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.

インライン計測に基づく高速TCP輻輳制御方式の提案

井口　智仁† 　長谷川　剛†† 　村田　正幸†

† 大阪大学情報科学研究科　〒 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，データ転送のトラヒック以外に計測用のトラヒックを必要としないインラインネットワーク計測手法を用いた，新しい TCP の輻輳制御方式を提案する．具体的には，送信ホストと受信ホスト間のパスの利用可能帯域をインライン計測によって取得し，数理生態学において生物の個体数の変化を表すロジスティック増殖モデルおよびロトカ・ヴォルテラモデルに基づいてウィンドウサイズを増減させる．これは従来の 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して，提案方式がネットワーク帯域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 TCP Reno よりも優れた性質を持つことを示す．

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

A New Congestion Control Mechanism of TCP with Inline Network 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 novel congestion control mechanism of TCP, by using an inline network measurement technique. By using information of available bandwidth of a network path between sender and receiver hosts, we construct quite a different congestion control mechanism from the traditional TCP Reno and its variants, based on logistic and Lotka-Volterra models 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, convergence time, fairness among connections, and stability.

Key words　congestion control, inline measurement, available bandwidth, logistic model, Lotka-Volterra model

1. はじめに

Transmission Control Protocol (TCP) は，インターネットのトランスポート層で最も広く使われているプロトコルである．TCP は 1970 年代に設計され，それを規定する Request for Comments (RFC) のうち最初のものは 1981 年に発行され [1]，その後のネットワークの発展に伴い頻繁に改良が行われている [2-4].

TCP が持つ機能の中で最も重要なものは輻輳制御機構 [5] である．それ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 コネクションは自身のウィンドウサイズを増減させることによってデータ転送速度を調節する．具体的には，ネットワーク内でパケット廃棄が発生しない限りウィンドウサイズを線形的に増加させ続け，パケット廃棄が発生するとウィンドウサイズを半減させる，というアルゴリズムを用いている．この方式は Additive Increase Multiplicative Decrease (AIMD) と呼ばれ，複数の TCP コネクションが独立に動作しても，それらの間で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，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，現在広く用いられている TCP Reno と呼ばれる方式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 [7-9]．それらの問題の多くは，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固定されていることに起因する．これらのパラメータは，ネットワーク環境によって変更されるべきものである．例えば，TCP コネクションが無線リンクを経由している場合に，そのスループットが低下する問題があることが [10-12] で指摘されている．これ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なのか，あるいは無線リンク上での廃棄なのかを区別できないために発生する．この場合には，特にウィンドウサイズの減少量を決定するパラメータを変更することで性能が改善することが知られている．

また，帯域や遅延が大きいネットワークにおいて，TCP の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．[13] に

おいて，TCP Reno はそのような環境で帯域を使い切ることができないことが示されている．なぜなら，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) あたりの増加量は 1 である)，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) ためである．

この問題点を解決する方法は数多く提案されている [11-16]．しかし，それらの多くは TCP Reno の輻輳制御方式の基本的機構，すなわち AIMD 方式によるウィンドウサイズの調節アルゴリズムを引き継いでおり，そのパラメータをネットワーク環境に応じて静的・動的に変化させることでスループットの改善を行っている．しかし，それらは特定のネットワーク環境を想定した修正であることが多く，他の環境において適用された場合にスループットを改善できるかどうかは不明であり，本質的な解決法とはいえない．

これらの問題点の本質的な原因は，TCP が送受信ホスト間のパスの利用可能帯域を知るための効率的な方法を持たないことである．上述のように TCP 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動的に調節する機構を持つため，ある意味で利用可能帯域を計測しているといえる．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，効率的ではない．すなわち，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 の改善手法は，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を避けられないという本質的な問題を持つといえる．

TCP が何らかの方法を用いて，送受信ホスト間のパスの利用可能帯域を素早く，高い精度で取得できるならば，輻輳制御に AIMD 方式を用いる必要はなく，より効率の良い方式を考えることが可能となる．ネットワークパスの利用可能帯域を計測するための方法はこれまでにも多数提案されている [17-19]．しかし，これらの方法を TCP の輻輳制御に直接適用することはできない．なぜなら，それらは大量の計測パケットを必要とし，計測結果を取得するまでに長い時間を必要とするためである．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) と呼ばれる方式を提案している [20, 21]．この方式は，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と 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素早く追随できる．

