Webサーバの高速・高機能化のための ソケットバッファ管理方式の実装と評価

寺井 達彦† **岡本 卓也**‡ **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@ics.es.osaka-u.ac.jp

‡ 大阪大学基礎工学部情報科学科
〒 560-8531 大阪府豊中市待兼山町 1-3
Phone: 06-6850-6616, Fax: 06-6850-6589
E-mail: tak-okmt@ics.es.osaka-u.ac.jp

§ 大阪大学サイバーメディアセンター
〒 560-0043 大阪府豊中市待兼山町 1-30
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

あらまし ネットワークの輻輳問題に対しては、これまでにも様々な解決策が提案・検討されているが、エンドホストに関してはこれまであまり検討がなされていない。その中で、エンドホストにおける通信処理速度の高速化手法の一つとして、SSBT (Scalable Socket Buffer Tuning) 方式が提案されている。SSBT 方式は、TCP によるデータ転送効率と複数コネクション間の公平性の向上を目的とした動的ソケットバッファ割り当て手法である E-ATBT (Equation-based Automatic TCP Buffer Tuning) 方式 [1]、及び高速データ転送時の通信処理軽減手法である SMR (Simple Memory-copy Reduction) 方式 [2] からなる。本稿では、SSBT 方式を Web サーバが稼働する実コンピュータ上に実装し、トランスポート層レベルでのデータ転送実験、及び Web サーバを用いたアプリケーション層レベルでのデータ転送実験を行うことによって、Web サーバの高速・高機能化についての検討を行っている。その結果、従来の方式に比べ、スループット、及び公平性の両面で優れていることを明らかにしている。

キーワード TCP (Transmission Control Protocol)、WWW (World Wide Web)、ソケットバッファ、公平性、スケーラビリティ

Implementation and Evaluation of SSBT for High-Performance Web Servers

Tatsuhiko Terai† **Takuya Okamoto**‡ **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@ics.es.osaka-u.ac.jp

‡Department of Infomatics and Mathematical Science
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: tak-okmt@ics.es.osaka-u.ac.jp

§Cybermedia Center, Osaka University
1-30, Machikaneyama, Toyonaka, Osaka 560-8531, Japan
E-mail: {hasegawa, murata}@cmc.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. We have already proposed a new architecture, which is called Scalable Socket Buffer Tuning (SSBT), to provide high performance and fair service for many TCP connections at the Internet endhosts. In this paper, we evaluate an effectiveness of our proposed algorithm through various kinds of implementation experiments. In the experiments, we confirm the behavior and effectiveness of the SSBT through both transport-layer benchmark results and application-layer benchmark results. In the transport-layer tests, we use the native TCP data transfer to evaluate SSBT performance. In the application-layer tests, we use HTTP data transfer from the Web server which transfer the data by TCP, considering the Web traffic on the Internet.

Keyword TCP (Transmission Control Protocol), WWW (World Wide Web), Socket Buffer, Fairness, Scalability

1 はじめに

近年、インターネットは急速に発展し、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よって引き起こされるネットワークの輻輳を解消するために、これまでに様々な解決案が提案・検討されている。一方、エンドホストにおけるデータ転送処理の高速・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に、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想定されなかったためである。しかし、現在は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加したため、エンドホストの処理がデータ転送処理のボトルネックとなりつつある。例えば、アクセス頻度の高い Web サーバではピーク時に毎秒数百のドキュメントに対するリクエストを処理しなければならなくなっており、ネットワークではなくエンドホスト自身がデータ転送全体のボトルネックになる状況が発生している。

エンドホストの高速化のためのプロトコル処理の軽減手法としては、例えば [3] 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によって、エンドホストにおける TCP (Transmission Control Protocol) によるデータ転送の際のボトルネックの一つであるメモリコピー処理を省略し、エンドホストの高速化を実現している。しかし、 [3, 4] 等の提案方式の多くは、エンドホストに接続している各コネクションへの有効な割り当てメモリ (バッファ) 量に関する検討や、コネクション間の公平性は考慮されていない。例えば、サーバホストに 64 [Kbps] のリンクで接続しているクライアント A と、100 [Mbps] のリンクで接続しているクライアント B が存在する時に、これら 2 つのコネクションに送信ソケットバッファを等分して割り当てると、各コネクションで必要なバッファサイズが異なるため、クライアント A にとってはバッファサイズが大きい、あるいはクライアント B にとってバッファサイズが小さいという状況が発生し得る。このような場合、クライアント A に割り当てられたバッファの内、使用されない余剰バッファをクライアント B に割り当てることによって、双方のコネクションが高スループットを得られる。特に、アクセス頻度の高い Web サーバにおいては、同時に処理する TCP コネクション数が多く、それらの TCP コネクションはラウンドトリップ時間、帯域等がそれぞれ大きく異なるため、各 TCP コネクションが必要とするソケットバッファサイズは大きく異なると考えられる。しかし、現在のほとんどの OS の実装においては、各 TCP コネクションに固定的にバッファを割り当てている。したがって、各 TCP コネクションの必要バッファ量を考慮した公平な送信ソケットバッファの割り当て方式の導入が重要であると考えられる。そのための一方式として、[5] において、各 TCP コネクションのウインドウサイズに基づいて要求バッファサイズを決定し、動的にバッファ割り当て量を変化させる Automatic TCP Buffer Tuning (ATBT) 方式が提案されている。しかし、TCP のウインドウサイズは変動が大きく、また必ずしも TCP のウィンドウサイズがスループットに比例するとは限らない [6] ため、ATBT 方式では公平かつ効率的なバッファ割り当てが行えない。

