透過的クライアント監視層によるネットワークシステムの性能評価システムの開発

斉藤　裕樹[†a)]　　中所　武司[†]

Development of Performance Evaluation System for Network Systems by Client Observation

Hiroki SAITO[†a)] and Takeshi CHUSHO[†]

あらまし　インターネット技術の普及に伴い，サーバへのアクセス時間や応答性能といった利用者の利便性を確保するための性能面でのネットワークサービスの品質管理が求められている．従来の性能評価はネットワーク層の機器の管理を目的としたものであり，アプリケーション層での品質を考慮したものではない．本論文では，利用者の利便性の観点でのサービス品質を評価するため，終点アプリケーションでの性能評価方式を提案する．この方式は，クライアントアプリケーションの振舞いの観測を行う透過的監視層を導入し，クライアントとサーバの間でやり取りされる要求–応答単位の通信に注目することで，性能の計測を可能にするものである．更に，透過的監視層の原理を用いた性能評価システムを実装し，稼働状態のシステムの性能を正確に測定できることを示し有効性を確認する．

キーワード　インターネット，ネットワーク性能，性能評価，性能管理

1. まえがき

インターネットの発展とその技術を用いた企業，大学などのネットワークの増加と規模の拡大に伴い，オンライン商取引などのエレクトロニック・コマース分野での応用や企業の基幹業務システムなど，社会生活に重要な役割を担うネットワークサービスが次々実現されつつある．

一方で，トラヒックの増加によってネットワークやホストが混雑することにより，WWW ページが表示されるのに時間がかかる，サーバが応答しないといった利用者の利便性を損なうような品質上の問題が顕在化している [1]．

従来，インターネット技術を用いたネットワークシステムの品質評価のために，端末間の到達性の保証やリンクの性能確保を目的とした性能評価技術が開発されてきた [2], [3]．しかし，従来の評価技術は主にネットワーク層の機器の管理を目的としたものであり，アプリケーション層での品質指標ではない．利用者の利便性確保を目的としたネットワークサービスの品質評価のためには，終点アプリケーションでの性能評価が求められる．

アプリケーション層でのネットワークシステムの性能には，帯域，ルータの処理能力，端末間のスループット，サーバシステムやクライアントシステムの性能などの多くの要因があり [4]，また，主なネットワークサービスで用いられている TCP の性能はネットワーク層やアプリケーション層の特性に左右されることが知られている [5], [6]．このため，これまでネットワークサービスの性能評価には，ネットワーク管理者の技術的な知識，勘や経験などが必要とされてきた．アプリケーション利用時のアクセス速度，応答速度，データ転送速度といったユーザの直接感じるネットワークサービスの性能を，勘や経験に頼らず客観的に評価できるシステムが望まれている．

本論文では，クライアントアプリケーションの振舞いを観測する監視層を導入しネットワークシステムの性能評価を行う方式を提案する．次に，監視層を用い

† 明治大学大学院理工学研究科基礎理工学専攻情報科学系，川崎市 Computer Science Course, Major in Sciences, Graduate School of Science and Technology, Meiji University, 1-1-1 Higashi-mita, Taka-ku, Kawasaki-shi, 214–8571 Japan
a) E-mail: hiro-s@cs.meiji.ac.jp

た性能評価システムの設計と実装について述べる．更に，性能評価システムを実際のネットワークシステムに適用し有効性を検証する．

2. ネットワークシステムの性能評価

2.1 ネットワークシステムの品質

利用者の視点でのネットワークシステムの品質とは，クライアントでの性能，すなわち終点アプリケーションでの性能である．本論文では終点アプリケーションの性能を評価するための汎用的な枠組みについて検討する．

従来のネットワークの性能評価においては，一方の端末から通信相手の端末に対して一方的に情報を転送する場合の性能について考えられてきた．しかしながら，多くのインターネット上のアプリケーションはクライアントから情報送信後サーバから応答情報を求めるような対話型アプリケーションであるため，応答時間を評価することが重要であると考えられる．具体的には，応答時間とはクライアントがリクエストメッセージを送信後サーバからレスポンスメッセージを受信するまでの時間である．

更に，応答時間は次の 3 項目の処理時間の和とみなすことができる．

（1） クライアントからサーバへのリクエストメッセージ転送時間

（2） サーバでのリクエストメッセージ処理時間

（3） サーバからクライアントへのレスポンスメッセージ転送時間

(1) はクライアントからサーバへの通信路の性能を表す指標であり,(2) はサーバアプリケーションの処理性能を示す指標であり，また (3) はサーバからクライアントへの通信路の性能を表す指標である．

