特別研究報告

題目

サービスオーバレイネットワークのための インラインネットワーク計測手法の実装および評価

指導教官

宮原 秀夫 教授

報告者

Insixiengmai Leuth

2004 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 15 年度 特別研究報告

サービスオーバレイネットワークのためのインラインネットワーク計測手法の実装および評価

Insixiengmai Leuth

内容梗概

サービスオーバレイネットワークにおいては、下位層ネットワークに相当する IP ネットワークの状態（利用可能帯域、輻輳レベル、ルーティング情報など）に関するできるだけ最新かつ詳細な情報を随時把握することで、サービス品質の向上が可能となる。しかし、既存のネットワーク計測アルゴリズムは、より正確な情報を得ることを目的とするため、計測に時間がかかる、多くの計測用のパケットを用いるため通常トラヒックに与える影響が大きいなどの問題点を持つ。これに対して我々の研究グループにおいては、エンド端末間の利用可能帯域を計測するためのインラインネットワーク計測手法を提案している。提案手法においては、過去の計測結果を統計的に利用することにより、次回の計測範囲を小さくし、必要となるパケット数を抑えることが可能である。また、計測アルゴリズムを TCP に組み込むことで、データ転送中の TCP コネクションのパケットを利用し、利用可能帯域を継続的かつ正確に計測するインラインネットワーク計測手法である ImTCP 方式を提案している。

提案方式の性能評価はこれまでシミュレーションによって行っているが、本手法の有効性を検証するためには、実コンピュータ、および実ネットワークを用いた実装実験が必須である。そこで本報告では、インラインネットワーク計測アルゴリズムを実コンピュータ上に実装し、実験ネットワークを用いた性能評価を行う。実装においては、UDP を用いたアプリケーションプログラムによって計測アルゴリズムを実現し、計測アルゴリズムの基本的性質を検証することにより、提案している計測アルゴリズムが実ネットワークにおいても有効であることを示す。また、TCP コネクションを用いた計測手法である ImTCP 実装に関する指針を示す。

実装実験の結果、UDP を用いたアプリケーションレベルの実装においては、カーネルとアプリケーションの間の通信遅延が無視することができず、計測性能に大きな影響を与える

ことがわかった。さらに、通信遅延の影響をできる限り小さくする手法を用いることで、コンピュータ上のシミュレーションで得られた結果とほぼ同等のパケット数で、同程度の精度の計測結果が得られることが明らかになった。

主な用語

TCP (Transmission Control Protocol)、 利用可能帯域、 インラインネットワーク計測、実装、サービスオーバレイネットワーク

目 次

図 目 次

表 目 次

1　はじめに

ネットワーク計測に関する研究はこれまで活発に行われてきており、数多くの計測ツールが開発されてきた [1-5]。これらのツールにより、リンクの物理帯域、エンドホスト間の利用可能帯域、伝送遅延時間など、様々なネットワークの特性を計測することができる。ネットワーク特性の計測結果は、ネットワーク内の故障箇所の特定や解決、およびネットワーク設計等に用いられる。ネットワーク計測手法は通常、受動的計測方式 (Passive Measurement) と能動的計測方式 (Active Measurement) の 2 種類に分類される。受動的計測方式は、ネットワーク内のある計測地点を通過するトラヒックを監視し、その情報を収集することでネットワーク特性の導出を行う。一方、能動的計測方式では、あるホストからネットワーク内に計測用のパケットを送出し、その結果 (転送遅延、パケット到着間隔など) からネットワーク特性を推測する。後者は計測のために余分なパケットをネットワークに送出する必要があるが、前者に比べより詳細なネットワーク特性情報を得ることができる。

一方、近年のネットワークサービスの多様化に伴い、サービスオリエンテッドなネットワーク (サービスオーバレイネットワーク) が拡がりつつある。例えば、ピア同士の直接的な通信を実現する P2P ネットワーク [6,7] 、ネットワーク上での分散計算環境を提供するグリッドネットワーク [8-11] 、コンテンツ配信を目的とした Contents Delivery Network (CDN) [12-14] 、IP ネットワーク上に仮想網を構築する IP-VPN [15] などである。これらのネットワークは、IP ネットワークを下位層ネットワークとして用いることにより、特定のサービスを提供する上位層ネットワークととらえることができる。したがって、これらのネットワークにおいてサービス品質を向上させるためには、下位層ネットワークである IP ネットワークを与条件として、サービス提供のためのコネクション設定要求が発生した時に、利用可能な下位層ネットワーク資源量を適切に把握することが重要である。

そこで我々の研究グループでは、エンドホスト間の利用可能帯域を計測するための、新たな計測手法を提案している [16]。利用可能帯域を測定するツールはこれまでにも多く提案されている [1-3] が、計測に長い時間がかかる、多くの計測用のパケットを用いるため外部トラヒックに与える影響が大きいなどの問題を持つ。サービスオーバレイネットワークにおける計測においては、利用可能なネットワーク資源量に関する常に最新の情報をネットワーク

内の他のトラヒックに悪影響を与えることなく取得することが重要であるため、既存の方式をそのまま適用することはできない。一方、我々が提案している計測手法においては、過去の計測結果を統計的に処理し、次回の計測における計測範囲を限定することによって、一回の計測に必要なパケット数を減らし、短い間隔で継続的に計測結果を導出することができる。また、より少ない計測パケット数で広い帯域範囲を計測するための効率的な計測パケット送出アルゴリズムを持つ。また [16] においては、計測アルゴリズムを TCP (Transmission Control Protocol) [17] に組み込み、外部トラヒックに影響を与えることなくデータ転送と利用可能帯域の計測を同時に行うことができる ImTCP (Inline measurement TCP) を提案している。

[16] においては、提案手法の有効性をコンピュータ上のシミュレーションによって検証している。その結果、提案した計測アルゴリズムが少ない計測パケット数で高い精度の計測結果を継続的に出力すること、ImTCP 方式が TCP データ転送速度を低下させることなく利用可能帯域の計測を行うことができること、などを確認している。しかし、ネットワーク計測手法の有効性の評価には、実コンピュータ、および実ネットワークを用いた実装実験が不可欠である。

そこで本報告においては、[16] において提案されたインラインネットワーク計測アルゴリズムの実装を行い、実験ネットワークを用いた実装実験を行うことにより、提案手法の実ネットワーク上での有効性を検証する。実装においては、UDP を用いたアプリケーションプログラムによって計測アルゴリズムを実現し、計測アルゴリズムの基本的性質を検証する。提案手法においては、データ転送中に得られる情報 (パケット送受信間隔) からエンドホスト間の利用可能帯域を推測している。しかし、UDP を用いたアプリケーションによる実験結果より、カーネルとアプリケーション間の通信遅延がパケットの送受信間隔に大きな影響を与え、計測精度や計測可能な帯域幅を低下させる原因になることが明らかになった。そのため、実装したアプリケーションにおいては、その遅延をできるだけ小さくするための改良を行う。さらに本報告では、TCP に計測アルゴリズムを組み込んだ ImTCP 方式を、FreeBSD [18] のカーネルシステムに実装し、計測機能を持つ TCP を提供するための実装指針を示す。

以下、2 章においてインラインネットワークの計測アルゴリズム、および TCP への組み込み方法に関する概略を述べ、3 章で UDP を用いたアプリケーションの概要および性能評価実験結果を示す。4 章においては、ImTCP 方式の実装に関する指針と問題点について議論する。最後に 5 章で本報告のまとめについて述べる。

2　インラインネットワーク計測手法の概要

本章では、[16] において提案し、本報告において実験を行うインラインネットワーク計測手法を簡単に説明する。2.1 節において、利用可能帯域の計測のために用いる計測アルゴリズムを示し、2.2 節で計測アルゴリズムを TCP に適用する ImTCP 方式の概略を示す。

2.1　計測アルゴリズム

提案方式は、送信側エンドホスト、受信側エンドホスト間のネットワークパスにおける現在の利用可能帯域値 A を導出する。図 1 にアルゴリズムの概略を示す。提案方式においては、まず送信側エンドホストが計測パケットを送出し、受信側エンドホストは受信した計測パケットをそのまま送信側エンドホストへ返送する。送信側端末は返送されたパケットの到着間隔から利用可能帯域の推測を行う。

