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.

HighSpeed TCP の性能評価とその性能改善方式の提案

徳田　航一†　　長谷川　剛††　　村田　正幸††

† 大阪大学大学院基礎工学研究科　〒 560–8531 大阪府豊中市待兼山町 1–3
†† 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
Phone: 06-6850-6863, Fax: 06-6850-6868
E-mail: †{kouichit,hasegawa,murata}@anarg.jp

あらまし　1 Gbps を越える帯域を持つリンク上で既存の TCP Reno を用いると、その輻輳制御方式が原因で、リンク帯域を十分使えないことがわかっている。それに対して、TCP のウィンドウサイズの増減のアルゴリズムを変更し、高スループットを得ることができる HighSpeed TCP と呼ばれる方式が最近提案されている。しかし、その性能は十分に明らかになっておらず、また従来の TCP Reno コネクションとネットワーク帯域を共有した時の性能、公平性についても考慮されていない。そこで本稿では、まず HighSpeed TCP の基本的な性能を数学的解析およびシミュレーションによって評価し、HighSpeed TCP はバースト的なパケット廃棄が原因でリンク帯域を効率的に使うことができない場合があること、また TCP Reno との公平性の観点から問題があることを明らかにする。さらに、それらの問題点を解決する新たな TCP の輻輳制御方式の提案を行い、その有効性を評価する。その結果、HighSpeed TCP に比べて高いスループットを維持しながら、TCP Reno との公平性を改善できることが明らかとなった。
キーワード　TCP Reno、HighSpeed TCP、公平性

Performance Analysis of HighSpeed TCP and its Improvement for High Throughput and Fairness against TCP Reno Connections

Koichi TOKUDA†, Go HASEGAWA††, and Masayuki MURATA††

† Graduate School of Engineering Science, Osaka University
1–3 Machikaneyama, Toyonaka, Osaka, 560–8531, Japan
†† Cybermedia Center, Osaka University
1–32 Machikaneyama, Toyonaka, Osaka 560–0043, Japan
Phone: +81-06-6850-6863, Fax: +81-06-6850-6868
E-mail: †{kouichit,hasegawa,murata}@anarg.jp

Abstract　It was reported that the traditional TCP Reno cannot achieve gigabit level throughput due to the essential nature of its congestion control mechanism. HighSpeed TCP was proposed to overcome this problem. It modifies the increasing/decreasing algorithms of the congestion window size. But the performance of HighSpeed TCP has not been fully investigated. Especially, the fairness between HighSpeed TCP and TCP Reno is not considered. In this paper, we first investigate the throughput and fairness properties of HighSpeed TCP through mathematical analysis and simulation studies. We present that HighSpeed TCP can provide higher throughput than TCP Reno, but it cannot fully utilize the link bandwidth because of bursty packet losses at the router buffer. It is also shown that HighSpeed TCP degrades the throughput of TCP Reno when they share the bottleneck link. We then propose a new congestion control algorithm of TCP that alleviates the problems of HighSpeed TCP. The simulation results show that the proposed scheme can achieve higher throughput than HighSpeed TCP, keeping the fairness with TCP Reno connections.
Key words　TCP Reno, HighSpeed TCP, Fairness

1. はじめに

近年注目されているデータグリッドネットワーク、ストレージエリアネットワーク等においては、エンド端末が 1 Gbps クラスの帯域を持つ高速ネットワークに直接接続され、データの取得・送出、データベースの更新、遠隔バックアップ等において、ギガバイトからテラバイト級のデータを高速に転送することが要求される。これらのサービスは、非常に大きなネットワーク帯域とディスク領域を必要とするが、それらのコストは年々低下していることから、このような要求を持つサービスは今後も多く登場すると考えられる。しかし、このような高速データ転送を行う場合に、現在のインターネットにおいて標準的に用いられている TCP Reno バージョンを用いると、大きなリンク帯域を十分使う程度のスループットを得ることができないという問題が指摘されている。例えば文献 [1] によると、送受信端末間のラウンドトリップ時間 (RTT) が 100 msec、リンク帯域が 10 Gbps であるような大きな帯域のリンク上で、パケット長が 1500 Byte である 1 本の TCP Reno コネクションを用いてデータ転送を行ったときに、リンク帯域を十分使う程度のスループットを得るためには、パケット廃棄率が 2×10^{-10} 以下である必要がある。さらに、一度パケット廃棄が起きると、リンク帯域を十分使う程度のスループットを得るために、ウィンドウサイズが必要な大きさになるまで、40000 RTT (約 4000 秒) 以上の時間を要する。すなわち、TCP Reno において 1 Gbps を超えるスループットを得ることは、現実的には不可能である。このように TCP Reno が高いスループットを得ることができないのは、TCP Reno においては、パケット廃棄を検出するとウィンドウサイズを大きく減少させるのに対し、パケット廃棄が発生しない場合のウィンドウサイズの増加スピードが非常に遅いためである。

上述の問題を解決するための一つの方法として、TCP Reno の輻輳制御方式を改変し、高いスループットを得ることができる HighSpeed TCP と呼ばれる方式が文献 [1] によって提案されている。HighSpeed TCP は、輻輳回避フェーズにおける 1 RTT ごとのウィンドウサイズの増加幅を大きくし、パケット廃棄が発生した際のウィンドウサイズの減少幅を小さくすることで、大きなウィンドウサイズを保つことができる。したがって、HighSpeed TCP は、TCP Reno に比べてより高速なデータ転送

を実現することができると考えられる。しかし、HighSpeed TCP の性能に関する評価はこれまでほとんど行われていないため、その性質は十分に明らかになっていない。また、HighSpeed TCP は従来方式である TCP Reno と同一ネットワーク内において共存することを想定していないため、それらの間の公平性に関してはまったく考慮されていない。しかし、上述のようなネットワークにおいては、大きな帯域を持つリンク上に、サーバ間の高速データ転送だけではなく、Web、電子メールなどの従来のトラヒックも存在すると考えられる。したがって、HighSpeed TCP コネクションが、共存する TCP Reno コネクションに与える影響を評価し、公平性に関する検討を行うことは重要である。

