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の輻輳制御方式の提案と評価

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 インターネットの発展によりサービスが多様化し、リアルタイム配信型アプリケーションなど、通信品質の確保を必要とするアプリケーションが注目されている。これまで、IP 層やアプリケーション層において高い通信品質を提供する手法が提案されているが、ネットワーク規模に対するスケーラビリティや導入コストなどの問題から実現が困難とされている。本稿では、トランスポート層において高い通信品質を実現する一方式として、TCP コネクションを用いてある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、TCP の輻輳制御方式を提案する。提案手法は、送信側 TCP の輻輳ウィンドウサイズの増加方法を変更することで、データ送信レートを制御する。提案手法の評価はシミュレーションによって行い、その結果、背景トラヒック量が多く、利用可能帯域がほとんど存在しない環境においても、物理帯域の約 10-20%のスループットを高い確率で獲得できることを示す。また、提案手法を用いた TCP コネクションが複数本混在する状況下では、それらのコネクションを統合的に処理する必要があることを示す。

キーワード TCP (Transmission Control Protocol), スループット保証, 輻輳制御方式, シミュレーション

Congestion control mechanism of TCP for achieving 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, users' demands for diversified services have increased due to the rapid development of the Internet. As a result, applications requiring QoS (Quality of Service) guarantee such as a real-time media delivery service become popular. Our research group has proposed transport-layer approaches to provide such high-level quality of network services to upper-layer applications. In this paper, we propose the congestion control mechanism of TCP for achieving predictable throughput. It controls data transmission rate by changing the increasing degree of congestion window size. We show the evaluation results of the proposed mechanism by simulation that we can achieve a TCP throughput of about 10-20% of the physical bandwidth, even when there is almost no available bandwidth on the end-to-end network path of the connection. Moreover, we indicate that we should introduce an additional mechanism when multiple connections utilizing the proposed mechanism are parallelized in the network.

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

1. はじめに

ADSL や FTTH といった広帯域アクセス網技術の進展により、近年ますますインターネットが発展し、サービスの高度化・多様化によって通信品質に対する要求が高まっている。その中でも、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発展がめざましいが、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。例えばパケット伝送遅延や遅延ジッタは、実時間ストリーミング配信アプリケーションなどの品質に大きく影響を与える。また音声や映像などのリッチコンテンツを効率よく配信するためには、一定のスループットを提供することが重要である。

インターネットにおけるデータ転送において、そのような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。例えば

DiffServ は、サービスの種類によってルータにおけるパケット処理の優先順位を決定することによって、ルータを通過する各フローの通信品質の差別化を行うことを目的としている。しかし、IntServ や DiffServ を実現するためには、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考えられる。

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、トランスポート層のプロトコルとして 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り、非効率的であると言える。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さまざまな通信品質を実現することを提案している。例えば [3] では、競合する他のトラヒックに影響を与えずに、ネットワークの空き帯域を利用してをデータ転送を行う、TCP によるバックグラウンド転送手法を提案している。Transmission Control Protocol (TCP) [4] に代表されるトランスポート層プロトコルは、ネットワークの輻輳状態などに応じてデータ転送速度を調整するフロー制御の役割を担っている。したがって、ネットワーク層プロトコルとの役割分担やアプリケーションからの利用容易性を考慮すると、高い通信品質を提供するための機能をトランスポート層において実現することは理想的であるといえる。またその際、TCP を改変してそれらの機能を実現することによって、TCP を用いる既存アプリケーションを容易に収容することができる。

そこで本稿では、そのような高い通信品質の一種として、一定のスループットを TCP の制御によって獲得することに着目する。本来 TCP は、RTT (Round Trip Time)、パケット廃棄率、また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 [5] 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、アプリケーションの性能維持に必要なスループットを高い確率で獲得することができれば、アプリケーション性能の向上が期待できると考えられる。

本稿では、ネットワークの輻輳レベルに関係なく、高い確率で一定のスループットを獲得することのできる、TCP の輻輳制御方式を提案する。本提案手法では、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することによって、アプリケーションに必要とされるスループットを獲得することを目標とする。その際、我々の研究グループで提案している Inline Mesurement TCP (ImTCP) [6] 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つつ、必要なスループットを提供する。本手法を用いることによって、例えば、Windows Media Player [7] や Real One Player [8] 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することができると考えられる。

