社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

インターネットサーバにおける TCP コネクション資源管理手法の実装評価

東　和弘[†]　岡本　卓也[†]　長谷川　剛[††]　村田　正幸[††]

† 大阪大学 大学院情報科学研究科　〒 560–8531 大阪府豊中市待兼山町 1–3
†† 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 1–32
E-mail: †{k-azuma,tak-okmt}@ist.osaka-u.ac.jp, ††{hasegawa,murata}@cmc.osaka-u.ac.jp

あらまし　本稿では、Web/Web プロキシサーバなどのインターネットサーバにおける、TCP を用いたデータ転送のために必要なサーバ資源の管理手法の実装実験結果について報告する。提案方式は、各コネクションが必要とする送受信ソケットバッファを必要量に応じて動的に割り当てるソケットバッファの動的割り当て方式、およびアイドル状態のコネクションを積極的に切断するコネクション管理方式からなる。ソケットバッファの動的割り当て方式では、TCP の送受信ソケットバッファを観測された情報を用いて推測した必要量に応じて割り当てる。また、コネクション管理方式は、Web プロキシサーバの負荷が大きく残存資源が少ないときに、データ転送を行っていない persistent TCP コネクションを切断し、その TCP コネクションが使用していた Web プロキシサーバ資源を解放する手法である。本稿では、Web プロキシサーバへの実装実験の結果から、提案方式によって Web プロキシサーバのスループットが最大 25%、クライアントが感じるドキュメント転送遅延時間を最大 90%改善できることを示す。

キーワード　Web プロキシサーバ、TCP、資源管理、実装

Implementation and Evaluation of Resource Management System for Internet Servers

Kazuhiro AZUMA[†], Takuya OKAMOTO[†], Go HASEGAWA[††], and Masayuki MURATA[††]

† Graduate School of Information Science and Technology, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka 560–8531, Japan
†† Cybermedia Center, Osaka University
1–32 Machikaneyama, Toyonaka, Osaka 560–0043, Japan
E-mail: †{k-azuma,tak-okmt}@ist.osaka-u.ac.jp, ††{hasegawa,murata}@cmc.osaka-u.ac.jp

Abstract　We propose a new resource management scheme for Web/Web proxy servers, which manages their resources for TCP connections, effectively and fairly. In the proposed system, we focus on the effective use of server resources by assigning dynamically the send/receive socket buffer according to the required size of each TCP connection, and terminating positively the idol persistent connection. In this paper, we validate the effectiveness of our proposed scheme through implementation experiments, and confirm conclusively that Web/Web proxy server throughput can be improved by 25% at maximum, and document transfer delay perceived by client hosts can be decreased by 90%.

Key words　Web proxy server, TCP, Resource management, Implementation

1. はじめに

インターネットの普及によるネットワークトラヒックの増大に対して、ネットワーク帯域の増強、輻輳の回避、解消に関する多くの研究が行われている。一方、ネットワークが高速になるにつれ、Web サーバ等のエンドホストがボトルネックとなる状況が生まれつつあるにも関わらず、エンドホストの高速・高機能化に関する研究は比較的少ない。これは、これまでネットワークの処理速度と比較してエンドホストにおけるデータ転送の処理速度は十分高速であったため、エンドホストがボトルネックとなる状況は想定されなかったためである。そこで我々の研究グループでは、Web サーバ等のインターネットサーバにおけるデータ転送処理の高速・高機能化のための手法として SSBT (Scalable Socket Buffer Tuning) 方式を提案した [1]。本方式は、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、動的な送信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、およ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) 方式からなる。[1] では、シミュレーションおよび実験ネットワークにおいて、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データ転送実験を行ない、SSBT 方式の有効性を確かめた。