そこで本稿では、HighSpeed TCP コネクションが従来の TCP Reno コネクションと同じリンクを共有する場合の、スループットおよび公平性に関して、数学的解析手法およびコンピュータ上のシミュレーションを用いて考察する。解析においては、ボトルネックとなるルータのバッファに着目し、HighSpeed TCP および TCP Reno コネクションの、パケット廃棄をきっかけとした周期的なウィンドウサイズの動きをモデル化し、それぞれのコネクションの平均スループットを導出する。解析結果およびコンピュータ上のシミュレーションによる考察の結果、HighSpeed TCP は従来の TCP Reno に比べて非常に高いスループットを得ることができるが、システム条件によっては大量のパケット廃棄によってスループットが著しく低下し、リンク帯域を十分使う程度のスループットを得ることができない場合があること、また、従来の TCP Reno と同じリンクを共有する場合、TCP Reno を用いたコネクションのスループットを大幅に低下させるため、両者間の公平性を維持することができない等の問題点を持つことを明らかにする。

さらに本稿では、解析によって明らかになった HighSpeed TCP が持つ問題点を解決し、高いスループットを得るとともに、TCP Reno コネクションとの公平性を改善する TCP の輻輳制御方式、Gentle HighSpeed TCP の提案を行う。Gentle HighSpeed TCP では、輻輳回避フェーズにおいて転送するデータパケットの RTT をパケットごとに観測し、その変化を基に、ウィンドウサイズの増減の幅を変化させる。さらに、スロースタートフェーズにおけるバースト的なパケット廃棄によるスループット低下を防止する、新たなスロースタートアルゴリズムの提案をあわせて行う。Gentle HighSpeed TCP の有効性はシミュレーションによって評価を行い、Gentle HighSpeed TCP 方式によって、従来の TCP コネクションとの公平性を維持しながら、高速なデータ転送を実現することができることを示す。

2. TCP の輻輳制御方式

TCP はネットワークの輻輳状況に応じてウィンドウサイズを動的に増減させ、ネットワークへ送出するデータ量を調節することにより、輻輳制御を行っている。本章では、本稿において対象とする TCP Reno および HighSpeed TCP の輻輳制御方式について、特に輻輳ウィンドウサイズを変化させるアルゴリズムに着目して説明する。それぞれの輻輳制御方式の詳細については、文献 [2]、文献 [1] 等を参照されたい。

2.1 TCP Reno

TCP Reno のウィンドウサイズは、パケット廃棄をきっかけにして周期的に変化する。ここで、$(i-1)$ 番目のパケット廃棄から i 番目のパケット廃棄までの一周期を周期 i とする。また、TCP Reno によるデータ転送を RTT ごとに区切り、周期 i の開始時から数えて j 番目の RTT における TCP Reno コネクションのウィンドウサイズを $W_{Reno}(i,j)$ と定義する。

TCP Reno の輻輳制御方式は、スロースタートフェーズおよび輻輳回避フェーズと呼ばれる 2 つのフェーズから構成され、それぞれにおいてウィンドウサイズの増加速度が異なる。スロースタートフェーズにおいては、1 つの ACK パケットを受信するごとにウィンドウサイズ $W_{Reno}(i,j)$ を 1 パケット増加させる。一方、輻輳回避フェーズにおいては、1 つの ACK パケットを受信するごとにウィンドウサイズ $W_{Reno}(i,j)$ を $1/W_{Reno}(i,j)$ パケットだけ増加させる。ウィンドウサイズの RTT ごとの変化に着目すると、$W_{Reno}(i,j)$ は以下のように表すことができる [2]。

$$W_{Reno}(i,j) = \begin{cases} 2 \cdot W_{Reno}(i,j-1) \\ \quad (\text{if } W_{Reno}(i,j-1) < S_{Reno}(i)) \\ W_{Reno}(i,j-1) + 1 \\ \quad (\text{if } W_{Reno}(i,j-1) \geqq S_{Reno}(i)) \end{cases}$$

ここで、$S_{Reno}(i)$ は 周期 i における TCP Reno がスロースタートフェーズから輻輳回避フェーズに移行する時のしきい値である。ネットワーク内でパケットが廃棄されると、TCP Reno はタイムアウト、あるいは、重複 ACK (シーケンス番号が同じ複数の ACK パケット) を受信することによってそれを検出し、廃棄されたパケットを再送する。タイムアウトによってパケット廃棄が検出されると、ウィンドウサイズを 1 パケットにし、スロースタートフェーズに移行する。一方、重複 ACK を受信することによってパケット廃棄が検出された場合には、ウィンドウサイズを現在のウィンドウサイズの 1/2 にし、輻輳回避フェーズに移行する。また、パケット廃棄が検出された場合には、$S_{Reno}(i)$ を現在のウィンドウサイズの 1/2 に設定する。TCP Reno はこのように、パケット廃棄をきっかけとした周期的な動作を行いながら、ネットワークへデータを送出し続ける。

2.2 HighSpeed TCP

TCP Reno が高いスループットを得ることができないのは、ウィンドウサイズの増加の幅が RTT ごとに 1 パケットと非常に小さいにもかかわらず、パケット廃棄を検出した際にウィンドウサイズを 1/2 以下へと大きく減少させるために、ウィンドウサイズがなかなか大きくならないことが原因である。この問題を解決するために、文献 [1] において HighSpeed TCP と呼ばれる方式が提案された。HighSpeed TCP がウィンドウサイズを増減させるアルゴリズムは、基本的には TCP Reno と同じであるが、輻輳回避フェーズにおける 1 RTT ごとのウィンドウサイズの増加幅、および、重複 ACK の受信によってパケット廃棄を検出した際のウィンドウサイズの減少幅が異なる。以下では、輻輳回避フェーズにおける HighSpeed TCP のウィンドウサイズの増減について説明する。ここで、2.1 節における説明と同様、周期 i の j 番目の RTT における HighSpeed TCP コネクションのウィンドウサイズを $W_{HS}(i,j)$ とする。

