情報処理学会論文誌 Vol.60 No.2 479–490 (Feb. 2019)

輻輳ウィンドウとその増加分に着目したパッシブなTCP輻輳制御アルゴリズムの推定方法

加藤 聰彦[1,a)] 小田 淳[1] 厳 笑凡[1] 山本 嶺[1] 大坐畠 智[1]

受付日 2018年5月6日, 採録日 2018年11月7日

概要：近年ネットワーク環境の多様化にともない，多くのTCP輻輳制御アルゴリズムが提案されている．TCPの輻輳制御はインターネットの通信状況に大きく影響するため，どのようなアルゴリズムが広く使用されているかを検知することは重要な課題である．このためにネットワーク事業者では，収集されたパケットトレースのみから個々のTCPフローの輻輳制御アルゴリズムを推定するパッシブ方式を採用する必要がある．本論文では，往復遅延時間（RTT）ごとの輻輳ウィンドウを推定し，輻輳ウィンドウとその増加分の対応をとることにより，輻輳制御アルゴリズムを推定する2種類の方法を提案する．第1が，パケットトレースとして双方向の情報が使用できる場合の方法で，データセグメントとACKセグメントの対応関係からRTTを推定し，その間のデータ送信量から輻輳ウィンドウサイズを推定し，その値と増加分を対応させる．第2の方法は，パケットトレースとして片方向の情報（データセグメントの情報）のみが格納されている場合に対応するもので，シーケンス番号の時間的変化をグラフ化し，それを1次から4次までの関数で近似し，その関数の1階微分と2階微分の係数の対応をとる．これらの方法を，TCP Reno，CUBIC TCP，Hamilton TCP，TCP Vegas，TCP Venoに適用し，輻輳制御方式を判別することが可能であることを示す．

キーワード：TCP輻輳制御アルゴリズム，パッシブモニタリング，輻輳ウィンドウ

Inferring TCP Congestion Control Algorithms Passively by Focusing on Congestion Window and Its Increment

Toshihiko Kato[1,a)] Atsushi Oda[1] Xiaofan Yan[1] Ryo Yamamoto[1] Satoshi Ohzahata[1]

Received: May 6, 2018, Accepted: November 7, 2018

Abstract: Recently, according to the diversification of network environments, various TCP congestion control mechanisms have been introduced. Since the TCP congestion control algorithms affect the performance of the Internet, it is important to analyze which algorithms are used widely. This paper focuses on a passive scheme to infer a congestion control algorithm from passively collected packet traces by estimating congestion window at round-trip time (RTT) intervals, and inferring congestion control algorithms by correlating estimated window sizes and their increments. Specifically, we propose two methods. One is a method for bidirectional packet traces that estimates congestion window sizes by mapping data and ACK segments. The other is a method for unidirectional traces including only data segments. It uses the curve fitting for sequence number vs. time graphs by applying the least squares method with linear through quartic functions, and maps the first-order and second-order differentiations. This paper shows the effectiveness of the proposed methods by applying them to various TCP congestion control algorithms including TCP Reno, CUBIC TCP, Hamilton TCP, TCP Vegas, and TCP Veno.

Keywords: TCP congestion control algorithms, passive monitoring, congestion window

1. はじめに

TCPの輻輳制御は，ネットワークの混雑時にエンド端末からのデータ送信を制限する．このため，送信側がデー

[1] 電気通信大学
University of Electro-Communications, Chofu, Tokyo 182–8585, Japan
[a)] kato@is.uec.ac.jp

