NAIST-IS-DT0161029

博士論文

大規模 WWW サーバの設計支援のための性能評価技術に関する研究

中山 貴夫

2003 年 12 月 25 日

奈良先端科学技術大学院大学
情報科学研究科 情報システム学専攻

本論文は奈良先端科学技術大学院大学情報科学研究科に
博士 (工学) 授与の要件として提出した博士論文である。

提出者：　中山 貴夫

審査委員：　砂原 秀樹 教授
山口 英 教授
藤川 和利 助教授

大規模 WWW サーバの設計支援のための性能評価技術に関する研究*

中山 貴夫

内容梗概

World-Wide Web (WWW) は現在，インターネットにおける情報流通の主流となっている．例えば，チケット予約ができるようになったりオークションが行われたり WWW 上でさまざまなサービスが展開されており，オリンピック，ワールドカップといった大規模なイベントの中継までも WWW 上で行われるようになった．また，新聞，テレビやラジオに加えた日常的な情報収集の手段としても利用されている．

こうして，WWW が日常の情報基板の一つとして用いられるようになると，WWW サービスは常に安定したサービス提供を求められるようになってきた．特に商用やイベント中継に用いられるような WWW サーバを安定して運用することは強く求められている．“安定したサービス” というのはユーザが要求したコンテンツが迅速に表示されたり，何かサーバに障害がおこれば即座に普及できるような体制が整っていることをいう．そのためには，メンテナンス体制を整えたり，サイトに応じた適切なサーバへの投資が重要となる．特に，新たに WWW サーバシステムを構築する際には，導入するシステムの正確な性能評価および性能の限界の調査は重要となる．

そこで本研究では，大規模な WWW サーバの設計時に有効となるサーバの性能評価を行うためのシステムを開発した．サーバシステムの性能評価はベンチマークを用いるのが一般的である．これまでにさまざまな WWW サーバベンチマークシステムが開発されてきた．しかし，WWW で扱われるコンテンツの内容は WWW が登場して以来変化を続けている．そのため，以下に挙げるように既存のベンチマークでは対応できない部分が多く存在する．

1) 従来の WWW ベンチマークシステムでは，ベンチマークの対象となるファイルは一つだけ，もしくはベンチマークシステムが用意したアクセスパターンによるものであった．しかし，実際の WWW サーバでは一つだけのファイルにアクセスされるということ

*奈良先端科学技術大学院大学 情報科学研究科 情報システム学専攻 博士論文, NAIST-IS-DT0161029, 2003 年 12 月 25 日.

は現実的ではく，アクセスパターンにもサーバによって異なるため，ベンチマークが用意したアクセスパターンでは十分でない場合がある．また，CGI のようなアクセスするたびに内容が変化する動的なコンテンツについても，その評価が行われていない．

2) WWW で個人情報を送受信する際には SSL による暗号化が用いられることが多い．特にオンラインショッピングやユーザ登録などを頻繁に行うような大規模なサーバでは SSL による暗号化を考慮した設計が必要となる．そのため，SSL を用いた際のサーバの性能評価を行う必要が出てくる．

3) 現在，次世代 IP である IPv6 の普及がすすんでおり，IPv6 で運用される WWW サーバも出てきている．そのため，IPv6 環境下でのサーバの観測，監視について考慮する必要が出てきている．しかし現状では IPv6 に対応したベンチマークは存在していなかった．

そこで，本研究では，これらのことを考慮した新たなベンチマークシステム ANMA を設計，実装した．つまり，ベンチマークを行う際に，SSL による暗号化を用いるか否か，アクセスするファイルをどうするかといったアクセスパターンを自由に指定でき，IPv6 と IPv4 の両方のアドレスファミリでのリクエストを送信可能なベンチマークである．このベンチマークにより，上記で述べた点を解決し，現実の WWW サーバに近い環境での性能測定が可能となった．

また，運用中のサーバの性能を評価するため，サーバに負荷をかけずリアルタイムにサーバの状態を観測できるシステムの設計，実装を行った．このシステムはサーバに負荷をかけないパケットモニタリングを利用し，その結果からサーバの状態を判別するシステムである．そして実際に運営されたサーバの観測結果やベンチマークによる実験から，このシステムを用いてサーバの状態を把握できることを確認した．

キーワード

インターネット，WWW，性能評価

Research on the performance evaluation technology for the design of a large-scale WWW server*

Takao Nakayama

Abstract

World-Wide Web (WWW) is the most important service on the Internet. WWW became an information sources like a newspaper or television. Moreover, WWW offers various services such as ticket reservation and broadcast of several events. Thus, the users of WWW are also increasing in number as WWW service becomes rich.

Consequently, users desire that service providers stabilize WWW services. We define the stable service as that demanded contents from users are displayed quickly and that a WWW server recovers immediately when the server break down. In case of large scale sites, the server maintenance is the most important.

Then, this thesis describes the performance evaluation technology of the WWW server for the design of a large-scale WWW server. Benchmark systems can make clearly the performance indices of the WWW servers. Server administrators can understand many performance indices with benchmark systems. Since WWW appeared, the contents treated by the WWW are changing. Therefore, the present benchmark does not correspond to the following points.

1: The target file of the conventional WWW benchmark is only one. A benchmark is performed using the access pattern that the benchmark system prepared. However, a WWW server receives the request to much file in practice. And, an access pattern changes with WWW servers. Therefore, a server administrator cannot evaluate a performance by the access pattern which the system prepared enough. Moreover, an administrator cannot evaluate the contents from which the contents differ each time.

2: When we transmit and receive personal information, encryption communication by SSL is often used. When an administrator designs the server of on-line shopping, he

*Doctor's Thesis, Department of Information Systems, Graduate School of Information Science, Nara Institute of Science and Technology, NAIST-IS-DT0161029, December 25, 2003.

have to take the load of the encryption communication by SSL. Therefore, we evaluates the performance of the server which used SSL.

3: With the spread of IPv6 technology, we can use WWW service in IPv6. Therefore, measurement in the IPv6 environment of a WWW server is needed. But principal WWW server benchmark softwares cannot use IPv6.

Therefore, I develop a new benchamrk system, called ANMA. This benchmark system has the request transmitting function in IPv6, a transmitting function in SSL and the function which accesses different contents in a server. And qwe can set up an access pattern freely. With this benchmark, a server administrator can measure the performance of a WWW server now in the environment near an actual server.

To evaluate the performance of the server in operation, I developed the real-time state observation system for WWW server, using the packet monitoring system. This system shows an administrator three states of normality, saturation, and a stop on real time. I showed the validity of our system through observation of an actual server and an experimental result using benchmark.

Keywords:

Internet, World-Wide Web, performace evaluation

目 次

図 目 次

表 目 次

第 1 章　序論

1946 年に ENIAC が開発されて以来 55 年あまりが過ぎた．ENIAC 以降，さまざまな計算機が開発され，現在も計算機の進歩はとどまることを知らない．初期の計算機は単体で利用するスタンドアロン型での利用しかされていなかった．また，大型で巨額であったため，利用者が計算機室に足を運んでの利用形態であった．1960 年代になると計算機利用者からの要求で，遠隔からジョブを投入するリモートジョブ入力やその結果を遠隔で取り出すためのリモートプリンタのために中央の計算機と遠隔の入出力装置との間を結合する形でのスター型のネットワークが構成された．

1960 年代の半ばになるとこのネットワークを利用し，一つのコンピュータのユーザ間でメールを用いた情報交換が行われるようになった．これが計算機ネットワークによるコミュニケーションの始まりである．

1969 年には ARPANET という全米規模の計算機ネットワークの構築が開始される．当初は 4 箇所を結んだパケット通信の実験から始まり，軍事目的や研究用のネットワークとして開発された ARPANET は 1986 年に NSFNET へと引き継がれた．NSFNET には多くの大学や研究機関，企業が参加し，これが現在のインターネットの基板となった．NSFNET が 1990 年に商用利用を認められてからは軍事，研究目的のみではなく，一般の利用者が爆発的に増加してきた．これらのネットワークでも複数の計算機の間でファイルの交換やメールの転送が実現され，計算機ネットワークによるコミュニケーションが行われていた．

そして 1989 年に開発された World-Wide Web (WWW) により，ネットワーク全体を一つのデータベースとみなすウェブという利用方式が考え出される．当初，WWW でやりとりされるデータはテキストのみであった．しかし 1993 年にイリノイ大の NCSA による WWW ブラウザ Mosaic[1] は画像を表示することが可能であった．このグラフィカルな分かりやすいユーザーインタフェース，ネットワークを使っているという意識なしに紙メディアよりも簡単に大量の情報にアクセス出来る，画像や音声も取り出せるとい

う特徴が広く受け入れられ，WWW により簡単に情報が引き出せるようになってから，インターネットが専門家以外に爆発的に広まった．Mosaic が登場した 1993 年にはインターネットのトラヒックが大幅に伸びたという報告もある [2].

初期の WWW についての研究は，WWW サーバのログ解析によるサーバの内部構造解析やモデリングが行われた [3]．そして，WWW で利用されている HTTP[4][5] についての解析が行われ，改良された [6][7][8][9]．これらの結果をもとにしたベンチマークも開発された [10][11][12].

このような研究を経て，WWW は現在，インターネットにおける情報流通の主流となっており，日常の情報基板の一つとして確立している．例えば，チケット予約ができるようになったりオークションが行われたり WWW 上でさまざまなサービスが展開されており，オリンピック，ワールドカップといった大規模なイベントの中継までも WWW 上で行われるようになった．このようにサービス充実に伴い，WWW の利用者も増加し，現在では，インターネット上のトラヒックの約 80%を WWW が占めるようになってきている [13]．また，2003 年の時点で約 4000 万の Web サイトがあるという報告もある [2].

こうして WWW サービスが普及するにつれ，サービス提供者には常に安定してサービスを提供する事が求められるようになってきている．安定したサービス提供がされている状態というのはユーザが要求したコンテンツが迅速に表示されたり，何かサーバに障害がおこれば即座に普及できるような体制が整っていることをいう.

サーバを安定して運用するためには，サーバシステムの管理，メンテナンス体制を整えることや，サイトに応じた適切なサーバへの投資が重要となる．そのためには現状のサーバシステムの正確な性能評価および性能の限界の調査，今後のアクセス予測などが必要となってくる.

そこで本論文では，WWW サーバを安定して運用できるよう，サーバシステムの評価手法や監視手法を研究の対象としている.

サーバシステムの監視手法については，運用中のサーバを監視しなくてはならないことから，サーバに負荷をかけずリアルタイムに観測できる手法としてパケットモニタリングを利用し，その結果からサーバの状態を判別するシステムを開発した．この詳細は 2 章で述べる.

サーバシステムの評価を行うには，ベンチマークを用いるのが一般的である．上記で述べた他，Httperf[14] や SPECweb[15]，Webstone[16] などの WWW ベンチマークシステムが開発されている．しかし，WWW とインターネットを取り巻く環境が変化するに

つれ，ベンチマークに求められる要素も変化している．

まず，インターネットのネットワーク層で用いられているインターネットプロトコル (IP) がバージョン 4 からバージョン 6 へと移行しようとしている．このことによってサーバの挙動がどう変化するか，また IPv4 で開発，研究されてきた計測技術が IPv6 の環境でどのように生かされるかについて検討し，新たな WWW ベンチマークシステムを開発した．この詳細は 3 章で述べる．一方，WWW 上で扱われるコンテンツについては当初はテキストデータのみであったが画像や音声，動画などのマルチメディアファイルが扱われるようになったり，Common Gateway Interface (CGI) を用いた掲示板のようなコンテンツも増加している．また，WWW 上で個人情報が取り扱われるようになってきていることから暗号化通信も行われるようになった．従来の WWW ベンチマークシステムではこのようなことは考慮されていないため，現在のコンテンツ状況に適応したベンチマークの開発を行った．この詳細については 4 章で述べる．

1.1. 本論文の構成

1 章では，計算機ネットワークとインターネットの歴史，および計算機ネットワークによるコミュニケーションにおける WWW の重要性について述べた．また，WWW についての過去の研究についても触れ，現状と問題点を挙げた．

2 章では，リアルタイムで WWW サーバの状態を監視するシステムについて述べる．このシステムは，パケットモニタリングによる WWW サーバ観測システム ENMA を用い，その出力結果から WWW サーバの状態を管理者に通知するシステムである．この章では，WWW サーバの状態の定義やシステムの設計と実装について述べる．

3 章では，WWW サーバのベンチマークシステム ANMA の設計と実装について述べる．ANMA は従来の WWW ベンチマークシステムにはない IPv6 によるリクエスト送信機能を備えている．この機能の設計と実装について述べ，実験により IPv6 と IPv4 での WWW サーバの挙動の違いを明らかにする．

4 章では，3 章で開発した ANMA の付加機能について述べる．従来の WWW ベンチマークシステムにはない，SSL による暗号化リクエストの送信機能とさまざまなファイルへのアクセス機能を実装した．この機能により，SSL による暗号化がサーバにどう影響を与えるか，やさまざまなアクセスパターンでのサーバ測定が可能となる．

5 章では，ANMA による性能測定の例としてキャッシュサーバの測定を行う．ANMA

を用いることで既存のキャッシュサーバベンチマークにはない自由なアクセスパターンによる性能評価が可能となる．この章では，日立製作所が開発してるキャッシュサーバの測定を行う．

第 2章　リアルタイム WWW サーバ状態監視システム

World-Wide Web (WWW) は日常の情報基板として利用されたり，チケットサービスやイベント中継など，インターネットで最も重要なサービスである．サービスの品質を考えた時，WWW サーバ管理者は常にサーバの挙動を監視する必要がある．特にチケット予約システムやオンラインショッピングが動作しているサーバにおいては，ユーザからのリクエストに対してサーバがすぐ応答できるような状態に保っておかなくてはならない．サーバを常によい状態に保っておくため，管理者はサーバの同時コネクション数や応答時間，スループットなどのさまざまな性能指標を観測している．しかし，サーバを運営する際にはこれらの指標では十分とは言えない．そこで，我々は運営のための別の指標として “WWW サーバの状態” を定義する．本章では，WWW サーバの状態の定義の説明，およびリアルタイムでサーバの状態を判別するシステムの設計および実装について述べる．

2.1.　まえがき

インターネットは急速に広まり，日常生活の情報基板として必要不可欠な物となっている．特に，WWW は社会的な情報インフラとして成り立っている．その理由としては，見栄えの良さや誰にでも手軽に扱える GUI ベースのクライアントアプリケーションの存在がある．さらに，チケット予約システムや WWW を通じた買物，ワールドカップなど大規模なイベントの中継も WWW 上で行われるようになっている．このようなサービスの充実に伴い WWW の利用者も増加している．インターネットのトラヒックの 80%を WWW が占めているという報告もある [13].

安定したサービスとは，ユーザが要求したコンテンツが迅速に表示され，サーバに障害が起こった時に，すぐに復旧できる体制が整っていることを指す．利用者はサービス提供者に安定した WWW サービスの提供を求めるようになってきた．特に大規模なサイ

トでは，サーバのメンテナンスは特に重要である．しかし，サーバ管理者は自分自身の感と経験に基づいてサーバを管理している．それゆえ，我々は実際のサーバ運営を通じてサーバの状態の定義を行った．そして定義したサーバの状態を利用して運営のための指標を導き出し，その指標に基づいてサーバ管理者の補助となるシステムを構築した．

2.2.　WWW サーバ観測手法

これまでにさまざまな WWW サーバの観測手法が開発されている．それは，ベンチマークシステム，カーネルモニタリングシステム，パケットモニタリングシステムである．本節では，これらの観測手法について説明し，どの手法がリアルタイムでのサーバ観測に適しているか議論する．

2.2.1　ベンチマークシステム

ベンチマークシステムにより，WWW サーバのさまざまな性能指標が明らかになる．代表的な WWW ベンチマークシステムとしては，Httperf[14]，SPECWeb[15]，Webstone[16] が挙げられる．しかし，ベンチマークによるサーバの計測を行う際には，サーバシステムを実環境から切り離さなくてはならない．性能評価を行うために運用中のサーバを停止することは一般に困難である．よって，運用中のサーバ観測にはベンチマークを適応できないと言える．

2.2.2　カーネルモニタシステム

カーネルモニタリングシステムは WWW サーバ計算機のディスク I/O や仮想メモリ，ネットワークの状況や CPU の利用状況などのシステムリソースを計測するシステムである．ほとんどの UNIX 系のシステムではこれらを測定するプログラムが標準で付属している．

例えば，vmstat コマンドにより仮想メモリの統計情報が分かり，netstat コマンドではアクティブソケットの一覧やトラヒックの状況といったネットワークに関する統計情報が得られる．また，iostat コマンドでは端末やデバイス，CPU 操作のカーネル I/O 統計が分かる．しかし，これらのシステムにより，サーバ計算機に余分な負荷を与える可能性がある．