HighSpeed TCP は、現在のウィンドウサイズが W_{low} より小さい場合、TCP Reno と同じアルゴリズムにしたがってウィンドウサイズを増減させる。一方、現在のウィンドウサイズが W_{low} よりも大きい場合、ACK パケットを受信した際に TCP Reno よりもウィンドウサイズを大きく増加させ、重複 ACK によってパケット廃棄を検出した際には、TCP Reno のように大きくウィンドウサイズを減少させない。このことにより、TCP コネクションのウィンドウサイズを大きな値に維持することができるため、高いスループットを得ることができる。ウィンドウサイズの増加幅および減少幅は、現在のウィンドウサイズの大きさを用いて決定される。ウィンドウサイズが w の時の、1 RTT ごとのウィンドウサイズの増加幅を $a(w)$、重複 ACK によってパケット廃棄を検出した際のウィンドウサイズの減少幅を $b(w)$ とすると、$a(w)$ および $b(w)$ は次のように表される [1]。

$$\begin{aligned} a(w) &= \frac{2w^2 \cdot b(w) \cdot p(w)}{2 - b(w)} \\ b(w) &= \frac{\log(w) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})}(b_{high} - 0.5) + 0.5 \quad (1) \\ p(w) &= \exp\Big[\frac{\log(w) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})} \\ &\qquad \cdot \{\log(P_{high}) - \log(P_{low})\} + \log(P_{low})\Big] \end{aligned}$$

式 (1) 中の P_{low} は、$P_{low} = \frac{1.5}{W_{low}^2}$ により定義され、TCP Reno において平均ウィンドウサイズが W_{low} となるときのパケット廃棄率を表している。式 (1) 中の P_{high}、W_{high}、b_{high}、W_{low} は HighSpeed TCP が持つパラメータであり、以下に述べる意味を持つ。HighSpeed TCP は、ネットワークにおけるパケット廃棄率が P_{high} のときに、平均ウィンドウサイズが W_{high} になるように、ウィンドウサイズの増加幅 $a(w)$、および、減少幅 $b(w)$ を決定する。また、現在のウィンドウサイズが W_{high} の時、重複 ACK の受信によってパケット廃棄を検出した際のウィンドウサイズを $(1 - b_{high}) \cdot W_{high}$ にする。これらのパラメータは、HighSpeed TCP を用いるネットワークの RTT、パ

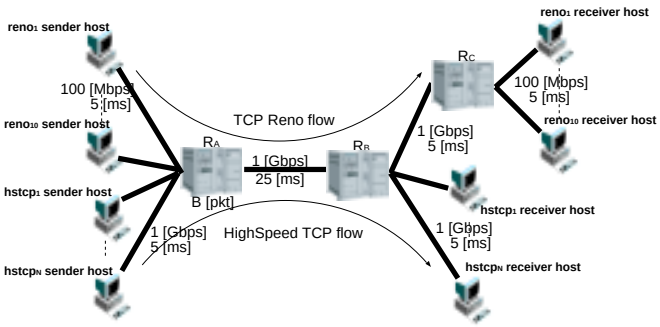


図 1　シミュレーションモデル

ケット廃棄率、および目標とするスループットに応じて設定することができる。たとえば、文献 [1] では、送受信端末間の往復伝搬遅延時間が 100 msec、パケット長が 1500 Byte の時に、10 Gbps のスループットを得ることを前提としたパラメータとして、$W_{low} = 31$、$P_{high} = 10^{-7}$、$b_{high} = 0.1$ が示されている。

3. HighSpeed TCP の性能評価

本章では、簡単なシミュレーションを行い、HighSpeed TCP の基本的な性質を明らかにし、スループットおよび公平性の両面において問題点があることを示す。

図 1 は、本章で行うシミュレーションで用いるネットワークモデルを示している。モデルは、2 つのローカルエリアネットワークが 1 Gbps の高速リンクで接続され、N 台のエンドホスト (hstcp$_1$, . . . , hstcp$_N$) が、その高速なリンクに直接接続され、HighSpeed TCP を用いてデータを転送しており、同時に、従来の TCP Reno を用いるエンドホスト (reno$_1$, . . . ,reno$_n$) が、100 Mbps の LAN を介して、HighSpeed TCP コネクションと高速リンクを共有している状況を想定している。すなわち、高速リンク上には、n 本の TCP Reno コネクションと N 本の HighSpeed TCP コネクションが存在する。ルータ R$_A$ とルータ R$_B$ を結ぶリンクの伝搬遅延時間を 25 msec、エンドホストとルータ間のリンクの伝播遅延時間を 5 msec、ルータ R$_B$ とルータ R$_C$ を結ぶリンクの帯域を 1 Gbps、伝搬遅延時間を 5 msec とする。なお、ボトルネックルータ R$_A$ には Taildrop ルータを用い、バッファ容量は 4167 packets (R$_A$ と R$_B$ 間を結ぶリンクの帯域遅延積の 2 倍に相当) に設定する。

図 2 は TCP Reno のコネクション数 n を 10 としたときの、HighSpeed TCP コネクションの本数 N に対する、TCP Reno および HighSpeed TCP コネクションの合計のスループットを示している。この図から、HighSpeed TCP コネクションが存在しない場合に、ボトルネックリンクの帯域が使い切られておらず、約 200 Mbps が利用されていないことがわかる。これは、ルータ R$_A$ でバッファ溢れによるパケット廃棄が発生し、TCP Reno コネクションのスループットが低下しているためである。また、HighSpeed TCP コネクションが存在すると、TCP Reno コネクションの合計スループットが低下しており、その割合は、HighSpeed TCP コネクションの本数が増加するにしたがって大きくなっている。これは、HighSpeed TCP は競合する TCP Reno コネクションとの公平性を考慮していないため、ウィンドウサイズを急激に大きくすることによって TCP Reno コネクションのスループットを不当に押し下げるためである。

