社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

スループット保証を実現するTCPの輻輳制御方式の実装評価

山根木果奈† 　長谷川　剛†† 　村田　正幸†

† 大阪大学 大学院情報科学研究科　〒 565-0871 大阪府吹田市山田丘 1-5
†† 大阪大学 サイバーメディアセンター　〒 560-0043 大阪府豊中市待兼山町 1-32
E-mail: †{k-yamanegi, murata}@ist.osaka-u.ac.jp, ††hasegawa@cmc.osaka-u.ac.jp

あらまし　動画像のストリーミング配信や VoIP などのリアルタイム配信型アプリケーションの普及が近年めざましく進んでいる。高い通信品質の確保を必要とするこれらのアプリケーションに対して、我々の研究グループではトランスポート層プロトコルである TCP の制御によって一定のスループットを上位アプリケーションに提供する手法を提案している。これまでの研究では、シミュレーションによって提案手法の性能を明らかにし、その結果ネットワークに空き帯域がほとんどない状況においても、物理帯域の 10–20 %のスループットを高い確率で獲得できることを示した。本稿では、提案手法を Linux OS 上に実装し、実機を用いた実験によって提案手法の実網における性能を評価する。その結果、提案手法が所望の TCP スループットを高い確率で獲得できることを示し、実網においても高い有効性があることを明らかにする。

キーワード　TCP (Transmission Control Protocol), スループット保証, 輻輳制御方式, 実装

Implementation and Evaluation of TCP Congestion Control Mechanism for Predictable Throughput

Kana YAMANEGI†, Go HASEGAWA††, and Masayuki MURATA†

† Graduate school of Information Science and Technology, 1-5 Yamadaoka, Suita, Osaka, 565-0871, Japan
†† Cybermedia Center, Osaka University, 1-32 Machikaneyama, Toyonaka, Osaka, 560-0043, Japan
E-mail: †{k-yamanegi, murata}@ist.osaka-u.ac.jp, ††hasegawa@cmc.osaka-u.ac.jp

Abstract　Recently, real-time media delivery services such as video streaming and VoIP have rapidly become popular. For these applications requiring high-level QoS guarantee, our research group has proposed a transport-layer approach to provide predictable throughput for upper-layer applications. In the previous paper, we have evaluated the performance of the proposed mechanism through simulation experiments and confirmed that the proposed mechanism can provide required throughput for TCP connections if the required throughput is not so large compared to the physical bandwidth, even when competing traffic occupies almost all of the bottleneck link bandwidth. In the present paper, we implement the proposed mechanism and evaluate it in the actual network . We investigate the performance through the experiments both in the experimental network and the commercial Internet environment, and validate that the proposed mechanism can achieve the required throughput with high probability as is the simulation results.

Key words　TCP (Transmission Control Protocol), throughput guarantee, congestion control mechanism, implementation

1. はじめに

ADSL や FTTH などの広帯域アクセス網技術の進展に伴い、多様化・高度化されたアプリケーションが増加している。例えば VoIP やテレビ会議システムなどのリアルタイム配信型アプリケーションの普及が進み、特に動画像のストリーミング配信などは近年ユーザ数が急増している。これらのアプリケーションが円滑に動作するためには、一定のネットワーク品質が必要となる。しかし現在のベストエフォート型のネットワークではネットワーク状況がアプリケーションの性能に大きく影響するため、高いネットワーク品質を確保する技術が必要である。

ネットワーク品質を実現する手法の一つとして、IntServ [1] や DiffServ [2] などの IP 層における技術が挙げられるが、これらの手法はフローが通過するすべてのルータに品質制御機能が実装されている必要があり、ネットワーク規模に対するスケーラビリティ、導入コストなどの面から実現は困難であると指摘されている。一方、動画像ストリーミングなどのアプリケーションでは、トランスポート層プロトコルとして User Datagram Protocol (UDP) を用い、転送速度などの制御はアプリケーションが行うことによって、安定したアプリケーション品質を確保

する手法が存在する。しかしこれらの手法は、アプリケーションの特性によって制御手法を変更する必要があるため実装の非効率性が問題として指摘される。また、異なるアプリケーションが発生させるトラヒック間の相互干渉や公平性などを考慮することができないため、複数のネットワークアプリケーションが同時に使用される状況下では、アプリケーションの品質が不確定的になる。

