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.

帯域計測に基づくTCPの輻輳制御方式の提案と評価

井口　智仁† 　長谷川　剛†† 　村田　正幸†

† 大阪大学大学院情報科学研究科　〒 565–0871 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 豊中市待兼山町 1–32
E-mail: †{t-iguti,murata}@ist.osaka-u.ac.jp, ††hasegawa@cmc.osaka-u.ac.jp

あらまし　本稿では，計測手法を用いることでネットワークの帯域に関する情報を取得し，その情報を使用したウィンドウサイズ制御アルゴリズムによって輻輳制御を行う，新しいTCPの輻輳制御方式を提案する．従来の方式であるTCP Renoはパケット廃棄を検出してネットワークの輻輳の有無を判断するが，提案方式はTCPのデータ転送と併用することが可能なインライン計測手法を用いて，送受信端末間の物理帯域および利用可能帯域に関する情報を直接取得する．そして，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を適用したアルゴリズムに基づいてウィンドウサイズを調節する．これはTCP Renoやその改善手法とは全く異なる方式であり，周期的なパケット廃棄を必要としない特徴を持つ．解析およびシミュレーションによる性能評価を通じて，提案方式がネットワークの帯域遅延積に対するスケーラビリティ，収束時間，コネクション間の公平性，安定性などに関して従来手法よりも優れた性質を持つことを示す．

キーワード　TCP, 輻輳制御, インライン計測手法, 利用可能帯域, ロジスティック増殖モデル, ロトカ・ヴォルテラ競争モデル

A New Congestion Control Mechanism of TCP based on Bandwidth Measurement

Tomohito IGUCHI†, Go HASEGAWA††, and Masayuki MURATA†

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

Abstract　In this paper, we propose a new congestion control mechanism of TCP, using bandwidth information obtained by inline network measurement to control the congestion window size. Whereas the traditional TCP Reno recognizes the network congestion by detecting packet losses, the proposed mechanism obtains directly the information of physical and available bandwidths by using an inline network measurement technique. It adjusts the congestion window size by using an algorithm based on logistic model and Lotka-Volterra competition model from biophysics. The proposed mechanism is intensively investigated through analysis and simulation evaluations, and we show the effectiveness of the proposed mechanism in terms of scalability with the network bandwidth-delay product, convergence time, fairness among connections, and stability.

Key words　TCP, congestion control, inline network measurement, available bandwidth, logistic model, Lotka–Volterra competition model

1. はじめに

Transmission Control Protocol (TCP) [1] は，現在のインターネットアプリケーションが標準的に使用するトランスポート層プロトコルである．TCPは1970年代に設計され，それを規定する最初のRequest for Comments (RFC) は1981年に発行された [2]．そして，その後のインターネットの発展に伴い改良が何度も行われている [3-5].

TCPは様々な機能を持っているが，その中でも重要なものは輻輳制御機構 [1] である．この機構により，TCPはネットワークの輻輳を回避・解消し，ネットワークの帯域を競合する複数のコネクション間で公平に分配するためにデータ転送速度を調節する．TCPはウィンドウサイズ (確認応答無しに一度に送出できるデータ転送量) を増減させることによってデータ転送速度の調節を行う．現在，TCPはRenoと呼ばれるバージョンが最も広く用いられている．TCP Renoのウィンドウサイズ制御アルゴリズムは，パケット廃棄の発生を検出するまでウィ

ンドウサイズを線形的に増加させ，パケット廃棄の発生を検出するとウィンドウサイズを半減させるというものである．この制御方式はその増減方法から Additive Increase Multiplicative Decrease (AIMD) 方式と呼ばれている．この方式を用いることで，ネットワーク輻輳に関する情報が各コネクションへ同時に伝わるのであれば，複数の TCP コネクションが独立してウィンドウサイズを制御しても，それらの間でネットワークの帯域を有効にかつ公平に利用できることが知られている [6].

しかし，近年インターネット環境の多様性，複雑性が増大してきたことによって，TCP Reno には多くの問題点があることが明らかになってきている [7-11]．これらの問題の多くは，TCP がパケット廃棄の発生をネットワーク輻輳の指標として用いていること，およびウィンドウサイズの増加量および減少量を決定するパラメータがネットワーク環境によって調節されず常に一定であるということに起因する．

例えば，無線リンクを経由するようなネットワーク環境において TCP Reno を用いた場合に，そのスループットが低下するという問題点が [10] で指摘されている．この問題は，ネットワーク内でパケット廃棄が発生した際に，それが輻輳による廃棄であるのか，無線リンク上での廃棄であるのかを区別することができないことに起因している．この問題に対する解決法には [12-14] などがある．