しかし、現在のインターネットでは、プロキシサーバを経由した HTTP アクセスが増加している [2]。図 1 に示すように、Web プロキシサーバでは、プロキシサーバから Web サーバへ張られる TCP コネクションと、クライアントホストからプロキシサーバへ張られる TCP コネクションの両方を処理する。したがって、ネットワークが輻輳しておらず、かつ Web サーバの稼働能力に十分な余裕がある場合でも、Web ドキュメント転送の際に経由する Web プロキシサーバの処理能力が不足することによって、Web プロキシサーバがスループット向上のボトルネックとなることが考えられる。さらに、CDNs (Content Delivery Networks) [3, 4] や P2P ネットワーク [5] のようなサービスオーバレイネットワークにおいても，繁忙なサーバ (ピア) は環境の異なる他の様々なホストに対して多数の TCP コネクションを確立し、Web プロキシサーバと同様に、TCP の送信側と受信側の処理を同時に行っている。そのため、TCP コネクションに関係する資源の効率的な管理はこれらのネットワークの性能を向上させるためには重要であると考えられる.

そこで我々の研究グループでは [6] において、 Web プロキシサーバの特性を考慮した動的資源管理方式として、送受信ソケットバッファの動的割り当て方式、およびドキュメント転送終了後の persistent TCP コネクションを管理するコネクション管理方式を提案した。送受信ソケットバッファの動的割り当て方式は、[1] で提案した E-ATBT 方式を Web プロキシサーバへ適用し、さらに TCP 受信ソケットバッファの動的割り当て、およびクライアントに対する TCP コネクションと Web サーバに対する TCP コネクションの依存関係を考慮した、統合的なソケットバッファ管理手法である。また、コネクション管理方式は、Web プロキシサーバの負荷が大きく残存資源が少ないときに、データ転送を行っていないアイドル状態の persistent TCP コネクションを切断し、その TCP コネクションが使用していた Web プロキシサーバ資源を解放する手法である。[6] では、コンピュータ上のシミュレーションおよび簡単な実験ネットワークを用いた実装実験によって、提案方式の基本的な性質を評価した。そこで本稿では、提案方式を Web プロキシサーバ上へ実装し、大規模な実験ネットワーク上において行ったデータ転送実験の結果を示す。その結果、提案方式がより現実的なネットワーク負荷のもとで有効であることを明らかにする。

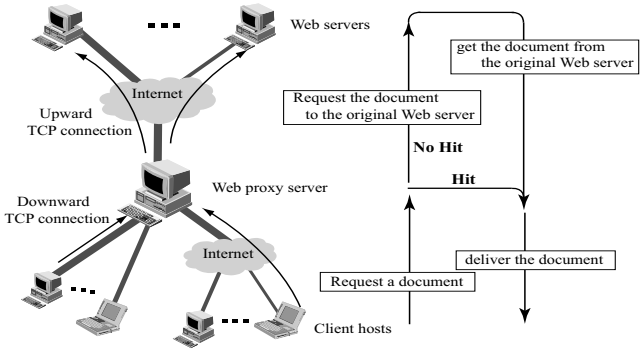


図 1　Web プロキシサーバ

以降、2 章では [6] において提案した資源管理方式の概略を示す。3 章では提案方式の実装指針について述べ、4 章で実装実験結果と考察を示す。最後に 5 章で本稿のまとめと今後の課題を示す。

2. インターネットサーバの動的資源管理方式

本章では [6] で提案した、Web/Web プロキシサーバに対する資源管理方式について簡単に説明する。

2.1 ソケットバッファの動的割り当て方式

現在のインターネットにおける繁忙な Web/Web プロキシサーバでは、数百から数千本の TCP コネクションを同時に扱わなければならない。そのため、Web/Web プロキシサーバの高速・高機能化のためには、Web/Web プロキシサーバのふるまいを考慮したソケットバッファの効率的な割り当てを行なう必要がある。そこで本節では、[6] で提案した送受信ソ

ケットバッファの動的に割り当て手法の概略を述べる。

2.1.1 送信ソケットバッファの動的割り当て方式

[1] において我々の研究グループは、Web サーバにおいてラウンドトリップ時間 (RTT)、再送タイムアウト時間 (RTO)、およびパケットロス率から各 TCP コネクションのスループットを解析的に推測し、そのスループットに応じて動的に送信バッファを割り当てる、E-ATBT 方式を提案した。本方式では、送信側が上述のパラメータを利用し、[7] に示されている解析手法により平均ウィンドウサイズを導出する。さらに RTO を考慮し、各 TCP コネクションの平均スループットを導出する。その後、平均スループットを基に各 TCP コネクションが必要としている送信バッファサイズを導出し、各コネクションに割り当てる。

