データ転送性能評価システムPEERの構築

本庄 利守 (honjo@is.titech.ac.jp)
東京工業大学大学院情報理工学研究科

中川 晋一 (snakagaw@ncc.go.jp)
国立がんセンターがん情報研究部

大野 浩之 (hohno@is.titech.ac.jp)
東京工業大学大学院情報理工学研究科

長 健二朗 (kjc@csl.sony.co.jp)
ソニーコンピュータサイエンス研究所

概要

インターネット上では通信を行なうためにデータの転送が行なわれている。このデータ転送の性能を評価するためには、さまざまなネットワーク環境のもとで実際にデータ転送を行い、そのデータ転送の様子を観測することが効果的である。しかし、さまざまなネットワーク環境を準備することが困難であるため、利用可能なネットワークを用いた観測しかできず評価が不十分になることがある。このような問題を解決するために筆者らは、エミュレーション環境を用いてデータ転送性能を評価するPEER(Performance Evaluation Environment for Research)システムを提案する。 PEERシステムは、帯域、遅延、パケットロス、キューサイズを手軽に設定できるルータと各インターフェイスを通過したパケットに関する統計情報を収集するツールから構成される。PEERシステムを用いたデータ転送性能の評価の一例として、BSD/OS2.1 における TCPの評価を行った。特に帯域と遅延の変化によるスループットへの影響について評価した。PEERシステムを用いることによりエミュレーション環境を使用してさまざまなネットワーク環境におけるデータ転送性能の評価が可能になった。本論文では、PEERシステムの設計と実装および評価の一例について述べ、最後に今後の課題と展望について述べる。

目次

1 はじめに
2 データ転送性能の評価
3 関連研究
4 PEERシステムの設計と実装
5 PEERシステムを用いたTCPによるデータ転送性能の評価
6 考察
7 おわりに
8 謝辞
9 参考文献

1 はじめに

データ転送性能の評価には、シミュレーションによる方法もしくは実際のデータ転送を観測する方法を用いることが多い。シミュレーションによる方法は、計算機上で評価を行えるため手軽であるが、OSやアーキテクチャ、そのプロトコルの実装による影響までを考慮することができず必ずしも有効な手段とは言えない。また、実際のデータ転送を観測する方法は、データ転送性能の評価を行うためにさまざまな形態のネットワークを準備し、観測しなければならないため評価が不十分になる場合がある。そこで、筆者らは手軽な装置によりさまざまな形態のネットワークを疑似的に作り出すシステムが必要であると考えた。以下では、装置を用いてさまざまな形態のネットワークを疑似的に作り出すことをネットワークのエミュレーションと呼ぶ。

本論文では、PEER(Performance Evaluation Environment for Research) システムと呼ばれるさまざまなネットワークをエミュレートして、データ転送性能を評価するシステムを提案する。PEERシステムは、帯域、遅延、パケットロス、キューサイズを手軽に設定できるルータと各インターフェイスを通過したパケットに関する統計情報を収集するツールから構成される。

2 データ転送性能の評価

インターネットの発達と共に、さまざまな通信プロトコルやアプリケーションが提案され、実装されてきた[1][2][3]。現在、それらのデータ転送性能の向上のために、評価の方法が問題となっている。

インターネットにおける データ転送性能を決める要因として「データ転送の遅延」、「データ損失量」、「データ伝送量」が用いられることが多い。物理層の性能を評価する場合にはデータ転送容量(スループット)を使用する場合が多い。

しかし、ネットワークの高速化に伴って、さまざまなリアルタイムアプリケーションが開発されてきており、スループットだけではなく、他の要因も重要になってきている。例えば、UDP を用いて大量のデータを送る場合には大きなスループットが得られるが、データの損失も多く起こると考えられる。一方、TCP のように輻輳回避制御や再送制御を備えたプロトコルを使用する場合には、データの損失はUDP ほどは多くないと考えられる。しかし、衛星回線のような広帯域、高遅延な伝送路を使用した場合には効率的に帯域を使えないことが指摘されている[4][5][6][7]。

これらのデータ転送性能の評価は、その物理層を利用して実際に通信実験を行うか、計算機によるシミュレーションにより行われてきた。インターネットにおけるデータ転送性能の評価では、実際に接続実験を行なわなければならない。しかし、通信プロトコルやアプリケーションの開発段階において実際の回線を使用すると、転送効率などの低下がプロトコルの問題なのか回線やネットワークの転送状態の変化によるものなのかを考慮しなければならず、開発効率を下げる要因となる。また、計算機シミュレーションでは、通信プロトコルやアプリケーションの実装方法による影響やデータリンクの特性までを考慮することができず必ずしも有効な手段とは言えない。

また、新しく開発される各種プロトコルやアプリケーションの性能の評価において、データ転送性能の評価を決める標準的な指標が存在しておらず、開発者にとってそのプロトコルやアプリケーションがどのような特性を持つかを評価することが困難である。

以上のような理由から著者らは、インターネットにおけるデータ転送性能の評価を行うシステムが必要であると考えた。そこで、さまざまなネットワーク環境を手軽に実現でき、かつ各種データ転送性能の評価が可能なPEERシステムを提案する。