また，帯域や遅延が大きいネットワークにおいて TCP Reno を用いた場合においてスループットが低下することが問題点として挙げられる．文献 [11] において，TCP Reno がそのような環境においてリンク帯域を使い切ることができないことが示されている．この問題は，ウィンドウサイズの増加量を決定するパラメータが小さく (1 ラウンドトリップ時間 (RTT) あたり 1 パケット)，ウィンドウサイズの減少量を決定するパラメータが大きい (パケット廃棄発生時に半減させる) ことに起因している．この問題に対する解決法は [11, 15, 16] などにおいて提案されている．それらの多くは TCP Reno のウィンドウサイズ制御の基本的機構である AIMD 方式を引き継いでおり，その増減の量を決定するパラメータをネットワーク環境に応じて静的あるいは動的に調節することでスループットの改善を行っている．しかし，それらの多くは特に帯域や遅延が大きいネットワーク環境を想定した修正であるため，他の環境において適用された場合にも問題点を持たないかどうかは不明であり，本質的な解決を行っているとはいえない．

ウィンドウサイズは 1 RTT の間に送出可能なパケット数であるので，ある TCP コネクションにとって最適なウィンドウサイズは送受信端末間の利用可能帯域と RTT の積で表される．TCP Reno は送受信ホスト間のパスの RTT を計測しているが，利用可能帯域を知るための効率的な方法を持たない．このことが特定の環境下においてウィンドウサイズを適切な値に調節することができない問題の本質的な原因である．上述のように TCP Reno はデータ転送速度を利用可能帯域まで到達させるためにウィンドウサイズを調節する機構を持っているため，ある意味では利用可能帯域を計測しているということができる．しかしながら，その方法はパケット廃棄が発生するまでウィンドウサイズを増加させるという単純なものであるために効率的ではない．このことは，現在のインターネットで主に用いられている TCP Reno を含め，AIMD 方式を用いる TCP の改善手法が，ネットワークの利用可能帯域に関する情報を得るために周期的なパケット廃棄の発生を避けられないという問題の原因でもある．すなわち，TCP が何らかの手法を用いて，送受信ホスト間のパスの帯域に関する情報をすばやく，高い精度で取得することができれば，ウィンドウサイズの制御に AIMD 方式を用いる必要はなく，より効率の良い輻輳制御方式を考えることが可能となる．

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数多く提案されてきた [17-21]．しかし，これらの手法は大量の計測パケットを必要としたり，新たな計測結果を取得するまでに多くの時間を必要とするため，TCP のウィンドウサイズ制御に直接利用することはできない．これらの性質を持たず，TCP のデータ転送と併用可能な計測を行う手法として，我々の研究グループでは Inline Measurement TCP (ImTCP) [22, 23] と呼ぶ手法を提案している．この手法は，送受信端末間パスの物理帯域および利用可能帯域を計測するために TCP コネクションがデータ転送に用いるデータパケットおよび ACK パケットのみを用い，それ以外に計測用パケットを必要としない．また，非常に短い間隔 (1–4 RTT) で計測結果を継続的に取得できるため，ネットワーク状況の変化にすばやく追随できる．また，TCP の輻輳制御アルゴリズムを変更するのではなく，送信側ホストにおいて TCP から IP へパケットが渡される部分に計測機構が組み込まれるため，任意の TCP の輻輳制御方式と組み合わせて用いることが可能である．

そこで本稿では，上記のインライン計測を用いて帯域に関する情報を取得し，その情報を用いてウィンドウサイズ制御を行うことによって，従来の TCP Reno における問題を本質的に改善するための新たな TCP の輻輳制御方式を提案する．ウィンドウサイズ制御のアルゴリズムは，帯域に関する情報を用いることによってウィンドウサイズを適切な値にすばやく調節すること，および他のコネクションが競合する際に公平に帯域を分配できることを目的として設計する．そのために，数理生態学において生物の個体数の変化を表すモデルとして有名な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 [24] を適用する．これらのモデルを TCP のウィンドウサイズ制御へ適用するために，生物の個体数をデータ転送速度に，個体数の収束値である環境容量を物理帯域に，および種間の競争を同一リンク上の複数コネクションの競合にそれぞれ変換する．

本稿では，提案方式の特性を数学的解析によって明らかにし，提案方式が持つパラメータ設定方法に関する議論を行う．また，シミュレーションを用いた性能評価を行い，TCP Reno およびその改善手法と比較して高い性能をを持つことを明らかにする．

以下，2 章では上記で述べた帯域計測およびウィンドウサイズ制御を用いた提案方式について説明する．3 章では提案方式の特性，および提案方式が持つパラメータの設定方法について数学的解析によって議論する．4 章では，シミュレーションによって提案方式の性能を評価する．最後に，5 章で本稿のまとめと今後の課題を示す．

2. 提 案 方 式

提案方式は，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．

2.1 ウィンドウサイズ制御アルゴリズム

ロジスティック増殖モデルは，ある環境において 1 種の生物の個体数変化を表すモデルであり，次の式で表される [24].

$$\frac{d}{dt}N = \epsilon\left(1 - \frac{N}{K}\right)N \tag{1}$$

ここで，N は生物の個体数を，K は環境容量を，ϵ は種の内的自然増殖率を表す ($0 < \epsilon$).