また、図 2 より、HighSpeed TCP コネクションを用いても、ボトルネックリンクの利用率が 100% に達していないことがわかる。これは、HighSpeed TCP は、1 RTT ごとにウィンドウサイズを大きく増加させるため、ルータ R$_A$ のバッファが溢れる際に、多くのパケットがバースト的に廃棄されるためである。一般的に、1 ウィンドウ内の複数のパケットがバースト的に廃棄された場合、TCP の Fast Retransmit アルゴリズムは効果的に動作せず、タイムアウトが発生する [3], [4]。したがって、HighSpeed TCP コネクションのパケットがバースト的に廃棄されることによってタイムアウトが発生し、スループットが大幅に低下しているために、ボトルネックリンクの帯域を十分使いきれない。このことは、HighSpeed TCP コネクションのウィンドウサイズの時間的変動を示した図 3 においても確認することができる。図より、ルータのバッファが溢れパケット廃棄が発生し、ウィンドウサイズが減少する際に、ウィンドウサイズは 1 パケットまで減少しており、タイムアウトが発生していることがわかる。

さらに、図 3 より、HighSpeed TCP は、コネクション開始直後のスロースタートフェーズにおいて、ウィンドウサイズを急激に上昇させ、大量のパケットが廃棄されていることが分かる。これは従来の TCP Reno において発生していた問題であり、HighSpeed TCP においても解決されていない。これに対して文献 [5] では、Limited Slow Start と呼ばれる方式を用いることを推奨している。これはコネクション確立直後のスロースタートフェーズにおけるウィンドウサイズの増加幅を減少させるものである。しかしこの方式は、ウィンドウサイズの増加幅を減少させるタイミングをパラメータとして与える必要があるため、本質的な解決方法ではない。

以上の結果から、HighSpeed TCP は TCP Reno に比べて高いスループットを得られる場合もあるが、TCP Reno コネクションのスループットを不当に減少させるために公平性が悪化する、また、ルータのバッファ溢れの際やデータ転送直後のスロースタートフェーズにおいて大量のパケットがバースト的に廃棄されるため、ボトルネックリンクの利用率が低下する、等の問題点を持つことが明らかとなった。

4. 公平性解析

3. 章におけるシミュレーション結果から、HighSpeed TCP が持つ問題点の一つとして、従来の TCP Reno コネクションのスループットを不当に下げてしまい、公平性が劣化する可能性があることがわかった。本章では、その性質を明らかにするために、数学的解析手法を用いてスループット解析を行い、公平性の評価を行う。

4.1 解析モデル

図 4 は、解析の対象とするネットワークモデルを示している。ネットワークモデルには、HighSpeed TCP コネクションの送信側/受信側ホスト、TCP Reno コネクションの送信側/受信側ホスト、ルータ、および、それらをつなぐリンクから構成されている。ルータ R$_A$ とルータ R$_B$ をつなぐリンクの帯域を μ packet/sec、ルータ R$_A$ のバッファサイズを B packets、送信側/受信側ホスト間の伝搬遅延時間を τ sec、HighSpeed TCP コネクションの送信側/受信側ホストとルータ間を結ぶリンクの帯域を μ_{HS} packet/sec、TCP Reno コネクションの送信側/受信側ホストとルータ間を結ぶリンクの帯域を μ_{Reno} packet/sec とする。ここで HighSpeed TCP コネクションは、帯域の大きなリンクに直接接続されていると仮定し、$\mu = \mu_{HS}$ とする。一方、従来の Web アクセス、ファイル転送等で使用される TCP Reno ホストのアクセスリンク帯域 μ_{Reno} は、ルータ R$_A$ と R$_B$ を結ぶリンクの帯域よりも小さいと考えられるため、$\mu < \mu_{Reno}$ とする。ボトルネックルータ R$_A$ には、Taildrop ルータが用いられているものとする。また簡単のために、TCP Reno コネクションと HighSpeed TCP コネクションはそれぞれ 1 本ずつ存在し、送信側ホストには、常に送信するデータがあるものとし、ウィンドウサイズが許す限りデータを送出し続けると仮定する。

4.2 解　　析

解析では、HighSpeed TCP および TCP Reno コネクションの、パケット廃棄をきっかけとした周期的なウィンドウサイズの変化をモデル化することによって、それぞれのコネクションの平均スループットを導出する。図 5 に、ウィンドウサイズの変化の様子を示す。ここで、$(i-1)$ 番目のパケット廃棄を検知しウィンドウサイズを減少させてから、i 番目のパケット廃棄を検知しウィンドウサイズを減少させるまでの期間を周期 i とし、周期 i 中の j 番目の RTT における TCP Reno と HighSpeed TCP のウィンドウサイズを、それぞれ $W_{Reno}(i,j)$、$W_{HS}(i,j)$ とする。また、周期 i における TCP Reno と HighSpeed TCP の $ssthresh$ の値を $S_{Reno}(i)$、$S_{HS}(i)$ とし、周期 i 終了後の TCP Reno と HighSpeed TCP のウィンドウサイズを $W'_{Reno}(i)$、$W'_{HS}(i)$ とする。