本稿では，上記のインライン計測によって取得する利用可能帯域の情報を用いた，全く新しい輻輳制御方式を提案する．この提案方式は，TCP Vegas [22] のように経験的な知見に基づくアルゴリズムを用いるのではなく，数学的な根拠に基づくアルゴリズムを用いる．これにより，アルゴリズムが引き起こす振る舞いを数学的に議論し，その性能を保証することが可能となる．さらに，パラメータを理論的な根拠に基づいて決定することができ，シミュレーション結果を通して経験的に決定する場合に生じる問題を回避することが可能となる．提案方式のアルゴリズムには，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 [23] を用いる．これらのモデルを TCP の輻輳制御へ応用するために，生物の個体数をデータ転送速度に，個体数の上限値である環境容量をボトルネックリンク帯域に，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とそれぞれ見なす．本稿では，上記の適用の詳細について述べた後に，提案方式を解析的に評価し，数理生態学において知られている性質の議論や結果を用いて，提案方式がネットワーク帯域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などに関して優れた特性を持つことを確認する．また，シミュレーションによる性能評価を通して，提案方式が従来の TCP Reno やその改善手法に比べて，ネットワークの帯域を効率良く，素早く，公平に利用することができることを確認する．

以下，2 章では数理生態学におけ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 TCP の輻輳制御方式に適用する方法を示す．3 章ではインラインネットワーク計測手法を組み込んだ新しい TCP の輻輳制御方式を提案し，その特性について議論する．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. 数理生態学におけるモデルと TCP の輻輳制御への応用

本章では，数理生態学において生物の個体数の変化を表す，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について簡潔に説明する．これらは，提案方式のウィンドウサイズ調節アルゴリズムの基本となるものである．

2.1 ロジスティック増殖モデル

ロジスティック増殖モデルは，ある限られた領域の中で生息している 1 つの種に属する個体の数の変化を表している．一般的に，個体数が多くなるにつれてその増加速度は大きくなる．しかし，自然界では環境や資源について様々な制約があるために，個体数の上限となる環境容量が存在する．これらの影響を考慮し，時間の経過にともなう生物の個体数の増殖過程を端的に表したものが，以下のロジスティック方程式である．

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

ここで，t は時刻を，N は個体数を，K は環境容量を，ϵ は種の内的自然増殖率を表す．図 1 は，$K = 100$ とし，ϵ を 0.6, 1.8, 2.4, 3.0 と変化させた場合における時刻に対する個体数 (N) の変化を示している．$\epsilon = 0.6, 1.8$ の場合における曲線から，ロジスティック方程式が次のような特徴を持つことが分かる．すなわち，N が K よりも小さい場合には，N が大きくなるに従って N の増加速度は大きくなる．一方，N が K に近づくと，N の増加速度は減少し，K へ収束する．ϵ を 0.6 から 1.8 に増加させると，収束に要する時間は少なくなるが，一時的に N が K を越えるオーバーシュートと呼ばれる現象が発生する．一方，ϵ を 2.4 あるいは 3.0 とすると，N は K へ収束せず不安定な状態となる．これはロジスティック方程式に関する特性として知られており，N を K へ収束させるためには ϵ を 2.0 以下にする必要があることが分かっている．

ロジスティック方程式における N の増加特性は，TCP のデータ転送速度の制御に適用できると考えられる．すなわち，N をデータ転送速度とし，K を物理的なボトルネックリンク帯域 (TCP コネクションのパス上のリンク帯域で最も小さいもの) とすることで，素早くかつ安定してリンク帯域を利用することが可能となる．しかし，実際のネットワークでは，2 本以上の TCP コネクションが存在するため，1 種の個体数の変化を表すロジスティック方程式をそのまま用いることはできない．次節では，2 種の個体数の変化を示したモデルを説明する．

2.2 ロトカ・ヴォルテラ競争モデル

ロトカ・ヴォルテラ競争モデルは，2 種間の相互作用を考慮した，生物の個体数の変化を表すものとして知られている．具体的には，ロジスティック方程式に，種内の競争による影響だけでなく種間の競争による影響も含めたものに相当する．2 種間の競争関係を表したロトカ・ヴォルテラ競争モデルにおいて，種の個体数はそれぞれ以下の式で表される．

$$\frac{dN_1}{dt} = \epsilon_1 \left(1 - \frac{N_1 + \gamma_{12} \cdot N_2}{K_1}\right) N_1 \quad (1)$$

$$\frac{dN_2}{dt} = \epsilon_2 \left(1 - \frac{N_2 + \gamma_{21} \cdot N_1}{K_2}\right) N_2 \quad (2)$$

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．γ_{ij} は，競争相手種 j の存在による種 i の増殖率低下を表す．