したがって，応答時間と（1）～（3）の指標を計測することで，ネットワークシステムの性能とシステムを構成するリソースの能力を評価することが可能になると考えられる．

2.2 性能評価に求められる要件

このようなネットワークシステム性能評価に必要とされる要件を以下に示す．

（1） 終点アプリケーションでの利用者の感じるスループットや応答性能を測定できること．

ネットワークシステムの性能評価の目標は，終点アプリケーションでの性能を知ることである．システムの全体的な性能は途中の経路の状況と相関があるとは限らない．したがって，途中の経路に注目するのではなく，終点アプリケーション自体の性能で評価する必要がある．

（2） 通信経路のデータリンクの種類によらず測定できること．

インターネットでは Ethernet，FDDI，ATM，ISDN，専用線，フレームリレーなど様々なデータリンクが利用されている．アプリケーションプロトコルはデータリンクの上位層であり，データリンク非依存の論理的な通信手順である．したがって，ネットワークシステムの性能測定はデータリンクの種類にかかわらず行えなければならない．

（3） 特定のアプリケーションに依存しないこと．

インターネットでは多くのアプリケーションが利用されているが，性能評価システムは，特定のアプリケーションやアプリケーションプロトコルを前提としたものではなく，これらの性能を共通の方法で測定できる必要がある．

（4） 既存のアプリケーションへの変更を必要としないこと．

現在稼働しているシステムやアプリケーションソフトウェアに変更を加えて測定機能を付加することはコストが大きく，また変更が容易でないアプリケーションも多い．このため，これらに変更を加えることなく，最小限の設定変更のみで測定が行える必要がある．

（5） 実際に稼働しているシステムに対して測定が行えること．

計算機上のシミュレーションは，すべての性能要因をシミュレートすることが難しく，ベンチマークによる測定では実際の稼働状態の再現が難しく，結果はベンチマーク環境に限定されたものである．したがって，利用されているシステムの稼働状態での測定が行えることが望ましい．

（6） 複数のクライアントでの性能を一元的に管理できること．

ネットワーク上のサーバは多くのクライアントから利用されるため，異なったネットワーク環境にある複数クライアントの性能を測定し，性能情報を収集し一括管理が行える機構が必要である．

2.3 従来の性能測定手法

これまで，ネットワークの性能評価を目的として様々な試みが行われてきた．以下に，従来の性能評価方式とその問題点を示す．

（1） サーバのアクセス記録から解析

アクセス記録から，アクセス数，転送データ量やリクエストの処理時間を統計的に計算する方法である．しかし，アクセス記録から得られる内容はサーバ自体の性能であり，クライアントの性能やネットワーク回線に関する性能ではないため，実際にユーザが感じる性能とは隔たりがある．また，記録がアプリケーション固有の形式であり，アプリケーションごとに解析プログラムが個別に必要となる．

更に，アクセス記録に性能に関する記録を行わないものやアクセス記録を行わないアプリケーションも多い．

（2） ネットワーク利用率や遅延時間の測定

SNMP [2], [3] を用いたネットワーク機器の遠隔監視として広く用いられている方法である．中継ノードやリンクの利用率を測定するため，特定のアプリケーションプロトコルに依存しない．しかし，ネットワークシステムの性能は，利用率や遅延だけではなく，ネットワーク層の特性，端末間のスループット，サーバシステムやクライアントシステムの性能などに大きく左右される．また，多くのアプリケーションで用いられている TCP の性能は，利用率や遅延にかかわらずネットワーク層やアプリケーション層の特性によって悪化することが知られている．

（3） ベンチマーク

これまで SPECweb [7]，WebStone [8]，ttcp や DBS [9] などのいくつかのベンチマークツールが開発されてきた．

これらは定量的な性能計測が行える反面，実際の利用者の振舞いや運用形態に適したベンチマーク条件を設定することが困難である．また，アプリケーション性能を測定するためにはベンチマークツールをアプリケーションごとに用意する必要がある．

（4） パケット収集

ネットワークを流れるパケットをアナライザで解析する方法である．しかし，すべてのパケットを観測し分析するには高い性能のアナライザが必要となり非現実的である．また，アナライザは特定のデータリンクにしか対応していない．例えば snoop，tcpdump [10] などのツールはトラヒック共有型のネットワークでしか利用できない．

