第III部

ネットワークトラフィック統計情報の収集と解析

第3部
ネットワークトラフィック統計情報の収集と解析

第1章　MAWI ワーキンググループ

MAWI (Measurement and Analysis on the WIDE Internet) ワーキンググループは、トラフィックデータの収集と解析、また、データの保存と利用に関する活動を行っている。MAWI WG では WIDE プロジェクトの特徴を活かした研究をするため、「広域」「多地点」「長期的」の三つの項目に重点を置いたトラフィックの計測・解析を行っている。

1. 広域で行う
   インターネットの最大の特徴は、大規模な広域ネットワークにある。しかし、トラフィック情報には組織の機密保持やプライバシー保護の問題が伴うため、特に不特定多数のトラフィックを含む広域データを第三者が入手するのは困難である。ひとつの企業や組織内といった狭い範囲でデータを取ることは各組織で出来るが、広域バックボーンでのデータ収集はバックボーンを持っている WIDE だからできる。
2. 多地点で見る
   インターネットのもう一つの大きな特徴は、自律したネットワークが相互接続して経路制御を行い、エンド・エンドで通信制御を行う分散システムにある。したがって、ある地点でトラフィックを観測しても、ネットワーク全体の状態を捉えることは不可能である。観測者はあくまでその観測点から見たインターネット像が得られるだけで、別の観測点からはまったく別の世界が見えているかも知れない。MAWI WG では、多地点で観測したデータを照らし合わせることによって、より広い範囲のネットワークの状態を把握する手法や、それを俯瞰で可視化することによって直観的に分かりやすく観測する手法について研究を行っている。
3. 長期間行う
   ネットワークのトラフィックの挙動は、TCP の特定のアルゴリズムが関係するようなミクロなものから、過去 10 年間のトラフィック量の推移のようなマクロなものまで、幅広い時間スケールに渡っている。また、ミクロな挙動についても、インターネットのマクロなレベルの発展に伴って、次第に変わっていく。したがって、一日や一週間といった短期間の計測も重要だが、何年間という長いスパンでデータを取り続けることが非常に重要になる。しかし、長期的にデータを収集し、その蓄積を持つことは、ある日誰かが思い付いてできるものではない。そこで、ワーキンググループとしてメンバーが協力して継続的なデータ収集を行っていくことが必要である。

今回の報告書では、第 2 章において、集約型トラフィックプロファイラを使った国際線の計測について報告する。このツールは、WIDE バックボーンのトラフィックをリアルタイムかつ長期的にモニタリングする目的で作られ、また、急増する分散型 DoS アタックの早期検出にも役立っている。ここでは、2001 年度の 1 年間の国際線トラフィックの傾向を報告する。

第 3 章では、WWW サーバーの評価手法として、ネットワーク遅延を考慮したベンチマークシステムを報告する。既存のベンチマークシステムでは、遅延のある広域環境を想定した評価が出来なかった。ここでは、遅延によるサーバー性能への影響を明らかにし、遅延を含めたより現実に近い環境でのシステム評価を可能にするシステムを提案し、既存のベンチマークシステムとの比較評価を行った。

MAWI WG では、バックボーンから得られたデータをホームページ上で公開している。公開されているデータは、プライバシーの保護を考慮し、必要に応じて解析に不要な部分を削除する、あるいは、アドレスをスクランブルするなどの加工を行っている。また、データ公開のためのルールや、そのために開発したツールも公開している。これらのデータは、研究目的であれば、WIDE 外の研究者でも自由に利用できる。

第 2 章　AGURI を用いた WIDE インターネット国際線のトラフィック傾向

2.1 はじめに

WIDE インターネットのような広域なネットワークを運用し続けていくためには、トラフィックモニタリングを多地点、かつ長期間行い、ネットワークの現状に適した通信機器の設置、設定を行う必要がある。

しかし、現存するネットワークモニタリングツールは長期に渡ってトラフィックの傾向を収集し続けることが難しい。

そこで、MAWI ワーキンググループでは収集したトラフィックを効果的に集約することによって、ネットワークの特徴を抽出することのできるトラフィックモニタリングツール AGURI[27] の設計、実装を行った。

本報告書では、実際に AGURI を用いて WIDE インターネット国際線でデータを収集し、対象とした国際線のトラフィックの傾向を示す。

2.2 AGURI

AGURI (Aggregation-based Traffic Profiler) は、1) トラフィック中の特徴的なフロー傾向を残しつつ、2) 短期間から長期間に渡って利用可能なトラフィックモニリングツールである。

AGURI をネットワーク運用に用いた場合、以下に示される 2 つの成果を期待できる。

1. 実トラフィックを考慮したネットワーク機器の設定ができる
   現在、ネットワークの再構成や機器設定は、ネットワーク管理者の予想や時間をかけて解析された情報に基づいて行われている。
   AGURI を使用することで、ネットワークを占有しているトラフィックの傾向が分かる。それにより、実トラフィックに基づいてネットワーク機器の設定を行うことができる。
2. ネットワークセキュリティ対策に応用できる
   現在、DoS (Denial of Service)[110] などの攻撃が増加しているが、実際に DoS 攻撃を受けているノード外での DoS 攻撃の検出は非常に困難である。
   AGURI を用いた場合、トラフィックの傾向なども即座に検出できるため、極端なトラフィックを被害者ノード以外から検出できる。それにより、DoS 攻撃を早期発見でき、被害を最小限に押さえられる。また、不審なトラフィックの増加も検出できるので、不正侵入なども事前に察知できる可能性をもっている。

AGURI は以下に示す 4 種類のネットワークサマリ情報を作成する。

- 送信元 IP アドレス
- 受信先 IP アドレス
- IP バージョン + プロトコル + 送信ポート番号
- IP バージョン + プロトコル + 受信ポート番号

この 4 種類のネットワークサマリを定期的に出力することによって、ある短時間のネットワーク状態の特徴を知ることができる。

更に、一度 AGURI で作成したネットワークサマリからもデータを入力することができ、複数のサマリを同時に入力することもできるので、ある短時間のサマリを組み合わせて AGURI に入力することによって、可変長の時間のネットワーク状態の特徴を知ることができる。

収集したパケットを蓄積し続けた場合、以下の 2 つの欠点が生じる。

1. フローの動的検知が困難になる
   木構造中に存在するノード数が増加するにつれて、計算の処理に時間がかかり、トラフィックの多いフローを抽出するまで時間がかかる。
   そのため、短期的なトラフィックモニタリングの効果を減少させる。
2. 保存するデータサイズが大きくなる
   木構造中に存在するノード数が増加するにつれて、保存するデータサイズ増大する。そのため、長期間に渡ってデータを蓄積することが難しくなる。

この問題を解決するために、AGURI では 1) パトリシア木アルゴリズムと、2) LRU (Least Recent Used) を用いた。

パケットを収集しツリー構造にデータを蓄積する時に、単純な 2 分木ではなくパトリシアツリーを用いることによってツリー構造全体のノード数を少な

くした。

更に、LRU を用いてツリー構造中のノード数を一定に保った。

この工夫によって、AGURI を短期から長期に渡って利用できる。

2.3 収集データ

WIDE プロジェクトでは以下に示す 3 地点において国際線のデータを収集している。

1. samplepoint1
   trans-Pacific line (18 Mbps CAR on 100 Mbps link)
2. samplepoint2
   US-Japan line (US side 60 Mbps POS)
3. samplepoint3
   US-Japan line (Japan side)

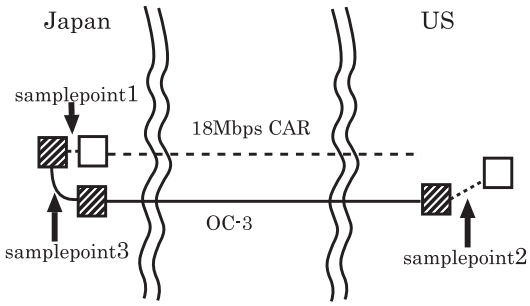


図 2.1. データ収集地点

WIDE プロジェクトで利用している 2 本の国際線のうち、1 本は他 AS と BGPpeer を張っているポイントを WIDE インターネットの入り口でデータ収集を行っている（samplepoint1）。

他の 1 本は WIDE の利用している国際線の US 側（samplepoint2）、日本側（samplepoint3）でそれぞれデータ収集を行っている。

後者の国際回線は 2002 年 1 月から利用しており、十分なデータ収集が行われていないため 2001 年度の報告書においては、前者の samplepoint1 のデータの長期的傾向を示す。