このモデルでは，両方の種の個体数が必ずしも正の値に収束するとは限らない．すなわち，場合によってはどちらかが絶滅する可能性がある．このことは，γ_{12}，γ_{21} の値に依存しており，以下の関係を満たす場合に両方の種が生存状態を保つことが知られている．

$$\gamma_{12} < \frac{K_1}{K_2},\ \gamma_{21} < \frac{K_2}{K_1} \quad (3)$$

両方の種が同じ特性を持つ，すなわち同じ K, ϵ, γ を持つと仮定すると，式 (1)(2) をそれぞれ以下のように書き換えることができる．

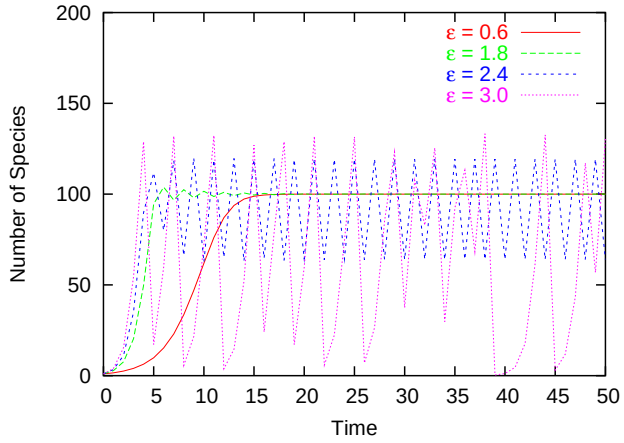


図 1　ロジスティック方程式 ($K = 100$)

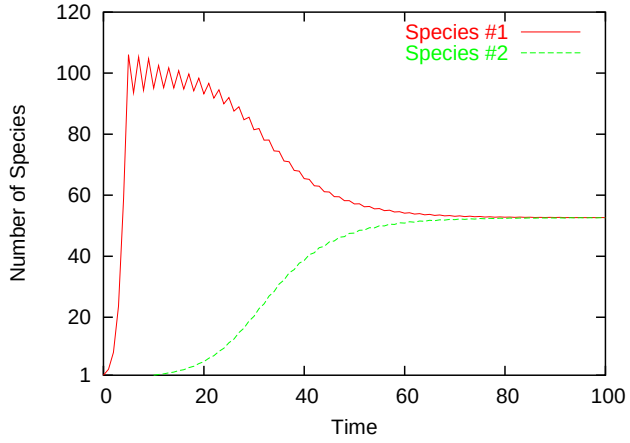


図 2　ロトカ・ヴォルテラ競争モデルにおける個体数変化 ($\epsilon = 1.95$, $\gamma = 0.9$, $K = 100$)

$$\frac{dN_1}{dt} = \epsilon \left(1 - \frac{N_1 + \gamma \cdot N_2}{K}\right) N_1 \tag{4}$$

$$\frac{dN_2}{dt} = \epsilon \left(1 - \frac{N_2 + \gamma \cdot N_1}{K}\right) N_2 \tag{5}$$

このとき，式 (3) の条件は $\gamma < 1$ となる．図 2 は，式 (4) (5) を用いた場合に，時刻に対するそれぞれの種の個体数の変化を表している．ここでは，片方の種が 10 単位時間だけ遅れて系内に出現するとしている．この図から，両方の種の個体数が素早く同じ値に収束することが分かる．我々は，この動作が TCP のデータ転送速度の制御として理想的であると考える．

2.3　データ転送速度の制御アルゴリズムへの適用

式 (4) (5) を n 種間の場合に拡張すると，以下のようになる．

$$\frac{dN_i}{dt} = \epsilon \left(1 - \frac{N_i + \gamma \cdot \sum_{j=1, i \neq j}^{n} N_j}{K}\right) N_i \tag{6}$$

式 (6) を TCP コネクション i のデータ転送速度 (N_i) の制御アルゴリズムとして考えた場合，コネクション i は同じボトルネックリンクを共有している他の全てのコネクションの転送速度を知る必要があることがこの式から分かる．この仮定は実際のインターネットでは現実的ではないが，インライン計測手法 [20] によってコネクション i が自身の利用可能帯域および物理帯域を知ることができるならば，以下の式を用いて近似することができる．

$$\sum_{j=1, i \neq j}^{n} N_j = K - A_i$$

これにより，式 (6) は以下のように書き換えられる．

$$\frac{dN_i}{dt} = \epsilon \left(1 - \frac{N_i + \gamma \cdot (K - A_i)}{K}\right) N_i \tag{7}$$

ここで，N_i および A_i はコネクション i のデータ転送速度および利用可能帯域を表す．K は物理的なボトルネックリンク帯域を表し，ここでは全てのコネクションが同じボトルネックリンクを共有すると仮定している．[20] で提案されているインライン計測手法は，ボトルネックリンクの物理帯域を計測することができないが，ここでは，利用可能帯域および物理帯域の両方を計測可能であると仮定している．提案方式は，上記の式 (7) を TCP のデータ転送速度を調節するためのアルゴリズムとして利用する．