2.1.2 受信ソケットバッファの動的割り当て方式

TCP の受信側においては、RTT、パケット廃棄率などを正確に観測することは困難であるため、2.1.1 節に示した送信ソケットバッファの動的割り当て方式と同様の方式を用いることはできない。そこで提案方式においては、TCP によるデータ転送の際の、受信バッファの利用量の変動の特性を利用して送信側の輻輳ウィンドウサイズを推測することによって、過不足なく受信バッファを割り当てる。

送受信ホスト間のネットワークにおいてパケット廃棄が発生しない場合には、送信側が送出したデータパケットが受信側に到着すると、受信バッファに格納された直後に上位層アプリケーションに渡される。したがって、データパケットは受信バッファにほとんど蓄積されないため、受信バッファの利用量は小さい。一方、送受信ホスト間のネットワークにおいてパケット廃棄が発生すると、それ以降に受信側に到着したデータパケットは、廃棄されたパケットが再送されて受信側に到着するまで、受信バッファに蓄積される。そのため、受信バッファの利用量が一時的に増加する。その際受信バッファに蓄積されるデータ量は、送信側の輻輳ウィンドウサイズにほぼ等しくなるため [6]、受信バッファの利用量を監視することで、送信側の輻輳ウィンドウサイズを推測することができる。

提案方式ではこの性質を利用して、受信バッファの利用率が 100% に近い場合には割り当てサイズを大きくし、利用率が低い場合には割り当てサイズを小さくする。これによって、送信側の輻輳ウィンドウサイズの大きさに合わせた受信バッファの割り当てを実現している。

2.1.3 TCP コネクション間の依存関係

Web プロキシサーバは、クライアントからのドキュメント転送要求を代理して Web サーバに対して行う。そのため、クライアントに対する TCP コネクションと Web サーバに対する TCP コネクションの間には依存関係が存在し、バッファ割り当ての際には、両コネクションのスループットの違いを考慮する必要がある。たとえば、クライアントに対する TCP コネクションのスループットが対応する Web サーバに対する TCP コネクションのスループットより大きい場合、E-ATBT 方式によってクライアントに対する TCP コネクションには大きなソケットバッファが割り当てられるが、Web サーバに対する TCP コネクションのスループットが小さいため、クライアントに対するコネクションに割り当てられたソケットバッファが使い切れず無駄が生じる。この場合、使い切れないソケットバッファをソケットバッファが不足している別の TCP コネクションに割り当てることによって、それらの TCP コネクションのスループットを向上させることができる。Web プロキシサーバの特性上存在するこのような TCP コネクション間の依存関係を考慮したソケットバッファの割り当ては、コネクション間の公平性、および効率的な資源利用の観点から重要である。しかし、OS のカーネルで TCP コネクションを識別することは、たとえば FreeBSD においては inpcb や tcpcb などのコントロールブロックを用いることで可能であるが、上述のような TCP コネクション間の依存関係を識別することはできない。

そこで、提案方式においては、Web プロキシサーバにおいてクライアントに対する TCP コネクションの送信側ソケットバッファの利用率を監視し、利用率が低い場合にバッファの割り当て量を減少させる方法を用いる。

2.2 コネクション管理方式

本節では、サーバ資源を効率的に利用するために Web プロキシサーバ上に存在する persistent TCP コネクションを管理するコネクション管理方式の概略を述べる。

提案方式では、Web プロキシサーバ上の TCP コネクション数が少なく残存資源が十分にあるときは、できるだけ多くの persistent TCP コネクションを切断せずに維持する。これは、persistent TCP コネクションを利用することで、新たなドキュメント転送の際に 3 ウェイハンドシェイクを避けることができ、データ転送をすばやく開始することができるためである。一方、Web プロキシサーバの資源が少ないときには、データ転送を行っていない persistent TCP コネクションをタイマ (persistent timer) が切れる前に切断し、その TCP コネクションが使用していた Web プロキシサーバ資源を解放し、他の TCP コネクションにその資源を与える。これにより、アイドル状態の persistent TCP コネクションが浪費していた Web プロキシサーバ資源を、新たに発生したドキュメント転送要求に対応するために資源を必要としている新規の TCP コネクションに割り当てることができるため、繁忙な Web プロ