2.4 WIDE 国際線のデータ傾向

2001 年度の WIDE 国際線の年間トラフィック傾向を図 2.2 から図 2.17 に示す。これらの図は AGURI のスクリプト群を用いて視覚化したものである。

図の出力は時期を四半期ごとに、対象を 1) 送信元 IP アドレス、2) 宛先 IP アドレス、3) 送信元ポート番号、4) 宛先ポート番号とする（表 2.1）。

図中に出て来る “4:6:80” とは IP バージョンが 4、プロトコル番号が 6（つまり TCP）、送信元ポート番号が 80（つまり HTTP）ということを示している。

ここに示した図は 2 つの情報を持っている。

- 折れ線グラフ
  回線を占めているトラフィックの属性を視覚的に見ることができる。
  今回取り上げた WIDE インターネット国際線の例では、HTTP データの割合を把握できる。
- 項目
  折れ線グラフの下にリストアップされる項目数は、AGURI によって設定することができる。この項目は全トラフィック中の占有率順にリストアップされるのため、回線を使用している組織や使われているアプリケーションを検知することができる。
  送信元、宛先 IP アドレスからは、特定の組織の IP アドレス空間を検出できた。
  送信元ポート番号からは、特定のポートを使用したアプリケーションを検出できた。
  特に 2000 年度の WIDE 報告書と比較した場合、2000 年度に検出できた GNUTELLA のデータは消え、今年度は WinMX や Napster といった P2P 型ファイル交換アプリケーションのリスト交換データである’4:6:6699’（IPv4:tcp:port6699 番）を抽出することができた。
  また、ICMP’4:1:0’（IPv4:ICMP）も抽出することができた。詳細に調べてみると ICMP による DoS アタックが頻発したことによって ICMP

表 2.1. トラフィック傾向一覧表

	4 月– 6 月	7 月– 9 月	10 月– 12 月	1 月– 3 月
送信元 IP アドレス	図 2.2	図 2.3	図 2.4	図 2.5
宛先 IP アドレス	図 2.6	図 2.7	図 2.8	図 2.9
送信元ポート番号	図 2.10	図 2.11	図 2.12	図 2.13
宛先ポート番号	図 2.14	図 2.15	図 2.16	図 2.17

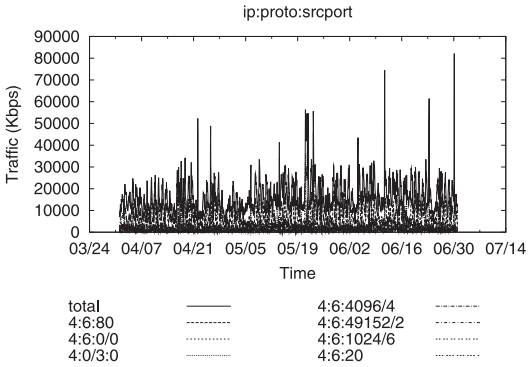


図 2.2.　送信元 IP アドレス（4 月– 6 月）

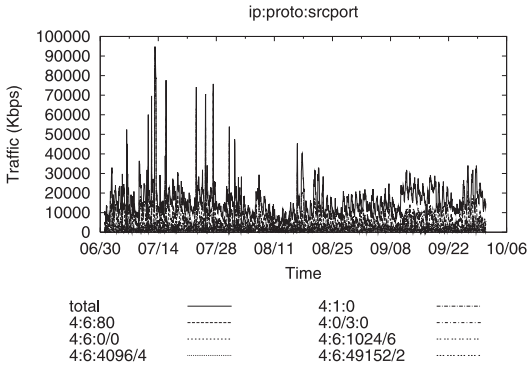


図 2.3.　送信元 IP アドレス (7 月– 9 月)

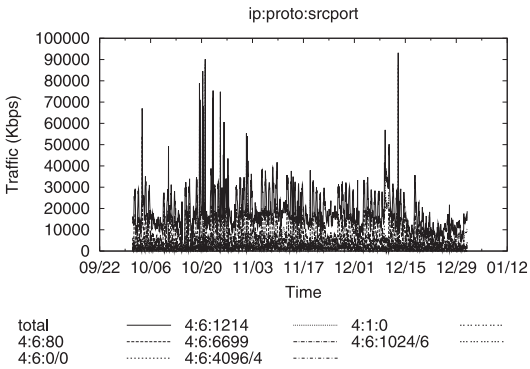


図 2.4.　送信元 IP アドレス (10 月– 12 月)

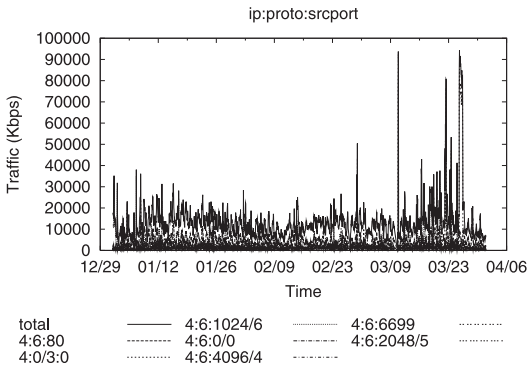


図 2.5.　送信元 IP アドレス (1 月– 3 月)

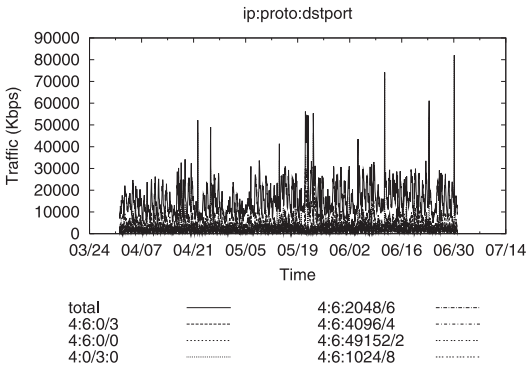


図 2.6.　宛先 IP アドレス (4 月– 6 月)

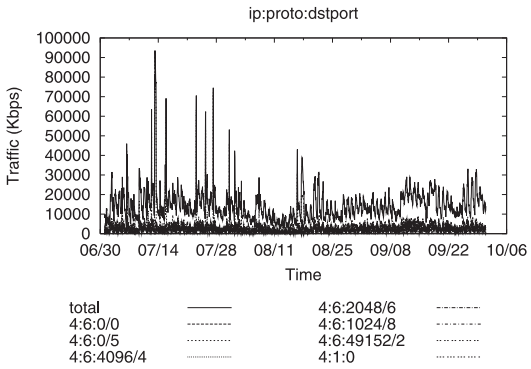


図 2.7.　宛先 IP アドレス (7 月– 9 月)

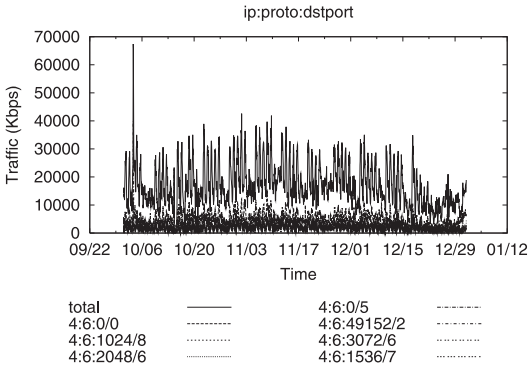


図 2.8.　宛先 IP アドレス (10 月– 12 月)

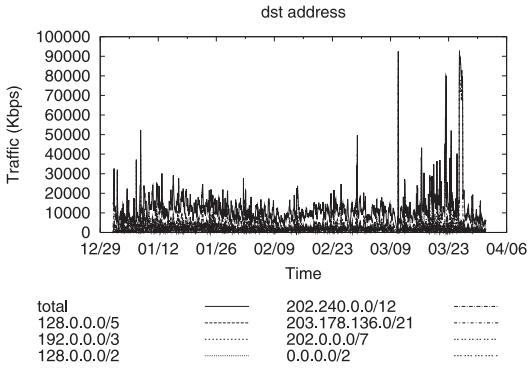


図 2.9.　宛先 IP アドレス (1 月– 3 月)