3 関連研究

データの転送性能を評価する方法には、シミュレーションによる方法、実測による方法、エミュレーションによる方法の 3 通りが考えられる。

シミュレーションによる方法を採用した実装には S.McCanneらによるns[8]やS.Keshavら

によるREAL[9]などがある。シミュレーションによる方法は、意図する環境を作ることが容易であることや測定者の提案するモデルなどを適用することが容易であることなどが利点として挙げられる。しかし、既存のシミュレータでは、データリンクの特性やアーキテクチャの特性などの細かな部分まではシミュレートできていないという欠点も挙げられる。これらのことからデータ転送性能の評価方法としてシミュレーションによる方法は必ずしも有効な手段であるとは言えない。

実測による方法を採用した実装にはT.C.Slatteryによるttcp、R.Jonkmanによるnetspec[10]、Hewlett-PackardによるNetperf[11]、村山らによるDBS[12]などがある。また、新しいプロトコルを開発し評価するためのプラットホームであるL.S.Brakmoらによるx-kernel[13]と呼ばれるシステムもある。実測による方法は、実際に測定したいOSやアプリケーションを使えるということが利点として挙げられる。また、実際に使用しているネットワークを用いて観測することができるため、他のトラフィックの影響も観測することができる。しかし、測定をおこなうために実際のネットワークを使用しなければないため、さまざまな環境におけるデータ転送を観測をすることが困難であるという欠点が挙げられる[14]。

エミュレーションによる方法は、実際に測定したいプロトコルやOS、アプリケーションの実装を使えることやさまざまなネットワーク環境をパラメータ等を設定するだけで実現できることなどが利点としてが挙げられる。また、実際のアプリケーションを使用することによって簡単にトラフィックを発生させることが可能であり、実際のトラフィックによる影響を観測することも容易である。

以上からプロトコルやアプリケーションの開発、評価の段階においてそれぞれの評価方法を適宜使い分けることが必要であると考えられる。筆者らは、エミュレーション環境を用いてデータ転送性能を評価する方法は、全ての実装を同一環境で測定し、比較することができるため有効な手段の一つであると考える。

4 PEERシステムの設計と実装

本章では、PEERシステムの設計理念などについて述べる。

4.1 概要

PEERシステムは、さまざまなネットワーク環境をエミュレートし、エミュレートした各ネットワーク環境において実際にデータ転送を行い、それを観測することによってデータ転送性能を評価するシステムである。PEERシステムの特徴は以下の通りである。

- 帯域、遅延、パケットロス、キューサイズのそれぞれをエミュレートできる。また、それらのパラメータを簡単に変更できる。
- データ転送における各種統計情報を収集できる。
- データ転送の様子をリアルタイムに可視化できる。

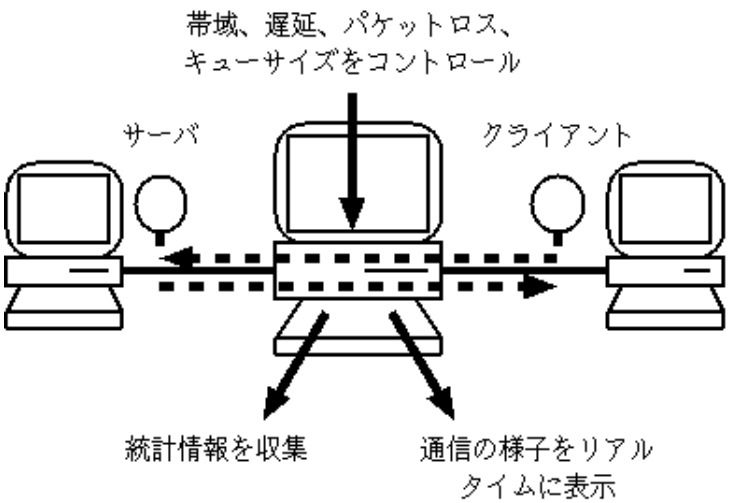

図1:PEERシステムの構成

本システムの構成を図1に示す。PEERシステムは2つ以上のネットワークインターフェイスを持ったPCから構成される。以下では、このPCをPEERルータと呼ぶ。データ転送性能を評価したいプロトコルを実装した 2台の計算機を用意して、PEERルータを介して通信が行えるように設定する。そして、実現したい環境をPEERルータ上で設定した後に通信実験を行い、PEERルータから観測した情報を取り出す。

以下では動作原理、実装および評価について述べる。

4.2 動作原理

PEERシステムの動作原理について述べる。PEERシステムは各インターフェイスにおける帯域、遅延、パケットロス、キューサイズのそれぞれをエミュレートするために、dummynet[15]を参考にして各インターフェイスごとに図2のような 2 つのキューを使用した。 以下では、このキューをそれぞれLimit_Queue、Delay_Queueと呼ぶ。

