Web プロキシサーバにおける動的資源管理方式の提案と実装

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, 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
Phone: 06-6850-6616, Fax: 06-6850-6589
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

あらまし 現在のインターネットでは Web サーバにアクセスする際に、Web プロキシサーバを経由する状況が多い。そこで本稿では、Web プロキシサーバがデータ転送を行う際に利用する資源に着目し、Web プロキシサーバの高速・高機能化を実現する新しい動的資源管理方式を提案する。提案方式は、Web プロキシサーバの特性と各コネクションのネットワーク環境を考慮した TCP コネクションへの動的なソケットバッファ割り当てアルゴリズム、及び Web プロキシサーバの資源を浪費するアイドル状態の persistent TCP コネクションを積極的に切断することによって、新規コネクションの要求を受理するアルゴリズムの 2 つから構成される。本稿では、これらの方式の有効性を様々な状況下でのシミュレーションを用いて検証し、Web プロキシサーバの特性を考慮した効率的な資源管理を行うことができることを示す。また、この方式を実コンピュータ上に実装するにあたって考慮すべき点、実装への指針についての検討を行う。
和文キーワード Web プロキシサーバ、TCP (Transmission Control Protocol)、persistent TCP コネクション、ソケットバッファ、コネクション管理

Design and Evaluation of Dynamic Resource Management Scheme for High-Performance Web Proxy 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, tak-okmt}@ics.es.osaka-u.ac.jp

‡Cybermedia Center, Osaka University
1-30, Machikaneyama, Toyonaka, Osaka 560-0043, Japan
Phone: +81-6-6850-6616, Fax: +81-6-6850-6589
E-mail: {hasegawa, murata}@cmc.osaka-u.ac.jp

Abstract In the current Internet, a significant amount of Web document transfer requests are through Web proxy servers. In this paper, we propose a new resource management scheme for Web proxy servers to improve their performance and to reduce Web document transfer time via the Web proxy servers. Our proposed scheme has two components. One is an enhanced E-ATBT (Equation-based Automatic TCP Buffer Tuning), which assigns send socket buffer to TCP connections on the proxy servers according to their characteristics and network condition. The other is a scheme that manages persistent TCP connections on Web proxy servers to avoid rejecting the newly arriving TCP connections due to lack of their resources. We validate an effectiveness of our proposed scheme through simulation experiments. We also discuss implementation issues of our proposed scheme on the actual Web proxy servers.
Keywords Web Proxy Server, TCP (Transmission Control Protocol), Persistent TCP Connection, Socket Buffer, Connection Management

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が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、Web サーバをはじめとする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る状況が生まれつつある。

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 [1]。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、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を用いたデータ転送実験を行い、Web サーバにおける SSBT 方式の有用性を確認した。しかし、現在のインターネットでは、Web サーバにアクセスする際に、Web プロキシサーバを経由する状況が多い [2]。Web プロキシサーバは、ISP (Internet Service Provider) が顧客へのサービスとして提供している場合が多いため、Web プロキシサーバが、顧客からの多数の HTTP アクセスを同時に処理しなければならない。さらに、Web プロキシサーバは、Web プロキシサーバから Web サーバに向けて張られる TCP コネクション (上向きのコネクション) と、クライアントホストから Web プロキシサーバに向けて張られる TCP コネクション (下向きのコネクション) を同時に扱わなければならない。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にもかかわらず、Web プロキシサーバの処理能力が十分でないために、Web ドキュメント転送処理のボトルネックとなる状況が考えられる。すなわち、Web ドキュメント転送時間を改善するためには、Web プロキシサーバの高速・高機能化について考慮する必要がある。

そこで本稿では、まず Web プロキシサーバにおいて TCP コネクションを扱う際の問題点として、ソケットバッファの割り当てに関する問題、及び persistent TCP コネクションの管理に関する問題を指摘する。次に、Web プロキシサーバの性能を改善し、HTTP による Web ドキュメント転送時間を減少させるための新しい資源管理方式を提案する。提案方式は 2 種類のメカニズムから構成される。1 つは、[1] で提案した E-ATBT (Equation-based Automatic TCP Buffer Tuning) 方式を Web プロキシサーバへ適用したもので、各 TCP コネクションのネットワーク環境に応じて、割り当てるソケットバッファサイズを調節する。Web プロキシサーバは、ドキュメントやインラインイメージをダウンロードする際には、TCP の受信ホストになることから、E-ATBT 方式では考慮していなかった受信側ソケットバッファの動的な割り当てを行うように拡張する。もう一つの方式は、Web プロキシサーバの負荷の増大により Web プロキシサーバの資源が不足した際に、新しいコネクションが確立できなくなる問題を解決し、Web ドキュメント転送時間を改善する方式である。この方式は、Web プロキシサーバの資源量を監視し、資源が減少し新規の TCP コネクションが確立できない場合に、長い時間アイドル状態になっている persistent TCP コネクションを強制的に切断し、新たなコネクションが確立できるように資源を解放するものである。本稿では、提案方式を様々な環境でのシミュレーションによって評価し、その結果、提案方式によって繁忙状態の Web プロキシサーバの性能が向上し、ドキュメントのダウンロード時間が改善されることを示す。また、提案方式を実コンピュータ上に実装する際に考慮すべき点、指針についての検討を行う。

2 Web プロキシサーバにおける資源管理

本章では、研究の背景として、Web プロキシサーバにおける資源管理について述べる。まず 2.1 節では、本稿で着目する Web プロキシサーバ、およびサーバが管理する資源について簡単に説明し、2.2 節において、Web プロキシサーバで persistent TCP コネクションを管理する際の問題点を指摘する。