我々の研究グループでは、これらエンドホストの高速･高機能化の手法の一つとして SSBT (Scalable Socket Buffer Tuning) 方式を提案している。SSBT 方式は、TCP によるデータ転送効率と複数コネクション間の公平性の向上を目的とした動的ソケットバッファ割り当て手法である E-ATBT (Equation-based Automatic TCP Buffer Tuning) 方式 [1]、及び高速データ転送時の通信処理軽減手法である SMR (Simple Memory-copy Reduction) 方式 [2] からなる。E-ATBT 方式は、サーバにおいて定期的に各 TCP コネクションのスループットを p (パケットロス率)、RTT (ラウンドトリップ時間)、RTO (再送タイマ時間) の値から数学的解析手法により推測し、その値に基づき各コネクションに送信ソケットバッファを割り当てる。また、バッファが不足する場合にはコネクション間の Max-Min Fairness を考慮して割り当てバッファサイズを決定する。SMR 方式は、TCP によるデータ転送時に必要なエンドホストにおけるメモリコピーの回数を削減するデータ転送処理方式であり、サーバホストにおける通信処理オーバヘッドの軽減を実現している。

本稿では、SSBT 方式を Web サーバが稼働する実コンピュータ上に実装し、トランスポート層レベルでのデータ転送実験、及び実ネットワーク上での Web アクセス分布を考慮したワークロードを用いたアプリケーション層レベルでのデータ転送実験を行うことによって、Web サーバの高速・高機能化についての検討を行う。その結果、従来の方式に比べスループット、公平性の両面で優れていることを示す。

以下、2 章では SSBT 方式の説明を行い、3 章で、SSBT 方式を実コンピュータ上に実装し、トランスポート層レベル、及びアプリケーション層レベルでの稼働実験による性能評価を行う。最後に 4 章で本稿の結論を述べる。

2 SSBT (Scalable Socket Buffer Tuning) 方式

本章では、本稿で用いる SSBT 方式を構成する 2 つの方式である E-ATBT 方式と SMR 方式のアルゴリズムについて説明する。

2.1 E-ATBT (Equation-based Automatic TCP Buffer Tuning) 方式

文献 [5] では、サーバホストにおける動的な送信ソケットバッファの割り当て手法の一つとして Automatic TCP Buffer Tuning (ATBT) 方式が提案されている。これは、各 TCP コネクションの現在のウインドウサイズの変動に応じて、割り当てソケットバッファサイズを変化させる方式である。もし、すべての TCP コネクションに必要なバッファサイズの合計がサーバホストの持つ送信ソケットバッファサイズよりも大きい場合には、Max-Min Fairness に基づいた公平なバッファ割り当てを行う。すなわち、サーバホストはまずバッファを各コネクションに対して均等に割り当てようと試みた後、割り当てられたバッファが要求量よりも大きいコネクションが存在する場合には、その余剰バッファを他のより大きなバッファを要求するコネクションに均等に再割り当てを行う。これにより、コネクションごとのバッファの要求量の違いを考慮した公平なバッファ割り当ての実現を試みている。しかし、TCP のウインドウサイズは変動が大きいため、各コネクションに対して現在のウインドウサイズに基づいて送信ソケットバッファの割り当てを行うと、割り当てバッファサイズが時間によって大きく変動し、安定したバッファ割り当てを行うことができない。特にネットワークの帯域が大きくなると、ウインドウサイズの変化も大きくなるため、その影響はより大きくなる。したがってコネクション数が増加し、ネットワーク帯域が大きくなるにつれて、バッファ割り当ての不公平性、不安定性が大きくなることが考えられる。また、TCP のウインドウサイズは必ずしもスループットに比例するとは限らないため [6]、ATBT 方式では、各コネクションのスループットを考慮した公平かつ効率的なバッファ割り当てが行えない。

そこで、我々の研究グループでは、文献 [1] において、上記の問題を解決するために、サーバホストにおいて公平かつ効率的なバッファ割り当てを行うアルゴリズムである Equation-based Automatic TCP Buffer Tuning (E-ATBT) 方式を提案した。E-ATBT 方式では、3 つのパラメータから各コネクションのスループットを推測し、その値に基づいて要求バッファサイズを決定する。TCP コネクションのスループット推測には、[7] に示されている解析結果を利用する。 [7] では、リンク帯域の異なる複数の TCP コネクションがボトルネックのルータを共有し、そのルータが RED アルゴリズム [8] を実装している場合を想定し、各 TCP コネクションの平均スループットを解析的に導出している。解析では以下の 3 つのパラメータを用いてスループットの導出を行っている。