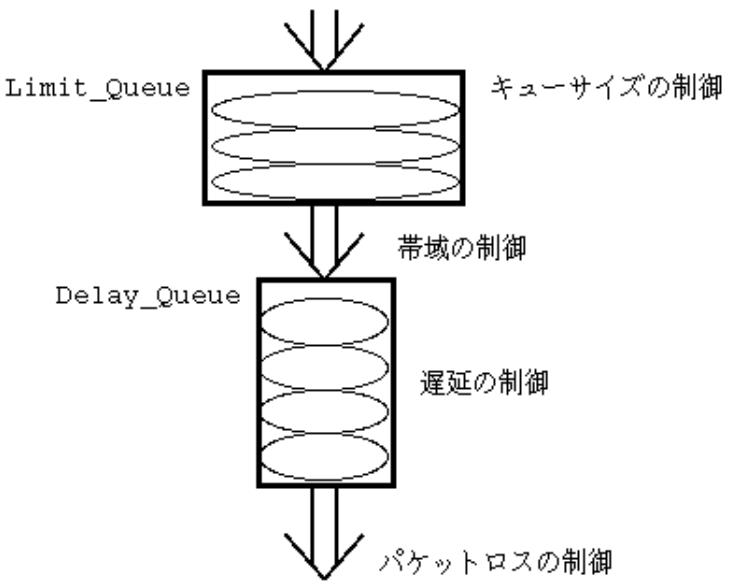

図2:PEERシステムの動作原理

- まずPEERルータはパケットを受け取ると経路表を参照して、出力すべきインターフェイスを決める。そして、そのパケットを該当するインターフェイスのLimit_Queueに入れる。Limit_Queueでは、ルータにおけるキューの大きさをエミュレートしている。もし、Limit_Queueに指定されたサイズより多くのパケットが入れられる場合には、そのパケットは破棄される。
- 次にパケットをLimit_QueueからDelay_Queueに移動する。この際に帯域の制御を行っている。Limit_QueueからDelay_Queueへのパケットの移動は一定時間をおいて定期的に行い、移動するパケットの量は設定された帯域から決める。例えば、帯域を *x*Kbytes/secに制限するために、*y*msecごとのタイマを使用する場合には一回の移動に *x*/(*y**1000)KBytesのパケットを移動する。
- Delay_Queueでは、遅延の制御を行っている。Delay_Queueでは、設定された時間の間、パケットを保持する。このようにすることによって遅延のエミュレートを行っている。最後にDelay_Queueから出力する際にパケットロスをエミュレートする。

以上のような動作原理を用いて各インターフェイスごとに帯域、遅延、パケットロス、キューサイズをエミュレートしている。

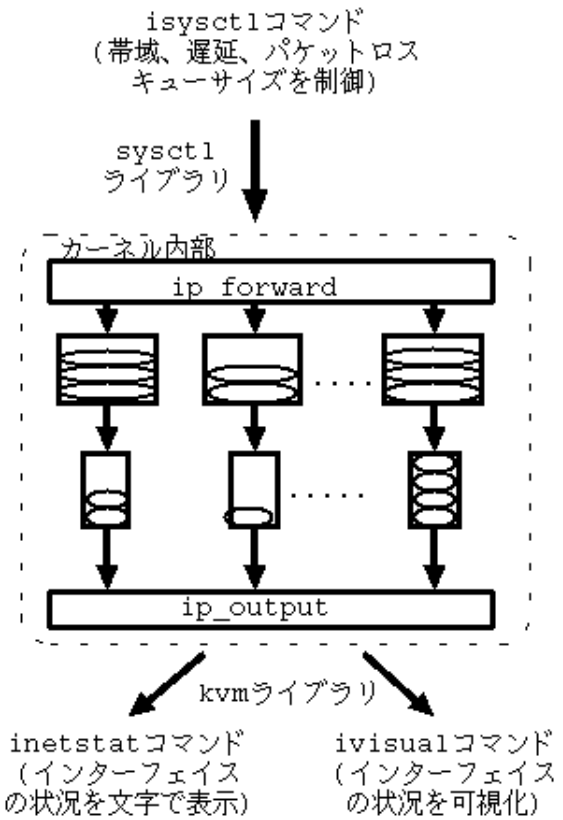

図3:PEERシステムの全体像

4.3 実装

実装は、オペレーティングシステムにBSD/OS 2.1を用い、IBM/PC互換機上で行った。PEERシステムの全体像を図3に示す。 以下では、カーネルへ行った変更および各種パラメータの設定方法、統計情報の収集方法について述べる。

4.3.1 カーネルの変更

BSD/OS 2.1 では、IPパケットを処理する部分を初期化するために関数ip_init()を実行する[16]。PEERシステムでは、関数ip_init()の内部で図2に示したようなキューを各インターフェイスごとに用意するように変更を加えた。

次に、パケットの転送処理する部分に変更を加えた。BSD/OS 2.1では図4 に示すように、インターフェイスから渡されたパケットを関数ip_intr()に渡す。ここではパケットヘッダを見て自分宛のパケットかどうかを判断し、自分宛であればトランスポート層の処理部分に渡し、自分宛でなければ関数ip_forward()に渡し、転送処理を行う。関数ip_forward()は渡されたパケットを出力すべきインターフェイスを決めた後に、しかるべきインターフェイスに出力する。PEERシステムでは、関数ip_forward()から関数ip_output()に引き渡す際に、前述のキューにパケットを入れるように変更した。

