で提案されている “fbuf (fast buffer)”方式が挙げられる。この方式は、エンドホストにおいてカーネル領域とユーザ領域のメモリを共有することよって、メモリコピーを省略し、エンドホストの高速化を実現している。本稿では、 4.1 節で示した、ファイルのデータを転送する際のファイルからアプリケーションバッファへのメモリコピーを回避することによって、エンドホストのデータ転送速度を向上する方式を提案する。

図 4 に提案方式を用いた場合の TCP によるデータ転送の際の処理方式を示す。提案方式においては、ユーザアプリケーションが TCP によるファイル転送を行う際に、新たに追加したソケットシステムコールによって、ファイルシステムからデータを直接カーネル領域内のソケットバッファに格納する。従来方式におけるソケットシステムコールは、ファイルからアプリケーションバッファにメモリコピーが必要となっていたために、アプリケーションバッファを引数としていたのに対し、新たに追加したソ

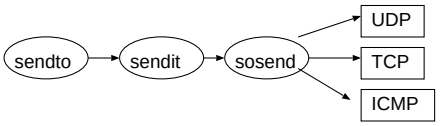

図 5: ソケットシステムコールの制御の流れ

ケットシステムコールは、ファイル記述子 (file descriptor) を引数とする。そして、カーネルは引数として受け取ったファイル記述子を利用し、ファイルシステム内のファイルからカーネル領域内のソケットバッファへ直接コピーを行うことによって、4.1 節で述べた冗長なメモリコピーを省略する。以降、FreeBSD におけるソケットシステムコールの制御の流れを示した図 5 に沿って、提案方式で実装したソケットシステムコールを具体的に説明する。

データ転送を行うためのシステムコールはすべて直接または間接的にシステムコール `sosend` を呼び出す。システムコール `sosend` は、データをユーザ領域からカーネル領域内のソケットバッファへコピーし、各プロトコルプロセッシングを行う処理へデータを送る役割を果たす。提案方式では、`sosend` で行うカーネル領域内のソケットバッファへのコピーの方法を、従来方式であるアプリケーションバッファを引数としてとるのではなく、ファイル記述子を引数として受け取り、それを利用してファイルシステムからソケットバッファへ直接コピーできるように `sosend` システムコールを改良した。また、新しいシステムコールがファイル記述子を利用できるように、`sendto` システムコールのとる引数をアプリケーションバッファからファイル記述子に変更し、`sendto` と `sosend` の橋渡しとなるシステムコール `sendit` がアプリケーションバッファを使ってデータを得ていた部分を、ファイル記述子を用いるように変更した。この 3 つのソケットシステムコールの追加 / 変更と、ファイル記述子

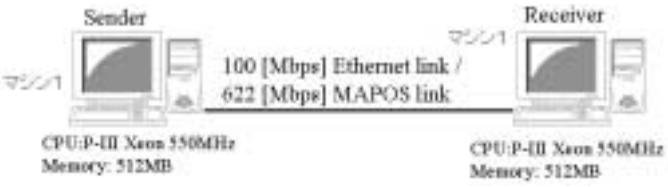

図 6: 実験ネットワーク環境

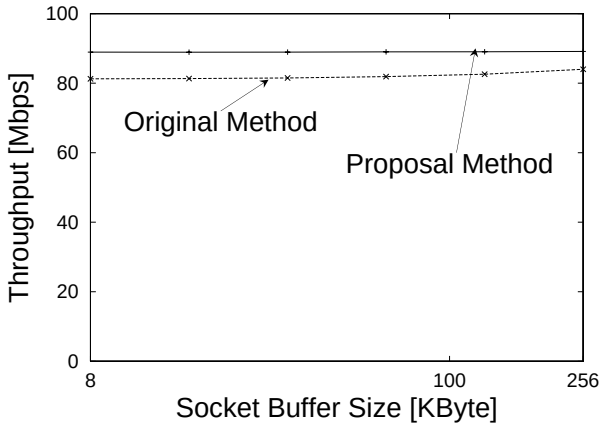

図 7: Ethernet におけるデータ転送実験結果

をヘッダ情報に含ませるために改良した 2 つの構造体によって、上述した提案方式を実装した。提案方式は、fbuf 方式に比べて、変更ソースコード数の観点から見ても実装が容易である。提案方式によって付け加えられたカーネルのソースコードは約 300 行だが、オリジナルのカーネルのソースコードからの実質的な変更行数は約 100 行である。特に、改良を加えたのはソケットレイヤのみであり、下位層のネットワークに依存しない実装が可能であることが利点として挙げられる。