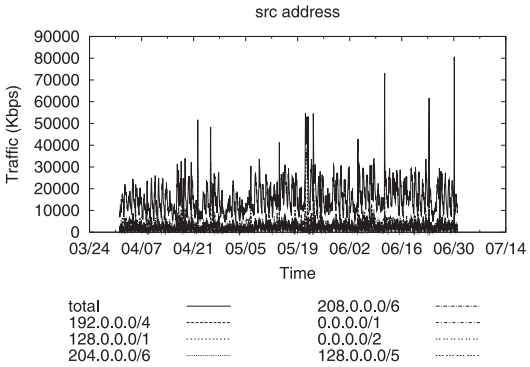


図 2.10. 送信元ポート番号 (4 月–6 月)

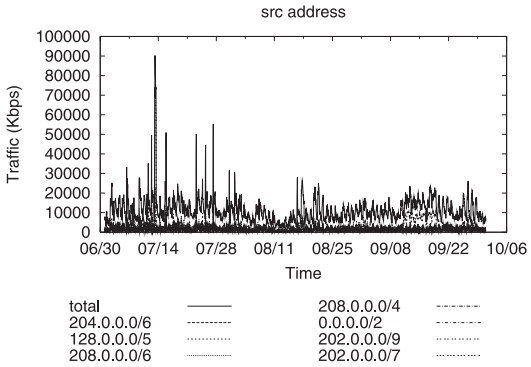


図 2.11. 送信元ポート番号 (7 月–9 月)

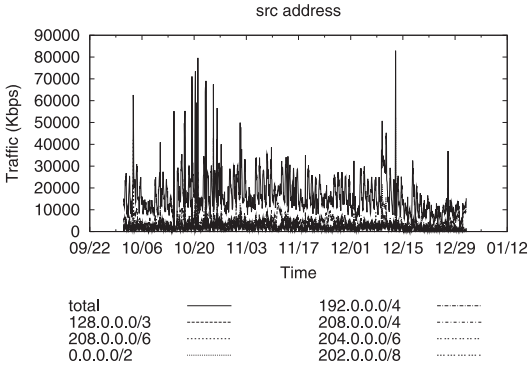


図 2.12. 送信元ポート番号 (10 月–12 月)

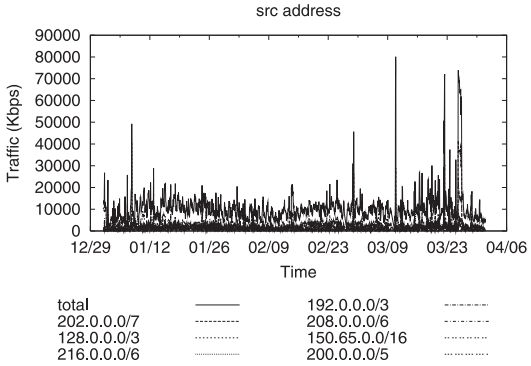


図 2.13. 送信元ポート番号 (1 月–3 月)

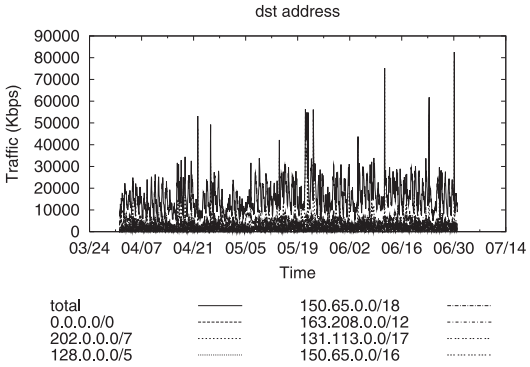


図 2.14. 宛先ポート番号 (4 月–6 月)

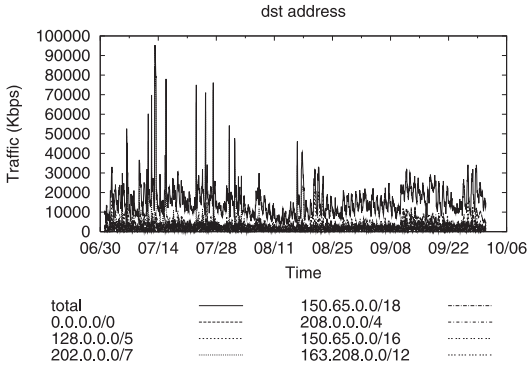


図 2.15. 宛先ポート番号 (7 月–9 月)

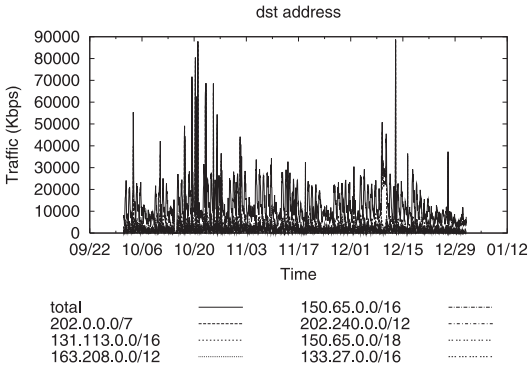


図 2.16. 宛先ポート番号 (10 月–12 月)

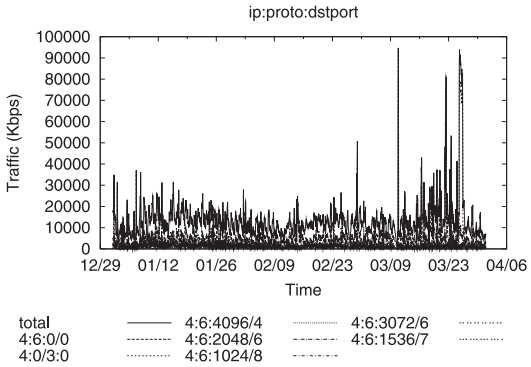


図 2.17. 宛先ポート番号 (1 月–3 月)

表 2.2.　識別された IP アドレス

graph	IP アドレス	hostname
図 2.5、2.6、2.7、2.8	150.65.0.0/16	jaist.ac.jp
図 2.6、2.8	131.113.0.0/17	keio.ac.jp
図 2.9	203.178.136.0/21	wide.ad.jp
図 2.8	133.27.0.0/16	keio.ac.jp

表 2.3.　識別されたポート番号

graph	ポート番号	プロトコル/アプリケーション
図 2.10、2.11、2.12、2.13	4:6:80	HTTP
図 2.11、2.12、2.11	4:1:0	ICMP
図 2.12、2.13	4:6:6699	WinMX[154]、Napster[107]

トラフィックが増大したことがわかった。

以上のように、折れ線グラフで表された情報とリストアップされた項目から、全トラフィックを構成している特徴的な要素を抽出することができた。

図 2.2 から図 2.17 に示された長期的トラフィック傾向から抽出できた情報を表 2.2、表 2.3 に示す。

2.5 結論

本報告書では、AGURI を用いた WIDE インターネット国際線のトラフィック傾向を述べた。

WIDE インターネットのような広域なネットワークを運用し続けていくためには、トラフィックモニタリングを多地点、かつ長期間行い、ネットワークの現状に適した通信機器の設置、設定を行う必要がある。

しかし、現存するネットワークモニタリングツールは長期に渡ってトラフィックの傾向を収集し続けることが難しい。

MAWI ワーキンググループでは収集したトラフィックを効果的に集約することによって、ネットワークの特徴を抽出することのできるトラフィックモニタリングツール AGURI を用い長期に渡る国際線のトラフィック傾向を明らかにした。

実際に AGURI を用いて WIDE インターネット国際線でデータを収集し、対象とした国際線のトラフィックの傾向を明らかにした。

WIDE プロジェクトでは、AGURI の開発をすすめると共に、WIDE インターネットのバックボーンにおいて AGURI を運用し続けている。これらのデータは、http://mawi.wide.ad.jp/ から参照可能である.

