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TCP輻輳制御方式の実ネットワークにおける性能評価

児玉　瑞穂† 　山根木果奈† 　長谷川　剛†† 　村田　正幸†

† 大阪大学 大学院情報科学研究科 〒 560-0871 大阪府吹田市山田丘 1-5
†† 大阪大学 サイバーメディアセンター 〒 560-0043 大阪府豊中市待兼山町 1-32
E-mail: †m-kodama@ics.es.osaka-u.ac.jp, ††{k-yamanegi,murata}@ist.osaka-u.ac.jp,
†††hasegawa@cmc.osaka-u.ac.jp

あらまし　我々の研究グループでは，トランスポート層における輻輳制御方式として，TCP Symbiosis を提案している. 従来の手法である TCP Reno やその改善手法はネットワークの輻輳の有無を判断するのにパケット廃棄の発生を指標としていたが，本手法はインラインネットワーク計測によって取得したネットワークパスの利用可能帯域に関する情報を利用する．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ウィンドウ制御アルゴリズムによって輻輳制御を行う. 本稿では，インターネット公衆回線を用いたデータ転送実験を通じて，TCP Symbiosis の性能評価を行い，本手法が周期的なパケット廃棄を必要としない効率的なデータ転送を可能にすること，およびネットワークの帯域遅延積に対する高いスケーラビリティを持つことを示す.

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・ヴォルテラ競争モデル

Implementation Experiments of a Congestion Control Mechanism of TCP with Inline Network Measurement

Mizuho KODAMA†, Kana YAMANEGI†, Go HASEGAWA††, and Masayuki MURATA†

† Guraduate School of Information Science and Technology, Osaka Unuversity 1-5 Yamadaoka, Suita, Osaka, 565-0871 Japan
†† Cybermedia Center, Osaka University, 1-32 Machikaneyama, Toyonaka, Osaka, 560-0043, Japan
E-mail: †m-kodama@ics.es.osaka-u.ac.jp, ††{k-yamanegi,murata}@ist.osaka-u.ac.jp,
†††hasegawa@cmc.osaka-u.ac.jp

Abstract　In this report, we investigate the performance of a new congestion control mechanism for TCP, TCP Symbiosis, which is proposed by our previous studies. Whereas the traditional TCP Reno recognizes the network congestion only by detecting packet losses, the proposed mechanism directly obtains the information of physical and available bandwidths of the network path, and utilize an algorithm based on logistic growth model and Lotka-Volterra competition model from biophysics to regulate the congestion window size. We investigate the performance of the proposed mechanism through experiments using the actual public Internet environment. We show the effectiveness of the proposed mechanism, especially in terms of scalability with the bandwidth-delay product of the network path.

Key words　congestion control, inline measurement, available bandwidth, logistic growth model, Lotka-Volterra 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ともない改良が繰り返し行われている (たとえば, [3-5]). TCP は様々な機能を持っているが，その中で最も重要なのは輻輳制御方式 [1] である．この機能により，TCP はネットワーク内で競合する複数のコネクション間でネットワーク帯域を公平に分配するために，データ転送速度を調節している．TCP はウィンドウサイズ，すなわち，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普及している．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り，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 がパケットの廃棄発生のみ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と減少量を決定するパラメータがネットワーク環境に関係なく常に一定であるということに起因する．例えば，ネットワーク帯域や遅延が大きい環境において，TCP Reno はリンク帯域を使い切ることが出来ないために，スループットを低下させてしまうことが挙げられる [11]．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）ことに起因する．この問題に対する解決法は [11-17] などにおいて提案されている. [11, 12] では，TCP Reno のパケット廃棄の発生のみをネットワーク輻輳と判断する基本機構を引き継ぎ，ウィンドウサイズの増減幅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，これらの方法は共存するコネクションとの公平性を維持することができない，および周期的にパケット廃棄が発生するという問題点がある. [13, 14] では，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を提案している．この手法は，単独で用いられる場合にはスループット，公平性および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 や前者の改善手法と混在した環境においては，この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指摘されている [15, 16] . [18, 19] ではネットワーク帯域が使いきれていないときにはウィンドウサイズを TCP Reno よりも大きく増加させ，輻輳発生時には，TCP Reno と同じ増加幅で輻輳ウィンドウサイズを増加させるという手法が提案されている．