タの送出量を規定する輻輳ウィンドウを独自に管理し，新たな ACK を受信するとその値を増加させ，輻輳を検出すると減少させる．TCP の輻輳制御が導入されてから [1]，長期にわたり，Tahoe，Reno，NewReno [2] といった限られたアルゴリズムのみが使用されてきた．近年，ネットワークの多様化に応じて，様々な輻輳制御アルゴリズムが提案されている [3]．たとえば，高速・高遅延のネットワークでは，High Speed（HS）TCP [4]，CUBIC TCP [5]，Scalable TCP [6]，Hamilton TCP [7] が使用されている．また TCP Westwood [8] は損失のある無線リンクでの利用を想定している．これらはパケット損失を輻輳の指標とする損失ベース方式である．そのほかに，RTT（Round-Trip Time）の増加を輻輳の指標とする遅延ベース方式，2 つを組み合わせたハイブリッド方式も存在する．TCP Vegas [9] は遅延ベース方式の例である．また TCP Veno [10]，TCP Illinois [11]，Compound TCP [12] は損失を契機に輻輳制御を起動するが，RTT の増加状況により輻輳ウィンドウの変更方法を調整するという意味で，ハイブリッド方式に位置づけられる．

輻輳制御はインターネットを流れるトラヒックに影響を与えるため，どのアルゴリズムが広く利用されているかを知ることは重要である．輻輳制御アルゴリズムの推定手法は大きく 2 つに分類される．1 つは受動的に収集されたパケットトレースに基づき推定を行うパッシブ方式である．もう一方はアクティブ方式で，テスタが対象となるノードとの間で（通常手順ではない）テスト用のセグメント列を送信し，その結果により輻輳制御アルゴリズムを推定する．

これまでに双方のアプローチに対していくつかの研究が報告されている．パッシブ方式に対しては，双方向の情報を含むパケットトレースに基づき，輻輳制御のアルゴリズムや内部パラメータ値を推定する提案がある [13], [14], [15], [16], [17], [18]．しかしこれらは推定するアルゴリズムの種類や条件が限られている．また文献 [19] ではパケットトレースが片方向の情報のみを含む場合を対象として，不当な再送動作を行う TCP フローの検出などを行っている．一方，アクティブ方式においては，近年提案・実装されている 13 の輻輳制御アルゴリズムを識別する方法が提案されている [20]．

これに対して，筆者らはパッシブ方式を用い．現在使用されている多くの輻輳制御アルゴリズムを識別する方法を提案している [21], [22], [23], [24]．この提案では，パケットトレースにおいて，再送が行われていない範囲（シーケンス番号が増加している範囲）から，RTT ごとの輻輳ウィンドウサイズを推定し，ウィンドウサイズ自身の値とその増加分の関係から輻輳制御アルゴリズムを識別するというアプローチを採用した．このうち，文献 [21] から [23] では双方向のパケットトレースを使用し，複数のアルゴリズムに対して区別可能であることを示している．

これまでの方法は輻輳ウィンドウサイズとその増加分の厳密な推定が必要であったため，データと ACK の双方のトレースが必須となる．しかしインターネットのバックボーンでは往路と復路の IP パケットが別の経路を経由する場合がある．このため，バックボーンの 1 カ所でパケットトレースを収集すると，データセグメント（または ACK セグメント）のみの情報しか得られないことになる．筆者らは，このような片方向のパケットトレースに上記の方法を適用する検討を行ったが [24]，うまく推定できない場合があった．そこで新たに，シーケンス番号の時間的変化のグラフを，1 次から 4 次の範囲で，最小二乗法により最もよく近似できる関数を決定し，その関数の 1 階微分と 2 階微分を，輻輳ウィンドウサイズとその増加分に比例した値と見なし，それらの対応関係からアルゴリズムを推定する方法を考案した．

本論文では，筆者らの考案した 2 つの推定方法の詳細と，TCP Reno，CUBIC TCP，Hamilton TCP，TCP Vegas，TCP Veno に対して適用した結果について述べる．本論文の構成は次のとおりである．2 章では輻輳制御アルゴリズムの推定に関する関連研究を示す．3 章で本論文での提案手法の基本原理を示す．4 章では提案を双方向のパケットトレースに適用する手法とその結果を，5 章ではデータセグメントの情報のみを含む片方向パケットトレースに適用する手法とその結果をそれぞれ述べる．最後に 6 章で結論を示す．