3.　提 案 方 式

3.1　提案アルゴリズム

TCP は，送信ホストが ACK パケットを受信した際にウィンドウサイズを変化させることで，データ転送速度を調節する．そのため，データ転送速度の変化を表す式 (7) を，ACK パケットの受信にともなうウィンドウサイズの変化を表す式へ変形する必要がある．ここでは，コネクション i のウィンドウサイズ w_i を，データ転送速度 N_i を用いて以下の式で表す．

$$w_i = N_i \cdot base\_rtt_i$$

$base\_rtt_i$ は，コネクション i がこれまでに転送したパケットの RTT の最小値を表す．上式を用いて式 (7) を以下のように書き換える．

$$\frac{dw_i}{dt} = \epsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right) w_i$$

TCP は RTT を基本とした動作を行うため，以下のように書き換える．

$$\frac{dw_i}{drtt} = \epsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right) w_i \tag{8}$$

最後に，TCP の送信ホストが ACK パケットを受信した際に増加させるウィンドウサイズの量を求める．w_i 個の ACK を 1 RTT の間に受信すると考えると，以下のように表すことができる．

$$\Delta w_i = \frac{dw_i}{dack} = \epsilon \left(1 - \frac{w_i + \gamma \cdot base\_rtt_i \cdot (K - A_i)}{K \cdot base\_rtt_i}\right)$$

この式が，提案方式でウィンドウサイズを増減するための基本的な式である．なお，この式はネットワークパスの利用可能帯域および物理帯域の情報を必要とするため，提案方式がそれらをインライン計測手法によって得られるまでは，従来の TCP Reno と同じアルゴリズムで動作するものとする．パケット廃棄が発生した場合には，提案方式はウィンドウサイズを TCP Reno と同様に半減させる．また，タイムアウトが発生した場合には，提案方式は全ての計測結果を破棄し，ウィンドウサイズを 1 に設定し，TCP Reno における Slow-Start フェーズを開始する．

3.2　提案方式の特性

a)　ネットワーク帯域に対するスケーラビリティ

ネットワーク内に 1 本の TCP コネクションが存在する場合，時刻 t におけるウィンドウサイズ $w(t)$ は式 (8) から以下のように求められる．

$$w(t) = \frac{w_0 \cdot K \cdot base\_rtt_i}{w_0 + (K \cdot base\_rtt_i - w_0) \cdot e^{-\epsilon t}} \tag{9}$$

ここで，w_0 はウィンドウサイズの初期値を表す．また，$w_0 = (1 - b) \cdot K \cdot base\_rtt_i$，$w_t = c \cdot K \cdot base\_rtt_i$ であるとする ($0 < b < 1$, $0 < c < 1$)．ウィンドウサイズが w_0 から w_t まで増加するために要する時間を T とすると，式 (9) より，

$$T = \frac{1}{\epsilon} \cdot log \left(\frac{c}{1 - c} \cdot \frac{b}{1 - b}\right)$$

となる．この式から，T が K に依存していないことが分かる．すなわち，提案方式はネットワークの帯域に関係なく，同じ時間でウィンドウサイズを増加させることができる．

b） 収 束 時 間

提案方式の収束に関する特性は元のロトカ・ヴォルテラ競争モデルと同じであるため，図 1，2 に見られるように，データ転送速度はある値に素早く収束する．なお，安定して収束するためには $\epsilon \leqq 2$ であることが求められる．

c） 安 定 性

ロトカ・ヴォルテラ競争モデルにおいて，2 種の生物の環境容量が同じ場合には，$\gamma < 1$ であることが両者の生存のために求められる．この条件は提案アルゴリズムでも同じである．しかし，実際のネットワークでは，全てのコネクションの間で物理的なボトルネックリンクが同じであるとは限らず，特にアクセスリンクがボトルネックとなる場合がある．この場合に起こる問題については 4 章で議論する．

d） 周期的なパケット廃棄の回避

ネットワーク内に n 本の TCP コネクションが存在する場合に，収束後のウィンドウサイズの合計値は以下の式で求められる．

$$\sum_{i=1}^{n} w_i = \frac{n}{1+(n-1)\cdot\gamma} \cdot K \cdot base\_rtt_i$$

これは，n が増加するとウィンドウサイズの合計が増加することを示している．しかし，この式において $\lim_{n\to\infty} \sum_{i=1}^{n} w_i$ を計算すると，$\frac{K \cdot base\_rtt_i}{\gamma}$ となることが分かる．すなわち，十分なバッファサイズがあるならばコネクション数に関わらずパケット廃棄は発生しないことを示している．このことは，従来の TCP Reno がそのアルゴリズムの性質上，バッファサイズをいくら大きくしても周期的なパケット廃棄の発生を避けられないことと対照的である．

e） コネクション間の公平性

TCP コネクションの収束後のウィンドウサイズは，式 (7) によって独立に求められる．よって物理帯域 K が同じであるならば，各コネクションのウィンドウサイズが同じ値となるのは明らかである．しかし，K がコネクション間で異なる場合には，問題が起こる可能性がある．この問題については 4 章で議論する．