次に示すロトカ・ヴォルテラ競争モデルは，ある環境において互いに競争関係にある 2 種の生物の個体数変化を表すモデルであり，式 (1) を拡張した以下の式で表される [24].

$$\frac{d}{dt}N_1 = \epsilon_1\left(1 - \frac{N_1 + \gamma_{12}N_2}{K_1}\right)N_1 \tag{2}$$

$$\frac{d}{dt}N_2 = \epsilon_2\left(1 - \frac{N_2 + \gamma_{21}N_1}{K_2}\right)N_2 \tag{3}$$

ここで，N_i，K_i，ϵ_i はそれぞれ種 i の個体数，環境容量，内的自然増殖率を表す ($0 < \epsilon_i$)．γ_{ij} は，競争相手種 j の存在による種 i の増殖率低下の度合いを表すパラメータである．ここで，両方の種が同一の環境下に存在し，さらに同じ特性を持つと仮定する．すなわち，$K = K_1 = K_2$，$\epsilon = \epsilon_i = \epsilon_j$，$\gamma = \gamma_{12} = \gamma_{21}$ とする．このとき，両種が競争関係を持ちながらも片方の種が絶滅せず，共生を実現するための条件が存在し，それは $0 < \gamma < 1$ であることが知られている [24]．さらに，式 (2)，(3) を n 種の生物の個体数変化を表すように拡張すると，

$$\frac{d}{dt}N_i = \epsilon\left(1 - \frac{N_i + \gamma\sum_{j=1,\ i\neq j}^{n} N_j}{K}\right)N_i \qquad (4)$$

となる．ここで，生物の個体数を表したモデルを TCP の輻輳制御に適用するため，N_i を TCP コネクション i のデータ転送速度 [packets/sec]，K を物理帯域 [packets/sec] とみなす．このとき，式 (4) 中の $\sum_{j=1,\ i\neq j}^{n} N_j$ は他のコネクションのデータ転送速度の合計値に相当するが，TCP コネクションはこの情報を直接知ることはできない．そのため，提案方式においては物理帯域 K と利用可能帯域 A_i [packets/sec] を用いて $\sum_{j=1,\ i\neq j}^{n} N_j \sim (K - A_i)$ と近似する．これにより，式 (4) は次のように書き換えられる．

$$\frac{d}{dt}N_i = \epsilon\left(1 - \frac{N_i + \gamma(K - A_i)}{K}\right)N_i \qquad (5)$$

TCP の輻輳制御は，ウィンドウサイズを制御することによって行われるため，データ転送速度で表された式 (5) をウィンドウサイズで表された式へ書き換える必要がある．ここで，ウィンドウサイズ w_i は，データ転送速度 N_i および RTT の最小値 τ_i を用いて $w_i = N_i\tau_i$ とする．これにより，式 (5) は次のように書き換えられる．

$$\frac{d}{dt}w_i = \epsilon\left(1 - \frac{w_i + \gamma(K - A_i)\tau_i}{K\tau_i}\right)w_i \qquad (6)$$

最後に，ウィンドウサイズを直接求める式とするため，微分方程式で表された式 (6) を積分して次の式で表す．

$$w_i(t) = \frac{w_i(0)\tau_i f_i(t)\left\{K - \gamma\left(K - A_i\right)\right\}}{w_i(0)(f_i(t) - 1) + \tau_i\left\{K - \gamma\left(K - A_i\right)\right\}} \qquad (7)$$

ただし，$f_i(t) = e^{\epsilon t\left\{1-\gamma\left(1-\frac{A_i}{K}\right)\right\}}$ (8)

式 (7) を用いて，ある時刻を基準 ($t = 0$) とし，その時のウィンドウサイズを $w_i(0)$ に代入することで，任意の時刻 t におけるウィンドウサイズ $w_i(t)$ を得ることができる．なお，式 (8) には e^x の計算が含まれているが，一般的に TCP が実装される OS のカーネルにおいては指数計算などの複雑な演算を行うことは大きな負荷となる．そこで，本稿における提案方式においては，式 (8) をテイラー展開によって簡易な式に近似する．その一例として，e^x を $x = a$ の周りで第 4 次の項までテイラー展開したもので近似すると，次の式で表すことができる．

$$e^x \sim e^a \sum_{k=0}^{4} \frac{1}{k!}(x - a)^k$$

ここで a は x の整数部分 (例えば，$0 \leqq x < 1$ においては $a = 0$) とし，計算のために必要となる e^a などは事前に計算結果をテーブルに保持する．以上のような近似計算を用いることで，式 (8) を小さい負荷で計算することができると考えられる．

2.2 輻輳制御アルゴリズム

提案方式の輻輳制御アルゴリズムは，従来の TCP Reno に ImTCP のインライン計測機能を組み込んだものを元に拡張したものである．提案方式は，計測によって帯域の情報が得られるまでは TCP Reno と同じウィンドウサイズ制御を行う．また，パケット廃棄が発生した場合には TCP Reno と同様にウィンドウサイズを半減させる．タイムアウトが発生した場合にも TCP Reno と同様にウィンドウサイズを 1 [packet] に設定し，それまでに得られた全ての計測結果を破棄する．