提案方式において利用可能帯域を計測する際には、図 2 に示すように、現在の利用可能帯域値が含まれると考えられる帯域の上限 B_u と下限 B_l を設定し、区間 $I = (B_l, B_u)$ の中から利用可能帯域を探す。この区間を探索区間と呼ぶ。ここで、探索区間の下限 B_l の最小値は 0、上限 B_u の最大値は、送信側エンドホストが接続しているリンク帯域である。探索区間を設定することで、不必要に高いレートで計測パケットを送出することを避けることができるため、外部トラヒックに与える影響を最小限に抑えることができる。また、Topp [2] のように、非常に広い帯域幅の中から利用可能帯域を探索しないため、計測の精度を保ちながら用いる計測パケット数を減少させることができる。

後述するように、探索区間は過去の計測結果を基に設定するため、ネットワーク状況の変化に伴い利用可能帯域が急激に変化した時に、探索区間内に利用可能帯域が存在しない場合もある。提案するアルゴリズムでは、そのような場合においても、数回の計測で新たな利用可能帯域を発見することができる。提案する利用可能帯域計測アルゴリズムにおける、送信側エンドホストの動作概略を以下に示す。

(1) 初期探索区間を決定する

(2) 探索区間を複数の小区間に分割する

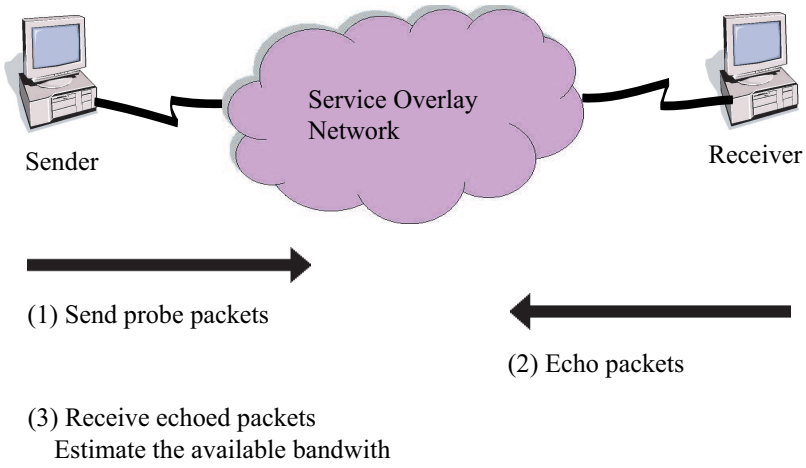


図 1: 計測アルゴリズムの概略

(3) 各小区間に対応する計測ストリームを送出・受信し、送出間隔と受信間隔の比較を行い、パケット間隔が大きくなったかどうかを判断する

(4) パケットの送受信結果から、利用可能帯域が含まれると考えられる小区間を選択する

(5) 選択した小区間に対応する計測ストリームの送出・受信結果から、利用可能帯域を算出する

(6) 探索区間の再計算を行い、(2) へ戻る

以下、ステップ (1)-(6) における詳細なアルゴリズムを示す。ここで、計測ストリームとは、計測のために一度にネットワークへ送出するパケット群のことを指す。また、1 つの計測ストリームで用いるパケット数を N とする。

(1) 初期検索区間の決定

まず、Cprobe のアルゴリズム [1] に基づいて 1 つの計測ストリームを送出し、導出された利用可能帯域値 A_{cprobe} を用いて、初期探索区間を $(A_{cprobe}/2, A_{cprobe})$ と設定する。

(2) 探索区間の分割

探索区間を、以下のように大きさの等しい k 個の小区間 $I_i = (B_{i+1}, B_i)(i = 1, \ldots, k)$ に分割する (図 2)。すなわち、

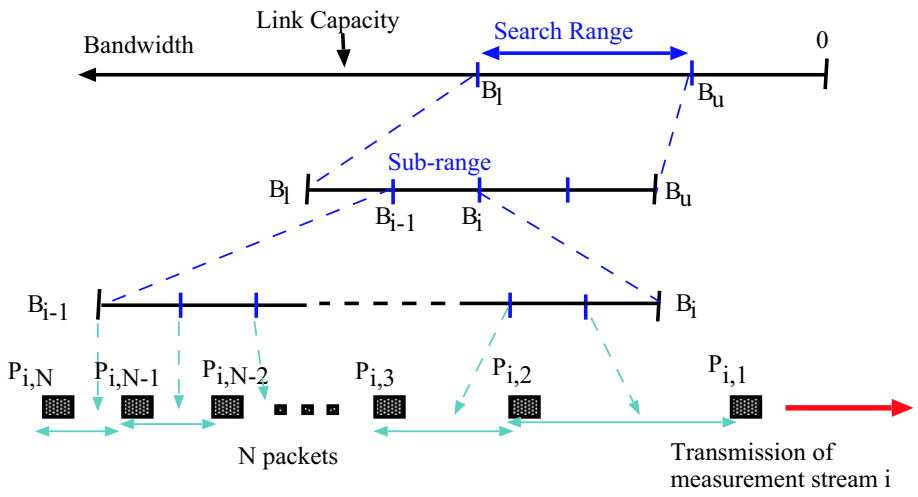


図 2: 検索区間、小区間、および計測ストリームの関係

$$B_i = B_u - \frac{B_u - B_l}{k}(i-1)(i = 1, \ldots, k+1)$$

となる。k が大きくなると、小区間の幅が小さくなるため、ステップ (4) 及び (6) においてより正確な判断が可能となるが、その反面、一回の計測に用いる計測ストリーム数が増加するため、計測にかかる時間が大きくなる。

(3) 計測ストリームの送出およびパケット間隔の比較

分割した k 個の小区間それぞれに対して、計測ストリーム $i(i = 1, \ldots, k)$ をネットワーク内に送出する。その際、1 つの計測ストリーム内のパケットの送信間隔を変化させることによって、小区間 $I_i = (B_{i+1}, B_i)$ が持つ帯域幅を全てカバーする (図 2)。つまり、計測ストリーム i 内の j 番目の計測パケット $P_{i,j}(1 \leq j \leq N)$ の送出時刻 $S_{i,j}$ は、以下の関係を満たす。ただし、$S_{i,1} = 0$ とし、計測パケットサイズを M とする。

$$\frac{M}{S_{i,j+1} - S_{i,j}} = B_{i+1} + \frac{B_i - B_{i+1}}{N}(j-1)$$

提案方式ではこのように、1 つの計測ストリーム内のパケットの送出間隔を変化させて送信するため、PathLoad のように計測ストリーム内のパケットを全て同じ送出間隔で送出する

方式に比べて計測誤差が大きくなる。しかしその反面、1 つの計測ストリームで様々な送信レートに対する計測結果を収集できるため、計測パケット数が少なくなり、短時間で計測結果を導出することができる。

次に、送出された計測ストリーム内のパケット $P_{i,j}$ が返送されてくると、その到着時刻を記録する。それを $R_{i,j}$ とする $(R_{i,1} = 0)$。送信パケット $P_{i,j}(1 \leq j \leq N-1)$ に対して、パケット送出間隔 $(S_{i,j+1} - S_{i,j})$ と到着間隔 $(R_{i,j+1} - R_{i,j})$ を PathLoad で用いられているアルゴリズム [3] を用いて比較し、パケット間隔に増加の傾向があるかどうかを判断する。すなわち、送出間隔よりも到着間隔の方が大きければ、その送出間隔で実現される計測パケットの送信レートが、利用可能帯域よりも大きいと考えられる。PathLoad では、計測ストリーム内の全ての計測パケットについて間隔の増加の傾向を調べることで、利用可能帯域の推測を行っている。

各計測ストリーム i の増加の傾向を示す変数 T_i を、以下のように定義する。

$$T_i = \begin{cases} 1 & \text{パケット間隔に増加傾向がある} \\ -1 & \text{パケット間隔に増加傾向がない} \\ 0 & \text{パケット間隔の増加の傾向を判断できない} \end{cases}$$