TCP Reno および既存の改善手法の特徴の 1 つとして，送受信端末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が挙げられる．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ネットワークパスにおいて現在利用可能な空き帯域（利用可能帯域）と RTT の積で表される．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もっているため，ある意味では利用可能帯域を探索しているということが出来る．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，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る TCP の改善手法が，周期的なパケット廃棄の発生を避けることができ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，送受信端末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出来れば，効率の良いウィンドウサイズ制御手法を用いることが可能となると考えられる．

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 [20-24]. 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出来ない．これらの問題点を持たず，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インラインネットワーク計測 [25, 26] と呼ばれる手法を提案している．この手法は送受信端末間の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るため，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することができる．また，TCP の輻輳制御アルゴリズムを変更するのではなく，送信側端末において 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．

我々の研究グループではさらに，上記の利用可能帯域のインラインネットワーク計測手法を用いた TCP Symbiosis [27] という新たな TCP の輻輳制御方式を提案している．TCP Symbiosis 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，ロトカ・ヴォルテラ競争モデルが用いられている. 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，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適用している．

提案方式の評価はこれまでコンピュータシミュレーションによって行われてきたが，ネットワークの計測結果を利用する本提案手法は，実ネットワーク環境における実験を通じた評価が不可欠である. そこで，本稿では，大阪－東京間のネットワーク，および大阪－米国カリフォルニア州間のインターネット公衆回線を用いたデータ転送実験を通じて TCP Symbiosis の性能評価を行う．これにより，TCP Symbiosis が周期的なパケット廃棄を必要としない効率的なデータ転送を可能にすること，およびネットワークの帯域遅延積に対する高いスケーラビリティを持ち，実ネットワーク環境においても性能を発揮できることを示す．

本稿の構成は以下のとおりである. 2 章ではインラインネットワーク計測手法および TCP Symbiosis の説明を行う．3 章ではインターネット上での TCP Symbiosis の性能評価を行う. 4 章で本稿のまとめと今後の課題を示す.

2. 関 連 研 究

2.1 インラインネットワーク計測

インラインネットワーク計測 [25, 26]（図 1 参照）とは，TCP コネクションがデータ転送に用いるデータパケットおよび ACK パケットのみを用いてエンドホスト間のネットワークパスの帯域情報を計測する技術である．この手法では，従来の手法とは異なり計測用のパケットを必要としないので，ネットワークに余計な計測負荷を掛けない．また，アクティブな TCP コネクションのデータ転送を利用することにより，送信端末の修正のみで計測手法を実装することができるという利点を持つ．我々のグループでは，この計測概念に基づく計測手法として ImTCP を提案している．この手法では，非常に短い周期 (1-4RTT) で継続的に利用可能帯域値を取得することができるため，ネットワークの状況の変化にすばやく追随することができる．

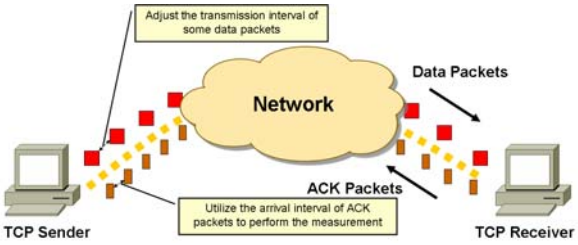


図 1: インラインネットワーク計測手法

利用可能帯域を計測する際には，現在の利用可能帯域が含まれていると考えられる帯域の上限と下限を過去の計測結果を用いて設定し，この区間の中から利用可能帯域を検索する（この区間を探索区間と呼ぶ）．探索区間を設定することで，不必要に高いレートでパケットを送出することが避けられるため，ネットワークに与える影響を最小限に与えることが出来る．また，計測に必要なパケット数を大幅に削減することも可能となる．探索区間は過去の計測結果を基に設定するため，ネットワーク状況の変化に伴い利用可能帯域が急激に変化した場合，探索区間内に利用可能帯域が存在しない場合が存在する．ImTCP では，そのような場合においても，数回の計測で新たな利用可能帯域を発見することが出来る．

2.2 TCP Symbiosis

TCP Symbiosis 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．

一般に，ある特定の環境下において，特性（環境容量，増殖率，および競争相手種の存在による増殖低下率）の等しい生物の個体数は収束し，収束する個体数は等しい．また，環境の変化が発生すると，各々の生物の個体数は直ちに変動する．したがって，生物の個体数の変化をデータ転送に適用すると，公平性および安定性，環境の変化に対する追随性などの利点が得られることが期待できる．

本章では，TCP Symbiosis のウィンドウサイズ制御アルゴリズム，および輻輳制御アルゴリズムを説明する．

2.2.1 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，ある限られた領域の中で生息している一種の生物の個体数 N の変化を表している．一般的に，領域中の個体数が多くなるにつれてその増殖速度は大きくなる．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 K 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は次式で表される [28].