以上のように、IP 層あるいはアプリケーション層において通信品質を維持・向上させることには、さまざまな問題点が存在する。そこで我々の研究グループにおいては、トランスポート層プロトコルを用いることで様々な通信品質を実現することに着目している。その一つとして、一定のスループットを TCP の制御によって獲得する手法を提案している [3]。本来 TCP は、ラウンドトリップ時間 (Round Trip Time: RTT)、パケット廃棄率、および競合するフロー数などによって、そのスループットが大きく影響を受ける [4] ため、一定のスループットを 100 % 確実に獲得することはできない。しかし、アプリケーション品質を維持するために必要となるスループットを高い確率で獲得することができれば、アプリケーション品質の向上が期待できると考えられる。提案手法では、TCP の輻輳制御方式のうち、ウィンドウサイズの増減アルゴリズムを変更し、送信側 TCP のみの変更によって、アプリケーションに必要とされるスループットを高い確率で獲得することを実現している。本手法を用いることによって、例えば、Windows Media Player [5] や Real One Player [6] など、TCP トラヒックをバッファリングすることによって動画像ストリーミングを行うアプリケーションの品質を大幅に向上させることができると考えられる。

提案手法は、我々の研究グループで提案している Inline Measurement TCP (ImTCP) [7] によって計測された利用可能帯域に関する情報を利用することによって、ネットワークの輻輳状態を考慮した制御を行う。ImTCP が行うインラインネットワーク計測は、送信側ホストで設定したデータパケットの送信間隔に対して、その ACK パケットの到着間隔の変化を観察することによって利用可能帯域を計測する手法である。ImTCP は、TCP コネクションがデータ転送に用いるデータパケットと ACK パケットのみを用いてネットワークの利用可能帯域を計測するため、計測用パケットを必要としない。さらに、ImTCP の機構は TCP 層の最下層部分に実装され、輻輳制御などの TCP 本来の機能からは分離することができる [8] ため、TCP の他の機能に影響を与えることなく、利用可能帯域を計測する機能を TCP に追加することができるという特徴を持つ。したがって提案手法においても、ImTCP アルゴリズムを用いて利用可能帯域に関する情報を得ることができる。

[3] では、提案手法を ns-2 [9] を用いたシミュレーションによって評価している。その結果、背景トラヒック量が多く、ネットワーク中の空き帯域がほとんどない環境においても、提案手法がアプリケーションから指定されたスループットを高い確率で獲得できることを明らかにした。本稿では、提案手法の実装を行い、実機を用いた実装実験によって実ネットワークにおける提案手法の有効性を明らかにする。本稿では提案手法を Linux 2.6.16.21 のカーネルシステムに実装を行う。

実験は Dummynet [10] を用いて構成される実験ネットワーク環境よび大阪-東京間の実インターネット環境において実験を行う。提案手法を用いるコネクションは動画像ストリーミングアプリケーションを想定した無限長データの転送を行い、アプリケーションから指定された時間間隔で要求スループットを達成できるか否かを評価する。

本稿の構成は以下のとおりである。2 章では提案手法の説明を行い、3 章では提案手法を Linux 2.6.16.21 のカーネルシステムに実装する際の指針について述べる。4 章では実機を用いた実験によって提案手法の性能評価を行い、最後に 5 章でまとめと今後の課題を述べる。

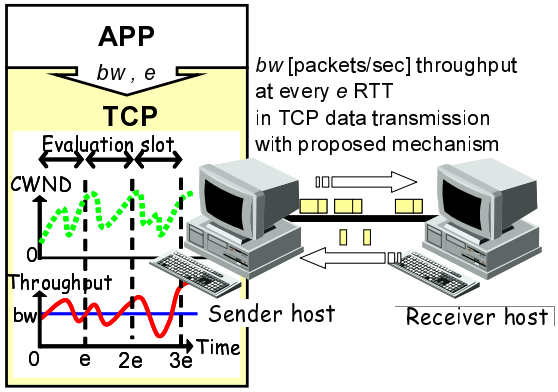

図 1　提案手法の概要

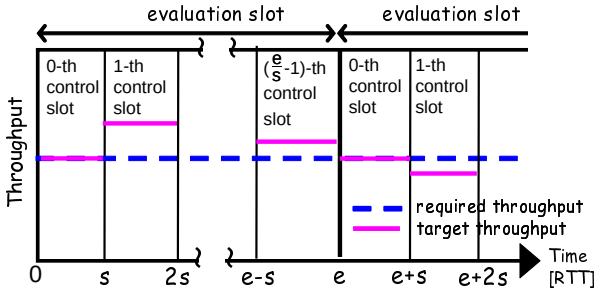

図 2　評価スロットと制御スロット

2. 提 案 手 法

図 1 は、本提案手法の概要を示したものである。提案手法においては、上位アプリケーションから目標スループット (required throughput) および時間間隔が指定されることを想定している。ここで目標スループットを bw (packets/sec)、時間間隔を e (RTT) とすると、提案手法は e RTT ごとの平均スループットが目標スループット bw 以上になることを目的としている。指定された時間間隔 e は評価スロット (evaluation slot) と呼ばれ、さらに評価スロットを複数の制御スロット (control slot) に分割する (図 2)。つまり制御スロット長を s とすると、$2 \leqq s \leqq e$ である。提案手法は、制御スロットごとにスループットの目標値 (target throughput) g を設定し、その値に基づいて輻輳ウィンドウサイズの増加量を動的に変動させることによってデータ転送速度を制御する。評価スロットよりも細かな時間間隔である制御スロットをサイクルとした制御を行うことによって、評価スロット終了時にその期間の平均スループットが bw 以上になることを意図している。