- p: RED アルゴリズムにおけるパケット廃棄率
- rtt: TCP コネクションの RTT (Round Trip Time) の平均値
- rto: TCP コネクションの RTO (Retransmission Time Out) の平均値

E-ATBT 方式ではこの解析結果を、RED アルゴリズムのパケット廃棄率 p をサーバホストで観測するパケットロス率とみなすことにより、各 TCP コネクションのスループット推測に適用する。その際、3 つのパラメータは以下のようにしてサーバホストで観測することができる。パケットロス率は、転送を完了したパケット数と、ACK パケットを基に推測するパケットロス数から算出する。また、rtt と rto についてはサーバホストの TCP が保持している変数を直接利用する。

各コネクション i の推定スループット $\overline{\rho_i}$ が求まると、各コネクションの要求バッファサイズ B_i を、

$$B_i = \overline{\rho_i} \times rtt_i \tag{1}$$

と決定する。ここで rtt_i はコネクション i の RTT である。

また、E-ATBT 方式では ATBT 方式と同様にバッファ割り当ての際に Max-Min Fairness を考慮するが、ATBT 方式では余剰バッファを各コネクションに均等に再割り当てしているのに対し、E-ATBT 方式では解析手法によって導出した各コネクションのスループットに比例して再割り当てを行う。これにより、余剰バッファの効率的な再割り当てが期待できる。

2.2 SMR (Simple Memory-copy Reduction) 方式

図 1 は従来の多くの OS における、TCP データ転送の際の処理方式を示したものである。図に示すように、従来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は、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に格納する。この際にもメモリコピーが行われるため、合計 2 回のメモリコピーが行われることになる。メモリコピーにかかる時間は、TCP データ転送時の他の処理に要する時間に比べて非常に大きいため [9]、2 回のメモリコピー処理はスループットの大きな低下につながる。

そこで、我々の研究グループでは、文献 [2] において、エンドホストでのメモリコピーの回数を削減することによって、データ転送処理速度を向上する Simple Memory-copy Reduction (SMR) 方式を提案した。SMR 方式においては、図 2 に示すように、新たに追加したソケットシステムコールによって、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、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るのに対して、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

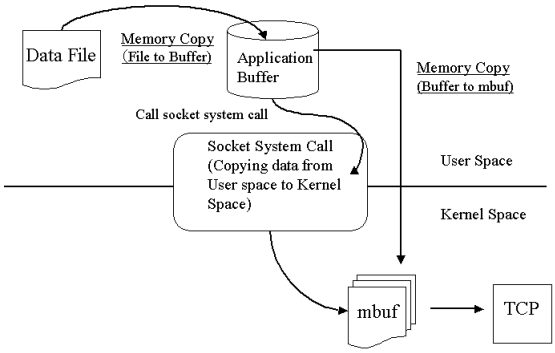


図 1: 従来方式における TCP データ転送の際の処理方式

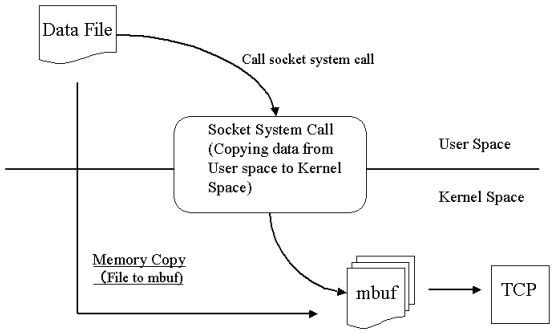


図 2: SMR 方式における TCP データ転送の際の処理方式

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、冗長なメモリコピーを省略できるため、サーバのデータ処理能力の向上が期待される。詳しい実装方法に関しては、文献 [2] を参照されたい。

2.3 Web ドキュメントの転送

E-ATBT 方式は、サーバホストが各 TCP コネクションから得られるパケットロス率等のパラメータを利用して、各 TCP コネクションのスループットを推測する。したがって、転送するドキュメントサイズが小さいと、パラメータの観測が十分に行えないため、正確なスループットの推測ができず、バッファ割り当てが不正確になることが考えられる。文献 [10] にあるように、Web サーバの平均ドキュメントサイズは 10 [KByte] 未満と非常に小さいため、E-ATBT 方式を Web サーバで利用すると、スループット推測が正確でなくなることが予測できる。また、文献 [11] では、Web ドキュメントのサイズは heavy-tail の特性を持っており、これは大きなサイズのドキュメントが無視できない確率で存在することを意味すると同時に、大部分が小さいサイズのドキュメントであることを示している。