第 3 章　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

WWW によるさまざまなサービスが提供されるにつれ、大規模な WWW サーバの安定運用が重要となる。大規模な WWW サーバの安定運用にはサーバシステムの性能上限を知ることが必要であり、その調査にはベンチマークシステムを用いることが一般的である。しかし、既存のベンチマークシステムは LAN 環境でテストを行うため、現実の環境におけるネットワーク遅延が考慮されていない。そのため、ベンチマークシステムの結果が現実の環境を反映しないことがある。そこで、実際のネットワーク環境に近い実験環境を実現するため、複数のサーバ・クライアント間のコネクション毎に異なる遅延時間を挿入したベンチマークシステムを提案、実装する。そして、提案したシステムをパケットモニタで観測することにより、既存のベンチマークシステムとの比較を行う。

3.1 はじめに

インターネットの急速な普及に伴い、WWW を用いたさまざまなサービスが提供されるようになっている。たとえばオリンピックやワールドカップなどの中継、ホテルや航空券の予約や、ショッピングでの利用などである。このような WWW を利用したサービスが一般的になり、WWW が社会基盤の一つとして重要な役割が求められるようになるにつ

れ、ユーザは安定したサービス提供および品質の向上を要求するようになった。一方、サービス提供者は、ユーザの要求に応えるため、サービス品質の維持、向上に努めなければならない。WWW サーバの管理者は、WWW サーバの性能に影響する要因を把握し、そのうえでキャパシティプランニングを行う必要がある。キャパシティプランニングとは、サービスを安定してユーザに提供するために必要なハードウェアおよびソフトウェアの規模、性能を見積ることである。WWW サーバのキャパシティプランニングに対しては、

- 目的に応じたサービス提供に必要十分なサーバ性能の見積り
- ユーザ数およびリクエスト数の予測

といったことを考慮する必要がある。しかし、さらに具体的なキャパシティプランニングのためには、「このサーバシステムでは、1 秒間あたり何リクエストまでを処理することができる」ということを明確にする必要がある。つまり、サーバシステムの性能上限を把握することが重要となる。サーバシステムの上限を調査するためには、一般的にベンチマークシステムが用いられる。しかし、既存のベンチマークシステムでは、それらの示す結果が実際に運用されるサーバシステムに適用できないことがある。この原因として、

- ワークロード（ファイルの種類、頻度）が一定でない
- Common Gateway Interface (CGI) など、動的に変化するページの影響が考慮されていない
- ネットワーク遅延、帯域の影響が考慮されていない

ことが挙げられる。本研究では、これらの中でも特にネットワーク遅延に注目した。既存のベンチマークシステムは、LAN 環境でテストを行うことが一般的である。実際にサーバシステムが運用されるインターネット上のネットワーク遅延特性は LAN 環境の特性と異なり、この違いがベンチマークシステムの出力に影響していると考えられる。つまり、実際に運用されているサーバとクライアントである各ユーザとのネットワーク的距離が一定でないため、LAN 環境における一定のネットワーク遅延でのベンチマーク結果は、実際のサーバシステムの挙動を示すものにならないと考えられる。

本研究では、これらの問題を解決するため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を提案し、実装する。そして、ネットワーク遅延を考慮することにより、WWW サーバへの負荷がどのように変化するかを実験により明らかにし、提案システムの有効性を示す。

3.2 関連研究

これまでに WWW サーバのベンチマークシステムはいくつか開発されている。主なものを以下に挙げる。

WebStone

WebStone[137] は、要求するドキュメントサイズの分布、そのアクセス頻度やクライアント数などを設定することのできるフリーのベンチマークツールである。

SPECWeb

SPECWeb[33] は、商用のベンチマークツールであり、固定のプロファイリングでテストを行うことにより、統一的な指標を出すことができる。

httperf

httperf[103] は、WWW サーバの処理能力以上の HTTP/1.1 リクエストを生成し、その負荷を維持することができる。また、独自のワークロードジェネレータにより、さまざまな負荷を設定することができるベンチマークツールである。今回、我々の提案するベンチマークシステムとの比較で httperf を用いた。

Web Polygraph

Web Polygraph[115] は、プロキシサーバ用のベンチマークシステムとして開発されたものであり、専用のクライアントとサーバを用いてその間のプロキシサーバの性能を測定するシステムである。

このようにさまざまな特徴を持つベンチマークシステムが開発されているが、これらのシステムは、一般的に LAN 環境でテストを行うため、その結果が実際に運用される WWW サーバに適用できない可能性がある。

3.3 現実のネットワーク遅延

我々は、既存の WWW サーバベンチマークシステムが現実の環境に適用できない主な原因としてネットワーク遅延に注目した。図 3.1 は、第 83 回全国高等学校野球選手権大会のインターネット中継に用い

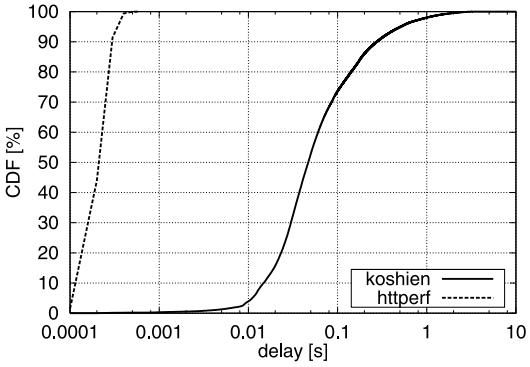


図 3.1.　遅延時間の累積頻度分布

られた WWW サーバにおける遅延時間（実線）と、LAN 環境で httperf を用いた WWW サーバのベンチマークテストの実行時における遅延時間（破線）の累積頻度分布である。この遅延時間は ENMA[164] を用いて計測した。ENMA とは、パケットモニタを用いた WWW サーバ性能評価ツールである。ENMA ではリアルタイムで HTTP リクエスト到着レート、HTTP セッション数、コネクション到着レート、トラフィック, パケット数などを観測することができる。また、TCP コネクションログや HTTP リクエストログを解析することにより、さまざまな性能指標を調査することが可能なツールである。ここでの遅延時間は、ENMA のコネクションログに記録される図 3.2 における (3) – (2) の時間としている。これはサーバ・クライアント間のラウンドトリップタイム（RTT）に相当する時間である。

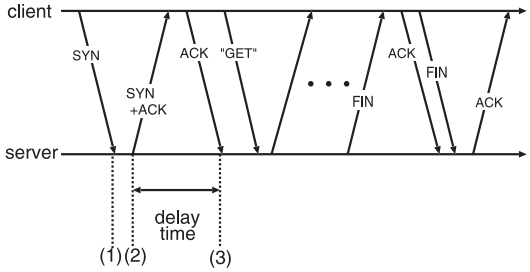


図 3.2.　遅延時間の定義

サーバ・クライアント間にルータおよびスイッチだけを配置した LAN 環境では、1 ms 以下の遅延しか発生していない。それに対し、実際に運用されている WWW サーバにおける遅延時間は、ほとんどが 1 ms 以上であり、幅広く分布し、LAN 環境での遅延時間の特性とは大きく異なることがわかる。我々は、この遅延時間の特性の違いがベンチマークの結果に影響を与えると考えた。

3.4 提案するベンチマークシステム

前章で述べたように、既存のベンチマークシステムは LAN 環境で実験を行うため、現実の環境での WWW サーバおよびクライアントの挙動とは異なったものとなる。テスト用に構築した LAN 環境では、現実のネットワーク環境をシミュレーションすることは困難である。

この問題を解決するため、現実に近いネットワーク環境を実現する WWW サーバベンチマークシステムを提案する。これは、複数の仮想クライアントから HTTP リクエストを送信し、サーバからのレスポンスを受信するリクエストジェネレータおよび、ネットワーク遅延を実現する Dummynet[125] から構成される。図 3.3 に提案するベンチマークシステムの構成を示す。

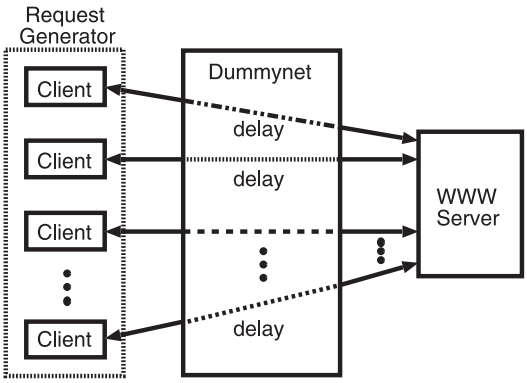


図 3.3.　提案するベンチマークシステムの構成

3.4.1 リクエストジェネレータ

Dummynet を IP アドレスベースで遅延時間の設定を行う方法で用いるため、各コネクション毎に異なる遅延時間を実現するためには、複数の IP アドレスから、リクエストを送信しなければならない。そこで、alias 機能で実現した複数の IP アドレスから、リクエストを送信できるように実装した。これにより、複数の IP アドレスからの HTTP リクエストのそれぞれに異なる遅延時間を設定することが可能となる。また、複数のクライアントをシミュレートするために多数の計算機を用いる必要もなくなる。

リクエストジェネレータの実行時には、WWW サーバへの各々のリクエストに対して、(1) コネクション確立要求開始時刻、(2) HTTP リクエスト送信時刻、(3) コネクション終了時刻をアプリケーションレベルで記録し（図 3.4）、実行終了時に統計情報を出力する。