計測ストリームは、小区間の帯域値が大きい方から順に送出するため、i が小さい時には $T_i = 1$ になり、i が大きくなると、計測ストリームの平均送出レートが小さくなるため、ある時点で $T_i = -1$ に転じる傾向を持つ。そこで、2 つの連続する計測ストリーム m および $m+1$ で、$T_m = T_{m+1} = -1$ になれば、$(m+2)$ 番目以降の計測ストリームは送出せずに、$T_i = -1(m+2 \leq i \leq k)$ とすることで、計測速度の向上を図っている。

提案方式ではこのように、各ストリームのパケット間隔に増加の傾向があるか否かのみを判断し、増加の度合いは用いない。これは、Cprobe や Topp のようにパケット間隔の増加の度合いを利用可能帯域値の算出に用いると、真の利用可能帯域の値に比べて計測パケットの送出レートが高い場合に、計測される利用可能帯域の値に誤差が生じるためである。

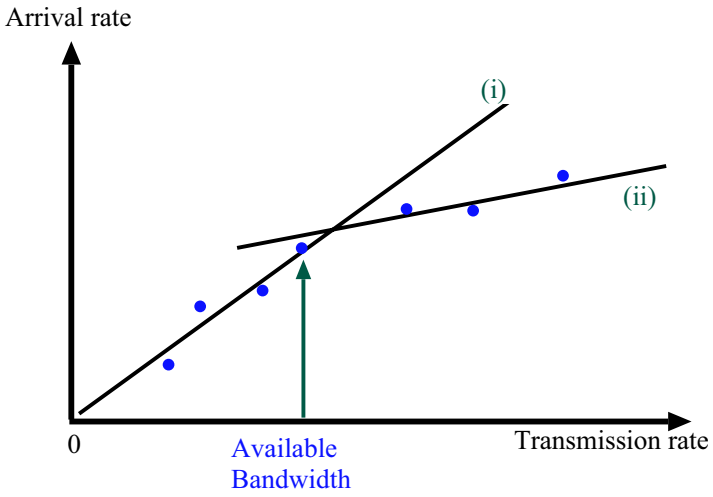


図 3: 小区間内の利用可能帯域導出方法

(4) 小区間の選択

全ての計測ストリームのパケット間隔の増加傾向 $T_i(1 \leq i \leq k)$ から、現在の利用可能帯域値 A が含まれると考えられる小区間を以下のように決定する。まず、$(\sum_{j=1}^{a} T_j - \sum_{a+1}^{k} T_j)$ が最大となる $a(0 \leq a \leq k+1)$ を求め、a が $1 \leq a \leq k$ を満たす場合には小区間 I_a を選択する。すなわち、I_a は、パケット間隔に増加の傾向がある部分とない部分の境界部分に存在する小区間となる。これは、パケット間隔の増加傾向が変化する境界部分に対応する計測ストリームの平均送出レートは、利用可能帯域の値に近いと考えられるためである。

また、$a = 0$ あるいは $a = k + 1$ の場合は、探索区間 (B_l, B_u) 内に利用可能帯域が存在せず、$a = 0$ の場合にはもっと大きな利用可能帯域値を持つ $(B_u < A)$、$a = k + 1$ の場合にはもっと小さな利用可能帯域を持つ $(A < B_l)$ ものとそれぞれ判断する。

(5) 利用可能帯域を導出

ステップ (4) において探索区間 (B_l, B_u) 内のある小区間 I_a 内に利用可能帯域が存在すると判断された場合は、以下のようにして利用可能帯域を導出する。まず、小区間 I_a に対応する計測ストリーム a 内の計測パケット $P_{a,j}(i = 1, \ldots, N)$ に対して、パケット送出レートおよびパケット到着レートをそれぞれ $\frac{M}{S_{i,j+1} - S_{i,j}}$ および $\frac{M}{R_{i,j+1} - R_{i,j}}$ と定義する。次に、図

3 に示すように、送出レートと到着レートの関係を線形回帰法によって 2 本の直線で近似する。ステップ (4) において述べたパケット到着間隔の増加の傾向から、2 本の直線 (i)、(ii) のうち、送出レートが低い (送出間隔が大きい) 点が含まれる直線 (i) は傾きが 1 に近く (送出レートと到着レートがほぼ等しい)、送出レートが高い (送出間隔が小さい) 点が含まれる直線 (ii) は傾きが 1 よりも小さい (送出レートよりも到着レートの方が大きい) という傾向を持つことがわかる。したがって、2 本直線の交点付近に利用可能帯域が存在すると考えられるため、直線 (i) のうち、最も送出レートが高いパケットの送出レートを、現在の利用可能帯域 A として導出する。

一方、ステップ (4) において探索区間 (B_l, B_u) 内に利用可能帯域が存在しないと判断された場合は、ネットワークの状況に変化があり、利用可能帯域が大きく変化していると考えられる。したがって、利用可能帯域の変化を陽に示すために、現在の利用可能帯域 A として暫定的に以下の値を導出する。

$$A = \begin{cases} B_l & a = 0 \\ B_u & a = k+1 \end{cases}$$

(6) 検索区間の再計算

ステップ (5) において小区間 I_a 内に利用可能帯域 A が発見され、かつその値がこれまでに蓄積されている利用可能帯域の計測値の集合から導出された 95% の信頼区間 $(B_{95,l}, B_{95,u})$ 内に含まれていれば、今回計測された利用可能帯域 A を新たに蓄積データとして保存し、新たに 95% の信頼区間を求め、その幅を次回の計測時の探索区間の幅とする。また、今回の計測結果 A を探索区間の中心とする。すなわち、次回の計測時の探索区間 (B'_l, B'_u) は以下のようになる。ここで $\bar{A}$ および S はそれぞれ統計データの平均および分散、q は蓄積データの個数である。

$$(B_{95,l}, B_{95,u}) = \left(\bar{A} - 1.96\frac{S}{\sqrt{q}}, \bar{A} + 1.96\frac{S}{\sqrt{q}}\right)$$

$$(B'_l, B'_u) = \left(A - max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right), A + max\left(1.96\frac{S}{\sqrt{q}}, \frac{B_m}{2}\right)\right)$$

ここで B_m は、探索空間が小さくなりすぎることを防止するために設定される探索区間の幅の最小値である。また、初回の計測時、および蓄積データが破棄された直後の計測時で、蓄積データが存在しない場合には、次回計測時の探索区間として、今回の探索区間を用いるものとする。

一方、ステップ (4) において探索区間 (B_l, B_u) 内に利用可能帯域が存在しないと判断された場合には、ネットワーク状況が変化した可能性があると判断し、これまでの利用可能帯域 A の蓄積を破棄する。そして、次回の探索空間 (B'_l, B'_u) を以下のように設定する。

$$(B'_l, B'_u) = \begin{cases} (B_l, B_u + \frac{B_u - B_l}{2}) & a = 0 \\ (B_l - \frac{B_u - B_l}{2}, B_u) & a = k + 1 \end{cases}$$

これは、ネットワーク状況が変化し、利用可能帯域に大きな変化がある場合に備えて、探索空間を利用可能帯域の変化の方向へ大きくすることを意味している。

2.2 TCP への組み込み方法

2.2.1 概略

TCP によるデータ転送においては、送信側 TCP がデータパケットを受信側 TCP に送信した後、受信側 TCP がそのパケットを受信すると、ACK パケットを送信側 TCP に向けて送り返す。提案手法では、これらのデータパケットと ACK パケットを計測アルゴリズムにおける計測用パケットとして使用することによって、計測アルゴリズムを TCP コネクションに適用する。

インラインネットワーク計測プログラムは図 4 に示すように、TCP 層の下位部分に組み込まれる。これは、後述するように、ImTCP は計測を効率的に行うために TCP のウインドウサイズを参照するためである。TCP 層で新たに生成されたデータパケットは、IP 層に渡される前に、一時的に中間バッファ(ImTCP バッファと呼ぶ) に保存され、計測アルゴリズムに基づいた時間間隔で送信される。一方、ACK パケットが送信側に到着すると、計測プログラムはその ACK パケットの到着時間を記録した後、TCP 層に渡す。