（5） ユーザへのヒアリング

実際に利用者へシステムの使用感をたずねる方法である．評価基準があいまいであり，技術的な指針となりにくい．

以上から，異なる環境下でも稼働状態のネットワークシステムに対して測定を可能とするような性能評価システムが必要だと考えられる．

3. 監視層による性能評価

3.1 クライアントアプリケーションの監視による性能評価

本論文では，クライアントアプリケーションの振舞いの観測を可能にする監視層を導入することにより，ネットワークシステムの性能を評価する手法を提案する．

提案方式は実際に稼働しているネットワークシステムの利用者から見た性能を計測できるため，終点アプリケーションのサービス品質を評価することが可能である．

本手法の動作原理を図 1 に示す．監視層はクライアントアプリケーションの挙動を観測することよって性能計測を行う．具体的には，クライアントアプリケーションからネットワーク上のサーバへの通信は，すべて OS 内のネットワーク機能のシステムコールを利用して行うため，システムコールの内容を監視するレイヤを新たに導入することにより，クライアントアプリケーションのネットワーク利用状況を把握することが

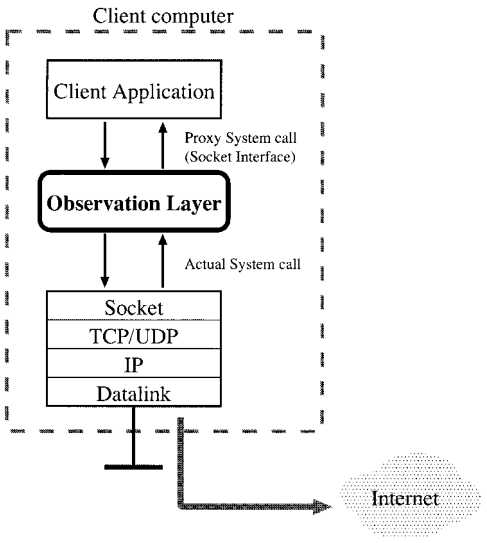

図 1 監視層による測定の動作原理
Fig. 1 Fundamental principle of measurement by observation layer.

できる．

監視層は，クライアントアプリケーションの挙動を直接監視することができるので，利用者の感じる性能に近い値を計測することが可能である．また，監視層を BSD Socket，TLI，Winsock といった OS のネットワーク機能と同一のインタフェースとすることにより，アプリケーションから透過的に扱えるのでアプリケーションの改造を必要としない．更に，複数クライアントの監視を行うために監視層の計測を遠隔操作できるようなアーキテクチャとする．

これらにより **2.2** の要求をすべて満たすことができる．具体的には，監視層は稼働中のクライアントアプリケーションを監視するものなので，終点アプリケーション性能を計測することが可能であり，標準的なシステムコールのインタフェースを用いるため既存のアプリケーションへの適用が容易であり，またこれらのシステムコールはデータリンク非依存である．以上によって（1）～（5）までの機能を実現できる．更に，遠隔監視用の管理マネージャによって（6）が実現できると考えられる．

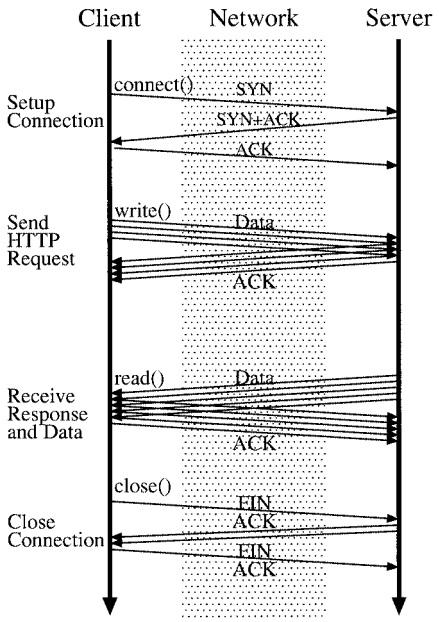

図 2　HTTP の通信フロー
Fig. 2　Communication flow in HTTP.