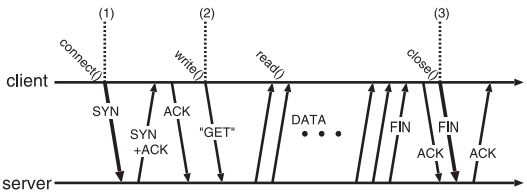


図 3.4. リクエストジェネレータが記録する時間

この (1)〜(3) の時間を記録することにより、WWW サーバの性能指標の一つである RTT（(3) – (1)）、コネクション確立に要する時間（(2) – (1)）を得ることができる。これらの時間の測定には、UNIX の `gettimeofday()` を用いた。`gettimeofday()` は、システムクロックの値を返し、少なくともミリ秒の精度を持つ。

3.4.2 ネットワーク遅延の実現

ネットワーク遅延は Dummynet を用いて実現する。Dummynet は、ネットワークプロトコルのテストのために開発されたツールであり、帯域制御を実現するツールとしても用いられている。Dummynet により、キューと帯域幅の制限、パケットの遅延やパケットロスのシミュレーションができる。

実装は FreeBSD 上で行われており、この機能を用いるためには、Dummynet を組み込んだカーネルを構築する。Dummynet はルータもしくはブリッジとして用いることができ、`ipfw` コマンドを利用して遅延の設定を行う。

提案システムでは、リクエストジェネレータから送信するリクエストがすべて Dummynet を通過するように Dummynet をルータとして用いる。そして、Dummynet の IP アドレス毎に異なる遅延を入れる機能により、コネクション毎に異なる遅延時間の挿入を実現した。

3.5 提案システムの評価

提案するシステムでは、これまでのベンチマークシステムでは考慮されていなかった複数のネットワーク遅延を含めてベンチマークを行うことが可能となった。まず、提案システムで実際のネットワーク遅延を再現できることを 2 種類の実測データを用いて確認する。次に、実験 1 で WWW サーバへの負荷が低い場合において、ネットワーク遅延を考慮することにより WWW サーバへの負荷がどのように変化するかを既存のベンチマークシステムとの比較により確認する。さらに、実験 2 では WWW サーバへの負荷が高い場合に、遅延設定をしない場合と遅延設定を行った場合でリクエストジェネレータの同時コネクション数を変化させ、WWW サーバにどの程度の負荷を与えることができるかを確認する。

3.5.1 観測システム

WWW サーバのネットワーク観測には ENMA[164] を用いる。また、WWW サーバ内部の性能調査には rep2[106] を用いる。これは、計測とログ出力を異なる計算機で実行できるカーネルモニタリングシステムであり、計測によってサーバシステムの負荷をほとんど与えないという特徴がある。

観測対象の WWW サーバ用計算機内でカーネルモニタ rep2 を実行する。ENMA は観測するサーバ計算機をスイッチングハブのポートミラー機能を用いたパケットのコピーにより計測を行う。ネットワークモニタリング用の計算機では、enma デーモンとそのリアルタイムデータの転送プログラム reportor を実行する。視覚化モジュール用計算機では、ENMA のリアルタイムデータ受信プログラム collector を実行し、ENMA のリアルタイムデータを収集する。収集されたリアルタイムデータは、ggraph でグラフ化される。各計算機の rep2 のデータおよび ENMA のリアルタイムデータの転送には、HTTP リクエストおよびレスポンスが流れるネットワークとは別に用意したコントロール用のネットワークを利用する。これにより、実験環境に影響を与えずに計測を行うことが可能である。

3.5.2 ネットワーク構成

実験 1 および実験 2 ともに図 3.5 に示すネットワーク構成で実験を行った。クライアント用計算機は、スイッチングハブを介してルータとして設定した Dummynet と接続されている。クライアントのネットワークインターフェースは 100BASE-TX で

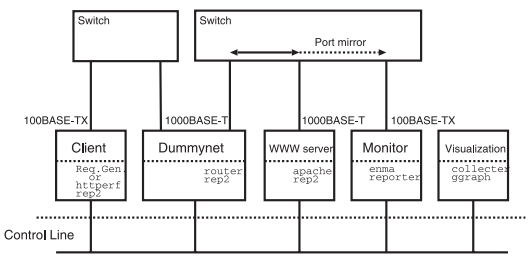


図 3.5. ネットワーク構成

ある。さらに Dummynet は、スイッチングハブを介して WWW サーバ用計算機に接続されている。Dummynet および WWW サーバのネットワークインターフェースは、1000BASE-T である。WWW サーバが接続されているスイッチングハブのポートは、ポートミラーの機能により、モニタリング用の計算機でも受信可能となっている。計測用計算機は、パケットの取りこぼしを避けるため観測対象の計算機と同等の性能の計算機を用いた。さらに計測データの視覚化のための計算機がコントロール用ネットワークに接続されている。上記の計算機群はすべてコントロール用ネットワークに接続されている。

3.5.3 遅延特性の評価

提案システムで、実際のネットワーク遅延特性が再現できることを確認するために以下の 2 種類の実測結果を用いて評価を行った。

- 第 81 回全国高等学校野球選手権大会
- 第 83 回全国高等学校野球選手権大会

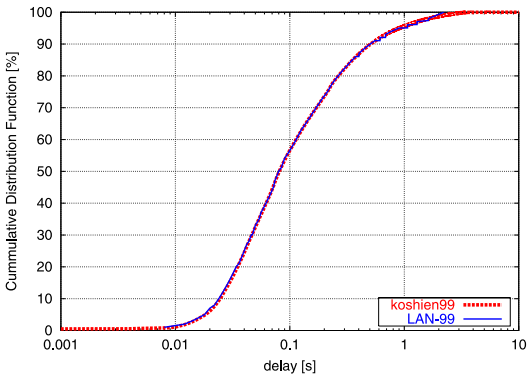
LAN 環境での遅延時間は、実測結果の累積頻度分布を 100 等分したものを、クライアントの IP アドレスにそれぞれ対応させて設定した。第 81 回全国高等学校野球選手権大会の実測および LAN 環境における提案システムによる遅延時間の累積頻度分布を図 3.6(a) に示す。第 83 回全国高等学校野球選手権大会の実測および LAN 環境における提案システムによる遅延時間の累積頻度分布を図 3.6(b) に示す。

それぞれの結果ともに、実測の遅延特性と LAN 環境における提案システムによる遅延特性はほぼ同一の分布となっている。この結果より、提案システムを用いることによって、LAN 環境において実測結果をもとにした実際のネットワーク遅延特性を再現することが可能であることが確認できた。

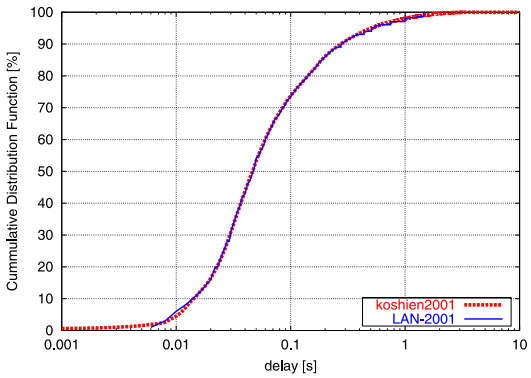
3.5.4 実験 1: 低負荷時の WWW サーバの測定

実験 1 では、低負荷時における WWW サーバの挙動について検討する。本実験では、提案システムと既存のベンチマークシステムを用いて、ネットワーク遅延を考慮することにより、WWW サーバへの負荷がどのように変化するかを明らかにする。既存のベンチマークシステムとしては一般によく用いられている httperf を用いた。

ネットワーク遅延を考慮することにより、1 つの HTTP セッションの継続時間が長くなる。そのた



(a) 第 81 回全国高等学校野球選手権大会



(b) 第 83 回全国高等学校野球選手権大会

図 3.6.　実測と提案システムによる遅延時間の累積頻度分布

表 3.1.　実験 1 の計算機環境

	WWW server	Client	Dummynet
CPU	PentiumIII 800 MHz	PentiumIII 800 MHz	PentiumIII 1 GHz × 2
Memory	512 MB	512 MB	512 MB
OS	FreeBSD 4.3-RELEASE	FreeBSD 4.3-RELEASE	FreeBSD 4.3-RELEASE
Kernel 設定	maxsocket = 8232 maxuser = 256	default maxuser = 32	HZ = 1000 maxuser = 512