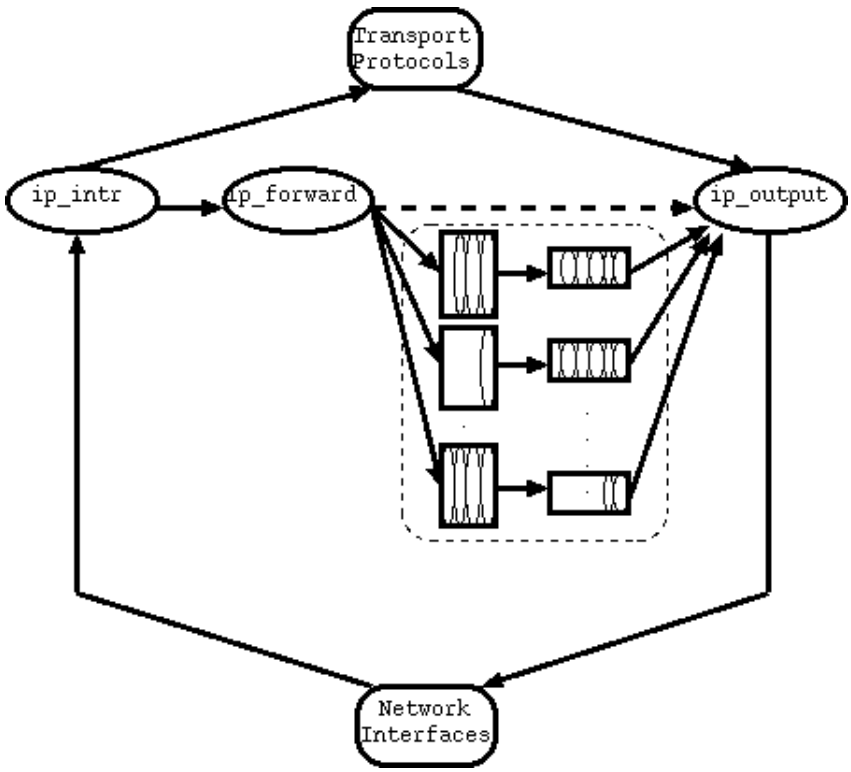

図4:パケットの流れ

関数ip_forward()からLimit_Queueに入れる際に、現在Limit_Queueに存在するパケットの数を確認して、設定された数より大きくなるようであれば、そのパケットを破棄する。そうでなければLimit_Queueに入れる。次に、Limit_Queueに蓄えたパケットを定期的にDelay_Queueに移動する。この際に動作原理のところで述べたように、一回に移動するパケットの量を帯域とタイマーから計算する。これを実現するためにタイマー関数である関数timeout() を使用している。このため帯域制御の精度の限界は、このタイマー関数の精度に依存している。そして、Delay_Queueにおいて指定した時間だけパケットを保持することによって遅延を生じさせている。この遅延の制御にも前述のタイマー関数を使用しているため、精度はタイマー関数に依存している。最後に、Delay_Queueから関数ip_output()に渡す。この際にパケットロスをエミュレートしている。

4.3.2 各種パラメータの設定方法

各インターフェイスの設定を行うためにisysctlコマンドを用意した。このコマンドはPEERルータをコントロールするためのコマンドであり、統計情報の初期化なども行う。このコマンドはsysctlライブラリ[17]を使用している。

4.3.3　統計情報の収集

　各インターフェイスの情報を取り出すためにinetstatおよびivisualコマンドを用意した。このコマンドはkvmライブラリ[17]を使用して、各種の統計情報をカーネル内部から読み出す。
　各インターフェイスの統計情報をキャラクタで見るためにはinetstatコマンドを利用する。 inetstatコマンドの出力例を図5に示す。

```
# ./inetstat -i 3
=======================<link 3>=======================
Total Through Packets           795             packets
Queue Length Average            3.652198        pieces
Queue Length Varience           1.813926
Queue Length Maximum            7               pieces
Total Packet Bytes              1167044         bytes
Packet Delay Average            77.544655       ms
Packet Delay Varience           1174.892090
Packet Delay Maximum            192.000000      ms
Total Drop Packet               45              packets
Total Drop Bytes                63652           bytes
Drop Burst Average              2.000000        packets
Drop Burst Varience             0.000000
Drop Burst Maximum              2               packets
=======================<link 3>=======================
#
```

図5:inetstatコマンドの出力例

　また、各インターフェイスを通過するパケットの様子をリアルタイムに観測するにはivisualコマンドを利用する。 このコマンドでは、kvmライブラリの他に、Tcl/TkおよびTkの拡張ライブラリであるBLTライブラリを使用した。ivisualを実行した様子を図6に示す。