インライン計測によって帯域の情報を取得できれば，前述したウィンドウサイズ制御アルゴリズムを用いて，TCP Reno と同様に ACK を受信するたびに以下のようにウィンドウサイズの調節を行う．j 個目の ACK を受信した時刻を t_j とする．j 個目の ACK を受信したときのウィンドウサイズは，式 (7) において，時刻 0 を $j-1$ 個目の ACK を受信した時刻 (t_{j-1}) とし，$t = t_j - t_{j-1}$ を代入することによって求める．

3. 提案手法の特性

3.1 スケーラビリティおよび収束速度

物理帯域 K および利用可能帯域 A が一定であると仮定すると，提案手法を用いることによってウィンドウサイズはある値に収束する．その値 w^* は式 (6) を用いて，$dw/dt = 0$ と考えることで以下のように得ることができる．

$$w^* = \{(1 - \gamma)K + \gamma A\}\tau \qquad (9)$$

このとき，ウィンドウサイズが任意の値 w_0 から ρw^* ($0 < \rho < 1$，$w_0 < \rho w^*$) へ増加するのにかかる時間 T_{proposed} は，式 (7) より，

$$T_{\mathrm{proposed}} = \frac{1}{\epsilon\left\{1 - \gamma\left(1 - \frac{A}{K}\right)\right\}} log\left(\frac{\rho}{1-\rho}\frac{w^* - w_0}{w_0}\right) < \frac{1}{\epsilon(1-\gamma)} log\left(\frac{\rho}{1-\rho}\frac{w^* - w_0}{w_0}\right) \qquad (10)$$

となる ($0 \leqq A \leqq K$ より)．一方，TCP Reno においてウィンドウサイズが w_0 から w^* へ増加するのにかかる時間 T_{reno} は，

$$T_{\mathrm{reno}} = (w^* - w_0)\bar{\tau} = [\{(1 - \gamma)K + \gamma A\}\tau - w_0]\bar{\tau} \qquad (11)$$

となる．ここで，$\bar{\tau}$ は平均のラウンドトリップ時間とする．また，Delayed ACK は無効であるとし，輻輳回避フェーズでウィンドウサイズが増加した場合の結果を示している．また，HighSpeed TCP (HSTCP) [11] を用いる場合には，

$$T_{\mathrm{hstcp}} \geqq \frac{w^* - w_0}{a_{\max}}\bar{\tau} = \frac{\{(1 - \gamma)K + \gamma A\}\tau - w_0}{a_{\max}}\bar{\tau} \qquad (12)$$

となる．ここで $a_{\max}$ は HSTCP が持つパラメータで，1 つの ACK パケットを受信したときのウィンドウサイズの増加量の最大値を表す (文献 [11] における $a(W)$ に相当)．

式 (11)，(12) から，TCP Reno および HSTCP においては，ウィンドウサイズの増加にかかる時間が，物理帯域 K および遅延時間 τ の増加に対して線形的に大きくなることがわかる．すなわち，帯域遅延積が大きくなるにつれてその帯域遅延積を使い切るまでの時間が線形的に大きくなることを示している．HSTCP は TCP Reno の高速ネットワークにおける低いリンク利用率を改善するための手法として文献 [11] において提案されているが，そのアルゴリズムは基本的に AIMD 方式であるため，帯域遅延積に対するスケーラビリティは本質的に低いといえる．一方，提案方式においては，ウィンドウサイズの増加にかかる時間が，帯域遅延積の増加に対して対数的に大きくなる．このことから，提案方式は TCP Reno や HSTCP よりも帯域遅延積の増加に対して高いスケーラビリティを持つことがわかる．

Scalable TCP (STCP) [15] は Multiplicative Increase Multiplicative Decrease (MIMD) のアルゴリズムを持ち，物理帯域の増加に対して高いスケーラビリティを持つことが知られている [15] が，アルゴリズムが理想的に動作したとしても，TCP Reno や HSTCP と同様に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，提案方式はウィンドウサイズが理想的な値に収束するため，パケット廃棄が発生しない．

3.2 パラメータ設定

提案手法は 2 つの制御パラメータ (ϵ および γ) を持つ．γ はボトルネックリンクを共有する他のコネクションの影響を考慮する度合いを表し，各コネクションの物理帯域 (K_i) の値にかかわらずウィンドウサイズが正の値に収束するためには $0 < \gamma < 1$ である必要があることがわかる．式 (9)，(10) より，γ の値を設定する際には，収束速度と収束時にボトルネックリンクに蓄積されるパケット量に関するトレードオフを考慮する必要があることがわかる．すなわち，γ を小さくすると収束速度が速くなる反面，収束時にボトルネックリンクに蓄積されるパケット量が大きくなる．したがって，ボトルネックリンクの

バッファサイズを考慮して γ を設定する必要があると考えられる．

一方 ϵ は，式 (6) に示されるように，収束速度を決定するパラメータである．一般的に，式 (1) を離散化した場合，ϵ が 2 より大きくなると個体数が振動し続けて収束しないことが知られている [24]．これに対し本手法においては，式 (6) を積分した式 (7) を用い，振動が発生しないような離散化を行っているため，この点に関しては ϵ の値に制限は発生しない．すなわち，ϵ を大きくすればするほど収束速度が速くなる．しかし，特に広帯域・高遅延ネットワークにおいては，ϵ を大きくすると 1 つの ACK パケットを受信した際のウィンドウサイズの増加量が大きくなり，多くのパケットをバースト的に送出しネットワークに影響を与えることが考えられる．