表 3.2. 実験 1 の測定条件

	ベンチマーク	リクエストレート	遅延設定
実験 1-1	提案システム	110 req/s	遅延なし
実験 1-2	提案システム	110 req/s	一定遅延（往復 48 ms）
実験 1-3	httperf	110 req/s	遅延なし
実験 1-4	httperf	110 req/s	一定遅延（往復 48 ms）
実験 1-5	提案システム	110 req/s	複数遅延

め、同じリクエストレートで WWW サーバにリクエストを送信すると、遅延を考慮しない場合に比べて、WWW サーバ内で処理中のリクエストが増加すると予想される。そこで本実験では、性能指標として、WWW サーバ内で処理中のコネクション数に着目した。

実験環境

実験で用いた計算機環境を表 3.1 に示す。OS はすべて FreeBSD 4.3-RELEASE である。WWW サーバソフトウェアには、Apache[51] (Ver.1.3.20) を利用し、その設定はすべてデフォルトとした。また、WWW サーバに用いた計算機のカーネル設定は、本実験で要求するリクエストレートに対応するため最大ソケット数（kern.ipc.maxsockets）を増加させたものを用いた。クライアント側のカーネル設定はデフォルトのままとした。Dummynet に用いた計算機は、遅延設定の精度を高めるためにカーネルの HZ 値をデフォルト値（100）の 10 倍にしたものを用いた。HZ 値はクロック割り込みの発生間隔を決めるカーネルパラメータであり、HZ = 1000 とすることで、1 ms 間隔でクロック割り込みを発生させることができる。

実験条件

表 3.2 に示すそれぞれの条件のもとで、クライアントから 110 リクエスト/秒のリクエストレートで合計 100,000 リクエスト（約 15 分間）を送信した。要求したファイルのサイズは 10 KB である。遅延時間は 3.3 での実測結果を用いた。一定遅延の場合は実測結果の中央値である往復 48 ms を、複数遅延の場合は累積頻度分布における遅延時間を 100 段階に分割し、100 のクライアントにそれぞれに対応した遅延時間を設定した。そして、この時の WWW サーバを ENMA で計測した。

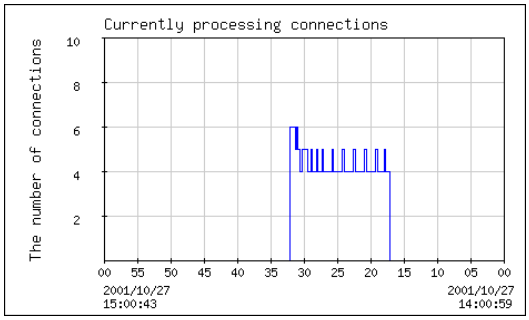
結果: 処理中のコネクション数の変化

図 3.7(a)〜(e) に表 3.2 に示した各条件で WWW サーバにリクエストを送信したときの、サーバ内で処理中のコネクション数のグラフを示す。

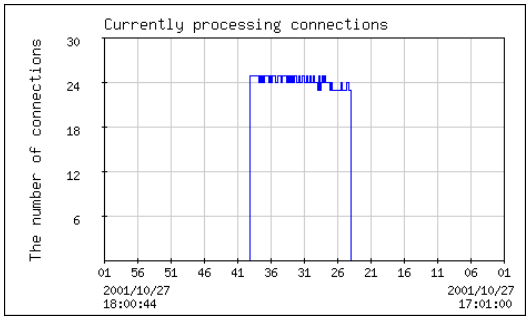
図 3.7(a) および図 3.7(c) は、遅延時間を設定せずに実験した時の結果で、リクエストが送信されている時間帯で 1〜6 とサーバ内で処理中のコネクションは少ない。図 3.7(b) および図 3.7(d) は、すべてのコネクションに対して一定の遅延を挿入した時の結果であり、20〜25 程度のコネクションを処理しており、遅延を設定しない場合に比べて増加している。図 3.7(e) は複数遅延を挿入した場合の結果であり、遅延設定をしない場合と一定遅延を挿入した場合にくらべて、処理中のコネクションがさらに増加し 60 程度となっていることがわかる。

考察

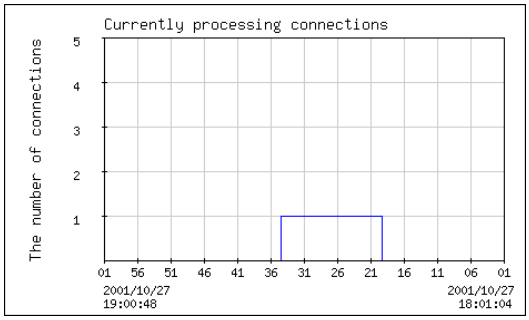
実験 1 において、WWW サーバへの負荷が比較的低い状態で、ネットワーク遅延を考慮することによる WWW サーバ内の処理中のコネクション数がどのように変化するかを確認した。文献 [105] において、WWW サーバが飽和状態になると、処理中のコネクションが増加し、WWW サーバのパフォーマンスに影響を与えることが報告されている。ネットワーク遅延がほとんどない LAN 環境では、WWW サーバがクライアントから次々に送られてくるリクエストの処理を完了する時間が短い。そのため、WWW サーバ内で処理中のコネクションは増加しない（実験 1-1、1-3）。一方、一定のネットワーク遅延を挿入した場合、リクエスト処理が完了するまでに必要な時間がネットワーク遅延の分だけ長くなる。そのため、ネットワーク遅延がほとんどない環境に比べると WWW サーバ内で処理中のコネクションを多く保持することになる（実験 1-2、1-4）。実測結果に基づく遅延を挿入した実験 1-5 においては、一定の遅延を挿入した場合に比べて約 2 倍のコネクション数となった。これは一定遅延の場合に比べ、2 倍程度サーバに負荷を与えていることになる。したがって、現実の環境と LAN 環境では、同程度のリクエストレートでもサーバに対する負荷の与え方が異なるこ



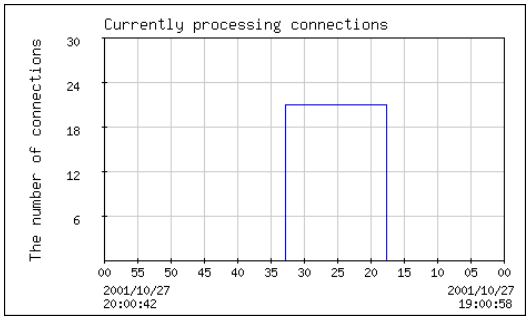
(a) 提案システム、遅延なし



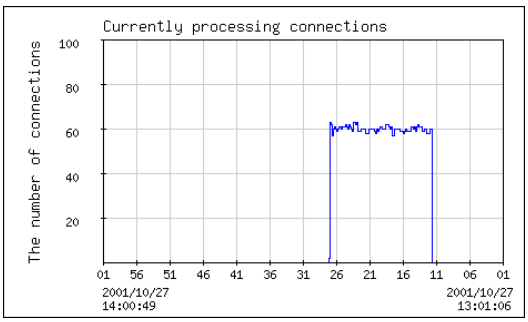
(b) 提案システム、一定遅延



(c) httperf、遅延なし



(d) httperf、一定遅延



(e) 提案システム、複数遅延

図 3.7.　WWW サーバ内で処理中のコネクション

とがわかる。

3.5.5 実験 2: 高負荷時の WWW サーバの測定

実験 1 では、リクエストレートを秒間 110 リクエストとし、WWW サーバへの負荷がそれほど高くない状態でネットワーク遅延が WWW サーバへ与える負荷を確認した。次に、実験 2 では WWW サーバへ要求するリクエストレートを高くし、WWW サーバが高負荷になった場合の挙動を検討する。ここでは WWW サーバへ高負荷がかかるようなリクエストを実現するために、クライアント側で同時コネクション数を固定してリクエストを送信した。

実際に運用されている WWW サーバにアクセスが集中して負荷が増加した場合、サーバへのアクセスが拒否されることが予想される。そのため実験 2 では、コネクションリセット率に着目し実験を行った。

実験環境

実験 2 で用いた計算機環境を表 3.3 に示す。実験 1 と同様に OS はすべて FreeBSD 4.3-RELEASE である。WWW サーバソフトウェアには、Apache[51] (Ver.1.3.20) を利用し、その設定はすべてデフォルトとした。実験 2 ではリクエストレートを高くするため、大量のソケットを扱えるようにする必要がある。そのためサーバ、クライアントともにカーネル設定の MAXUSER 値を 2048 とした。これにより、最大ソケット数（kern.ipc.maxsockets）の値がデフォルトの 1,064 から 65,576 へ増加する。

