特別研究報告

題目

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

指導教官

村田 正幸 教授

報告者

寺井 達彦

平成 12 年 2 月 23 日

大阪大学 基礎工学部 情報科学科

平成 11 年度 特別研究報告

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

内容梗概

近年、インターネットは急速に発展を遂げ、多くの情報が WWW (World Wide Web) から得られるようになっており、インターネット利用者が急増している。それに伴うネットワークトラヒックの増加に対して、例えば、WDM (Wavelength Division Multiplexing) を始めとするさまざまなデータ転送技術によってネットワークの高速化が図られるとともに、ネットワークの輻輳に対しては TCP (Transmission Control Protocol) 等のトランスポート層プロトコルの輻輳制御方式に対する多くの改善案が検討されている。しかし、これまでは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これまであまり検討がなされていなかった。しかしながら、ネットワークの高速化に伴い、ネットワークではなくエンドホストがボトルネックとなる状況がすでに生まれつつあり、今後のデータ転送技術のさらなる発展を考え併せると、エンドホストの高速化技術の重要性はますます高まっている。

そこで本報告においては、現在のインターネットによるデータ転送で使用されている主要なプロトコルであり、HTTP (Hyper Text Transfer Protocol) や FTP (File Transfer Protocol)、telnet 等を実現しているトランスポート層プロトコルである TCP に着目し、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を分析し、実コンピュータの仕様に基づいてデータ転送における処理ボトルネックを調べた。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く、データ転送を行う際に必要な時間のうち最も大きな部分を占めることを示した。

すなわち、TCP によるデータ転送を行う際のメモリアクセス回数の削減がデータ転送の高速化のためにもっとも重要であり、それを実現する新たなデータ転送処理方式の提案を行った。本報告ではさらに、提案する処理方式を実コンピュータ上に実装し、従来の処理方式との比較実験による性能評価を行った。その結果、提案する処理方式によって、従来方式に比べてデータ転送速度を約 30%高速化できることを示した。

主な用語

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

目 次

図 目 次

表 目 次

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) [2] や MAPOS (Multiple Access Protocol over SONET/SDH) [3] に代表される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における Web ドキュメント転送で主に用いられている HTTP (Hyper Text Transfer Protocol) [4] や、ファイル転送で主に用いられている FTP (File Transfer Protocol) [5] 等を実現しているトランスポート層プロトコルである TCP (Transmission Control Protocol) [6] 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 [7-11]。

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能力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、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。

そこで本報告では、最大 622 [Mbps] でのデータ転送が可能な高速ネットワークである MAPOS ネットワークを用いることにより、ネットワークではなくエンドホストのデータ転送処理がボトルネックとなる状況を想定し、TCP によるデータ転送の高速化を図り、データ転送実験によってその妥当性を検証する。そのためにまず、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。

以下、 2 章において、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に関する説明を行う。 3 章では、TCP によるデータ転送時のエンドホストの様々な処理に要する時間の分析を行い、データ転送処理のボトルネックについての考察を行う。 4 章では、TCP によるデータ転送速度の向上のための TCP の改良について検討を行う。 5 章では、TCP によるデータ転送時のメモリアクセス回数を削減する処理方式の提案を行い、その性能評価実験を行う。最後に 6 章で本報告の結論を述べる。

2　エンドホストにおけるプロトコル処理機構の概略

コンピュータネットワークにおいては、通信に関わるすべての処理を単一のプロトコルによって行わず、複数のプロトコルを用いて処理を行う [12]。その際、複数のプロトコルを階層的に構成し、ある層のプロトコルと隣接する階層のプロトコルの間に明確なインターフェースを規定している。そのため、隣接するプロトコルは互いに依存せず、それぞれの階層で適切なプロトコルを独立に選択して通信を行うことが可能となっている。

代表的なプロトコルスタックの参照モデルとして、OSI 参照モデル [12] と TCP/IP 参照モデル [12] が存在する。OSI 参照モデルは、国際標準化機構 (ISO) によって提案されたもので 7 層で構成されている。一方、TCP/IP 参照モデルは ARPANET (Advanced Research Project Agency NETwork) で開発され、後に標準として採用されたモデルで、現在のインターネットにおける事実上の標準プロトコルモデルとなっている。TCP/IP 参照モデルは 4 層で構成されている。OSI 参照モデルと TCP/IP 参照モデルの概略を図 1 に示す。本報告においては、TCP/IP 参照モデルに基づくシステムを対象とする。

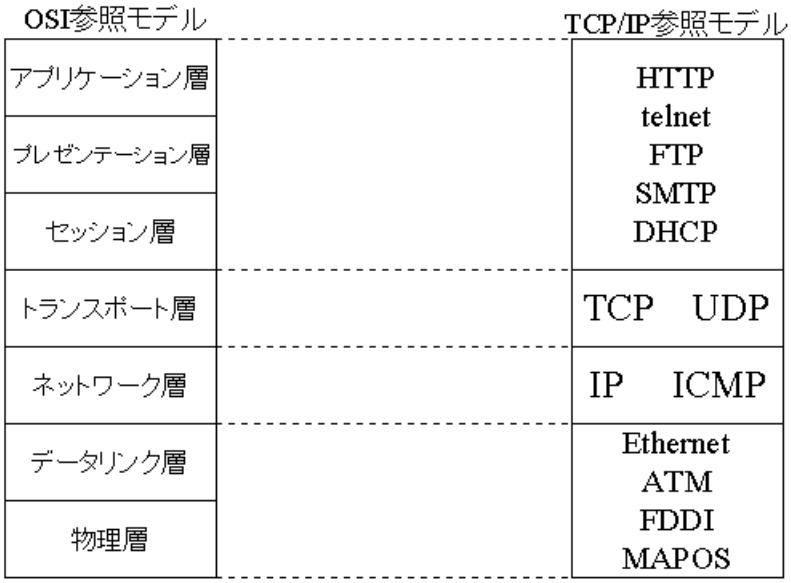

図 1: TCP/IP プロトコル

また、本報告では、API (Application Program Interface) としてソケットインターフェース [13] 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。本報告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 [14] 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 2 を用いて各層ごとに順に説明を行う。データ送信の際の処理は以下のようになる ([15] 参照)。