2.1 Web プロキシサーバ

Web プロキシサーバとは、図 1 に示すように、クライアントから Web サーバへの Web ドキュメント転送要求を代理で行う機能を持つサーバである。Web プロキシサーバは、クライアントから Web ドキュメント転送要求を受信すると、要求された Web ドキュメントをオリジナルの Web サーバからダウンロードし、その Web ドキュメントはキャッシュして保存する。そのため、同じドキュメントへの転送要求が別のクライアントから発生した際には、キャッシュしたドキュメントを利用してクライアントへ転送することができるので、ドキュメント転送時間を小さくすることができる。文献 [3] では、Web プロキシサーバを利用することによって、Web ドキュメント転送時間が約 30%向上することが示されている。また、キャッシュしたドキュメントをクライアントへ転送する場合、Web プロキシサーバは、Web サーバ向けに新規に TCP コネクションを確立する必要がないため、ネットワーク内のトラヒック量を抑え、オリジナルの Web サーバの負荷を軽減することができる。

Web プロキシサーバは、Web サーバと同様に様々な環境から非常に多くの TCP コネクションを収容する。しかし、図 1 で示すように、Web プロキシサーバにはその特性上、クライアントへの下向きの TCP コネクションと Web サーバへの上向きの TCP コネクションが存在し、これらの TCP コネクション間には、依存関係が存在する。また、Web プロキシサーバは、上向きの TCP コネクションにおいては受信側として動作し、下向きのコネクションにおいては送信側として動作するため、Web プロキシサーバで TCP コネクションの管理を効率的に行わなければ、Web プロキシサーバ資源が浪費され、Web プロキシサーバで満足なサービスを提供できなくなる。現在のインターネットでは、多くの HTTP アクセスが Web プロキシサーバを経由しているため、Web ドキュメント転送時間を改善するためには、Web サーバの高速・高機能化だけではなく、Web プロキシサーバの高速・高機能化による性能向上が不可欠である。しかし、現在のインターネットにおいて用いられている Web プロキシサーバ [4, 5] では、Web プロキシサーバの持つ資源の管理を考慮した実装はなされていない。

Web プロキシサーバの持つ資源には、mbuf やファイルディスクリプタ、TCP コネクション情報を格納するためのメモリ領域、ソケットバッファ等が挙げられる。これらの資源のうち、mbuf、ファイルディスクリプタ、メモリ領域は TCP コネクションの確立に必要な資源、ソケットバッファはデータの送受信の際に各 TCP コネクションに割り

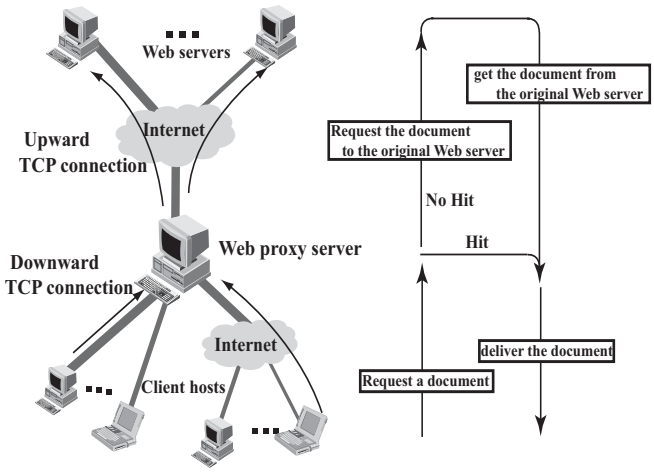


図 1: Web プロキシサーバを介した Web アクセス

当てられる資源である。これらの資源は、Web プロキシサーバが Web ドキュメントを転送するために用いられる TCP コネクションと密接に関係する。たとえば、Web プロキシサーバが収容する TCP コネクション数が増加し、TCP コネクションに必要なサーバ資源が不足すると、新たなドキュメント転送要求に対応するために用いる新規の TCP コネクションを確立できなくなる。この場合、そのドキュメント転送要求は、アイドル状態の TCP コネクションが切断されて資源が解放されるのを待つか、クライアントに 'HTTP_SERVICE_UNAVAILABLE' エラーが返され、接続要求が拒否される。

以降、TCP コネクションに関連する Web プロキシサーバ上の各資源について詳しく説明する。サーバ資源に関して調査した OS は FreeBSD-4.0-RELEASE [6] であるため、FreeBSD に特化したサーバ資源もあるが、Linux 等の他の OS に関しても同様の議論が可能である。

2.1.1 Mbuf

mbuf は、TCP コネクションが確立される時に各コネクション毎に割り当てられる、カーネル領域のメモリ領域である。mbuf は、ネットワークインターフェースとソケットバッファ間のデータ移動に使用される。転送するデータサイズが mbuf のサイズよりも大きい場合には、他のメモリ領域に mbuf cluster を作って、mbuf はその mbuf cluster へのポインタを保持する。mbuf cluster は、転送するデータサイズに応じて作成される。各 TCP コネクションには、最低 1 つの mbuf が割り当てられるので、サーバが同時に確立できる TCP コネクションは、カーネル作成時に指定する mbuf の数と一致する。また、確保できる mbuf cluster の数が小さくなると、各 TCP コネクションが送受信データを格納するメモリ領域が減少するため、メモリ領域が解放されるのを待たなければならず、ドキュメント転送時間が増大する。

しかし、FreeBSD が用意する mbuf 数の初期値は 4096、mbuf cluster 数の初期値は、その 1/4 程度であり、繁忙な Web プロキシサーバにおいては非常に小さい。これらの値はカーネル作成時に指定することができるので、繁忙な Web プロキシサーバを想定した場合には、十分な数の mbuf と mbuf cluster を用意する必要がある。