提案手法は、ACK パケットが到着するたびに次式の k に適切な値を代入することによって輻輳ウィンドウサイズの値を決定する。

$$cwnd \leftarrow cwnd + \frac{k}{cwnd} \tag{1}$$

k に代入する値は次のように決定する。i 番目の制御スロット (以降、制御スロット i と呼ぶ) のスループットの目標値を g_i と表す。また制御スロット i が開始してから j 番目の ACK パケットが返ってきたとき、制御スロット i の平均スループットが g_i となるための輻輳ウィンドウサイズの増加量を k_j^{bw} と表す。k_j^{bw} は、スループットの目標値の獲得に必要なパケット数を制御スロット内で全て送信できるように以下の式を用いて計

算する。

$$k_j^{bw} = \frac{2\{g_i \cdot srtt_i \cdot s - a_j - (s - n_j - 1)cwnd_{n_j}\}}{(s - n_j - 1)(s - n_j)} \qquad (2)$$

ここで $srtt_i$ は、制御スロット i が開始したときに TCP が持つ RTT の指数移動平均である smoothed RTT (sRTT) の値であり、また a_j は、現在の制御スロットが開始してから j 番目の ACK パケットが返ってくるまでに送信したパケット数である。また j 番目の ACK パケットが返ってきたとき、現在の制御スロットが開始してから n_j RTT 経過しているとすると、$cwnd_{n_j}$ は、現在の制御スロットが開始してから n_j RTT 経過したのときの輻輳ウィンドウサイズの値を示す。

提案手法は、(1) 式における k の値に関して、下限値 k_{min} および上限値 k_{max} を設定する。これはネットワークが十分空いている場合にも、提案手法が TCP Reno と同等のスループットを獲得することを可能する一方で、ネットワークが混雑している場合に、輻輳ウィンドウサイズの増加量が極度に大きく設定されることによる性能低下を防ぐためである。すなわち輻輳ウィンドウサイズの下限値 k_{min} は、TCP Reno が輻輳ウィンドウサイズの増加量として採用している 1 (packet) に設定し、また上限値 k_{max} は、ImTCP アルゴリズムによって計測された利用可能帯域の値に基づき、ネットワークの空き帯域を考慮した値に設定される。

以上をまとめると、(1) 式の k は k_j^{bw}、k_{min} および k_{max} を考慮して以下の式により決定される。

$$\begin{cases} \text{if } \{(k_j^{bw} < k_{min}) \text{ or } (k_{max} < k_{min})\} \ k \leftarrow k_{min} \\ \text{elseif } \{k_j^{bw} > k_{max}\} \ k \leftarrow k_{max} \\ \text{else } \ k \leftarrow k_j^{bw} \end{cases}$$

また上記の式において、下限値 k_{min} の設定を行わなければ、ネットワークの空き帯域が目標スループット以上の場合には、目標スループット以上のスループットを獲得しない制御を行うことができる。これは観測データや財務データなどの定常的に一定レートでデータが発生するようなアプリケーションに適していると考えられる。提案手法の詳細は関しては [3] を参照されたい。

3. 提案手法の実装

本章では、提案手法を Linux 2.6.16.21 カーネルシステムに実装する際の実装指針ついて述べる。図 3 に送信端末のカーネルシステムに実装する提案手法の構成を示す。通常、アプリケーションから Socket インター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 TCP のプロトコル処理を受けた後、関数 ip_output() によって IP のプロトコル処理を受け、ネットワークへ送出される。一方、受信側から到着した ACK パケットは関数 ip_input() によって IP 処理を受け、関数 tcp_input() に渡される。輻輳ウィンドウサイズは、関数 tcp_input() 内で TCP プロトコル処理を行うときに更新されるため、輻輳制御を行う提案手法のプログラムは関数 tcp_input() 内に実装する必要がある。Linux 2.6.16 では、輻輳制御機構のためのインターフェースを統一し、輻輳制御アルゴリズムをモジュール化して組み込むことを可能にしている。今回は Linux 2.6.16.21 を用い、提案手法をモジュールとして実装する。