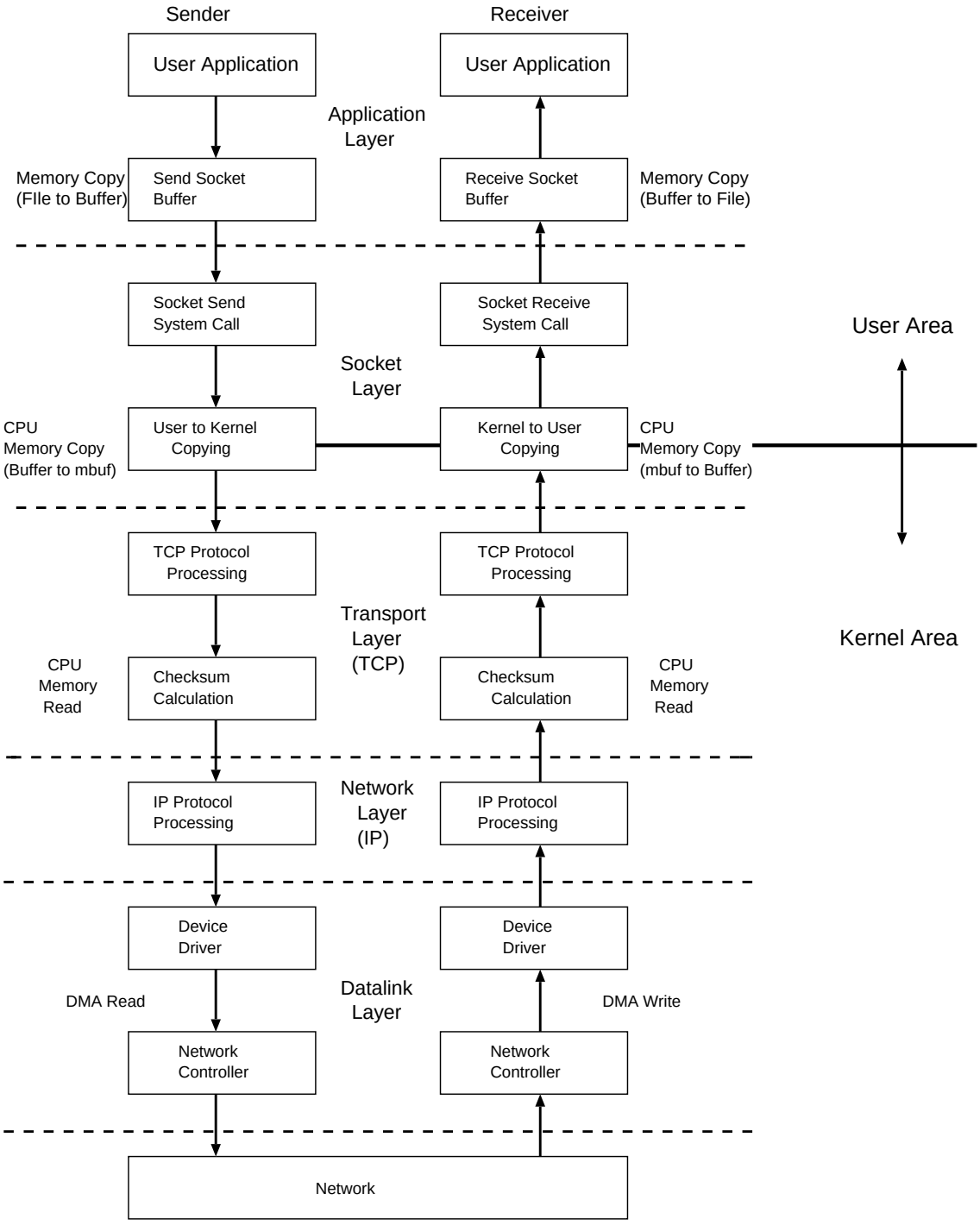

図 2: エンドシステムにおけるプロトコル処理機構

- **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 のプロトコル処理を行う（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 2 に示すように 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おいては送信側と同様の処理を逆の手順で行う。

次章においては、CPU による TCP/IP のプロトコル処理とメモリアクセス方式を説明し、本章で説明を行った各処理に必要な時間の分析を行い、TCP によるデータ転送時の処理のボトルネックがどこにあるかを明らかにする。

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

本章では、データ転送時の TCP/IP の処理とメモリアクセス機構に関しての詳細な分析を行い、その分析結果と、実際のマシンの CPU 速度およびメモリアクセス速度を基にして、転送を行う際の各処理に必要な時間を推定し、その比較を行う。

3.1 TCP/IP のプロセッシング命令数

文献 [1] 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 ソースコードの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 [16]。本報告では、文献 [1] の結果を用いて、TCP/IP の処理に必要な時間を算出する。図 3 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。

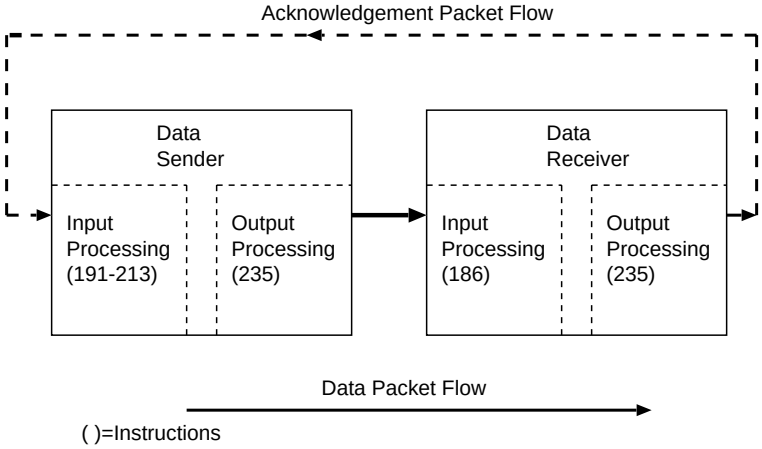

図 3: TCP の処理行程 (文献 [1])

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 2 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 TCP/IP 通信における送受信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 [17] 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できる。表 1 に本報告で用いた 2 種類のマシンのベンチマークテストの結果を示す。表より、CPU 速度が低下するとそれに従ってメモリアクセスに関わる速度も低下することが分かる。

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	Pentium-III Xeon 550MHz	Pentium-II 266MHz
Main Memory [KB]	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 に送信側ホストによる処理時間の推定結果を、表 3 に受信側ホストによる処理時間の推定結果をそれぞれ示す。また、表 4 にはマシン 2 を送信側ホストにした場合の処理時間の推定結果を示している。なお、DMA は CPU のプロセッシング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 4 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 [18]、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。