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

ここで，ϵ は種の内的自然増殖率を表す $(\epsilon > 0)$.

次に示すロトカ・ヴォルテラ競争モデルは，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をも考慮した，生物の個体数の変化を表すモデルである．種の個体数は式 (1) を拡張して次式で表される [28].

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

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

ここで，N_1，K_i，ϵ_i はそれぞれ種 i の個体数，環境容量，および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，$K = K_1 = K_2$，$\epsilon = \epsilon_1 = \epsilon_2$，および $\gamma = \gamma_{12} = \gamma_{21}$ 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 $0 < \gamma < 1$ であることが知られている [28]．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し，次のように TCP の輻輳制御に適用する．生物 i の個体数を TCP コネクション i のデータ転送速度に，K を物理帯域，他の生物の個体数の合計を他のコネクションのデータ転送速度の合計（物理帯域 K と利用可能帯域 A_i を用いて $K - A_i$ に近似する）とみなす．これにより，式 (2) , (3) は次のように書き換えられる．

$$\frac{d}{dt}N_i = \epsilon\left(1 - \frac{N_i + \gamma(K - A_i)}{K}\right)N_i \tag{4}$$

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 (4) 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 TCP コネクションの 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 (4) は次のように書き換えられる．

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

最後に，微分方程式で表された式 (5) を積分することにより，次式のように $w_i(t)$ を導出することができる．

$$w_i(t) = \frac{w_i(0)\tau_i f_i(t)\{K - \gamma(K - A_i)\}}{w_i(0)(f_i(t) - 1) + \tau_i\{K - \gamma(K - A_i)\}} \tag{6}$$

ただし，

$$f_i(t) = e^{\epsilon t\left\{1 - \gamma\left(1 - \frac{A_i}{K}\right)\right\}} \tag{7}$$

であり，$w_i(0)$ は時刻 0 におけるウィンドウサイズとする．式 (6) 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，任意の時刻におけるウィンドウサイズ $w_i(t)$ を得ることができる．

2.2.2 輻輳制御アルゴリズム

TCP Symbiosis 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，重複 ACK パケットの受信によってパケット廃棄を検出した場合には TCP Reno と同様にウィンドウサイズを半減させる．タイムアウトによってパケット廃棄を検出した場合には，TCP Reno と同様にウィンドウサイズを 1 [packet] に設定し Slow-Start フェーズを開始し，それまでに得られた全ての計測結果を破棄する．

ImTCP によってネットワークパスの帯域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 とすると，j 個目の ACK を受信したときのウィンドウサイズは，式 (6) 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．

2.2.3 実　　装

ImTCP および TCP Symbiosis は，Linux のカーネルバージョン 2.6.16.21 を変更することによって実装している．ディストリビューションは Fedora Core 5 である．以下，実装のために変更および追加したファイルを記す．

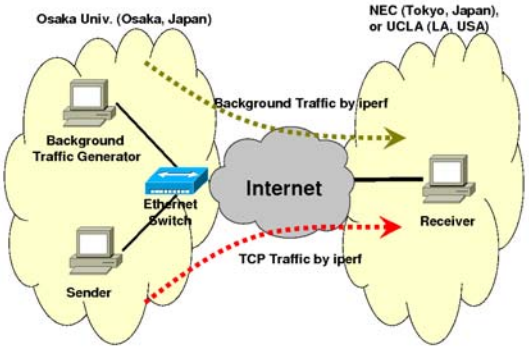


図 2: インターネット公衆回線における実験環境

- 変更したファイル

<net/ipv4/tcp_input.c>, <net/ipv4/tcp_output.c>, <net/ipv4/tcp_output.c>, <net/ipv4/tcp_ipv4.c>, <net/ipv4/sysctl_net_ipv4.c>, <include/linux/tcp.h>, <include/linux/sysctl.h>, <include/linux/jiffies.h>, <include/net/inet_timewait_sock.h>, <arch/i386/kernel/smpboot.c>, <arch/i386/kernel/cpu/proc.c>, <init/calibrate.c>, <kernel/timer.c>