2.1.2 ファイルディスクリプタ

ファイルディスクリプタは、カーネルとユーザアプリケーションがファイルを識別できるように、ファイルシステム内の各ファイルに割り当てられる。また、ソケットファイルディスクリプタは、TCP コネクションが確立された際に、各コネクションに割り当てられる。そのため、サーバが同時に確立できるコネクション数は、あらかじめ OS で用意されたファイルディスクリプタの数に制限される。FreeBSD のファイルディスクリプタの初期値は 1064 である。

ファイルディスクリプタは、mbuf と違って、カーネルを起動後にもその数を変更することができる。しかし、Squid [4] を始めとするユーザアプリケーションは、アプリケーション起動の際に、使用できるファイルディスクリプタの数に基づいてメモリ領域を確保するため、実行中のアプリケーションに対して、使用できるファイルディスクリプタ数の変動を反映させるのは困難である。したがって、アプリケーション側でファイルディスクリプタの数を動的に変化させることはできない。

2.1.3 その他のメモリ領域

新規の TCP コネクションを確立する際には、`inpcb` や `tcpcb`、ソケット構造体といったコネクション情報を格納するためのメモリ空間を、カーネル領域に確保する必要がある。`inpcb` 構造体は、確立した TCP コネクションの送信元、送信先ホストの IP アドレスやポート番号などを格納するために用いられる。また、`tcpcb` 構造体は、RTT (Roung Trip Time) や RTO (Retransmission Time Out) 等のネットワーク環境の情報を格納するために用いられる。ソケット構造体は、確立したソケットに関する情報を格納するために用いられる。これらの構造体は、各 TCP コネクションに 1 つずつ割り当てられ、TCP による通信を行う際には必要不可欠である。FreeBSD で指定されている、これらの構造体を作成できる数の初期値は 1064 である。この数は、カーネルを作成する時に決定されるため、カーネルの起動後に動的に変更することはできない。

2.1.4 ソケットバッファ

ソケットバッファは、TCP によるデータ転送において、各 TCP コネクション毎に送受信するデータを格納するための領域として割り当てられる。ユーザアプリケーションが TCP による転送を行う時、送信データは、まず送信ソケットバッファにコピーされ、続いて mbuf にコピーされネットワークに送出される。各 TCP コネクションに割り当てられるソケットバッファサイズは、従来の OS の実装においては一定値であるが、各 TCP コネクションは RTT、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。そのため、エンドホスト資源の一つであるソケットバッファを有効に利用するためには、それぞれの TCP コネクションの持つリンク容量 (帯域遅延積) に応じたソケットバッファの割り当てを考慮する必要がある。

我々の研究グループでは、Web サーバにおいて TCP コネクションのスループットを RTT、RTO、パケットロス率などから解析的に推測し、そのスループットに応じて動的にソケットバッファを割り当てる、E-ATBT 方式を提案し、Web サーバへ適用することで、その有効性を確かめた [1]。一方、Web プロキシサーバは、Web サーバと同様に様々な環境から非常に多くの TCP コネクションを収容しているため、E-ATBT 方式の適用が可能であると考えられる。しかし、Web プロキシサーバにはその特性上、Web サーバへの上向きの TCP コネクションとクライアントへの下向きの TCP コネクションの間に依存関係が存在する。また、Web プロキシサーバは、オリジナルの Web サーバへの TCP コネクションにおいては受信側として動作するため、E-ATBT 方式で考慮していない受信側のソケットバッファの制御が必要となる。この問題を解決するための改善方式は 3 章で示す。

2.2 Persistent TCP コネクション

近年、多くの Web サーバや Web ブラウザにおいて採り入れられている HTTP/1.1 [7] の重要な機能の一つに

persistent TCP コネクションが挙げられる。これまでの HTTP/1.0 によるドキュメント転送では、サーバからクライアントへのドキュメント転送が終了すると、すぐにサーバとクライアント間の TCP コネクションが切断されるため、Web ドキュメント内にある複数のインラインイメージを転送するためには、それらのインラインイメージ毎に TCP コネクションを確立して転送を行う。新規 TCP コネクションを確立する時には、3-way ハンドシェイクが行われるため、ドキュメント転送時間の増大の原因となることがこれまでの研究で指摘されている [8]。

これに対し、persistent TCP コネクションを利用した転送では、要求されたドキュメントの転送が終了した後もサーバは一定時間 TCP コネクションを切断せずに保持し、別のリクエストが送信された場合には、その TCP コネクションや保持していたネットワーク情報を再利用することができる。そのため、Web ドキュメント内にある複数のインラインイメージはそのコネクションを再利用して転送できるので、ドキュメント転送時間の改善につながる。しかし、Web サーバや Web プロキシサーバは、persistent TCP コネクションが転送に使われているか否かに関わらず一定時間保持するため、persistent TCP コネクションが転送に使用されていない場合には、サーバの資源が無駄に使用されてしまう。そのため、特に多くの TCP コネクションを収容している Web プロキシサーバにおいては、persistent TCP コネクションを収容するために、多くの資源が浪費されていると考えられる。

[9] では、Web プロキシサーバ上に確立された persistent TCP コネクションが、実際にデータ転送を行っている時間的割合を、Web アクセスモデル等を用いて解析的に導出し、その結果、クライアントと Web プロキシサーバ、及び Web プロキシサーバと Web サーバ間の TCP コネクションの RTT やパケットロス率にかかわらず、persistent TCP コネクションがアイドル状態である時間が非常に長いことがわかった。アイドル状態の persistent TCP コネクションは、Web プロキシサーバの資源を保持し続けるため、persistent TCP コネクションを多く収容することによって、Web プロキシサーバ上の多くの資源が浪費されていると考えられる。