表 2: 送信側ホストにおける処理時間の推定結果

処理内容	導出過程	処理時間 (μs)
Memory Copy (file to buffer)	4 [MB]/180.32 [MB/sec]	22182
Memory Copy (buffer to mbuf)	4 [MB]/180.32 [MB/sec]	22182
TCP output (data pkt)	235 [ins] $\times$ 258 [pkt]/550M	105
Memory Read (for checksum)	4 [MB]/383.12 [MB/sec]	10441
IP output (data pkt)	61 [ins] $\times$ 258 [pkt]/550M	27
IP input (ack pkt)	57 [ins] $\times$ 258 [pkt]/550M	25
TCP input (ack pkt)	213 [ins] $\times$ 258 [pkt]/550M	91
合計		55053

表 3: 受信側ホストにおける処理時間の推定結果

処理内容	導出過程	処理時間 (μs)
IP input (data pkt)	57 [ins] $\times$ 258[pkt]/550M	25
TCP input (data pkt)	186 [ins] $\times$ 258[pkt]/550M	83
Memory Read (for checksum)	4 [MB]/383.12 [MB/sec]	10441
Memory Copy (mbuf to buffer)	4 [MB]/180.32 [MB/sec]	22182
TCP output (ack pkt)	235 [ins] $\times$ 258[pkt]/550M	105
IP output (ack pkt)	61 [ins] $\times$ 258[pkt]/550M	27
Memory Copy (buffer to file)	4 [MB]/180.32 [MB/sec]	22182
合計		55045

表 4: マシン 2 を送信側ホストにした場合の処理時間の推定結果

処理内容	導出過程	処理時間 (μs)
Memory Copy (file to buffer)	4 [MB]/51.98 [MB/sec]	76953
Memory Copy (buffer to mbuf)	4 [MB]/51.98 [MB/sec]	76953
TCP output (data pkt)	235 [ins] $\times$ 258 [pkt]/266M	217
Memory Read (for checksum)	4 [MB]/203.54 [MB/sec]	19652
IP output (data pkt)	61 [ins] $\times$ 258 [pkt]/266M	56
IP input (ack pkt)	57 [ins] $\times$ 258 [pkt]/266M	53
TCP input (ack pkt)	213 [ins] $\times$ 258 [pkt]/266M	197
合計		174081

4 データ転送速度の向上のための TCP の改良

本章においては、TCP による高速データ通信を実現するための改良方法として、TCP の Delayed ACK オプション及びソケットバッファサイズに関しての検討を行う。

4.1 Delayed ACK オプション

Delayed ACK オプション [19] は、TCP のオプション機能のひとつであり、受信したデータパケットすべてに対して ACK パケットを返送せず、FreeBSD の実装では最大 200 [ms] 遅延させて返送する機能のことである。また、FreeBSD の実装においては 2 個のデータパケットを受信した場合にも、即座に ACK パケットを返す。Delayed ACK オプションを使用する目的として、

- ACK パケットを送出するタイミングを遅らせることによって、ACK を返す方向に送出されるデータを ACK パケットと一緒に送信する (piggyback)。
- 受信したデータパケットすべてに対して ACK パケットを送るのは、エンドホストの処理の増加、またネットワークの輻輳につながるため、ACK パケットを送る回数を減少させる。

等が挙げられる [14]。図 4 はそれぞれ Delayed ACK を行う場合と行わない場合のデータ転送の概略図である。図に示すように、データ転送開始直後は送信側端末は 1 つしかデータパケットを送出しないので、そのパケットに対する ACK パケットが最大 200 [ms] 遅れて送信される。従って、転送データサイズが小さい場合、転送時間が不必要に大きくなる可能性がある。特に、高速ネットワークにおいては全体の転送時間が短いため、Delayed ACK による遅延時間が無視できない。そこで、本節では、Delayed ACK オプションの有無が転送速度に与える影響を評価する。

図 5 は、100 [Mbps] のリンク帯域を持つ Ethernet リンクを用いて、2 台のマシンを接続し、TCP によるデータ転送を行う際に Delayed ACK オプションを使用した場合と使用しない場合の、転送データサイズに対する転送時間の変化を示したものである。図より、転送

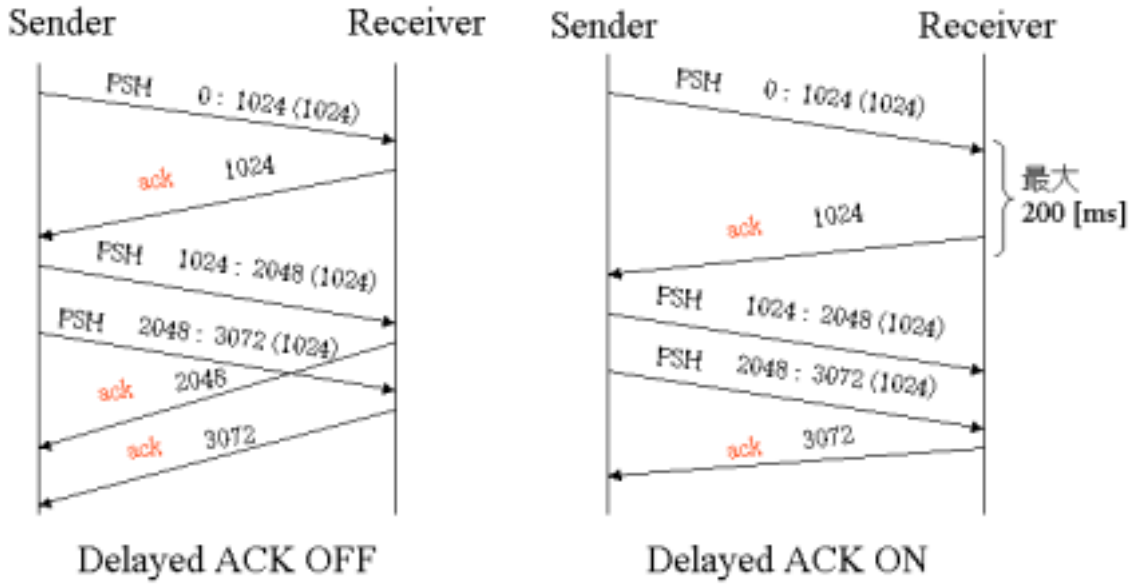

図 4: Delayed ACK によるパケットの送受信の様子

データサイズが 1 [KB] の場合、Delayed ACK オプションの有無に関わらずほぼ同じ転送時間であることがわかる。これは、Ethernet での転送を行う際に用いられるパケットサイズが 1 [KB] であり、1 度の転送でデータ転送が終了するため、Delayed ACK の影響を受けないためである。また、転送データサイズが、パケットサイズを越えると Delayed ACK の影響を受け、Delayed ACK オプションを使用している場合の転送時間は急激に大きくなっている。これは、Delayed ACK オプションを使用している場合、最初のデータパケットに対する ACK パケットを待つ時間 (図 4 参照) が大きな影響を与えているためである。また、さらに転送データサイズが大きくなると、転送時間の差は小さくなっている。これは、最初のパケットに対する Delayed ACK の影響が、全体の転送時間に対して相対的に小さくなるためである。