5 性能評価実験

本節では、 4.2 節で述べた提案方式を実コンピュータ上に実装し、データ転送実験を行うことによって提案方式の性能評価を行う。

5.1 実装環境

実装に使用した OS は、FreeBSD-2.2.8 である。実験に使用したネットワーク環境は、図 6 に示すような 2 台のマシンを最大 100 [Mbps] のリンク帯域を持つ Ethernet、あるいは 622 [Mbps] のリンク帯域を持つ MAPOS で直接接続したネットワークである。実験を行ったマシンは、表 1 に示したマシン 1 及びマシン 2 である。

また、Delayed ACK オプションは使用しない。それ以外の TCP の実装に関する変更は行っていない。

5.2 Ethernet での実験結果

まず、100 [Mbps] のリンク帯域を持つ Ethernet リンクで両ホストを接続したネットワークを用いて、1 本の TCP コネクションを設定してデータ転送を行った場合の実験結果を示す。 転送実験では、転送するデータサイズを 500 [MB] に固定し、データ転送時間を計測することによって転送速度を算出する。図 7 は従来方式と提案方式を用いた場合のソケットバッファサイズに対する転送速度の変化を表している。図より、Ethernet リンクを用いた場合では、提案方式によるデータ転送速度が、従来方式と比較して僅かしか向上していないことが分かる。これは、エンドホストのデータ転送処理速度に対してネットワークが低速であり、ネットワークがデータ転送の際

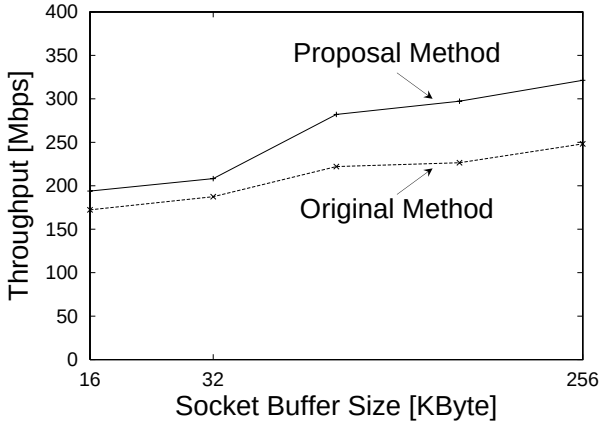

図 8: MAPOS におけるデータ転送実験結果

のボトルネックとなっているため、送信側ホストにおける処理速度を向上させる提案方式の有効性が大きく現れないためである。また、ソケットバッファサイズを変化させてもスループットに変化がないのは、Ethernet リンクにおける帯域遅延積が小さいので、最大スループットを得るために必要な送信ソケットバッファサイズが小さいためである。

5.3 MAPOS ネットワークでの実験結果

次に、622 [Mbps] のリンク帯域を持つ MAPOS リンクを用いた場合 (図 6) の性能評価実験の結果を示す。

まず、MAPOS リンク上にホスト間で 1 本の TCP コネクションを張り、 5.2 節における実験と同様に 2 台のホスト間で TCP によるデータ転送を行い、転送速度の計測を行った。

図 8 は従来方式と提案方式を用いた場合のソケットバッファサイズに対する転送速度の変化を表した図である。図より、提案方式によるデータ転送速度が、従来方式と比較して約 30%向上していることがわかる。これは、エンドホストの処理速度に対してネットワークが非常に高速であるため、メモリコピーを 1 度省略する提案方式の効果が大きく現れているためである。