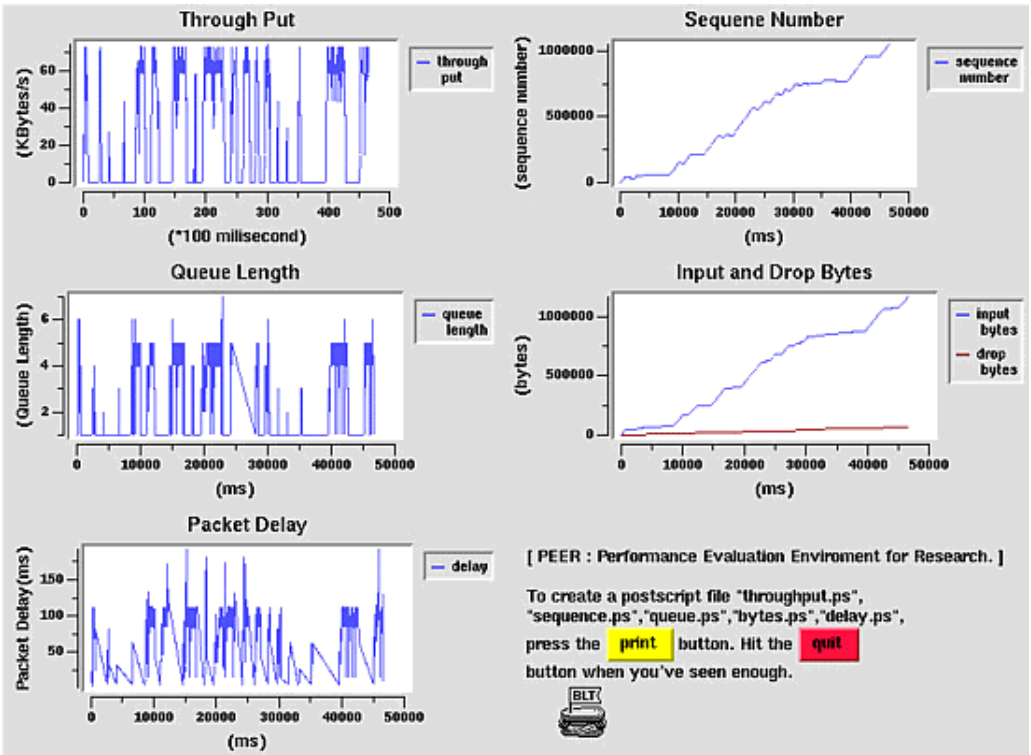

図6:ivsualコマンドによる表示例

4.4 評価

PEERシステムの評価を行なった。評価実験に用いた実験環境を図7に示す。PEERルータには、CPUにPentium133MHzを使用し、メモリを40MB搭載したラップトップPCを用いた。Host1はCPUにPentium100MHz、Host2はCPUに486DX2 50MHzを使用したラップトップPCを用いた。今回の評価実験では、特にPEERルータの帯域制御機能および遅延制御機能の評価を行なった。

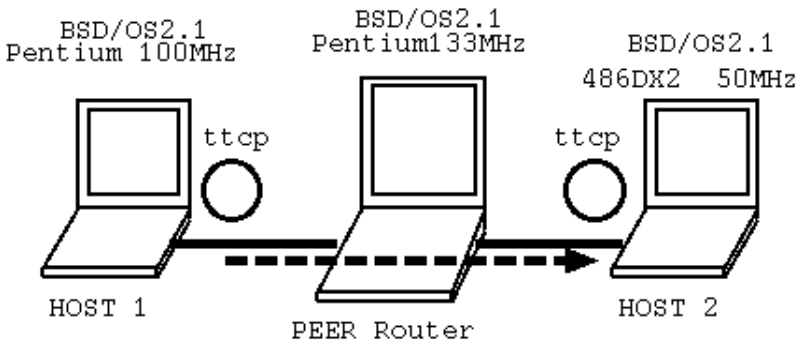

図7:実験環境

帯域制御機能の評価ではttcpを用いてデータ転送を行ない、そのトラフィックを観測した。各インターフェイスにおいて帯域を設定した後に、ttcpのTCPのバッファサイズを1024Bytes、メッセージサイズを1024Bytesに設定し、1MByteのデータ転送を行なった。各インターフェイスにおいて帯域を設定する際に、遅延、パケットロス、キューサイズはそれぞれ0ms、0%、99個に設定した。測定結果を図8に示す。

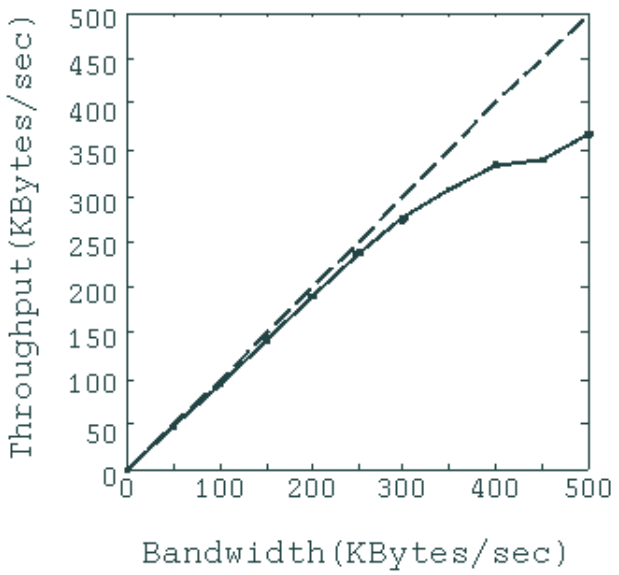

図8:帯域制御機能

また、PEERシステムにおいて帯域を最大に設定した場合およびPEERシステムを使用しない場合の比較も行なった。PEERシステムにおいて帯域、遅延、パケットロス、キューサイズをそれぞれ999KBytes/sec、0ms、0%、99個に設定した場合は369KBytes/secであり

、PEERシステムを使用しない場合は、430KBytes/secであった。

遅延制御機能の評価ではpingコマンドを用いてRTTを測定した。各インターフェイスにおいて遅延を設定した後に、pingコマンドを用いてRTTを測定した。各インターフェイスにおいて遅延を設定する際に、帯域、パケットロス、キューサイズはそれぞれ999KBytes/sec、0%、99個に設定した。測定した結果を図9に示す。