次に、同様の実験を 622 [Mbps] のリンク帯域を持つ MAPOS [3] リンクを用いて行った場合の結果を図 6 に示す。MAPOS は、NTT が開発した SONET/SDH 上を HDLC (High-Level Data Link Control) フレーマを用いて通信を行う高性能通信プロトコルである。現在 MAPOS がサポートしているリンク速度は、OC-3c (155Mbps)、OC-12c (622Mbps)、及び

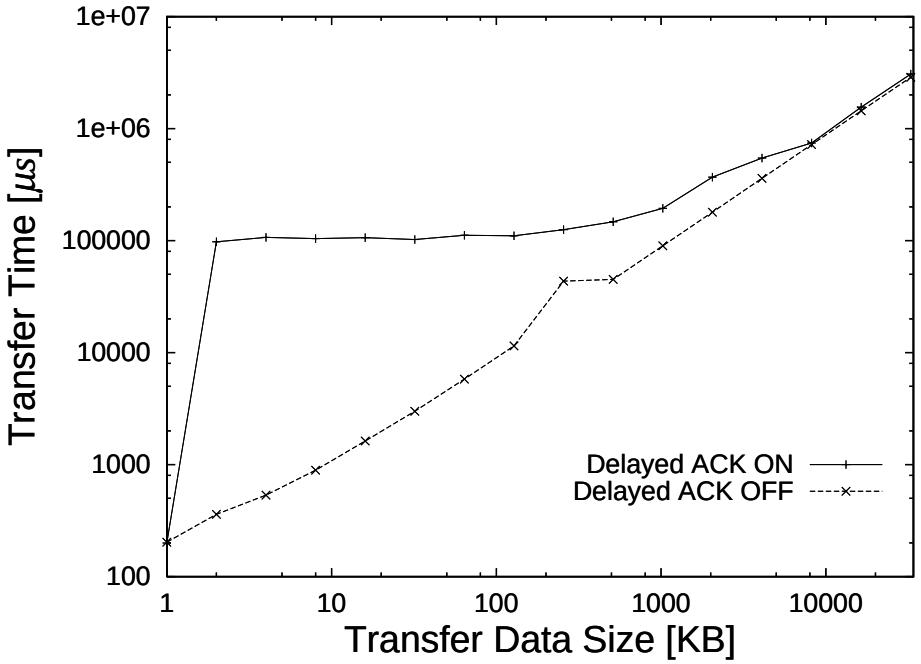

図 5: Ethernet における Delayed ACK の影響に関する性能評価実験結果

OC-48c (2.4Gbps) の 3 種類である。今回の実験では OC-12c を用いてデータ転送実験を行った。図より、転送データサイズが 16 [KB] 以下の場合は、Delayed ACK オプションの有無に関わらず、ほぼ同じ転送時間である。これは、MAPOS での転送を行う際に用いられるパケットサイズが 16 [KB] であり、1 回の転送でデータ転送が終了するために Delayed ACK の影響を受けないためである。転送データサイズが 16 [KB] 以上の場合は、Ethernet の場合 (図 5) と同様に、Delayed ACK による遅延の影響を大きく受け、また転送データサイズが大きくなるにつれて Delayed ACK による遅延の影響が小さくなっている。従って、転送速度向上のために、 5 章に示す性能評価実験では Delayed ACK オプションは使用しないこととする。

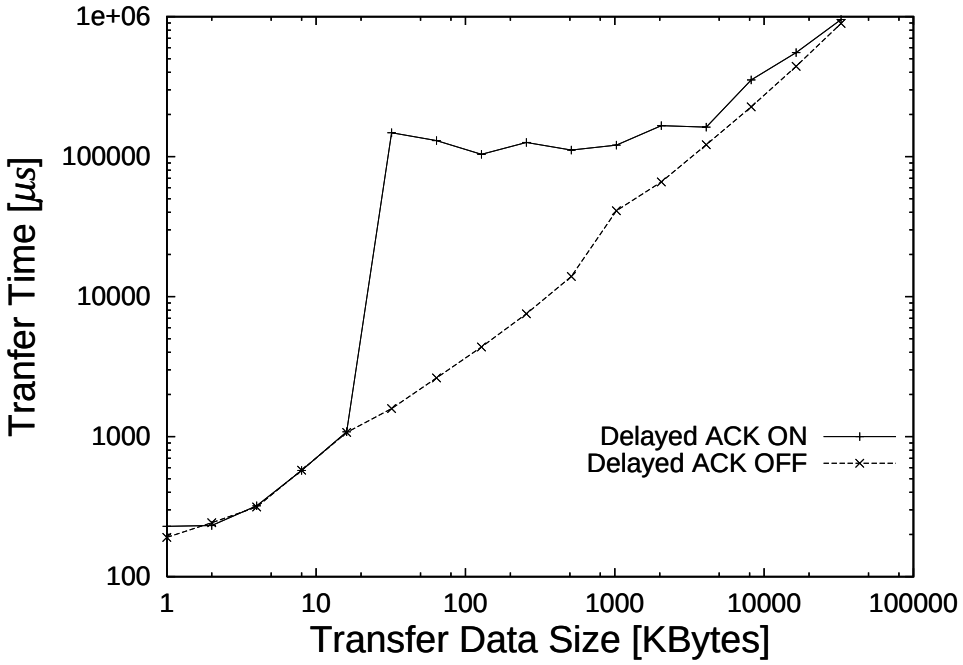

図 6: MAPOS における Delayed ACK の影響に関する性能評価実験結果

4.2 ソケットバッファサイズ

TCP によるデータ転送の際には、TCP のパケット再送機能を実現するために、送信側ホストは対応する ACK パケットを受信するまで、送信済のデータパケットを送信ソケットバッファに保持し続ける必要がある。従って、送信ソケットバッファサイズが小さい場合には、再送用パケットが送信ソケットバッファの大半を占め、新たに送信するデータパケットを送信ソケットバッファに保持することができないため、送出可能なデータパケット数が制限され、スループットが低下する。