2. 関連研究

パッシブ方式を用いた初期の研究 [13], [14], [15] では，パケットトレースの情報から，TCP の送信側の内部状態や内部変数（輻輳ウィンドウサイズ（cwnd）やスロースタート閾値（ssthresh）など）を推定し，使用されている輻輳制御を類推する．これらの研究では Tahoe，Reno，NewReno などの限られた輻輳制御アルゴリズムのみを対象としている．近年の研究である文献 [16] も同様なアプローチを用いており，NewReno を対象とし cwnd と ssthresh をリアルタイムに推定する方法を提案している．文献 [17], [18] では，Linux に実装されている 13 の輻輳制御アルゴリズムの識別を対象としているが，13 のアルゴリズムのうち任意の 2 つの組を対象として，そのどちらが実装されているかを区別することを目的としている．推定にあたっては，RTT ごとの cwnd を推定し，その値の増加率，1 セグメント分だけ増加した回数の割合などの特徴量を求め，その特徴量をクラスタ分析するという手法を用いている．以上の提案では双方向のパケットトレースの使用を前提としている．

文献 [19] は，片方向のパケットトレースを用いたもので，輻輳制御アルゴリズムとは別の視点からの研究である．具体的には，cwnd の初期値，再送中に輻輳ウィンドウを減少させない不当なフローの識別，データ転送の動作周期（フ

ロークロック）の抽出によるアプリケーションの識別を行う方法を提案している．

一方，アクティブ方式の研究例 [20] では，テスタが Web サーバに対してテスト用のセグメントを送り，サーバが 13 の輻輳制御方式のいずれを実装しているかを判定する．この方法では，データ転送開始後に一定数のセグメントをサーバから受信するとあえてタイムアウトを発生させ，その後の ACK セグメントの送信タイミングを調整するといったテスト系列を用いている．

3. 提案手法の基本原理

3.1 概要

本論文では，TCP の輻輳回避フェーズの動作に着目して輻輳制御アルゴリズムを識別することを考える．TCP では輻輳回避フェーズの動作に関して，いくつかの規定方法がある．第 1 は，TCP Reno，HS TCP，Hamilton TCP などで行われているように，ACK セグメントを受信するごとに cwnd をどのように変化させるかを定める方法である．次が，TCP Vegas のように，RTT ごとの cwnd 変更方法を定めるものである．さらに，CUBIC TCP のように，輻輳検出時（セグメント再送時）からの時間で cwnd を規定する方法もある．異なる方法で規定された輻輳制御アルゴリズムを，特定の 1 つの手法で識別するために，RTT ごとの cwnd を推定し，その値の変化に着目することとした．これは TCP Reno のように ACK ごとの動作を規定している場合も，RTT ごとの変化を想定しているためと，パケットトレースから推定できる cwnd は，最大セグメントサイズ（MSS：Maximum Segment Size）の大きさで丸められ，送信ノード内部の正確な cwnd の値は検知できないためという 2 つの理由による．

本論文では，前述のように，推定された cwnd の値とその増加分の対応に着目する．TCP Reno や TCP Vegas では cwnd の値に関係なくその RTT ごとの増加分を定めている．一方 HS TCP では，cwnd の値により増加分を変化させている．また CUBIC TCP や Hamilton TCP では cwnd が直前の輻輳検出時からの時間の関数となっているが，この場合も cwnd とその増加分の対応関係を求めることは可能である．このため，cwnd とその増加分（以降 Δcwnd と呼ぶ）に対して，Δcwnd を輻輳制御アルゴリズム固有の cwnd の関数として定めることを提案する．具体的には，パケットトレースから求めた cwnd と Δcwnd に対して，(cwnd, Δcwnd) をプロットし，その結果がどの関数に相当するかを調べることで，輻輳制御方式を識別する．