- インラインネットワーク計測の実装ファイル

<include/linux/tcp_imtcp.h>, <include/linux/imtcp.h>, <net/ipv4/tcp_imtcp.c>, <net/ipv4/imtcp.c >

- TCP Symbiosis の実装ファイル

<net/ipv4/tcp_biosis.c>

3. インターネット環境における性能評価

本章では，インターネット公衆回線を用いてデータ転送実験を行い，TCP Symbiosis の性能評価を行う．実験ネットワークとして，帯域遅延積の小さい環境である大阪大学（大阪府）と NEC（東京都）間，および帯域遅延積の大きい環境である大阪大学と University of California, Los Angeles; UCLA（米国カリフォルニア州）間でそれぞれ実験を行った．実験ネットワークは図 2 のように構成し，送信側端末およびクロストラヒック発生用の端末を大阪大学に，受信側端末を他方に設置する．Sender より TCP トラヒックを，Background Traffic Generator よりクロストラヒックとして UDP トラヒックを受信側に流す．これらのトラヒックは iperf [29] により発生させる．受信側では Delayed ACK オプション [30] を無効にしている．1 秒間に発生するシステムのタイマー割り込み回数 (=HZ) は 20000 であり，計測の最小粒度は 50 μs となる．また，TCP Symbiosis のパラメータである γ，ϵ は $\gamma = 0.9$，$\epsilon = 1.95$ と設定して実験を行う．

3.1 帯域遅延積の小さい環境における性能評価

まず，帯域遅延積の小さい環境における評価を行う．大阪－東京間のネットワーク環境は，送受信端末間の往復伝播遅延時間が約 16 ms，ボトルネックリンク帯域は約 40 Mbps である．実験ネットワークを構築する各ホストの性能を表 1 に示す．本実験においては，利用可能帯域が 0-20 sec で 20 Mbps，20-40 sec で 10 Mbps，40-60 sec で 30 Mbps となるように UDP トラヒックをクロストラヒックとして流し，TCP コネクションのスループットを受信側で tcpdump [31] を用いて監視する．TCP Symbiosis と TCP Reno のスループットを比較したものを図 3(a) に示す．

この実験環境のように帯域遅延積が小さい環境においては，TCP Reno を用いた場合においてもリンク帯域を十分に使い切ることができることが知られている．一方，TCP Symbiosis はインラインネットワーク計測の計測結果が正確な場合 (20-60 sec) には TCP Reno と同程度のスループットを発揮でき，リンク帯域を使いきれているということができる．しかし，インラインネットワーク計測の計測結果が実際の利用可能帯域より小さい場合 (40-60 sec) には TCP Reno に比べてスループットが劣ってしまう．

したがって，帯域遅延積が小さい環境においてはインラインネットワーク計測の計測精度が低下すると，TCP Symbiosis のスループットも低下するということがいえる．

表 1: 大阪－東京間の実験環境を構築する端末の性能

	送信側端末 (実験トラヒック用)	送信側端末 (クロストラヒック用)	受信側端末
CPU	Pentium 4 3.40 GHz	Xeon 3.60 GHz	Xeon 2.66 GHz
Memory	1,024 MB	2,048 MB	1,024 MB
Kernel	Linux 2.6.16.21	Linux 2.6.15	Linux 2.4.21

表 2: 大阪－米国カリフォルニア間の実験環境を構築する端末の性能

	送信側端末 (実験トラヒック用)	送信側端末 (クロストラヒック用)	受信側端末
CPU	Pentium 4 3.40 GHz	Xeon 3.60 GHz	Xeon 3.06 GHz
Memory	1,024 MB	2,048 MB	1,024 MB
Kernel	Linux 2.6.16.21	Linux 2.6.15	Linux 2.6.19

3.2 帯域遅延積の大きい環境における性能評価

次に，帯域遅延積の大きい環境においての TCP Symbiosis の評価を行う．大阪－米国カリフォルニア間のネットワーク環境は，送受信端末間の往復伝播遅延時間が約 118 ms，ボトルネックリンク帯域が約 95 Mbps となるようにネットワークを構成している．実験ネットワークを構築する各ホストの性能を表 2 に示す．本実験においては，利用可能帯域が 0-20 sec で 80 Mbps，20-40 sec で 30 Mbps，40-60 sec で 50 Mbps となるように UDP トラヒックをクロストラヒックとして流す．TCP Symbiosis と TCP Reno のスループットを比較したものを図 3(b) に示す．