実験条件

実験 2-1、2-2 では、同時コネクション数を 1、10、50、100、200、500、800、900、1000、1100、1200

表 3.3. 実験 2 の計算機環境

	WWW server	Request Generator	Dummynet
CPU	PentiumIII 800 MHz	PentiumIII 800 MHz	PentiumIII 1 GHz × 2
Memory	512 MB	512 MB	512 MB
OS	FreeBSD 4.3-RELEASE	FreeBSD 4.3-RELEASE	FreeBSD 4.3-RELEASE
Kernel 設定	maxuser = 2048	HZ = 1000 maxuser = 2048	HZ = 1000 maxuser = 2048

表 3.4. 実験 2 の測定条件

	遅延	ファイルサイズ	同時コネクション数	時間
実験 2-1	なし	2 KB	1、10、50、100、200、500、800、900 1000、1100、1200、1500、2000	各 3 分間
実験 2-2	複数遅延	2 KB	1、10、50、100、200、500、800、900 1000、1100、1200、1500、2000	各 3 分間

と変化させ、2 KB のファイルをリクエストを 3 分間要求した。遅延設定をしない場合と複数遅延を入れた場合で実験し、ネットワーク遅延を考慮することにより、WWW サーバへの負荷がどのように変化するかを検討する。ここでの複数遅延は実験 1 と同様に、3.3 での実測結果の累積頻度分布における遅延時間を 100 段階に分割し、Dummynet で異なる 100 の IP アドレスの通信のそれぞれに対応する遅延時間を設定した。

結果: 実験 2-1、遅延なし

図 3.8(a)〜(d) に遅延設定をせずに、同時コネクション数を変化させた時の ENMA による観測結果を示す。またその時の WWW サーバとクライアントの rep2 による CPU 使用率をそれぞれ図 3.9(a)、(b) に示す。各図において、同時コネクション数 1 の実験が 19:04 付近から始まり、各実験の間隔は 1 分であり、同時コネクション数 2000 の実験の終了は 19:55 付近である。

図 3.8(a) は WWW サーバ内で処理中のコネクション数を示し、実験条件で示したように、1、10、100、...、2000 とコネクション数が変化していることがわかる。

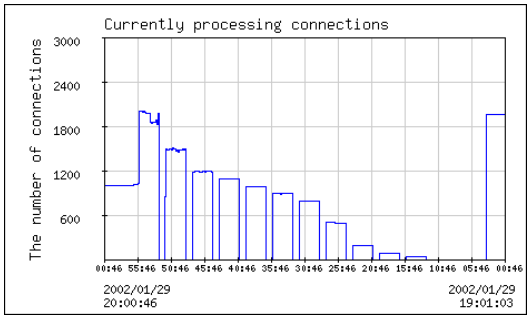
図 3.8(b) および (c) は、それぞれ WWW サーバのコネクション到着レート、HTTP リクエスト到着・応答レートを示し、同時コネクション数が 1 の場合は、約 600 リクエスト/秒のコネクションが到着し、それに対してサーバも同じレートで返答していることがわかる。同時コネクション数が 1 の場合は、一つのコネクションの処理が終るまで次のコネクションを要求しない。そのため、同時コネクション数が 1 より大きい場合に比べてリクエストレートが下がっている。同時コネクション数が 10〜1200 までは、ほぼ 1700 リクエスト/ 秒で一定のリクエストレートが出ている。同時コネクション数 1500、2000 の場合、それ以下のコネクション数の場合に比べ下がっている。

図 3.8(d) はコネクション処理レートを示している。同時コネクション数が 900 以上の場合に、サーバリセット（S_RESET）が発生し、HTTP/1.0 の正常終了状態であるサーバからのコネクションクローズ（S_NORMAL）が到着コネクションより少なくなっていることがわかる。

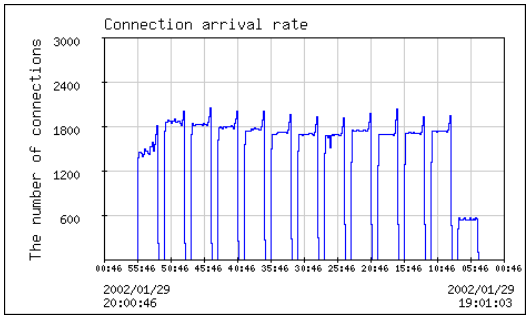
図 3.9(a) は WWW サーバの CPU 使用率であり、同時コネクション数が 10〜1200 の間はアイドル率が 0%となっている。図 3.9(b) はベンチマークシステムを実行しているクライアントの CPU 使用率を示している。同時コネクションが 500 以上の場合でアイドル率が 0%となっている。また同時コネクションが 1500 と 2000 の場合、カーネルの CPU 使用率が 85%以上となっており、クライアント側がかなり高負荷な状態になっていることがわかる。

結果: 実験 2-2、複数遅延

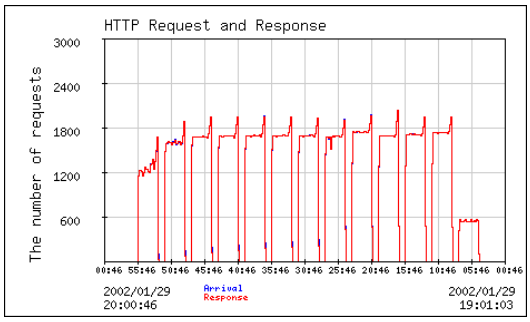
図 3.10(a)〜(d) に遅延を挿入し、同時コネクション数を変化させた時の ENMA の観測結果を示す。



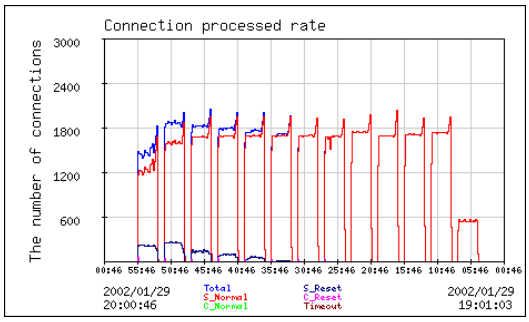
(a) 処理中のコネクション数



(b) コネクション到着レート

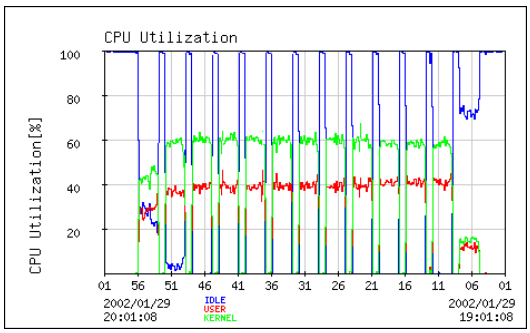


(c) HTTP リクエスト到着・応答レート

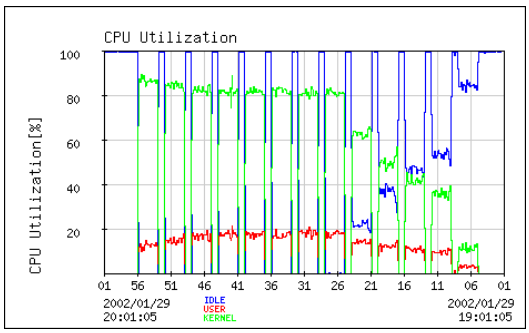


(d) コネクション処理レート

図 3.8.　ENMA による WWW サーバの観測結果 (実験 2-1)



(a) WWW サーバの CPU 使用率



(b) クライアントの CPU 使用率

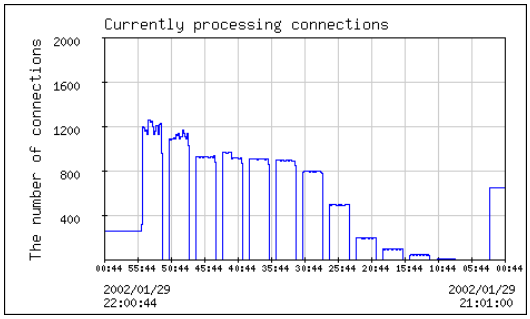
図 3.9.　CPU 使用率（実験 2-1）

またその時の WWW サーバとクライアントの rep2 による CPU 使用率をそれぞれ図 3.11(a)、(b) に示す。各図において、同時コネクション数 1 の実験が 21:04 付近から始まり、各実験の間隔は 1 分であり、同時コネクション数 2000 の実験の終了は 21:55 付近である。