キシサーバの資源を効率良く利用することができる。

3. 実　　装

本章では、特に 2.1.2 節で述べた受信バッファの動的割り当て方式の実装方法について述べる。他の提案方式の実装方法に関しては [6,8] を参照されたい。

2.1.2 節で述べたように、受信ソケットバッファの動的割り当て方式においては、各 TCP コネクションの受信バッファの利用率、およびパケット廃棄の発生を監視する必要がある。受信バッファの利用率を求めるためには、カーネル内でソケットバッファを管理するために用いられている構造体 sockbuf 中の変数 sb_hiwat と sb_cc を利用する。これらの変数はそれぞれ、割り当てられたソケットバッファサイズ、および 現在のソケットバッファの利用量を表しているため、この 2 つの変数の値を比較することで受信バッファの利用率を求めることができる。提案方式においては、受信バッファの利用率を 1 秒間隔で監視している。また、受信側でパケット廃棄を厳密に検出するためには、TCP コネクションの RTT を知る必要がある。しかし、受信側で RTT を観測することは困難であるため、提案方式においては、Fast Retransmit アルゴリズムと同様、受信側において 3 つ以上の重複 ACK パケットを送信した場合に、パケット廃棄が発生したと判断する。

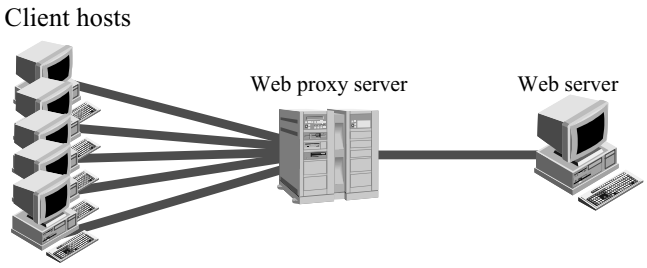
受信バッファサイズを動的に変化させる際には、以下の 2 つの問題が発生することが考えられる。1 つは、ACK パケットの広告ウィンドウサイズを用いて受信ソケットバッファサイズの変化を送信側へ通知する際に、送信側が ACK パケットを受信する前に受信ソケットバッファサイズを変化させると、受信バッファにおいてバッファ溢れが発生する可能性がある問題である。提案方式においては、受信バッファサイズの変化を通知する ACK パケットを送信してから十分大きな時間 (本稿における実験では 0.5 秒) が経過してから受信バッファサイズを変更することで、バッファ溢れを防止している。もう 1 つは、受信バッファサイズを減少させたときに発生する可能性がある Window Shrink 問題 [9,10] である。Window Shrink 問題とは、受信側が大きな広告ウィンドウサイズを持つ ACK パケットを送信側へ送信した後に、大きなデータを受け取ることなく広告ウィンドウサイズを急激に小さくした場合に、送信側においてまだ ACK パケットを受信していないデータパケット数よりも受信した ACK パケットの広告ウィンドウサイズが小さくなり、次に送信するデータパケットを特定することができない問題である。広告ウィンドウは受信側の受信バッファの空き容量を ACK パケットを通して送信側に通知するために利用されているため、提案方式のように動的に受信バッファを割り当てることにより、Window Shrink 問題が発生することが考えられる。しかし提案方式では、受信バッファの利用率を監視し、パケット廃棄発生後の受信バッファの利用率が閾値未満であれば、受信バッファサイズが送信側の輻輳ウィンドウサイズに比べて十分大きいと判断し、受信バッファの割り当てサイズを減少させるため、Window Shrink 問題は発生しないと考えられる。