3.2 個別の輻輳制御アルゴリズムへの適用

(1) TCP Reno

TCP Reno では，輻輳回避フェーズにおいて cwnd が RTT の間に 1 セグメントだけ増加するように，ACK セグメントを受信するごとに 1/cwnd セグメントずつ cwnd を増加させる．しかし実際には，連続したデータ転送における遅延 ACK により，2 データセグメントに 1 つの ACK セグメントが送信されるのみである．このため cwnd の RTT ごとの増加分は 1/2 セグメントとなる．パケットトレースから推定される cwnd は MSS の整数倍となるため，TCP Reno における RTT ごとの cwnd の増加状況は cwnd の値に関係なく．$\Delta\,\mathrm{cwnd} = 1$ or 0 となる．

(2) CUBIC TCP

CUBIC TCP では，cwnd は直前の輻輳事象からの経過時間の関数として以下の 3 次式で与えられる．

$$\mathrm{cwnd} = C\left(T - \sqrt[3]{\beta \cdot \frac{\mathrm{cwnd}_{max}}{C}}\right)^3 + \mathrm{cwnd}_{max} \quad (1)$$

ここで cwnd_{max} は直前の輻輳発生時の cwnd の値，C は定数，β は減少パラメータ，T は直前の輻輳事象からの経過時間である．RTT 間の cwnd の変化は，以下のように，cwnd を T で微分した値に RTT をかけたもので近似できる．

$$\begin{aligned}\Delta\,\mathrm{cwnd} &= RTT \cdot \frac{d(\mathrm{cwnd})}{dT} \\ &= RTT \cdot 3C\left(T - \sqrt[3]{\beta \cdot \frac{\mathrm{cwnd}_{max}}{C}}\right)^2 \end{aligned} \quad (2)$$

ここで cwnd の定義式を用いて，Δcwnd を cwnd の関数で表すと以下のようになる．

$$\Delta\,\mathrm{cwnd} = 3RTT \cdot \sqrt[3]{C}\left(\sqrt[3]{\mathrm{cwnd} - \mathrm{cwnd}_{max}}\right)^2 \quad (3)$$

すなわち，CUBIC TCP の場合，Δcwnd は cwnd の 2/3 乗の関係にあると考えられる．この関数は，cwnd_{max} のときに 0 となり，その値を中心に対称な形をとる．

(3) Hamilton TCP

Hamilton TCP では，ACK セグメントを受信するごとに cwnd を α/cwnd セグメントだけ増加させるとし，α を直前の輻輳事象からの経過時間の関数（以下の式）としている．

$$\alpha = \begin{cases} 1 + 10(T - T_{low}) + 0.5(T - T_{low})^2 & (T \geqq T_{low}) \\ 1 & (T < T_{low}) \end{cases} \quad (4)$$

ここで，T は直前の輻輳事象からの経過時間，T_{low} はしきい値である．すなわち，輻輳事象から T_{low} までは Reno と同様に振る舞い，その後は増加関数が T の 2 次式となる．

Reno と同様に遅延 ACK を考慮すると，$\alpha/2$ が RTT 間の cwnd の増加分となると考えることができる．そこで式 (4) の α の半分の値を Δcwnd とする．式 (4) の $(T \geqq T_{low})$ の部分について考えると以下のようになる．平方完成することにより，次式を得る．

$$\Delta\,\mathrm{cwnd} = \frac{1}{4}(T - T_{low} + 10)^2 - \frac{49}{2} \quad (5)$$

cwnd はこの値を 0 から RTT まで積分することにより得られると近似する．その結果 cwnd は以下の式で与えられる．

$$\mathrm{cwnd} = \frac{2}{3}\left(\sqrt{\Delta\,\mathrm{cwnd} + \frac{49}{2}}\right)^3 - 49\sqrt{\Delta\,\mathrm{cwnd} + \frac{49}{2}} + C \tag{6}$$