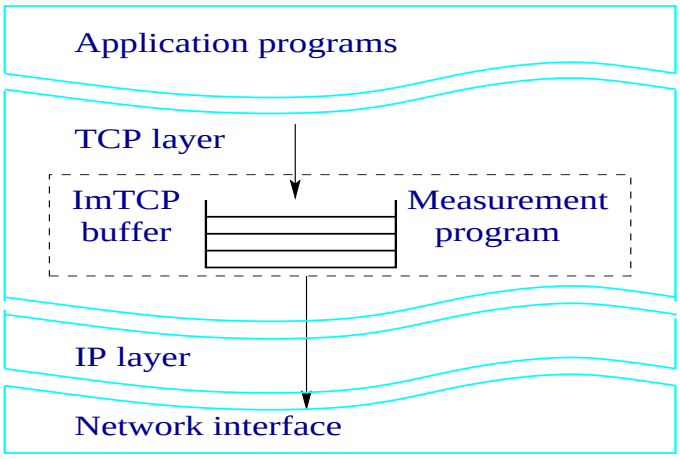


図 4: ImTCP の構造

計測プログラムの動作は、TCP コネクションのウインドウサイズの大きさに大きく影響を受ける。ウインドウサイズが計測ストリームに必要となるパケット数より小さい場合、計測ストリームを構成することができないため、TCP 層から渡されたパケットはバッファに蓄積せず、直ちに IP 層に渡される。それに対して、現在のウインドウサイズが十分大きい場合、計測アルゴリズムは複数の計測ストリームをウインドウサイズ内で作成することができる。このとき ImTCP においては、1RTT で最大で 1 回の計測を行うものとしている。

2.2.2 パケット蓄積・送信機構

図 5 は、3 つのユニットから構成される計測プログラムの構造を表している。以下に各部の概略を示す。

- ImTCP Buffer

  データパケットを格納し、Control Unit の制御によってデータパケットを IP 層に渡す。また、新しいパケットがバッファに到着すると、Control Unit にパケットの到着を通知する。

- Control Unit

  データパケットを ImTCP バッファに格納するかどうかを決定する。また、Measurement Unit から得られた計測区間を用いて、計測ストリームを生成する。

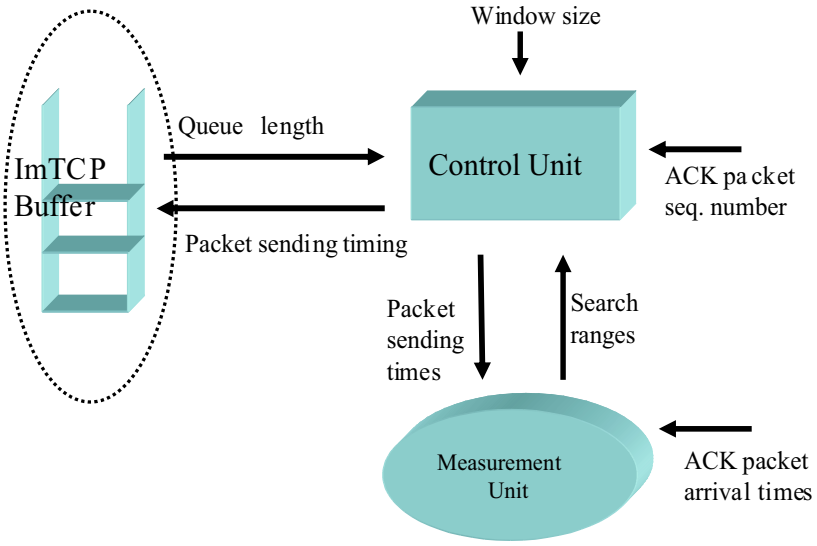


図 5: 計測プログラムの構造

- Measurement Unit

  データパケットの送信間隔と、ACK の受信間隔を用いて、2.1 節で示した計測アルゴリズムに基づいて利用可能帯域を推測する。

Measurement Unit の詳細は 2.1 節に示した通りである。ここでは、Control Unit について説明する。Control Unit は 4 つの状態 (STORE_PACKET、PASS_PACKET、SEND_STREAM、EMPTY_BUFFER) を持つ。図 6 は Control Unit の状態遷移の様子を示している。各状態における動作を下に示す。

- STORE_PACKET state
  - タイマをセットし、計測ストリームを生成するためのパケットを蓄積し始める。
  - 現在の TCP コネクションのウインドウサイズが N (計測ストリーム当たりのパケット数) より小さいならば、EMPTY_BUFFER state へ遷移する。
  - ImTCP バッファ内のパケット数が N より大きいならば、SEND_STREAM state へ遷移する。
  - 計測に必要なすべてのストリームの送信が終了すれば、PASS_PACKET state

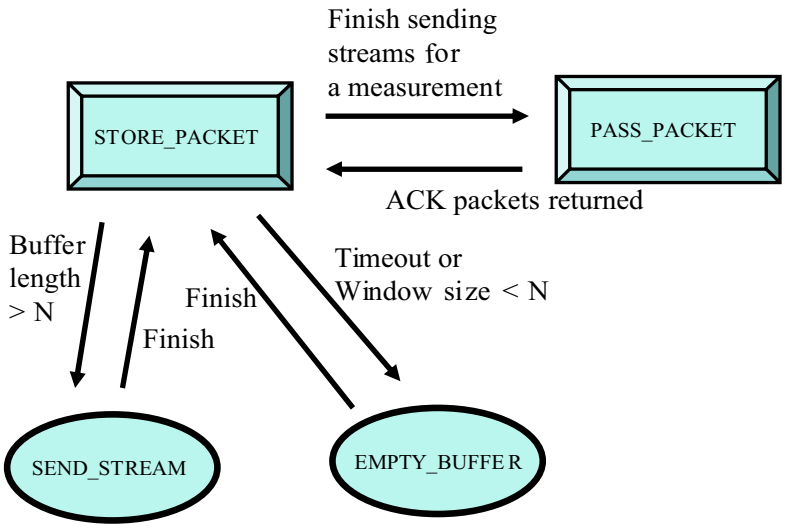


図 6: Control unit の状態遷移

へ遷移する。

- EMPTY_BUFFER state
  - タイマがセットされている場合、それを解除する。
  - ImTCP バッファ内にあるすべてのパケットを IP 層へ渡す。
  - STORE_PACKET state へ遷移する。
- SEND_STREAM state
  - タイマがセットされている場合、それを解除する。
  - 計測ストリームを送信する。そのときのパケット送信間隔は、計測アルゴリズムに基づいて決定される。送信中に新たにパケットが到着した場合、ImTCP バッファに格納する。
  - STORE_PACKET state へ遷移する。
- PASS PACKET state
  - ImTCP バッファ内にあるパケットをすべて IP 層へ渡す。また、新たに到着したパケットは、ImTCP バッファに蓄積せずに直ちに IP 層へ渡す。
  - 送信した計測ストリームに対応するすべての ACK パケットを受信したら、STORE_PACKET state へ遷移する。

