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.

高速・高遅延ネットワークのためのトランスポート層プロトコル

長谷川　剛†　　村田　正幸††

† 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
†† 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
E-mail: †{hasegawa,murata}@ist.osaka-u.ac.jp

あらまし　1970-80 年代にその基本設計が行われた TCP は、さまざまなネットワークを取り込むことによって大規模化・複雑化している現在のインターネットにおいては、十分な性能を発揮できない場合があることが近年数多く指摘されている。本稿では、それらの問題のうち、高速・高遅延ネットワークにおいて従来の TCP がネットワーク帯域を効率良く利用することができない問題に着目し、その改善手法に関する近年の研究動向の紹介を行う。特に、従来の TCP がパケット廃棄の発生のみをネットワーク輻輳の指標として用いているのに対して、改善手法の多くがラウンドトリップ時間などの新たな指標を導入している点に着目する。また本稿では、我々の研究グループが提案している、エンドホスト間のネットワークパスの帯域情報の計測結果を基にした輻輳制御方式の紹介を行う。
キーワード　TCP、高速・高遅延ネットワーク、輻輳制御方式、トランスポート層プロトコル

Transport-layer protocols for high-speed and long-delay networks

Go HASEGAWA† and Masayuki MURATA††

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

Abstract　In this report, we discuss the modifications of the congestion control mechanism of TCP, which is the major transport-layer protocol in the current Internet, for high-speed and long-delay networks. We categorize them by the network congestion indications which a TCP sender utilizes to increase/decrease its congestion window size. We also introduce recent works of our research group, which is an inline network measurement method which measures the bandwidth information of the network path between sender and receiver hosts, and TCP Symbiosis which control its congestion window size based on the measured bandwidth information.
Key words　TCP, high-speed and long-delay networks, congestion control mechanism, transport-layer protocols

1. ま え が き

現在のインターネットにおける多くのネットワークアプリケーションは、トランスポート層プロトコルとして Transmission Control Protocol (TCP) [1] を用いており、TCP トラヒックは現在のインターネットトラヒックの大部分を占めている [2]。また、インターネットは様々な種類のネットワークを取り込むことによって大規模化し、指数関数的な拡大を続けている [3]。その結果、TCP が誕生した 1970 年代当初には想定することができなかったネットワーク環境が発生し、TCP によるデータ転送の問題点が明らかになりつつある。例えば、ネットワーク輻輳に加えてリンクエラーによりパケット廃棄が発生する無線ネットワーク [4, 5] や、端末の移動によりスループット低下が発生するモバイル環境 [6] などが挙げられる。これらを始めとするさまざまな問題を解決するために、これまでに多くの TCP に対する改善が行われてきた (例えば [7-9])。

また、近年のネットワークの高速化により、TCP コネクションが利用できるネットワークの帯域遅延積 (リンク帯域とエンドホスト間の伝播遅延時間の積) が飛躍的に増大している。例えば、ラウンドトリップ時間 (RTT) が約 130 msec となる太平洋を狭んだ 2 台のエンドホスト間の最低帯域が 100 Mbps から 1 Gbps 程度である、という環境も一般に利用可能となりつつある [10]。このような高速・高遅延ネットワーク環境において、現在多くの OS の TCP 実装が基本としている TCP Reno を用いると、その輻輳制御方式の特徴が原因となって、リンク帯域を十分使うことができない、という問題が指摘されている。これは、TCP Reno が旧来の低速ネットワークを想定して設計されていること、またインターネットユーザがよりスループットなどの性能に敏感になっていることなどに起因していると考えられる。

この問題に対し、高速・高遅延ネットワークのための TCP の輻輳制御方式の改善手法が数多く提案されている [11-18] 。それらの改善手法の多くは、ネットワーク輻輳の指標としてパ

ケット廃棄だけではなく、ラウンドトリップ時間の変動、現在のデータ転送速度やウィンドウサイズそのものなどを指標として用いている。本稿では、それらの改善手法を紹介するとともに、その特徴を抽出し、エンドホスト間プロトコルである TCP が高速・高遅延ネットワーク環境に対応するための方策について議論する。さらに本稿では、我々の研究グループで提案している、アクティブな TCP コネクションが送受信しているデータ・ACK パケットを利用して、エンドホスト間ネットワークパスの帯域に関する情報を取得するインラインネットワーク計測技術、および取得した情報を利用し、数理生態学における個体数増殖モデルに基いて輻輳ウィンドウサイズを制御する TCP Symbiosis 方式を紹介し、既存方式に対する優位性について述べる。

以下、2. 章では TCP Reno の輻輳制御方式を簡単に説明し、高速・高遅延ネットワーク環境における問題点を指摘する。3. 章においては既存の改善手法を紹介する。さらに、我々の研究グループで提案しているインラインネットワーク計測手法、および TCP Symbiosis 手法の紹介を 4. 章で行う。最後に 5. 章で本稿のまとめと今後の課題について述べる。