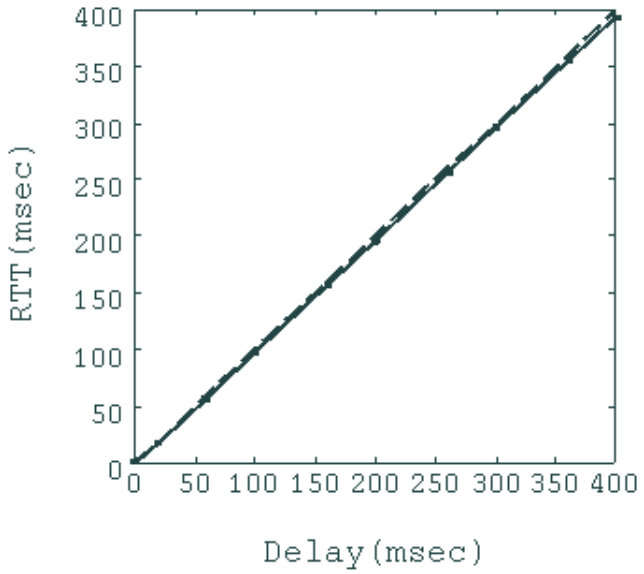

図9:遅延制御機能

以上の測定結果から帯域制御機能および遅延制御機能の両者とも十分に機能していることが分かる。これらの機能の実現にはカーネルのタイマーを使用しているため10msec未満の精度によるパケットの制御はできない。このためPEERシステムの限界はハードウェアの性能の他にもカーネルのタイマーに依存している。

5 PEERシステムを用いたTCPによるデータ転送性能の評価

PEERシステムを用いたデータ転送性能評価の一例として、今回はBSD/OS2.1におけるTCPの性能評価実験を行った。本実験は**4.4**と同様の実験環境で行なった。本実験ではttcpを用いてデータ転送を行いトラフィックを観測した。TCPのバッファサイズを1024Bytes、メッセージサイズを1024Bytesに設定し、1MByteのデータ転送を行なった。

今回の評価実験では、以下の 2 点に関してBSD/OS2.1におけるTCPのデータ転送性能を評価した。

1. 衛星回線のような広帯域、高遅延の回線を使用した場合のデータ転送
2. 帯域、遅延の変化によるスループットへの影響

5.1 広帯域、高遅延の回線におけるデータ転送

衛星回線のエミュレートをするために各インターフェイスにおいて帯域を2Mbps、遅延を250msecに設定した。スループットの変化を図10、シーケンスナンバーの変化を図11に示す。

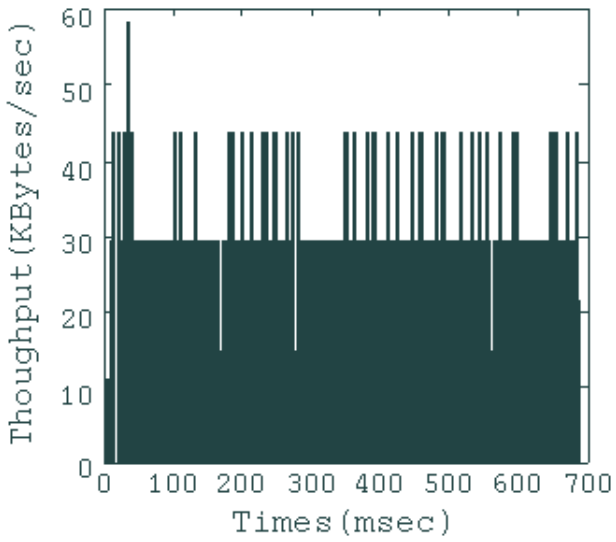

図10:スループットの変化

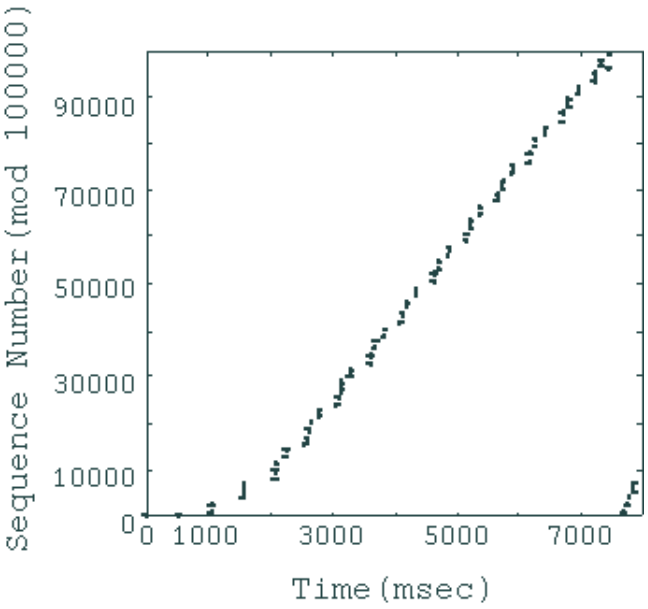

図11:シーケンス番号の変化

文献[5][6]で指摘されているように高遅延のためにTCPのウィンドウサイズが十分に大きくならないため、帯域を十分に使用できていないことが分かる。

5.2 帯域、遅延の変化によるスループットへの影響

衛星回線をエミュレートした実験を受けて、さらに帯域と遅延の変化によるスループットへの影響について詳しく調べた。測定した結果から得られた帯域、遅延、スループットの関係を図12に示す。また、特に帯域2Mbpsにおけるスループットと遅延の関係を図13に示す。