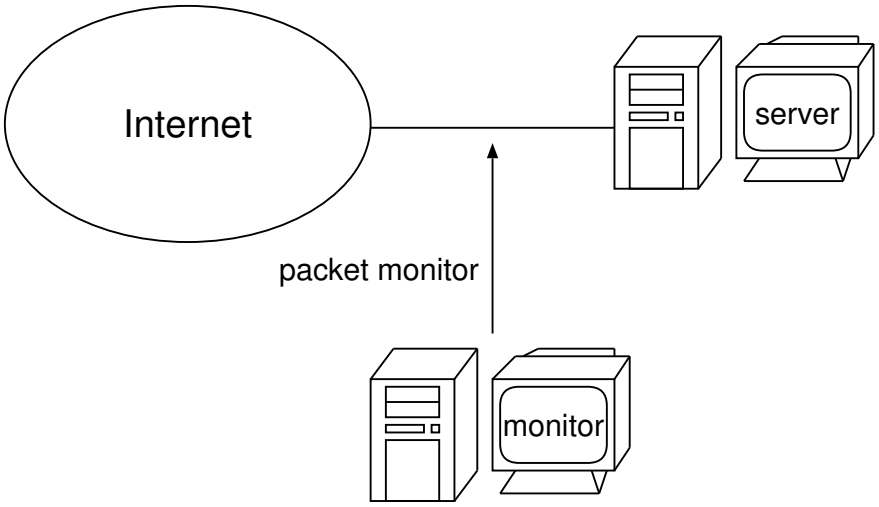

図 2.1 パケットモニタによる観測

2.2.3 パケットモニタシステム

図 2.1 にパケットモニタによる WWW サーバ観測の概念を示す．この手法による WWW サーバ観測の特徴を以下に挙げる．

- この手法は，WWW サーバを出入りするパケットを観測し，その結果から性能指標を導き出す．
- WWW サーバと同じネットワークに計測する計算機を必要とする．
- サーバ計算機に特別な仕掛けをする必要がなく，サーバ計算機に余分な負荷をかけることがない．

代表的なパケットモニタリングツールそしては tcpdump が挙げられる．tcpdump は多くの UNIX 系システムに付属している汎用パケットモニタツールである．このツールにより，ネットワーク上のトラヒックデータをダンプできる．tcpdump により得られたログを視覚化するツールも多数開発されている．しかし，tcpdump は全てのパケットを捕捉しているため，大規模な WWW サーバではパケットロスが起こる可能性があり，このようなサーバには適応できない場合がある．

それに対し，HTTP に特化したパケットモニタリングシステムとして ENMA [17] というツールがある．このツールは各 HTTP コネクション毎にログを作成しているのでリ

アルタイムでのログ出力のオーバーヘッドが小さい事が特徴である．

2.2.4　リアルタイム観測システムに適切な観測手法

本章では，サーバ管理の補助をするため，運営中のサーバをリアルタイムで監視してその状態を把握する事を目標としている．そこで本節では，リアルタイム観測システムに用いる観測手法として適しているのはどのシステムであるのかを考察する．

サーバ管理者は，ベンチマークシステムを用いて多くの性能指標を把握できる．しかし，運用中の WWW サーバに対してベンチマークを行うのは困難である．よって，リアルタイム観測システムには適応できないと言える．

カーネルモニタシステムも，多くの性能指標を導き出せる．しかし，サーバ計算機に余分なシステムを導入する事になるのでサーバが計算機の性能を完全に引き出すことができなくなる可能性がある．さらに，商用 OS ではカーネルに手を加える事が無理なものもある．よって，すべてのシステムでこの手法による観測が難しいため，リアルタイム観測システムには適応できないと言える．

パケットモニタシステムは外部から観測する手段であるので，サーバの規模や OS に依存しない．また，サーバ計算機に特別な仕掛けをする必要がない．サーバ計算機に余分な負荷をかける事がなく，運用中のものも観測可能である．

以上の事より，リアルタイムでのサーバ監視システムには，パケットモニタによる観測が適していると考えられる．我々は，WWW サーバに特化したパケットモニタリングシステムである ENMA を用いて監視システムを設計，実装する．

2.3.　WWW サーバの状態の定義

本節では，WWW サーバ運営の指標としての WWW サーバの状態の定義を行う．

まず，ユーザの視点から見た WWW サーバの状態を考える．リンクを選択したり URL を入力してもすぐに内容が表示されない場合，利用者は別のサーバのコンテンツを選択する場合がある．つまり，WWW サービスの利用者は，要求したコンテンツが即座に表示されることを望んでいると言える．

そこで，我々は利用者がどの程度ストレスなくコンテンツを受信できているかに着目してサーバの状態を定義する．

表 2.1 主な WWW ブラウザによる中止，リロードボタンを押した時に送られる信号

	中止ボタン	リロードボタン
Netscape Navigator	RSET	RSET
Internet Exproler	FIN	RSET

2.3.1 利用するパラメータ

ここでは，サーバの状態を表すのに用いるパラメータについて説明する．

利用者がストレスなくコンテンツを受信できているか，という点をサーバの視点からみれば，要求されたリクエストに対してどの程度コンテンツを配送できているか，という点に着目すればよいことになる．リクエストに対するコンテンツ配送の状況は，それぞれのリクエストの TCP コネクションの終了形態により明らかになる．そこで，我々はこのパラメータを用いてサーバの状態を把握する．ENMA では，各コネクションの終了形態を以下の 4 つに分類して記録している．

正常終了 クライアントからのリクエストに対してコンテンツを配送し終えた場合で，サーバからの “FIN” によりコネクションが終了したものである．

クライアントからの切断による終了 クライアントからの “FIN” や “RSET” によりコネクションが終了したものである．WWW ブラウザの「中止ボタン」を押したりコンテンツの配送中に「リロードボタン」を押すことにより発生する．主なブラウザの「中止ボタン」や「リロードボタン」を押した時にどちらの終了形態になるかを表 2.1 に示す．

サーバによるリセットによる終了 ポート番号の違いが原因で，サーバからの “RSET” により切断されたものである．

タイムアウトによる終了 リクエストを受信してから長時間にわたり継続されたコネクションをタイムアウトとしている．ENMA ではサーバがリクエストを受信してから 10 分間以上継続されたコネクションをタイムアウトとしている．

2.3.2 サーバの状態定義

我々は，サーバ状態として以下の 3 つを定義した．

正常状態

この状態は，全てのリクエストに対してコンテンツの配送が出来ている状態である．ここで，ユーザがコンテンツの配送中に他のリンクを選択したりブラウザの中止ボタンを押したことで配送が中断された場合は例外とする．サーバ管理者は，サーバを常にこの状態に保っておくのが理想である．

飽和状態

クライアントからのリクエストにサーバの処理が追い付いていない状態である．原因として，アクセスの集中などが挙げられる．この状態では一部のリクエストでコンテンツの配送が完全に終了できていない．

停止状態

飽和状態よりさらに状況が悪化しており，ほとんどのリクエストに対しコンテンツの配送が出来ていない状態である．また，サーバプログラムや計算機の再起動中など，サーバが動作していないときもこの状態とする．

2.3.1 節で挙げたパラメータとここで定義した状態の関係は次のようになると予想される．WWW サーバが正常状態の時はほぼ全てのクライアントで問題無くコンテンツが表示されているため，利用者はほとんどリロードボタンや停止ボタンを押さない．よって，ほぼすべてのコネクション終了状態は S_NORMAL になると予想される．

一方，飽和状態や停止状態といったサーバの状態が悪化している時には，ユーザは頻繁に中止ボタンやリロードボタンを押すといった行動から，クライアントからの切断が増えると予想される．

2.4.　観測値の解析による検証

本節では，実際に運営された WWW サーバの観測を行い，をの結果を解析し，前節で挙げたパラメータとサーバの状態の関係を明らかにする．

観測対象とした WWW サーバは，第 81 回全国高等学校野球選手権大会のインターネットライブ中継に用いられた WWW サーバである．この WWW サーバでは大会開始約一週間前から大会終了まで運営され，一日当たり数千万のアクセスが得られた．

2.4.1 システム及びネットワーク構成

観測対象としたWWWサーバには3台の計算機を用いた．そのうち2台がSun Enterprise450，残る1台がSun Enterprise250である．それらはレイヤ4スイッチにより接続されている．また，観測時にパケットの取りこぼしが発生するのを避けるため，観測に用いる計算機にはサーバと同等の計算機を用いている．表2.2にSun Enterprise450を用いたサーバとその計測に用いた計算機の性能と用いたアプリケーションを，図2.2に観測時の計算機構成を示す．WWWサーバプログラムには一般に良く用いられているApache[18]を用いた．

大会期間中に実際にWWWサーバにアクセスした状況から正常状態，飽和状態，停止状態を判断し，その時間帯の観測結果を解析した．

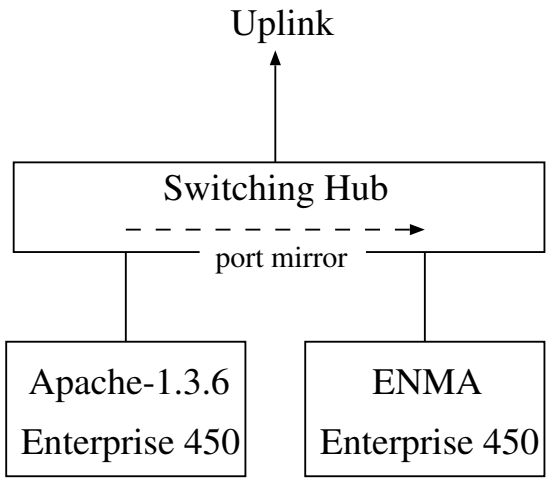

図 2.2 ネットワーク構成

2.4.2 解析手法

ここではログの解析手法について述べる．コネクション終了形態については1秒毎の割合を求めた．その結果，S_RESETの割合は全ての時間帯で0.1%以下であったため，この値は無視した．また，C_RESETとC_NORMALは共にクライアントからの切断を表わしているため，合算して計上した．本実験では，過去20分の割合から状態を判断した．

表 2.2 計算機構成

	WWW サーバ	測定用計算機
OS	SunOS 5.6	SunOS 5.6
CPU	Ultra Sparc II 300MHz × 2	Ultra Sparc II 300MHz × 2
メモリ	512M	256M
ソフトゥェア (version)	Apache (1.3.6)	ENMA (19990823)

2.4.3 解析結果

図 2.3 に正常状態での結果を示す．このグラフでは，X 軸 (左方向) が S_NORMAL，Y 軸 (右方向) が C_RESET と C_NORMAL の合計，Z 軸 (上方向) が TIMEOUT の割合をそれぞれ表わしている．すると常に $X + Y + Z = 100$ となるため，それぞれの点はこの三角形上にプロットされる．WWW サーバが正常状態の時はおおよそ 60〜70%のコネクションが正常終了であり，クライアントからの切断が 30〜40%，タイムアウトによるものは 10%以下であることが分かる．

図 2.4 は飽和状態での結果である．点は二つの領域に分布している．一つは，クライアントからの切断が多い場所，もう一つは TIMEOUT が 70%を越える範囲である．

サーバが停止状態にある時の結果を図 2.5 に示す．これは，サーバプログラムを再起動した前後の結果である．正常状態と同じ分布と，TIMEOUT の割合が多い二つの分布が見られる．

これらの結果より，サーバが正常状態にある時は S_NORMAL の割合が非常に高く，状態が悪くなると TIMEOUT やクライアントによる切断が増えると言える．

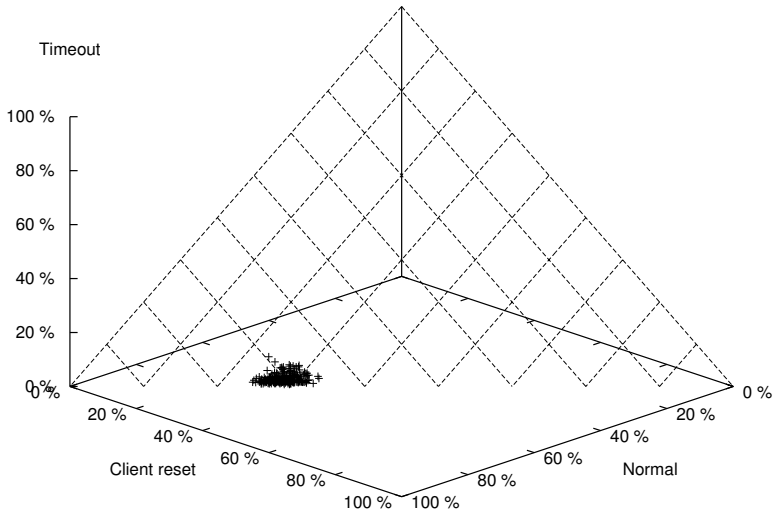

図 2.3 コネクション終了形態 (正常状態)

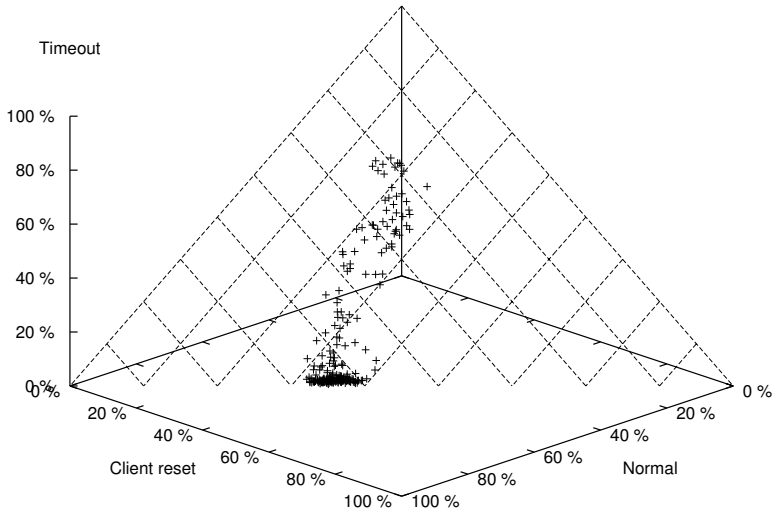

図 2.4 コネクション終了形態 (飽和状態)

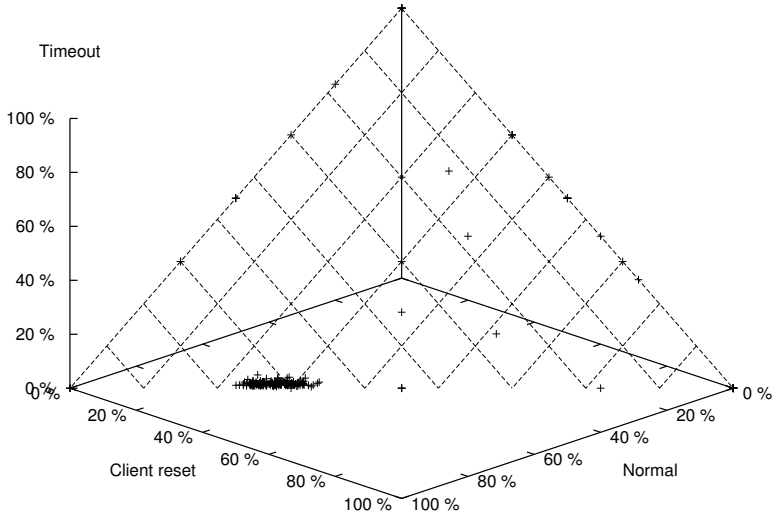

図 2.5　コネクション終了形態 (停止状態)

2.5. リアルタイム状態判別システム

前節の実験により，WWW サーバの状態を TCP コネクションの終了形態により表わせることが分かった．しかし，その実験はサーバ運営後のログ解析により得られた物である．実際の運営でこの指標を用いるには，サーバの運営中にリアルタイムで状態を把握する必要がある．そこで，リアルタイムでサーバの状態を把握するシステムを構築した．本節では，そのシステムについて述べる．

2.5.1 設計

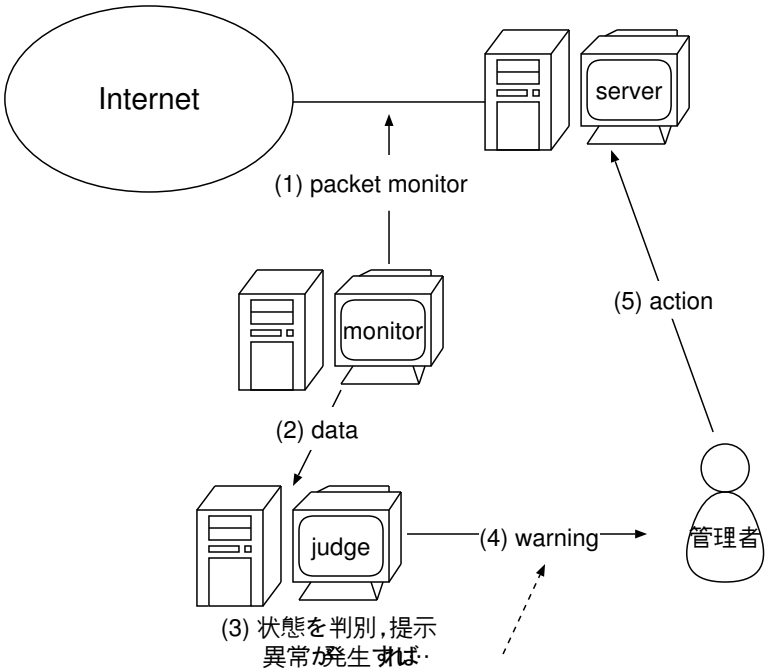

図 2.6 本システムによるサーバ管理の流れ

図 2.6 に本システムによるサーバ管理の流れを示す．

(1) 観測用計算機は WWW サーバと同セグメント上に位置し，観測システムにより常にパケットモニタを行う．

(2) 一定時間毎に各コネクションの継続時間や終了形態などの統計情報を判別用計算機に送信する．この通信はユーザには関係ない管理用トラヒックであり，WWW サー

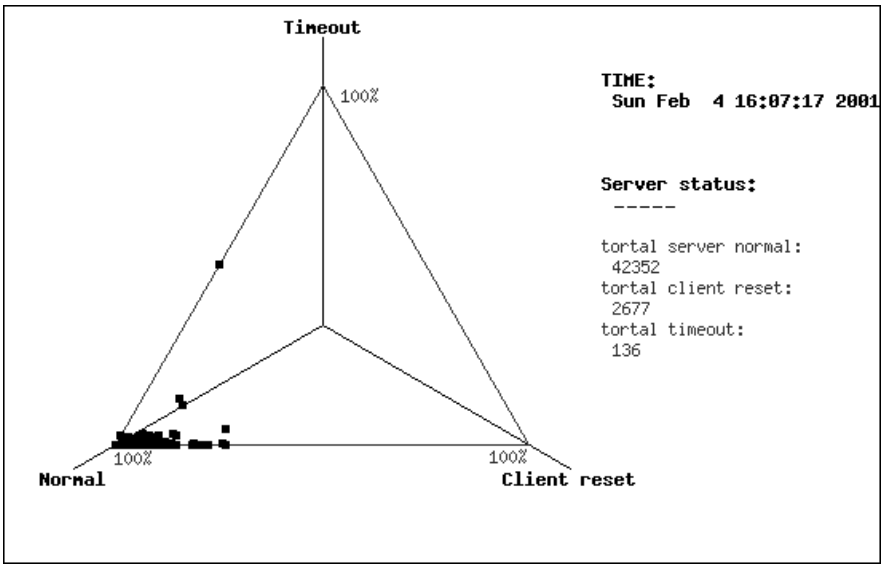

図 2.7 出力イメージの例

バが接続されているセグメント (ユーザ用セグメントと呼ぶ) には大量の HTTP トラヒックが流れている．そのため，ネットワークが混雑してきた時にこの通信が滞ることも考えられる．よって，ユーザ用セグメントとは別に管理用のセグメントを設け，そこでこの通信を行うのが望ましい．

(3) 判別用計算機は送信された統計情報をもとに WWW サーバの状態を判別し，ディスプレイに図 2.7 で示したような図を表示することで管理者に提示する．

(4) もし WWW サーバが飽和，停止状態に陥ったことが判明すれば正常状態の時とは別の方法で管理者に通知する．

(5) 通知を受けた管理者は WWW サーバに対して適切な処置を行う．

2.5.2 状態判別のための実験

サーバの状態を判別するための実験について述べる．ベンチマークにより HTTP リクエストをサーバに送信，その様子を構築したシステムにより観測した．アクセスするファイルのサイズは 4K バイトとし，ベンチマーク実行時にブラウザでも同じファイルを表示させ，その時の状況によりサーバの状態を判断した．停止状態については，サーバプロ

グラムを起動していない状態でリクエストの送信を行った.