3.3 時間遅れを考慮したモデル

提案方式の基となっているロジスティック増殖モデル (式 (1)) においては，現在の個体数に応じて増殖速度が決定される．提案方式は，帯域情報の計測結果に基づいてウィンドウサイズの増加量を決定する．しかし，提案方式が用いる計測手法は，ネットワークへ送出したデータパケットに対する ACK パケットの到着時間を用いて物理帯域および利用可能帯域を導出するため，取得できる帯域情報には時間的な遅れが存在する．したがって，式 (1) を利用するのではなく，時間遅れ τ_d を考慮したモデルが必要であると考えられる．一般的に，式 (1) を以下のように変更することにより，個体数の情報の時間遅れを考慮することができる [25]．

$$\frac{d}{dt}N(t) = \epsilon\left(1 - \frac{N(t-\tau_d)}{K}\right)N(t) \tag{13}$$

式 (13) に基づいて個体数が変動する場合，以下の条件を満たすならば連続時間モデルにおいても個体数が振動し収束しないことが知られている．

$$\tau_d > \frac{\pi}{2\epsilon}$$

すなわち，時間遅れが存在する場合において個体数が収束するためには，$\epsilon \leqq \pi/2\tau_d$ を満たす必要がある．提案方式のアルゴリズム (式 (6)) におけるパラメータ ϵ に関しても，同様の制限が発生すると考えられる．このことは，得られる帯域に関する情報に時間遅れが存在する場合には，急激にウィンドウサイズを変動させることによって，振動が発生することを意味する．

提案方式において発生する時間遅れは，計測を行うボトルネックリンクをデータパケットが通過してから，対応する ACK パケットが到着するまでの時間に相当する．ImTCP は帯域情報の計測に 1–4 RTT かかることを考慮すると，時間遅れ τ_d は 1–2 RTT 程度になる．すなわち，時間遅れの大きさがコネクションのラウンドトリップ時間によって変動するため，各 TCP コネクションのラウンドトリップ時間に応じて ϵ を設定することによって，振動を発生させることなく収束速度を向上させることができると考えられる．しかし，ϵ をコネクションによって変化させることによって，それらのコネクションが同時にネットワーク内に存在する場合に，最終的な収束値は変化しないものの，収束速度が違うために短期的なコネクション間の不公平性が発生することが考えられる．これらに関しては今後の課題としたい．

4. シミュレーション結果

本章では，提案方式の性能を ns-2 [26] を用いたシミュレーションによって評価する．比較対象として，TCP Reno, HighSpeed TCP (HSTCP) [11], Scalable TCP (STCP) [15], FAST TCP [16] を用いる．提案方式は本来，物理帯域および利用可能帯域の情報をインラインネットワーク計測手法 (ImTCP) を用いて取得するが，このシミュレーションにおいてはコネクション数が 1 本の場合のみ，開発途中のバージョンである ImTCP を用いて利用可能帯域の計測を行う．物理帯域およびコネクション数が 2 本以上の場合における利用可能帯域は，正確な値を直接与えるものとする．提案方式で使用されるパラメータ ϵ, γ はそれぞれ 1.95，0.9 とする．HSTCP および STCP のパラメータはそれぞれ [11] および [15] において示されているものとし，SACK オプション [27] を有効にする．FAST TCP のパラメータ α は，リンク帯域の大きさに応じて変化させる必要がある [16]．ここでは [28] に示されている帯域と α の比を参考に，リンク帯域が 100 [Mbps] の場合は $\alpha = 100$，1 [Gbps] の場合は $\alpha = 1000$ としている．

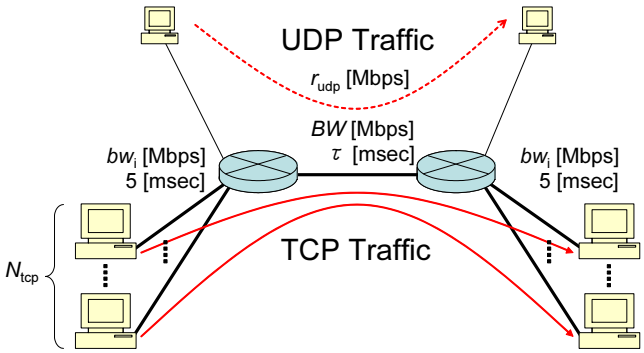


図 1　シミュレーションモデル

シミュレーションに用いるネットワークモデルは図 1 に示すとおりであり，N_{tcp} 本の TCP 送受信ホスト，バックグラウンドトラヒックを発生させるための UDP の送受信ホスト，2 つのルータ，それらを接続するためのリンクで構成されている．バックグラウンドトラヒックとして，実際のインターネットを流れるトラヒックの構成を近似したもの [29] を，UDP を用いてデータ転送速度 r_{udp} で発生させる．共有リンク，コネクション i のアクセスリンクの帯域をそれぞれ BW，bw_i で表し，共有リンクの伝播遅延時間を τ で表す．パケットサイズは 1500 [Bytes] とする．ルータバッファには TailDrop 方式を用い，そのサイズは TCP コネクションの帯域遅延積に等しいとする．