従来の TCP/IP の実装においては、高速なネットワークを想定しておらず、TCP コネクション設定時の送信ソケットバッファサイズの既定値は小さな値に設定されている。例えば FreeBSD-2.2.8 の実装においては 16 [KB] に設定されており、高速なネットワークに対しては送信ソケットバッファが不足する。本節では、622 [Mbps] の帯域を持つ MAPOS リンクを用いて、送信バッファサイズが MAPOS リンク上の TCP コネクションのスループット

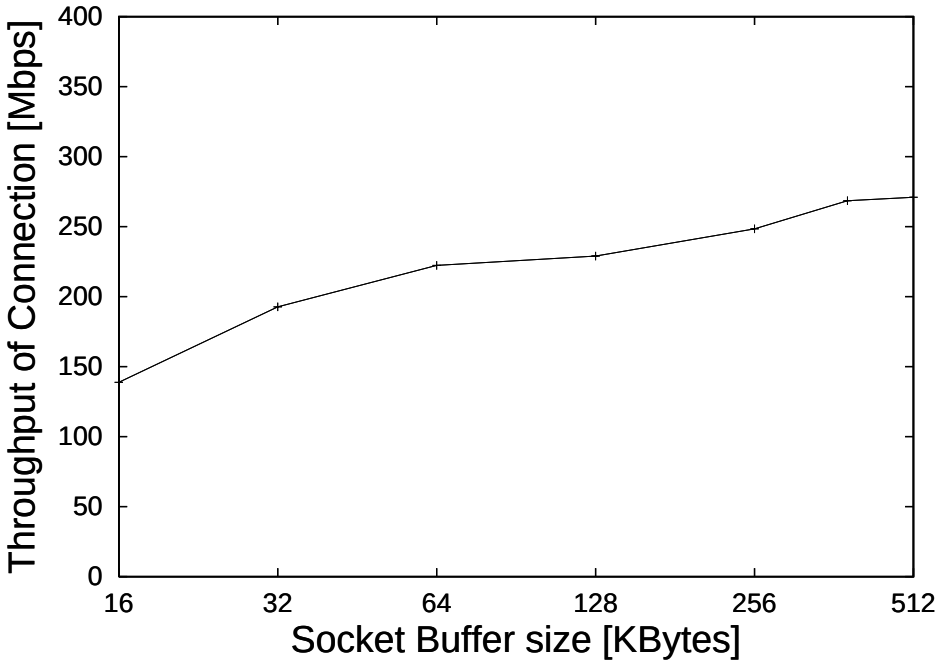

図 7: ソケットバッファサイズを変化させた場合のデータ転送実験結果

に与える影響をデータ転送実験を行うことによって評価する。図 7 にソケットバッファサイズに対する転送速度の変化を示す。なお、ここでは転送するデータサイズを 500 [MB] に固定している。

図 7 に示されるように、FreeBSD-2.2.8 の既定値である 16 [KB] を用いた場合は、送信ソケットバッファサイズが不足し、十分なスループットが得られていない。また、送信ソケットバッファサイズを大きく設定するにつれスループットが向上し、ソケットバッファサイズが 384 [KB] 以上になるとスループットの向上は見られなくなる。これは送信ソケットバッファサイズが十分に大きくなると、上記で説明したような送信ソケットバッファサイズがボトルネックになるような状況が解消されるためである。TCP による転送を行う際に必要となるソケットバッファサイズは、TCP コネクションの帯域遅延積 (bandwidth delay product) の 2 倍であることが文献 [20] より明らかになっている。この実験における MAPOS リンクの RTT (Round Trip Time) は約 0.01 [s] であるので、最大スループットである 270 [Mbps] を得るために必要な送信ソケットバッファサイズは、

$$\text{必要なソケットバッファサイズ} = 270\ [\text{Mbps}] \times 0.01\ [\text{s}] = 345.6\ [\text{KB}] \qquad (1)$$

と求められる。(1) 式により得られた結果は、実験により得られている結果 (図 7) とほぼ一致している。

5　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システム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。

5.1　従来方式による冗長なメモリコピー

図 8 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 [21] 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。

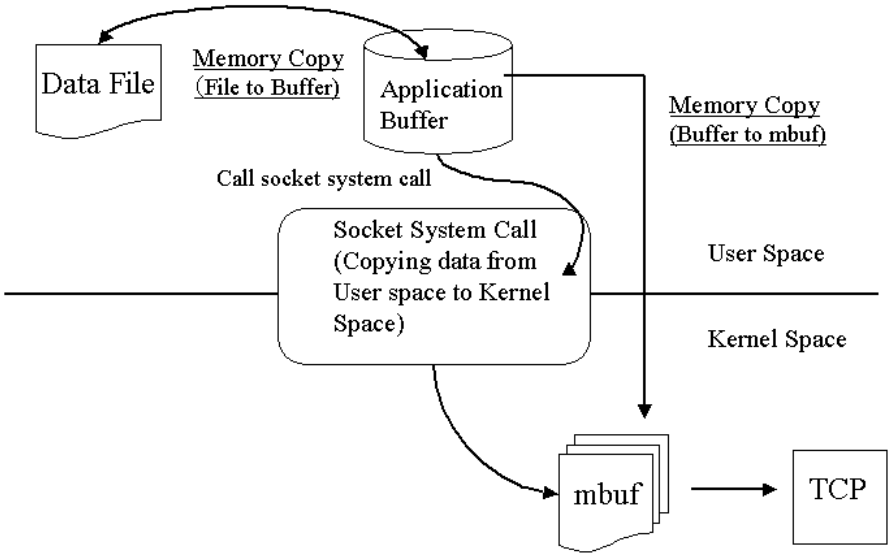

図 8: 従来方式における TCP データ転送の際の処理方式

5.2 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 [22] 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。本報告では、 5.1 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 9 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

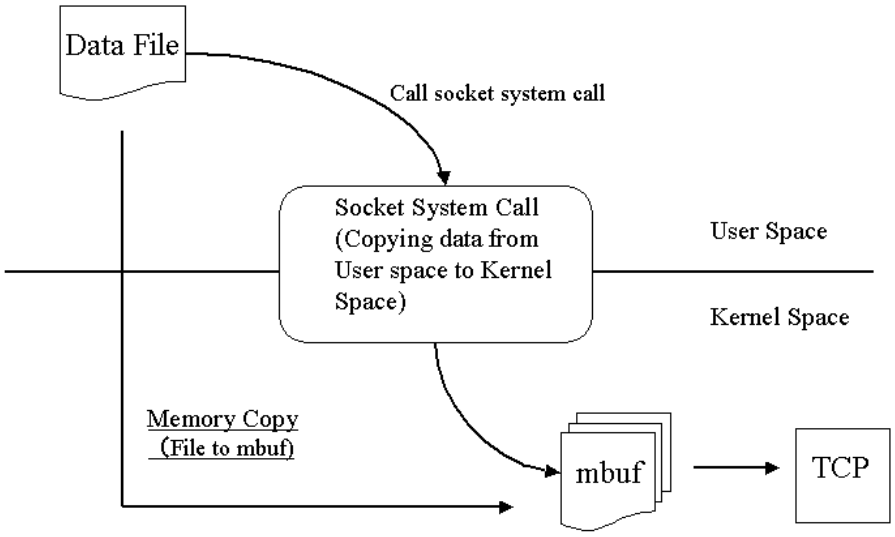

図 9: 提案方式におけるエンドシステム処理

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、 5.1 節で述べた冗長なメモリコピーを省略する。以降、FreeBSD [23] 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 10 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