実験環境

実験に用いた計算機のスペックおよび使用したアプリケーションを表 2.3 に，実験ネットワーク構成を図 2.8 に示す．各計算機のネットワークインタフェースは 100Base-TX であり，全二重通信が可能なハブにより接続されている．

本実験では WWW サーバに高い負荷を与えて飽和状態に陥らせる必要がある．そこでリクエスト生成には WWW サーバと同等のスペックの計算機を 2 台使用し，MAXUSER や SOMAXCON といったカーネルパラメータもよりチューニングされたものを使用している．それぞれの計算機の OS は FreeBSD-4.1.1-RELEASE，サーバプログラムには Apache-1.3.12 を，ベンチマークのクライアントには httperf-.06 を用いた．

表 2.3 実験に用いた計算機

	WWW サーバ	ベンチマーク	測定及び判定
OS	FreeBSD 4.1.1 RELEASE	FreeBSD 4.1.1 RELEASE	FreeBSD 3.4 RELEASE
CPU	Pentium III 700MHz	Pentium III 700MHz	Pentium II 450MHz
メモリ	256M	256M	256M
ソフトウェア (version)	Apache (1.3.12)	httperf (0.6)	ENMA(観測) j-graph(判定)

実験結果

リクエストレートを 600~800(req/sec) と変化させた時の結果を図 2.9 に示す．このとき，ブラウザによるアクセスでは問題なく表示できたため，正常状態と判断した．多くのコネクションが S_NORMAL で終了しており，TIMEOUT とクライアントからの切断は多くても 25%であると言える．

続いて，リクエストレートを 900~1200 と変化させた．このとき，ブラウザでのアクセス状況が悪くなり，すぐに表示されない状態であった．よって，このときはサーバは

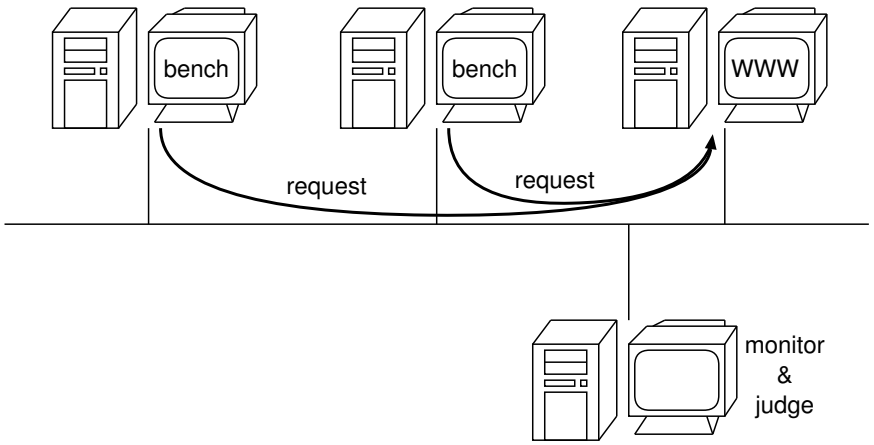

図 2.8　実験ネットワーク構成

飽和状態にあったと判断した．観測結果は図 2.10 である．クライアントからの切断と TIMEOUT の割合が非常に大きくなっていることが分かる．

さらに，正常状態から飽和状態への遷移を計測するため，リクエストレートを 600~1300 と変化させた．このときの観測結果が図 2.11 である．リクエストレートが低い時は左よりの，高い時は右よりの分布になっていた．これらの結果をまとめると，正常状態と飽和状態の分布は図 2.12 であると言える．

最後に，停止状態における分布を調べるため，WWW サーバプログラムを停止させてリクエストを送り観測した．しかし，グラフ上のどの位置にも点は表示されなかった．また，WWW サーバ計算機の電源を落したりネットワークケーブルを抜いた状態でリクエストを送っても同様の結果であった．

2.6.　まとめ

WWW はインターネットで最も重要なサービスである．安定したサービス提供のため，我々は運用の指針となる「WWW サーバの状態」を定義した．それは正常状態，飽和状態，停止状態の三つの状態で表わされる．この状態を表わすパラメータとして，各 TCP コネクションの終了形態に着目した．実際に運営されたサーバの観測結果より，その割合を計測することでサーバの状態を表わすことができることが分かった．

そこで，この割合をリアルタイムで計測し，運営中にサーバの状態を把握するシステ

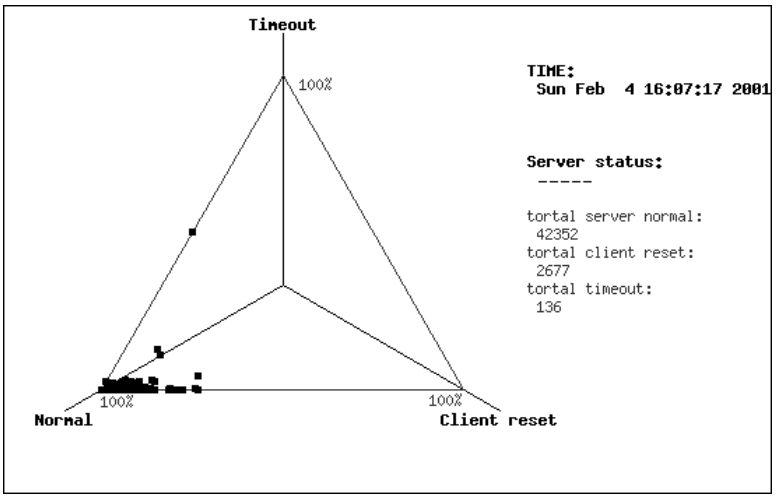

図 2.9 正常状態での分布

ムを構築した．そして，ベンチマークを用いた実験により，サーバの各状態と TCP コネクションの終了形態の割合の閾値を求めた．

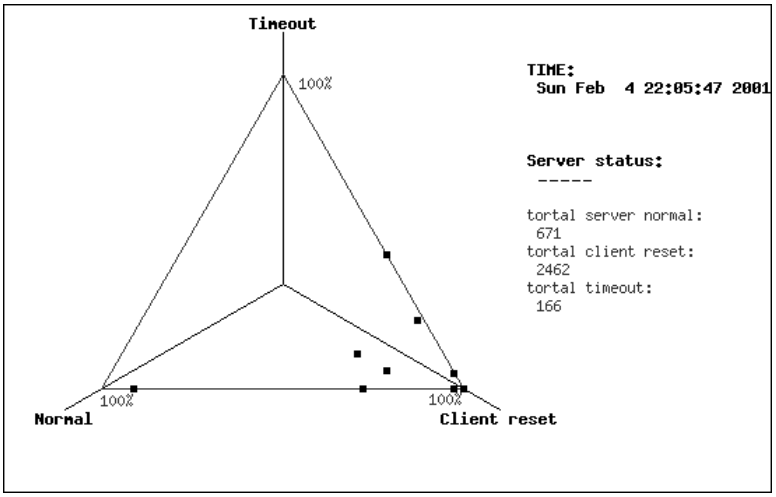

図 2.10　飽和状態の分布

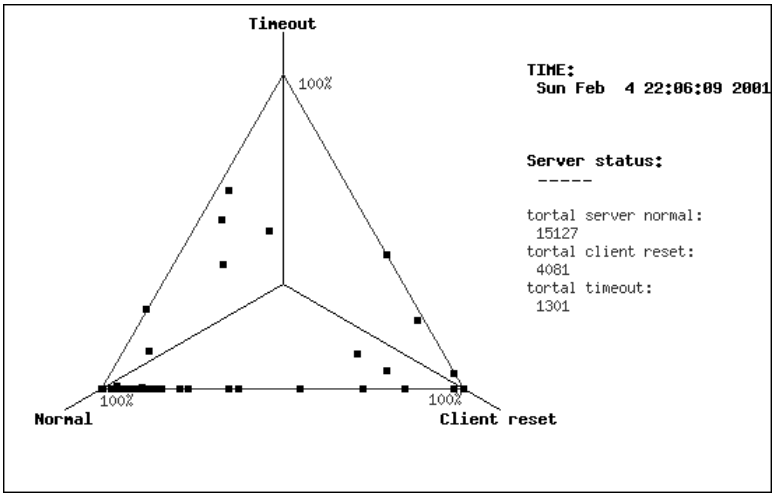

図 2.11　飽和，正常両状態の分布

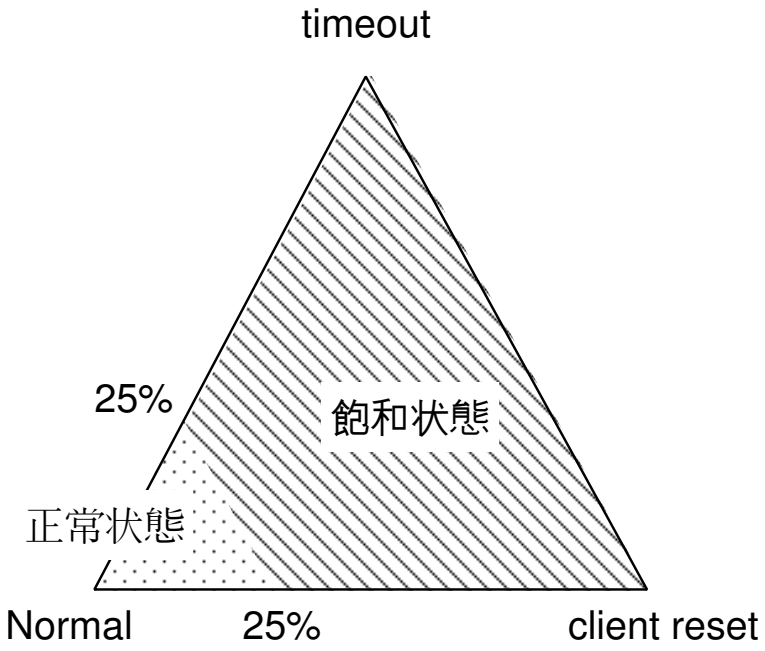

図 2.12 正常状態と飽和状態の分布

第 3 章　IPv6 環境を考慮した WWW ベンチマークシステム

現在，次世代 IP である IPv6 の普及がすすんでおり，様々なサービスが IPv6 に対応するようになってきている．そのため，サーバやネットワークを管理してきたネットワーク管理者も IPv6 環境下でサーバやネットワークを観測，監視する必要が出てきた．我々は，インターネットにおける最も主要なサービスである WWW に着目し，そのサーバを IPv6 環境下で観測，監視できるシステムについて考える．従来おこなわれて来た WWW サーバの観測手法であるカーネルモニタ，パケットモニタ，ベンチマークのうち，ベンチマークだけが IPv6 環境に対応していない．そこで，IPv6 と IPv4 の割合を自由に変化できる新たなベンチマークを提案，実装する．そして，様々な条件でベンチマークを行い，IPv6 と IPv4 のリクエストによるサーバの挙動の違いを明らかにする．

3.1.　まえがき

インターネットは急速に広まり，今日では日常生活に無くてはならない情報基盤として確立している．しかし，現在インターネットで用いられているインターネットプロトコルバージョン 4 (IPv4) では，数年後にアドレス空間の枯渇するという予測がある [19]．また，商業目的のインターネット利用の増加からセキュリティ機能の強化が求められてきている．

これらの問題を解決する次世代の IP プロトコルとして，インターネットプロトコルバージョン 6 (IPv6) が開発され，一部の ISP では IPv6 の接続サービスも始まっている．多くのネットワークアプリケーションも IPv6 を扱えるようになってきており，メールや WWW といった主要なサービスを提供することも可能となってきた．

一方，ネットワーク管理者は彼らの管理するサーバおよびネットワークを観測，監視

してきた．しかし，IPv6 の広まりに伴い，IPv6 環境化でも従来と同様の観測を行う必要が出てきている．特に現在は，IPv4 から IPv6 への移行の過渡期であるため，両方の環境でサーバやネットワークの観測，監視を行えるのが望ましいと考えられる．

そこで，我々はインターネットで最も主要なサービスである World Wide Web (WWW) に着目し，そのサーバを IPv6 環境下で観測，監視するシステムを考える．WWW サーバの観測手法は，大きく分けてパッシブ計測とアクティブ計測に分類される．パッシブ計測とは計測系から計測対象に直接働きかけずに観測し，その結果から性能指標を得る観測手法である．カーネルモニタリングシステムやパケットモニタリングシステムがパッシブ計測の代表として挙げられる．

カーネルモニタリングシステムは，サーバ内部のディスク I/O やネットワークの状況，システムリソースをモニタし，その結果を報告することでサーバの負荷状況を調査するシステムである．ほとんどの UNIX 系 OS では，これらを測定するプログラムが標準で付属している．例えば，*vmstat* コマンドによりプロセス，仮想メモリ，ディスク，トラップ，CPU の動作状況についてのカーネルが持っている統計情報を取得でき，*netstat* コマンドではアクティブなソケットの一覧やトラフィックの状況を得られる．*iostat* コマンドにより端末や CPU の I/O 状況を把握でき，*fstat* コマンドでオープンされているファイルについての情報を把握できる．

パケットモニタシステムはサーバが送受信するパケットを外部から補足することによる観測手法であり，以下に挙げる利点から，大規模なサーバの観測，監視において特に有効とされている．

- サーバ自身の性能劣化を引き起こさない．
- サーバの OS やサーバプログラムに依存しない．

パケットモニタシステムの代表的なものとして *tcpdump* プログラムが挙げられる．

一方，アクティブ計測とは計測対象に対してテストデータを送信し，その結果から性能指標を得る計測手法であり，ベンチマークシステムがこれにあたる．

Httperf [14] や SPECweb [15]，Webstone [16] に代表されるベンチマークシステムは擬似的にリクエストを発生させ，その応答状況から処理レートやスループットなどの WWW サーバの様々な性能指標 (詳細は 3.2.3 節で述べる) を明らかにできる．

WWW サーバの性能解析やチューニング指針を得る場合は，これらの手法を単体で用いるのではなく，組み合わせて用いる場合が多い．そこで，上記の手法が IPv6 を扱えるかをまとめると，次のようになる．

カーネルモニタシステムによって得られる情報はネットワークに関するものではないため，問題なく利用できる．パケットモニタシステムの代表的なものである *tcpdump* はIPv6 パケットを観測できる．我々の研究グループで開発している WWW に特化したパケットモニタシステム [17] も一部 IPv6 に対応している．しかし，代表的な WWW ベンチマークシステムについては，IPv6 を扱うことができない．

つまり，IPv6 環境での WWW サーバ性能解析を行うには，ベンチマークシステムが必要だと言える．

そこで本章では，IPv6，IPv4 両方のプロトコルスタックでリクエストを送信できる新たなベンチマークシステムの実装を行う．そして，実験環境下でサーバシステムのベンチマークを行い，その有用性を示す．

3.2. システムの概要

本節では，本章で提案するベンチマークシステム (ANMA) の概要を述べる．このベンチマーク設計の際に考慮した点を以下に挙げる．

- IPv6 と IPv4 両方のプロトコルスタックでリクエストの送信ができること．
- ベンチマークの結果がプロトコルスタック別に出力できること．

3.2.1 リクエストレートの制御

WWW ベンチマークにおいては，リクエストレートの制御が重要となる．リクエストレートの制御方法には主に次の二種類が考えられる．

- 一定時間 (多くの場合一秒) あたりに送信するリクエスト数を制御する．
- サーバが処理中のコネクション数が常に一定であるように制御する．

前者の手法では，WWW サーバのレスポンス状況に関係なくリクエストの送信を続ける．そのため，瞬間的に WWW サーバにリクエストが集中するような状況のシュミレートが可能となる．一方，後者の手法では，WWW サーバは常に一定数のコネクションを処理することになるため，定常的に負荷をかけることができる．しかし，意図的に急激に負荷を高めるようなリクエストの送信は難しい．