TCP Symbiosis のスループットに着目すると，TCP Reno より高いスループットでデータ転送を行っていることがわかる．これは，インラインネットワーク計測の誤差が大きい 20-60 sec の間でパケット廃棄が周期的に発生しているものの，ウィンドウサイズの増加が TCP Reno に比べて大きいためである．

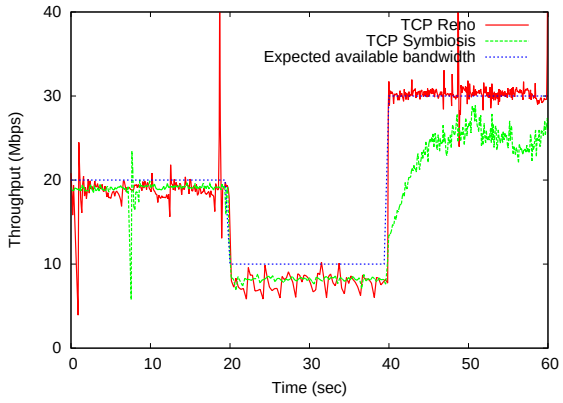
したがって，帯域遅延積の大きい環境においては，インラインネットワークの計測誤差を考慮に入れても，TCP Symbiosis は TCP Reno に比べて高いスループットでデータ転送を行うことができるということができる．

3.3 TCP Symbiosis と他の高速 TCP との比較

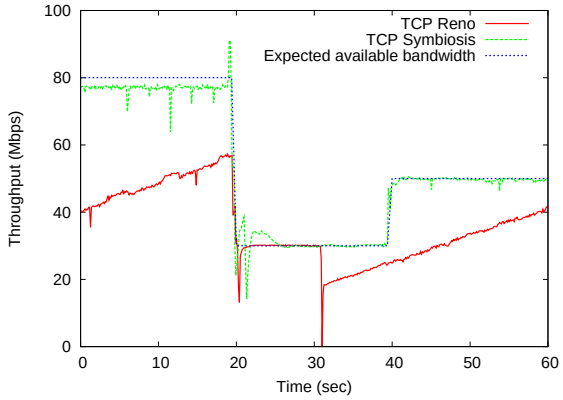
大阪大学 － UCLA 間のネットワーク（ボトルネックリンク帯域 95 Mbps, RTT 118 ms）に iperf による TCP トラヒック 1 本を 180 秒流すことにより，TCP Symbiosis と他の高速 TCP のスループットを比較する．比較対象は TCP Reno, HighSpeed TCP (HSTCP) [11] , Scalable TCP (STCP) [12], CUBIC [17], および Compound TCP (CTCP) [18] である．

本実験においては，利用可能帯域が 0-60 sec で 75 Mbps，60-120 sec で 27 Mbps，120-180 sec で 45 Mbps となるように UDP トラヒックをクロストラヒックとして流す．ウィンドウサイズ，スループットをデータ転送の送信側で監視する．各 TCP のウィドウサイズの変化を図 4 に，スループットの変化を図 5 に示す．

図 4 より，TCP Reno は利用可能帯域が大きい場合，スループットが上がるのに時間がかかることがわかる．また，TCP Reno, HSTCP, STCP, および CUBIC のコネクションは，周期的にパケット廃棄が発生し，その度にウィンドウサイズが小さくなっている．CTCP のコネクションにおいても 60-120sec でも