2. TCP Reno の輻輳制御方式とその欠点

本章では、現在多くの OS の TCP 実装のベースとなっている TCP Reno の輻輳制御方式のうち、輻輳ウィンドウサイズの増減アルゴリズムについて簡単に紹介し、本稿で着目する高速・高遅延ネットワークにおける問題点を指摘する。なお本稿はウィンドウベースの輻輳制御など、TCP の輻輳制御方式の基本的な知識を前提としている。TCP の輻輳制御方式などの詳細については [19] などを参照されたい。

2.1 TCP Reno の輻輳制御方式

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、それぞれにおいて輻輳ウィンドウサイズ (cwnd) 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輻輳ウィンドウサイズ 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輻輳ウィンドウサイズをその逆数分だけ増加させる。すなわち、TCP Reno の輻輳ウィンドウサイズを w_{reno} とすると、その更新アルゴリズムは以下のように表すことができる。

$$w_{\mathrm{reno}} \leftarrow \begin{cases} w_{\mathrm{reno}} + 1 & (w_{\mathrm{reno}} < s_{\mathrm{reno}}) \\ w_{\mathrm{reno}} + \dfrac{1}{w_{\mathrm{reno}}} & (w_{\mathrm{reno}} \geq s_{\mathrm{reno}}) \end{cases} \quad (1)$$

ここで、s_{reno} は、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 ssthresh である。一方、パケット廃棄を検出した場合には、次式のように輻輳ウィンドウサイズを減少させる。

$$w_{\mathrm{reno}} \leftarrow \begin{cases} w_{\mathrm{reno}}/2 & (\text{重複 ACK による検出}) \\ 1 & (\text{タイムアウトによる検出}) \end{cases} \quad (2)$$

すなわち、TCP Reno はパケット廃棄を検出するまで輻輳ウィンドウサイズを増加させ続け、パケット廃棄をきっかけに減少させる。これは、TCP Reno がパケット廃棄の発生をネットワーク輻輳の指標と見なしていることに起因する。本稿ではこのようにパケット廃棄をネットワーク輻輳の指標として用いる手法を loss-based 手法と呼ぶ。

2.2 高速・高遅延ネットワーク環境における問題点

上述した TCP Reno を高速・高遅延ネットワーク環境において用いると、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 [11]。図 1 に、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の、輻輳ウィンドウサイズの変化の様子を示す。この環境においては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 [11]。これは現在の光ファイバ技術では実現困難な性能である。また図 1 から、一度パケット廃棄が発生すると、輻輳ウィンドウサイズが回復するまでに、40000 RTT (約 4000 秒) 以上の時間を要することがわかる。これは、TCP Reno において 1 Gbps を超えるスループットを継続的に得ることは現実的には不可能であることを表している。この問題は、式 (1) からわかるように、輻輳ウィンドウサイズの増加の幅がラウンドトリップ時間 (RTT) ごとに 1 パケットと非常に小さいにもかかわらず、式 (2) に示すように、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、輻輳ウィンドウサイズがなかなか大きくならないことに起因している。

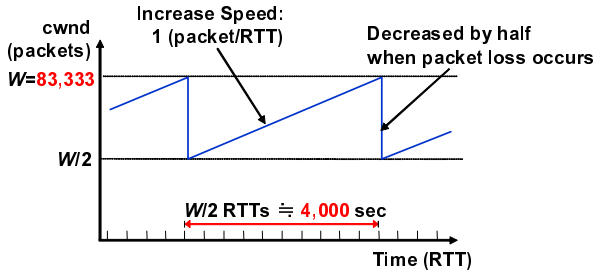


図 1　TCP Reno の高速・高遅延環境における問題点

3. 既存の改善方式

2.2 節において指摘した高速・高遅延ネットワークにおける TCP Reno の問題点に対する数多くの改善手法が近年提案されている。本章では、それらのうち主要なものを紹介する。それらの改善手法の多くは、TCP Reno がパケット廃棄をネットワーク輻輳の指標として判断しているのに対して (あるいはそれに加えて)、別の指標を導入している。本章では新たに導入している指標によって分類を行っている。また本章では、特に指定しない限りは、輻輳回避フェーズにおける輻輳ウィンドウサイズの増減アルゴリズムについて説明する。なお、スロースタートフェーズに関しては、多くの改善手法が TCP Reno と同じアルゴリズム (式 (1) の 1 行目) を採用している。また、輻輳回避フェーズにおいては、輻輳ウィンドウサイズがある値以下である場合には、TCP Reno と同じアルゴリズム (2.1 節) を用いることで、低速ネットワーク環境における TCP Reno との親和性を確保している手法も存在する。