周期 i 開始時のウィンドウサイズは、周期 $(i-1)$ の最後のウィンドウサイズに等しいため、以下の式が成り立つ。

$$W_{Reno}(i,1) = W'_{Reno}(i-1)$$
$$W_{HS}(i,1) = W'_{HS}(i-1)$$

2. 2 節の式 (1) から、スロースタートフェーズおよび輻輳回避フ

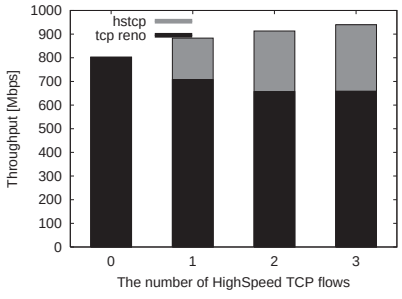


図 2　HighSpeed TCP のスループット

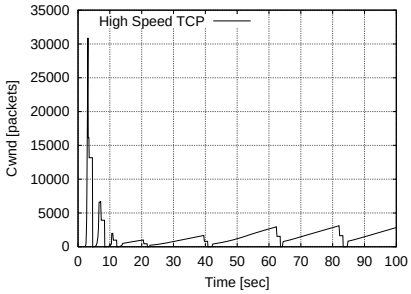


図 3　ウィンドウサイズの変動

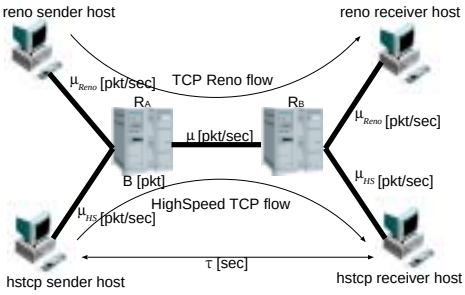


図 4　解析モデル

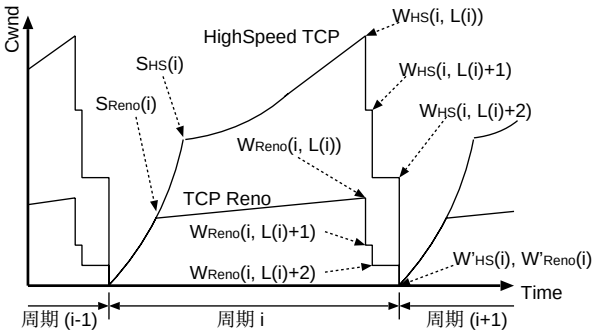


図 5　周期 i におけるウィンドウサイズの時間的変動

ェーズにおける、HighSpeed TCP のウィンドウサイズ $W_{HS}(i,j)$ は次のように表される。

$$W_{HS}(i,j) = \begin{cases} 2 \cdot W_{HS}(i,j-1) \\ \qquad (\text{if } W_{HS}(i,j-1) < S_{HS}(i)) \\ W_{HS}(i,j-1) + a(W_{HS}(i,j-1)) \\ \qquad (\text{if } W_{HS}(i,j-1) \geqq S_{HS}(i)) \end{cases} \quad (2)$$

一方、TCP Reno コネクションのウィンドウサイズについては、TCP Reno ホストのアクセスリンク帯域 μ_{Reno} は、ルータ R_A と R_B を結ぶリンクの帯域よりも小さいと考えられるため、$\mu_{Reno} < \mu$ であることを考慮する必要がある。まず、TCP Reno が 1 RTT の間にネットワーク中に送出可能なパケット数の最大値 W_{lim} を求める。TCP Reno は、最大 μ_{Reno} packet/sec でパケットを送出可能であるため、W_{lim} は次のように表される。

$$W_{lim} = 2\tau\mu_{Reno}$$

したがって TCP Reno は、1 RTT ごとに $\min(W_{lim}, W_{Reno}(i,j))$ 個のパケットをネットワークに送出するため、このとき受信する ACK パケット数は、$\min(W_{lim}, W_{Reno}(i,j))$ 個である。したがって、スロースタートフェーズおよび輻輳回避フェーズにおける、TCP Reno のウィンドウサイズ $W_{Reno}(i,j)$ は次のように表される。

$$W_{Reno}(i,j) = \begin{cases} W_{Reno}(i,j-1) + \min(W_{lim}, W_{Reno}(i,j-1)) \\ \qquad (\text{if } W_{Reno}(i,j-1) < S_{Reno}(i)) \\ W_{Reno}(i,j-1) + \dfrac{\min(W_{lim}, W_{Reno}(i,j-1))}{W_{Reno}(i,j-1)} \\ \qquad (\text{if } W_{Reno}(i,j-1) \geqq S_{Reno}(i)) \end{cases} \quad (3)$$

式 (2)、(3) にしたがって、各コネクションのウィンドウサイズが増加していくと、ルータ R_A のバッファにパケットが蓄積され始め、パケット廃棄が発生する。ここで、周期 i のパケット廃棄が $L(i)$ 番目の RTT で発生するものとすると、$L(i)$ 番目の RTT において、以下の式が成立する。

$$\min(W_{Reno}(i,L(i)), W_{lim}) + W_{HS}(i,L(i)) > 2\tau\mu + B$$

その時に廃棄されるパケットの個数を $D(i)$ とすると、$D(i)$ は以下のように表される。

$$D(i) = W_{Reno}(i,L(i)) + W_{HS}(i,L(i)) - (2\mu\tau + B)$$

廃棄される $D(i)$ パケットのうち、TCP Reno コネクションの廃棄パケット数 $D_{Reno}(i)$ と HighSpeed TCP コネクションの廃棄パケット数 $D_{HS}(i)$ の割合は、パケット廃棄が起きる瞬間の、両コネクションのウィンドウサイズの割合に等しくなるものと考えられる。したがって、$D_{Reno}(i)$ および $D_{HS}(i)$ は、次のように表される。

$$D_{Reno}(i) = \frac{W_{Reno}(i,L(i))}{W_{Reno}(i,L(i)) + W_{HS}(i,L(i))} \cdot D(i)$$

$$D_{HS}(i) = \frac{W_{HS}(i,L(i))}{W_{HS}(i,L(i)) + W_{HS}(i,L(i))} \cdot D(i)$$

次に、パケット廃棄が発生した後のウィンドウサイズを導出する。本解析においては、ボトルネックルータ R_A に Taildrop ルータを用いているため、ルータ R_A のバッファが溢れてパケット廃棄が起きるとき、バースト的にパケットが廃棄される、すなわち $D(i) > 1$ であると仮定する。以下では、各コネクションにおいて、3 パケット以上廃棄された場合の、ウィンドウサイズの導出方法について述べるが、1 パケットおよび 2 パケット廃棄された場合についても、以下に述べる方法により、パケット廃棄後のウィンドウサイズを求めることができる。