さらに本稿では、提案手法を用いたコネクションがネットワーク内に複数本存在する場合に起こる問題点を指摘し、それら複数の TCP コネクションをまとめて管理・制御することによって、複数コネクションの合計スループットを確保することの必要性を指摘する。複数コネクションをまとめてある一定のスループットを確保することができれば、例えば広域イーサネットサービスなどを用いて複数拠点ネットワークが相互接続されたような企業ネットワークにおいて、特定の拠点間のトラヒックに対して一定のスループットを提供するなどのサービスが可能となる。

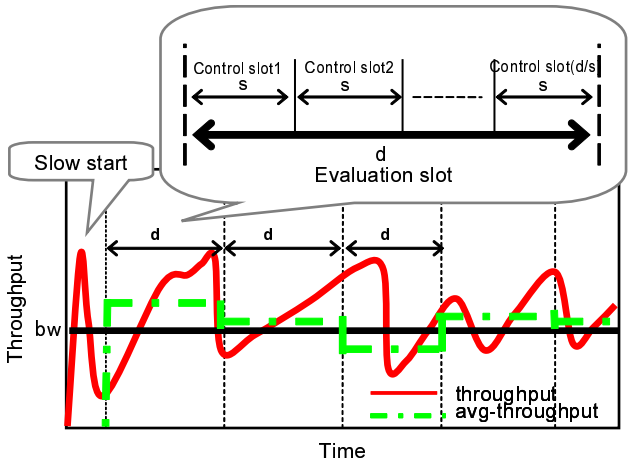

図 1　評価スロットと目標スループット

提案手法は、ns-2 [9] を用いたシミュレーションを行うことで評価する。ボトルネックリンクを共有する複数のコネクションが存在するネットワークモデルにおいて、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う。この際、背景トラヒックとして通常の TCP Reno コネクションを加え、背景トラヒックの大きさに応じて、指定されたスループットを確保できる時間確率の変化を観測することで、提案手法の性能評価を行う。

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 章ではシミュレーションにより提案手法の性能を評価する。最後に 4 章でまとめと今後の課題を述べる。

2. 提 案 手 法

2.1 概　　要

本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 bw (pkt/sec) が指定されることを想定している。また、TCP によるデータ転送において、常に一定のスループットを維持することは困難であるため、指定されたスループットを達成する時間間隔 t に関して、RTT あるいは秒単位で指定を受けるものとする。すなわち、bw が同じ場合でも、t が小さい場合にはより短い時間でのスループット確保が必要となるため、t が小さいほど、より厳しい条件であると言える。

提案手法は、指定された時間間隔 t ごとの平均スループットが、指定されたスループット bw 以上となるように制御を行う。ここで指定された時間間隔を評価スロット (Evaluation slot) と呼び、その長さを d (RTT) とする。アプリケーションからの時間間隔の指定が RTT 単位である場合には $d = t$ とし、秒単位の場合には指定時間を最小の往復伝播遅延時間で割った値を d に設定する。また、データ転送開始時およびタイムアウト発生直後のスロースタートフェーズでは、TCP Reno と同様の動作をする。図 1 に、提案手法における目標スループットと評価スロットの関係を示す。

2.2 スロットの設定

評価スロットを複数の制御スロット (Control Slot) に分割し、その長さを s (RTT) とする (図 1)。ただし s の範囲は $2 \leqq s \leqq d$ のとする。提案手法は、制御スロットごとに平均スループットの目標値 g を設定し、その値に基づいて転送速度を制御する。制御スロットをサイクルとする制御を繰り返すことで、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 以上になることを目指す。

評価スロット長 d および制御スロット長 s は RTT (Round Trip

Time) を単位とした変数であるが、これは TCP の持つ RTT の指数移動平均値である sRTT (smoothed RTT) の値を単位量とする。sRTT は時間とともに変動するため、提案手法では次の方法で制御スロットの長さを決定する。ある評価スロットにおける i 番目の制御スロット (以降、制御スロット i と呼ぶ) が開始した時点の sRTT の値を $srtt_i$ (sec) とし、制御スロット i のスロット長を $(srtt_i \times s)$ 秒とする。スロットの終了判定はタイマを用いず、以下のように行う。送信側 TCP が受信側からの ACK パケットを受け取った時、制御スロット i が開始してから $(srtt_i \times s)$ 秒以上経過していれば制御スロット i を終了し、制御スロット $i+1$ を開始する。