4. 実験結果および考察

図 2 に、本稿で用いた実験ネットワークの概略を示す。Web プロキシサーバには Dual Intel Xeon Processor 2 GHz、メモリ 2 GBytes の PC 、および Web サーバには Intel Xeon Processor 2 GHz、メモリ 2 GBytes の PC を用いる。5 台のクライアントホストは Pentium-III 1.13 GHz、メモリ 512 MBytes の PC である。Web/Web プロキシサーバマシン上では、FreeBSD-4.6-RELEASE が、また 5 台のクライアントマシン上では FreeBSD-4.8-RELEASE が動作している。また、NIST Net [11] を用いて Web プロキシサーバと Web サーバ間の RTT を 300 ms 、Web プロキシサーバとクライアントホスト間の RTT をそれぞれ、500 ms、150 ms、50 ms、10 ms、1 ms と設定している。Web プロキシサーバでは、同時に処理できる TCP コネクション数を 1000 本に設定し、提案方式において persistent TCP コネクションを切断し始める閾値としてのコネクション数を 800 本とした。Web プロキシサーバでの persistent コネクションの持続時間は 15 秒とし、キャッシュサイズはキャッシュヒット率が約 0.5 となるように設定した。

クライアントホストは httperf [12] を用いて、Web プロキシサーバに Web ドキュメント転送要求を送る。httperf を用いることで、クライアントホスト上に Web ドキュメント転送要求を行う複数のユーザをエミュレートすることができる。本稿における実験では、各クライアントホストにおいて 20、60、100、140、180、200、300、400 人のユーザをエミュレートし、合計で 100、300、500、700、900、1000、1500、2000 人のユーザをエミュレートする。ユーザ数が多くなり、Web プロキシサーバで処理するコネクション数が 1000 本以上になった場合、そのすべてのコネクションを確立することはできなくなる。クライアントホストでの各ユーザのアクセスモデルは [13] に記されているものを用いる。以上の実験環境において、Web プロキシサーバに提案方式を実装した場合、および従来方式を用いた場合の実験を行い、その結果を以下に示す。

図 3 は、ユーザ数を変化させた時の、Web プロキシサーバの平均スループットと平均ドキュメント転送時間を示している。ここで、Web プロキシサーバの平均スループットをプロ



# of users in each client host
20, 60, 100, 140,
180, 200, 300, 400

upper limit : 1000 connections
threshold : 800 connections
persistent timer : 15 sec
cache hit rate : 0.5

図 2　実験ネットワーク

キシサーバからクライアントホストへ転送されたドキュメントと Web サーバからプロキシサーバに転送されたドキュメントの総量を実験時間 (500 秒) で割った値、およびドキュメント転送時間をクライアント上のユーザがドキュメント転送要求をプロキシサーバに送ってから要求したドキュメントを受信するまでの時間と定義する。これらの図には、5 回の実験における平均値とともに、実験結果の 95% の信頼区間をあわせて示している.

図 3(a) から、送受信ソケットバッファにそれぞれ (16KB、16KB) を割り当てた従来方式では、ユーザ数に関わらずプロキシサーバのスループットが低いことがわかる。これは、割り当てられた送受信バッファサイズが小さく、ドキュメント転送速度が低く抑えられているためである。さらに、ユーザ数が 700 以上の場合、ユーザ数の増加にともない Web プロキシサーバのスループットが低下している。これは、Web プロキシサーバの資源の多くがアイドル状態の persistent コネクションによって浪費されているため、クライアントホストからのドキュメント転送要求が棄却されるためである。また、図 3(b) よりユーザ数の増加とともにドキュメント転送時間が大きく増加していることがわかる。これは、Web プロキシサーバでのコネクション数が 1000 本を超えると、クライアントからのドキュメント転送要求が Web プロキシサーバによって棄却され始めるためである。

また、送受信ソケットバッファにそれぞれ (256KB、256KB) を割り当てた従来方式の場合は、(16KB、16KB) を割り当てた従来方式の場合と比較して Web プロキシサーバのスループットが大きく増加していることがわかる。これは、各 TCP コネクションが大きなサイズの送受信バッファを用いることで十分なスループットを得られていることを示している。しかし、クライアントが感じるドキュメント転送時間は改善されず、図 3(b) より、ユーザ数が大きい場合には (16KB、16KB) を割り当てた従来方式の場合よりも転送時間が大きくなっていることがわかる。これは、Web プロキシサーバでのコネクション数の制限に加えて、送受信バッファに大きなサイズを割り当てていることによるバッファ資源の不足によって、新たに到着するドキュメント転送にともなう TCP コネクションを確立することができないためである。