同一ウィンドウ内で 3 個以上のパケットが廃棄されると、TCP Reno は Fast Retransmit アルゴリズムによる再送を 2 度行い、その後タイムアウトが発生する [3], [4]。HighSpeed TCP においても、廃棄されたパケットの再送方法については TCP Reno と同じである。したがって、1 度目の再送後の、HighSpeed TCP/TCP Reno コネクションのウィンドウサイズは次のように表される。

$$W_{Reno}(i,L(i)+1) = \frac{W_{Reno}(i,L(i))}{2}$$

$$W_{HS}(i,L(i)+1) = W_{HS}(i,L(i)) \cdot (1 - b(W_{HS}(i,L(i))))$$

同様に、2 度目の再送後のウィンドウサイズは以下のように表される。

$$W_{Reno}(i,L(i)+2) = \frac{W_{Reno}(i,L(i)+1)}{2}$$

$$W_{HS}(i,L(i)+2) = W_{HS}(i,L(i)+1) \cdot (1 - b(W_{HS}(i,L(i)+1)))$$

2 度目の再送後にタイムアウトが発生すると、ウィンドウサイズは両コネクションとも 1 パケットになり、$ssthresh$ は以下のように更新される。

$$S_{Reno}(i+1) = \frac{W_{Reno}(i,L(i)+2)}{2}$$

$$S_{HS}(i+1) = \frac{W_{HS}(i,L(i)+2)}{2}$$

同様に、1 パケット廃棄された場合、2 パケット廃棄された場合についても、周期 i 終了時のウィンドウサイズを求めると、次式が成立する。

$$W'_{Reno}(i) = \begin{cases} W_{Reno}(i,L(i)+1) & \text{if } D_{Reno}(i) = 1 \\ W_{Reno}(i,L(i)+2) & \text{if } D_{Reno}(i) = 2 \\ 1 & \text{if } D_{Reno}(i) \geqq 3 \end{cases}$$

$$W'_{HS}(i) = \begin{cases} W_{HS}(i,L(i)+1) & \text{if } D_{HS}(i) = 1 \\ W_{HS}(i,L(i)+2) & \text{if } D_{HS}(i) = 2 \\ 1 & \text{if } D_{HS}(i) \geqq 3 \end{cases}$$

以上で、周期 i における各コネクションのウィンドウサイズの変化を導出することができた。次に、これまでの解析により求めたウィンドウサイズを用いて、TCP Reno コネクションおよび HighSpeed TCP コネクションの平均スループットを導出する。周期 i の j 番目の RTT におけるボトルネックルータ R_A のキュー長は $\max(W_{Reno}(i,j)+W_{HS}(i,j)-2\mu\tau,0)$ で表されるため、周期 i の j 番目の RTT におけるボトルネックルータ R_A のキューイング遅延 $Q(i,j)$ は、次のように表される。

$$Q(i,j) = \frac{\max(W_{Reno}(i,j)+W_{HS}(i,j)-2\mu\tau,0)}{\mu}$$

したがって、TCP Reno の平均スループット ρ_{Reno}、HighSpeed TCP の平均スループット ρ_{HS} は次のように表される。

$$\rho_{Reno} = \frac{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)} W_{Reno}(i,j)}{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}(Q(i,j)+2\tau)}$$

$$\rho_{HS} = \frac{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)} W_{HS}(i,j)}{\sum_{i=1}^{\infty}\sum_{j=1}^{L(i)}(Q(i,j)+2\tau)}$$

TCP Reno、HighSpeed TCP の平均スループットは、ρ_{Reno} と ρ_{HS} がある値に収束する i まで、上式を計算することによって求められる。

4.3 数 値 例

本節では、前節で述べた解析の妥当性を、解析結果とシミュレーション結果を比較することによって評価し、公平性に関する議論を行う。シミュレーションでは、図 4 に示すネットワークモデルを用い、$\mu=\mu_{HS}=1$ Gbps、送信側/受信側ホストとルータを結ぶリンクの伝搬遅延時間を 5 msec、ルータ間を結ぶリンクの伝搬遅延時間を 25 msec、パケット長を 1500 Byte、バッファサイズ B を 4167 packets とし、HighSpeed TCP のパラメータとして 2. 章で示した値を用いた。図 6 は、μ_{Reno} の帯域を 100 Mbps、12 Mbps、1.5 Mbps と変化させた時の、ルータ R_A とルータ R_B を結ぶボトルネックリンクの伝搬遅延時間に対する、各コネクションの平均スループットの変化を示したグラフである。いずれの場合においても、本章で述べた解析は、TCP Reno、HighSpeed TCP コネクションのスループットをほぼ正確に推測できていることがわかる。

また、ボトルネックリンクの伝播遅延時間が大きくなると、HighSpeed TCP コネクションのスループットが上昇しているのに対し、TCP Reno コネクションのスループットが減少していることが分かる。これは、TCP Reno および HighSpeed TCP のウィンドウサイズの増減アルゴリズムの違いが原因であり、以下のように説明することができる。TCP Reno は、現在のウィンドウサイズに関係なく、ウィンドウサイズを増減させるのに対し、HighSpeed TCP では現在のウィンドウサイズが大きいと、1 RTT ごとのウィンドウサイズの増加幅を大きくする。したがって、ボトルネックリンクの伝播遅延時間が大きくなり、リンクの帯域遅延積が大きくなると、HighSpeed TCP はウィンドウサイズをより素早く増加させてその帯域遅延積を利用しようとするのに対し、TCP Reno のウィンドウサイズの増加速度は一定である。このことが、伝播遅延時間の変化に対するスループット変化の違いを発生させている。

さらに、TCP Reno コネクションのスループットに着目すると、特に μ_{Reno} が大きい場合に、TCP Reno コネクションのスループットが μ_{Reno}、つまり、アクセスリンクの帯域に比べて非常に小さくなっていることがわかる。これは、HighSpeed TCP コネクションが急激にウィンドウサイズを増加させるために、ボトルネックリンクにおいて不当に帯域を占有し、TCP Reno コネクションのスループットを圧迫していることを示している。

一方、HighSpeed TCP は、ウィンドウサイズが W_{low} よりも小さい場合、TCP Reno と同じ動作をする。これは、ネットワーク帯域が小さく、ウィンドウサイズが小さいときに、TCP Reno との公平性を維持することを目的としている。しかし、ネットワークが高速になり、HighSpeed TCP がより積極的にウィンドウサイズを増加させる領域においては、従来の TCP Reno との公平性はまったく考慮されていない。しかし、1. 章でも述べたように、大きな帯域を持つリンク上に、サーバ間の高速データ転送だけではなく、Web、電子メールなどのトラヒックも存在する。したがって、HighSpeed TCP コネクションが、リンクを共有する TCP Reno コネクションとの公平性を維持できないことは問題である。特に、ボトルネック帯域の伝播遅延時間が小さい時に、両コネクションの合計スループットがリンク帯域を大幅に下回っており、ネットワーク帯域を有効に利用できていないことがわかる。これは、2. 章でも述べたように、HighSpeed TCP が大きくウィンドウサイズを増加させるために、バッファ溢れの際に多くのパケットがバースト的に廃棄されるために、両コネクションにおいてタイムアウトが発生していることが原因である。そこで、次章では、HighSpeed TCP が持つこれらの問題点を解決する Gentle HighSpeed TCP の提案を行う。