2.3 制御スロットの目標スループットの決定

各制御スロットの目標スループット値 g を決定する方法を説明する。制御スロット i における目標スループットを g_i (pkts/sec)、得られた平均スループットを $tput_i$ (pkts/sec) とすると、g_i は次式によって得る。

$$\begin{cases} g_i = bw + (g_{i-1} - tput_{i-1}) \\ g_1 = bw \end{cases}$$

すなわち、アプリケーションの目標スループット bw に加えて、直前の制御スロットにおけるスループットの過不足分を反映させ、次の制御スロットの目標スループット値とする。これにより、例えば直前の制御スロットにおけるスループットが目標値に到達していなければ、次の制御スロットの目標値を高く設定し、不足したスループットを取り戻すように動作する。逆に、直前の制御スロットで目標値を越えたスループットを獲得することができた場合、次の制御スロットでは目標値を低くすることで、制御を緩やかにすることができる。また評価スロットが終了すると、その過不足分を無効にする。

2.4 輻輳ウィンドウサイズによる転送レートの制御

TCP において一定のスループットを維持するための一つの方法として、輻輳ウィンドウサイズ ($cwnd$) を一定値に保ち、パケット廃棄が発生してもウィンドウサイズを下げないことが考えられる。しかしこの手法は、輻輳発生時に輻輳の解消が行なえない、競合するトラヒックとの公平性が著しく低下するなどの問題があるのは明らかである。そこで、競合する通常の TCP Reno コネクションとの公平性を可能な限り維持すると共に、瞬間的に発生する生存時間の短い TCP トラヒックのスループットを不当に下げないために、パケット廃棄時は TCP Reno と同じ挙動を行い、目標スループットを確保するために、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転送速度を制御する。

2.4.1 概　　要

ここでは提案手法が用いる、輻輳回避フェーズにおける輳ウィンドウサイズの制御について説明する。受信側 TCP からの ACK パケットが到着すると、送信側 TCP は次式を用いて $cwnd$ の値を更新する。

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

以降、k を傾き、あるいは cwnd の増加量と呼ぶ。提案手法は、傾き k を動的に増減させることで転送速度を制御する。k の計算方法は、次節で説明する。

パケット廃棄時の輻輳ウィンドウサイズの減少量については、TCP Reno と同様の手法を用いる。すなわちパケット廃棄が発生した場合には、次式のように Fast Retransmit の場合は $cwnd$ を半減させ、タイムアウトの場合は $cwnd$ の値を 1 にする。

$$\begin{cases} \text{FastRetransmit}： & cwnd \leftarrow \dfrac{cwnd}{2} \\ \text{タイムアウト}： & cwnd \leftarrow 1 \end{cases}$$

2.4.2 輻輳ウィンドウサイズの増加量の決定

制御スロット i における平均スループットが g_i となるための、$cwnd$ の増加量を k_{bw} とする。そのためには、制御スロット i の長さである s RTT の間に $(g_i \times srtt_i \times s)$ 個のパケットを送る必要がある。しかし、送信側 TCP において受信側にパケットが到着したことを確認するためには、送信側 TCP がパケットを送信してから 1 RTT を要する。また、廃棄されたパケットの再送には最低でも 1 RTT を要するため、制御スロットが終了する直前にパケット廃棄が発生すると、必要とされるパケット数を送信できない可能性が高まる。以上の理由により、提案手法は、$(s-2)$ RTT の間に必要なパケット数を送信するように k_{bw} を求める。