HTTP/1.1 を用いずに HTTP/1.0 を用いて、ドキュメントの転送のたびに TCP コネクションを切断すれば上記の問題は解消されるが、HTTP/1.1 が持つパイプライニングやコンテンツネゴシエーション等の様々な特長 [7] が活用できない。そこで、サーバの負荷が増加しサーバ資源が不足した時には、アイドル状態となっている persistent TCP コネクションを強制的に切断して資源を解放し、新規 TCP コネクションに与えることによって、効率的な資源の利用が可能になると考えられる。次章においてこの資源管理方式の詳細を述べる。

3 提案方式

本章では、前章で説明した Web プロキシサーバにおける資源管理の問題点を解決するための新しい資源管理方式を提案する。

3.1 E^2-ATBT：動的ソケットバッファ管理方式

前章で述べたように、Web プロキシサーバは、Web サーバと同様に様々な環境から非常に多くの TCP コネクションを収容しているため、2.1.4 節において説明した E-ATBT 方式 [1] の適用が可能であると考えられる。しかし、Web プロキシサーバに E-ATBT 方式を適用するためには、Web プロキシサーバの特性上存在する、上向きの TCP コネクションと下向きの TCP コネクション間の依存関係、および E-ATBT 方式が考慮していない受信側のソケットバッファ制御が必要となる。以下でこれらの問題の詳細を述べ、その問題を解決する方法である、E^2-ATBT (Enhanced E-ATBT) 方式を提案する。

3.1.1 コネクションの依存関係

Web プロキシサーバは、クライアントから Web サーバへのドキュメント転送要求を代理で行う。そのため、上向きの TCP コネクションと下向きの TCP コネクションの間には依存関係が存在する。たとえば、下向きの TCP コネクションのスループットが、対応する上向きの TCP コネクションのスループットより大きい場合、E-ATBT 方式によって下向きの 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、上向きの TCP コネクションのスループットが小さいため、下向きのコネクションに大きく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 TCP コネクションに割り当てることによって、これらの 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。このように Web プロキシサーバの特性上存在する 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管理は、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、カーネルで TCP コネクションを識別することは `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。この問題に対する改善方法として、Web プロキシサーバが Web サーバに対しドキュメント転送要求を行う際に、TCP コネクション間の依存関係を表す情報をパケットヘッダに付加する方法が考えられる。この方式は、より正確なソケットバッファの制御が行える半面、Web プロキシサーバおよび Web サーバ双方の対応、および HTTP との連携が必要である。次章におけるシミュレーションにおいては、正確なソケットバッファの制御が行えると仮定し、TCP コネクション間の依存関係を考慮した提案方式の有効性を評価する。現在、実コンピュータ上に E^2-ATBT 方式を実装する方法について検討を行っているが、実装可能性を含めた統合的な評価に関しては今後の課題とする。

3.1.2 受信側ソケットバッファの動的管理

これまで行われてきた TCP の輻輳制御方式、およびスループットに関する研究の多くは、ネットワーク速度がエンドホストの処理に比べて低速であり、TCP コネクションの受信側ソケットバッファの大きさは十分に大きいと仮定されてきた。そのため、従来の多くの OS は TCP コネクションの状態に関係なく、例えば FreeBSD では固定的に 16KB のソケットバッファを受信側に割り当てる。しかし、近年のインターネットの発展によって、ネットワーク帯域が飛躍的に増加し、Web サーバとなるマシンの処理能力が向上している状況では、割り当てられる受信側ソケットバッファサイズが小さく、エンドホストがデータ転送のボトルネックになることが考えられる [10]。

受信側ソケットバッファがボトルネックとなることを避けるためには、受信側のソケットバッファサイズを送信側のウィンドウサイズ以上に設定するようにすればよい。そのためには、受信側で送信側のウィンドウサイズを監視する必要がある。受信側 (ここでは Web プロキシサーバ) が送信側 (ここでは Web サーバ) のウィンドウサイズを知るためには、受信側ソケットバッファの使用率を監視する、あるいはパケットヘッダにウィンドウサイズに関する情報を付加することによって可能である。次章でのシミュレーションにおいては、Web プロキシサーバが TCP コネクションが要求しているソケットバッファサイズを正確に知ることができ、その要求サイズに応じてソケットバッファを制御できると仮定し、提案方式の基本的性質を明らかにする。

3.2 コネクション管理方式

本節では、サーバの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新しい方式を説明する。

提案方式においては、Web プロキシサーバの負荷が小さく、残存資源が十分にあるときは、できるだけ persistent TCP コネクションを収容する。これは、persistent TCP コネクションを利用することで、ドキュメント転送の際に 3-way 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る Web プロキシサーバ資源を解放し、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的に利用することができる。

さらに効果的な資源管理として、persistent TCP コネクションに割り当てられている資源の量を徐々に減らすことが考えられる。TCP コネクションがデータ転送を行っていない時には、ソケットバッファは必要ではない。さらに、persistent 状態にある時間が長くなるにつれて、ネットワーク状態が変化し、保持しているネットワーク環境変数 (ソケットバッファサイズ、ウィンドウサイズ) は不正確になるため、そのコネクションが再利用された際にスループットの低下につながることが考えられる。そこで提案方式では、persistent TCP コネクションに割り当てられている送受信用ソケットバッファの大きさを、アイドル状態である時間に応じて徐々に減少させる。