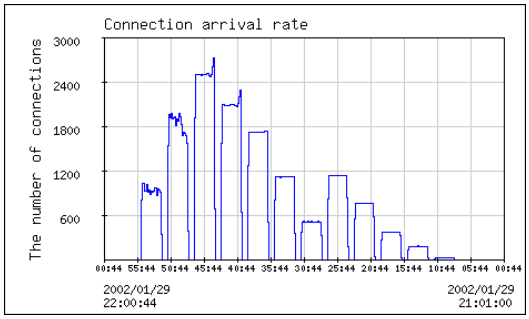
図 3.10(a) は WWW サーバ内で処理中のコネクション数を示している。同時コネクション数が 900 の場合までは、同時コネクション数に比例して処理中のコネクションが増加していることがわかる。900 以上の場合、指定したコネクション数より低くなっている。

図 3.10(b) は、WWW サーバのコネクション到着レートを示し、同時コネクション数が 1200 の場合までは同時コネクションにほぼ比例してコネクション到着レートが増加している。図 3.10(c) は HTTP リクエスト到着・応答レートを示し、同時コネクションが 1000 の場合に約 1600 リクエスト/秒のリクエストレートとなっている。図 3.10(d) はコネクション処理レートを示し、同時コネクション数が 1000 以上の場合で、サーバリセットが発生し、同時コネクション数が増加するに従い、サーバリセットの割合が増加している。

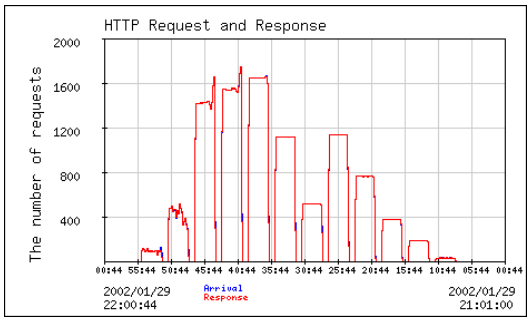
図 3.11(a) は WWW サーバの CPU 使用率である。カーネルの使用率がコネクション到着レートにほぼ比例して増加していることがわかる。図 3.11(b)



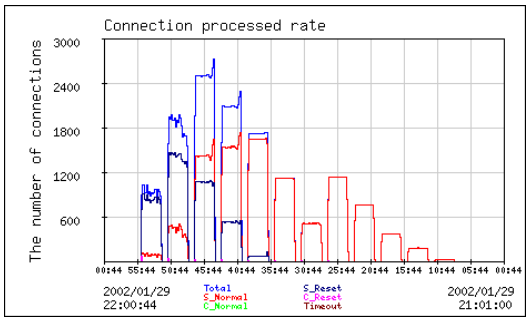
(a) 処理中のコネクション数



(b) コネクション到着レート

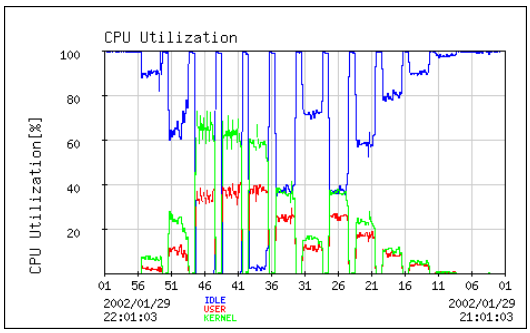


(c) HTTP リクエスト到着・応答レート

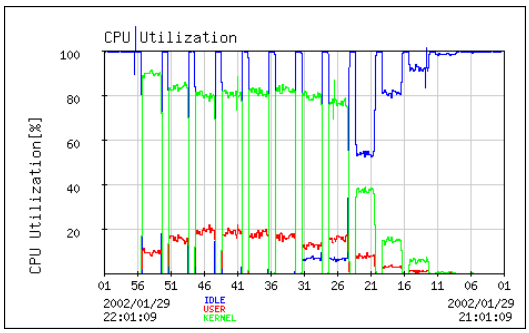


(d) コネクション処理レート

図 3.10. ENMA による WWW サーバの観測結果 (実験 2-2)



(a) WWW サーバの CPU 使用率



(b) クライアントの CPU 使用率

図 3.11. CPU 使用率 (実験 2-2)

はクライアントの CPU 使用率で、同時コネクション数 1200 以上の場合、カーネルの CPU 使用率が 85%以上となっており、遅延がない場合と同様にクライアント側がかなり高負荷な状態になっていることがわかる。

考察

実験 2 では、クライアントの同時コネクション数を固定して WWW サーバに高い負荷をかけ、高負荷時における WWW サーバの計測を通して、ネットワーク遅延を考慮することによる影響を検証した。

まず、ネットワーク遅延を考慮しない場合（実験 2-1）において、同時コネクション数が 10〜1200 の場合、WWW サーバのコネクション到着レートは、同時コネクション数に関係なくほぼ一定である。HTTP リクエストレートも同様に一定となっている。同時コネクション数が 2000 の時、コネクション到着レートが減少している。そして、この時のクライアントのカーネルの CPU 使用率は、90%近くになっている。つまり、クライアントに負荷がかかりすぎ、コネクションレートを保てなくったものと考えられる。

実験 2-1 および実験 2-2 の双方で同時コネクション数が 1000 以上になるとコネクション処理レートのサーバリセットが増加している。これは、クライアントとサーバ間のコネクションが、TCP のスリーウェイハンドシェイクによって確立がされた後に、サーバからのリセットによりコネクションが切断さ

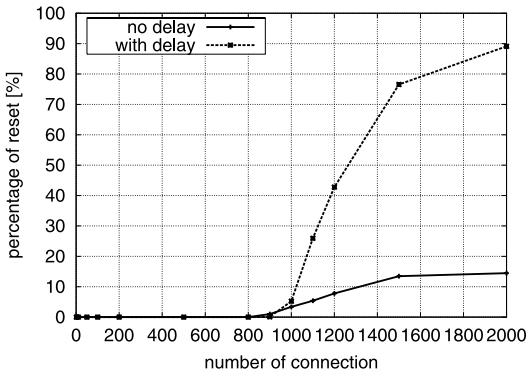


図 3.12.　サーバリセット率

れる現象の結果である。WWW サーバソフトウェア内では、複数のコネクションの処理を `select()` によって切替える。同時コネクション数が大きくなり、一度に扱うコネクション数が多くなると、このコネクション処理を行う `select()` がハンドリングを行えなくなり、サーバ側からリセットしてしまうものと考えられる。

図 3.12 は、サーバのコネクションリセット率を示したものである。ネットワーク遅延を考慮しない場合、サーバリセット率は、20%以内である。しかし、ネットワーク遅延を挿入した場合、同時コネクション数が 1000 以上になると急激に増加し、それにともないリクエストレートも低下していると考えられる。

3.6 おわりに

WWW はインターネットの発展の原動力となり、今後さらに社会基盤の一つとして重要な役割を求められる。WWW システムの管理者の立場では、WWW システムを安定に運用し、ユーザにとって快適なサービスを提供することが重要となる。そのためには、WWW サーバの性能限界を知ることが重要であり、その調査のためにベンチマークシステムが用いられてきた。既存のベンチマークシステムは、ネットワーク遅延がほとんどない LAN 環境でテストを行い、その結果を WWW サーバの性能としていた。しかし、実際にインターネットで運用されている WWW サーバとクライアント間には、ネットワーク遅延が存在する。そのため、実際に運用されている WWW サーバで、既存のベンチマークシステムが出力した性能を発揮しないことがある。

そこで本研究では、実際に運用されている大規模 WWW サーバの計測を行い、LAN 環境との遅延時間特性の違いを明らかにした。次に複数のネットワーク遅延を設定することが可能な WWW サーバベンチマークシステムを構築した。そして、提案システムの評価として、低負荷時と高負荷時にわけて WWW サーバのベンチマークを行い、ネットワーク遅延を考慮することにより、WWW サーバへの負荷が増加することが確認できた。

今後、提案システムを用いて運用前の大規模サーバシステムのベンチマークを行い、そのシステムが実際に運用する時の性能を表すことができているかを検討する必要がある。