この問題に対する解決法の一つとして、HTTP/1.1 で標準化された'persistent connection' を利用する方法が挙げられる。HTTP/1.1 では、サーバがドキュメント転送終了時に TCP コネクションの情報を保存し、同一ホストから

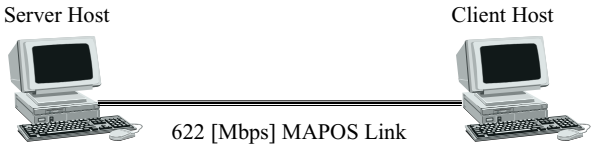


図 3: 実験ネットワーク (1)

の接続要求に対して、保存した情報を再利用する。したがって、各 TCP コネクションの生存時間が長くなるので、E-ATBT を用いることによって、効率的なバッファ割り当てが行えると考えられる。

3 実装実験による評価

本章では、SSBT 方式を実コンピュータへ実装し、稼働実験によって性能評価を行う。SSBT 方式の実装は、FreeBSD 4.0-RELEASE [12] を用いて行った。

3.1 トランスポート層での実験

3.1.1 SMR 方式の評価

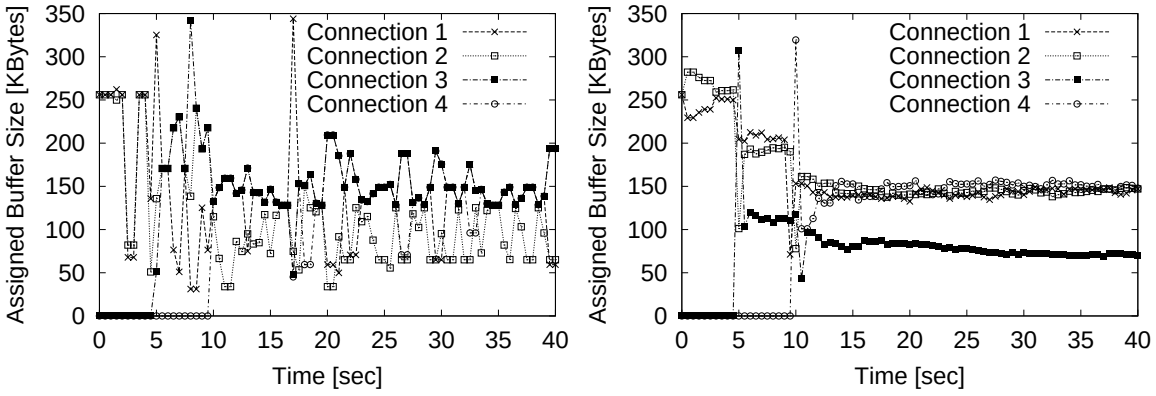
ここでは、SMR 方式の効果を評価するために、図 3 に示す実験ネットワークを用いて TCP によるデータ転送実験を行う。実験に使用したネットワークは、2 台のマシンを 622 [Mbps] のリンク帯域を持つ MAPOS (Multiple Access Protocol Over SONET/SDH) [13] で直接接続したものである。実験に使用したマシンの CPU は Pentium-III Xeon 550 [MHz]、メモリは 512 [MByte] である。

転送実験では、MAPOS リンク上に 1 本の TCP コネクションを設定し、500 [MByte] のデータを転送して転送時間を計測することによって転送速度を算出した。図 4 は、従来方式と SMR 方式を用いた場合のサーバホスト、クライアントホスト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、SMR 方式によるデータ転送速度が、従来方式と比較して最大約 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、メモリコピー回数を削減する SMR 方式の効果が大きく現れているためである。

3.1.2 E-ATBT 方式の評価

次に、E-ATBT 方式の性能評価を行うために、図 5 に示す実験ネットワークを用い、サーバホストに E-ATBT 方式を実装してデータ転送実験を行った。図中のサーバホスト及びクライアントホスト 1 には、Pentium-III 1 [GHz]、メモリ 512 [MByte] の PC を、クライアントホスト 2 には、Pentium-III Xeon 550 [MHz]、メモリ 512 [MByte] の PC を用いた。また、2 台のルータは、PC に ALTQ [14] を導入することで実現し、RED ルータとして動作させた。

まず、サーバに接続される TCP コネクション数が変化した時の各コネクションへのバッファ割り当ての時間的変化を図 6 に示す。ここで、4 本の TCP コネクションはそれぞれコネクション 1、2、4 がルータ 1 を通るコネク