提案するシステムでは，次に挙げる理由からコネクション数を一定に保つ後者のレート制御手法を用いた．

- 時間あたりのレート制御は，タイマ割り込みのシグナルによる実装が考えられる．また，各コネクションの読みだし，書き込みに関しても非同期 I/O で実装する必要があるため，ここでもシグナルによる割り込み制御が発生する．そのため，時間あたりに送信するリクエスト数が増えると，タイマ制御，シグナル制御の回数が非常に多くなる．そして，シグナル処理のオーバーヘッドにより，正しくレート制御が行なわれなかったり，送信できるリクエスト数が減少する可能性がある．
- サーバへの負荷を急激に高めるような場合は，複数の計算機によるベンチマークを行う．そして，それぞれのコネクション数を変化させたり，時間差をつけて送信を開始することで対応できる．

3.2.2　デュアルスタックのリクエスト送信

提案手法では，IPv6 と IPv4 両方のプロトコルスタックでのリクエスト送信を目標としている．また，IPv6，IPv4 のデュアルスタックでサーバが運用されることを想定した場合，送信する全リクエスト中の IPv6 と IPv4 の割合を自由に指定できるようになければならない．

通常の WWW ベンチマークシステムでは，送信元の IP アドレスは一定である．しかし，我々の目標である IPv6 と IPv4 のリクエスト割合を設定可能にするためには，プログラム実行時に意図した順序で送信元の IP アドレスを変化させながらソケットを作製する必要がある．一つのプログラムから複数の送信元アドレスでソケットを利用するには `bind()` システムコールでソケットにアドレスを割り当てる．その際にはアドレスファミリを指定する必要があるため，送信元アドレスから `getaddrinfo()` などの関数でアドレスファミリを判断しなくてはならない．

そこで，ソースアドレスとして用いる IP アドレスのリストを作製し，そこからソケット作製毎にソースアドレスを読みだす手法を用いた．この手法により，ソケット作製の度に意図したソースアドレス，アドレスファミリの指定が可能となる．また，アドレスリストに含まれる IPv6 アドレスと IPv4 アドレスの割合がリクエストの IPv6 と IPv4 の割合となり，アドレスファミリに関するレート制御が可能となる (図 3.1)．例えばリスト中の IPv6 アドレスと IPv4 アドレスの割合が 8 : 2 であれば，IPv6 によるリクエストが

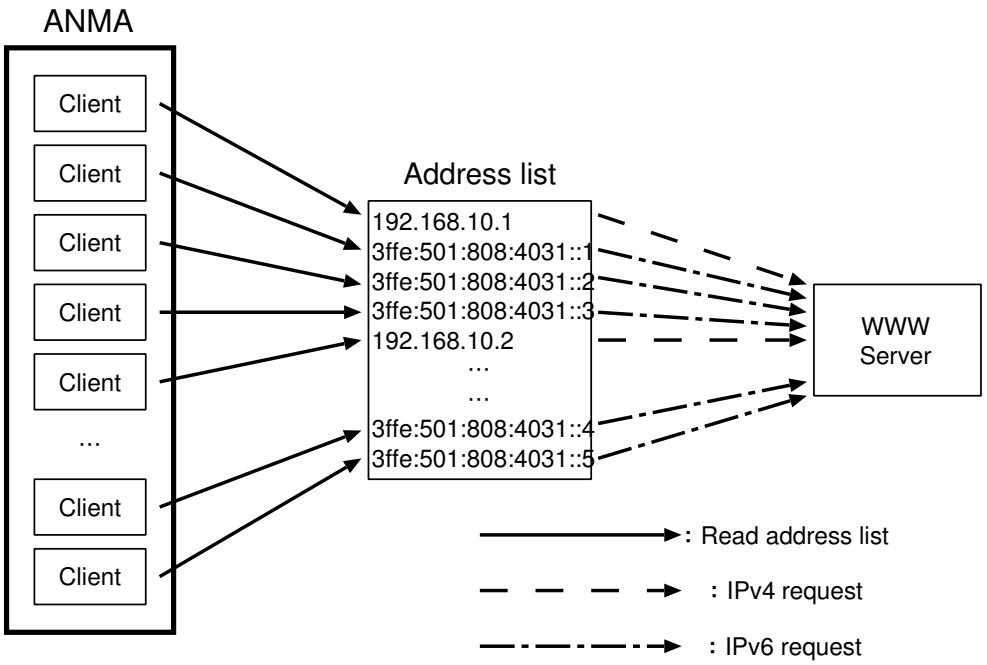

図 3.1 アドレスリストを用いたリクエスト送信

80%，IPv4 によるリクエストが 20%となる．

3.2.3 WWW サーバの性能指標と ANMA が記録する情報

本節では，代表的な WWW サーバの性能指標として用いられる指標 [16][20][21] について述べる．ANMA では，これらの指標を求めることができる．

リクエスト処理レート

WWW サーバでは，一定時間内に処理可能なリクエスト数には限界がある．その上限を越えるリクエストが発生した場合，処理効率が極端に低下し，処理が停止する場合もある．そこで，一定時間内に処理したリクエストの数が性能指標となる．ピーク時の性能が要視されることが多いため，秒間あたりの処理リクエスト数が重要であるとされている．

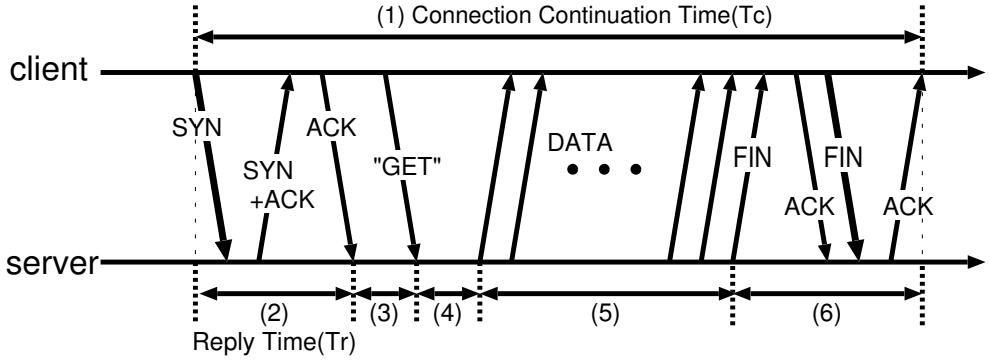

図 3.2　HTTP のパケットシーケンス

スループット

スループットは秒間あたりの転送バイト数で表される．この指標はコンテンツのサイズや処理したリクエスト数に依存する．また，ネットワークの状況を表す指標としても用いられる．

コネクション継続時間 (Tc)

コネクション継続時間 (以下 Tc とする) は図 3.2 に示した HTTP のパケットシーケンスにおける (1) の時間にあたり，TCP/IP 接続に要した時間 (2)，クライアントからリクエスト送信時間 (3)，サーバでのリクエスト処理 (4)，コンテンツ送信 (5)，コネクション切断処理 (6) の全てを含んだ HTTP における通信全体の長さである．

Tc はサーバ計算機の OS やサーバプログラム自身の他，コンテンツサイズやサーバ・クライアント間のネットワーク的距離やその時のネットワーク状況など，さまざまな要因が影響する．また，TCP/IP 接続に要した (2) の時間も応答時間 (以下 Tr とする) として WWW サーバの性能指標として用いられる．

各コネクション終了形態

リクエスト処理レートの項目で述べたように，WWW サーバへのリクエストが集中し，負荷が高くなると処理が滞る．このとき，送信したリクエストは処理されずに切断されたり，長時間処理されなくなったりする．このことを利用し，各コネクションの終了状

態の統計により現在の WWW サーバの状態を知ることができる [22][23].

3.3. 実装

本章では，ANMA の実装について説明する．我々は，様々な UNIX プラットフォームで動作するシステムを目標として本システムを実装した．そのため，C 言語による実装を行った．開発および実行環境は FreeBSD 4.6,4.7–STABLE および NetBSD–1.6 である．

3.3.1 リクエスト送信の手順

3.2.1 節で述べたように，ANMA では，あらかじめ同時に送信するリクエスト数の上限を定めることで，一定数のコネクションを保持する．

図 3.3 にリクエスト送信のフローチャートを示す.

1 仮想クライアントの IP アドレスを決定するため，アドレスリストから IP アドレスを読み込む.

2 `getaddrinfo()` 関数により，読み込まれたアドレスが IPv6 か IPv4 かを判断する．`getaddrinfo()` は与えられた文字列の IP アドレスに対して *sockaddr* 構造体を返す.

3 `socket()` システムコールにより新たなソケットを作製する．このとき，2. で得られた *addrinfo* 構造体の `ai_family` メンバにより IPv6，IPv4 どちらのアドレスファミリのコネクションかを決定する．作製したコネクションを `bind()` システムコールでアドレスを割り当てる．また，多数のコネクションを同時に扱うため，ソケットは `fcntl()` システムコールを用いて非同期 I/O とする.

4 `connect()` システムコールにより，サーバにコネクション確立要求を送る.

5 コネクションの確立を `select()` システムコールにより検出する.

6 `write()` システムコールにより，リクエストを送信する.

7 リクエストに対するサーバのレスポンスを `select()` で検出する.

8 コネクションが読みだし可能になった場合，`read()` システムコールによりサーバからのレスポンスを受信する.

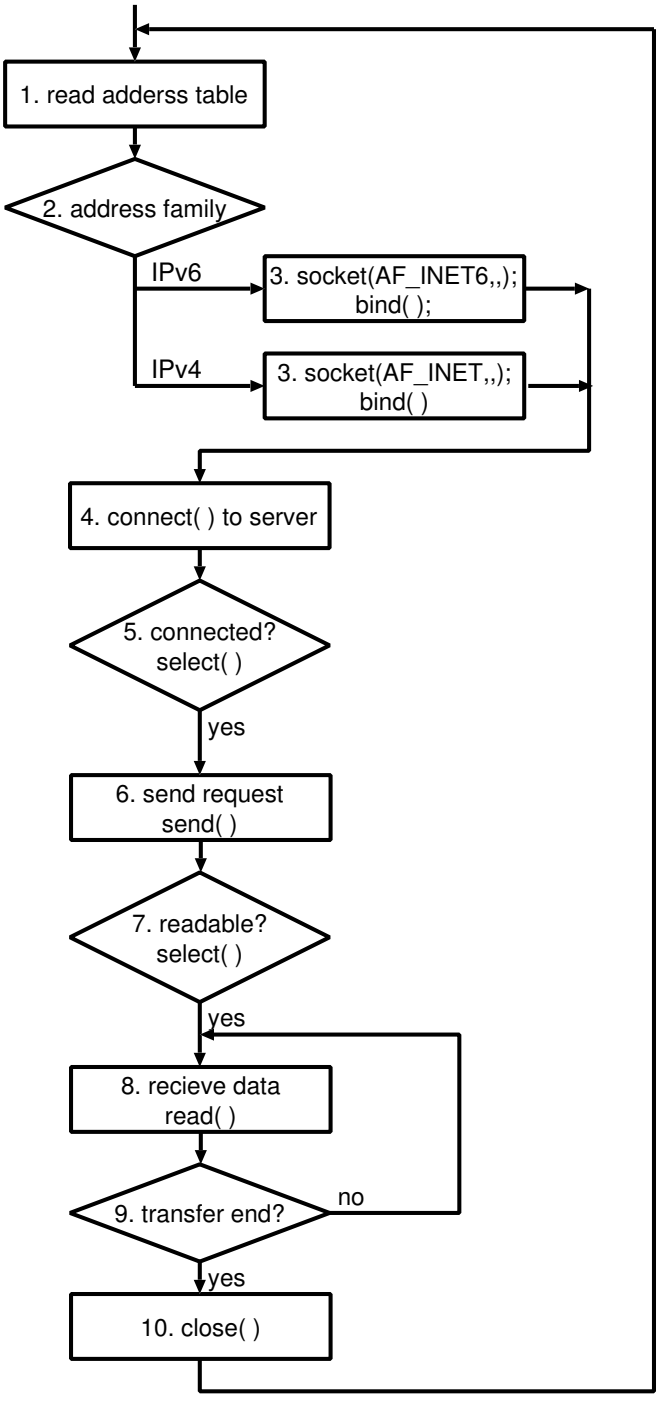

図 3.3　リクエスト送信の流れ

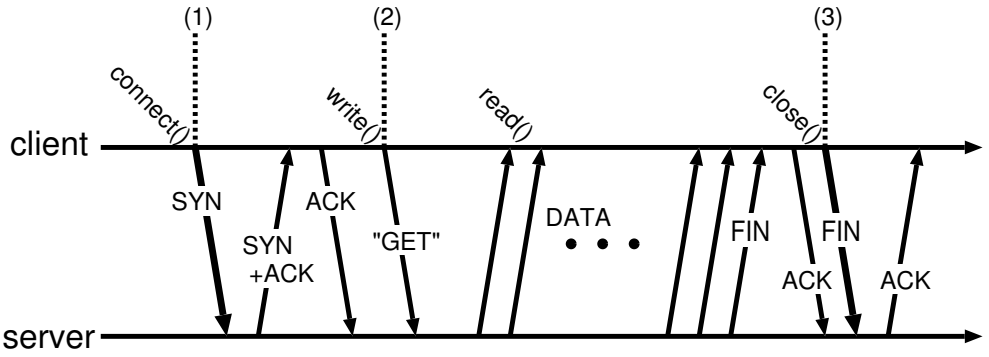

図 3.4 ANMA が記録する時間

9 サーバがコネクションをクローズするまでレスポンスを受信する.

10 サーバがコネクションをクローズした場合，`close()` システムコールによりソケットを閉じる.

3.3.2 記録する時間

3.2.3 節で述べた Tc の記録のため，ANMA では，コネクション毎に以下のタイミングで時間を記録する (図 3.4).

1 `connect()` システムコールが呼ばれた時間.

2 `write()` システムコールにより，リクエストが送信された時間.

3 コネクション切断のため `close()` システムコールが呼ばれた時間.

(1)∼(3) の時間を計算することで Tc が算出できる．また，Tr は (1)∼(2) の時間を算出すればよい．これらの時間の記録には `gettimeofday()` システムコールを用いた．`gettimeofday()` はマイクロ秒までの値を記録できる．しかし関数自体のオーバーヘッドがあるため，実験環境ごとに精度をあらかじめ確認する必要がある.

3.3.3 コネクション情報の記録

3.2.3 節で述べたコネクション情報の記録は次のようにする.

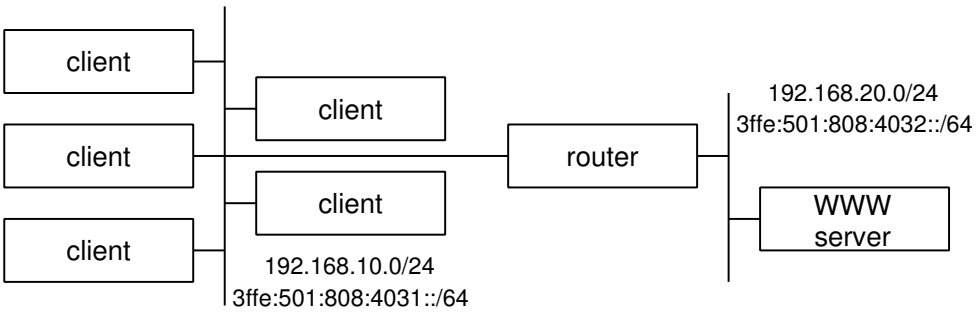

図 3.5　実験ネットワーク

各コネクションがどのような形で終了したかを次の 3 つに分けて記録する．それによりベンチマーク時にサーバがどのような状態にあったかを判断できる．サーバの負荷がそれほど高くなかった場合は多くのコネクションは NORMAL で終了するが，RESET や TIMEOUT の割合が高かった場合，サーバが高負荷だったことが分かる．

NORMAL サーバからリクエストしたコンテンツを正常に受信した場合．

RESET なんらかの理由でサーバからコネクションが切断された場合．

TIMEOUT リクエスト送信後一定時間経過してもサーバから応答がない場合．ここでは 5 秒とした．

3.4.　実証実験

ANMA の有効性を検証するため，いくつかの条件下でベンチマーク実験を行った．

3.4.1　実験環境

実験ネットワークのトポロジを図 3.5 に，用いた計算機のスペックとソフトウェアを表 3.1 に示す．3.2.1 節で述べたように，サーバへの負荷を変化させることができるように，クライアントには 5 台の計算機を用いた．すべての計算機の OS は FreeBSD 4.7–STABLE である．

本実験では，クライアント，サーバともに多数のコネクションを同時に扱う必要がある．そのため，カーネルパラメータの MAXUSER の値を 1024 とし，最大ソケット

表 3.1 実験に用いた計算機とソフトウェア

ホスト	OS	CPU	メモリ	ソフトウェア
クライアント	FreeBSD–4.7–STABLE (2002/11/09)	Pentium–III 800MHz	512M	anma–1.0
中間ルータ	FreeBSD–4.7–STABLE (2002/11/09)	Pentium–III 1GHz (Dual)	512M	—
サーバ	FreeBSD–4.7–STABLE (2002/11/09)	Pentium–III 800MHz	512M	Apache–1.3.27 + ipv6 patch

数 (kern.ipc.maxsockets) を増加させた．

サーバプログラムにはWWWサーバとして最もよく利用されている apache [18] のバージョン 1.3.27 に ipv6 unofficial patch[1]を適応した．apache の設定はすべてデフォルトとした．クライアントには，提案システムである ANMA を用いた．なお，ネットワークは全てギガビットイーサネットである．