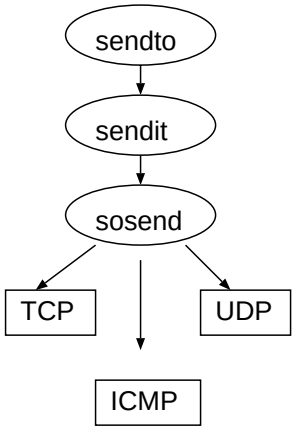

図 10: ソケットシステムコールの制御の流れ

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。

5.3 性能評価実験

本節では、 5.2 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。

5.3.1 実装環境

実装に使用した OS は、FreeBSD-2.2.8 [23] である。実験に使用したネットワーク環境は、図 11、12 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 (Multipul Access Protocol Over SONET/SDH) [3] 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、CPU が PentiumIII-Xeon 550 [MHz]、メインメモリが 512 [MB] のマシン、及び CPU が Pentium-II 266 [MHz]、メインメモリが [128MB] のマシンである。

また、Delayed ACK オプション (4.1 節参照) は使用しない。また、それ以外の TCP の実装に関する変更は行っていない。

5.3.2 Ethernet での実験結果

まず、図 11 で示すような 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 13 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。

5.3.3 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 12) 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、5.3.2 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。

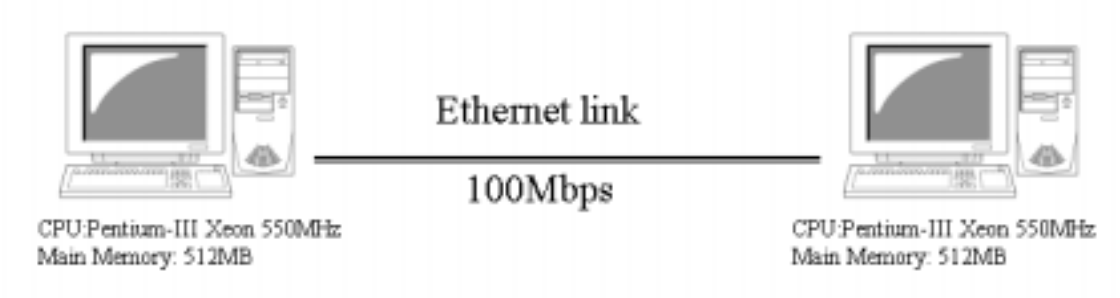

図 11: Ethernet リンクを用いた実験ネットワーク環境

図 12: MAPOS リンクを用いた実験ネットワーク環境

図 14 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 15 (従来方式) 及び図 16 (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

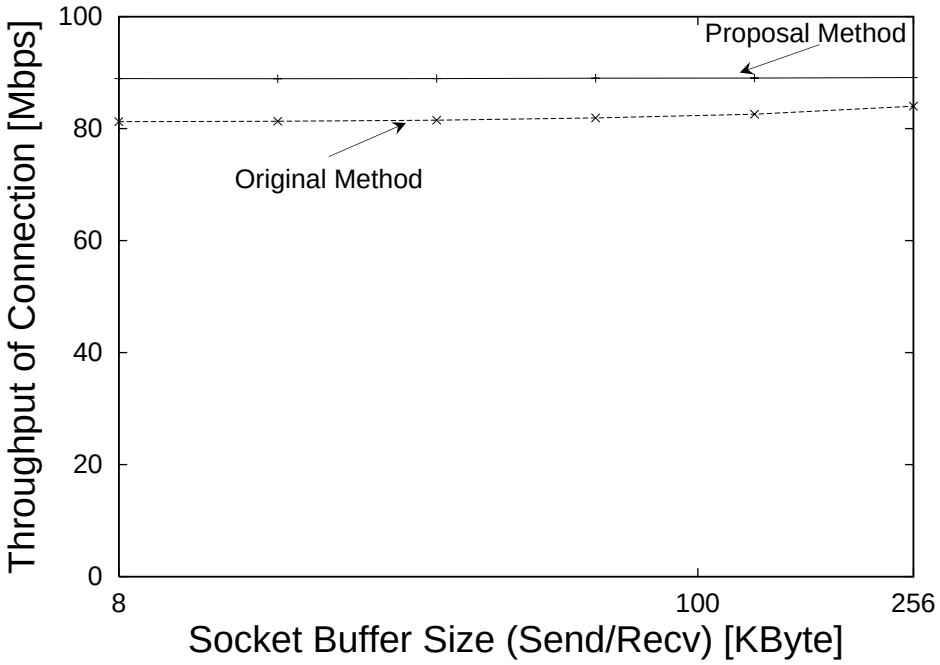

図 13: Ethernet におけるデータ転送実験結果

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。

最後に、図 17 に示すように、送信側に CPU 速度の小さいマシンを用いた場合の実験結果を示す。図 18 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 14 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 14 の場合と比べて大きく (表 4 参照)、エンドホストにおけ