(a) ATBT 方式　　(b) E-ATBT 方式

図 6: 割り当てバッファサイズの時間的変化

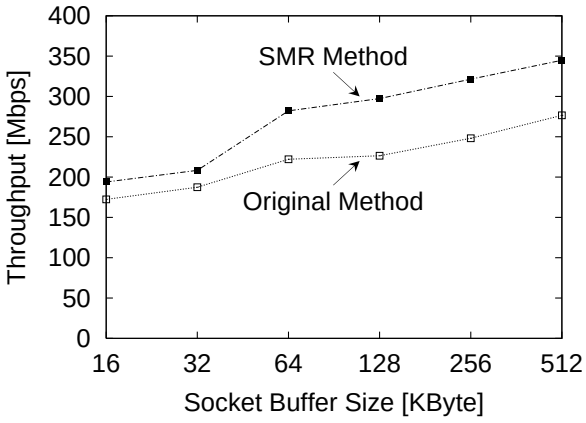


図 4: MAPOS におけるデータ転送実験結果

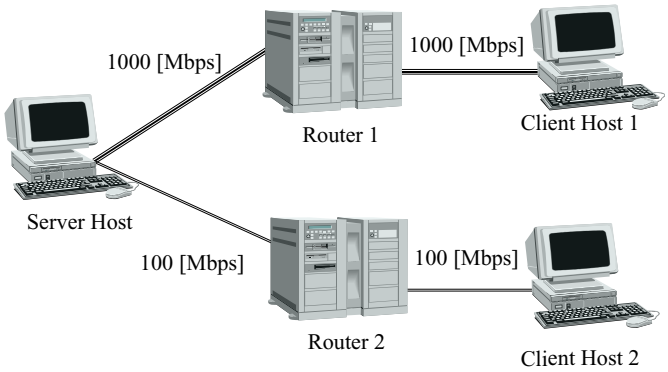


図 5: 実験ネットワーク (2)

ション、コネクション 3 がルータ 2 を通るコネクションであり、コネクション 1、2 は 0 [sec]、コネクション 3 は 5 [sec]、コネクション 4 は 10 [sec] でデータ転送を開始する。また、サーバホストのソケットバッファサイズは 512 [Kbyte] としている。図 6(a) から、ATBT 方式では各コネクションのバッファサイズが大きく振動していることがわかる。これは、ATBT 方式は各 TCP コネクションのウインドウサイズに基づいてバッファサイズの割り当てを行っているためである。また、大きな帯域を持つリンクを通過するコネクション 1、2、4 に大きなバッファが与えられていない。これは、 2 章で指摘したように、TCP のウインドウサイズは各 TCP コネクションのスループットに必ずしも結びつかないためである。一方、図 6(b) に示すように、E-ATBT 方式は安定したバッファ割り当てを行っている。これは、各 TCP コネクションの推測スループットを利用してバッファが割り当てられるため、各 TCP コネクションの要求バッファサイズがウインドウサイズの変化に依存しないためである。ここでは、コネクション 1、2、4 に必要なバッファサイズに比べてコネクション 3 に必要なバッファサイズが小さいので、コネクション 1、2、4 により多くのバッファが割り当てられている。

次に、サーバホストが同時に処理するコネクション数に対するスケーラビリティに着目した E-ATBT 方式の評価を行う。この実験においては、上述の実験と同様、図 5 に示すネットワークを用いる。ここで、ルータ 1 には転送遅延発生ツールである NIST Net [15] を導入し、サーバホストとクライアントホスト 1、2 間の往復伝搬遅延時間を 3 [ms] に設定し、ルータにおけるパケット廃棄率を 0.1%とした。また、サーバホストのソケットバッファサイズは 512 [KByte] とした。この実験ネットワークを用いて、1 本の TCP コネクションがサーバホストとクライアントホスト 1 の間のパス上でデータ転送を行い (以下、Gigabit コネクションと呼ぶ)、同時にサーバホストとクライアントホスト 2 の間のパス上で複数の TCP コネクション (以下、Ethernet コネクションと呼ぶ) が転送を行う。

図 7 は、Ethernet コネクションの本数を変化させた場合の、Gigabit コネクションのスループットの変化を示している。ここでは、各コネクションに対して等分にバッファサイズの割り当てを行う EQ 方式、ATBT 方式、及び E-ATBT 方式の評価結果を示している。図に示すように、EQ 方式では Ethernet コネクションが必要とするバッファサイズは Gigabit コネクションに比較して小さいにもか

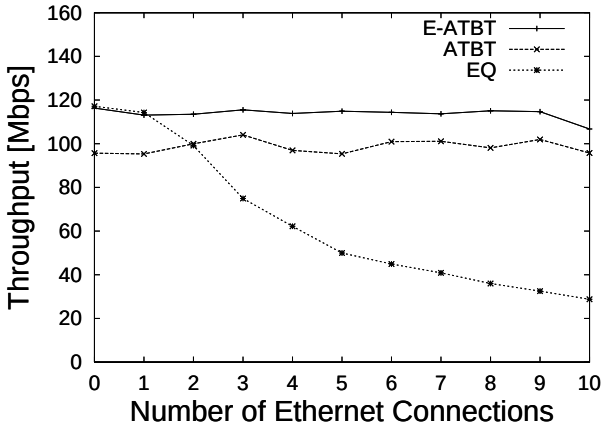


図 7: Ethernet コネクションの本数と Gigabit コネクションのスループットの関係

かわらず、すべてのコネクションに等しくバッファを割り当てるため、Ethernet コネクションの本数が増加するにつれて Gigabit コネクションに割り当てられるバッファサイズが減少し、スループットが大きく低下している。