4. シミュレーション結果

本章では，3 章で提案した TCP の輻輳制御方式の性能をシミュレーションにより評価する．シミュレーションには ns-2 [24] を用いた．提案方式の比較対象として，TCP Reno, HighSpeed TCP (HSTCP) [13]，および Scalable TCP [15] を用いた．提案方式は，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によって取得するが，本シミュレーションにおいては物理帯域の情報は直接与えられるものとする．提案方式のパラメータは $\epsilon = 1.95$，$\gamma = 0.9$ とし，HSTCP および Scalable TCP のパラメータはそれぞれ [13] および [15] において示されているものとする．

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．それは TCP の送受信ホスト，UDP の送受信ホスト，2 つのルータ，およびホストとルータ間を接続するリンクで構成されている．$N_{\rm tcp}$ 本の TCP コネクションが TCP の送信ホスト i および受信ホスト i の間で確立される．バックグラウンドトラヒックを生成するため，データ転送速度 $r_{\rm udp}$ の UDP トラヒックを発生させる．すなわち，$N_{\rm tcp}$ 本の TCP コネクションと UDP トラヒックがルータ間のリンクを共有する．ボトルネックリンクおよび UDP 送受信ホストのアクセスリンクの帯域を BW で表し，伝播遅延時間をそれぞれ τ，$\tau_{\rm u}$ で表す．TCP の送信ホスト i のアクセスリンク帯域，伝播遅延時間をそれぞれ bw_i，τ_i とする．ルータバッファには TailDrop 方式を用い，そのサイズはルータ間リンクの帯域遅延積の 2 倍とする．

まず初めに，提案方式の基本的な振る舞いを確認するため，コネクション数 $N_{\rm tcp}$ を 1 としてシミュレーションを行う．ここでは，$bw_1 = 100$ Mbps, $\tau_1 = 5$ msec, $BW = 100$ Mbps, $\tau = 40$ msec, $\tau_{\rm u} = 5$ msec, $r_{\rm udp} = 50$ Mbps とする．図 4 は，TCP Reno，HSTCP，Scalable TCP，および提案方式のコネクションにおけるウィンドウサイズの変化を表している．この図から，TCP Reno，HSTCP，および Scalable TCP のコネクションにおいてはバッファ溢れによるパケット廃棄が周期的に発生していることが分かる．一方，提案方式においてはウィンドウサイズが理想的な値に素早く収束し，パケット廃棄が発生していないことが分かる．さらに，提案方式のウィンドウサイズの増加速度は HSTCP, Scalable TCP よりも大きい．これらの結果は，提案方式がリンク帯域を効率良く利用できることを示している．