まず最初に，ウィンドウサイズの変化を観測することで各方式の基本的なふるまいを確認する．ここでは，$N_{\mathrm{tcp}} = 1$, $BW = 100$ [Mbps], $bw_1 = 200$ [Mbps], $\tau = 25$ [msec] とする．バックグラウンドトラヒックは発生させない．図 2 は，提案方式と比較対象の TCP のウィンドウサイズの変化を表している．TCP Reno，HSTCP および STCP は輻輳発生のシグナルとしてパケット廃棄の発生を用いるため，パケット廃棄が繰り返し発生し，ウィンドウサイズは常に変化して安定状態へ収束しない．一方，キューイング遅延を輻輳発生のシグナルとして用いる FAST TCP，および帯域情報を計測によって取得する提案方式は，周期的なパケット廃棄を回避し，ウィンドウサイズが一定の値に収束している．次に，共有リンク帯域，アクセスリンク帯域をともに 10 倍の 1 [Gbps] にした場合の結果を図 3 に示す．この図から，リンク帯域が大きい場合には，TCP Reno のウィンドウサイズの増加速度が非常に遅いことがわかる．また，HSTCP においてはウィンドウサイズの増加にかかる時間がやや増加していること，その他の方式はリンク帯域に関係なく短い時間でウィンドウサイズが増加していることがわかる．また，HSTCP および STCP はウィンドウサイズの増加速度が大きいため，TCP Reno に比べて多くのパケット廃棄が発生する．それに対して，図 2 においては SACK が有効に機能するためタイムアウトの発生を回避することができているが，図 3 においてはパケット廃棄の量が多くなり，SACK がパケット廃棄情報を保持するためのブロックの容量を超えてしまうため，タイムアウトが発生している．

次に，ネットワークの帯域遅延積を使い切るまでにかかる収束時間を検証することで，各方式の帯域遅延積に対するスケーラビリティを評価する．ここでは，収束時間は TCP コネクションがリンク帯域の 99%を利用するまでに要する時間とする．ネットワーク環境は N_{tcp}=1，bw_1=200 [Mbps]，BW=100 [Mbps]，r_{udp}=20 [Mbps] とする．図 4 は，共有リンクの遅延 τ を 10 [msec] から 500 [msec] まで変化させた際の収束時間の変化を表しており，10 回のシミュレーションの実行によって得た平均値，および 95%の信頼区間を示してい

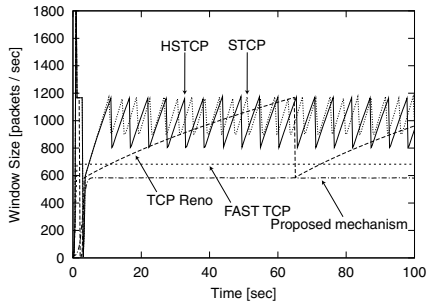


図 2　各方式のウィンドウサイズの変化 (BW=100 [Mbps])

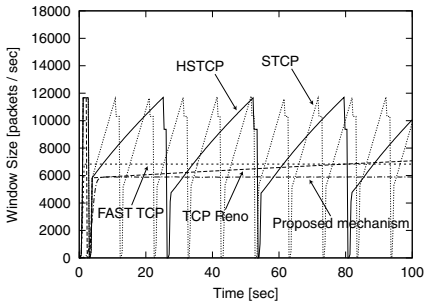


図 3　各方式のウィンドウサイズの変化 (BW=1 [Gbps])

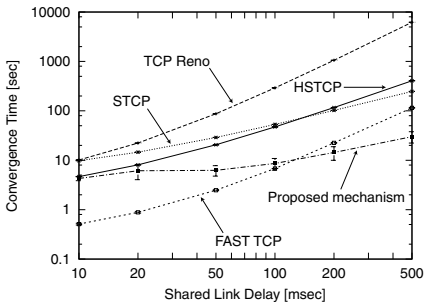


図 4　共有リンク遅延に対する収束時間

る．この図から，TCP Reno の収束時間が最も大きく，遅延時間が大きくなると収束時間が急激に大きくなっていることがわかる．HSTCP の収束時間は TCP Reno と比較すると小さいが，遅延時間が大きくなると収束時間が大きくなっている．これは，3.1 節で示したように，HSTCP は TCP Reno の問題点を改善するアルゴリズムを持つが，帯域遅延積に対するスケーラビリティを本質的には改善していないことが原因である．また，FAST TCP は遅延時間が小さい場合には他方式に比べて収束時間が短いものの，遅延時間が大きくなると収束時間は急激に大きくなる．このことから，FAST TCP の帯域遅延積に対するスケーラビリティは低いといえる．STCP は 3.1 節で述べたように帯域に対しては高いスケーラビリティを持つが，遅延時間が増加すると収束時間が大きく増加している．これは，STCP は MIMD に基づくアルゴリズムを持つが，従来の TCP と同様，ACK パケットの受信数に応じてウィンドウサイズを増加させるため，RTT の増加にともないウィンドウサイズの増加速度が緩やかになることが原因である．