4 シミュレーションによる性能評価

本章では、3 章で提案した Web プロキシサーバにおける動的資源管理方式に関するシミュレーション結果を示し、その性質に関する考察を行う。シミュレーションは、ns-2 [11] を用いて行った。シミュレーションで用いたネットワークトポロジーを図 2 に示す。図 2 において、クライアントと Web プロキシサーバ間、および Web プロキシサーバと Web サーバ間のリンク帯域を 100 [Mbps] とする。また、リンクのパケット廃棄率を 0.0001、0.0005、0.001、0.005、0.01 から選択し、クライアントと Web プロキシサーバ間の伝播遅延時間を 10 [ms] から 100 [ms] まで、Web プロキシサーバと Web サーバ間の伝播遅延時間を 10 [ms] から 200 [ms] とし、様々な環境のサーバ・クライアント間のドキュメント転送を考慮する。Web サーバの台数を 50 台、クライアントの数を 50、100、200、500 台と変化させて、シミュレーションを行い、提案方式の有効性を確認する。シミュレーション時間は 1000 [sec] とし、Web プロキシサーバが保持するソケットバッファサイズの合計は 3200 [KB] とする。これは、従来方式で固定的に割り当てられる送信/受信バッファサイズである 16 [KB] に、Web プロキシサーバが同時に扱えるコネクション数 (このシミュレーションでは 200 本とする) を掛けたものに相当する。また、E^2-ATBT 方式がソケットバッファの割り当てサイズを計算する間隔を 1 秒とした。

シミュレーションにおいては、各クライアントは 50 台のサーバからランダムに 1 台を選択し、[12] で示されている Web ドキュメント分布、およびアクセスモデルに従って Web プロキシサーバに対してドキュメント転送要求を行う。本稿では、Web プロキシサーバにおける資源管理および TCP コネクションの管理を目的としているため、Web プロキシサーバでのキャッシュ管理は考慮せず、クライアントから要求されたドキュメントが Web プロキシサーバにキャッシュされている確率 (ヒット率) H_r を 0.5 と設定する。Web プロキシサーバはそのヒット率に基づいてクライアントから要求されたドキュメントを Web プロキシサーバから直接クライアントへ転送するか、Web サーバからダウンロードを行った後にクライアントへ転送するかを決定する。

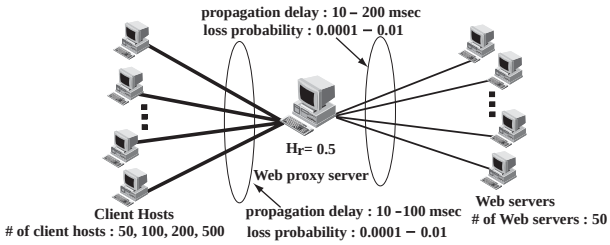


図 2: シミュレーショントポロジー

評価に際しては、以下に示す 4 方式の比較を行う。

- [方式 1] 従来方式
- [方式 2] E^2-ATBT 方式
- [方式 3] 方式 2 に 3.2 節で述べたコネクション管理を加えた方式
- [方式 4] 方式 3 にアイドル状態の persistent TCP コネクションのソケットバッファを徐々に減らす機能を加えた方式

また、Web プロキシサーバにおける資源量が不足している状況を想定するために、Web プロキシサーバが同時に確立できる TCP コネクションの最大数 N_{max} を Web プロキシサーバの資源量の限界を表す閾値とする。方式 1、2 においては、Web プロキシサーバで扱うコネクション数が N_{max} を越えると、新規のリクエストを受理しないが、方式 3、4 ではアイドル状態の persistent TCP コネクションを積極的に切断することによって、新規の TCP コネクションを受けつける。また、方式 4 ではアイドル状態の persistent TCP コネクションに割り当てられているソケットバッファを、サイズが 1 [KB] になるまで 3 秒ごとに半分に減らすものとする。

以降では、以上の条件の下でシミュレーションを行った結果を示し、提案方式と、各 TCP コネクションに固定的に 16 [KB] の送信側/受信側ソケットバッファを割り当てる従来方式との比較を行う。

4.1 HTTP/1.0 と HTTP/1.1 の比較

ここでは、Web ドキュメントを転送するプロトコルとして、ドキュメント転送後すぐに TCP コネクションを切断する HTTP/1.0 と persistent TCP コネクションを標準機能として有する HTTP/1.1 を用いた場合の比較をシミュレーションを用いて行う。図 3 は、クライアント数に対する、シミュレーション時間内に Web プロキシサーバが Web サーバおよびクライアントに対して送受信を行った総転送データ量との関係を表している。それぞれ、HTTP/1.0 と HTTP/1.1 を用いて、方式 1、2 について評価を行っている。

まず、どちらの HTTP プロトコルを用いた場合も、方式 1 に比べて方式 2 のデータ転送量が増加していることがわかる。これは、E^2-ATBT 方式によって、各 TCP コネクションに適切なソケットバッファサイズが割り当てられた結果、スループットが向上したためである。また、Web プロキシサーバが繁忙な状況にならないクライアント数が 50、100 の場合、HTTP/1.1 を使用することで Web プロキシサーバの性能が向上していることがわかる。これは、HTTP/1.1 の機能である persistent TCP コネクションを利用して、Web ドキュメントのインラインイメージが新規の TCP コネクションを確立するときに必要な 3-way ハンドシェイクを省略する効果が現れたためである。一