それに対して ATBT 方式、E-ATBT 方式は、Ethernet コネクションの本数が増加しても、Gigabit コネクションのスループットは低下しない。これは、ATBT 方式はウインドウサイズに応じて、また E-ATBT 方式は各コネクションの推測スループットに応じてそれぞれバッファサイズを決定するため、Gigabit コネクションにより多くのバッファが割り当てられるためである。しかし、E-ATBT 方式における Gigabit コネクションのスループットは、ATBT 方式よりも高くなっている。これは、図 6(a) で示したように、ATBT 方式では割り当てバッファサイズの変動が大きいので、割り当てバッファサイズが必要なサイズよりも小さい場合や、必要なバッファサイズが小さいコネクションに多くのバッファが割り当てられる場合があるが、E-ATBT 方式は、図 6(b) に示すように安定したバッファ割り当てを実現しているためである。また、E-ATBT 方式は余剰バッファの再割り当てを要求バッファサイズに比例して公平に行うため、Gigabit コネクションに割り当てることのできるバッファサイズがより大きくなり、Gigabit コネクションのスループットの劣化を防いでいる。

3.2 アプリケーション層での実験

次に、SMR 方式を用いて TCP によるデータ送信を行うサーバとして Web サーバを想定し、クライアントホストから Web ドキュメント転送要求を行い、ドキュメント転送時間を計測することにより、SMR 方式の性能評価を行う。

3.2.1 Web アクセス分布モデル

クライアントが発生させる Web ドキュメント転送要求のモデルとして、文献 [16] で示されている Web アクセス分布モデルを使用する。[16] は、Web アクセス分布として、転送ドキュメントサイズ (Request Sizes)、サーバに蓄積されているファイルサイズ (File Sizes)、ドキュメント内で参照されているファイルの数 (Embedded References)、ドキュメント転送間隔時間 (Inactive OFF times)、ドキュメント内で参照されているファイルの転送間隔時間 (Active OFF times) 等に分別し、それぞれ実際の Web サーバのログデータを基に統計的手法を用いてモデル化を行っている。 [16] で示されている Web アクセス分布モデルを表 1 に示す。本稿の実装実験では、HTTP 転送要求発生ツール httperf [17] を、表 1 の Web アクセス分布モデルを考慮してドキュメント転送要求を行うように修正し、それをクライアントホストにおいて用いることで、Web サーバへのドキュメント転送要求を行う。

表 1: Web アクセス分布モデル [16]

	モデル
File Sizes - Body	Lognormal 分布
File Sizes - Tail	Parato 分布
Request Sizes	Parato 分布
Active OFF Times	Weibull 分布
Inactive OFF Times	Parato 分布
Embedded References	Parato 分布