次に、両端末のバッファサイズを固定し、転送するファイルサイズを変化させた場合の実験結果を図 9 (従来方式) 及び図 10 (提案方式) に示す。図から、ソケットバッファサイズが小さい場合には、エンドホストがデータ転送におけるボトルネックとなるため、転送速度が低下していることが分かる。そのため、バッファサイズを大きくするにつれてデータ転送速度は増加している。また、転送データサイズが大きくなるにつれて、TCP のウインドウサイズが大きい状態でデータ転送を行う時間が相対的に大きくなり、それに対してコネクション設定等のオーバーヘッドがデータ転送時間に対して相対的に小さくなるため、スループットが向上している。

また図より、提案方式における、転送データサイズの増加及びバッファサイズの増加に対するスループットの向上の割合が、従来方式に比べて高くなっていることも分かる。特にバッファサイズが大きくなった場合のスループット向上率が高い。これは、データを転送する際の最初のメモリコピーが省かれているため、送信ソケットバッファにデータを格納するまでの時間が小さくなるので、転送データサイズの増加によって全体の転送時間が長くなると、転送処理速度を向上させる方式である提案方式の効果がより大きくなるためである。

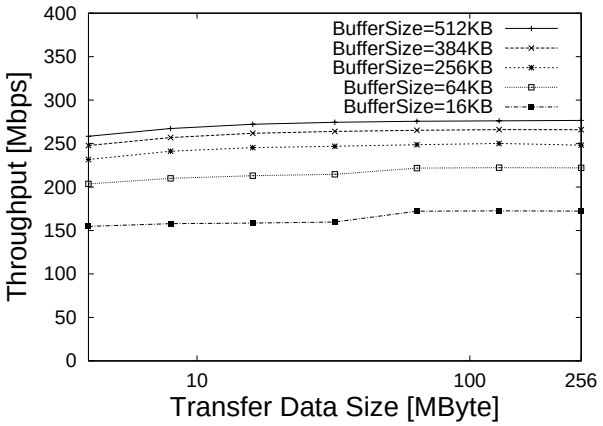

図 9: 従来方式による MAPOS でのデータ転送実験結果

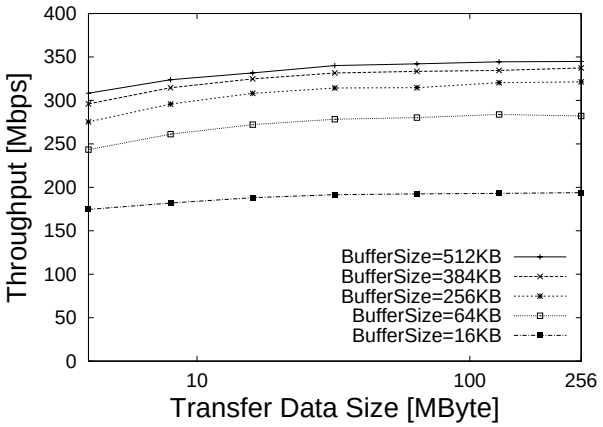

図 10: 提案方式による MAPOS でのデータ転送実験結果

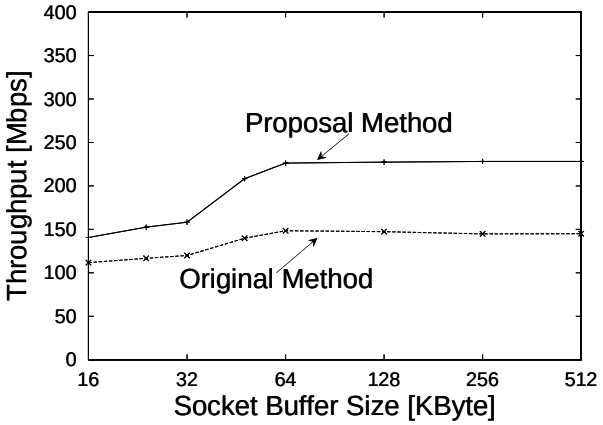

図 11: 送信側マシンの CPU が低速な場合の MAPOS でのデータ転送実験結果

最後に、送信側に低速なマシン 2 を用いた場合の実験結果を図 11 に示す。図から、提案方式によるデータ転送では、従来方式によるデータ転送と比較して約 50%の向上が見られることがわかる。図 8 で示した実験結果よりスループットの向上率が高いのは、送信側ホストが低速であるために、データ転送の際のメモリコピーに要する時間が図 8 の場合と比べて大きく (表 2 参照)、エンドホストにおけるデータ転送処理時間に対するメモリコピーの時間の割合が大きいために、メモリコピー回数の削減の効果がより大きくなるためである。このことから、送信側のホストが低速であるほど、提案方式におけるメモリコピーの回数を削減することの効果が大きく現れることがわかる。

6 おわりに

本稿では、TCP によるデータ転送のためのエンドホストにおける処理の高速化を目的とし、まず TCP によるデータ転送を行う際のエンドホストにおける処理の分析を行い、CPU 処理等と比較して、メモリアクセスの処理に非常に大きな時間を要していることを示した。そして、エンドホストの通信処理軽減手法として、TCP によるデータ転送時のメモリアクセスを削減する処理方式の提案を行った。なお、本稿で提案した方式は文献 [10] で提案されている SABT (Scalable Automatic Buffer Tuning) 方式において使用され、良好に動作することが確認されている。

本稿では、2 台のマシンを直接ネットワークリンクで接続した簡単なネットワークトポロジーを用いたが、今後の課題として、スイッチ、ルータ等を用いて、より複雑なネットワークトポロジーにおけるデータ転送実験を行い、提案方式の性能評価を行うことが挙げられる。

謝辞