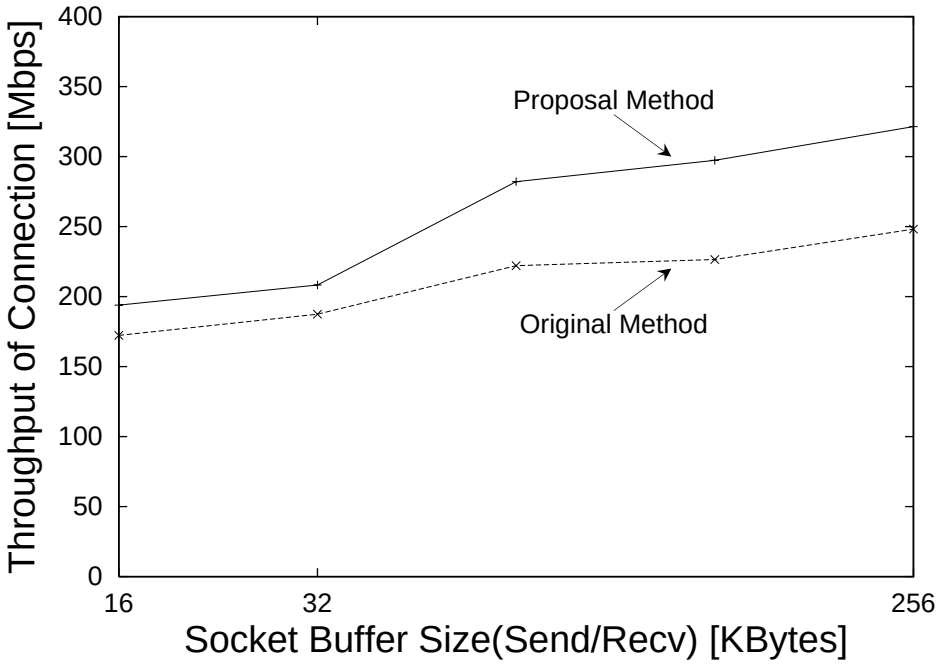

図 14: MAPOS におけるデータ転送実験結果

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。

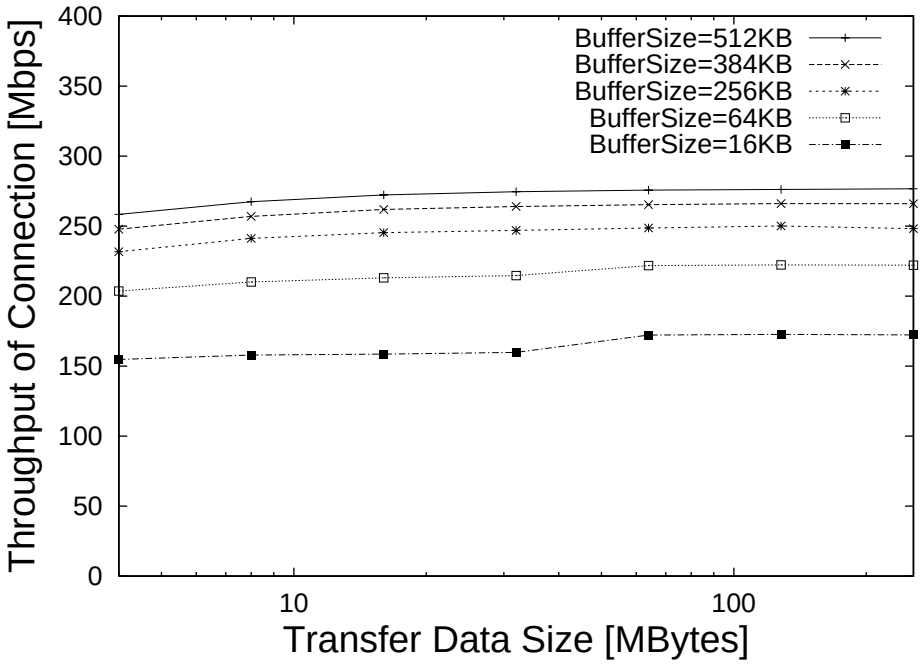

図 15: 従来方式による MAPOS でのデータ転送実験結果

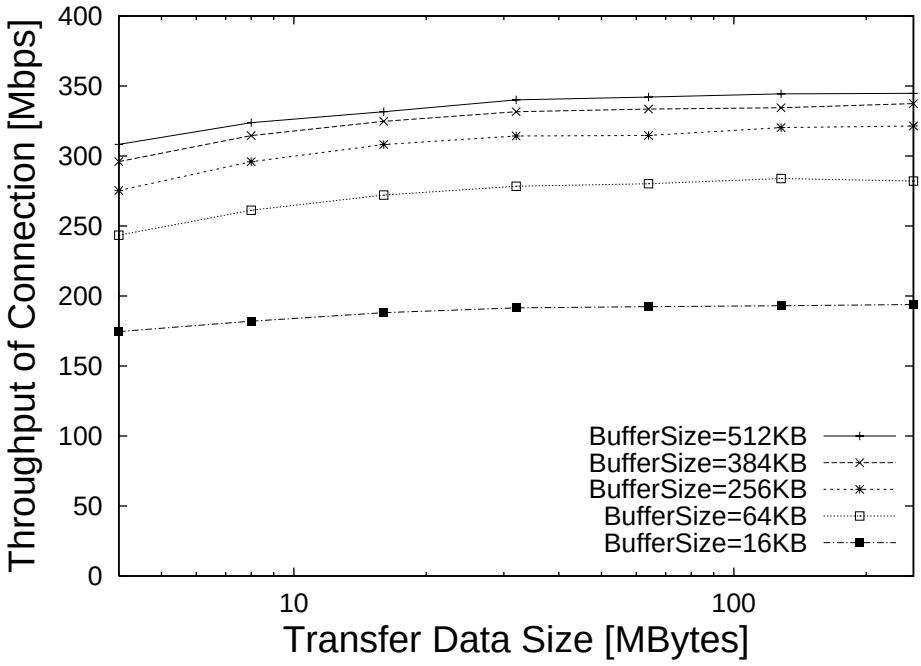

図 16: 提案方式による MAPOS でのデータ転送実験結果

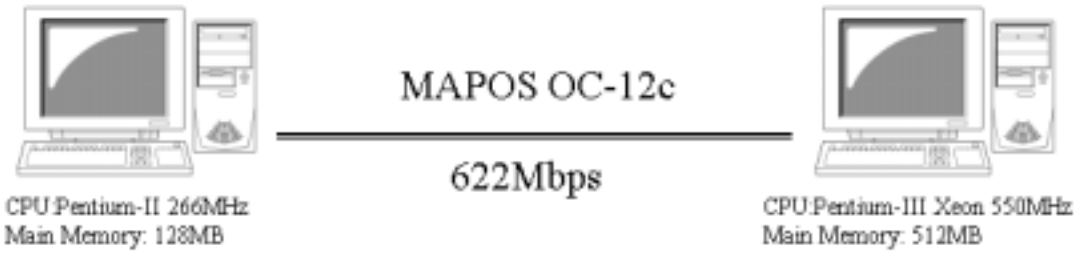

図 17: 送信側に低速なマシンを用いたネットワーク環境

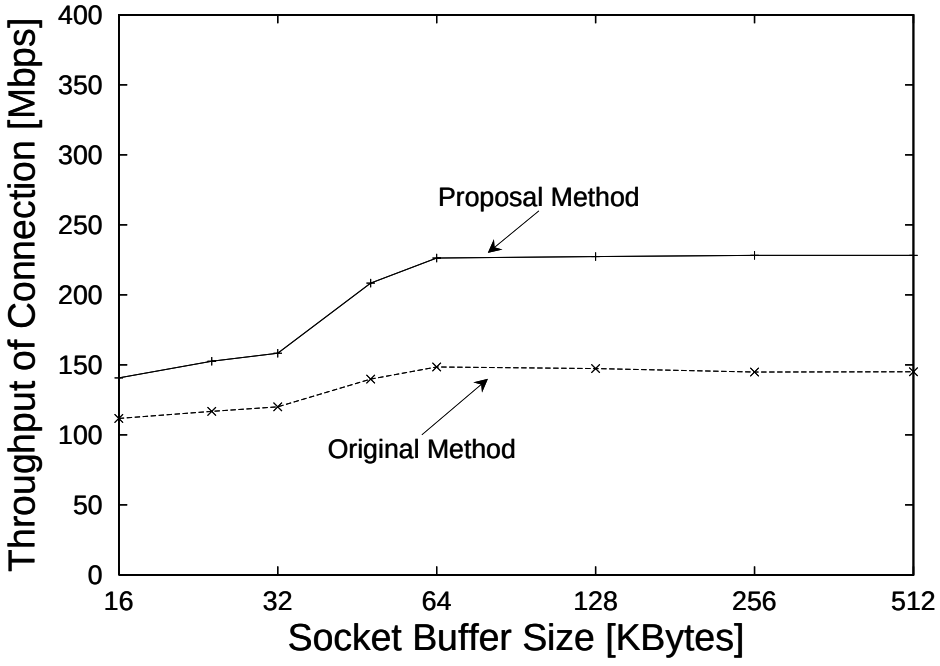

図 18: 送信側マシンの CPU が低速な場合の MAPOS でのデータ転送実験結果

6　おわりに

本報告では、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。また、提案方式による処理を実コンピュータ上に実装し、従来の処理方式との比較実験による性能評価を行い、その結果、従来方式に比べてデータ転送速度を約 30%高速化できることを示した。

なお、本報告で提案した方式は文献 [24] で提案されている SABT (Scalable Automatic Buffer Tuning) 方式において使用され、良好に動作していることが確認されている。

本報告では、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

本報告を終えるにあたり、村田正幸教授より御指導、御教授頂いたことに心より感謝致します。また、本報告において長谷川剛助手には日頃から熱心な御指導を授かり心からお礼を申し上げます。本報告を行うにあたって、大阪大学大学院基礎工学研究科の宮原秀夫教授、大阪府立看護大学医療技術短期大学部の菅野正嗣助教授、大阪大学大型計算機センターの馬場健一助教授、大阪大学大学院基礎工学研究科の若宮直紀講師、大阪大学情報処理教育センターの大崎博之助手からも御指導及び御助言を頂き深く感謝の意を申し上げます。さらに本報告を行うにあたって御助言及び御協力頂いた松尾孝広氏に深く感謝致します。

最後になりましたが、御協力頂いた大阪大学基礎工学部情報科学科村田研究室の皆様に心より感謝致します。

参考文献

[1] David D. Clark, Van Jacobson, John Romkey and Howard Salwen, "An Analysis of TCP Processing Overhead," *IEEE Communications Magazine*, pp. 23–29, June 1989.

[2] Jon Anderson, James S. Manchester, Antonio Rodriguez-Moral and Malathi Veeraraghavan, "Protocols and Architectures for IP Optical Networking," *Bell Labs Technical Journal*, January–March 1999.

[3] Ken'ichiro Murakami and Mitsuru Maruyama, "MAPOS - Multiple Access Protocol over SONET/SDH Version 1 etc.," *RFC 2171-2176*, June 1997.

[4] J. Franks, P. Hallam-Baker, J. Hostetler, S. Lawrence, P. Leach,A. Luotonen and L. Stewart, "Hypertext Transfer Protocol – HTTP/1.1," *RFC 2616*, June 1999.

[5] J. Postel and J. Reynolds, "File Transfer Protocol (FTP)," *RFC 959*, October 1985.

[6] J. B. Postel, "Transmission Control Protocol," *RFC 793*, September 1981.

[7] D. D. Clark, V. Jacobson, J. Romkey, and H. Salwen, "An Analysis of TCP Processing Overhead," *IEEE Communications Magazine*, vol. 27, pp. 23–29, June 1989.