5. Gentle HighSpeed TCP

本章では、解析およびシミュレーション結果によって明らかになった HighSpeed TCP が持つ問題点を解決し、高いスループットを得るとともに、TCP Reno コネクションとの公平性を改善する Gentle HighSpeed TCP の提案を行う。

5.1 アルゴリズム

Gentle HighSpeed TCP は、スロースタートフェーズおよび輻輳回避フェーズの 2 つのフェーズを持つ。輻輳回避フェーズは、TCP Reno モードと HighSpeed TCP モードの 2 つのモードから構成される。Gentle HighSpeed TCP では、パケット廃棄の発生やラウンドトリップ時間の計測により、ネットワークの状態を判断し、Gentle HighSpeed TCP の状態を遷移させる。それぞれの状態では、主にウィンドウサイズの増加アルゴリズムが異なる。以下、それぞれの状態における Gentle HighSpeed TCP のアルゴリズムの詳細について説明する。

5.1.1 スロースタートフェーズ

2. 章で示した、コネクション確立直後のスロースタートフェーズにおいて発生する、バースト的なパケット廃棄を回避するために、これまで様々な方法が提案されてきた [5]。しかし、それらの方法の多くは、パラメータの適切な設定が必要とされる、遅延の大幅な変動に対応できないなどの問題点を持ち、本質的な解決方法ではない。ネットワークの状態についての情報が得られていないデータ転送開始直後に、バースト的なパケット廃棄を避けながらウィンドウサイズをすばやく適切な値に設定することは困難である。

そこで Gentle HighSpeed TCP では、コネクション確立直後のスロースタートフェーズにおいて発生するパケットのバースト的な廃棄を防止するのではなく、バースト的なパケット廃棄後に、タイムアウトを発生させず、直ちにパケット送信を再開する方式を提案する。Gentle HighSpeed TCP では、スロースタートフェーズ中に、重複 ACK を受信した場合、従来の TCP Reno のような 1 パケットの再送を行わず、$ssthresh$ と現在のウィンドウサイズを、それぞれ、重複 ACK を受信した時のウィンドウサイズの 1/2 と 1/4 に設定し、再送要求があったパケットから送信を再開する。

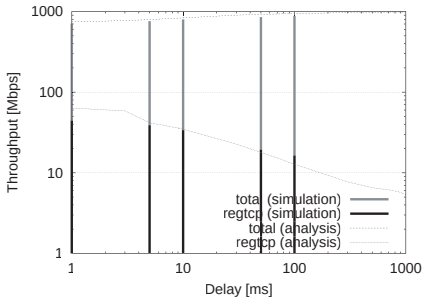
また、従来の TCP Reno と同じく、ウィンドウサイズが $ssthresh$ を越えた場合にスロースタートフェーズから輻輳回避フェーズに移行するが、Gentle HighSpeed TCP においてはさらに、送信側ホストがウィンドウサイズ分のパケット送信を終了する前に、そのウィンドウ内の先頭のパケットに対する ACK を受信した場合にも、スロースタートフェーズから輻輳回避フェーズに移行する。

5.1.2 輻輳回避フェーズ

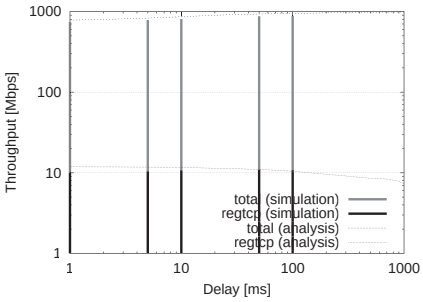
本稿において提案する Gentle HighSpeed TCP は、ネットワークのリンク帯域に余裕がある時は、ウィンドウサイズを大きく上昇させることによって空き帯域を有効に使って、高速な転送を実現するとともに、同じリンクを共有する TCP Reno コネクションとの公平性を維持することを目標とする。Gentle HighSpeed TCP においては、輻輳回避フェーズで、ネットワークのリンク帯域に余裕がある時に大きくウィンドウサイズを上昇させる HighSpeed TCP モードと、ネットワークのリンク帯域を十分に利用できている時に TCP Reno コネクションとの公平性を維持する TCP Reno モードの 2 つのモードから構成されるものとする。以下では、HighSpeed TCP モードおよび TCP Reno モードの詳細なアルゴリズムの説明を行う。

HighSpeed TCP モード

HighSpeed TCP モードにおいては、ウィンドウサイズの増減のアルゴリズムは、HighSpeed TCP と同じアルゴリズムを使用する。これによって、ウィンドウサイズがすばやく増加するため、リンクの空き帯域の効率的な利用が可能になる。HighSpeed



(a) μ_{Reno}=100 [Mbps]



(b) μ_{Reno}=12 [Mbps]

図 6　シミュレーション結果と解析結果

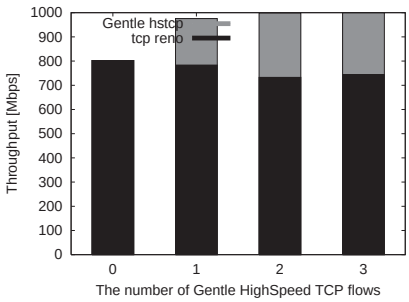


図 7　提案方式の性能評価

TCP モードでは、さらに、リンク帯域を十分に利用しているか否かの判断を行い、リンク帯域を十分に利用していると判断された場合、TCP Reno モードへ移行する。そのために Gentle HighSpeed TCP では、送信したパケットの RTT を利用し、RTT の変化が増加傾向を示していれば、リンク帯域を使い切っていると判断する。具体的なアルゴリズムを以下の通りである。ウィンドウサイズが n の時に送信した n 個のパケットの RTT を、それぞれ、$d_1,\cdots,d_n$ とし、パケットを送信した時刻をそれぞれ $t_1,\cdots,t_n$ とする。$d_1,\cdots,d_n$ と $t_1,\cdots,t_n$ の間に正の相関関係があるか否かを統計的検定によって調べ、相関関係が認められれば、RTT が増加傾向にあると判断し、動作モードを TCP Reno モードへ移行させる。