本研究の一部は、日本学術振興会未来開拓学術研究推進事業における研究プロジェクト「高度マルチメディア応用システム構築のための先進的ネットワークアーキテクチャの研究」、科学技術庁の平成 10 年度科学技術振興調整費による「高度医療ネットワークに関する研究調査」、通信・放送機構「次世代広帯域ネットワーク利用技術の研究開発プロジェクト」、及び (財) 電気通信普及財団の研究助成「超高速ネットワークのためのトランスポート層プロトコルに関する研究」によっている。ここに記して謝意を表す。

参考文献

[1] K. Murakami and M. Maruyama, “MAPOS - Multiple Access Protocol over SONET/SDH Version 1 etc.,” *RFC 2171-2176*, June 1997.

[2] T. V. Lakshman and U. Madhow, “Performance Analysis of Window–based Flow Control Using TCP/IP: Effect of High Bandwidth–Delay Product and Random Loss,” in *Proceedings of HPN’94*, pp. 135–149, June 1994.

[3] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, *The Design and Implementation of the 4.4BSD Oparating System*. Reading, Massachusetts: Addison-Wesley, 1996.

[4] W. R. Stevens, *TCP/IP Illustrated, Volume1: The Protocols*. Reading, Massachusetts: Addison-Wesley, 1994.

[5] X. Xiao, L. Ni, and W. Tang, “Benchmarking and Analysis of the User-Perceived Performance of TCP/UDP over Myrinet,” *Tech. Rep., Michigan State Univ.*, December 1997.

[6] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen, “An Analysis of TCP Processing Overhead,” *IEEE Communications Magazine*, pp. 23–29, June 1989.

[7] lmbench Home Page, available at `http://www.bitmover.com/lmbench/`.

[8] G. R. Wright and W. R. Stevens, *TCP/IP Illustrated, Volume2: The Implementation*. Reading, Massachusetts: Addison-Wesley, 1995.

[9] P. Druschel and L. L. Peterson, “Fbufs: a High-bandwidth Cross-domain Transfer Facility,” in *Proceedings of the Fourteenth ACM symposium on Operating Systems Principles*, pp. 189–202, December 1993.

[10] 松尾 孝広、長谷川 剛、村田 正幸、宮原 秀夫, “スケーラビリティおよび公平性を考慮した TCP バッファ制御,” **電子情報通信学会技術研究報告** (SSE99-167), pp. 37–42, March 2000.