また，3.3.2 節で述べたように `gettimeofday()` の精度を調べる必要がある．本実験では，Pentium の CPU クロックカウントを読み込むアセンブラ命令を用いてあらかじめ `gettimeofday()` のオーバーヘッドを計測した．その結果，この実験環境では 10 マイクロ秒の精度があると確認した．

3.4.2 実験 1

リクエスト送信の検証

ANMA により，IPv6 と IPv4 のリクエストが指定した割合で正しく送信できているかの確認を行った．

要求するファイルのサイズは 4 キロバイトとし，同時コネクション数を 1000 に，総リクエスト数を 10000 と固定した．そして，アドレスリストの内容を A∼E の 5 パターン用意し (表 3.2)，1 台の計算機からベンチマークを行った．

[1]Available: `http://motoyuki.bsdclub.org/data/IPv6/`

表 3.2　実験パラメータ

	リスト中の数		同時	総
	IPv6	IPv4	コネクション数	リクエスト数
A	5	5	1000	10000
B	8	2	1000	10000
C	2	8	1000	10000
D	10	0	1000	10000
E	0	10	1000	10000

表 3.3　コネクションデータの結果

	送信リクエスト数		NORMAL		RESET	
	IPv6	IPv4	IPv6	IPv4	IPv6	IPv4
A	5000	5000	5000	5000	0	0
B	8000	2000	8000	2000	0	0
C	2001	7999	2001	7999	0	0
D	10000	0	10000	0	0	0
E	0	10000	0	10000	0	0

結果

表 3.3 にコネクションデータに関する結果を示す．A∼E どのパターンにおいてもアドレスリストで指定した割合でリクエストが送信されていることが分かる．これについては，ANMA の出力だけでなく，WWW サーバのアクセスログも参照して確認した．また，どの場合も全てのリクエストが NORMAL で終了しており，サーバが正常に動作していたことが分かる．

表 3.4 に Tc と Tr に関する結果を示す．Tc の平均は IPv4 が 1.9 ミリ秒，IPv6 が約 2 ミリ秒であり，若干 IPv6 のほうが時間がかかっていることが分かる．一方，Tr の平均については，IPv4 が 0.5 マイクロ秒程度，IPv6 に関しては 1 マイクロ秒である．しかし，IPv4 のみのリクエスト送信の場合では Tr は 1.5 マイクロ秒と，IPv6 より時間がかかっている場合もある．

表 3.4 応答時間に関する結果

	リクエスト処理	平均 Tr (ms)		平均 Tc (ms)	
	レート (req/sec)	IPv6	IPv4	IPv6	IPv4
A	394.5	0.0008	0.0004	2.0054	1.9822
B	395.9	0.0008	0.0005	2.0039	1.9940
C	395.8	0.0015	0.0006	2.0010	1.9668
D	393.1	0.0012	—	1.9350	—
E	417.6	—	0.0015	—	1.8486

しかし，3.4.1 節で述べたように，本実験環境における `gettimeofday()` の精度は 10 マイクロ秒であり，Tr に関しては測定誤差の可能性が大きい．

リクエスト処理レートについては，どの場合でも，ほぼ同じ値となっている．これらの値は，IPv6 のみ (IPv4 のみ) のリクエスト送信時でもほぼ同じ値である．よってこの時間の違いは，リクエストを混ぜるのが原因ではないといえる．

3.4.3 実験 2

ファイスサイズや同時コネクション数の影響

前節の実験により，IPv6 によるリクエストが IPv4 によるものより Tc や Tr が長くなっていることが分かった．そこで，要求するファイルサイズや同時コネクション数を変化することによるこれらの値の変化を検証する．

アドレスリストは実験 1 と同様の 5 種類を用い，それぞれについて，要求するファイルサイズを 1，4，8，16，32 キロバイトの 5 種類，同時コネクション数を 1000，2000，5000 の 3 種類に変化させた．

結果

図 3.6 に，Tc の平均についての結果を示す．いずれの場合も，IPv4 より IPv6 のほうが長くなっていることが分かる．最も時間の短かったファイルサイズが 1 キロバイト，同時コネクション数が 1000 の時，IPv6 の平均は 1.025 ミリ秒であり，IPv4 の 0.998 ミリ

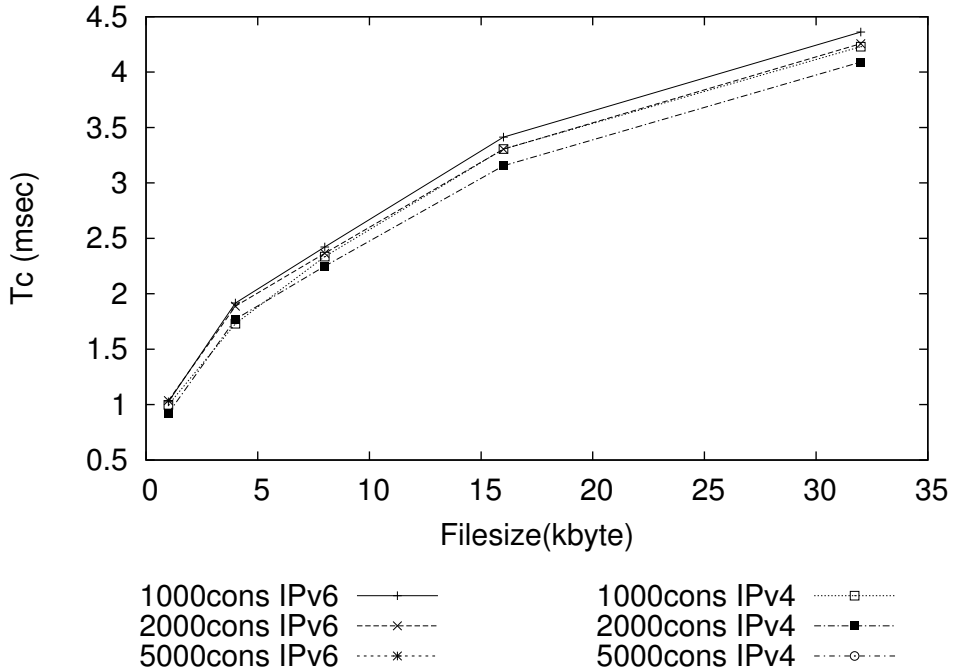

図 3.6　実験 2 の結果

秒と比べ約 0.03 ミリ秒長くなっている．また，最も時間が長かったファイルサイズ 32 キロバイト，同時コネクション数 5000 の場合は IPv6，IPv4 それぞれ 4.342 ミリ秒，4.142 ミリ秒となり，その差は 0.2 ミリ秒である．

要求するファイルサイズ数を増加させるにつれて *Tr* そのものと IPv6，IPv4 の差は大きくなることが分かる．しかし，同時コネクション数の増加に関しては *Tr* にはあまり影響が出ていない．

なお，コネクションデータに関しては，前節と同様，全て NORMAL となっていた．

3.4.4　実験 3

複数クライアントによるベンチマーク

実験 1 および 2 では，クライアントを 1 台だけ用いてベンチマークを行った．しかし，全てのコネクションが NORMAL で終了し，RESET で終了するものはなかった．また，ファイルサイズを増加させると RTT が長くなったが，コネクション数の増加は影響しな

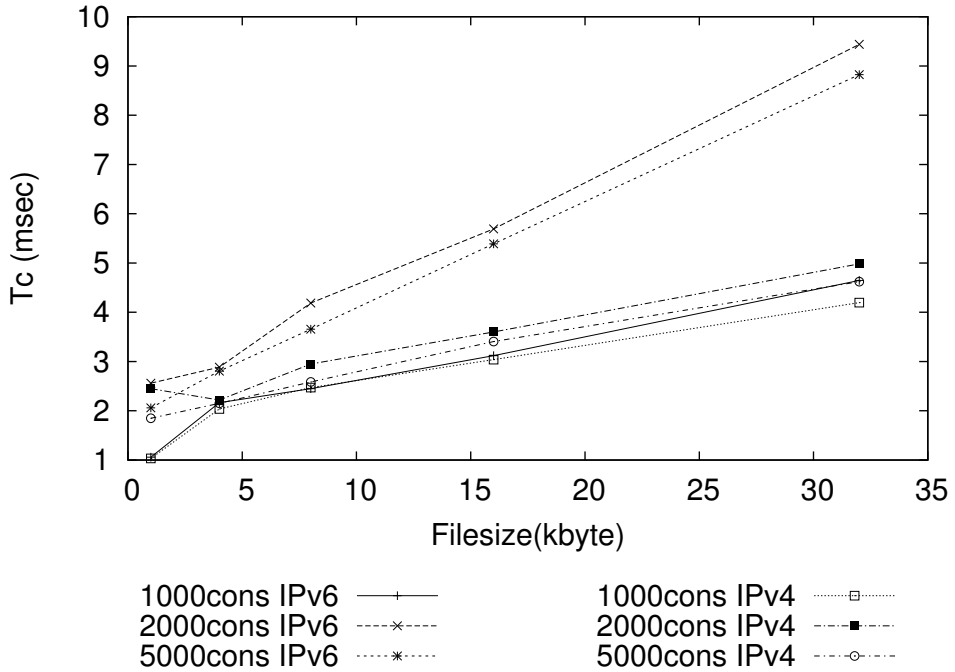

図 3.7 実験 3 の結果

かった．それは，クライアントの性能限界により，サーバに負荷をあまりかけることができなかったためと考えられる．

そこで，5 台のクライアントから同時にリクエストを送信することにより，サーバにかける負荷を高める実験を行った．各クライアントからはドキュメントサイズを 1，4，8，16，32 キロバイトに，同時コネクション数を 200，400，1000 として (5 台合計で 1000，2000，5000 となる) 送信した．

結果

表 3.7 に Tc の結果を示す．いずれの場合も実験 1，2 と同様で，同じ条件下では IPv4 より IPv6 のほうが時間がかかっている．また，要求ファイルサイズが大きくなるにつれて，その差の割合が大きくなっている．例えば，同時コネクション数 200，ファイルサイズ 1 キロバイトの場合では IPv4 は IPv6 の約 95%の時間であったが，ファイルサイズが 8 キロバイトの場合では 70%，32 キロバイトの場合は 52%の時間で転送を終えている．

表 3.5　実験 4 の結果 (Mbps)

	1k	4k	8k	16k	32k
IPv6	160.07	161.68	162.23	162.53	163.46
IPv4	166.86	168.02	168.40	168.68	168.87

同時コネクション数を増やした場合でも同様の割合となっている．

次に，コネクション終了形態について述べる．実験 2 では，すべての場合で RESET で終了するコネクションは観測されなかった．しかし，実験 3 では同時コネクション数が 200 (5 台で 1000) の場合は RESET は観測されなかったが，400 の場合で約 10%が，1000 の場合で約 20%が RESET で終了していた．この結果は実験 2 よりサーバの負荷が高くなっていたことを示している．また，本実装では同時コネクション数を増やしてサーバに高負荷をかける際には，複数のクライアントマシンを用いて同時にリクエスト送信をしたほうがよいことを示している．

3.4.5　実験 4

実験 1~3 の結果より，IPv6 によるリクエストが IPv4 のリクエストより時間がかかっていると分かった．そこで，この差の原因がプロトコルスタックによるものか，サーバプログラムの実装であるのかを明らかにするため，netperf[24] による計測を行なった．netperf ではネットワーク層レベルのスループットが計測できるため，IPv6 と IPv4 の観測結果の差が OS のプロトコルスタックの実装差と判断できる．

WWW サーバとして使用した計算機で netperf のサーバ (netserver) を動作させ，クライアントとして使用した計算機から netperf を実行した．実行時の条件はソケットの送信バッファ，受信バッファともに FreeBSD の標準サイズである 16 キロバイト，送信ファイルサイズは実験 2，3 と同様の 1，4，8，16，32 キロバイトとした．

結果

表 3.5 に結果を示す．実験 3 ではファイルサイズや同時コネクション数を増やし，WWW サーバの負荷が高まるに伴い，IPv4 と IPv6 の速度差が大きくなっていた．しかし実験 4 の結果では，いずれの場合も IPv4 のほうが若干スループットが高くなっているが，そ

の差は最大でも 5Mbps 程度であり，4%の差でしかない．

つまり，IPv4 と IPv6 の速度差はプロトコルスタックの部分より WWW サーバプログラム内の差が大きいと判断できる．

3.5. むすび

IPv6 への移行に伴い，IPv6 環境下におけるサーバやネットワークの観測，監視システムが必要となってきている．我々は，インターネットで最も主要なサービスである WWW に着目し，そのサーバを IPv6 環境下で観測，監視する手法について検討した．その結果，従来から用いられている WWW サーバ観測手法であるカーネルモニタ，パケットモニタ，ベンチマークのうち，ベンチマークについては実用になるものは見当たらなかった．ネットワークの測定に関しては netperf が IPv6 を扱える．しかし，netperf ではネットワーク層レベルのスループットしか計測できないため，WWW サーバの性能指標である Tc などは計測できない．

そこで，本章では IPv6 も扱える新たなベンチマーク ANMA を提案，実装した．ANMA により，IPv6 と IPv4 両方のプロトコルスタックでリクエストの送信ができる．そして，いくつかの条件でベンチマークを行い，その有効性を示した．また，netperf によるスループット計測の結果を組み合わせることにより，IPv6 と IPv4 の速度差は WWW サーバプログラム内部の差であると確認できた．

ANMA を用いることで，IPv6 による WWW ベンチマークが可能となった．パケットモニタやカーネルモニタによるサーバ側の情報やネットワークの状況と組み合わせることで，IPv6 環境下でも今までと同様の性能解析やチューニング指針を得ることができると期待できる．

ANMA における今後の課題としては，WWW 上で個人情報を送受信する際によく用いられる暗号通信手法である SSL によるリクエストを送信できるようにすることや，リクエスト対象のコンテンツを様々に変化できるようにすることが挙げられる．また，アドレスリストの自動生成も必要と思われる．

一方，WWW サーバ性能解析全般にわたる今後の課題として，IPv6 でのネットワーク遅延の実現がある．実験環境をより現実の環境に近づけるためネットワーク遅延を考慮する必要がある．しかし，ネットワーク遅延を実現する際に最も良く用いられる Dummynet [25] は IPv6 に対応していないためである．IPv6 環境でのネットワーク遅延実現は WWW

サーバのベンチマークだけではなく，様々なシミュレートを行う際に必要となると考えられるため，今後の重要な課題である．

第 4章　SSLや複数ファイルへのアクセスを可能とするベンチマークシステム

WWW が登場してから現在まで，扱われるコンテンツはさまざまに変化している．そのため，WWW ベンチマークシステムもその変化に対応する必要がある．

まず，従来の WWW ベンチマークでは，ベンチマークの対象となるファイルは一つだけであった．しかし，実際の WWW サーバでは一つだけのファイルにアクセスされるということは現実的ではない．そのため，さまざまなファイルへアクセスできる機能が必要となる．次に，WWW を利用したオンラインショッピングやユーザ登録などでは，SSL による暗号化を用いることが多い．よって，SSL にを用いた場合と用いなかった場合のサーバの挙動の変化の調査は重要である．

本章では，3 章で開発したベンチマークシステムに，SSL によるリクエスト送信機能と複数のファイルへのリクエスト送信機能を実装し，これらによるサーバの挙動の違いを明らかにする．

4.1.　まえがき

WWW が登場した当初，扱うデータはテキストファイルのみであった．しかしインターネットの普及に伴い，WWW で扱われるデータの種類も変化してきている．

WWW ブラウザ Mosaic の登場により，画像ファイルが扱われるようになった．そして現在では，音声や動画などのマルチメディアファイルも扱われるようになっている．また，掲示板や抽選キャンペーンなど，cgi を用いたユーザ参加型のコンテンツも増加している．このように WWW で扱われるコンテンツが変化するにつれ，WWW ベンチマークもそれに対応する必要がある．

まず，従来のベンチマークシステムでは，単一のファイルにアクセスをし，その結果

から性能指標を得ている．しかし，現実の WWW サーバで単一のファイルのみにリクエストが発生することは通常考えられない．単一ファイルへのベンチマークでは，アクセスされるファイルが WWW サーバのディスクキャッシュやメモリキャッシュの影響で実際の WWW サーバの挙動を正しく反映されない可能性がある．

そこで我々は，ベンチマークを行う際に任意のファイルへのアクセスを可能とするベンチマークシステムを開発した．このベンチマークにより，実際の WWW サーバと同様のアクセスパターンを再現することが可能となる．また，ディスクやメモリのキャッシュなども考慮した性能改善指針や，パラメータチューニング指針も得ることが可能になる．

一方，WWW が社会に普及し，さまざまな用途で用いられるようになっており，WWW 上でのオンラインショッピングや商取引，オークションなどもさかんに行われるようになってきた．これらのコンテンツではユーザの住所や名前，電話番号やカードの番号など，重要な個人情報が送受信される場合が多く見られる．このような情報の送受信の際には，個人情報保護の観点から暗号化通信が行われることが多い．

WWW 上での暗号化通信には一般的に SSL (Secure Socket Layer) が用いられる．SSL とは，既存のネットワーク技術やプロトコルに変更を加えずに，インターネットなど安全性の保証されていないネットワークでやりとりされる情報を暗号化することを目的として Netscape Communication 社によって開発された．その後，IETF[26] で管理され，現在は TLS1.0 (Transport Layer Security)[27] として標準化されている．しかし，現状では代表的な WWW ブラウザであるインタネットエクスプローラや mozilla が標準で対応していることから，SSL Ver.3.0 が標準的に用いられている．

上記で述べたように，WWW 上で個人情報の送受信が行われるようになるにつれ，SSL による暗号化通信も増加していると考えられる．本学の学外との通信の計測結果でも WWW の通信の約 3%程度が SSL による通信であると判明している．今後もこのような SSL による暗号化通信が増加することは容易に予想できるため，SSL による通信が WWW サーバの挙動にどう変化を与えるかを測定するのは重要である．