3.1 Loss-based 手法

3.1.1 HighSpeed TCP (HSTCP) [11]

HSTCP は、高速・高遅延環境向けの改善手法として比較的初期に提案された手法である。HSTCP は TCP Reno と同様、パケット廃棄のみをネットワーク輻輳の指標として利用するが、現在の輻輳ウィンドウサイズの大きさに合わせて、輻輳ウィンドウサイズの増加速度およびパケット廃棄検出時の減少幅を次式にしたがって調整する。

$$w_{\mathrm{hstcp}} \leftarrow \begin{cases} (\text{パケット廃棄未検出時}) \\ w_{\mathrm{hstcp}} + \dfrac{a(w_{\mathrm{hstcp}})}{w_{\mathrm{hstcp}}} \\ (\text{パケット廃棄検出時}) \\ (1 - b(w_{\mathrm{hstcp}}))w_{\mathrm{hstcp}} \end{cases} \quad (3)$$

$$a(w) = \frac{2w^2 \cdot b(w) \cdot p(w)}{2 - b(w)}$$

$$b(w) = \frac{\log(w) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})}(b_{high} - 0.5) + 0.5$$

$$p(w) = \exp\left[\frac{\log(w)-\log(W_{low})}{\log(W_{high})-\log(W_{low})} \cdot\{\log(P_{high})-\log(P_{low})\}+\log(P_{low})\right]$$

なお、TCP Reno は $a(w)=1$ および $b(w)=1/2$ に相当する。また、P_{high}、P_{low}、W_{high} および W_{low} のパラメータは、ネットワークのパケット廃棄率と目標とするスループットから算出される値であり、[11] においてはパケットサイズが 1500 バイト、パケット廃棄率が 10^{-7} の環境において、10 Gbps のスループットが達成できるようなパラメータが一例として紹介されている。この式は、HSTCP は現在の輻輳ウィンドウサイズが大きいほど、その増加速度を大きくし、減少幅を小さくすることを意味している。これにより、HSTCP はネットワークの帯域遅延積の大きさに応じたウィンドウサイズの増加・減少速度を用いることができる。

しかし、HSTCP や後述する Scalable TCP などの、TCP Reno に比べて積極的に輻輳ウィンドウサイズを増加させるプロトコルが、TCP Reno コネクションと共存した場合、それらのプロトコルは高いスループットを獲得する反面、共存する TCP Reno コネクションのスループットを低下させるという問題点がある。この問題に対する改善手法として我々の研究グループでは gentle HighSpeed TCP 手法を提案している [14]。

3.1.2 Scalable TCP (STCP) [12]

STCP は、TCP Reno の輻輳回避フェーズのような輻輳ウィンドウサイズを直線的に増加させるフェーズは持たず、スロースタートフェーズと同じ、輻輳ウィンドウサイズを指数的に増加させる式のみを用いる。

$$w_{\mathrm{stcp}} \leftarrow \begin{cases} w_{\mathrm{stcp}}+0.01 & (\text{パケット廃棄未検出時}) \\ w_{\mathrm{stcp}}\cdot 0.875 & (\text{パケット廃棄検出時}) \end{cases} \quad (4)$$

STCP は、ウィンドウサイズが現在の値から k 倍になるために必要な RTT 数が、現在のウィンドウサイズの値そのものに依存せず一定となる、という特徴を持つ。これにより、リンク帯域の大きさに関係なく、一定のリンク利用率を維持することができる。また式から、TCP Reno や HSTCP が Additive Increase Multiplicative Decrease (AIMD) にしたがってウィンドウサイズを増減するのに対して、Multiplicative Increase Multiplicative Decrease (MIMD) を採用していることがわかる。[20] によると、ボトルネックリンクを共有している全ての送信側端末へ、ネットワーク輻輳の指標が同時に伝達される環境においては、AIMD はフロー間の公平性を維持することができるが、MIMD では公平性を維持することができない。したがって、STCP は、4.2 節で示すように、RTT が同じ STCP コネクション間においても公平性を達成できないという問題を持つ。

3.2 Delay-based 手法

Loss-based 手法はネットワーク内でパケット廃棄が発生するまで輻輳ウィンドウサイズの増加を停止しないため、理想的に動作した場合においても、周期的なパケット廃棄の発生を避けることができない。一方、ルータの出力リンクにおいて高負荷時にパケットが蓄積されるバッファが FIFO 規律に従っている場合、バッファが一杯になりパケット廃棄が発生する前に、そのリンクにおいてバッファリング遅延が増大することが期待される。そこで、各パケットの RTT を監視し、その増大をネットワーク輻輳の初期段階の指標として利用する手法 (本稿では delay-based 手法と呼ぶ) が提案されている。理想的に動作すると、loss-based 手法では避けることのできないパケット廃棄を完全に回避することが可能となる。