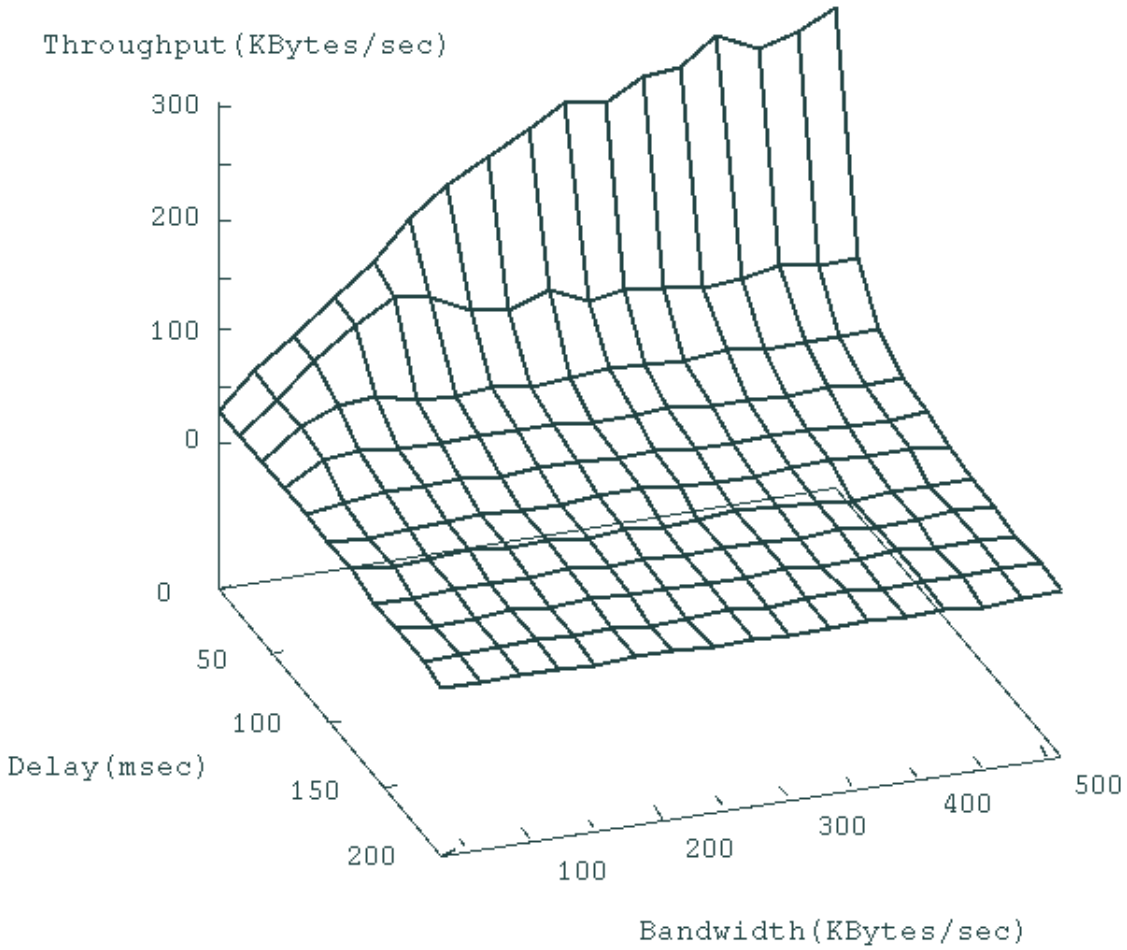

図12:帯域、遅延、スループットの関係

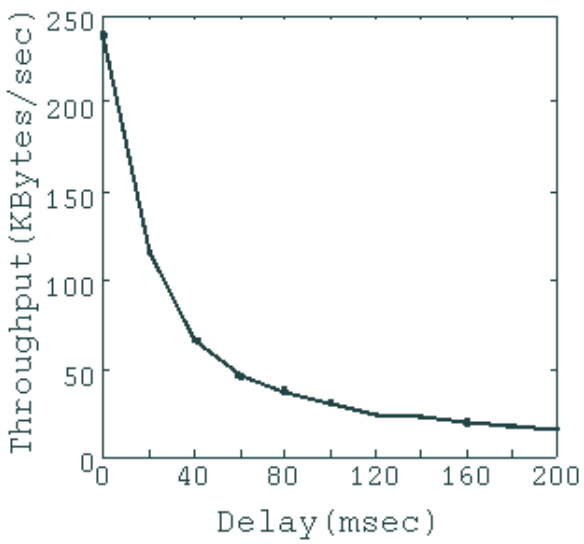

図13:帯域2Mbpsにおけるスループットと遅延の関係

以上の測定結果からBSD/OS2.1におけるTCPによるデータ転送では帯域の変化よりも遅延の変化の方がスループットに大きな影響を与えることが分かる。次世代TCP では、さま

ざまな状況に対応して、最大のスループットが得られるようにすることが望まれる。今後は、さらにさまざまなOSにおけるプロトコルの実装やTCP Vegas[18] のような新しい実装に関しても比較検討をしなければならない。