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 が開始してから n_j RTT が経過しているとする。また、その時の輻輳ウィンドウサイズの値を $cwnd_i^{n_j}$ とする。n_j RTT 経過後、評価スロット終了まで k_{bw} を用いることによって、ウィンドウサイズは 1RTT ごとに k_{bw} (packets) ずつ増加する。すなわち、評価スロット終了までの $(s-n_j-2)$ RTT の間に送信することができるパケット数 p_{snd} は次式によって得られる。

$$p_{snd} = (s-n_j+1)cwnd_i^{n_j} + \frac{k_{bw}}{2}(s-n_j-1)(s-n_j) \tag{2}$$

一方、n_j RTT 経過後の last_ack の値を $ack_i^{n_j}$ と表すと、平均スループットを g_i に到達させるために送信すべきパケット数 p_{need} は、次式によって得られる。

$$p_{need} = g_i \cdot srtt_i \cdot s - (ack_i^{n_j} - ack_i^0) \tag{3}$$

以上より、輻輳ウィンドウサイズの増加量を k_{bw} と設定したとき制御スロット i で送ることができるパケット数と、平均スループットが g_i に到達するためのパケット数が明らかになった。これより、$p_{snd} = p_{need}$ となる k_{bw} を求めることで、制御スロット i における平均スループットが g_i となるための、$cwnd$ の増加量が得られる。(2) および (3) 式より、k_{bw} は次式により得られる。

$$k_{bw} = \frac{2(g_i \cdot srtt_i \cdot s - (s-n_j-1)cwnd_i^{n_j} - ack_i^{n_j} + ack_i^0)}{(s-n_j-1)(s-n_j)}$$

$k = k_{bw}$ と設定することで、制御スロット i における平均スループットが g_i となるなるためのウィンドウサイズの増加量 k を決定することが出来る。提案手法においては、送信側 TCP が新たに ACK パケットを受信するたびに k の更新を行う。

2.4.3 利用可能帯域に応じた k の範囲の設定

前節において、制御スロット i における平均スループットが g_i となるための $cwnd$ の増加量 k_{bw} を求めた。しかし、急激に $cwnd$ を増加させることは、バースト的なパケット廃棄を発生させ、タイムアウトを引き起こす。一方、ネットワーク帯域が十分空いている場合に $cwnd$ の増加量が小さく設定されると、そのスループットが TCP Reno よりも低くなることがある。そこで本節では、$cwnd$ の増加量 k の上限値および下限値を決定する方法を述べる。k_{bw} がその範囲内であれば $k = k_{bw}$ とし、範囲外であれば k を上限値もしくは下限値に設定する。

次式に示すように、$cwnd$ の増加量 k の上限値を k_{max}、下限値を k_{min} とおく。

$$k_{min} \leqq k \leqq k_{max}$$

下限値 k_{min} は、ネットワーク帯域が十分に利用可能である場合のスループットを TCP Reno と同等にするため、$k_{min} = 1$ とする。

上限値 k_{max} は、ネットワークの利用可能帯域に基づいて決定することで、ネットワークの許容量を越えない $cwnd$ の増加量を求める。ここで利用可能帯域に関する情報は、ImTCP [6]

によるインラインネットワーク計測手法によって得る。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。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る [10] 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。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。

ImTCP アルゴリズムを用いて測定した利用可能帯域を A (pkts/sec) とおく。制御スロット i において、提案手法を用いたコネクションが他のトラヒックに影響を与えることなくネットワーク内を占めることができる最大のパケット数は $A \cdot srtt_i$ と表すことができる。また現在ネットワーク内に存在する提案手法のパケット数は $cwnd$ であることから、新たに送出するパケット数が $A \cdot srtt_i - cwnd$ 以下であれば、ボトルネックルータのバッファ溢れによるパケット廃棄が発生しないと考えられる。したがって送信側 TCP が ACK パケットを受信するたびに、次式を用いて k_{max} を更新する。

$$k_{max} = A \cdot srtt_i - cwnd \tag{4}$$

しかし、ネットワークが混んでいる、すなわち背景トラヒック量が多い場合、(4) 式で得られた上限値 k_{max} を用いて $cwnd$ の増加量 k を制限すると、要求されたスループットを獲得することができない。したがって背景トラヒック量が多い場合は、利用可能帯域分を超えて k_{max} を設定する必要がある。ここで制御スロット i における利用可能帯域の平均値を $avgA_i$ とおく。ネットワークが混雑している時、スループットは利用可能帯域によって抑えられると考えられることから、制御スロット i における平均スループットは $avgA_i$ となると考えられる。目標スループット g_i が $avgA_i$ に比べて大きい場合には、指定のスループットに $(g_i - avgA_{i-1})$ だけ不足していると見なすことができるため、利用可能帯域に加えて $(g_i - avgA_{i-1})$ 分だけ余分に帯域を確保するように、k_{max} を決定する。また物理帯域 P (pkts/sec) 以上の帯域を確保することはできないため、次式によって k_{max} を求める。

$$k_{max} = min(A + g_i - avgA_{i-1}, P) \cdot srtt_i - cwnd \tag{5}$$

制御スロット i において、同じ評価スロット内の過去の制御スロットの一つでも、そのスループットが目標値を下回っていた場合、ネットワークが混んでいると判断し、制御スロット i 以降の制御スロットでは (5) 式を用いる。