3.2.1 TCP Vegas [21]

TCP Vegas は 1995 年に発表された手法であり、高速・高遅延ネットワーク向けに提案された手法ではないが、その基本アルゴリズムはその後登場した多くの手法において利用されている。TCP Vegas においては、以下の式を用いることによって、ネットワーク内で滞留している (バッファリングされている) と考えられるパケット数を推測する。

$$Expected = cwnd/baseRTT \quad (5)$$

$$Actual = cwnd/RTT \quad (6)$$

$$Diff = (Expected - Actual)\cdot baseRTT \quad (7)$$

ここで、$cwnd$ は輻輳ウィンドウサイズ、$baseRTT$ はこれまでに観測された最小の RTT、RTT は現在の RTT、$Diff$ はネットワーク内滞留パケット数の推測値である。TCP Vegas は $Diff$ の値に基づいて以下の式にしたがって輻輳ウィンドウサイズを RTT に 1 回増減させる。

$$w_{\mathrm{vegas}} \leftarrow \begin{cases} w_{\mathrm{vegas}}+1, & \text{if } Diff < \frac{\alpha}{base\_rtt} \\ w_{\mathrm{vegas}}, & \text{if } \frac{\alpha}{base\_rtt} < Diff < \frac{\beta}{base\_rtt} \\ w_{\mathrm{vegas}}-1, & \text{if } \frac{\beta}{base\_rtt} < Diff \end{cases}$$

すなわち、パケットの RTT の増大にともない $Diff$ が大きくなると、パケット廃棄が発生していなくても輻輳ウィンドウサイズを小さくする。これにより、周期的なパケット廃棄を避けることができる。

しかし、輻輳ウィンドウサイズの増加・減少速度が TCP Reno の輻輳ウィンドウサイズの増加速度である 1 packet/RTT であるため、高速・高遅延ネットワーク環境においては TCP Reno と同様の問題を持つと考えられる。

3.2.2 FAST TCP [13]

FAST TCP は TCP Vegas と同様に、観測された最小の RTT である $baseRTT$ と現在の RTT である RTT を利用し、以下の式にしたがって輻輳ウィンドウサイズを増減させる。

$$w_{\mathrm{fast}} \leftarrow \min\left\{2w_{\mathrm{fast}}, (1-\gamma)w_{\mathrm{fast}} + \left(\frac{baseRTT}{RTT}w_{\mathrm{fast}}+\alpha\right)\right\}$$

α はパラメータであり、ネットワーク内滞留パケット数の目標値に相当する。TCP Vegas と異なり、目標となる輻輳ウィンドウサイズが現在値に比べて大きい場合には輻輳ウィンドウサイズを指数的に増加させるため、ネットワークの帯域遅延積が大きい場合にもすばやくネットワーク利用率を向上させることができる点が特長である。その反面、パラメータ α の適切な設定がやや難しいという欠点を持つ。

3.3 Delay-based 手法と loss-based 手法の組み合わせ

TCP Vegas や FAST TCP などの delay-based 手法は、それが単独で用いられる場合にはスループット、公平性、収束速度などの面で優れていることが明らかになっている。しかし、TCP Reno や HSTCP のような loss-based 手法と混在した環境においては、delay-based 手法を用いるコネクションのスループットが低下するという問題が [22, 23] などにおいて指摘されている。これは以下の理由による。混在環境においてネットワーク帯域が使い切られ、ルータバッファにパケットが蓄積し始めると、RTT が増加する。その際、delay-based 手法は RTT の増加にともない輻輳ウィンドウサイズを小さくするが、loss-based 手法はパケット廃棄が発生するまで輻輳ウィンドウサイズを大きくし続ける。したがって、ボトルネックリンクを両手法のコネクションが共有した場合、delay-based 手法のコネクションは、loss-based 手法のコネクションに比べてスループットが低下する。

この問題に対し、delay-based 手法に loss-based 手法を組み合わせる手法が提案されている ([15, 16])。これらの手法は、通常の delay-based 手法と同様に、RTT が増加しておらずネットワーク帯域が使い切られていないと判断された場合には輻輳ウィンドウサイズを TCP Reno よりも大きく増加させる。その際、[16] においては TCP Reno と同じ速度で輻輳ウィンドウサイズを増加した場合の仮想値を管理しておく。また [15] においては、輻輳ウィンドウサイズの増加幅を TCP Reno 相当の部分と、そうでない追加部分に分けて管理する。その後、RTT が増加し始めると、輻輳ウィンドウサイズの (大幅な) 増加を停止する。その後、TCP Reno と同じ増加幅で輻輳ウィンド

ウサイズを増加させ、loss-based 手法を用いる (パケット廃棄の発生まで輻輳ウィンドウサイズを大きくし続ける)。すなわち、ネットワークの未使用帯域がある場合には、delay-based 手法によってそれを高速に使い切るように動作し、輻輳時には loss-based 手法で動作することによって、共存する TCP Reno との公平性を維持している。