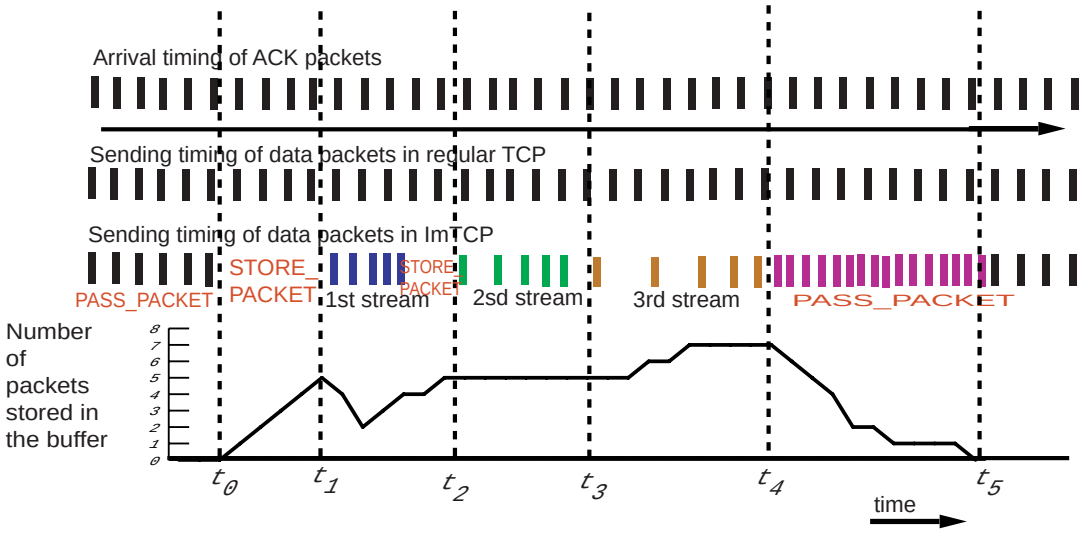


図 7: ImTCP パケットの送信例

図 7 は、ImTCP の動作の様子 (ストリーム当たりのパケット (N) は 5、一回の計測で用いる計測ストリーム数 (k) は 3) 、および ImTCP バッファ内の変化の様子を示す。図 7 においては、TCP は ACK パケットを受信すると直ちに次のパケットの転送を行うものとする。図中の時刻 t_0 までは、ImTCP は通常の TCP として動作している (PASS_PACKET state)。時刻 t_0 の後、ImTCP は STORE_PACKET state へ遷移し、計測用パケットをバッファに蓄積し始める。時刻 t_1 において、バッファ内のパケット数が計測ストリームを生成するためのパケット数 (5) に達したため、ImTCP は SEND_PACKET state に入り、最初の計測ストリームを送信し、STORE_PACKET state に遷移する。その後、時刻 t_2 までに、バッファ内のパケット数が再び N まで増加するため、2 番目の計測ストリームが送信される。時刻 t_2 から t_3 の間においては、パケット転送レートと ACK の受信レートが同じであるため、バッファ内パケット数は一定値を保ち、2 番目の計測ストリームの送信が終了した直後に、3 番目の計測ストリームの送信が開始される。時刻 t_4 において、計測用の 3 つのストリームの送信が終了したため、ImTCP は PASS_PACKET state へ遷移する。その時点で、パケットがバッファ内に残っているため、それらのパケットは時刻 t_5 までに高いレートで転送される。その後、ImTCP は再び通常のデータ転送を行う。

3　インラインネットワーク計測手法の実装

本章では、UDP を用いたアプリケーションレベルのプログラムの実装方法を述べ、実験ネットワークを用いた性能評価実験結果を示すことにより、提案アルゴリズムの有効性を検証する。

3.1　実装方法

UDP を用いた実装においては、UDP パケットを計測アルゴリズムに基づいて送信する送信側プログラムと、そのパケットを受信し、送信側にエコーする受信側プログラムの作成を行った。図 8 に送信側・受信側プログラムの動作概要を、表 1 に実装環境およびプログラムサイズを示す。

3.1.1　送信側プログラム

送信側プログラムの主な動作は 2.1 節で説明した計測アルゴリズムに基づいている。ここでは、実装時の問題点などについて説明を行う。

- 計測用パケットのサイズの決定方法

  本実装においては、計測用のパケットサイズをエンドホスト間のパスにおけるの最大セグメントサイズ (MSS: Maximum Segment Size) に設定する。ただし、UDP ソケットから計測パケットが通過するパスの MSS を得ることはできないため、計測を行う前に、送受信端末間に TCP コネクションを一時的に確立し、TCP コネクションの Path MTU Discovery [19] 機構によって MSS を得る。

- 計測パケット送信時刻の記録

  本実装においては、メモリ資源を節約し、計測パケットの送信間隔の管理機構をできるだけ単純化するために、計測パケットをネットワークへ送信する際に、パケットのシーケンス番号、およびパケット送信時刻をパケットのヘッダに追加してから送信を行う。送信したパケットが受信側に到着すると、受信側はパケットに含まれている情報 (シーケンス番号、送信時刻) を取り出し、これらの情報を計測パケットより小さい

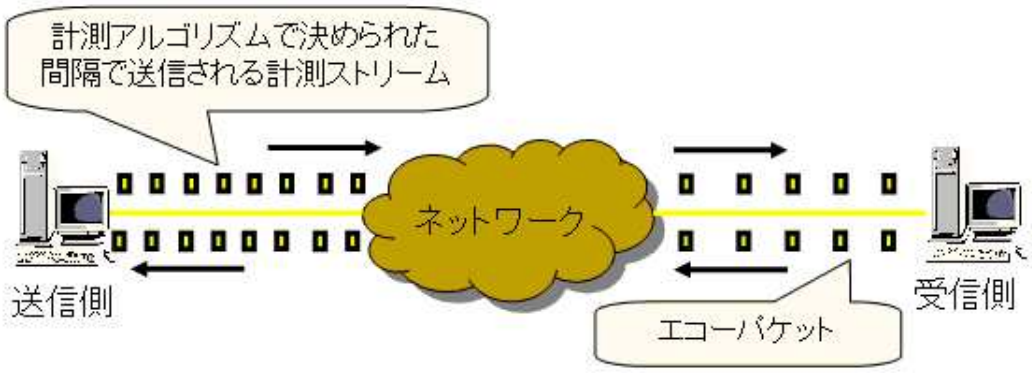


図 8: 送信側プログラム・受信側プログラムの動作

パケットに書き込み、エコーパケットとして送信側に向けて返送する 。これは、計測パケットより小さいパケットをエコーパケットとして返送することにより、送受信端末間のトラヒックを減らすことができ、ネットワーク資源を効率的に使用することが可能になるためである。

- 計測パケットの受信時刻の記録

  提案している計測アルゴリズムにおいては、少ないパケット数から構成されている計測ストリームの送受信間隔に基づいて利用可能帯域を推測するため、計測パケットの受信時刻を正確に把握することが重要である。しかし、一般的にカーネルからユーザアプリケーションにデータが受け渡される際には、大きくかつ予測できない遅延時間が発生する。そのため、パケットがアプリケーションに到着した時刻を記録すると、パケット受信時刻に大きな誤差が発生し、利用可能帯域の推定精度に大きな影響を与える。そこで、本実装においては、libpcap ライブラリ [20] を用いることにより パケットがネットワークインターフェースに到着した時刻を用いることによって、パケット到着時刻の誤差を小さくし、計測の精度を向上している。

表 1: UDP を用いた実装環境

実装したシステム	FreeBSD 4.9
使用言語	C
使用ライブラリ	libpcap
ソースコード行数	送信側:1800 行、受信側:650 行
実行形式のファイルサイズ	送信側:31Kbytes、受信側:14Kbytes

3.1.2 受信側プログラム

受信側プログラムは基本的には、受信したパケットを直ちにエコーするのみの動作を行っている。しかし、アプリケーションプログラムにおいて計測パケットの受信を検出し、その受信時刻に基づいてエコーパケットの送信を行うと、カーネルとアプリケーション間の転送遅延の影響を二度受けるために、転送遅延の変動の影響を大きく受ける。そこで本実装における受信側プログラムにおいては、送信側プログラムと同様に、libpcap ライブラリを用いて計測パケットの受信時刻を記録する。受信側プログラムは計測ストリームのパケットをすべて受け取ると、記録したパケット受信間隔に基づいて計測パケットを送信側端末に返送する。これにより、カーネルとアプリケーション間の転送遅延の影響を半減することができるため、計測精度の向上が期待される。

3.2 実装実験結果

本章では、3.1 節に示した実装方法に基づいて作成したアプリケーションプログラムを用いて、実装実験を行った結果を示し、実ネットワーク上での計測アルゴリズムの有効性を検証する。実験環境を図 9 に示す。実験環境は、100 Mbps のイーサネットによって構築され、PC ルータを介してバックグラウンドトラヒックを発生させるエンドホスト (Traffic generator)、および利用可能帯域の計測を行う送受信エンドホスト (Sender, Receiver) が接続されている。実験に用いる端末のスペックを表 2 に示す。