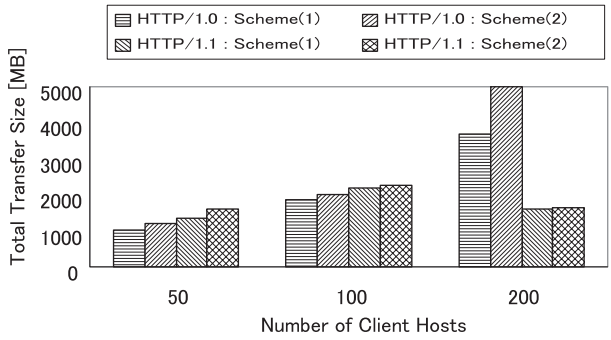


図 3: HTTP/1.0 と HTTP/1.1 の比較

方、クライアント数が 200 の場合、Web プロキシサーバは繁忙な状況になり、アイドル状態の persistent TCP コネクションによって資源が浪費されてしまう HTTP/1.1 の性能が、HTTP/1.0 に比べて悪化している。HTTP/1.0 は、ドキュメント転送後すぐにコネクションを切断するため、繁忙な状況でも新規のリクエストにすばやく対応することができる。一方、HTTP/1.1 の場合、アイドル状態の persistent TCP コネクションが資源を占有してしまい、タイマによってコネクションが切断されるまで、新規の TCP コネクションを確立できないため、性能が劣化する。このように、Web プロキシサーバの資源に余裕がある状況では、persistent TCP コネクションは Web プロキシサーバのスループットを向上させることができるが、Web プロキシサーバの資源が不足するような繁忙な状態では、逆にアイドル状態の persistent TCP コネクションの影響により Web プロキシサーバの性能が劣化することが明らかとなった。

4.2 コネクション管理方式の性能評価

次に、persistent TCP コネクションの管理方式の性能に関する評価結果を示す。図 4 は、クライアントの台数に対する、シミュレーション時間内に Web プロキシサーバが Web サーバおよびクライアントに対して送受信を行った総転送データ量との関係を表している。図 4(a)、図 4(b)、図 4(c) はそれぞれ Web プロキシサーバが persistent TCP コネクションを保持する時間を 5 秒、15 秒、30 秒と変化させた場合の結果である (多くのアプリケーションでは、15 秒に設定される)。図より、クライアント数が多くなり Web プロキシサーバの資源が不足するような状況では、方式 3 によって Web プロキシサーバの性能が飛躍的に向上することがわかる。これは、Web プロキシサーバがすべてのコネクション確立要求を処理できなくなるような状況の場合、方式 3 は新規のリクエスト要求に対応するためにアイドル状態の persistent TCP コネクションを積極的に切断するためである。また、方式 4 はクライアント数が少ない場合に効果が表われていることが、クライアント数が多い場合、あまり性能が改善されていないことがわかる。これは次のような理由が考えられる。クライアント数が少ない場合は、Web プロキシサーバの資源に余裕があるため、ほとんどの TCP コネクションを収容でき、徐々に減少させたソケットバッファを他の TCP コネクションに再割り当てを行う方式 4 が大きな効果を示す。一方で、クライアント数が多く Web プロキシサーバが資源が不足するような状況では、アイドル状態の persistent TCP コネクションのソケットバッファを減少させる前に、そのコネクション自体が切断されてしまうため、persistent TCP コネクションに対して方式 4 がほとんど機能しない。

次に、persistent TCP コネクションを保持する時間を変化させた場合の評価を行う。図 4(a)、 4(b) の方式 1、2 を比較すると、クライアント数が少ない場合、persistent TCP コネクションを保持する時間が短いほど連続したリクエスト要求を同じコネクションを再利用して行う persistent TCP コネクションの効果が小さくなることがわかる。逆に、クライアント数が多くなり、Web プロキシサーバが繁忙な状況になると、保持する時間が短い方がアイドル状態の persistent TCP コネクションを切断しやすくなるため、Web プロキシサーバの性能が向上している。一方、図 4(b)、4(c) の方式 1、2 を比較すると、persistent TCP コネクションを保持する時間が長くなると、クライアント数が少ない場合でも、アイドル状態の persistent TCP コネクションを長い時間保持してしまうために、Web プロキシサーバの性能が劣化する。以上の結果より、persistent TCP コネクションの効果は、コネクションを保持する時間、および Web プロキシサーバの負荷状態に大きく依存していることが明らかとなった。

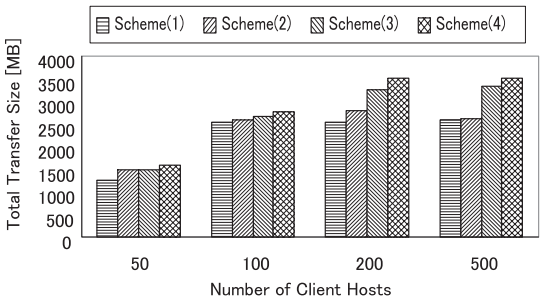
これに対して、提案方式である方式 3、4 は、クライアント数、persistent TCP コネクションを保持する時間の大小にかかわらず、最も優れた性能を示していることがわかる。これは、クライアント数が少ない場合には、persistent TCP コネクションの効果によって転送効率が上がり、クライアント数が多い場合には、積極的にアイドル状態の persistent TCP コネクションを切断する提案方式によって、Web プロキシサーバ資源の浪費を防ぎ、ドキュメント転送要求を効率的に処理できるためである。

しかし、前節の HTTP/1.0 を用いた場合の結果と比べると (図 3、4(b) 参照)、提案方式の方が総データ転送量が小さいことがわかった。これは次のような理由が考えられる。HTTP/1.0 の場合、データ転送が終了するとすぐにコネクションを切断し、資源を解放するため、アイドル状態の TCP コネクションは発生しない。そのため、確立された TCP コネクション上では常にデータ転送が行われている。これに対し、提案方式では、persistent TCP コネクションを最低 1 秒保持するため、HTTP/1.0 の場合に比べると短い時間ではあるがアイドル状態の TCP コネクションが存在し、資源の浪費が生じてしまうためである。この問題を解決するためには、persistent TCP コネクションの保持時間のタイマ粒度を細かくする等の、Web プロキシサーバのタイマ管理の改善が必要であると考えられる。この改善によって、上記の問題は解決し、かつ HTTP/1.1 が持つパイプライニングやコンテンツネゴシエーション等の様々な特徴を活用することができるため、飛躍的な性能改善が期待できる。