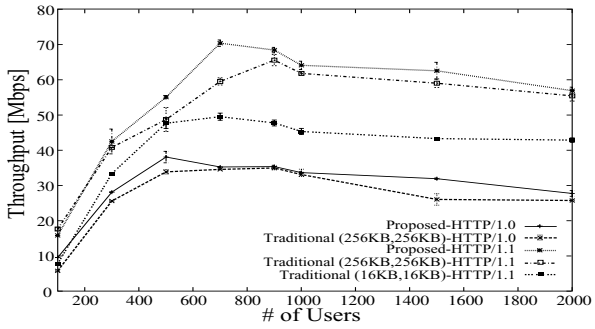
一方、提案方式はユーザ数に関わらず最も高いプロキシサーバのスループットを実現している。これは、提案方式を用いることによって、各 TCP コネクションが必要とするサイズにしたがって、送受信ソケットバッファが割り当てられていることを意味する。さらに、ユーザ数が増加してもドキュメント転送時間はほとんど増加していない。これは、コネクション管理方式によって、資源を浪費しているアイドル状態の TCP コネクションを切断し、新しく到着する TCP コネクションを受理するためである.

しかし、HTTP/1.0 を用いることによって、ドキュメント転送終了後に TCP コネクションは直ちに切断されるため、上述のような persistent コネクションに起因する問題は発生しないと考えられる。図 3(b) より、HTTP/1.0 を用いた場合には送受信ソケットバッファにそれぞれ (256KB、256KB) を割り当てた従来方式においても、ドキュメント転送時間が小さいことがわかる。しかし、図 3(a) から Web プロキシサーバのスループットが HTTP/1.1 の場合に比べて低いことがわかる。これは、HTTP/1.0 を用いた場合、1 つのドキュメント転送が終了すると直ちにコネクションが切断されるため、連続したドキュメント転送の際に 3 ウェイハンドシェイクによるオーバヘッドが大きくなるためと考えられる。すなわち、persistent コネクションの長所を保ちつつ、短所をできるだけ解消する提案方式によってのみ、プロキシサーバのスループットおよびドキュメント転送時間の双方を改善することができる。

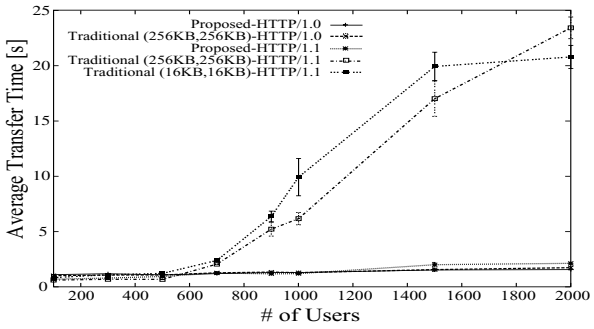
図 4 は、Web プロキシサーバにおける全ての TCP コネクションに割り当てられたソケットバッファの平均サイズを示している。この図から、提案方式がプロキシサーバの資源利用量の観点からも高い性能を示していることがわかる。送受信ソケットバッファにそれぞれ (256KB、256KB) を割り当てた従来方式の場合は、提案方式と比べて約 5 倍のソケットバッファを使用しているにも関わらず、スループットは提案方式よりも低い。また、送受信ソケットバッファにそれぞれ (16KB、16KB) を割り当てた従来方式と比較しても、提案方式はそれほど多くのソケットバッファを使っていないことがわかる.