3.2　要求–応答単位の通信モデル

ここで，監視層による性能測定方式が対象とするクライアント–サーバ間の通信モデルについて述べる．HTTP [11]，SMTP，POP3 といった多くのインターネットのアプリケーションプロトコルでは，サーバとクライアントの通信モデルとして，要求と応答を単位とした以下の手順に従っている．

（1） コネクションの確立（`connect()` システムコール）
（2） 要求メッセージの送信（`write()` システムコール）
（3） 応答メッセージの受信（`read()` システムコール）
（4） …（2),(3）の繰返し
（5） コネクションの切断（`close()` システムコール）

このような通信モデルを「要求–応答単位の通信モデル」と呼ぶ．

要求–応答単位の通信モデルの例として，HTTP の通信フローを図 2 に示す．クライアントは，まずサーバとの間でコネクションを確立し，次に，サーバに GET，PUT などの要求を送信し，更に応答としてコンテンツを受け取る．最後にコネクションをクローズする．

3.3　性 能 指 標

監視層では，クライアントと OS 間のシステムコールを監視することによって計測を行うことが可能である．前述の HTTP における要求–応答単位の通信モデルをもとにした性能指標を図 3 に示す．監視層での計測内容は T_1〜T_7 の時間及び送受信されたデータ量である．これにより計測可能な性能指標を以下に示す．

（1） トランザクション時間 T_t（T_1–T_7 間）
`connect()` システムコールの開始から `close()` システムコールの終了までに要した時間．処理開始から処理終了までの合計時間．この値は，ユーザの感じる処理時間に最も近いと考えられる．

（2） コネクション確立時間 T_e（T_1–T_2 間）
TCP コネクションを確立するため `connect()` システムコールに要した時間．ネットワークの RTT（Round Trip Time）と端末の TCP/IP 機能の性能を示す．

（3） リクエスト送信時間 T_s（T_3–T_4 間）
リクエストデータを送信するための `write()` システムコールに要した時間．リクエストデータの送信に要した時間．

（4） リクエスト処理及び応答受信時間 T_r（T_5–T_6 間）

リクエストに対する処理結果を受信するための `read()` システムコールに要した時間．サーバがリクエストの処理に要した時間とデータ転送に要した時間の合計である．

（5） リクエストデータ送信量 D_s

リクエストデータを送信するための `write()` システムコールでサーバに送られたデータの大きさ．

（6） 応答データ受信量 D_r

応答データを受信するための `read()` システムコールでサーバから送られてきたデータの大きさ．

（7） リクエストデータ送信レート R_s（D_s/T_s）

クライアントからサーバに送られたリクエストデータの送信レート．

（8） 応答データ受信レート R_r（D_r/T_r）

サーバからクライアントに送られた応答データの受信レート．

（9） 全データ処理レート R_a（$(\sum D_s + \sum D_r)/T_t$）

通信に要したすべての時間に対する全転送データ量の割合．扱うデータ量に応じたユーザの感じる性能の指標となる．

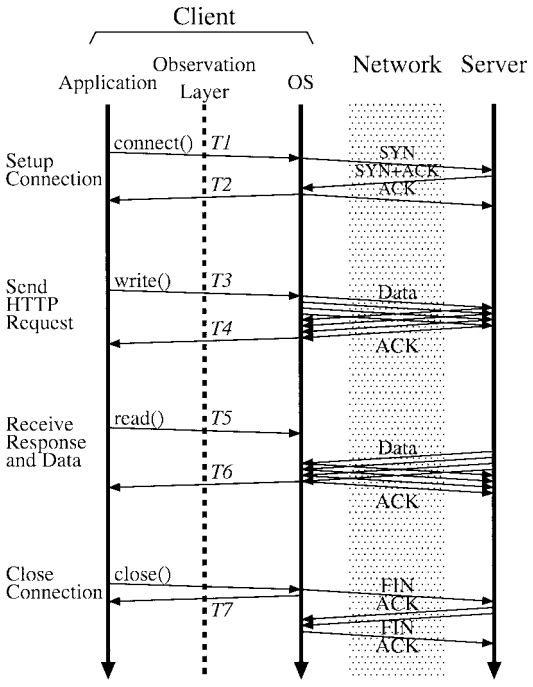

図 3 監視層で計測される性能指標
Fig. 3 Performance indices by observation layer.

4. 監視層による性能評価システム

4.1 システム構成

監視層を用いた性能評価システムの設計について述べる．システム構成を図 4 に示し，各部分について説明する．

（1） 監視層（Observation Layer）

アプリケーションと OS 間のやりとりを監視し，通信時間や転送データ量を測定する部分．

（2） データスライサ（Data Slicer）

指定した条件によって測定データのフィルタリングを行い加工する部分．

（3） データベース（Database）

加工済み測定結果を格納する部分．

（4） マネージャ（Manager）

データの収集，可視化，分析を行う管理マネージャ．

4.2 動作手順

監視層のコネクション管理部分でネットワーク利用に関するシステムコールが監視され，コネクションの状態，T_1〜T_7 までの各タイムラインの時刻，転送データ量が測定される．次にデータスライサによって計測条件によるフィルタリングが行われ，計測結果がデータベースに格納される．