そこで，SSL によるリクエスト送信機能をベンチマークに実装し，SSL による暗号化，復号化の処理が WWW サーバの挙動にどう影響を与えるかを確認する．

上記二点を実現するため，我々は 3 章で開発したベンチマークシステム ANMA に SSL によるリクエスト送信機能と複数のファイルへのリクエスト送信機能を実装し，これらによるサーバの挙動の違いを明らかにする．

4.2. システムの概要

我々は3章で開発したベンチマークシステム ANMA に以下の機能を追加した．本節ではこれらの機能について説明する．

- さまざまなファイルへのアクセス機能
- SSL によるリクエスト送信機能

4.2.1 多数のファイルへのリクエスト送信機能

IPv6 によるリクエスト送信機能と同様，あらかじめアクセス対象となるファイルのリスト (アクセスリストと呼ぶ) を用意し，それを元にリクエストパターンを決定する．アクセスリストのそれぞれの行は以下の項目からなる．

- 送信元 IP アドレス
  3章で述べたように，IPv6 と IPv4 の両方のプロトコルスタックでのリクエスト送信を可能とするための項目である．
- URI
  アクセス対象とするファイル名を指定する．
- SSL
  SSL による暗号化通信を行うか否かの項目である．

```
#souce IP        URI              SSL
192.168.10.13    /index.html      0
192.168.10.13    /test.html       0
192.168.10.13    /banner-1.png    0
```

図 4.1 アクセスリストの例

図 4.1 にアクセスリストの例を示す．このアクセスリストを用いた場合，3つのファイル対して順番にリクエストを送信する．つまり，アクセスリストに含まれる URI をさま

ざまに変更することで，さまざまなファイルに対してアクセスし，より現実に近いアクセスパターンを再現することができる．

また，このアクセスリストを読み込むタイミングについて以下に述べる．ソケット作成毎にファイルを読み込むと，ベンチマークを行う計算機内でのファイルのオープン，クローズの回数が増え，オーバーヘッドになる可能性がある．

そこで，ベンチマークを開始する前に，送信するリクエスト数の構造体を作成する．構造体のメンバには対象 WWW サーバのアドレス，アクセス対象となるファイル名（コンテンツ名)，SSL による暗号化を行うか否かなどが含まれており，ソケット作成の際にはこの構造体から必要な情報を読み取る．このことにより，クライアント側でのファイルを読み書きする回数を減らし，オーバーヘッドを解消している．

4.2.2　SSL によるリクエスト送信機能

SSL によるリクエスト送信は OpenSSL[28] ライブラリを用いて実現した．本実装は FreeBSD 4.7,4.8–STABLE および NetBSD–1.6 で開発したが，OpenSSL ライブラリは多くの UNIX で利用可能であり，さまざまな UNIX プラットフォームで動作可能である．

4.3.　記録する性能指標

本実装で記録する性能指標は 3.2.3 節で述べたものと同様，スループット，コネクション継続時間 (Tc)，応答時間 (Tr)，各 HTTP コネクションの終了形態である．

4.4.　実験

本実装による有効性を示すための実験を行った．

4.4.1　実験環境

実験ネットワークのトポロジを図 4.2 に，用いた計算機のスペックとソフトウェアを表 4.1 に示す．本実験では，クライアント，サーバともに多数のコネクションを同時に扱う必要がある．そのため，カーネルパラメータの MAXUSER の値を 1024 とし，最大ソケット数 (kern.ipc.maxsockets) を増加させた．

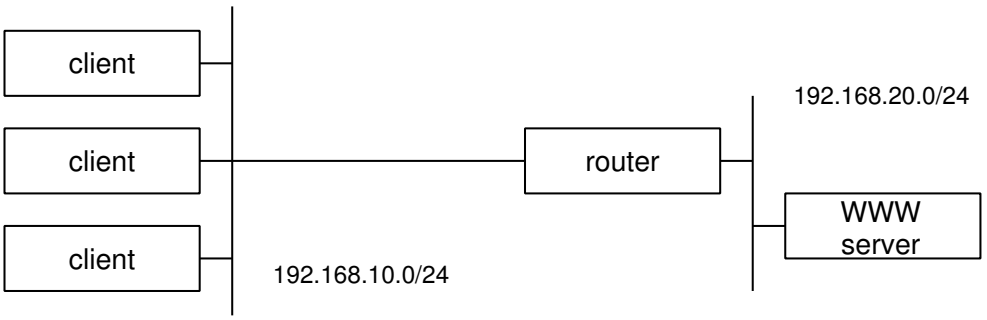

図 4.2 実験ネットワーク

表 4.1 実験に用いた計算機とソフトウェア

ホスト	OS	CPU	メモリ	ソフトウェア
クライアント	FreeBSD–4.7–PRERELEASE (2003/09/24)	Pentium–III 800MHz	512M	anma–1.0
中間ルータ	FreeBSD–4.7–PRERELEASE (2003/09/24)	Pentium–III 1GHz (Dual)	512M	—
サーバ	FreeBSD–4.7–STABLE (2003/09/24)	Pentium–III 800MHz	512M	Apache–1.3.27 + mod_ssl

サーバプログラムにはWWWサーバとして最もよく利用されている Apache [18] のバージョン 1.3.27 に，SSL による暗号化リクエストを処理するモジュールである mod_ssl[29] を追加した．適応した．apache の設定はすべてデフォルトとした．クライアントには，提案システムである ANMA を用いた．なお，ネットワークは全てギガビットイーサネットである．

4.4.2 実験 1

まず，SSL による暗号化による負荷がどの程度であるかを検証する実験を行った．

サーバにより高い負荷をかけるため，クライアントは 3 台の計算機を用い，3 台から同

表 4.2　実験 1 のアクセスパターン

	ファイル サイズ	SSL を 用いるか
A	8k	×
B	8k	○
C	16k	×
D	16k	○

時にリクエスト送信を行った．アクセスパターンは表 4.2 に示すように，要求するファイルのサイズを 8 キロおよび 16 キロバイトとし，1 台当たりの同時コネクション数を 1000, 2000，3000 と変化させた．また，総リクエスト数は同時コネクション数の 10 倍とし，これについて SSL を用いた場合と用いない場合での挙動の差を明らかにする．

結果

表 4.3　実験 1 の結果 (A と B の比較)

同時	平均 Tr (ms)		平均 Tc (ms)	
コネクション数	A	B	A	B
1000	172	7766	251	16424
2000	393	9838	466	33778
3000	746	9882	844	36926

表 4.4　実験 1 の結果 (C と D の比較)

同時	平均 Tr (ms)		平均 Tc (ms)	
コネクション数	C	D	C	D
1000	176	7521	277	21493
2000	379	10897	482	41575
3000	790	9459	884	37477

表 4.3 にファイルサイズ 8 キロバイトの場合の，表 4.4 に 16 キロバイトの場合のコネクション継続時間 (Tc)，応答時間 (Tr) に関する結果を示す．

SSL を用いた場合と用いない場合では大きな時間の差があるとわかる．その差はいずれも 10 倍以上となっており，SSL による暗号化，復号化の処理は通常の HTTP によるファイル転送に比べて非常に負荷が高いと言える．しかし，ベンチマーク内でも SSL による処理を行っているため，この時間の差は全てがサーバに起因するものではない．

また，Tr は同時コネクション数には依存しているが，ファイルサイズに依存していない．それに対し，Tc はファイルサイズが大きくなると時間が長くなっている．これは Tr は要求されたファイルの転送開始までの時間であるが，Tc はファイルの転送時間も含んでいるからである．ここで，SSL を用いない場合は，同時コネクション数によらず，$Tc - Tr$ の値はほぼ一定である (8 キロの場合で約 80 ミリ秒，16 キロの場合で約 100 ミリ秒)．このことから，同時に多数のリクエストがサーバに到着した場合，始めのコネクション確立は遅くなるが，一旦確立されたコネクションについては滞り無くファイルの転送が行われているといえる．一方，SSL 用いた場合は，同時コネクション数が増加すると $Tc - Tr$ の値も増加している．これは，SSL による負荷が大きいため，同時にたくさんの暗号化処理を行うことでファイルの送信にも影響を与え，結果として Tc も長くなっていると考えられる．

なお，全ての場合において，コネクションの終了形態に関するデータは全て NORMAL となっており，サーバは正常の状態であったと言える．

4.4.3 実験 2

次に，複数のファイルにアクセスできる機能を用いた実験を行った．本実験では，転送する合計のファイルサイズが同じ場合，単一のファイルの場合と複数のファイルの場合でのサーバの挙動の違いを明らかにする．ここでは，いずれの場合も SSL による暗号化は用いない．アクセスパターンを表 4.5 に示す 4 パターンとした．パターン A は 32 キロバイトのファイルに対して同時コネクション数を 1000，2000，3000 の 3 通りである．パターン B は種類の違う 16 キロバイトのファイル二つに対して交互にリクエストを送信する．同時コネクション数については，32 キロバイトのファイル一つの転送と 16 キロバイトのファイル二つの転送での負荷の違いを調べるため，2000，4000，8000 の 3 通りとした．パターン C，D についても同様である．

表 4.5　実験 2 のアクセスパターン

	ファイル サイズ	同時 コネクション数
A	32k	1000,2000,3000
B	16k，16k	2000,4000,6000
C	16k，8k，8k	3000,6000,9000
D	8k，8k，8k，8k	4000,8000,12000

結果

表 4.6　実験 2 の結果 (時間に関するもの)

同時	平均 Tr (ms)				平均 Tc (ms)			
コネクション数	A	B	C	D	A	B	C	D
1000	175	388	689	1022	303	489	806	1160
2000	404	1061	1949	1917	564	1212	2181	2147
3000	824	2067	2028	1774	999	2312	2174	1909

表 4.6 にコネクション継続時間 (Tc)，応答時間 (Tr) に関する結果を示す．同時コネクション数は 1000，2000，3000 となっているが，パターン A での値であり，パターン B の場合は 2000，4000，6000 であり，C，D についても同様である．

同じ転送量の設定の場合，一つのファイルに対するアクセスより多くのファイルに同時にアクセスする方が時間がかかっていることが分かる．これは，総転送量は同じでも，同時に処理するリクエスト数が増えており，コネクション確立のオーバーヘッドが影響していると考えられる．コネクションの終了形態に関しては表 4.7 に示す割合になっていた．ここでも，同じ転送量の場合でも，同時コネクション数を大きくした場合の方が RESET で終了するコネクションが増えており，転送量自体よりも同時に処理するコネクション数の影響が大きいことが分かる．

つまり，多くのリクエストを処理する WWW サーバの場合，総サイズが同じであれば，コンテンツを複数に分けず，なるべく一つのファイルとした方がサーバの負荷が低いと言える．

表 4.7 実験 2 の結果 (コネクションデータに関するもの)

同時	RESET の割合 (%)			
コネクション数	A	B	C	D
1000	0	0	0	0
2000	0	0.5	2.5	5.5
3000	0.5	2.0	4.7	6.5

また，$Tc - Tr$ はパターン A よりパターン D の方が短くなっている．転送するファイルのサイズ自体はパターン A が 32 キロバイトに対し，パターン D が 8 キロバイトであるため，このような結果になったと思われる．ここでも実験 1 と同様，同時コネクション数を多くするとコネクション確立に要する時間には影響を与えるが，一旦確立したコネクションに関しては滞り無くファイルの転送が行われていることが分かる．

4.4.4 実験 3

実験 1 および実験 2 では，ファイルに対するリクエストを送信した．しかし，実際の WWW サーバでは単にファイルを返すだけではなく，cgi などによってサーバ内部でなんらかの処理をした結果を返す場合もある．そこで，cgi に対するリクエストではファイルに対するリクエストとサーバの挙動がどのように異なるかを調査した．そのため，図 4.3 で示すようなファイルを転送するだけの cgi プログラムに対するリクエストを送信した．要求するファイルサイズは 1，4，8，16，32 キロバイトの 5 種類，同時コネクション数は 1000，2000，5000 の 3 通りとした．同じ条件で cgi を用いずに直接ファイルにリクエストを送信した場合 (3.4.3 節の実験による) と比較する．また，この実験ではクライアントの計算機は 1 台だけを用いた．

結果

表 4.8 に Tr の平均を示す．全体の傾向としてはファイルサイズの増加に伴い，両者の差も大きくなっているが，コネクション数の増加はあまり影響していないことが分かる．しかし，単にファイルを返すだけ場合に比べて，大幅に時間が長くなっている．この原因として，サーバ内部で perl 呼び出しているために，処理が滞るためと考えられる．

```
#!/usr/bin/perl

$file="index.html";
open(IN, "$file") || die("Can't open $file");
print <<"HERE";
Content-type: text/html

HERE
while(<IN>) {
    print $_;
}

  close(IN);
```

図 4.3　実験に用いた cgi (perl スクリプト)

コネクション情報に関しては，全てのコネクションが NORMAL で終了していた．

4.5.　むすび

WWW 上で扱われるコンテンツの種類は日々変化を続けている．現在では，SSL を用いた暗号化通信が行われるようになってきている．そこで本章では，3 章で開発したベンチマーク ANMA に SSL による暗号化リクエスト送信機能を実装した．そして実験の結果，SSL による暗号化を行うと，通常のコンテンツ配送に比べて 10 倍以上の時間がかかることが判明した．つまり，SSL を頻繁に用いるようなサーバでは，SSL のアクセスのみを処理するサーバを用意したり，SSL を処理する専用のハードウェアを導入して負荷分散を行う必要があると考えられる．

また，既存のベンチマークのアクセスパターンは，単一ファイルのみであるか，システムがあらかじめ用意したパターンのみであった．しかしそのアクセスパターンでは，cgi を用いたインタラクティブなコンテンツの負荷や，実際の WWW サーバのアクセスパターンを反映できないという問題点がある．そこで，ベンチマークを行う際に，どのファ

表 4.8 実験 4 の結果 (msec)

	1000cons		2000cons		5000cons	
	file	cgi	file	cgi	file	cgi
1k	0.998	10.33	0.923	10.27	0.946	10.27
4k	1.730	10.98	1.711	10.83	1.870	10.85
8k	2.336	11.87	2.249	11.67	2.318	11.70
16k	3.306	13.80	3.155	12.80	3.226	12.80
32k	4.229	13.72	4.091	13.65	4.142	13.66

イルに何回アクセスするかというパターンを自由に設定できる機能実装した．そして，単一ファイルのみのリクエスト送信と複数のファイルに対するリクエスト送信を行い，その違いを検証した．その結果，総転送量が同じ場合単一ファイルへのアクセスよりも複数のファイルに対するアクセスの方がコンテンツ配送に時間がかかることが分かった．また，クライアント側からコネクションを切断する現象もみられた．このことから，多くのリクエストを処理する WWW サーバの場合，総サイズが同じであれば，コンテンツを複数に分けず，なるべく一つのファイルとした方がサーバの負荷を下げることができるといえる．

cgi に関する負荷については，指定されたファイルを返すだけの単純なものでも配送時間が大幅に長くなることが分かる．実際の WWW サーバで用いられる cgi は本章での実験で用いた単純なものではなく，さらに負荷が高くなり，応答時間が長くなると考えられる．ユーザ登録などに用いる cgi では，データベースに専用のサーバを用いるなどの処置が必要になると思われる．

しかし，WWW 上でのユーザ登録の場合など，サーバが受信した登録内容を一度クライアントに送信し，クライアントからの確認アクセスを要求するという場合は本章の機能では実現できない．このように，クライアントがサーバからの返答に応じたリクエストを再度サーバに送信するようなインタラクティブなコンテンツについても性能評価を行えるよう，サーバの返答に応じた新たなリクエストを送信する機能が必要である．これは ANMA の今後の課題である．

第 5 章　高速 WWW キャッシングシステムの計測

3 章や 4 章で実装したベンチマークシステムは WWW サーバのみでなく，WWW キャッシュサーバの性能測定も可能である．

本章では，WWW キャッシュサーバの測定例として，日立製作所が開発した WWW キャッシュサーバ，AWG (Active Web Gateway) の性能評価を行う．

5.1.　はじめに

キャッシュサーバに特化したベンチマークシステムとして，Winsconsine Proxy Benchmark (WPB) [30] や Polygraph [31] が挙げられる．しかし，これらの代理サーバベンチマークシステムではそれぞれのシステムであらかじめ用意されたパターンのアクセスでしか評価できない．

しかし，我々が実装した ANMA を用いることにより，さまざまなアクセスパターンにより代理サーバの性能を評価できる．本章では，その一例として，日立製作所が開発した WWW キャッシュサーバ，AWG の性能評価を行う．

本章では以下に挙げる項目の評価を行った．性能評価の際には ANMA と共に ENMA[17] によるパケットモニタも行った．

- 動的なコンテンツと静的なコンテンツでのキャッシュサーバの有効性
- 代理サーバのキューイング機能の有効性
- 代理サーバの HTTP ヘッダの取扱いに関する検証

5.2.　実験環境

本節では実験環境について述べる．

5.2.1　実験ネットワーク環境

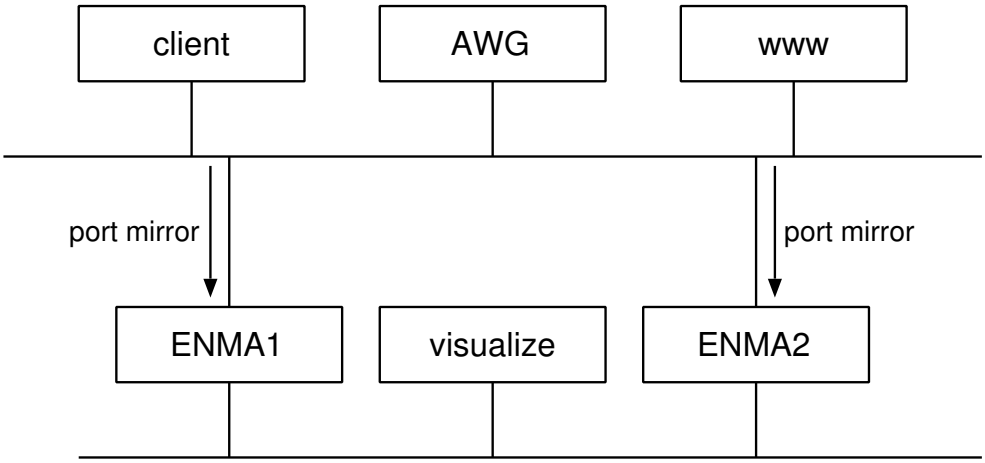

図 5.1　実験ネットワーク環境

図 5.1 に本実験のネットワーク図を示す.

client，AWG，www を同一セグメントに設置し，client に流れるパケットの観測用に ENMA1 を，www の観測用に ENMA2 を設置した．ENMA1，ENMA2 で得られた統計情報は管理用セグメントを通じて visualize に送信され，視覚化を行なう.

client，AWG および www が接続しているネットワークはギガビットイーサネットを，スイッチは Extremenetworks 社の Alpine3808 を用いた．パケットのモニタには，同スイッチの Mac アドレス学習機能を無効にし，flooding 機能を有効にすることで実現した.