3.4 その他の手法

その他、パケット廃棄が発生した時の輻輳ウィンドウサイズを記憶し、その値を基にその後の輻輳ウィンドウサイズの制御を行う BIC-TCP [17] や CUBIC-TCP [18]、また ACK パケットの到着間隔から現在のスループットを推測し、その値を輻輳ウィンドウサイズの制御に用いる TCP Westwood [24] などが提案されている。

3.5 並列 TCP 手法

これまでに紹介した改善手法は TCP の輻輳制御方式そのものを変更することによって問題を解決する手法であるが、その他の手法として、複数本の TCP (Reno) コネクションを並列的に用いてデータ転送を行う並列 TCP 手法が考えられる。並列 TCP 手法は OS のカーネルの改変が必要なく、アプリケーションプログラムによって実現可能であるため、アプリケーションのデータ転送スループットを向上させる方法としては非常に有用である。例えば GridFTP [25] には並列 TCP 手法によるデータ転送方式が組み込まれている。

並列 TCP 手法によって効率的なデータ転送を行う際には、同時に利用する TCP コネクション数の適切な設定が重要である。例えば [26] においては並列 TCP 手法によるデータ転送スループットを数学的解析によって導出している。また [27] においては並列コネクション数を動的に調整する手法が提案されている。しかし、適切なコネクション数はリンク帯域、伝播遅延時間、競合するコネクション数、用いる TCP の輻輳制御方式の特性など、多くのパラメータに大きく依存する [28]。また、並列 TCP コネクション数の動的な制御は、上述した TCP の輻輳制御方式の改善手法に比べてネットワーク環境の変動への追随性や端末負荷などの面で劣ると考えられる。

4. Bandwidth-based 手法

我々の研究グループにおいても、高速・高遅延ネットワークに対応するための TCP の輻輳制御方式の改善手法を提案している。提案手法が 3. 章で紹介した既存手法と大きく異なるのは、ネットワーク輻輳の指標として、パケット廃棄やラウンドトリップ時間などの間接的な指標を使うのではなく、エンド間パスの帯域に関する情報を直接的に用いる (本稿では Bandwidth-based 手法と呼ぶ) 点、および、輻輳ウィンドウサイズの増減アルゴリズムに数理生態学における個体数増殖モデルを用いている点にある。本章では、エンド間パスの帯域情報を取得するためのインラインネットワーク計測手法、および取得した帯域情報を基に輻輳ウィンドウサイズの増減を行う手法である TCP Symbiosis の紹介を行う。

4.1 インラインネットワーク計測

エンドホスト間ネットワークパスの帯域情報を計測するための手法はこれまでにも多く提案されている (例えば [29-32]) が、それらのツールは計測に長い時間がかかる、高いレートで計測用パケットを送出するためネットワークに与える影響が大きい、などの問題がある。それに対して我々の研究グループにおいては、エンドホスト間の物理帯域および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。提案手法は計測用のパケットを新たにネットワークへ送出するのではなく、アプリケーションデータの転送などに用いられているアクティブな TCP コネクションが送受信するデータ/ACK パケットを利用して計測を行う (図 2)。すなわち、送信側 TCP から送出されるデータパケットの送出間隔を帯域計測アルゴリズムに基いて調整し、対応する ACK パケットの到着間隔を観測することで、エンドホスト間パスの物理帯域 [33] および利用可能帯域 [34, 35] を計測することができる。提案手法は計測用パケットを必要としないため、ネットワークに影響を与えることなく高い精度のアクティブ計測を可能としている。また、非常に短い周期 (1-4 RTT) で継続的に計測結果を取得することができるため、ネットワーク状況の変化に素早く追随することができる。

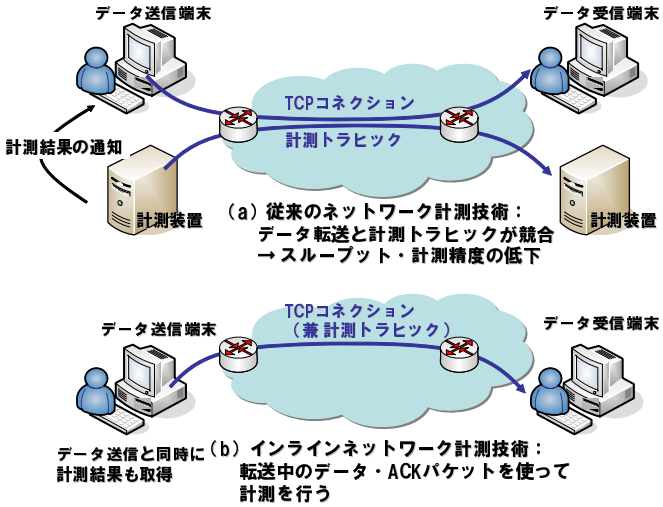


図 2　インラインネットワーク計測手法

取得した帯域情報は上位のアプリケーションにおけるサーバ/ピア選択やオーバレイパスの切り替えなどに用いることができる。また、TCP のデータ転送そのものに帯域情報を利用することで、より高度なトランスポートサービスを実現することが可能となる。我々の研究グループにおいては、インラインネットワーク計測によって得られた情報を用いた TCP によるトランスポートサービスとして、バックグラウンド転送 [36]、スループットを確保したデータ転送 [37]、次節で紹介する高速・高遅延環境向けデータ転送 [38] などを実現する手法を提案している。また、インライン計測手法および応用手法の FreeBSD および Linux の実装コードを Web サイトにて公開している [39]。