また、パケット廃棄が発生し、Fast Retransmit アルゴリズムにより再送を行った場合は、HighSpeed モードを継続し、タイムアウトが発生した場合は、ネットワークに輻輳が発生していると考えられるため、通常の TCP Reno と同じように、$ssthresh$ とウィンドウサイズを、それぞれ、現在のウィンドウサイズの半分、1 パケットに設定し、スロースタートフェーズに移行する。

TCP Reno モード

Gentle HighSpeed TCP が TCP Reno モードで動作している場合、ネットワーク中のリンク帯域が十分利用され、ルータバッファにパケットが蓄積されているものと考えられる。このとき、HighSpeed TCP のように、ウィンドウサイズを大きく上昇させると、ルータのバッファに蓄積されるパケット数が増加するだけでなく、競合する TCP Reno コネクションのスループットを大幅に低下させてしまう。また、TCP Vegas [6] のようにウィンドウサイズを減少させると、逆に TCP Reno コネクションにネットワーク帯域を奪われてしまう [7]〜[9]。そこで、TCP Reno モードにおける Gentle HighSpeed TCP のウィンドウサイズの増加幅を、TCP Reno のアルゴリズムと同じにすることにより、TCP Reno と Gentle HighSpeed TCP 間の公平性を維持する。

なお、パケット廃棄が発生し、Fast Retransmit アルゴリズムにより廃棄されたパケットを再送した場合、HighSpeed TCP と同じようにウィンドウサイズを減少させ、HighSpeed TCP モードに移行する。一方、タイムアウトが発生した場合、ネットワークに輻輳が発生しているものと判断し、スロースタートフェーズに移行する。また、TCP Reno モードにおいても、1 パケットごとに RTT の計測を行い、HighSpeed TCP モードと同じ検定方法により、RTT が小さくなっているか否かの検定を行う。検定の結果、RTT が小さくなっていると判断された場合、HighSpeed TCP モードに移行する。

5.2 性能評価

本節では、シミュレーションにより Gentle HighSpeed TCP の性能評価を行う。シミュレーションに用いたネットワークモデルは、3. 章で行ったシミュレーションのネットワークモデルと同じ (図 1) である。図 7 は、HighSpeed TCP コネクションの本数 N に対する、TCP Reno および Gentle HighSpeed TCP を用いたコネクションの合計のスループットを示している。図より Gentle HighSpeed TCP は、ボトルネックリンクをほぼ 100% 利用することができていることがわかる。これは、Gentle HighSpeed TCP はネットワークに空き帯域が存在すると、素早くウィンドウサイズを増加させ、空いている帯域を効率的に利用し、ルータバッファにパケットが蓄積され始めると、ウィンドウサイズの増加幅を縮小するため、バースト的なパケット廃棄が発生せず、タイムアウトにならないためである。

さらに、HighSpeed TCP と比較して、Gentle HighSpeed TCP は共存する TCP Reno コネクションのスループットを減少させていないことが分かる。これは、Gentle HighSpeed TCP はルータバッファにパケットが蓄積され始めると、ウィンドウサイズの増加アルゴリズムを TCP Reno の増加アルゴリズムと同じにするため、TCP Reno コネクションとの公平性が維持できるためである。

以上の結果から、Gentle HighSpeed TCP はネットワークのリンク帯域に余裕がある時は、ウィンドウサイズを大きく上昇させることによって空き帯域を有効に使い、高速な転送を実現するとともに、同じリンクを共有する TCP Reno コネクションとの公平性を維持することができることがわかった。

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における特定領域重点型研究開発プロジェクト「ユビキタスインターネットのための高位レイヤスイッチング技術の研究開発」、および及び平成 14 年度文部科学省科学研究費基盤研究 (B)(13450158) によっている。ここに記して謝意を表す。

文　　献

[1] S. Floyd, “HighSpeed TCP for Large Congestion Windows,” *Internet Draft draft-floyd-tcp-highspeed-01.txt*, Aug. 2002.

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

[3] J. C. Hoe, “Start-up Dynamics of TCP’s Congestion Control and Avoidance Schemes,” *Master Thesis, Massachusetts Institute of Technology*, June 1995.

[4] K. Fall and S. Floyd, “Simulation-based Comparisons of Tahoe, Reno and SACK TCP,” *Computer Communication Review*, vol. 26, pp. 5–21, July 1996.

[5] S. Floyd, “Limited Slow-Start for TCP with Large Congestion Windows,” *Internet Draft draft-floyd-tcp-slowstart-01.txt*, Aug. 2002.

[6] L. S. Brakmo, S. W. O’Malley, and L. L. Peterson, “TCP Vegas: New Techniques for Congestion Detection and Avoidance,” in *Proceedings of IEEE SIGCOMM’ 94*, pp. 24–35, 1994.

[7] O. Ait-Hellal and E. Altman, “Analysis of TCP Vegas and Reno,” *Journal of Telecommunication Systems*, vol. 15, no. 3,4, pp. 381–404, 2000.

[8] G. Hasegawa, K. Kurata, and M. Murata, “Analysis and Improvement of Fairness between TCP Reno and Vegas for Deployment of TCP Vegas to the Internet,” in *Proceedings of IEEE ICNP ’2000*, pp. 177–186, Nov. 2000.

[9] J. Mo, R. J. La, V. Anantharam, and J. Walrand, “Analysis and Comparison of TCP Reno and Vegas,” in *Proceedings of IEEE INFOCOM ’99*, pp. 1556–1563, Mar. 1999.