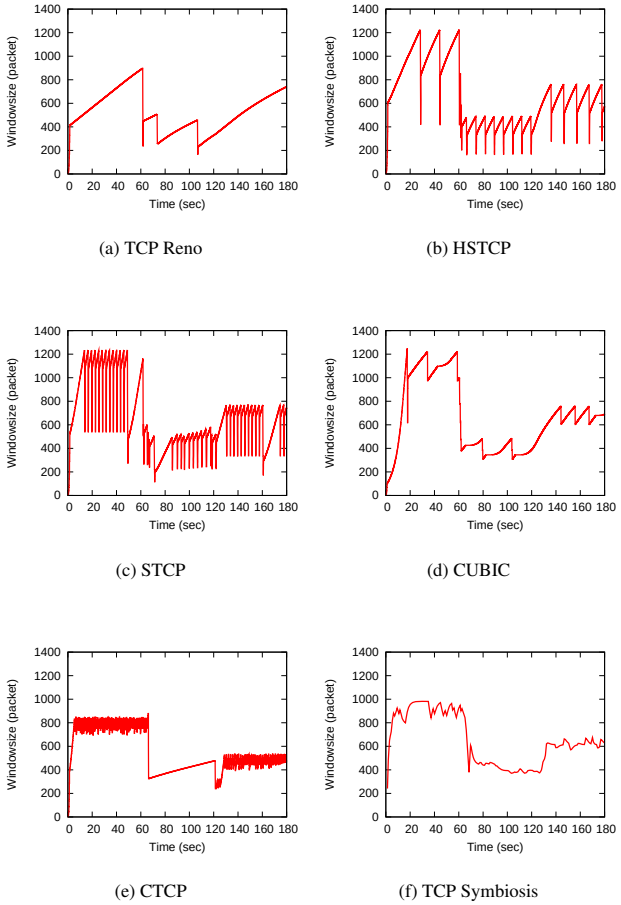


(a) 大阪-東京間



(b) 大阪-米国西海岸間

図 3: TCP Symbiosis と TCP Reno とのスループット比較



(a) TCP Reno (b) HSTCP

(c) STCP (d) CUBIC

(e) CTCP (f) TCP Symbiosis

図 4: 各 TCP のウィンドウサイズの変化

同様にパケット廃棄の発生によりウィンドウサイズが小さくなっている．加えて，STCP, HSTCP, および CUBIC はパケット廃棄の発生によるスループットの振動が発生している．それに対し，TCP Symbiosis においてはインラインネットワーク計測の計測結果によってウィンドウサイズが上下するものの，周期的なパケットサイズは発生しない．そのため，他の TCP にくらべ TCP Symbiosis は安定したスループットでデータ転送を行うことができる（図 5 参照）．

図 5 より，利用可能帯域が 25 Mbps となるとき (60-120 sec), HSTCP，および STCP のようなウィンドウサイズを急激に上げる TCP が UDP を押しのけて高めのスループットを得ていると言える．加えて，TCP Symbiosis 以外の TCP が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のに対し，TCP Symbiosis はインラインネットワーク計測の計測に基づいてウィンドウサイズを制御するので，他の TCP に比べるとスループットが小さくなっていることがわかる．

最後に，CTCP が RTT に基づいてウィンドウサイズを制御しているとき (0-60, 120-180 sec) における性能を TCP Symbiosis と比較する．0-60, 120-180 sec の間は CTCP も TCP Symbiosis のように周期的なパケット廃棄が発生していない．しかし，CTCP は RTT に基づいてウィンドウサイズを制御しているため，RTT の変動によりウィンドウサイズが振動している．それに対し，TCP Symbiosis は利用可能帯域の値に応じてウィンドウサイズを制御しているため，ウィンドウサイズはインラインネットワーク計測の計測結果により上下するものの、振動しない．したがって，スループットも CTCP に比べて安定している．