以上をまとめると、$cwnd$ の増加量の最小値 k_{min} および最大値 k_{max} は以下のように表せる。

$$k_{min} = 1$$

$$k_{max} = \begin{cases} A \cdot srtt_i - 2 \cdot cwnd \\ \qquad (\forall x\{(1 \leqq x < i) \vee (tput_x < g_x)\}) \\ min(A + g_i - avgA_{i-1}, P) \cdot srtt_i - 2 \cdot cwnd \\ \qquad (\exists x\{(1 \leqq x < i) \wedge (tput_x < g_x)\}) \end{cases}$$

2.5 制御スロット長の動的設定

一般に、制御スロット長 s の値が小さいほど、制御が細やかになり、アプリケーションから要求されたスループットを得ることが容易になる反面、背景トラヒックへの影響が大きくなる。すなわち、要求されたスループットを獲得することができる最大の値を s として設定することが望ましいといえる。しかし、理想的な値はネットワーク構成や背景トラヒック量などのネットワーク状況に依存するため、理想的な固定値をパラメータとして設定することは困難である。そこで提案手法は、制御スロット長 s を動的に変化させることで、ネットワークに応じた値を設定する。

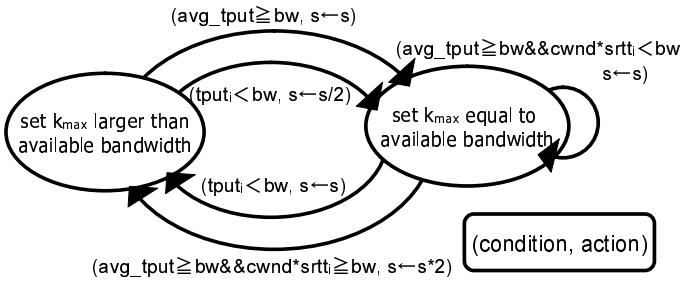

図 2　スロット長 s の決定方法と k_{max} の関係

簡単のため、評価スロット長 d および制御スロット長 s を 2 の乗数とし、s を半減/倍増させることによってスロット長を変更する。すなわち、評価スロットが終了した時、現在の s では次の評価スロットにおいて要求されたスループット bw を獲得できないと判断した場合は、s を半減させる。一方、s を増加させても次の評価スロットにおいて要求されたスループットを獲得できると判断した場合は、s を倍増させる。どちらにも当てはまらない場合は s の値を変更しない。具体的なアルゴリズムは以下の通りである。

直前の評価スロットにおいて、$cwnd$ の増加量の上限値 k_{max} を (5) 式を用いて利用可能帯域を越えて設定したにもかかわらず、目標スループットに到達できなかった場合、次の評価スロットにおいて制御スロット長 s を半減させる。すなわち、s を小さくすることでより細やかな制御を行い、目標スループットの獲得を目指す。一方、直前の評価スロットにおいて k_{max} を (4) 式を用いて利用可能帯域内に設定し、目標スループットを獲得でき、かつ、直前の評価スロット終了時の輻輳ウィンドウサイズが $cwnd \cdot srtt_i \geq bw$ を満たす場合に、スロット長 s を倍増させる。これは、ネットワーク帯域に十分な余裕があり、かつ評価スロット開始時点で目標スループットに相当する高い転送速度で通信していることから、s を大きくしても目標スループットを獲得できると考えられることによる。図 2 に s の変更条件と k_{max} の関係を示した状態遷移図を示す。