一方，提案方式は遅延時間の大きさにかかわらず短い収束時間を持ち，遅延時間の増加に対する収束時間の増加はきわめて緩やかである．これは，3.1 節で示したように，提案手法は帯域遅延積の増加に対してその収束時間が対数的に大きくなるためである．また，遅延時間が大きくなった場合に，収束時間が若干増大しているのは，ImTCP により利用可能帯域の計測に時間がかかっているためであると考えられる．

最後に，提案方式の環境の変化に対する追随性およびコネクション間の公平性を評価する．ここでは，$N_{\rm tcp}$=5, bw_i=200 [Mbps], BW = 100 [Mbps], τ=25 [msec] とし，バックグラウンドトラヒックは発生させないものとする．5 本の TCP コネクションは，シミュレーション開始からそれぞれ 0，100，300，500，700 [sec] 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 [sec] 後にデータ転送を終了して離脱する．図 5 は，TCP Reno，HSTCP，STCP，FAST TCP，および提案方式における，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 5(a) および図 5(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 5(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，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 5(d) から，FAST TCP について次のような性質が見られる．FAST TCP は，輻輳発生のシグナルとしてパケット廃棄を用いる上記の 3 方式とは異なり，そのシグナルとしてキューイング遅延を用いている．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パケット廃棄をともなわずにウィンドウサイズを調節することが可能である．しかし，既存のコネクションと新たなコネクションとの間で公平性を必ずしも実現できない．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ているために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り，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 5(e) で示されている．

5. おわりに

本稿では，インラインネットワーク計測手法を用いることでネットワークの帯域に関する情報を取得し，その情報を利用したウィンドウサイズ制御アルゴリズムによって輻輳制御を行う TCP の輻輳制御方式を提案した．解析およびシミュレーションによる性能評価を通して，提案方式がリンクの帯域遅延積に対するスケーラビリティを持ち，収束速度，公平性，安定性において優れていることを示した．

提案方式にとって，帯域情報を得るために用いるインライン計測手法の性能は非常に重要な要素である．現在の ImTCP は，まだ利用できる環境が限定されており，本稿におけるシミュレーションにおいては，一部の場合においてのみ利用した．今後の課題としては，帯域に関する情報を全てインライン計測手法を用いて取得する場合における提案方式の性能評価が挙げられる．また，提案方式を用いる TCP コネクションと従来の TCP Reno コネクションがネットワーク内に共存する場合の公平性について評価することが挙げられる．

謝辞　本研究の一部は、平成 16 年度文部科学省科学研究費補助金若手 (A)(16680003) によっている。ここに記して謝意を示す。

文　献

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

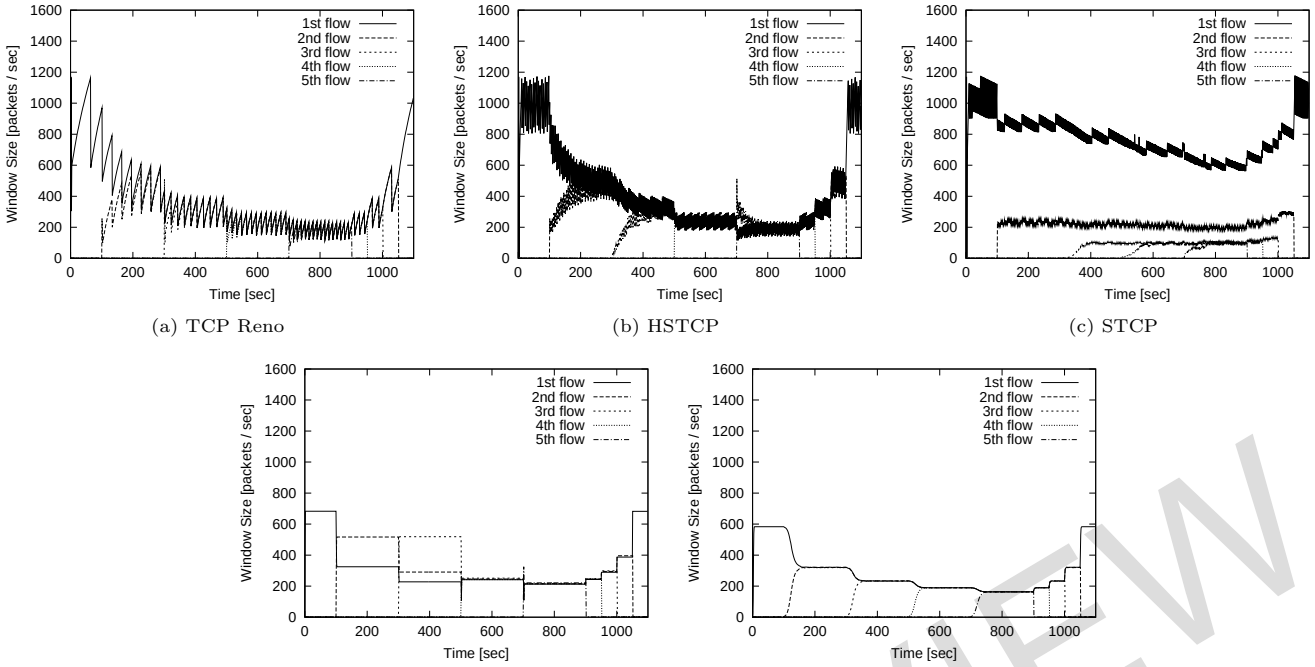
[2] J. B. Postel, “Transmission control protocol,” *Request for Comments 793*, Sept. 1981.