5. おわりに

本稿では、インターネットサーバにおける TCP コネクショ



(a) Web プロキシサーバの平均スループット



(b) 平均ドキュメント転送時間

図 3　実験結果 (1)

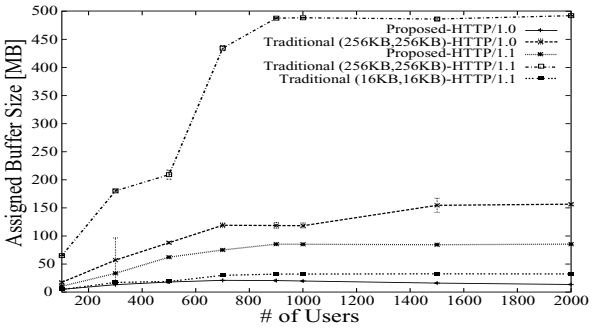


図 4　実験結果 (2)

ン資源管理手法の実装実験結果を示し、その有効性を確認した。提案方式は 2 つのアルゴリズムからなる。1 つは様々な TCP コネクションの性質に基づいて効率良く送受信ソケットバッファを割り当てるソケットバッファの動的割り当て方式であり、Web プロキシサーバでのクライアントに対する TCP コネクションと Web サーバに対する TCP コネクションの間の依存関係を考慮している。もう 1 つは、persistent TCP コネクションを管理するための方式であり、サーバの残存資源量を監視し、サーバ資源が不足したときにアイドル状態の TCP コネクションを積極的に切断する。本稿において行った実験結果から、提案方式を導入することによって、利用するサーバ資源量を低く保ちつつ、Web プロキシサーバのスループット、および Web クライアントホストが感じるドキュメント転送時間が大きく改善されることを示した。

1 章で述べたように、提案方式の適用領域は Web/Web プロキシサーバに限定されない。たとえば P2P ネットワークでは、本稿で提案したソケットバッファの動的割り当て方式を用いることで各ピアにおいて処理することができる TCP コネクション数を増加させることができる。したがって、4 章で示したものと同様の性能向上が期待できるものと考えられる.

文　　献

[1] G. Hasegawa, T. Terai, T. Okamoto, and M. Murata, “Scalable socket buffer tuning for high-performance Web servers,” in *Proceedings of IEEE ICNP 2001*, Nov. 2001.

[2] Proxy Survey, available at *http://www.delegate.org/survey/proxy.cgi*.

[3] G. Peng, “CDN: Content distribution network,” Computer Science Department, Stony Brook University, Tech. Rep. Research Proficiency Exam report, Jan. 2003.

[4] K. L. Johnson, J. F. Carr, M. S. Day, and M. F. Kaashoek, “The measured performance of content distribution networks,” in *Proceedings of the 5th International Web Caching and Content Delivery Workshop*, May 2000.

[5] D. Milojicic, V. Kalogeraki, R. Lukose, K. Nagaraja, J. Pruyne, B. Richard, S. Rollins, and Z. Xu, “Peer-to-Peer Computing,” HP Laboratories, Tech. Rep. HPL-2002-57, Mar. 2002.

[6] T. Okamoto, T. Terai, G. Hasegawa, and M. Murata, “A resouorce/connection management scheme for HTTP proxy servers,” in *Proceedings of Second International IFIP-TC6 Networking Conference*, pp. 252–263, May 2002.

[7] G. Hasegawa, T. Matsuo, M. Murata, and H. Miyahara, “Comparisons of packet scheduling algorithms for fair service among connections on the Internet,” in *Proceedings of IEEE INFOCOM 2000*, Mar. 2000.

[8] 岡本卓也, 長谷川剛, and 村田正幸, “Web プロキシサーバにおける TCP 受信バッファの動的割り当て手法,” **電子情報通信学会技術研究報告** *(IN2002-92)*, pp. 13–18, Oct. 2002.

[9] J. Postel, “Transmission control protocol (TCP),” *Request for Comments (RFC) 793*, Sept. 1981.

[10] R. Braden, “Requirements for internet hosts – communication layers,” *Request for Comments (RFC) 1122*, Oct. 1989.

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

[12] D. Mosberger and T. Jin, “httperf: A tool for measuring Web server performance,” *Performance Evaluation Review*, vol. 26, no. 3, pp. 31–37, Dec. 1998.

[13] P. Barford and M. Crovella, “Generating representative Web workloads for network and server performance evaluation,” in *Proceedings of the 1998 ACM SIGMETRICS International Conference on Meas urement and Modeling of Computer Systems*, pp. 151–160, July 1998.