3. シミュレーションによる性能評価

本章では、提案手法の性能を ns-2 を用いたシミュレーションによって評価する。シミュレーションに用いるネットワークモデルを図 3 に示す。ボトルネックリンクの帯域および伝播遅延時間は 100 Mbps (12500 pkt/sec に相当) および 5 msec とする。ボトルネックルータは DropTail ルータを使用し、バッファサイズは 30 packets とする。提案手法を用いる送受信ホストがそれぞれ N_{tput} 台、背景トラヒックを発生させる TCP Reno の送受信ホストがそれぞれ N_{reno} 台存在し、アクセスリンクの帯域および伝播遅延時間はそれぞれ 1 Gbps および 2.5 msec とする。パケットサイズは 1000 Bytes とする。提案方式および TCP Reno コネクションは共に、無限長のデータ転送を行うものとする。

提案手法のパラメータとして、$d = 32$ とする。なおこの環境においては、32 RTT は約 1 秒に相当する。制御スロット長 s の初期値は d の 1/2 である 16 とする。また、今回のシミュレーションにおいては、利用可能帯域の計測手法として ImTCP は用いておらず、

利用可能帯域 = 物理帯域 − 背景トラヒックのスループット

とし、背景トラヒックのスループットは常に正確かつ最新の値が取得可能であると仮定している。これは、本稿における評価は、提案手法の本質的な性質を明らかにすることが目的である

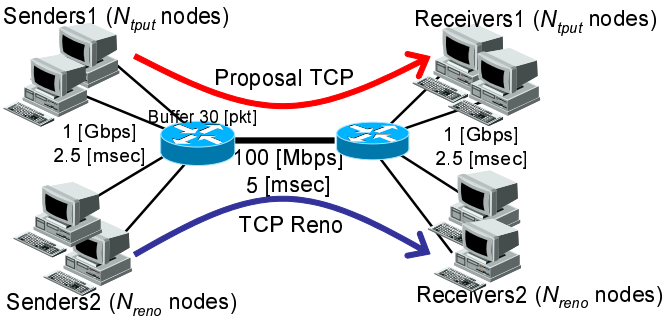

図 3　シミュレーションモデル

ためである。利用可能帯域に関する情報を ImTCP によって取得する場合における提案手法の評価は今後の課題としたい。

3.1　提案方式を用いるコネクションが 1 本存在する場合

まず、提案手法を用いるコネクションが 1 本の場合の性能評価を行う。ここでは $N_{tput} = 1$ とし、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 2500 (pkt/sec) 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、シミュレーション開始後、5 秒ごとに TCP Reno を用いるコネクション数 N_{reno} を 1、5、30 と変化させる。図 4 に、輻輳ウィンドウサイズおよび評価スロットごとの平均スループットの物理帯域に対する比率の変動を、図 5 に、制御スロット長 s の変動を示す。また図中の縦方向のグリッドは、評価スロットの境界を表している。

図 4 より、TCP Reno のコネクション数が 1 本であるシミュレーション開始後 5 秒間において、提案手法は TCP Reno とほぼ同様の速度で輻輳ウィンドウサイズを増加させ、かつアプリケーションから要求されたスループットを獲得でき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 2 本のコネクションが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う。このことから、提案手法は要求されるスループットを十分達成可能な場合には、TCP Reno との公平性を保つことが可能であることが明らかとなった。

また、TCP Reno のコネクション数が 5 本であるシミュレーション開始後 5-10 秒の期間は、TCP Reno と比較して輻輳ウィンドウサイズの増加速度が大きいことが分かる。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。結果として、提案手法は要求されたスループットを獲得し、背景トラヒックである TCP Reno の輻輳ウィンドウサイズが押し下げられている。

さらに TCP Reno のコネクション数が 30 本であるシミュレーション開始 10 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。また図 5 より、制御スロット長 s が 11 秒以降で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 30 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。