関数 tcp_input() は、ACK パケットを受け取るたびに関数 cong_avoid() を呼び出し、輻輳ウィンドウサイズを更新する。提案手法モジュールはこの関数内でアルゴリズムに基づいて輻輳ウィンドウサイズの増加量を計算し、輻輳ウィンドウサイズを決定する。また ACK を受け取った時間を監視し、時間を評価/制御スロットに分割する。一方 ImTCP が計測する利用可能帯域の値は、関数 tcp_input() において算出される [8]。提案手法は ImTCP が利用可能帯域の計測に成功するたびにその値を受け取り、輻輳ウィンドウサイズの増加量の上限値を変更する。

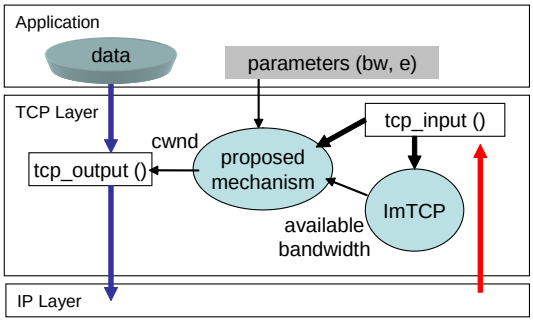

図 3　提案手法の構成図

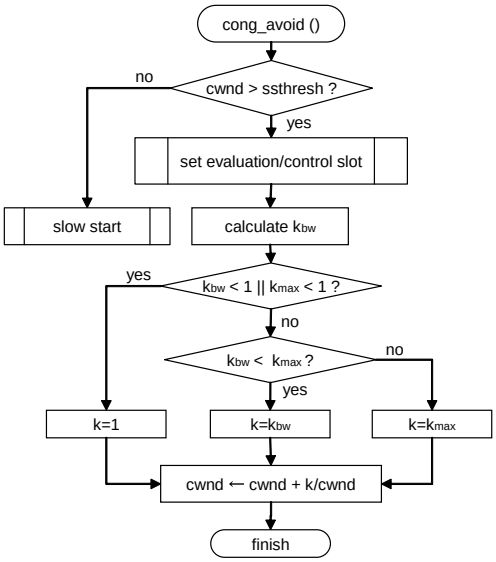

図 4　関数 cong_avoid() のフローチャート

図 4 は、提案手法モジュールにおける関数 cong_avoid() のフローチャートを示している。関数 cong_avoid() が呼び出されると、はじめに $cwnd$ と $ssthresh$ を比較する。その結果 $ssthresh$ の方が大きければ、TCP Reno と同様にスロースタートフェーズの制御を行い、提案手法による制御は輻輳回避フェーズのみで動作する。輻輳回避フェーズでは、まずスロットの経過時間を調べ、スロットの終了判断を行う。現在の評価/制御スロットが開始してからの経過時間が評価/制御スロット長以上であれば、スロットにおける平均スループットを算出し、次のスロットのために各変数を初期化する。次に (2) 式に基づいて k_{bw} を計算し、k_{max} および k_{min} の値を考慮して 2 章で説明したアルゴリズムに従って輻輳ウィンドウサイズの増加量を決定する。最後に、決定した輻輳ウィンドウサイズの増加量を用いて (1) 式によって $cwnd$ の値を更新し、関数を終了する。

4. 性能評価

本章では、提案手法を 3 章に基づいて実装した実機を用いて実験ネットワーク環境および実インターネット環境下において実験をすることにより、提案手法の実ネットワーク上における有効性を評価する。

4.1 実験ネットワーク環境における評価

図 5 に実験に用いたネットワーク環境を示す。実験ネットワーク環境は、100 Mbps のイーサネットによって構築され、Dummynet がインストールされた PC ルータを介して提案手法

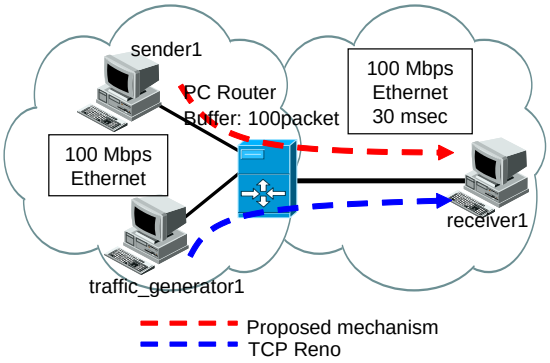

図 5　実験ネットワーク環境

表 1　実験ネットワーク環境実験に使用した PC の性能

	sender1	traffic_generator1	receiver1
CPU	Pentium 4 1.90GHz	Pentium 4 1.7GHz	Xeon 2.80GHz
Memory	1024MB	2048MB	2048MB
Kernel	Linux 2.6.16.21	Linux 2.6.16.21	Linux 2.6.16.21