5.2.2　ハードウェアおよびソフトウェア構成

実験に使用したハードウェアおよびソフトウェアを表 5.1 に示す．使用した OS は全て FreeBSD であり，キャッシュサーバが高負荷にならないよう，AWG を動作する計算機のみ CPU とメモリを他の計算機より高性能なものを用いた.

5.3.　cgi に対するアクセス実験

まず，アクセスされるたびにその内容が変化するコンテンツ (動的なコンテンツと呼ぶ) とそうではないコンテンツ (静的なコンテンツと呼ぶ) について，キャッシュサーバを

表 5.1 ハードウェアおよびソフトウェア構成

Host	OS	CPU	Memory	Sofoware
clinet	FreeBSD 4.7-RELEASE-p7	Pentimun-III 1GHz	512M	polygraph-2.5.4 anma
www	FreeBSD 4.7-RELEASE-p7	Pentimun-III 1GHz	512M	apache-1.3.27
AWG	FreeBSD 4.7-RELEASE-p7	Athlon MP1900+	1G	AWG/2.0 (Rev 2.0 2002/09/18)
ENMA	FreeBSD 4.7-RELEASE-p7	Pentimun-III 1.2GHz	512M	ENMA-1.0a
visualize	FreeBSD 4.7-RELEASE-p7	Pentimun-III 1.2GHz	512M	ENMA-1.0a

経由した場合とそうでない場合，キャッシュサーバと WWW サーバの挙動がどのように異なるかを観測した．

5.3.1 実験概要

動的なコンテンツと静的なコンテンツに対して交互にリクエストを送信し，その挙動を観測した．

動的なコンテンツとしては perl による cgi プログラムを用いた．このプログラムにアクセスすると以下のような動作を行う．

1 あらかじめ指定されたファイルを読み込む．

2 読み込んだファイルから 10 行ランダムに選択し，変更する．

3 クライアントにはテストページのみを送信する．テストページのサイズは 0.1k バイトである．

これに対する静的なコンテンツは，上記の cgi で指定するファイルそのものとした．

ANMA による同時コネクション数を 100 から 1000 へと変化させながら，キャッシューサーバを利用したアクセスと直接 WWW サーバへのアクセスを交互に行なった．また，cgi で対象とするファイルのサイズは 50，100，150，200 キロバイトの 4 通りで実験した．

5.3.2　測定結果

図 5.2 ，5.3 に ENMA による観測結果を示す．左の図がクライアント側の観測結果，右の図がサーバ側の観測結果である．図 5.2 は HTTP リクエストの到着レートである．クライアント側は到着レートが 50/sec 程度となっているのに対し，サーバ側では 25/sec 程度となっている．また，図 5.3 に示したトラヒック状況でも，サーバ側のトラヒックがクライアント側の約半分になっている．これらの結果より，約半数のリクエストがキャッシュされていると判断できる．

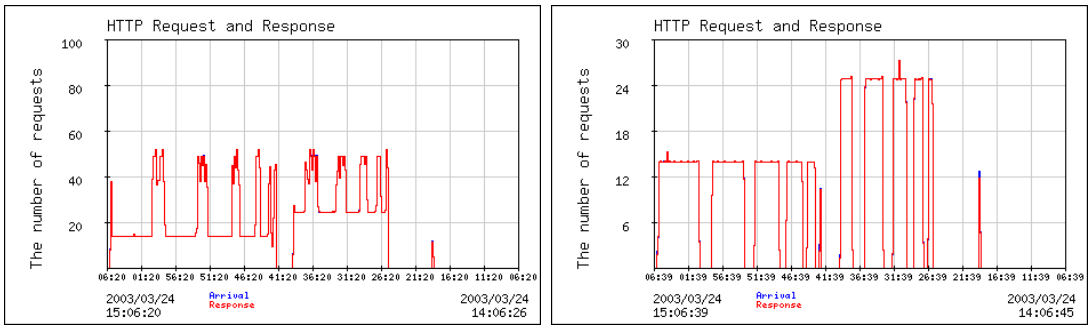

図 5.2　HTTP リクエスト到着レート

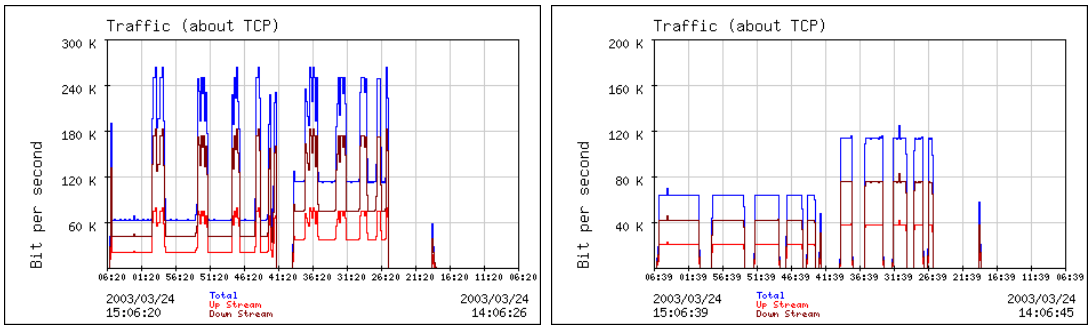

図 5.3　トラヒック状況

表 5.2 コネクション継続時間の平均

	キャッシュサーバ利用 (msec)	キャッシューサーバなし (msec)
50k	20.8	9.45
100k	20.6	9.54
150k	20.0	9.55
200k	20.2	9.63

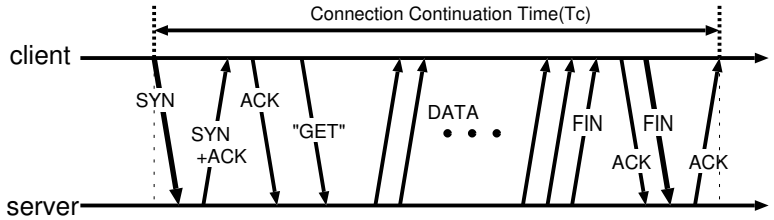

図 5.4 コネクション継続時間

次に，ANMA により測定したコネクション継続時間の平均を表 5.2 に示す．ここでのコネクション継続時間とは TCP/IP の接続開始からデータの転送，コネクションの切断まで全体の長さである (図 5.4)．全ての場合において，キャッシュサーバを利用した場合の方が 1 コネクションあたりの時間が長くなっている．これは，約半分のリクエストがキャッシュされずに WWW サーバへ転送されるため，キャッシュサーバでのヒット確認，リクエスト転送が行なわれるためと考えられる．

5.4. キューイング機能に関する実験

AWG のキューイング機能の有効性を検証するための実験を行なった．

5.4.1 実験概要

キューイング機能の有効時，無効時にそれぞれ，cgi によるアクセスを行なった．用いた cgi は前節の実験と同じものである．

用いたクライアントは独自に作製したベンチマークプログラム (ANMA) で，ベンチ

マーク中に ENMA により，サーバ側とクライアント側の両方を観測した．リクエスト送信については，同時コネクション数を 100，300，500，700，900 と増加させた．また，cgi で利用するファイルのサイズは 200 キロバイトとした．

キューイングを有効にする場合はキャッシューサーバの uri_ctl 機能により，WWW に対する全てのアクセスに対して同時処理数を 10 にした．

5.4.2　測定結果

まず，図 5.5 にキューイング有効時のリクエスト到着レートに関するグラフを示す．左の図がクライアント側，右の図がサーバ側の観測結果である．この図では 3 セットのリクエスト送信をしているが，クライアント側ではそれぞれの初めの到着レートが高くなっている．これは ANMA の性質上，実験開始時に多数のコネクションを確立しようとするために発生する．しかし，サーバ側のグラフによるといずれも 12 程度に収まっており，クライアント側で見られるようなスパイクは観測されない．これは，キャッシューサーバのキューイング機能により，WWW サーバに送信されるリクエストが押えられているためと考えられる．

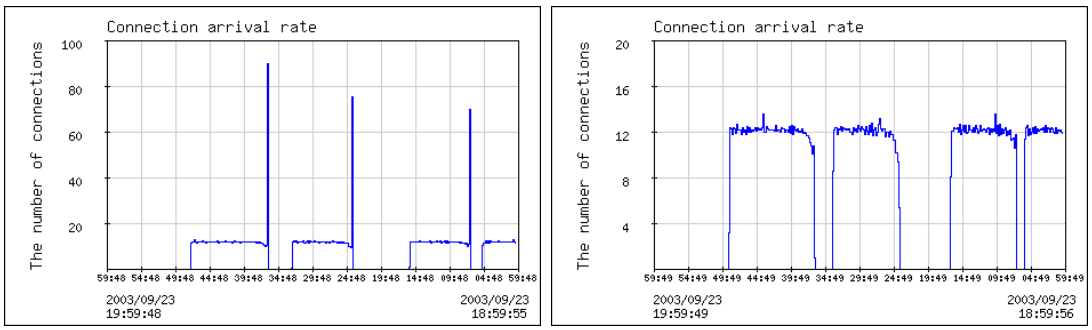

図 5.5　キューイング機能有効時のリクエスト到着レート

また，図 5.6 にキューイング有効時の HTTP の同時処理数に関するグラフを示す．図 5.5 同様，左の図がクライアント側，右の図がサーバ側の観測結果である．クライアント側の観測結果では同時処理数が 500，700，900 と変化しているのに対し，サーバ側では常に 10 以下になっている．ここでも，AWG のキューイング機能によってサーバへのリクエストが押えられていることが分かる．

一方，キューイング機能が無い場合のリクエスト到着レートと HTTP の同時処理数の

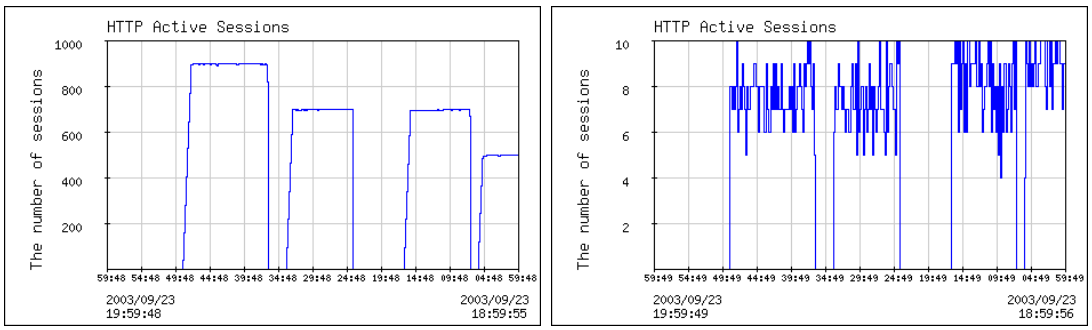

図 5.6 キューイング機能有効時の同時処理数

結果をそれぞれ図 5.7，5.8 に示す．図 5.7 より，キューイング機能を無効にしたことでクライアント側で観測される実験開始時のスパイク状のアクセスがサーバ側でも見られるようになった．また，図 5.8 でも，クライアント側の同時コネクション数を増加させるにつれてサーバ側のでも同じ数の同時コネクション数が観測されている．このことからも，キューキング機能の無効化により全てのリクエストがサーバに転送されていることが分かる．しかし，クライアント側の同時処理数が 700 を越えるもて，サーバ側には 600 の同時処理数しか観測できていない．これはキャッシュサーバが同時処理数 600 を越えると処理が滞っていると予想される．

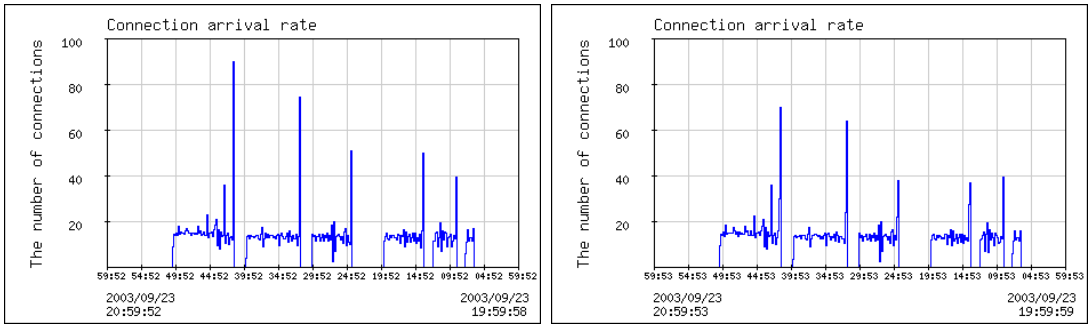

図 5.7 キューイング機能無効時のリクエスト到着レート

次に，設定した同時コネクション数毎にコネクション継続時間の平均を算出した結果を表 5.3 に示す．この結果より，全ての場合においてキューイングが無い場合の方が時間が短くなっている．これは，cgi はキャッシュが有効に働かないことと，キューイングすることでサーバが同時に処理するリクエストが減少することが原因と思われる．

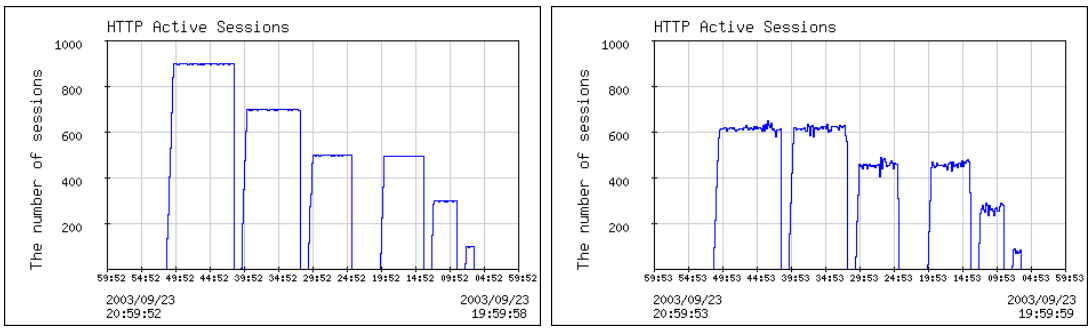

図 5.8　キューイング機能無効時の同時処理数

表 5.3　コネクション継続時間の平均

	キューイングあり (sec)	キューイングなし (sec)
100	8.45	7.33
300	23.5	21.6
500	39.0	35.5
700	54.8	49.0
900	70.0	55.5

5.5.　ヘッダの長さに関する実験

本章で測定の対象としたキャッシューサーバには HTTP のレスポンスヘッダの大きさに応じて，バッファサイズを変化させる機能がある．これは，サーバがリクエストに対して cookie 付きで返答を返す場合に影響がある機能である．この機能の有効性を検証するための実験を行った．

5.5.1　実験概要

HTTP のレスポンスヘッダの長さを変化させてキャッシュサーバ経由でのアクセスを行なった．用いたキャッシュサーバのバッファサイズは 4K バイトで変化する．変化する前後での挙動を確認するため，特定ファイルへのアクセスのみ 4k バイト分のヘッダを追加するよう WWW サーバを設定した．アクセスするファイルのサイズは 2k バイト程度とし，同時コネクション数は 100 および 200 とした．そしてヘッダを追加したファイル

と追加しなかったファイルにアクセスを行なった.