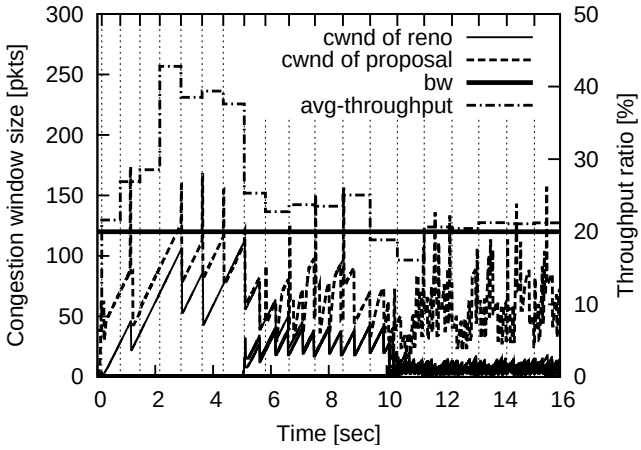

図 4　輻輳ウィンドウサイズおよびスループットの変化

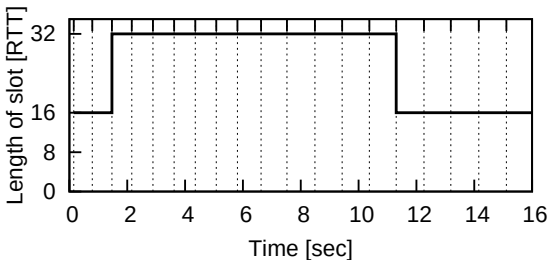

図 5　制御スロット長 s の変化

次に、$N_{tput} = 1$ とし、bw を物理帯域の 10% (1250 pkt/sec) および 20% (2500 pkt/sec) と設定した場合について、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時間確率を評価する。図 6 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。ここでのシミュレーション時間は 120 秒である。図には、比較のために、TCP Reno を用いた場合、および、輻輳ウィンドウサイズを $cwnd = bw \cdot srtt_{min}$ に固定し、パケット廃棄が発生しても輻輳ウィンドウサイズを小さくしない方式を用いた場合の結果を合わせて示している。ここで $srtt_{min}$ は sRTT の最小値である。なおこれらの比較手法においては、便宜上評価スロットを設定し、目標スループットを獲得できた評価スロット数の割合を導出している。

図 6 より、目標スループットが物理帯域の 10%の場合には、競合するトラヒック量が多い場合でも、高い割合で指定のスループットを獲得できていることが分かる。これは、TCP Reno および $cwnd$ を固定した手法と比較しても明らかである。TCP Reno は、競合する背景トラヒックと帯域を公平に共有するため、背景トラヒック量が少ない場合は 100%の確率で目標スループットを獲得することができる。しかし背景トラヒック量が多くなった場合は、それら全てのコネクションと帯域を分け合うため、目標スループットを獲得できる確率が急激に低下する。また $cwnd$ を固定した手法も、背景トラヒック量が多い場合は目標スループットを獲得できていない。これは $cwnd$ を固定した転送により輻輳が発生している状況においても大量のパケットをネットワークに送出するため、バースト的なパケット廃棄が発生し、転送効率が低下するためである。

一方、目標スループットが物理帯域の 20%の場合は、提案手法は競合するトラヒックのコネクション数が 15 本以上の場合に、要求されたスループットを獲得できる割合が大きく低下していることが分かる。この結果から、このシミュレーション環境における提案手法の限界が示唆される。すなわち、このシ