4.2 TCP Symbiosis

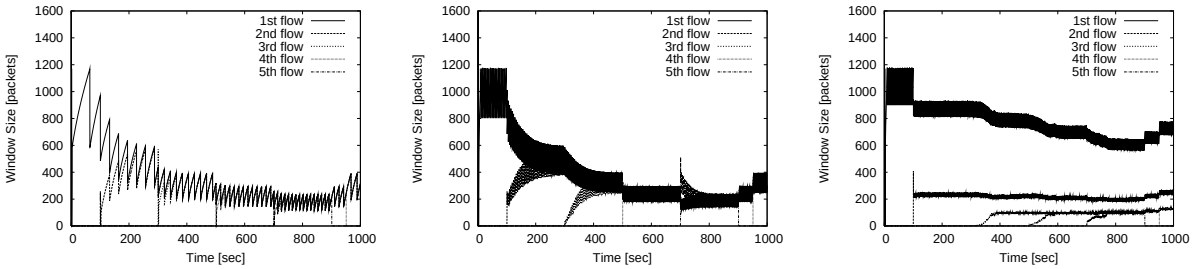
[38] において提案している TCP Symbiosis は、前述したインラインネットワーク計測手法を用いることによって送受信端末間の物理帯域および利用可能帯域を取得し、数理生態学において生物の個体数の変化を表すモデルであるロジスティック増殖モデル、およびロトカ・ヴォルテラ競争モデル [40] を適用したウィンドウサイズ制御アルゴリズムを用いて輻輳制御を行う。具体的には、生物の個体数の TCP のデータ転送速度、環境容量をエンドホスト間パスの物理帯域とみなして適用する。また、元のモデルは自分以外の生物種の個体数が既知であることが前提とされているため、提案手法においては、計測で得られた物理帯域と利用可能帯域を基に、他のコネクションが利用している帯域量を推測する。TCP Symbiosis においては、以下の式に基いて輻輳ウィンドウサイズを制御する。

$$\frac{d}{dt}w_{\mathrm{sym}} = \epsilon\left(1 - \frac{w_{\mathrm{sym}} + \gamma(K-A)\tau}{K\tau}\right)w_{\mathrm{sym}} \qquad (8)$$

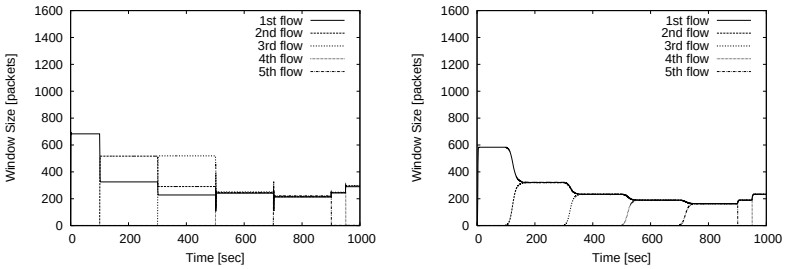
ここで w_{sym} は輻輳ウィンドウサイズ、t は時刻、τ は TCP コネクションの最小 RTT、K および A はエンドホスト間ネットワークパスの物理帯域および利用可能帯域である。ϵ および γ はロトカ・ヴォルテラ競争モデルが持つパラメータであり、それぞれ個体の内的自然増殖率および種内・種間競争係数比を表す。

$$w_{\mathrm{sym}}(t) = \frac{w_{\mathrm{sym}}(0)\tau f(t)\{K-\gamma(K-A)\}}{w_{\mathrm{sym}}(0)(f(t)-1)+\tau\{K-\gamma(K-A)\}} \qquad (9)$$

ここで、$f(t) = e^{\epsilon t\left\{1-\gamma\left(1-\frac{A}{K}\right)\right\}}$ である。TCP Symbiosis の大きな特長の 1 つに、既存の改善手法の多くが解決していない、