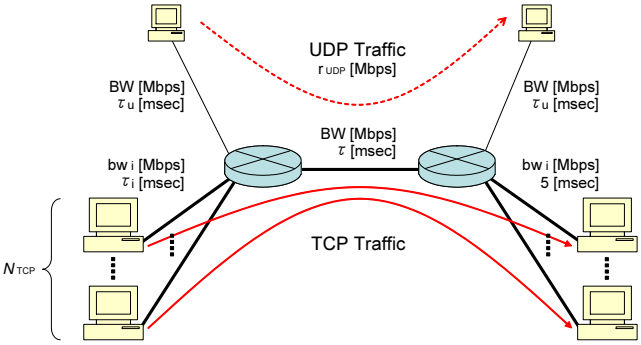


図 3 シミュレーションモデル

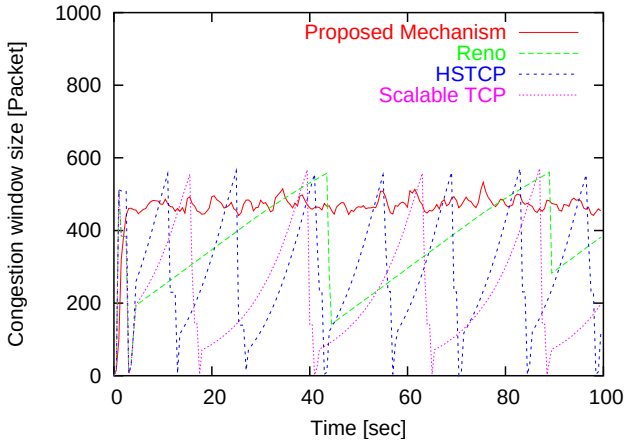


図 4 1 本のコネクションにおけるウィンドウサイズの変化

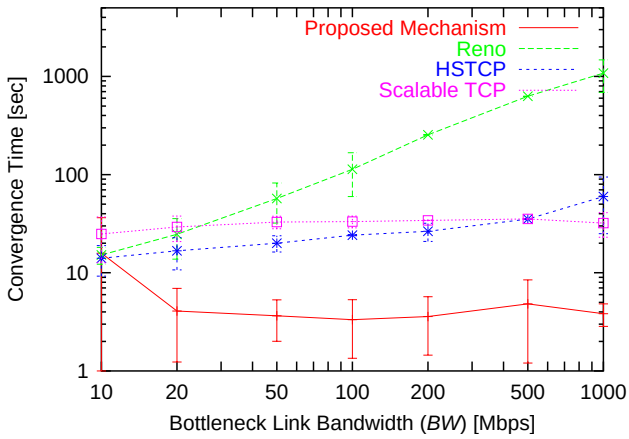


図 5 ボトルネックリンク帯域に対する収束時間

次に，提案方式のリンク帯域に対するスケーラビリティを，収束時間を確認することで評価する．ここでは，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を収束時間と定義し，$N_{\rm tcp} = 1$, $\tau_1 = 5$ msec, $\tau = 40$ msec, $\tau_{\rm u} = 5$ msec とする．また，$r_{\rm udp} = (0.2 \cdot BW)$ Mbps, $bw_1 = BW$ Mbps とする．図 5 は，BW を 10 Mbps から 1 Gbps 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る．この図から，TCP Reno のコネクションはリンク帯域を使い切るまでに非常に長い時間を必要とすることが分かる．これは，ウィンドウサイズの増加量を決定するパラメータがリンク帯域の大きさに関わらず固定されているためである．HSTCP の場合は，TCP Reno の場合に比べて収束時間が大きく減少している．しかし，リンク帯域が大きくなると，それに従って収束時間が大きくなる性質は変化していない．このことは，HSTCP が TCP Reno の問題点を根本的に解決できていないことを示

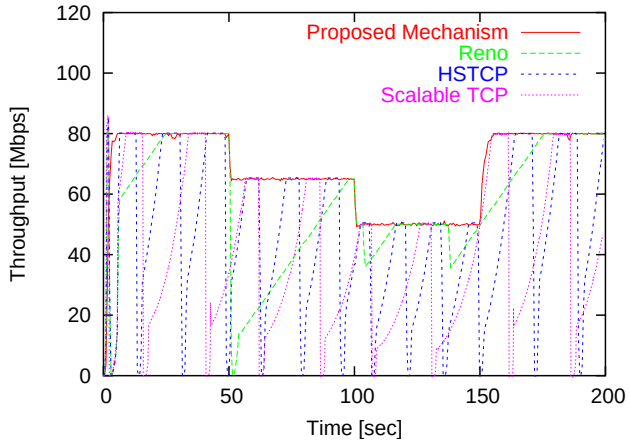


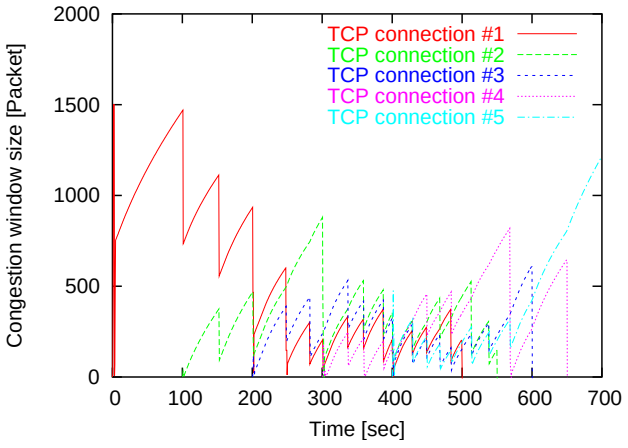
図 6　利用可能帯域の変化への追随性

している．Scalable TCP の場合は，リンク帯域が大きくなっても収束時間がほとんど変化せず, [15] で意図された挙動を示していることが分かる．提案方式における収束時間はリンク帯域に関わらずほぼ一定であり，また 4 つの方式の中で最も小さい．このことは，提案方式がリンク帯域に対する優れたスケーラビリティを有することを示している．なお，信頼区間が大きい原因は，インライン計測において計測誤差が生じることにある．