送信側エンドホストは、2 章で述べた計測アルゴリズムにしたがって、受信側エンドホス

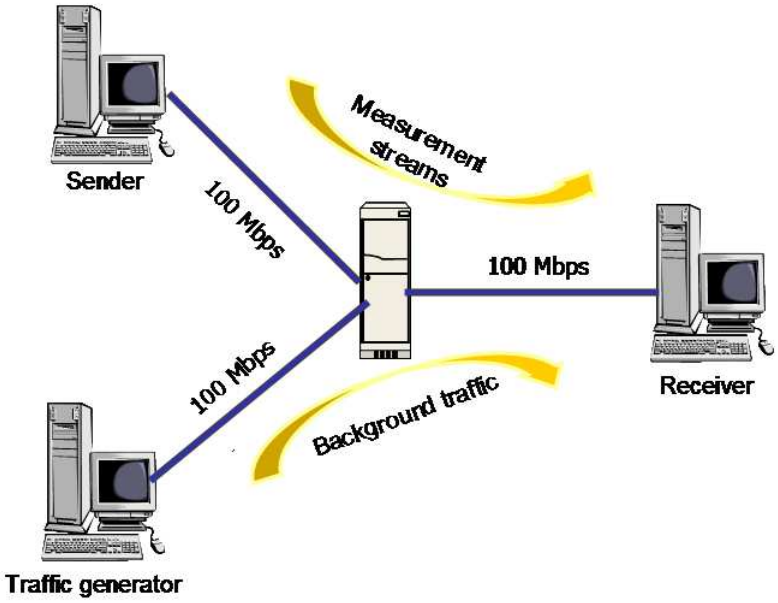


図 9: 実験ネットワーク環境

トとの間のパスの利用可能帯域を計測するため、本実験では、PC ルータと受信側エンドホスト間のボトルネックリンクの利用可能帯域を計測する。計測アルゴリズムにおける検索区間の小区間への分割数 k は、検索区間と前回の利用可能帯域の計測結果 A_{prev} の大きさに応じて以下のように決定する。

$$k = \begin{cases} 2 & (0 \leq \frac{B_u - B_l}{A_{prev}} \leq 0.15) \\ 3 & (0.15 \leq \frac{B_u - B_l}{A_{prev}} \leq 0.2) \\ 4 & (0.2 \leq \frac{B_u - B_l}{A_{prev}}) \end{cases}$$

また、検索区間の幅の最小値 B_m は、計測された利用可能帯域 A の 10% とする。計測パケットの大きさは 1500 KBytes とする。

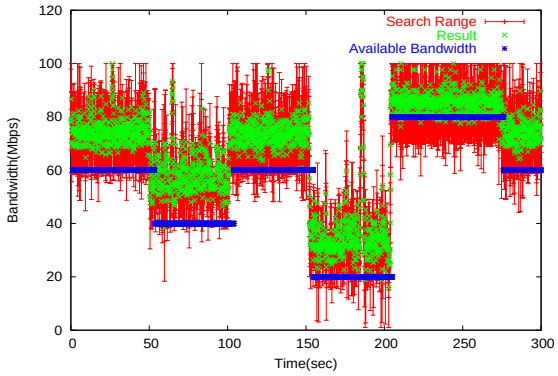
図 10 は、バックグラウンドトラヒック量を時間によって変動させることにより、ボトルネックリンクの実際の利用可能帯域 (Available bandwidth) が 0 秒から 50 秒までが 60 Mbps、50 秒から 100 秒までが 40 Mbps、100 秒から 150 秒までが 60 Mbps、150 秒から 200 秒までが 20 Mbps、200 秒から 270 秒までが 80 Mbps、270 秒から 300 秒までが 60 Mbps と変化した時の利用可能帯域の計測結果 (Result)、検索区間 (Search Range)、および利用

表 2: 実験環境を構築する PC のスペック

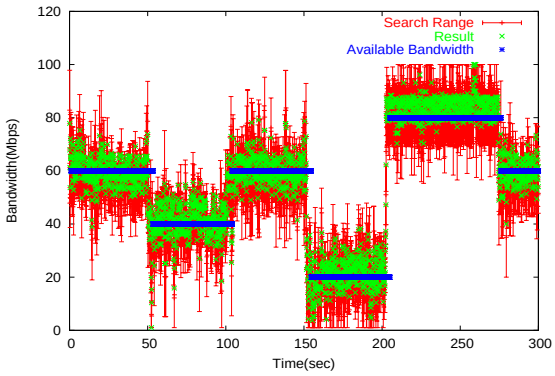
CPU	Intel Pentium 3 1.26GHz
Memory	512MB SDRAM
OS	FreeBSD 4.9

可能帯域の真の値 (Available Bandwidth) を示している。図 10(a)-(c) は、1 つの計測ストリーム内の計測パケット数 N をそれぞれ 3、5、および 8 とした時の結果である。図から、1 つの計測ストリーム内の計測パケット数が少ない場合には、計測の精度が低下し、利用可能帯域の推定がうまく行えていないことがわかる。またその反面、計測パケット数を多くすることにより、計測精度が向上しており、急激な利用可能帯域の変化がある場合にも、素早く対応して正確な測結果を導出している。これは、計測ストリーム内のパケット数が少なくなると、計測アルゴリズムのステップ (3) におけるパケット到着間隔の増加傾向の判断が困難になり、ステップ (4) における適切な小区間の選択が不正確になるためである。しかし、N が 5 以上であれば、利用可能帯域をほぼ正確に計測することができる。N を 5 から 8 に大きくすることにより計測の精度は向上しているが、本研究で提案している計測アルゴリズムは、計測の精度を向上することよりも、計測パケット数をできるだけ少なくし、計測速度、および頻度を向上させることを優先的な目的としているため、この場合においては N=5 と設定することが望ましいと考えられる。しかし、適切な計測パケット数 N は、さまざまなネットワーク状況 (利用可能帯域の大きさ、変動の大きさ、背景トラヒックの性質等) によって変化すると考えられる。適切な N の設定方法に関しては今後の課題としたい。

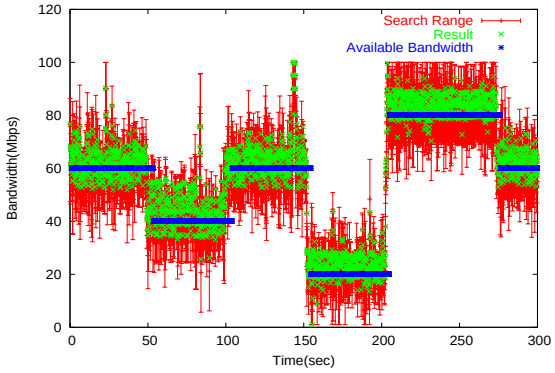
次に、利用可能帯域の大きさが、図 10 のように急激に変化するのではなく、0 sec から 50 sec までは 60Mbps で、その後 100 sec までに 40 Mbps に減少、150 sec までに 60Mbps に増加、200 sec までに 20Mbps に減少し、270 sec までに 80Mbps に増加し、300 sec までは 40Mbps である場合の計測結果を図 11 に示す。ここでも、1 つの計測ストリーム内の計測パケット数 N をそれぞれ 3、5、8 とした時の結果を示している。この場合においても、N が 3 の場合には正確な計測が行えていないが、N が 5 以上であれば、十分な精度で利用可能帯域を推定できていることがわかる。これらの結果から、本報告において実装を行った