[3] V. Jacobson, R. Braden, and D. Borman, “TCP extensions for high performance,” *Request for Comments 1323*, May 1992.

[4] M. Allman, H. Balakrishnan, and S. Floyd, “Enhancing TCP’s loss recovery using limited transmit,” *Request for Comments 3042*, Jan. 2001.

[5] E. Blanton, M. Allman, K. Fall, and L. Wang, “A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP,” *Request for Comments 3517*, Apr. 2003.

[6] 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*,



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

(d) FAST TCP (e) 提案方式

図 5 コネクション数の変化の影響

pp. 1–14, June 1989.

[7] S. Shenker, L. Zhang, and D. D. Clark, "Some observations on the dynamics of a congestion control algorithm," *ACM Computer Communication Review*, vol. 20, pp. 30–39, Oct. 1990.

[8] J. C. Hoe, "Improving the start-up behavior of a congestion control scheme of TCP," *ACM SIGCOMM Computer Communication Review*, vol. 26, pp. 270–280, Oct. 1996.

[9] L. Guo and I. Matta, "The War Between Mice and Elephants," *Technical Report BU-CS-2001-005*, May 2001.

[10] Z. Fu, P. Zerfos, H. Luo, S. Lu, L. Zhang, and M. Gerla, "The impact of multihop wireless channel on TCP throughput and loss," in *Proceedings of IEEE INFOCOM 2003*, Apr. 2003.

[11] S. Floyd, "HighSpeed TCP for large congestion windows," *RFC 3649*, Dec. 2003.

[12] C. Casetti, M. Gerla, S. Mascolo, M.Y.Sanadidi, and R. Wang, "Tcp westwood: End-to-end congestion control for wired/wireless networks," *In Wireless Networks Journal 8*, pp. 467–479, 2002.

[13] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 747–756, May 2004.

[14] E. H.-K. Wu and M.-Z. Chen, "JTCP: Jitter-based TCP for heterogeneous wireless networks," *IEEE Journal on Selected Areas in Communications*, vol. 22, pp. 757–766, May 2004.

[15] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," in *proceedings of PFLDnet '03: workshop for the purposes of discussion*, Feb. 2003.

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

[17] B. Melander, M. Bjorkman, and P. Gunningberg, "A new end-to-end probing and analysis method for estimating bandwidth bottlenecks," in *Proceedings of IEEE GLOBECOM 2000*, Nov. 2000.

[18] 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.

[19] V. Ribeiro, R. Riedi, R. Baraniuk, J. Navratil, and L. Cottrell, "pathChirp: Efficient available bandwidth estimation for network paths," in *Proceedings of NLANR PAM2003*, Apr. 2003.

[20] R. L. Carter and M. E. Crovella, "Measuring bottleneck link speed in packet-switched networks," Tech. Rep. BU-CS-96-006, Boston University Computer Science Department, Mar. 1996.

[21] C. Dovrolis, P. Ramanathan, and D. Moore, "What do packet dispersion techniques measure?," in *Proceedings of IEEE INFOCOM 2001*, Apr. 2001.

[22] M. L. T. Cao, "A study on inline network measurement mechanism for service overlay networks," Master's thesis, Graduate School of Information Science, Osaka University, Feb. 2004.

[23] M. L. T. Cao, G. Hasegawa, and M. Murata, "Available bandwidth measurement via TCP connection," in *Proceedings of IFIP/IEEE MMNS 2004*, Oct. 2004.

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

[25] 巌佐 庸, **数理生物学入門–生物社会のダイナミックスを探る**. 共立出版, 1998.

[26] The VINT Project, "UCB/LBNL/VINT network simulator - ns (version 2)." available from `http://www.isi.edu/nsnam/ns/`.

[27] M. Mathis, "TCP selective acknowledgement options," *Request for Comments 2018*, Oct. 1996.

[28] C. Jin, D. X. Wei, and S. H. Low, "Internet draft: FAST TCP for high-speed long-distance networks," *Internet draft draft-jwl-tcp-fast-01.txt*, June 2003.

[29] A. Technologies, "Mixed packet size throughput." available from `http://advanced.comms.agilent.com/n2x/docs/journal/JTC_003.html`.