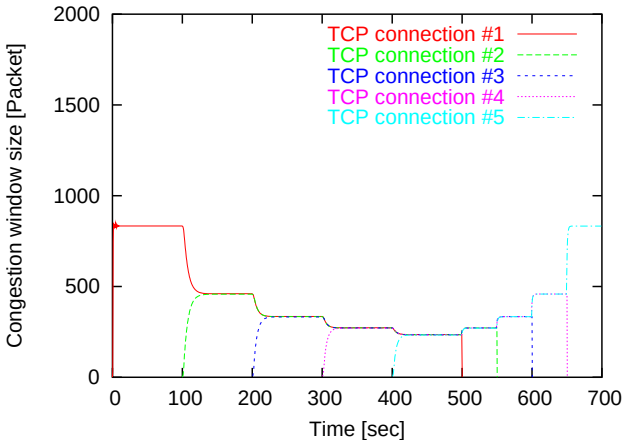
利用可能帯域の変化に追随してデータ転送速度を変更することもまた，トランスポート層プロトコルの重要な性質である．提案方式がこの性質を持つことを確認するため，$N_{\rm tcp} = 1$, $bw_1 = 100$ Mbps, $\tau_1 = 5$ msec, $BW = 100$ Mbps, $\tau = 40$ msec, $\tau_{\rm u} = 5$ msec とし，利用可能帯域が 0 から 50 sec には 80 Mbps，50 から 100 sec には 65 Mbps，100 から 150 sec には 50 Mbps，150 から 200 sec には 80 Mbps となるように $r_{\rm udp}$ を変化させる．図 6 はこのシミュレーションにおける各 TCP コネクションのスループットの変化を表している．この図から，提案方式が利用可能帯域の変化に追随できていることが分かる．また，利用可能帯域が突然減少した場合においても，パケット廃棄が発生していない．一方，TCP Reno, HSTCP, および Scalable TCP の場合には，多くのパケット廃棄が発生しており，リンク帯域を十分に使い切ることができていない．

次に，TCP コネクション数が変化した場合の，提案方式の適応性および公平性を評価する．ここでは，$N_{\rm tcp} = 5$, $bw_i = 100$ Mbps, $\tau_i = 5$ msec $(1 \leqq i \leqq 5)$, $BW = 100$ Mbps, $\tau = 40$ msec とする．なお，このシミュレーションでは UDP トラヒックを発生させない．5 本の TCP コネクションは，それぞれ 0, 100, 200, 300, 400 秒後にネットワーク内に参加しデータ転送を開始し，それぞれ 500, 550, 600, 650, 700 秒後に終了する．図 7 は，TCP Reno および提案方式における，時刻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．これらの図から，TCP Reno においては，コネクション間で公平性を維持できていないことが分かる．これは，公平な状態へ移行するまでに多くの時間を必要とするためである．また，パケット廃棄が周期的に発生することを避けられない．一方，提案方式においては，新しいコネクションが参加した際にウィンドウサイズが素早く収束し，パケット廃棄が発生しない．さらに，TCP コネクションがデータ転送を終了する際にも，提案方式は空いた帯域を素早く使い切ることが可能である．

最後に，アクセスリンク帯域が異なる場合の TCP コネクションの振る舞いを調べる．ここでは，$N_{\rm tcp} = 2$, $bw_1 = 10$ Mbps, $bw_2 = 20$ Mbps, $\tau_1 = \tau_2 = 5$ msec, $\tau = 40$ msec とし，BW は 5 Mbps から 30 Mbps の間で変化させる．このシミュレーションでは UDP トラヒックを発生させない．図 8 は，TCP Reno および提案方式のそれぞれについて，BW の変化に対する 2 本の TCP コネクションのスループットの変化を表す．これらの図から，TCP Reno は，BW の大きさに関わらずボトルネックリンク帯域を公平に分配していることが分かる．一方，提案方式は興味深い性質を示している．$BW < bw_1$ である場合には，2 本のコネクションは公平にボトルネックリンク帯域を分配する．しかし，$bw_1 < BW < bw_2$ である場合には，ボトルネックリンク帯域を bw_1 と bw_2 の比に応じて分配する．この性質は，式 (7) から説明することができる．収束後のデータ転送速度を $\hat{N}_i$ とすると，2 本のコネクションが異なる物理帯域 K_i を持つ場合には以下の式で表される．



(a) TCP Reno



(b) 提案方式

図 7　コネクション数の変化の影響

$$\hat{N}_i = \frac{K_i}{\sum_{i=1}^{n} K_i} \cdot BW \qquad (10)$$

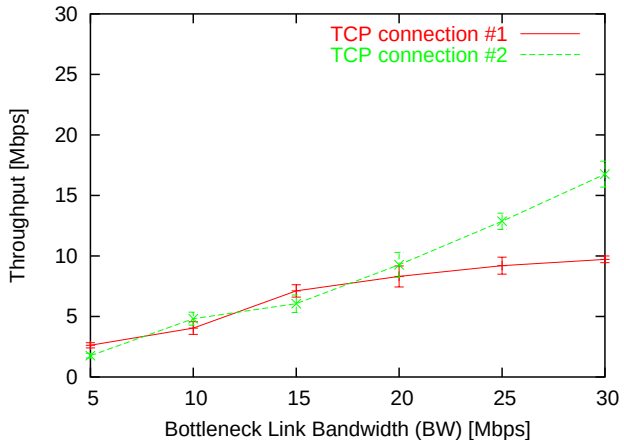
この式は，$\gamma < 1$ の条件の下で成立し，ボトルネックリンク帯域がそれぞれの TCP コネクションの物理帯域の比に応じて分配されることを示している．図 8(b) における提案方式の振る舞いは，式 (10) に合致している．