(a) N=3



(b) N=5

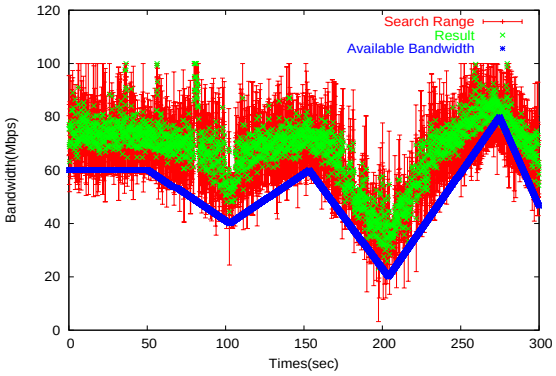


(c) N=8

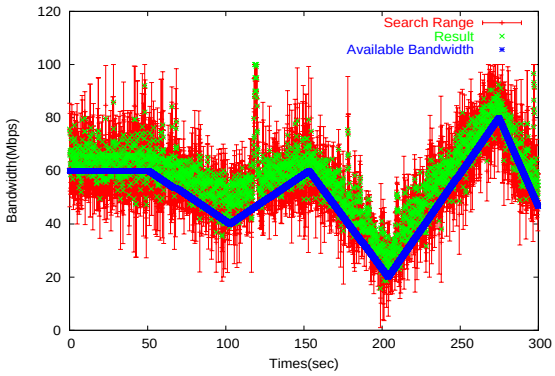
図 10: 利用可能帯域が急激に変化する場合

計測アルゴリズムは、利用可能帯域の変化の大きさに関係なく、十分な精度で利用可能帯域を推定できることがわかる。

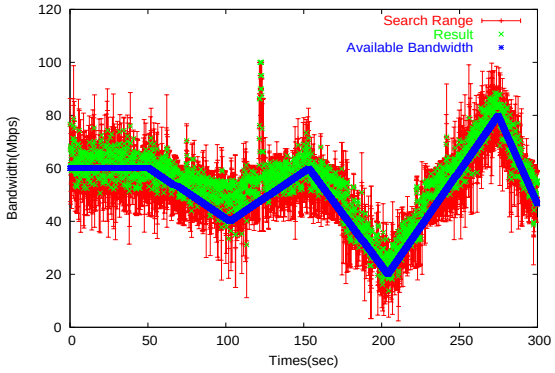
最後に、本報告における実装において考慮した点である、アプリケーションとカーネルの間の遅延の影響をできるだけ排除したことの効果を確認するそのために、libpcap ライブラリを用いず、アプリケーションプログラムにパケットが到着した時刻を用いて利用可能帯域の推測を行った場合の結果を示す。図 12 は、N=5 の時の計測結果を示す。図から、利用可能帯域が急激に変化する場合 (図 12(a))、緩やかに変動する場合 (図 12(b)) の双方において、利用可能帯域を実際の値よりも大きく推測しており、図 10、図 11 の N=5 の場合に比べて



(a) N=3



(b) N=5

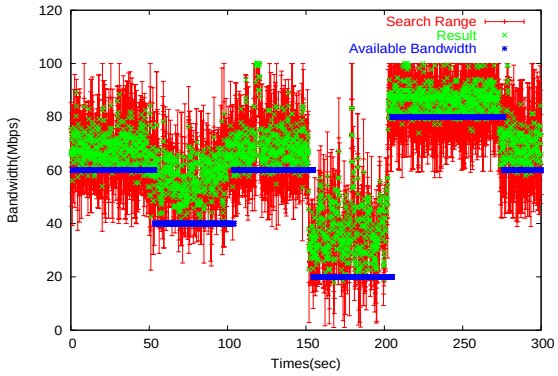


(c) N=8

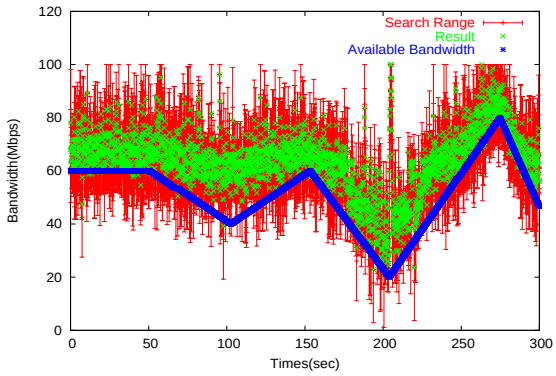
図 11: 利用可能帯域が緩やかに変化する場合

精度が著しく低下している。これは、アプリケーションプログラムにおけるパケット到着時刻が、端末のネットワークインタフェースへの到着時刻と大幅に異なることが原因である。以下に詳細な原因を示す。

パケットが端末のネットワークインタフェースに到着すると、OS の割り込みが発生してアプリケーションプログラムにパケットの到着が伝えられ、それに反応してアプリケーションプログラムがパケットをカーネルから取得する。その際、割り込みが発生してからアプリケーションプログラムがパケットの取得を開始するまでには大きな遅延時間が発生するため、パケットの取得開始時には複数のパケットが既にネットワークインターフェースに到



(a) 利用可能帯域が急激に変化する場合



(b) 利用可能帯域が緩やかに変化する場合

図 12: libpcap ライブラリを用いない場合 (N=5)

着している。その際、アプリケーションプログラムはそれらのパケットを一度に取得するため、それらのパケットの到着時刻がほぼ同時になる。つまり、ネットワークの利用可能帯域が小さいことが原因でパケット到着間隔が大きくなったとしても、計測アルゴリズムにそのことが正確に伝えられない。これは計測結果を実際よりも大きな値にする効果を持つため、図 12 に示すような計測結果を生む。それに対して本報告で作成したアプリケーションプログラムは、libpcap ライブラリを用いることによって、計測パケットおよびエコーパケットの受信時の到着間隔を正確に取得している。パケット送信時にも受信時と同様の遅延が発生していると考えられるが、上記のようにパケットを複数個同時に処理する現象が、パケット受信時に顕著に現れるため、本報告で作成したアプリケーションプログラムの計測精度が大幅に向上していると考えられる。

4 ImTCP の実装指針

本章では、インライン計測手法を TCP に組み込み、データ転送中の TCP コネクションを用いて計測を行う ImTCP 方式の実装指針と、実装に際して発生すると考えられる問題点について述べる。ImTCP 方式の概要は 2.2 節で述べたので、ここでは ImTCP 方式を FreeBSD [18] に実装する際の構成について述べる。

図 13 に、送信側端末のカーネルシステムに実装する ImTCP 方式の構成を示す。アプリケーションから Socket インタフェースを通じて渡されたデータパケットは、関数 tcp_output() によって通常 TCP のプロトコル処理を受けた後、関数 ip_output() に渡され、IP のプロトコル処理を受け、ネットワークへ送出される。2.2 節に示したように、ImTCP 方式のメカニズムは TCP コネクションのウィンドウサイズなどをパラメータとして用いるため、TCP レイヤの下部に実装されるべきである。そのため、計測プログラムは関数 tcp_output() 内に実装する。具体的には、カーネル内のメモリを用いて ImTCP バッファを作成し、TCP プロトコル処理が終了したパケットを関数 ip_output() へ渡される前に ImTCP バッファへ格納する。格納されたパケットは計測アルゴリズムに基づいたタイミングで関数 ip_output() に渡される。

計測プログラムは受信した ACK パケットの処理を行う関数 tcp_output() 内にも必要となる。すなわち、受信側端末から返送された ACK パケットは関数 ip_input() によって IP のプロトコル処理を受け、関数 tcp_output() へ渡される。その際に、パケットの到着時刻を記録し、送信側 (関数 tcp_output()) の計測プログラムへ渡される。送信側の計測プログラムは受け取った ACK パケット受信時刻を用いて、計測アルゴリズムに基づいて利用可能帯域の推測を行う。

3 章において行った UDP を用いたアプリケーションプログラムにおいては、パケットヘッダに計測パケットのシーケンス番号と送信時刻を書き込むことで、送信側端末での処理オーバヘッドを削減した。TCP を用いた計測手法である ImTCP 方式の実装においては、シーケンス番号は TCP パケットのシーケンス番号を利用し、パケットの送信時刻に関しては TCP の Timestamp オプション [17] を用いてデータパケットに書き込むことで管理を行う。