5.5.2 測定結果

図 5.9 にリクエスト到着レートに関するグラフを示す．前節の実験同様，左の図がクライアント側，右の図がサーバ側の観測結果である．また，グラフの 44 分付近 (右側) がヘッダを追加しなかった場合，19 分付近 (左側) がヘッダを追加した場合である．ヘッダを追加しなかった場合，サーバ側でリクエストの到着は観測されていないことがわかる．

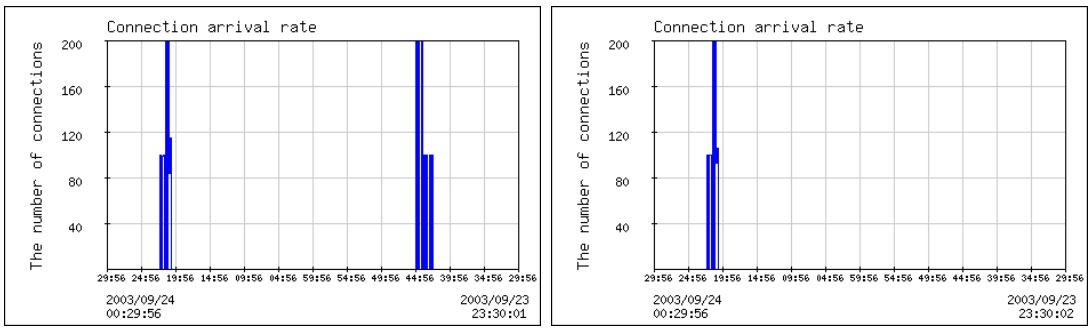

図 5.9 ヘッダサイズ変化時のリクエスト到着レート

このときのトラヒック状況を図 5.10 に示す．ここからも，ヘッダを追加しない場合はサーバ側ではトラヒックが観測されず，リクエストが全てキャッシュされていることが分かる．また，クライアント側の結果 (左の図) より，ヘッダを追加した時の方がより多くのトラヒックが流れていることも分かる．

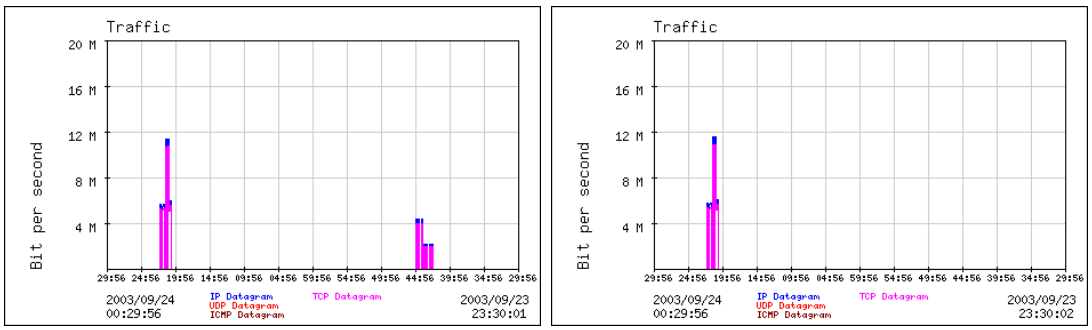

図 5.10 ヘッダサイズ変化時のトラヒック

次に，コネクション継続時間に関する結果を表 5.4 に示す．平均時間と最大時間にに大きな差はみられないが，最小のものについてはヘッダを追加した方が大幅に短くなっていることが分かる．

表 5.4　コネクション継続時間の平均

	ヘッダ追加 (msec)	ヘッダ追加なし (msec)
平均	95.1	98.3
最小	9.98	2.00
最大	1220	1190

5.6.　まとめ

本章では，ANMA のキャッシュサーバの測定例として，日立製作所の AWG を対象として評価実験を行った．

cgi によるアクセス実験では，約半数のリクエストがキャッシュサーバで処理され，残りの半数が WWW サーバへ転送されていることが分かった．しかし，キャッシューサーバを用いた場合，用いなかった場合に比べて 1 コネクションあたりの時間が長くなっている．また，cgi など動的なコンテンツの正確を考えるとキャッシューサーバで cgi をキャッシュするべきではないと考えられる．続いて，キャッシュサーバのキューイング機能に関する実験を行った．キューイング機能により WWW サーバの同時処理数を指定すると，指定した数のリクエストに押えられており，正しく動作していることが分かった．

このように，ANMA を用いることで従来のキャッシュサーバ専用ベンチマークでは実現できない，さまざまな角度からの性能評価が行えることが判明した．

第 6 章　結論

大規模な WWW サーバを設計するにあたって，あらかじめサーバの性能を評価することは重要である．従来から，WWW サーバの性能評価手法としてはベンチマークが一般に用いられてきた．しかし，ベンチマークシステムも WWW で取り扱われるコンテンツの内容の変化に対応する必要がある．しかし，既存のベンチマークではアクセスパターンが一定である，SSL による暗号化を考慮されていない，IPv6 に対応してないなどの問題がある．

そこで本論文では，このようなコンテンツの変化に対応し，現実の WWW サーバに近い環境で性能測定を可能とするため，新たなベンチマークシステム (ANMA) を開発した．ANMA は上記で挙げた問題点を解決し，アクセスパターンを自由に設定できるベンチマークシステムである．これにより，より現実に近い環境でのベンチマークが可能になった．

まず，2 章で運用中のサーバの性能を計測するためのシステムを開発した．このシステムは，パケットモニタリングを基にし，各 HTTP コネクションがどのように終了しているかを分類することでサーバの状態を把握するシステムである．このシステムを用いた実験の結果，HTTP のコネクションの約 25%がクライアントから切断されるような状況になると，サーバの処理が滞っていることが明らかとなった．

3 章では，ソースアドレスのアドレスファミリを基にしてアクセスパターンを決定する，IPv6 環境に対応するベンチマークの設計と実装を行った．そして，両方のプロトコルスタックでのリクエスト送信を確認し，IPv6 と IPv4 での WWW サーバの性能の違いを測定した．その結果，IPv6 のほうが応答時間や 1 コネクションに時間がかかることが判明した．さらにこの時間の差は OS のプロトコルスタックによるものではなく，WWW サーバプログラムの問題であることを確認した．

4 では，3 章で開発したベンチマークに SSL による暗号化リクエスト送信と，さまざまなコンテンツにリクエストを送信し，アクセスパターンを自由に指定できる機能を実装

した．そして，SSL を用いた場合，通常のコンテンツに比べて 1 コネクション当たり 10 倍以上の時間がかかることが明らかとなった．このことから，SSL を頻繁に用いるようなサーバでは，SSL のアクセスのみを処理するサーバを用意するなどの負荷分散が必要であると考えられる．また，総量が同じ場合でも，複数のコンテンツに分割すると，同時コネクション数が増加し，1 コネクション当たりの時間が長くなることも判明した．多くのリクエストを処理するサーバでは，細かいコンテンツを数多く用意するよりも，一つのファイルにまとめた方がサーバの負荷は低いといえる．

そして 5 章では，ANMA によって WWW サーバだけでなく，WWW キャッシュサーバの性能測定も可能であることを示した．ここでは日立製作所が開発している AWG について，アクセスするたびに内容が変化する CGI のような動的なコンテンツをキャッシュするとキャッシュサーバを用いない場合より応答時間が長くなることが判明した．また，キューイング機能の検証や HTTP ヘッダを取り扱う際のバッファサイズに関する検証も行った．これらの項目はいずれも一定のアクセスパターンしか持たない既存のベンチマークやキャッシュサーバ専用のベンチマークでは測定できない項目であり，ANMA の優位性を示したとも言える．

6.1.　今後の課題

本節では，本研究で明らかになった今後の課題を述べる．

6.1.1　IPv6 におけるネットワーク遅延の実現

3 章で述べたように IPv6 環境でのサーバやネットワークの性能解析の必要が出てきている．特にベンチマークによる性能評価を考えた場合，実験環境をいかにして現実の環境に近づけることがでるかが問題となる．そのためには，ネットワーク遅延の考慮も必要である．IPv4 環境では，Dummynet [25] を用いることでネットワーク遅延をエミュレートでき，これを利用したベンチマークシステムの開発も行われている [32][33]．しかし IPv6 にはネットワーク遅延をエミュレートできるシステムはまだ開発されていない．IPv6 環境でのネットワーク遅延実現は WWW サーバのベンチマークだけではなく，様々なシミュレートを行う際に必要となると考えられるため，今後の重要な課題といえる．

6.1.2 アクセスパターンに関する検討

4 章では，アクセスパターンを自由に設定できるベンチマークを実装した．この実装では，指定したコンテンツに対して繰り返してリクエスト送信が可能である．しかし，インタラクティブなコンテンツの場合，クライアントはサーバからの返答に応じたリクエストを再度サーバに送信することが考えられる．例えば，WWW 上でのユーザ登録の場合に，サーバが受信した登録内容を一度クライアントに送信し，クライアントからの確認アクセスを要求する場合がこれにあたる．このように一度のリクエスト送信とコンテンツ転送で終了しないアクセスをどのように再現するかを検討する必要がある．

6.1.3 アクセス予測に関する問題

新たに WWW サーバを構築する場合，そのサーバにどの程度のアクセスがあるのかを予想しなくてはならない．これは，性能評価を行うに際，サーバがどの程度の負荷に耐えられれば良いかを判断するためである．現存の WWW サーバを置き換える場合は現在や過去のアクセス状況からある程度の判断は可能であると思われるが，新規のサーバの場合，アクセス数の予想は困難である．新規のサーバの場合，これに加えてネットワークの帯域やネットワーク機器も含めた検討が必要である．

謝辞

本研究を行う機会を与えていただき，研究方針に関する御指導はもちろん，研究外のことがらにおいてもさまざまな御指導をいただきました，奈良先端科学技術大学院大学情報科学センター砂原 秀樹教授に心から感謝の意を表わします.

研究に関する御指導とともに，論文執筆におけるさまざまな御指導をいただきました奈良先端科学技術大学院大学情報科学センター藤川 和利助教授に深く感謝致します．また，私の拙い英語を直していただきましたことも重ねて感謝します.

本研究でのサーバ観測に用いたENMAの作者でもある奈良先端科学技術大学院大学情報科学センター中村 豊助手は大学院入学以来，常に研究の方針や内容について御指導いただきました．また，研究内容だけでなく，さまざまな面でもお世話になりました，重ねて感謝致します.

本研究で開発したANMAは本学OBであり，日本ヒューレット・パッカード株式会社の西村 敦隆氏が開発したベンチマークシステムが基になっています．研究の基礎を築いていただき，ここに感謝の意を表わします.

本研究では全国高等学校野球選手大会のインターネット中継に用いたWWWサーバを観測の対象とさせていただきました．貴重なデータを提供して頂いた皆様，特に朝日放送株式会社の吉田 豊一氏，香取 啓志氏，またNTTスマートコネクト株式会社の沖本 忠久氏，白波 瀬章氏に心から感謝致します.

奈良先端科学技術大学院大学情報科学センターの職員の皆様，同センター事務補佐員の能勢 佳苗女史は，さまざまな面から研究活動を支えてくださり，多くの興味深い話をいただきました．感謝いたします．また，同センター砂原研究室の学生，OB，OGの皆様，特に蟻川 浩氏，垣内 正年氏，和泉 順子女史には入学当初からさまざまな面で研究活動を支えていただきました，感謝致します.

最後に，研究活動を影で支えていただいた家族に心より感謝致します.

参考文献

[1] *NCSA Mosaic*.
`http://www.ncsa.uiuc.edu/SDG/Software/Mosaic/NCSAMosaicHome.html`.

[2] Hobbies' internet timeline v6.1.
`http://www.isoc.org/guest/zakon/Internet/History/HIT.html`.

[3] Louis P. Slothouber. A model of web server performance. In *Fifth International World Wide Web Conference*, Paris, France, May 6-10 1996.

[4] R.Fielding, H.Frystyk, and T.Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.0*, May 1996. `http://www.ietf.org/rfc/rfc1945.txt`.

[5] R.Fielding, J.Gettys, J.Mogul, H.Frystyk, and T.Berners-Lee. *Hypertext Transfer Protocol – HTTP/1.1*, Jan 1997. `http://www.ietf.org/rfc/rfc2068.txt`.

[6] Jeffrey C. Mogul. The case for persistent-connection http. In *SIGCOMM '95 Symposium on Communications Architectures and Protocols*, pp. 229–313, Cambridge, MA, August 1995.

[7] Venkata N. Padmanabhan and Jeffrey C. Mogul. Improving http latency. `http://www.ncsa.uiuc.edu/SDG/IT94/Proceedings/DDay/mogul/HTTPLatency.html`.

[8] John Heidemann. Performance interactions between p-http and tcp implementations. *ACM Computer Communication Review*, Vol. 27, No. 2, pp. 65–73, April 1997.

[9] Balachander Krishnamurthy, Jeffrey C. Mogul, and David M. Kristol. Key differences between http/1.0 and http/1.1. In *Eighth International World Wide Web Conference*, Toronto, Canada, May 1999.

[10] Almeida Jussara Almeida Virgilio and Murta Cristina. Performance analysis of a www server. Technical report, Computer Science Department, Boston University, August 1996.

[11] V. Almeida J. Almeida and D. Yates. Measuring the behavior of a world-wide web server. Technical report, Computer Science Department, Boston University, October 1996.

[12] Gaurav Banga and Peter Druschel. Measuring the capacity of a web server. In *USENIX Symposium on Internet Technology and Systems*, Monterey, California, USA, December 1997.

[13] K. Thompson, G. J. Miller, and R. Wilder. Wide-area Internet traffic patterns and characteristics. In *IEEE Network*, Vol. 11, pp. 10–23, Nov 1997.

[14] David Mosberger and Tai Jin. httperf: A Tool for Measuring Web Server Performance. In *First Workshop on Internet Server Performance*, pp. 59–67. ACM, Jun 1998.

[15] Standard Performance Evaluation Corporation. SPECweb 99 benchmark, performace results, 2000.
`http://www.spec.org/osg/web99/`.

[16] G. Trent and M. Sake. WebSTONE: The first generation in HTTP server benchmarking, 1995. Silicon Graphics White Paper.

[17] 中村豊, 知念賢一, 砂原秀樹, 山口英. パケットモニタによる WWW サーバの性能計測システムの設計と実装. 電子情報通信学会論文誌, Vol. J83-D-I, No. 3, pp. 329–338, Mar 2000.

[18] Apache Software Foundation. The Apache web server.
`http://www.apache.org/`.

[19] Robert H'obbes' Zakon. Hobbes' Internet Timeline - the definitive arpanet & internet history, Apr 2002.
`http://www.zakon.org/robert/internet/timeline/`.

[20] Venkata N. Padmanabhan and Jeffrey C. Mogul. Improving HTTP latency. *Computer Networks and ISDN Systems*, Vol. 28, No. 1–2, pp. 25–35, Dec 1995.

[21] R. McGrath. Performance of Several Web Server Platform, Jan 1996. R.E. McGrath, Performance of Several Web Server Platform, NCSA University of Illinois Urbana Champaign, 5 February, 1996.

[22] Yutaka Nakamura, Ken-ichi Chinen, Suguru Yamaguchi, and Hideki Sunahara. An Analysis of WWW Server Status by Packet Monitoring. *IEICE TRANSACTIONS on Information and Systems*, Vol. E83-D, No. 5, pp. 1012–1019, May 2000.

[23] Takao Nakayama, Yutaka Nakamura, and Hideki Sunahara. The Real-Time State Observation System for WWW Server. In *INET2002*, Jun 2002.

[24] Hewlett-Packard. The netperf network performance benchmark. `http://www.netperf.org/`.

[25] Luigi Rizzo. Dummynet: a simple approach to the evaluation of network protocols. *ACM Computer Communication Review*, Vol. 27, No. 1, pp. 31–41, Jan 1997.

[26] *IETF(The Internet Engineering Task Force)*. `http://www.ietf.org/`.

[27] T.Dierks and C.Allen. *The TLS Protocol Version 1.0*, Jan 1999. `http://www.ietf.org/rfc/rfc2246.txt`.

[28] Openssl: The open source toolkit for ssl/tls. `http://www.openssl.org/`.

[29] Ralf S. Engelschall. mod_ssl: The Apache Interface to OpenSSL. `http://www.modssl.org`.

[30] Jussara Almeida and Pei Cao. Wisconsin proxy benchmark 1.0. `http://www.cs.wisc.edu/ cao/wpb1.0.html`.

[31] *Web Polygraph*. `http://www.web-polygraph.org/`.

[32] 西村敦隆. ネットワーク遅延を考慮した www サーバベンチマークシステム. 工学, 奈良先端科学技術大学院大学, 情報科学研究科情報システム学専攻, Feb 2002. NAIST-IS-MT0051078.

[33] 西村敦隆, 中山貴夫, 中村豊, 砂原秀樹. ネットワーク遅延を考慮した WWW サーバベンチマークシス テムの開発. 電子情報通信学会技術研究報告, pp. 63–69, Nov 2001.

著者研究業績

B.1. 学術論文

1 中山貴夫，中村 豊，砂原 秀樹: “IPv6 環境を考慮した WWW ベンチマークシステム”. 電子情報通信学会 和文論文誌，Vol.J86-B，No.8，pp.1515–1522, Aug 2003.

B.2. 国際会議 (査読付き)

1 Takao Nakayama, Yutaka Nakamura, and Hideki Sunahara: “The Real-Time State Observation System for WWW Server.” In proceedings of INET2002, Jun 2002.

2 Takao Nakayama, Yutaka Nakamura, and Hideki Sunahara: “A WWW Server Benchmark System in IPv6 Environment.” In proceedings of 2003 Symposium on Applications and the Internet Workshops, pp.258–261, Jan 2003.

B.3. 国際会議 (査読無し)

1 Takao Nakayama: “A WWW Server Management Support System”. 2003 NAIST COE International Symposium - Ubiquitous Networked Media Computing -, pp.19–21, Oct 2003.

B.4. 国内研究会

1 中山貴夫, 中村豊, 知念賢一, 砂原秀樹.: “WWW サーバにおける特徴抽出のための過渡解析手法”. 電子情報通信学会技術研究報告, pp.7–12, Jan 2000.

B.4.1 共著

1 寺田直美, 中山貴夫, 中村豊, 砂原秀樹.: “ストリーミング配信サーバにおける管理支援システムの設計”. 電子情報通信学会技術研究報告, 第 102 巻, pp.13–18, Dec 2002.

2 寺田直美, 中山貴夫, 中村豊, 砂原秀樹.: “ストリーミング配信サーバにおける管理支援システムの設計”. インターネットコンファレンス 2002 論文集 (Work in Progress), p.118, Oct 2002.

3 西村敦隆, 中山貴夫, 中村豊, 砂原秀樹.: “ネットワーク遅延を考慮した WWW サーバベンチマークシステムの開発”. 電子情報通信学会技術研究報告, pp.63–69, Nov 2001.