6 考察

6.1 ネットワークのエミュレーション

データ転送性能をエミュレーション環境を利用して評価するシステムとして、実際のネットワークにできるだけ近い環境をエミュレートすることは重要である。今回のPEERシステムの実装では、ネットワークの形態を定量的に表すパラメータである帯域、遅延、パケットロスのそれぞれをエミュレートすることが可能である。

しかし、実際のネットワークに近い環境をエミュレートするために、さらにこれらのパラメータの連携までを考える必要がある。例えば、実際のネットワークでは、遅延とパケットロスには密接な関係があり、パケットロスが発生する前に次第に遅延が大きくなることが観測されている。今後は実際のネットワークに近い環境をエミュレートできるようにエミュレーション方法を検討していく必要がある。

6.2 データ転送性能評価の比較方法

PEERシステムを用いることによりさまざまなネットワーク環境におけるデータ転送性能を評価することが可能になった。現在のPEERシステムでは、各実装のさまざまなネットワーク環境におけるデータ転送を観測するだけに留まっている。しかし、今後はさらに各実装の評価を比較できるようにすることが必要である。主に以下の点を考慮して、各実装の評価を定量的に行えるようにしたい。

1. 利用可能な帯域をどれだけ効率的に使えているか。
2. 帯域を越えた量のパケット送出によるパケットロスの発生がいかに少ないか。

さらに、各実装のデータ転送性能を評価するために測定の基準を決める必要がある。そこで、測定の基準となるネットワーク環境を決め、さらに測定の基準となる測定ツールを提供できるようにしたい。

7 おわりに

本論文では、データ転送性能の評価の必要性について述べ、さまざまなネットワークをエミュレートして、データ転送性能の評価するPEERシステムを提案し、実装および評価を行なった。今後は、考察で述べたように、エミュレーション方法の検討および評価の比較方法の検討を行いPEERシステムをデータ転送性能の評価システムとして、有用なものにする予定である。さらに、さまざまなOSやアプリケーションの実装を比較検討する予定である。PEERシステムを用いることにより、より良いデータ転送性能を持ったプロトコルやアプリケーションの実装が促されることを望む。

8 謝辞

本研究を進めるにあたり、多くのコメントを頂いた神戸大学総合情報処理センターの岡村耕二氏、電気通信大学総合情報処理センターの小林克志氏、東京工業大学工学部の山岡克式氏、情報技術開発株式会社の力武健次氏およびWIDEプロジェクトのメンバーに感謝する。

9　参考文献

[1] Henning Schulzrinne, "RTP: The real-time transport protocol", MCNC 2nd Packet Video Workshop, vol. 2, Research Triangle Park, North Carolina December 1992.
[2] H. Schulzrinne, "QoS for real-time services: playout delay and application control", in Proc. of 46th RACE Concertation Meeting (RCM), March 1995.
[3] S. McCanne and V. Jacobson, "vic: A flexible framework for packet video", in Proc. of ACM Multimedia ’95, ACM, November 1995.
[4] Mark Allman, Chris Hayes, Hans Kruse and Shawn Ostermann, "TCP Performance over Satellites Links", Proceedings of the Fifth International Conference on Telecommunications Systems,Nashville, TN, March 1997.
[5] WIDEプロジェクト, "1994年度WIDEプロジェクト研究報告書", WIDE, 1994.
[6] 西田佳史,峯尾淳一,寺岡文男,中村修, "Wishにおけるインターネットトラフィックの解析", 情報処理学会第49回(平成6年後期)全国大会論文集(1),情報処理学会, September 1994.
[7] V. Jacobson, R. Braden and D. Borman, "TCP Extensions for High Performance",Request for Comments 1323, May 1992.
[8] A.Heybey, "The network simulator", Technical report,MIT, September 1990.
[9] S.Keshav, "REAL: A Network Simulator", Technical report88/472,UC Berkeley, 1988.
[10] Roel Jonkman, "NetSpec: A Network Performance Evaluation and Experimentation Tool", http://www.tisl.ukns.edu/Projects/AAI/products/netspec/, 1995.
[11] Hewlett-Packard Company, "Netperf: A Network Performance Benchmark Revision 2.1", 1995.
[12] 村山公保,門林雄基,山口英, "TCP性能評価システムDBSの構築", インターネットコンファレンス’96論文集, July 1996.
[13] N.C.Hutchinson, L.L.Peterson, "The x-kernel: An Architecture for Implementing Network Protocols", IEEE Transactions on Software Engineering17(1),IEEE Press, january 1991.
[14] 加藤紀康,鎌形映二,伊藤隆治,芹澤陸, "マルチメディア非対称無線ネットワーク(AWL)システムの試作と評価", マルチメディア,分散,協調とモーバイルワークショップ論文集,情報処理学会, July 1997.
[15] Luigi Rizzo, "Dummynet: a simple approach to the evaluation of network protocols", Dipartimento di Ingegneria dell’Informazione, Universita di Pisa, September 1996.
[16] Gary R. Wright, W. Richard Stevens, "TCP/IP Illustrated, Volume2, The Implimentation ",Addison Wesley, 1996.
[17] Marshall Kirk McKusick, Keith Bostic, Michael j. Karels and John S. Quarterman, "The Design and Implementation of the 4.4BSD Operating System", Addison Wesley, 1996.
[18] L.S.Brakmo, S.W.O’Malley and L.Peterson, "TCP Vegas: New Techniques for Congestion Detection and Avoidance", Proceedings of SIGCOMM’94, August 1994.