管理マネージャは計測対象システムから計測結果を受け取り，データの可視化，他クライアントの計測結果の比較などを行うことができる．

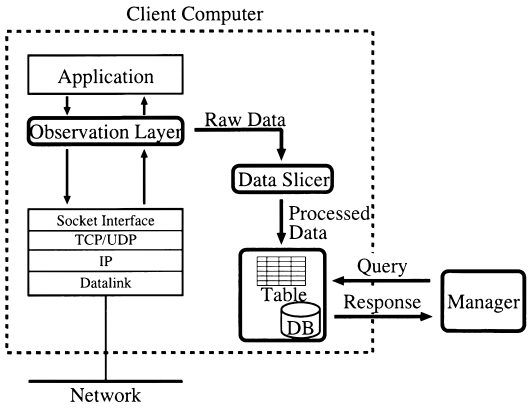

図 4 システム構成
Fig. 4 System structure.

5. 実　　装

WWW システムを対象とした監視層による性能評価システムの実装を行った．

クライアント側の監視システムの実装は ANSI C を用い Sun Solaris 上で行った．監視層は Socket インタフェースをもつ代理システムコールからなる共有ライブラリーとして実装されている．動的リンク機能を利用し，アプリケーション実行時に共有ライブラリーのプリロードを用い通常のシステムコールを監視層の代理システムコールで上書きすることにより，クライアントアプリケーションの変更や再コンパイルすることなく計測を可能にしている．また，このような動的リンク機能は主要な UNIX プラットホームや Windows にも備わっているため，他のプラットホームへの対応は容易であると考えられる．また，たとえこのような機能が備わっていないシステムでもアプリケーションを再コンパイルするのみでソースコードを変更する必要はない．

アプリケーションが Socket インタフェースにアクセスすると，監視層の計測用の代理 Socket インタフェースの関数が呼び出される．代理 Socket インタフェースは，Socket の生成，コネクションの確立，切断，中断及びデータの送受信を解析し，T_1 より T_7 の各タイムラインの記録及び転送データ量をカウントしたうえで，実際の OS 内の Socket インタフェースのシステムコールを起動する．計測結果はデータスライサでフィルタ処理が行われ，データベースに記録される．これらの処理はコネクション単位に逐次的に行われている．

マネージャは現在シェルスクリプト及び C で記述しており，データの可視化機能として Gnuplot へのインタフェースが実装されている．

6. 検　　証

監視層による性能評価システムの有効性を確認するため，実験環境上に構成した WWW システム上で検証実験を行った．

6.1 監視オーバヘッドの検証

本システムでは，データを転送するたびに時刻と転送量を記録している．この処理は CPU の負荷となり測定結果に影響を及ぼす可能性がある．負荷に対するスケーラビリティの評価のため，図 5 に示す実験環境下でのクライアントでのスループットをアプリケーションが監視層を介した場合（クライアント 1）と，アプリケーション単体の場合（クライアント 2）とで比較し，測定のオーバヘッドを検証した．

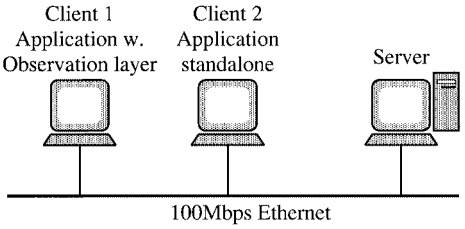

図 5　オーバヘッドの検証：実験ネットワーク
Fig. 5　Network in experiment on overhead.

表 1　オーバヘッドの検証：実験環境
Table 1　Configuration in experiment on overhead.

サーバ	Sun Ultra 60（Solaris2.6）
クライアント 1，2	Sun Ultra 10（Solaris2.6）
サーバプログラム	Apache 1.3.12
クライアントプログラム	ApacheBench 1.3
1 トランザクション当りのデータサイズ	51200 byte

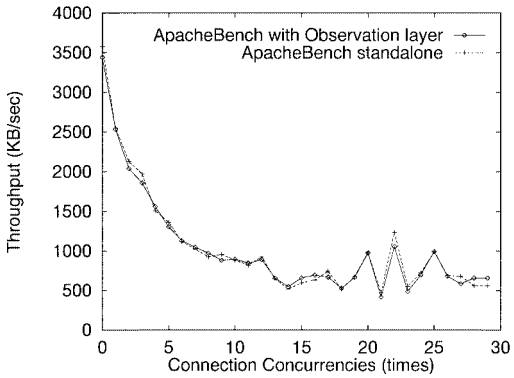

図 6　オーバヘッドの検証：スループットの比較結果
Fig. 6　Comparison on throughput.