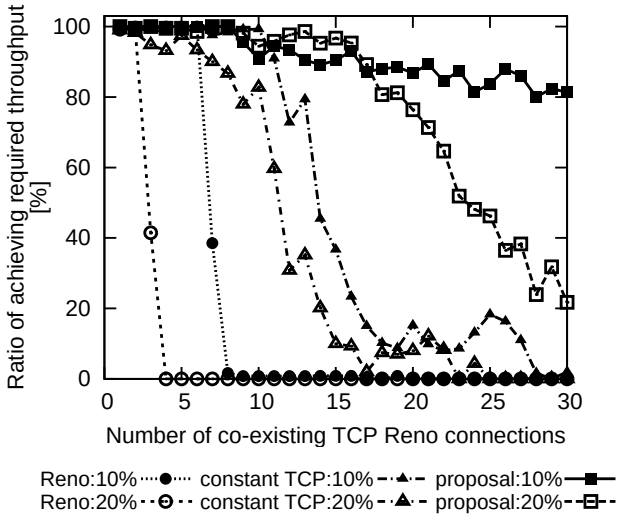

図 6　提案方式が 1 本の場合の目標スループット獲得割合

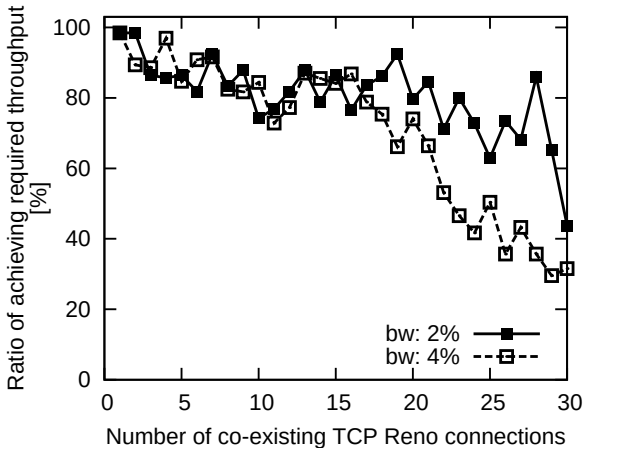

図 7　提案手法のコネクションが複数存在する場合の目標スループット獲得割合

ミュレーション環境においては、物理帯域の 20%のスループットを確保することは、背景トラヒックが非常に多い場合には不可能であるといえる。今後の課題として、提案方式の性能を数学的解析によって明らかにすることで、提案方式の適用範囲を明らかにすることが挙げられる。

3.2　提案手法を用いるコネクションが複数本存在する場合

次に、提案手法を用いるコネクションが複数本存在する場合の性能評価を行う。ここでは $N_{tput} = 5$ とし、bw を物理帯域の 2% (250 pkt/sec)、4% (500 pkt/sec) とそれぞれ設定した場合についてシミュレーションを行った。すなわち、提案手法を用いるコネクション 5 本の合計スループットは、それぞれ物理帯域の 10% (1250 pkt/sec)、20% (2500 pkt/sec) を占めることを意図している。図 7 に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、要求されたスループットを獲得できた割合を示す。割合は提案手法を用いるコネクション 5 本の平均値である。

図 7 より、図 6 と比較すると、提案手法によって獲得したい合計のスループットは同じであるにもかかわらず、コネクション数が増加することにより、要求されたスループットを獲得できる確率が低下していることが分かる。これは以下の原因によると考えられる。提案手法において、輻輳ウィンドウサイズの増加量の最大値 k_{max} は、ImTCP アルゴリズムにおける利用可能帯域の計測結果を用いて設定している。したがって、提案手法を用いるコネクションが複数本存在する場合は、それらのコネクションが同時に利用可能帯域に相当する量のパケットを送出することで、ネットワークの許容量を越えるパケットがボトルネックリンクに到着するため、バースト的なパケット廃棄が生じ、タイムアウトを引き起こす。以上より、提案手法を用いるコネクションが複数本存在する場合においては、複数本のコネクションを統合的に管理する制御システムが必要であると考えられる。

4. おわりに

本稿では、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対して提供することに着目し、高い確率で一定のスループットを獲得することができる、TCP の輻輳制御方式を提案した。シミュレーションによる提案手法の評価結果から、背景トラヒック量が多く、利用可能帯域がほとんど存在しない環境下であっても、物理帯域の 10%に相当するスループットを高い確率で獲得できることが明らかとなった。また、提案手法を用いるコネクションが複数本存在する場合に起こる問題を明らかにし、対策が必要であることを示した。

今後の課題としては、利用可能帯域に関する情報を ImTCP を用いて獲得する場合の評価、および提案手法の数学的解析などが挙げられる。また、提案手法を用いる複数本のコネクションを、統合して制御を行うシステムを提案したい。

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] T. Tsugawa, G. Hasegawa, and M. Murata, “Background TCP data transfer with Inline network measurement,” in *Proceedings of APCC 2005*, Oct. 2005.

[4] W. R. Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*. Addison-Wesley, 1994.

[5] 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.

[6] L. T. M. Cao, G. Hasegawa, and M. Murata, “An Inline measurement method for capacity of end-to-end network path,” in *Proceedings of IM’2005 E2EMON Workshop 2005*, May 2005.

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

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

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

[10] 津川 知明, 長谷川 剛, 村田 正幸, “インラインネットワーク計測手法 ImTCP およびその応用手法の実装および性能評価,” **電子情報通信学会技術研究報告** (IN2005-120), Dec. 2005.