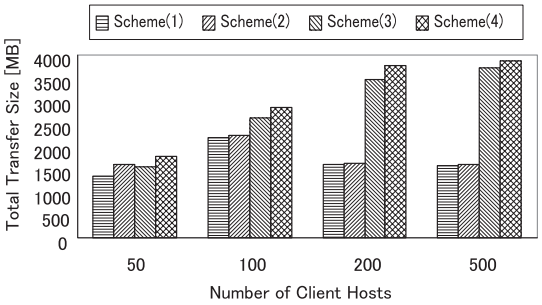
4.3 応答時間の評価

最後に、ユーザが感じる性能指標としてドキュメント転送要求に対する応答時間の評価結果を示す。ここでは、応答時間をクライアントがドキュメント転送要求を送信してから、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。図 5 は、ドキュメントサイズに対する平均応答時間を示している。図より、クライアント数が増加すると、提案方式によって応答時間が大きく改善していることがわかる (図 5 (b)-(d))。しかし、クライアント数が 50 台の場合、提案方式の効果があまり現れていない。これは、Web プロキシサーバの資源に余裕があり、新規の TCP コネクションをすぐに確立することができるためである。

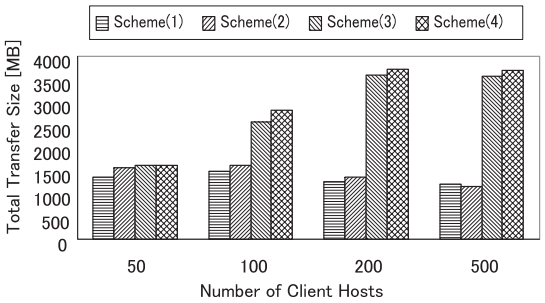
また、前節の結果と同様、方式 3 はクライアント数が多いときに、また、方式 4 はクライアント数が少ないときに効果的であることが図 5(c)、5(d) よりわかる。これは前節で示した結果と同様、クライアント数が少ない場合は、方式 4 がアイドル状態の persistent TCP コネクションに割り当てられているソケットバッファを徐々に減らし、他の TCP コネクションに割り当てるので、それらの TCP コネクションの応答時間が向上し、クライアント数が多い場合は、方式 3 によってアイドル状態の persistent TCP コネクションが積極的に切断され、Web プロキシ



(a) コネクション保持時間：5 秒



(b) コネクション保持時間：15 秒



(c) コネクション保持時間：30 秒

図 4: コネクション管理方式の性能評価

サーバが新規の TCP コネクションを確立することができるようになるため、新規の TCP コネクションの応答時間が向上するためである。

以上の結果より、本稿のすべての提案を含んだ方式 4 が、Web プロキシサーバの性能評価、応答時間の性能評価の結果より、クライアント数、およびプロキシサーバの負荷状態にかかわらず、最も優れた性能を示すことが明らかとなった。

5 実装に関する検討

本章では、3 章で述べた提案方式の実コンピュータ上への実装の指針について述べる。なお、実装に関しては FreeBSD-4.0-RELEASE を用いている。

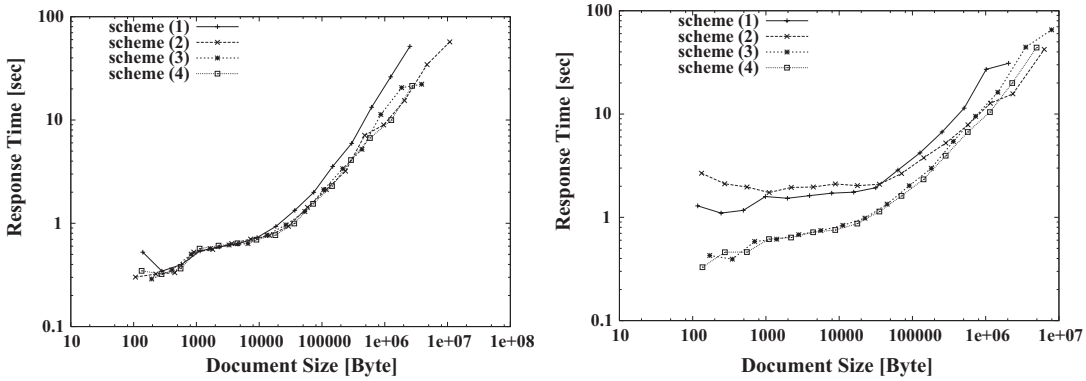
動的ソケットバッファ管理方式に関しては、 [1] において Web サーバへの実装を行い、その有効性を確認している。3.1 節で説明したように、動的ソケットバッファ管理方式を Web プロキシサーバに適用するためには、上向きの TCP コネクションと下向きの TCP コネクションの依存関係の考慮、および受信側ソケットバッファの制御が必要となる。しかし、カーネルでプロキシサーバにおける TCP コネクション間の依存関係を識別することはできない。また、受信側ではスループットを推測するために必要なネットワーク情報 (RTT、パケットロス率) が取得できないため、E-ATBT 方式をそのまま適用することができない。これに対する改善する方法として以下の 2 つが考えられる。

- Web プロキシサーバにおいて、クライアント間の TCP コネクションの送信側ソケットバッファの使用率を監視し、割り当てられたソケットバッファを使い切れていない場合には割り当て量を強制的に減少させる。
- Web プロキシサーバが Web サーバにドキュメント転送を要求する時に、依存関係、ソケットバッファに関する情報をパケットヘッダに付加する。