(a) TCP Reno (b) HSTCP (c) STCP



(d) FAST TCP (e) TCP Symbiosis

図 3 シミュレーション結果

RTT の違いに起因するスループット不公平性を大きく改善していることが挙げられる [38]。

シミュレーション結果の一例を図 3 に示す。ダンベル型のネットワークにおいて、5 本の TCP コネクションがシミュレーション開始からそれぞれ 0、100、300、500、700 秒後にネットワーク内に参加してデータ転送を開始し、それぞれ 900、950、1000、1050、1100 秒後にデータ転送を終了して離脱する。ボトルネックリンクのリンク帯域は 100 Mbps、伝播遅延時間は 25 msec とし、TCP コネクションの最小 RTT は約 70 msec となる。図 3 は、TCP Reno、HSTCP、STCP、FAST TCP、および TCP Symbiosis における、シミュレーション時間に対する各コネクションのウィンドウサイズの変化を示している。図 3(a) および図 3(b) から、TCP Reno と HSTCP は、AIMD 方式によるウィンドウサイズ制御を行うため、パケット廃棄を繰り返すことによって新たに参加したコネクションと既存のコネクションとの間の公平性が実現されていることがわかる。図 3(c) から、STCP においても周期的なパケット廃棄が発生するが、コネクション間の不公平性が発生している。これは、3. 1. 2 節で述べたように、STCP が MIMD 方式によるウィンドウサイズの制御を行うためであると考えられる。

また、図 3(d) から、FAST TCP においては後からネットワークに参加するコネクションの輻輳ウィンドウサイズが小さくなり、コネクション間の公平性が損なわれる場合があることがわかる。これは、すでに存在するコネクションによってリンク帯域が使い切られ、ボトルネックリンクにおいてキューイングが常に発生している状態であるため、新たに参加するコネクションが制御に必要とする、ボトルネックリンクでキューイングが発生していない状態での RTT を計測することができないためである。一方、コネクションが離脱した際に、残りのコネクション間で公平性を保つことができるのは、一時的にキューが空の状態となり、各コネクションが RTT の最小値を正しく計測することができるからである。一方、提案方式は、計測した帯域情報をウィンドウサイズの制御に用いる。そのため、新たなコネクションがデータ転送を開始した際には利用可能帯域の情報が更新され、パケット廃棄を発生させることなくウィンドウサイズを調節し、さらにコネクション間の公平性をすばやく実現することができることが図 3(e) からわかる。

5. おわりに

本稿では、高速・高遅延ネットワークにおけるトランスポート層プロトコルの輻輳制御方式について、近年の研究動向をまとめると共に、我々の研究グループにおける提案方式の紹介を行った。

Delay-based 手法は loss-based 手法に比べてネットワーク輻輳を早期段階で検出することができる。しかし、今後さらにネットワークのリンク帯域が増大すると、伝播遅延時間は小さくならないため、相対的にバッファリング時間の絶対値およびその変動の粒度が小さくなる。CPU などの端末の処理速度はネットワーク帯域に比べて高速化の速度が鈍いため、端末における RTT 変動の検出は、今後困難になると考えられる。そのような環境においては、我々の研究グループが提案している bandwidth-based 手法が有用であると考えられる。

Delay-based 手法と loss-based 手法とを組み合わせた手法は、高速性を維持しつつ TCP Reno との公平性を維持することができるため、その有用性は高いと考えられる。例えば Compound TCP は、Windows Vista に搭載されている [41]。しかしその方針は、loss-based 手法の TCP と競合している場合は、自身も loss-based 手法を使う、というものであり、loss-based 手法の持つ本質的な問題である、周期的なネットワーク輻輳およびパケット廃棄を回避できない、という点を解決できていないと考えられる。一方、純粋な delay-based 手法や我々が提案している bandwidth-based 手法は、loss-based 手法よりも早期にネットワーク輻輳を検出することによってパケット廃棄を完全に回避することができる。今後は、純粋な delay-based 手法や bandwidth-based 手法のエンドホストへの効率的な適用方法に関する検討が、今後の課題として挙げられる。

文 献

[1] J. B. Postel, “Transmission control protocol,” *Request for Comments 793*, Sept. 1981.

[2] M. Fomenkov, K. Keys, D. Moore, and k claffy, “Longitudinal study of Internet traffic in 1998-2003,” in *Proceedings of*

Winter International Symposium on Information and Communication Technologies (WISICT 2004), Jan. 2004.

[3] Hobbes ' Internet timeline v8.2. available at `http://www.caida.org/home/`.

[4] F. Lefevre and G. Vivier, "Understanding TCP's behavior over wireless links," in *Proceedings of Communications and Vehicular Technology*, pp. 123–130, Oct. 2000.

[5] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *Wireless Communications and Mobile Computing*, vol. 2, pp. 3–20, Feb. 2002.

[6] T. Goff, J. Moronski, D.S.Phatak, and V. Gupta, "FreezeTCP: A true end-to-end TCP enhancement mechanism for mobile environments," in *Proceedings of IEEE INFOCOM 2000*, Mar. 2000.

[7] V. Jacobson and R. Braden, "TCP extensions for long-delay paths," *Request for Comments 1072*, Oct. 1988.