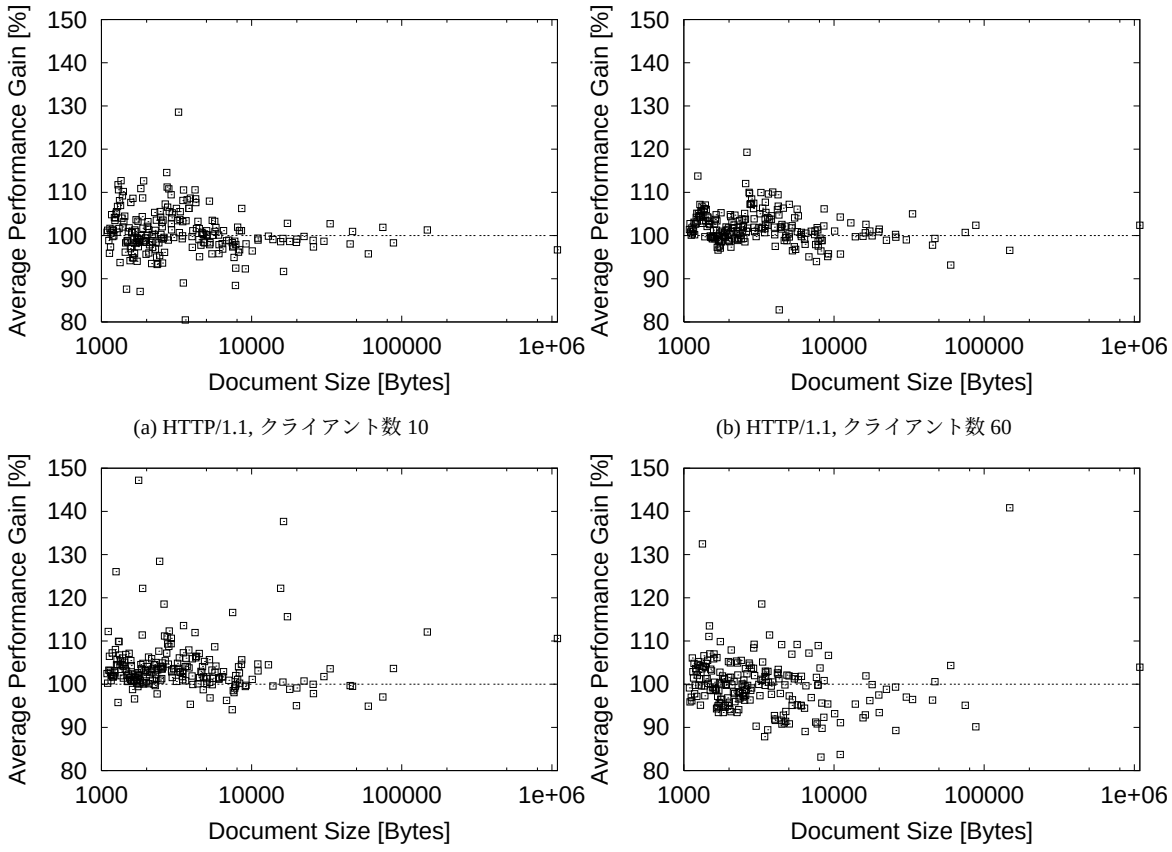
3.2.2 評価結果

本節では、Web サーバ Apache [18] (バージョン 1.3.14) が稼働する実コンピュータ上に SMR 方式を実装し、ドキュメント転送を行った場合の結果を示す。実験に使用したネットワーク環境は、図 3 に示した、2 台のマシンを 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。サーバホストでは SMR 方式を使用できるようにシステムコールを変更した Web サーバを稼働させ、クライアントホストから httperf を用いて 3.2.1 節で述べたモデルを用いて、Web サーバに対して転送要求を行う。ここでは、Web ドキュメント転送のためのプロトコルとして HTTP/1.0 と HTTP/1.1 の 2 つのプロトコルを用いた。サーバホスト、及びクライアントホストのソケットバッファサイズはそれぞれ 64 [KByte] とし、30 分間の転送実験を行った。これは 1 クライアントがおよそ 25,000 回のドキュメント転送要求を行う時間に相当する。

図 8 は、SMR 方式の転送ドキュメントサイズ i に対する、転送時間の平均改善率 G_i を示したものである。平均改善率は以下のように定義する。

$$G_i = \left(\frac{\sum_{j=1}^{M_{smr,i}} \frac{i}{T_{smr,i,j}}}{M_{smr,i}} \right) \Bigg/ \left(\frac{\sum_{j=1}^{M_{orig,i}} \frac{i}{T_{orig,i,j}}}{M_{orig,i}} \right) \quad (2)$$

ただし、$M_{smr,i}$、$M_{orig,i}$ をそれぞれ SMR 方式を行った場合と行わなかった場合のドキュメント要求回数とし、同様に j 個目のドキュメントに要した転送時間をそれぞれ $T_{smr,i,j}$、$T_{orig,i,j}$ とする。



(a) HTTP/1.1, クライアント数 10

(b) HTTP/1.1, クライアント数 60

(c) HTTP/1.1, クライアント数 600

(d) HTTP/1.0, クライアント数 600

図 8: ドキュメントサイズに対する SMR 方式の平均改善率

図 8(a)、8(b)、8(c) はそれぞれクライアント数が 10、60、600 で HTTP/1.1 を用いて転送した場合、図 8(d) はクライアント数が 600 で HTTP/1.0 を用いて転送した場合の、ドキュメントサイズと平均改善率の関係を示している。図 4 で示したように、SMR 方式は TCP によるデータ転送のスループットを約 30%改善することができるにもかかわらず、図 8(a) のように同時に接続しているコネクション数が小さい場合には、ドキュメント転送時間はほとんど向上していないことがわかる。これは、転送するドキュメントの大半のサイズが小さいため、SMR 方式の効果がほとんど現れないためである。一方、 図 8(a)–図 8(c) から、クライアント数が増加するにつれて、平均改善率は安定して向上していることがわかる。これは、同時に接続するクライアント数が増えるにつれて、サーバの負荷が高くなり、メモリコピー回数を削減することによってエンドホストにおける通信処理を軽減する SMR 方式の効果が大きく現れるためである。

また、図 8(c) と図 8(d) から、SMR 方式による性能向上は HTTP/1.0 に比べて HTTP/1.1 の方が大きいことがわかる。HTTP/1.0 の場合は、ほとんど SMR 方式の効果が現れていない。これは、HTTP/1.0 は persistent connection をサポートしておらず、1 つのドキュメントを転送を行う度に、TCP の 3-way ハンドシェイクによるコネクション確立が行われるため、コネクションを確立するために要する時間が、ドキュメントを転送する時間のほとんどを占めるためである。一方、HTTP/1.1 は persistent connection をサポートしており、ドキュメント転送後一定時間 TCP コネクションの情報を保持し、時間内に同じホストに転送するドキュメントに対しては、そのコネクションを再度利用して転送を行う。したがって、TCP の 3-way ハンドシェイクによる影響を受けることがないため、SMR 方式の効果が大きく現れる。