を用いるエンドホスト (sender1)、クロストラヒックを発生させるエンドホスト (traffic_generator1)、それぞれのホストからパケットを受信するエンドホスト (receiver1) が接続されている。PC ルータと受信ホスト間の伝播遅延時間は、Dummynet によって 15 msec に設定している。実験ネットワークを構築する各ホストの性能を表 1 に示す。

提案手法のパラメータとして、$e = 32$、制御スロット長 s の初期値は e の 1/2 である 16 と設定する。また、背景トラヒックを発生させる traffic_generator1 の TCP ソケットバッファを制限し、1 本の TCP コネクションの最大スループットが約 4 Mbps となるように設定する。また提案手法を用いる sender の *HZ* パラメータは 20000 に設定している。

4.1.1　輻輳ウィンドウサイズとスループットの変化

ここでは、背景トラヒック量の変化に対する提案手法の挙動を評価する。提案手法のパラメータ bw (アプリケーションから指定されたスループット値) を物理帯域の 20 %に相当する 20 Mbps と設定し、提案手法を用いるコネクションを 1 本用いる。また、背景トラヒック量を変動させるために、データ転送開始後、20 秒ごとに traffic_generator1 が発生させる TCP Reno を用いるコネクション数を 5、25、40 と変化させる。図 6 に、輻輳ウィンドウサイズおよび制御スロット長 s の変動を、図 7 に評価スロットごとの平均スループットの変動を示す。

図 6 より、TCP Reno のコネクション数が 5 本であるデータ転送開始後 20 秒間において、提案手法は TCP Reno のような速度で輻輳ウィンドウサイズを増加させ、かつ図 7 より、獲得したスループットはアプリケーションから要求されたスループットを大きく超えていることが分かる。この場合、物理帯域が 100 Mbps であるネットワークに最大スループットが 4Mbps に制限された背景トラヒックが 5 本存在する状況であるため、アプリケーションから要求されたスループットを獲得するためには十分な空き帯域が存在しているといえる。そのため提案手法は、輻輳ウィンドウサイズの増加量を最小値である k_{min} (=1) に設定してデータ転送を行っている。

また、TCP Reno のコネクション数が 25 本であるデータ転送開始後 20-40 秒の期間は、0-20 秒の期間と比較して輻輳ウィンドウサイズの増加速度が大きくなっていることが分かる。これは背景トラヒック量が増加したことによって、TCP Reno と同じ挙動をすると指定されたスループットを達成することができないために、輻輳ウィンドウサイズの増加速度を増加させているためである。

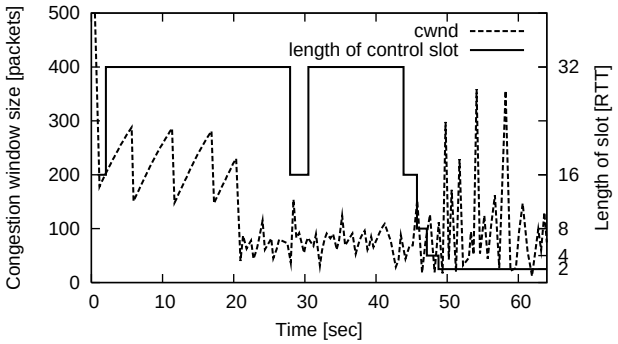

図 6　輻輳ウィンドウサイズと制御スロット長 s の変化

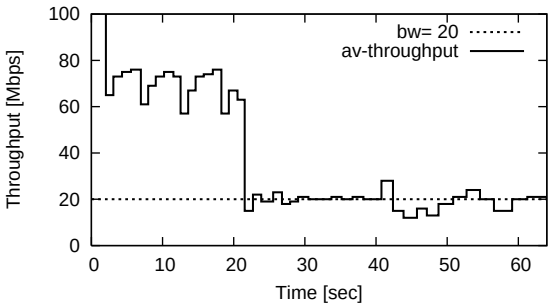

図 7　スループットの変化

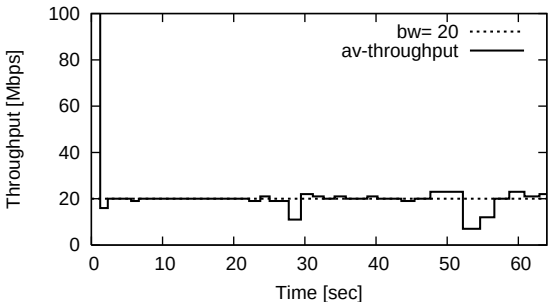

図 8　輻輳ウィンドウサイズおよびスループットの変化

さらに TCP Reno のコネクション数が 40 本であるデータ転送開始 40 秒以降は、提案手法の輻輳ウィンドウサイズの増加速度がさらに速くなっていることが分かる。またこの時図 6 より、制御スロット長 s が小さくなっていることが分かる。これは、輻輳ウィンドウサイズの増加量を大きくするだけでは要求されるスループットに到達できないため、制御スロット長を短くして制御を細かく行うためである。この制御によって、背景トラヒックとして 40 本の TCP Reno コネクションが存在する環境下においても、ほぼ要求されたスループットを獲得できている。これらの結果から、提案手法は、背景トラヒック量に応じて適切にウィンドウサイズの増加速度および制御スロット長を変化させ、指定されたスループットを確保できることが示された。