表 1 に示す条件で，100 Mbps の Ethernet に接続されたサーバに対し，2 種類のクライアントから同時にアクセスを行った．1 コネクション当りの平均のスループットを測定した結果を図 6 に示す．

結果から，実験環境ではオーバヘッドは平均 2%であった．したがって，大量のデータ転送を伴わない通常のインターネットで用いられるようなアプリケーションでは，監視層のオーバヘッドによる影響は実用上問題ないものと考えられる．

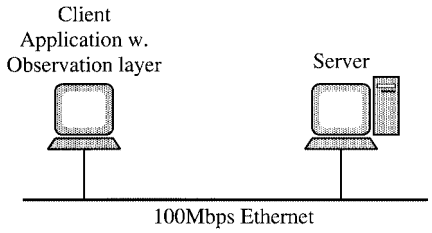

図 7　サーバアクセス記録との比較：実験ネットワーク
Fig. 7　Network in experiment on comparison with access log in server.

表 2　サーバアクセス記録との比較：実験環境
Table 2　Configuration in experimant on comparison with access log in server.

サーバ	Sun Ultra 60 (Solaris2.6)
クライアント	Sun Ultra 10 (Solaris2.6)
サーバプログラム	Apache 1.3.12
クライアントプログラム	ApacheBench 1.3

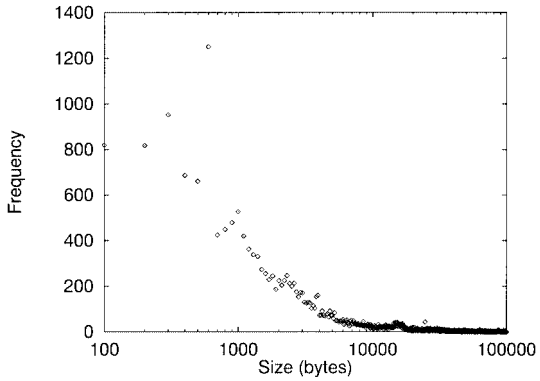

図 8　実験サーバ上のファイルサイズの頻度分布
Fig. 8　Frequency of file size in server.

6.2　サーバ上アクセス記録との比較

監視層が性能を正確に計測することができるか検証するため，稼働中の WWW システムの性能を監視層で計測した場合と，サーバのアクセス記録で計測した場合の比較を行った．

実験環境を図 7 と表 2 に示す．実験サーバ上に，明治大学理工学部情報科学科に設置されている WWW サーバ内のコンテンツと同サイズのファイル約 20,000 個を用意した．ファイルサイズの頻度分布を図 8 に示す．

この実験では，クライアントからサーバ上のファイルに対しランダムに 10,000 アクセスを行い，監視層でのトランザクション時間 T_t とサーバのアクセス記録からわかるサーバ上での処理時間の計測を行った．結果の頻度分布を図 9 に示す．サーバのアクセス記録では 10 ms 付近に最も多く分布しているのに対し，監視層では 200 ms 付近に分布している．計測値の平均値は監視層では 815 ms，サーバのアクセス記録では 418 ms であり，サーバのアクセス記録の結果は監視層の結果に対して約 51% の差があることが確認された．

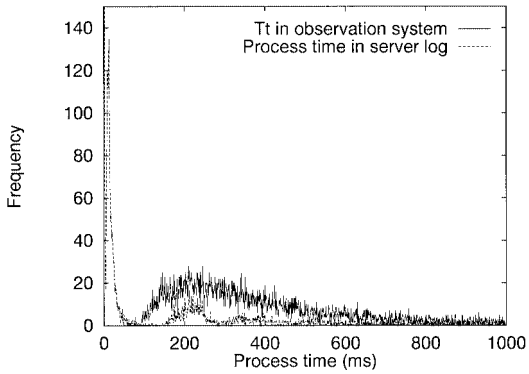

図 9　監視層での T_t とサーバアクセス記録での処理時間の頻度分布の比較
Fig. 9　Comparison T_t in observation layer with process time in server log.