前者の方式は Web プロキシサーバの変更を行うだけで実現可能であるが、ソケットバッファの使用率は瞬間的に大きく変動するため、一定時間ごとの観測では適切なソケットバッファサイズの割り当てが行えないことを確認した。後者の方式は、パケットヘッダに送受信側それぞれの状況を書き込むことにより、正確なソケットバッファの制御が行うことができる半面、Web プロキシサーバおよび Web サーバ双方の対応、および HTTP プロトコルとの連携が必要である。現在は、後者の方式を含めたより簡単な実装方法を検討している段階である。

3.2 節で述べたコネクション管理方式を実現するためには、まず Web プロキシサーバの残存資源の監視を行う必要がある。TCP コネクションを確立するために必要な Web プロキシサーバの資源としては、2.1 節で述べたように、mbuf、mbuf cluster、ファイルディスクリプタ、およびその他のメモリ空間が挙げられる。カーネルの起動後にこれらの資源量を動的に変更することはできないが、現在使われている資源量および使用可能な資源量は `sysctl` システムコール等を用いて取得することができる。そこで、それぞれの資源の使用率を監視し、一つでもその使用量があらかじめ設定した閾値を越えれば、データ転送を行っていない persistent TCP コネクションを切断し、Web プロキシサーバの資源の確保を行う。

次に、persistent TCP コネクションの管理方法について説明する。Web プロキシサーバは、Web サーバから Web プロキシサーバ、あるいは Web プロキシサーバからクライアントへのドキュメント転送が終了した後に、その TCP コネクションが persistent 状態になったことを検出すると、新しく作成したシステムコールを使用して、その TCP コネクションに関する情報 (ソケットファイルディスクリプタ、プロセス番号) を、カーネル領域に新しく作成した構造体に格納する。そして、persistent 状態にいる時間が長いもの、つまり古い persistent TCP コネクションから順に参照できるように、リスト型のデータ構造を使用する。以後、このリストを ‘time scheduling list’ と呼ぶ。新規に persistent 状態になった TCP コネクションは、リストの末尾に追加されるので、Web プロキシサーバは persistent TCP コネクションを古い順に管理することが可能である。また、persistent 状態の TCP コネクションを使用したデータ転送が再開された時、およびタイマ



(a) クライアント数 50

(b) クライアント数 100

(c) クライアント数 200

(d) クライアント数 500

図 5: シミュレーション結果: 応答時間

によって persistent TCP コネクションが切断される時は、同様にシステムコールを使用して、該当する TCP コネクションを time scheduling list から削除する。この操作によって、その TCP コネクションは persistent 状態ではないと認識される。Web プロキシサーバの負荷が増加し、サーバ資源が不足したときには、time scheduling list の先頭にあるコネクション、すなわち最も古い persistent TCP コネクションから切断し、新規の TCP コネクションのための資源の確保を行う。time scheduling list は、persistent 状態の時間が長い TCP コネクションから順に管理しているため、このような操作が可能となる。これらの TCP コネクションに対する操作は、すべてポインタ操作などによって高速に行うことができる。現在、以上の指針に基づいて提案方式のコンピュータ上への実装を行っている段階である。

6 まとめと今後の課題

本稿では、動的ソケットバッファ管理方式である E-ATBT 方式を拡張した E^2-ATBT 方式、および persistent TCP コネクションの動的管理方式からなる Web プロキシサーバにおける動的資源管理方式を提案した。提案方式の有効性は、シミュレーションによって評価し、その結果、提案方式によって、Web プロキシサーバの性能が向上し、ドキュメントのダウンロード時間を短縮することが可能であることを確認した。また、提案方式の実コンピュータ上への実装の指針についても述べた。

今後の課題としては、5 章で述べた実装の指針に基づき、本提案方式を実 Web プロキシサーバへ実装し、実ネットワーク上で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、通信・放送機構創造的情報通信技術研究開発推進制度「インターネットにおける多ユーザ間の公平性に関する研究」、及び (財) 電気通信普及財団の研究助成「次世代高速インターネットのための Web サーバ技術に関する研究」によっている。ここに記して謝意を表す。

参考文献

[1] G. Hasegawa, T. Terai, T. Okamoto, and M. Murata, “Scalable Socket Buffer Tuning for High-Performance Web Servers,” in *Proceedings of IEEE ICNP2001*, Nov. 2001.

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

[3] A. Feldmann, R. Caceres, F. Douglis, G. Glass, and M. Rabinovich, “Performance of Web Proxy Caching in Heterogeneous Bandwidth Environments,” in *Proceedings of IEEE INFOCOM ’99*, pp. 107–116, 1999.

[4] Squid Home Page, available at `http://www.squid-cache.org/`.

[5] Apache proxy mod_proxy, available at `http://httpd.apache.org/docs/mod/mod_proxy.html`.

[6] FreeBSD Home Page, available at `http://www.freebsd.org/`.

[7] 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.

[8] B. Krishnamurthy, J. C. Mogul, and D. M. Kristol, “Key Differences between HTTP/1.0 and HTTP/1.1,” *Proceedings of the WWW-8 Conference*, May 1999.

[9] 岡本 卓也、寺井 達彦、長谷川 剛、村田 正幸, “HTTP Proxy サーバにおける動的コネクション管理方式,” **電子情報通信学会技術研究報告** (TM2001-51), pp. 47–52, Nov. 2001.

[10] Mark Allman, “A Web Server’s View of the Transport Layer,” *ACM Computer Communication Review*, vol. 30, Oct. 2000.

[11] The VINT Project, ”UCB/LBNL/VINT network simulator -ns (version 2).” available at `http://www.isi.edu/nsnam/ns/`.

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