ただし C は定数である．この結果は cwnd が Δcwnd の 3/2 乗の関係にあることを示す．したがって，Hamilton TCP での Δcwnd の変化は，Reno と同様な部分の後に，CUBIC TCP と同様に，cwnd の 2/3 乗の部分が続くことになる．

(4) TCP Vegas

TCP Vegas では，RTT を測定することにより，ボトルネックとなるルータでのバッファサイズを，以下の式により予想する．

$$BufferSize = \mathrm{cwnd} \times \frac{RTT - RTT_{min}}{RTT} \tag{7}$$

ここで，RTT_{min} はその TCP コネクションで観測された中で最小の RTT である．この値により，輻輳回避フェーズにおいて，以下のように cwnd の値を変化させる．

$$\mathrm{cwnd} = \begin{cases} \mathrm{cwnd} + 1 & (BufferSize < A) \\ \mathrm{cwnd} & (A \leqq BufferSize \leqq B) \\ \mathrm{cwnd} - 1 & (BufferSize > B) \end{cases} \tag{8}$$

Linux では $A = 2$，$B = 4$（単位 MSS）で実装されている．したがって，輻輳回避フェーズにおいて Δcwnd の値は $\Delta\,\mathrm{cwnd} = 1, 0$ or -1 となる．

(5) TCP Veno

TCP Veno は Vegas と Reno を組み合わせて名付けられたもので，式 (7) の *BufferSize* を用いて輻輳回避フェーズにおける cwnd の増加を調整する．$BufferSize > B$ の場合は（B は Vegas で用いられた B），新たにデータセグメントの確認応答を行う ACK セグメントに対して，1 つおきに 1/cwnd セグメントずつ cwnd を増加させる．そうでない場合は TCP Reno と同様に cwnd を増加させる．このため，遅延 ACK を想定すると，cwnd の値に関係なく $\Delta\,\mathrm{cwnd} = 1$ or 0 となる．これは TCP Reno と同様であるが，$BufferSize > B$ の場合は $\Delta\,\mathrm{cwnd} = 1$ と $\Delta\,\mathrm{cwnd} = 0$ の割合が 1 : 3，$BufferSize \leq B$ の場合は 1 : 1 となる．

以上のように，輻輳制御アルゴリズムごとに異なる $(\mathrm{cwnd}, \Delta\,\mathrm{cwnd})$ の関係を定めることが可能である．なお筆者らの文献 [23] においては，上記に加えて，HS TCP，TCP Westwood+，TCP Illinois に対して検討を行っている．

3.3 提案方法に関する考察

(1) 既存方法との比較

提案方法は，パケットトレースの情報から推定した cwnd の時系列の差分に着目した方法である．その意味では，2 章で紹介した文献 [17] と [18] で示された大塩らの提案との関連性が深いといえる．大塩らの方法では，RTT ごとに cwnd を推定しその時系列を求める．さらに，cwnd の変化分 δ と cwnd の増加率（cwnd の値の比 r）を求め，δ が 1 セグメントである割合（SBS：Step by Step），r の平均値（R：Rate），cwnd が変化しなかった（δ が 0 である）割合（N：Nochange），δ の変動した割合（RC：RateChange）という 4 つの指標を定める．これらの指標に基づいて，クラスタ分析によりアルゴリズムを区別する．Reno の場合は指標 SBS が増加し，Vegas では指標 N が増加するなどにより，区別が可能となるというわけである．この方法では，パケットトレースに含まれる 2 つのアルゴリズムが既知であるという条件の下で，TCP セッションの輻輳制御方式が，どちらのアルゴリズムに従っているかを判定することを目指している．