監視層による計測値はコネクション確立から終了までのトランザクション全体の処理時間であるのに対し，アクセス記録による計測値は，サーバ上のアプリケーション層での性能であるためコネクション確立，終了に要した時間と，ネットワークでのデータ転送に要した時間を記録することができない．例えば，TCP ではコネクション確立のために 3 ウェイハンドシェーク手順が必要であるが，これには最低 1.5 RTT の時間が必要である．これらが原因で異なった結果になったと考えられる．

更に，監視層で計測された処理時間はデータサイズの分布に対して広く分散しており，サーバのアクセス記録で計測された処理時間は集中していることがわかる．この原因を明らかにするため，各アクセスにおけるデータサイズと処理時間の関係を検証した．監視層の T_t から求めた分布を図 10 に，サーバのアクセス記録から求めた分布を図 11 に示す．

監視層からは，同一データサイズであっても，データサイズに対し線形な処理時間以上の範囲で広範に分

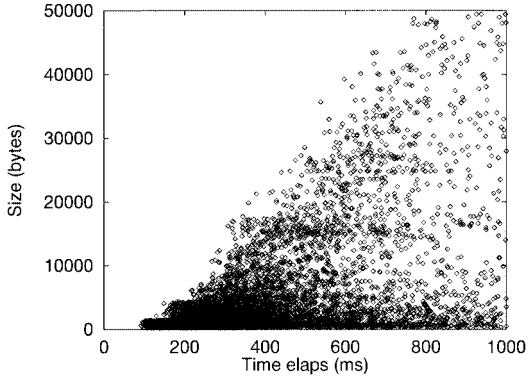

図 10　監視層の計測結果から分析したデータサイズと処理時間の分布
Fig. 10　Distribution of data size and process time in observation layer.

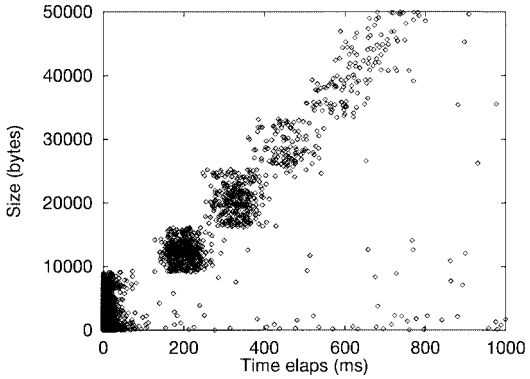

図 11　サーバのアクセス記録から分析したデータサイズと処理時間の分布
Fig. 11　Distribution of data size and process time in server log.

布していることが計測された．これはネットワークのふくそうが原因と考えられる．ふくそうによる TCP セグメントの再送は全体の処理時間に与える影響が大きく，また，再送はコネクションごとに独立な間隔で行われる．更に，実験のような小さなサイズのデータ転送では，処理時間の大部分がコネクション確立/終了に要する時間であるため，この範囲ではデータサイズと処理時間の関係が小さくなる．このような要因によって処理時間が広範に分布したと考えられる．

一方，サーバのアクセス記録からは，データサイズに対する処理時間の分布が一定のデータサイズの倍数を境界とした階段状の分布として計測された．データサイズが 0〜約 8,500 byte のときに約 10 ms 付近に多く分布し，以降約 17,000 byte までが 200 ms 付近に分布し，約 26,000 byte までが約 330 ms に分布している．

サーバアプリケーション上では，ソケットバッファに送信データのコピーが行われた時点で送信終了とみなされてしまい，実際のデータ転送時間が計測できない．実験に用いた Solaris では，送信ソケットバッファは約 8 kbyte に設計されており，送信処理において，送信データの残りが約 8 kbyte 以下になった時点で処理が終了したものと考えられる．

実験に用いたサーバ上のファイルの約 90% が 30 kbyte 以下，すなわちバッファサイズの数倍以内に収まるサイズであったため，バッファサイズ単位で処理が繰り上げられ図 11 のような階段状の分布になったと考えられる．

一般にインターネットの WWW サーバ上のオブジェクトサイズの大半が 8 kbyte 以下と考えられている [12] ことから，サーバのアクセス記録では性能を正確に計測できないことが，この実験で明らかになった．

更に，実際の運用時において，使用ネットワークの帯域や遅延に合わせて送信バッファサイズの増減を行う場合，サーバのアクセス記録ではバッファリングの影響で計測自体が不正確なものとなり，本来のシステムの性能を見誤る可能性がある．

これらの結果から，監視層は実際に稼働しているシステムの性能を正確に計測することが可能であると考えられる．

7. むすび

本論文では，ネットワークサービスの利便性を確保するためのサービス品質について論じ，利用者の利便性の観点での性能評価を可能にする監視層によるネットワークアプリケーションの性能測定方式を提案した．