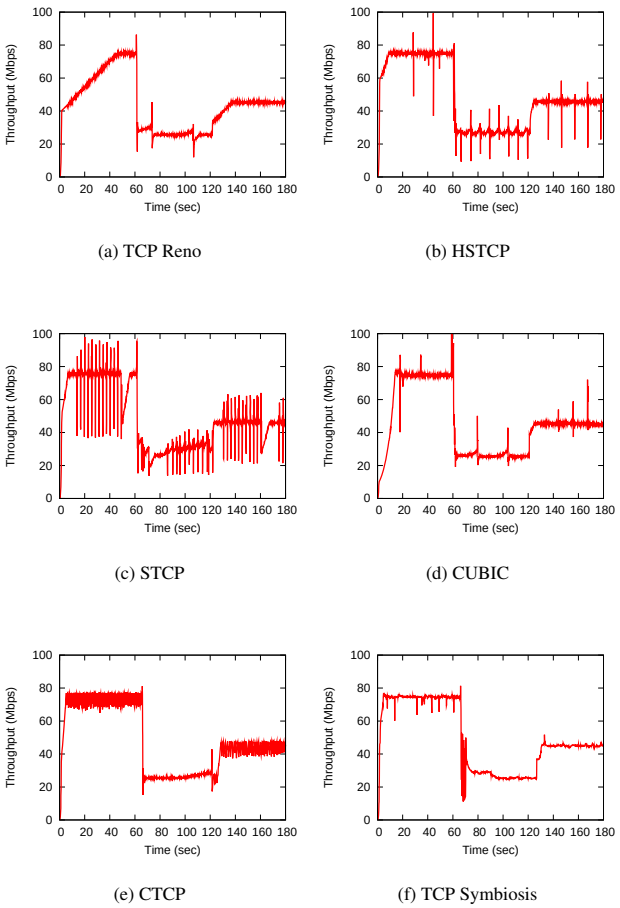
4. おわりに

本稿では，TCP の輻輳制御方式である TCP Symbiosis の性能評価をインターネット公衆回線上での実験を通して行った．その結果，TCP Symbiosis は帯域遅延積の大きいネットワークにおいてインラインネットワーク計測の粒度による計測誤差が生じたとしても，既存の方式である TCP Reno より高いスループットが得られることがわかった．また，インラインネットワーク計測の計測結果が正確な限りは周期的なパケット廃棄を避けることができ，帯域遅延積の小さいネットワークにおいても TCP Reno と同程度のスループットを得ることを示した．また，TCP Reno および他の改善手法と比較した結果，本手法のみが周期的なパケット廃棄を避けることができることを示した．

また，TCP Symbiosis の制御がインラインネットワークの計測結果に強く依存しており，その精度次第ではスループットが低下してしまうという問題が明らかとなった．インラインネットワークの計測誤差の原因は粒度が関係しており，計測精度そのものの向上は難しいと考えられる．そこで，今後の課題としては，インラインネットワーク計測の計測結果を効率的にフィルタリングすることにより，TCP Symbiosis のスループット向上を目指したい．また，より高速なネットワーク環境における性能評価を行うことも今後の課題として挙げられる．

なお，提案方式のベースとなっている ImTCP および TCP Symbiosis のソースコードは `http://www.anarg.jp/imtcp/` で公開している．

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



(a) TCP Reno (b) HSTCP

(c) STCP (d) CUBIC

(e) CTCP (f) TCP Symbiosis

図 5: 各 TCP のスループットの変化

制度委託研究「ネットワークサービスの早期展開を実現するオーバレイネットワーク基盤の研究開発」、および平成 18 年度文部科学省科学研究費若手研究 (A)「10Gbps のスループットを達成する真にスケーラブルな TCP の輻輳制御方式の開発」によっている。ここに記して謝意を表す。

文　献

[1] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated Volume 2 : The Implementation (Addison-Wesley Professional Computing Series)*. Addison-Wesley Pub (Sd), 1994.

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

[3] V. Jacobson, R. Bradon, and D.Borman, "TCP extensions for high performance," *Requenst for Comments 1323*, May 1992.

[4] M. Allman, H. Balakrishman, and S. Floyd, "Enhancing TCP's loss recovery using limited transmit," *Request for Comments 3042*, Jan. 2001.

[5] E. Blanton, M. Allman, and L. W. K. Fall, "A conservatic selective acknowledgement (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*, 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 convergestion 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," *Request for Comments 3649*, Dec. 2003.

[12] T. Kelly, "Scalable TCP: Improving perfomance in highspeed wide area networks," in *Proceedings of PFLDnet 2003*, Feb. 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] L. S. Barkmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Jounal on Selected Areas in Communications*, vol. 13, pp. 1465–1480, Mar. 1995.

[15] J. Mo, R. J. La, V. Anantharam, and J. Walrand, "Analysis and comparison of TCP Reno and Vegas," in *Proceedings of IEEE INFOCOM '99*, 1999.

[16] 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.

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

[18] 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.

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

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

[21] 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.

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

[23] 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.

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

[25] Cao Le Thanh Man, Go Hasegawa, Masayuki 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, Issue 10, June 2006.

[26] Cao Le Thanh Man, Go Hasegawa, Masayuki 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.

[27] Go Hasegawa and Masayuki Murata, "TCP Symbiosis: Congestion control mechanisms of TCP based on Lotka-Volterra competition model," in *Proceedings of Workshop on interdisciplinary systems approach in performance evaluation and design of computer & communications systems (Inter-Perf 2006)*, Oct. 2006.

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

[29] NARANR, "NLANR/DAST: Iperf 1.7.0 - The TCP/UDP Bandwidth Measurement Tool." available from `http://dast.nlanr.net/Projects/Iperf/`.

[30] B. R., "Requirement for internet hosts – communication layers," *Request for Comments 1122*, Oct. 1989.

[31] L. N. R. Group, "TCPDUMP public respository." available from `http://www.tcpdump.org/`.