これに対し提案方法は，輻輳制御アルゴリズムの原理から，cwnd の変化分 Δcwnd（δ に相当）と cwnd の関係を明確化し，パケットトレースにおけるパケットロスが発生していない部分を評価し，直接的にアルゴリズムを推定しようとしている．この結果，提案方式では，パケットトレース中の個々の TCP セッションに対して，その輻輳制御アルゴリズム自身を特定することが可能であると考えられる．

(2) フロー制御用ウィンドウの影響

TCP のデータ転送は，cwnd を用いた輻輳制御と，受信側のバッファサイズに対応する広告ウィンドウサイズ（Advertised Window Size：awnd）に従うフロー制御により管理される．提案方法では，パケットトレースのデータ転送が cwnd により制御されており，awnd は cwnd よりも大きいと仮定している．実際 TCP では Dynamic Right-Sizing [25] と呼ばれる手順が採用され，awnd が cwnd の 2 倍程度の大きさとなるように受信バッファを割り当てる処理が行われている．このため，多くの場合は上記の仮定は妥当であると考えられる．しかし，受信側ノードの処理能力が低いなどの理由で，awnd が減少する場合があり，その場合は正しい推定ができないことになる．

4. 双方向パケットトレースへの適用方法と結果

本章では，双方向のパケットトレースに提案手法を適用する方法とその結果について述べる．

4.1 cwnd の推定

本論文で提案する方法では RTT ごとに以下の式より cwnd の推定を行うこととする．

$$\mathrm{cwnd}_i = \left\lfloor \frac{\mathrm{seq}_i - ack_i}{MSS} \right\rfloor \quad [\mathrm{segments}] \tag{9}$$

ここで，cwnd_i は i 回目の RTT に対応する cwnd，ack_i は

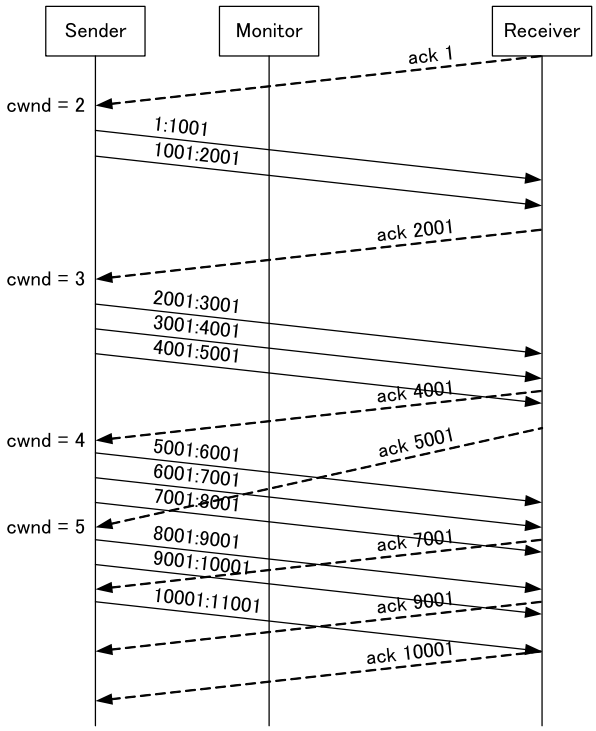


図 1 RTT に関連する cwnd の推定
Fig. 1 Estimation of cwnd associated with one RTT.

i 回目の RTT の途中での最後の ACK セグメント（またはその RTT の直前の ACK セグメント）の確認応答番号，seq_i は $i+1$ 回目の RTT の最初のデータセグメントのシーケンス番号（すなわち i 回目の RTT の中で送られたバイトのシーケンス番号 +1），*MSS* はその TCP コネクションで使用される MSS である．また記号 $\lfloor x \rfloor$ は x の整数部分を示す．

さらに，パッシブ方式ではネットワークの途中に位置したモニタ装置によりパケットトレースを収集するのが一般的であるため，トレース上で測定される RTT は実際の送信ノードにおける RTT とは異なる．このため，タイムスタンプオプションを用いて，次のような方法で cwnd を推定することを想定する．