[8] Z. Wang and J. Crowcroft, "Eliminating Periodic Packet Losses in 4.3–Tahoe BSD TCP Congestion Control," *ACM Computer Communication Review*, vol. 22, pp. 9–16, April 1992.

[9] 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.

[10] J. Hoe, "Start-up Dynamics of TCP's Congestion Control and Avoidance Schemes," Master's thesis, MIT, June 1995.

[11] V. Paxson, *Measurements and Analysis of End-to-End Internet Dynamics*. PhD thesis, University of California, Berkeley, April 1997.

[12] Andrew S. Tanenbaum, *Computer Networks Third Edition*. Upper Saddle Riverm New Jersey, 07458: Prentice Hall, 1996.

[13] Marshall Kirk McKusick, Keith Bostic, Michael J. Karels and John S. Quarterman, *The Design and Implementation of the 4.4BSD Oparating System*. Reading, Massachusetts: Addison-Wesley, 1996.

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

[15] Xipeng Xiao, Lionel Ni and Wenting Tang, "Benchmarking and Analysis of the User-Perceived Performance of TCP/UDP over Myrinet," *Tech. Rep., Michigan State Univ.*, December 1997.

[16] V. Jacobson, "Congestion Avoidance and Control," in *Proceedings of ACM SIGCOMM '88*, pp. 314–329, August 1988.

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

[18] E. Nahum, D. Yates, J. Kurose, and D. Towsley, "Cache Behavior of Network Protocols," in *Proceedings of ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems*, pp. 169–180, June 1997.

[19] V. Jacobson and R. Braden, "TCP extensions for long-delay paths," *RFC 1072*, October 1988.

[20] M. Mathis and J. Mahdavi, "Forward Acknowledgment: Refining TCP Congestion Control," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 281–291, October 1996.

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

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

[23] FreeBSD Home Page, available at *http://www.freebsd.org/*.

[24] Takahiro Matsuo, "Scalable Automatic Buffer Tuning to Provide High Performance and Fair Service for TCP Connections," *Master Thesis,Osaka University*, February 2000.