次に、2 章で述べたように提案手法が輻輳ウィンドウサイズの下限値 k_{min} を設定しない場合について同様に実験を行う。図 8 に評価スロットごとの平均スループットの変動を示す。これよりネットワークが十分空いている 0-20 秒の期間においても、提案手法が目標スループット以上のスループットを獲得していないことが分かる。また背景トラヒックが増加する実験開始後 20 秒以降も、安定して目標スループットを獲得している。これより、k_{min} を設定しない場合は常に目標スループットを獲得する制御を行うことができることを示した。

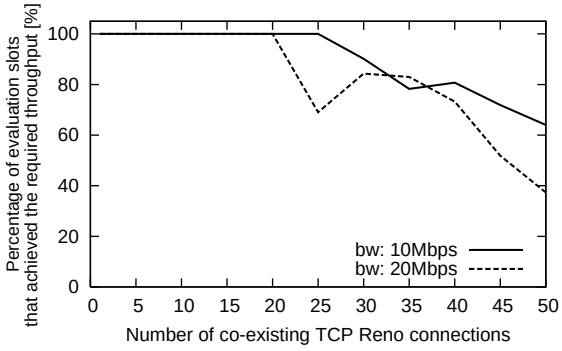

図 9　目標スループットを獲得できた割合

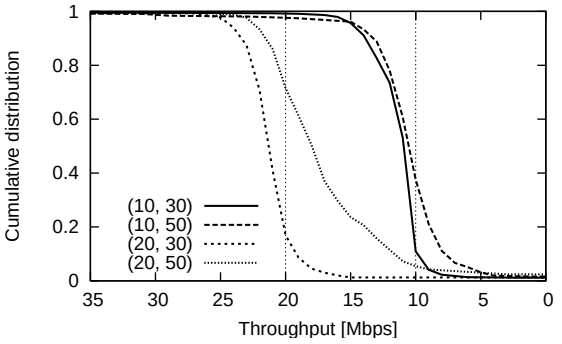

図 10　実験ネットワークにおける平均スループットの累積密度関数

4.1.2　目標スループットを獲得できる割合

ここでは、背景トラヒック量を変動させたときの、提案手法が指定されたスループットを獲得できる確率を評価する。提案手法のパラメータとして $bw = 10, 20$ Mbps とし、提案手法を用いるコネクションがネットワーク中に 1 本存在するときについて実験を行う。図 9 は、背景トラヒックを生成する TCP Reno コネクション数に対する、指定されたスループットを獲得できた評価スロット数の割合を示したものである。なお本図は、120 秒の実験を 5 回行ったときの平均値をプロットしている。

図 9 より、背景トラヒックのコネクション数が増加しても、提案手法は高い割合で帯域の 10 %および 20 %に相当する帯域を獲得できていることが分かる。これは [3] で示したシミュレーションによる評価と同等の結果である。しかし目標スループットを獲得できる割合が一時低下している場合があることが分かる。これは、提案手法においては制御スロット長を倍増/半減させるというように大きく変化するため、ネットワーク状況によっては適当な制御スロット長の値が見つからないことが原因として考えられる。そのため制御スロット長が頻繁に変更され、制御が安定しないことで目標スループットを獲得できる割合が低下している。この問題に対しては、制御スロット長の制御アルゴリズムの再検討を今後の課題としたい。

次に、提案手法のパラメータ bw を $10, 20$ Mbps、背景トラヒックのコネクション数を 30、50 本としたそれぞれの場合について、評価スロットにおける平均スループットの累積密度関数を示す (図 10)。

図 10 より、背景トラヒックが 30 本の場合は、ほとんどの評価スロットにおいて目標スループット以上のスループットを獲得していることが分かる。一方、背景トラヒックが 50 本の場合は、背景トラヒックが 30 本の場合と比較して目標スループットを獲得した評価スロットが少ないことが分かる。しかし、評価スロットにおいて獲得したスループットが目標スループット近くに集中し、目標スループットに到達できなかった多くの評価スロットにおいても、目標に近いスループットを獲得できていることが分かる。したがって、提案手法は目標スループットを獲得できた割合が低い場合においても、ほぼ全ての評価スロットにおいて目標スループットに近いスループットを獲得することができることが明らかになった。