提案手法は，クライアントとサーバの間でやり取りされる要求–応答単位の通信に注目し，クライアントアプリケーションの挙動を観測することにより，終点アプリケーションの性能を評価することが可能である．

この方式の有効性を確認するため，既存の性能評価の方法と比較し機能面で優れていることを示した．また，計測処理が与えるオーバヘッドの影響は小さなものであることを実験によって確認した．更に，実際の運用形態に近い実験ネットワークに適用し，これまで正確に把握することが難しかった稼働状態のシステム

の性能を明確に評価できることを示した．

文　献

[1] 間瀬憲一，“インターネットの品質・トラヒック管理［I］—ネットワークのモデル化と品質・トラヒック管理の概要—，”信学誌，vol.82, no.10, pp.1054–1061, Oct. 1999.

[2] A. Leinwand, “Accomplishing Performance Management with SNMP,” Proc. INET93, San Francisco, CA, Aug. 1993.

[3] A. Leinwand, and K.F. Conroy, Network Management: A Practical Perspective, Second Edition, Addison-Wesley, MA, 1996.

[4] 斉藤裕樹，中所武司，“イベント駆動型管理戦略に基づくネットワーク性能管理手法の提案，”情処学 DPS 研報，99-DPS-91, pp.97–102, Jan. 1999.

[5] M. Allman, C. Hayes, H. Kruse, and S. Osterman, “TCP Performance over Satellites Links,” Proc. the 5th International Conference on Telecommunications Systems, 1997.

[6] B. Vandalore, S. Kalyanaraman, R. Jain, R. Goyal, and S. Fahmy, “Simulation Study of World Wide Web traffic over the ATM ABR Service,” Conference on Performance and Control of Network Systems II, Proc. SPIE Symposium on Voice, Video and Data Communications, vol.3530, pp.415–422, Boston, MA, Nov. 1998.

[7] SPECweb99 Benchmark, Standard Performance Evaluation Corporation, http://www.spec.org/osg/web99/.

[8] WebStone — The Benchmark for Web Servers, http://www.mindcraft.com/benchmarks/webstone/ .

[9] 村山公保，門林雄基，山口　英，“TCP 性能評価システム DBS の構築，” コンピュータソフトウェア，vol.15, no.2, pp.24–37, March 1998.

[10] tcpdump, LBNL's Network Research Group, http://ee.lbl.gov/.

[11] R. Fielding, J. Getty, J. Mogul, H. Frystyk, and T. Berners-Lee, “Hyper Text Transfer Protocol - HTTP/1.1,” RFC2068, Jan. 1997.

[12] 名部正彦，馬場健一，村田正幸，宮原秀夫，“インターネット・アクセスネットワーク設計のための WWW トラヒックの分布とモデル化，” 信学論（B-I），vol.J80-B-I, no.6, pp.428–437, June 1997.

[13] Y. Nakamura, K. Chinen, H. Sunahara, S. Yamaguchi, and Y. Oie, “ENMA: The WWW server Performance Measurement System via Packet Monitoring,” Proc. INET99, San Jose, CA, July 1999.

[14] 斉藤裕樹，中所武司，“透過的モニタリング層によるネットワーク性能管理方式の提案，”情処学第 59 回全大（3），pp.471–472, Sept. 1999.

[15] 斉藤裕樹，中所武司，“透過的クライアント監視によるネットワークシステム性能管理手法，” インターネットコンファレンス’99, pp.47–55, Dec. 1999.

[16] H. Saito and T. Chusho, “Design and Implementation of Network Performance Evaluation System through Client Observation,” Proc. INET2000, Yokohama, July 2000.

（平成 12 年 8 月 3 日受付，10 月 25 日再受付）

斉藤　裕樹

平 8 明大・理工・情報科学卒．平 10 同大大学院理工学研究科基礎理工学専攻情報科学系博士前期課程了．同年より同博士後期課程在学中．分散システム，インターネットアーキテクチャ，ネットワーク管理技術に関心をもつ．情報処理学会，日本ソフトウェア科学会各会員．

中所　武司　（正員）

昭 44 東大・工・電子卒．昭 46 同大大学院修士課程了．同年（株）日立製作所入社．同社システム開発研究所主任研究員を経て，平 5 より明大・理工・情報科学・教授，現在に至る．ソフトウェア工学の研究に従事．コンポーネントベースのアプリケーション開発方法論に関心をもつ．工博．情報処理学会，日本ソフトウェア科学会，人工知能学会，日本信頼性学会，IEEE Computer Society，ACM 各会員．