我々は，この特性がインターネットにおける輻輳制御の方針として理想的であると考える．インターネットにおけるアクセスネットワークの帯域とバックボーンネットワークの帯域の比は常に変動している [25]．すなわち，アクセスネットワーク資源に比べて，バックボーンネットワーク資源は不足であったり過剰であったりする．このとき，バックボーンの資源が少ない場合には，各コネクションはその資源をアクセスリンク帯域に関係なく公平に分配するべきであり，多い場合にはアクセスリンク帯域の比に応じて分配するべきである．図 8(b) および式 (10) で示される性質は，このような資源分配方針を実現するものであるといえる．

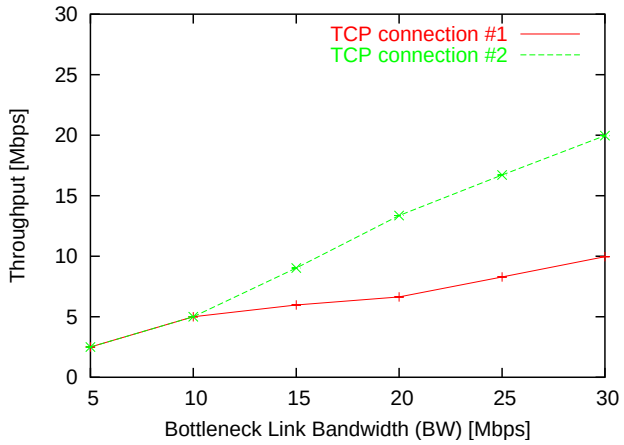
5. おわりに

本稿では，インラインネットワーク計測手法を用いた新しい TCP の輻輳制御方式を提案した．提案方式は，生物の個体数の変化を表す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に基づいている．これらの 2 つのモデルを TCP のデータ転送速度の制御へ応用し，TCP コネクションのウィンドウサイズの制御アルゴリズムを提案した．解析およびシミュレーションによる性能評価を通して，提案方式がリンク帯域に対するスケーラビリティ，収束速度，公平性，安定性に関して優れた性質を持つことを示した．

我々の研究グループでは，ネットワークにおける輻輳制御に



(a) TCP Reno



(b) 提案方式

図 8　アクセスリンク帯域の変化の影響

とって重要な情報である送受信ホスト間のパスの利用可能帯域および物理帯域を計測によって取得することで，理想的な輻輳制御が可能になると考えている．インラインネットワーク計測手法が今後進歩することで，TCP に従来とは異なる輻輳制御方式を適用することが可能となり，その結果 TCP の性能が大きく向上すると考えられる．本稿で提案した方式は，そのような新しい試みの第一歩である．

今後の課題としては，異なる RTT を持つ提案方式のコネクション間における公平性，提案方式のコネクションが TCP Reno のコネクションと競合した場合の公平性，利用可能帯域および物理帯域の計測誤差が提案方式の動作に与える影響など，提案手法の基本的な性質を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] J. B. Postel, “Transmission control protocol,” *Request for Comments 793*, Sept. 1981.

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

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

[4] 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.

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

[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*, 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] 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.

[12] 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.

[13] S. Floyd, “HighSpeed TCP for large congestion windows,” *RFC 3649*, Dec. 2003.

[14] M. Gerla, M. Y. Sanadidi, R.Wang, A. Zanella, C. Casetti, and S. Mascolo, “TCP Westwood: Congestion window control using bandwidth estimation,” in *Proceedings of IEEE GLOBECOM ’01*, Nov. 2001.

[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] 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.

[21] M. L. T. Cao, G. Hasegawa, and M. Murata, “Available bandwidth measurement via TCP connection,” to be presented at IFIP/IEEE MMNS 2004, Oct. 2004.

[22] L. S. Brakmo, S. W.O’Malley, and L. L. Peterson, “TCP Vegas: New techniques for congestion detection and avoidance,” in *Proceedings of ACM SIGCOMM’94*, pp. 24–35, Oct. 1994.

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

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

[25] J. Crowcroft, S. Hand, R. Mortier, T. Roscoe, and A. Warfield, “QoS’s downfall: At the bottom, or not at all!,” in *Proceeding of ACM SIGCOMM 2003 Workshop on Revisiting IP QoS (RIPQOS)*, Aug. 2003.