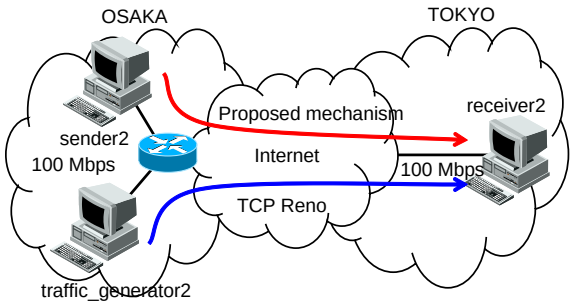

図 11　インターネット環境

表 2　インターネット実験に使用した PC の性能

	sender2	traffic_generator2	receiver2
CPU	Pentium 4 3.40GHz	Xeon 3.60GHz	Xeon 2.66GHz
Memory	1024MB	2048MB	1024MB
Kernel	Linux 2.6.16.21	Linux 2.6.17	Linux 2.4.21

4.2　インターネット環境における評価

次に東京-大阪間の通信回線 (図 11) を用いてデータ転送実験を行う。この通信回線は、インターネットを介して提案手法を用いるエンドホスト (sender2) および背景トラヒックを発生させるエンドホスト (traffic_generator2)、それぞれのホストからパケットを受信するエンドホスト (receiver2) が接続され、各エンドホストのアクセスリンクは 100 Mbps のイーサネットによって構築されている。sender2 および traffic_generator2 は大阪側に、また receiver2 は東京側にそれぞれ接続されている。各エンドホストの性能を表 2 に示す。以降の実験では、東京側のエンドホストを受信ホストとして、大阪側のエンドホストを送信ホストとして用いる。インターネット回線では正確な帯域を特定することはできないが、本実験で用いた通信回線は最大で 70 Mbps 程度の値を示すことが多く、また ping コマンドによって得られた往復伝播遅延時間は約 17 msec であった。

提案手法のパラメータとして、$e = 32$ とし、また制御スロット長 s の初期値は e の 1/2 である 16 とする。また、背景トラヒックを発生させる traffi_ generator2 の TCP ソケットバッファを制限し、1 本の TCP コネクションの最大スループットが約 3 Mbps となるように設定する。sender2 および receiver2 の TCP ソケットバッファは十分大きい値に設定している。

まず、輻輳レベルが変化したときの提案手法の挙動について評価を行う。提案手法のパラメータ bw (アプリケーションから指定されたスループット値) を物理帯域の 20 %に相当する 14 Mbps、提案手法を用いるコネクションは 1 本とする。また、背景トラヒックの大きさを変動させるために、データ転送開始後、20 秒ごとに traffic_generator2 が TCP Reno を用いるコネクション数を 0、5、25、40 と変化させる。図 12 に、輻輳ウィンドウサイズおよび制御スロット長 s の変動を、図 13 に評価スロットごとの平均スループットの変動を示す。

図 12 より、ネットワークに 1 本の提案手法を用いるコネクションを張っているデータ転送開始後 20 秒間は、評価スロットごとの平均スループットが約 70Mbps になっていることが分かる。この結果から、本実験ネットワークの帯域が約 70 Mbps であることが分かる。またデータ転送開始後 20 秒以降は、図 12、13 より、4.1.1 章で示した結果と同等の結果が得られていることが分かる。すなわち 20-40 秒間は、ネットワーク帯域が十分