[8] S. Floyd and T. Henderson, "The NewReno modification to TCP's fast recovery algorithm," *Request for Comments 2582*, Apr. 1999.

[9] S. Floyd, "A proposal to add explicit congestion notification (ECN) to IP," *RFC 2481*, Jan. 1999.

[10] C. Marcondes, A. Persson, M. Sanadidi, M. Gerla, H. Shimonishi, T. Hama, and T. Murase, "Inline path characteristic estimation to improve TCP performance in high bandwidth-delay networks," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[11] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments 3649 (Experimental)*, Dec. 2003.

[12] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 32, Apr. 2003.

[13] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[14] Z. Zhang, G. Hasegawa, and M. Murata, "Performance analysis and improvement of HighSpeed TCP with Tail-Drop/RED routers," *IEICE Transactions on Communications*, vol. E88-B, pp. 2495–2507, June 2005.

[15] K. T. J. Song, Q. Zhang, and M. Sridharan, "Compound TCP: A scalable and TCP-friendly congestion control for high-speed networks," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[16] H. Shimonishi, T. Hama, and T. Murase, "TCP-Adaptive Reno: Improving efficiency-friendliness tradeoffs of TCP congestion control algorithm," in *Proceedings of PFLDnet 2006*, Feb. 2006.

[17] L. Xu, K. Harfoush, and I. Rhee, "Binary increase congestion control for fast long-distance networks," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.

[18] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *Proceedings of PFLDnet 2005*, Feb. 2005.

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

[20] D.-M. Chiu and R. Jain, "Analysis of the increase and decrease algorithms for congestion avoidance in computer networks," *Journal of Computer Networks and ISDN Systems*, vol. 17, pp. 1–14, June 1989.

[21] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, Oct. 1995.

[22] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP reno and vegas," in *Proceedings of IEEE INFOCOM'99*, Mar. 1999.

[23] 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*, Nov. 2000.

[24] TCP WESTWOOD Home Page. available at `http://www.cs.ucla.edu/NRL/hpi/tcpw/`.

[25] W. Allcock, "GridFTP: Protocol extensions to FTP for the Grid," *Available at: http://www.ggf.org/documents/GFD.20.pdf*, Apr. 2003.

[26] T. Hacker and B. Athey, "The end-to-end performance effects of parallel TCP sockets on a lossy wide-area network," in *Proceedings of the 16th IEEE-CS/ACM International Parallel and Distributed Processing Symposium (IPDPS)*, Aug. 2001.

[27] T. Ito, H. Ohsaki, and M. Imase, "Automatic parameter configuration mechanism for data transfer protocol GridFTP," in *Proceedings of the 2006 International Symposium on Applications and the Internet (SAINT 2006)*, Jan. 2006.

[28] Z. Zhang, G. Hasegawa, and M. Murata, "Reasons not to parallelize TCP connections for long fat networks," in *Proceedings of SPECTS 2006*, Aug. 2006.

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

[30] J. Strauss, D. Katabi, and F. Kaashoek, "A measurement study of available bandwidth estimation tools," in *Proceedings of Internet Measurement Conference 2003*, Oct. 2003.

[31] J. Navratil and R. Cottrell, "ABwE: A practical approach to available bandwidth estimation," in *Proceedings of the 4th Passive and Active Measurement Workshop (PAM 2003)*, Apr. 2003.

[32] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "PathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of the 4th Passive and Active Measurement Workshop (PAM 2003)*, Apr. 2003.

[33] L. T. M. Cao, 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.

[34] L. T. M. Cao, G. Hasegawa, and M. Murata, "ImTCP: TCP with an inline measurement mechanism for available bandwidth," *Computer Communications Journal special issue of Monitoring and Measurements of IP Networks*, vol. 29, pp. 1614–1626, June 2006.

[35] L. T. M. Cao, G. Hasegawa, and M. Murata, "ICIM: An inline network measurement mechanism for highspeed networks," in *Proceedings of NOMS 2006 E2EMON Workshop 2006*, Apr. 2006.

[36] T. Tsugawa, G. Hasegawa, and M. Murata, "Background TCP data transfer with inline network measurement," *IEICE Transactions on Communications*, vol. E89-B, pp. 2152–2160, Aug. 2006.

[37] 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.

[38] G. Hasegawa and M. Murata, "TCP symbiosis: congestion control mechanisms of TCP based on Lotka-Volterra competition model,"

[39] ImTCP's Homepage. available at `http://www.anarg.jp/imtcp/`.

[40] J. D. Murray, *Mathematical Biology I: An Introduction*. Springer Verlag Published, 2002.

[41] Microsoft Corporation, "The Cable Guy - 2005 年 10 月: Windows Vista および Windows Server "Longhorn "の IPv6 への変更," *Available at: http://www.microsoft.com/japan/technet/community/columns/cableguy/cg100%5.mspx*, June 2006.