計測プログラムをカーネル内に実装することにより、本報告で実装を行ったアプリケー

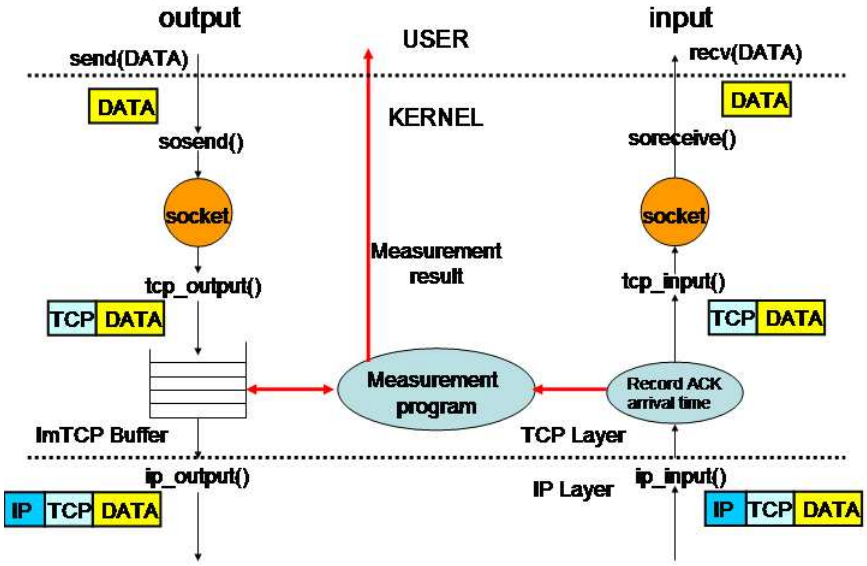


図 13: ImTCP の実装方針

ションプログラムの大きな問題点である、アプリケーションとカーネル間の通信遅延の問題を回避することができるため、計測精度の向上が期待される。しかし、カーネル内で用いるタイマの粒度は、一般的にアプリケーションが用いるタイマの粒度よりも粗い [21] ため、逆に計測精度が低下することが考えられる。これらのトレードオフに関しては、実装実験を行うことによって評価することができる。また、カーネルが libpcap ライブラリに相当するようなパケットの入出力の監視を行うための機構を提案することも、今後の課題として挙げられる。

5　おわりに

本報告では、インラインネットワーク計測アルゴリズムを実コンピュータ上に実装し、実験ネットワークを用いた性能評価を行った。実装においては、UDP を用いたアプリケーションプログラムによって計測アルゴリズムを実現し、計測アルゴリズムの基本的性質を検証した、その結果、アプリケーションレベルの実装においては、カーネルとアプリケーションの間の通信遅延が無視することができず、計測性能に大きな影響を与えることがわかった。さらに、通信遅延の影響をできる限り小さくする手法を用いることで、コンピュータ上のシミュレーションで得られた結果とほぼ同等のパケット数で、同程度の精度の計測結果が得られることが明らかになった。さらに、本報告においては、ImTCP 方式の実装に関する指針と、考えられる問題点について議論を行った。

今後の課題としては、ImTCP 方式を実装し、実験ネットワークにおいて性能評価を行うことが上げられる。また、計測アルゴリズムの更なる性能評価として、計測可能な帯域値の上限に関する評価が考えられる。これは、計測アルゴリズムはパケットの到着時刻を関することによって利用可能帯域を推測しているため、計測可能な帯域値の上限が、端末のタイマ粒度、およびパケットサイズに依存すると考えられるためである。またその際、タイマ粒度を改善することによって計測性能を向上させることは、ハードウエアの制約により困難であると考えられるため、複数のパケットをまとめて 1 つの計測パケットとみなすことで、仮想的なパケットサイズを大きくし、計測可能な帯域値を大きくする手法についての検討を行いたい。

謝辞

本報告を終えるにあたり、御指導、御教授を頂いた宮原秀夫教授に心より感謝致します。また、本報告において日頃から熱心に御指導を頂いた村田正幸教授に深く感謝致します。並びに、適切な助言を頂いた長谷川剛助教授、大阪府立看護大学医療技術短期大学の菅野正嗣助教授、サイバーメディアセンターの馬場健一助教授、大学院情報科学研究科の若宮直紀助教授、および大崎博之助教授、大阪市立大学の阿多信吾講師、大阪大学大学院経済学研究科の荒川伸一助手、および宮原研究室の牧一之進助手に心から感謝致します。最後に、ご協力を頂いた大阪大学大学院情報科学研究科情報ネットワーク専攻の宮原研究室および村田研究室の皆様に心からお礼を申し上げます。

参考文献

[1] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in paket-switched networks," Tech. Rep. TR-96-006, Boston University Computer Science Department, Mar 1999.

[2] B. Melander, M. Bjorkman, and P. Gunningburg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceeding of IEEE GLOBECOM 2000*, Nov 2000.

[3] M. Jain and C. Dovrolis, "End-to-end available bandwidth: Measurement methodology, dynamics, and relation with TCP throughput," in *Proceeding of ACM SIGCOMM 2002*, Aug 2002.

[4] S.Seshan, M. Stemm, and R. H. Katz, "SPAN: Share passive network performance discovery," in *Proceeding of 1st Usenix Symposium on Internet Technologies and Systems (USITS '97)*, pp. 135–146, Dec 1997.

[5] K. Lai and M. Baker, "Nettimer: A tool for measuring bottleneck link bandwidth," in *Proceeding of the USENIX Symposium on Internet Technologies ans Systems*, Mar 2001.

[6] A. Rao, K. Lakshminarayanan, S.Surana, R.Karp and I.Stoica, "Load balancing in structured P2P systems," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Feb. 2003.

[7] F. Dabek, B. Zhao, P. Druschel, J. Kubiatowicz and I. Stoica, "Towards a common API for structured peer-to-peer overlays," in *Proceedings of the 2nd International Workshop on Peer-to-Peer Systems (IPTPS '03)*, Feb. 2003.

[8] Czajkowski, S. Fitzgerald, I. Foster, C. Kesselman, "Grid information services for distributed resource sharing.," in *Proceedings of the Tenth IEEE International Symposium on High-Performance Distributed Computing (HPDC-10), IEEE Press*, Aug.

2001.

[9] Zhao, Y., Hu, Y., "Gress - a grid replica selection service," in *Proceedings of the 16 International Conference on Parallel and Distributed Computing Systems (PDCS-2003)*, Aug. 2003.

[10] ZB. Allcock, J. Bester, J. Bresnahan, A. L. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnal, S. Tuecke, "Data management and transfer in high performance computational grid environments," *Parallel Computing Journal*, vol. 28, May 2002.

[11] S. Vazhkudai and S. Tuecke and I. Foster, "Replica selection in the globus data grid," in *Proceedings of International Workshop on Data Models and Databases on Clusters and the Grid (DataGrid 2001). IEEE Computer Society Press, 2001*, May 2001.

[12] Akamai Home Page, *http://www.akamai.com/*.

[13] Exodus Home Page, *http://www.exodus.com/*.

[14] G. Pierre and M. van Steen, "Design and implementation of a user-centered content delivery network," in *Proceedings of the third IEEE Workshop on Internet Applications*, June 2003.

[15] J. Jha and A. Sood, "An architectural framework for management of IP-VPNs," in *Proceedings of the 3rd Asia-Pacific Network Operations and Management Symposium*, Sept. 1999.

[16] C. L. T. Man, G. Hasegawa, and M. Murata, "A new available bandwidth measurement technique for service overlay network," in *Proceeding of 6th IFTP/IEEE MMNS 2003*, Sep 2003.

[17] W. S. M. Allman, V. Paxson, "TCP Congestion Control," RFC 2581, IETF, 1999.

[18] FreeBSD Home Page, *http://www.freebsd.org/*.

[19] J. Mogule, DECWRL, and S.Deering, "Path MTU Discovery," RFC 1191, Standford University, November 1990.

[20] tcpdump/libpcap public repository, *http://www.tcpdump.org/*.

[21] M. K. McKusick, K. Bostic, M. J. Karels, and J. S. Quarterman, *The Design and Implementtation of the 4.4 BSD Operating System. Reading.* Massachusetts: Addison-Wesley, 1999.