E-ATBT 方式の Web サーバ上における効果に関しては、現在同じトラヒックモデルを用いて性能評価実験を行っている。本節で説明した通り、HTTP/1.1 を用いることで、転送ドキュメントサイズが小さい場合にも、SMR 方式の効果が現れることがわかった。2.3 節で述べたように、E-ATBT 方式にはドキュメントサイズが小さい場合、ネットワークの観測が十分に行えないという問題点があるが、HTTP/1.1 を用いることで、TCP コネクションの生存時間が長くなり、ネットワークの状態を高い精度で観測できるため、E-ATBT 方式の効果は大きく現れると期待できる。

4 おわりに

本稿では、エンドホストの高速・高機能化の手法の一つとして提案された SSBT 方式を実コンピュータ上に実装し、トランスポート層レベルでのデータ転送実験、及びイン

ターネットにおける Web アクセス分布を考慮したワークロードを用いた HTTP によるデータ転送実験によって、Web サーバの高速・高機能化についての検討を行った。実験を通じて、SMR 方式によって、TCP によるデータ転送時に必要なエンドホストにおけるメモリコピーの回数を削減し、サーバホストにおける通信処理オーバヘッドが軽減する結果、転送速度が最大約 30%向上することを示した。また、E-ATBT 方式は、解析手法によって推定した各 TCP コネクションのスループットに基づく効率的なバッファ割り当てを行うため、スループット及び公平性の両面で従来方式に比べ優れていることを明らかにした。また、TCP によるデータ送信を行うサーバとして想定した Web サーバを用いた実験においても、SMR 方式の効果が確認された。

今後の課題としては、Web サーバが稼働する実コンピュータに E-ATBT 方式を実装し、稼働実験によって性能評価を行うことが挙げられる。その際には、スイッチ、ルータ等を用いて、コネクション間の遅延時間、パケットロス率等が異なるような、より一般的な実験ネットワークを用いた稼働実験を行いたい。さらに、HTTP/1.1 の persistent connection を用いたコネクションキャッシングを考慮した、効率的なバッファ割り当てに関しても、今後検討していく予定である。

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] T. Matsuo, G. Hasegawa, M. Murata, and H. Miyahara, “Scalable Automatic Buffer Tuning to Provide High Performance and Fair Service for TCP Connections,” in *Proceedings of IEEE INET2000*, July 2000.

[2] 寺井 達彦 , 松尾 孝広 , 長谷川 剛 , 村田 正幸, “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,” **電子情報通信学会 技術研究報告** (IN2000-23), pp. 45–50, May 2000.

[3] 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.

[4] J. Chase, A. Gallatin, and K. Yocum, “End-System Optimizations for High-Speed TCP,” *to appear in IEEE Communications, special issue on high-speed TCP*, June 2000.

[5] J. Semke, J. Mahdavi, and M. Mathis, “Automatic TCP buffer tuning,” in *Proceedings of ACM SIGCOMM’98*, pp. 315–323, August 1998.

[6] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” in *Proceedings of ACM SIGCOMM’98*, pp. 303–314, August 1998.

[7] T. Matsuo, G. Hasegawa, M. Murata, and H. Miyahara, “Comparisons of packet scheduling algorithms for fair service among connections,” in *Proceedings of Internet Workshop ’99*, pp. 193–200, February 1999.

[8] S. Floyd and V. Jacobson, “Random early detection gateways for congestion avoidance,” *IEEE/ACM Transactions on Networking*, vol. 1, pp. 397–413, August 1993.

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

[10] M. Nabe, M. Murata, and H. Miyahara, “Analysis and modeling of world wide web traffic for capacity dimensioning of internet access lines,” *Performance Evaluation*, vol. 34, pp. 249–271, December 1999.

[11] M. E. Crovella and A. Bestavros, “Self-similarity in World Wide Web traffic: Evidence and possible causes,” *IEEE/ACM Transactions on Networking*, vol. 6, pp. 835–846, Dec. 1997.

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

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

[14] ALTQ (Alternate Queueing for BSD UNIX) , available at *http://www.csl.sony.co.jp/~kjc/software.html*.

[15] NIST Net Home Page, available at *http://www.antd.nist.gov/itg/nistnet/*.

[16] P. Barford and M. Crovella, “Generating Representative Web Workloads for Network and Server Performance Evaluation,” in *Proceedings of ACM SIGMETRICS ’98*, 1998.

[17] D. Mosberger and T. Jin, “httperf – A Tool for Measuring Web Server Performance,” *Technical Report, Hewlett-Packard Laboratories, HPL-98-61*, March 1998.

[18] Apache Home Page, available at *http://www.apache.org/*.