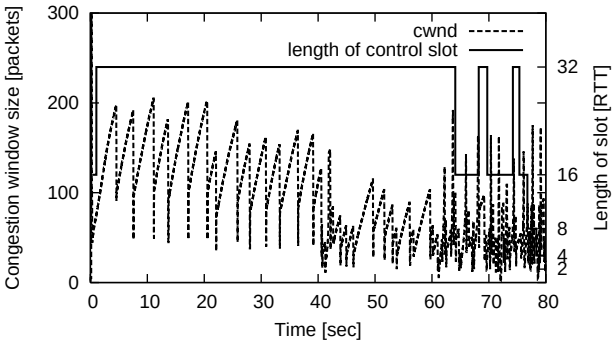

図 12　インターネット実験における輻輳ウィンドウサイズと制御スロット長 s の変化

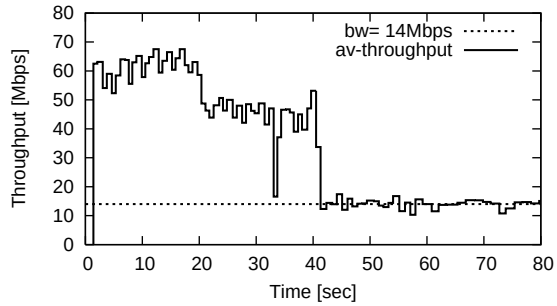

図 13　インターネット実験におけるスループットの変化

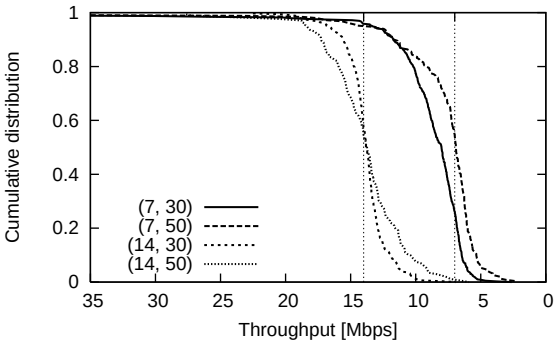

図 14　インターネット実験における平均スループットの累積密度関数

空いているため TCP Reno と同様の挙動で目標スループット以上のスループットを獲得し、データ転送開始後 40 秒以降は輻輳ウィンドウサイズの増加量を大きくすることによって目標スループットに近いスループットを獲得している。さらにデータ転送開始後 60 秒以降は、さらに背景トラヒック量が増加するため、提案手法は制御スロット長 s の値を小さくし、制御を細かくすることで結果として目標スループットを獲得することに成功している。これより提案手法はインターネット環境においても、ネットワークの輻輳レベルに応じて動的に制御を行うことで目標スループットを獲得できることが示された。

次に、評価スロットにおける平均スループットの評価を行う。提案手法のパラメータ bw を 7, 14 Mbps とし、提案手法のコネクションを 1 本用いる。実験は背景トラヒックのコネクション数を 30、50 本としたそれぞれの場合について、120 秒のデータ転送を 5 回行う。図 14 は、評価スロットにおける平均スループットの累積密度関数を示している。

図 14 と図 10 と比較すると、インターネット環境における結果の方が目標スループットを獲得できる割合が低いことが分かる。これはインターネット環境において発生する web トラヒックなどの、short-lived トラヒックが提案手法に影響を与えるためであると考えられる。しかし図 14 より、目標スループット付近で累積密度の変化が大きくなっていることから、提案手法はほぼ全ての評価スロットにおいて目標スループット、もしくはそれに近いスループットを獲得することに成功している。したがって提案手法は、インターネット環境においても安定した確率で目標スループットを獲得できることが明らかになった。

5. おわりに

本稿では、我々の研究グループがこれまでに提案した一定のスループットを TCP の制御によってアプリケーションに提供する手法の実装を行った。実ネットワークを用いた実装実験を通して、提案手法が実ネットワーク環境においても高い確率で一定のスループットを獲得できることが明らかとなり、シミュレーションによる評価結果と同等の有効性を実証した。

今後の課題としては、制御スロット長の変動アルゴリズムを改良することでより安定した性能保持を目指すことや、動画像のストリーミングアプリケーションを用いて実証実験を行うことにより、提案手法の実用性を示したい。

謝　辞

本研究の一部は、総務省戦略的情報通信研究開発推進制度委託研究「ネットワークサービスの早期展開を実現するオーバレイネットワーク基盤の研究開発」によっている。ここに記して謝意を表す。

文　献

[1] J. Wroclawski, “The use of RSVP with IETF integrated services,” *RFC 2210*, Sept. 1997.

[2] S. Blake, D. Black, M. Carlson, E. Davies, Z. Wang, and W. Weiss, “An architecture for differentiated service,” *RFC 2475*, Dec. 1998.

[3] K. Yamanegi, G. Hasegawa, and M. Murata, “Congestion control mechanism of TCP for achieving predicatable throughput,” in *Proceedings of ATNAC 2006*, pp. 117–121, Dec. 2006.

[4] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: a simple model and its empirical validation,” in *Proceedings of ACM SIGCOMM’98*, Sept. 1998.

[5] Microsoft Corporation, “Microsoft Windows Media - Your Digital Entertainment Resource,” available from `http://www.microsoft.com/windows/`.

[6] RealNetworks Corporation, “Rhapsody & RealPlayer,” available from `http://www.real.com/`.

[7] C. L. T. Man, G. Hasegawa, and M. Murata, “A simultaneous inline measurement mechanism for capacity and available bandwidth of end-to-end network path,” *IEICE Transactions on Communications*, vol. E89-B, pp. 2469–2479, Sept. 2006.

[8] T. Tsugawa, G. Hasegawa, and M. Murata, “Implementation and evaluation of an inline network measurement algorithm and its application to TCP-based service,” in *Proceedings of NOMS 2006 E2EMON Workshop 2006*, Apr. 2006.

[9] T. V. Project, “UCB/LBNL/VINT network simulator - ns (version 2),” available from `http://www.isi.edu/nsnam/ns/`.

[10] L. Rizzo, “IP_DUMMYNET,” available from `http://info.iet.unipi.it/~luigi/ip_dummynet/`.