- 再送データに対する ACK セグメントなど，特定の ACK セグメントに着目する．
- 着目した ACK セグメントの TSval（Timestamp Value）の値を TSecr（Timestamp Echo Reply）に持つデータセグメントを探す．
- そのデータセグメントを最初に確認応答する ACK セグメントを探す．
- 第 2 の ACK セグメントの TSval の値を TSecr に持つデータセグメントを探す．

これらの結果，第 2 のデータセグメントのシーケンス番号を seq_i とし，第 2 の ACK セグメントの 1 つ前の ACK セグメントの確認応答番号を ack_i とする．

具体的な例を**図 1** に示す．ここでは説明を容易にするためにコネクション確立直後のスロースタートフェーズを用いる．MSS は 1,000 バイトとし，データセグメントは実線，ACK セグメントは点線で示す．データセグメントには，ヘッダ中のシーケンス番号とそのセグメントの最後のバイトに付与された番号＋ 1 の値を示す．また ACK セグメントには応答確認番号を示す．

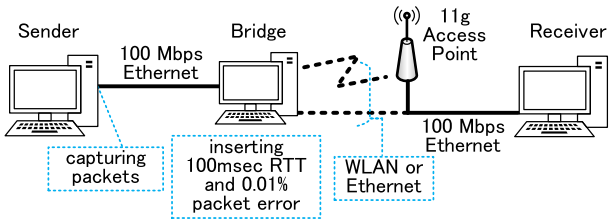


図 2 実験ネットワーク構成
Fig. 2 Experiment network configuration.

モニタで収集されたトレースから ack1 に着目する．次にその ACK セグメントの TSval を TSecr に持つデータセグメント 1 : 1001 に着目し，これを ACK する ack2001 を識別する．その TSval を TSecr に持つデータセグメント 2001 : 3001 を識別する．Ack2001 の前の ACK セグメントは ack1 であるため，1 : 1001 のデータセグメントから始まる RTT の間の cwnd は，$(2001-1)/1000=2$ から 2 セグメントと判断する．

同様に，次の 2 回の RTT に対しては，$(5001-2001)/1000=3$，$(10001-5001)/1000=5$ が求められる．

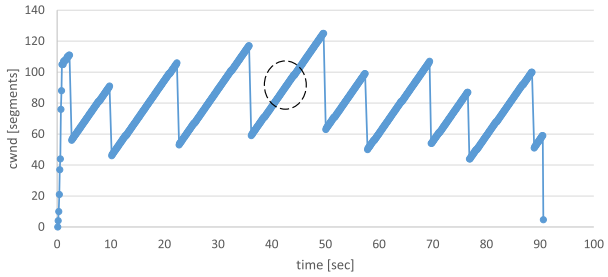
4.2 実験環境

提案手法を評価するために，**図 2** に示すようなネットワーク構成でパケットトレースを収集した．3 台の Linux PC を TCP 送信者，受信者，ブリッジとして使用した．TCP 送信者とブリッジとは，100 Mbps イーサネットで接続し，ブリッジと受信者は，イーサネットまたは IEEE802.11g 無線 LAN のいずれかで接続した．送信側において，輻輳制御アルゴリズムを切り替えた．またブリッジにより往復 100 ミリ秒（片方向 50 ミリ秒ずつ）の遅延および 0.01%のパケット損失を発生させた．パケットトレースの収集は，送信者の出力ポートで行った．このため送信者とモニタの間の遅延はほとんどないが，前節に述べた方法で cwnd を推定している．また，ブリッジと受信者を結ぶリンクについては，イーサネットと無線 LAN のそれぞれで実験を行い，トレースを収集した．下記に示す適応結果では，輻輳制御アルゴリズムごとに，2 種類のトレースのうちから特徴的な結果が得られたものを用いて論ずることとする．

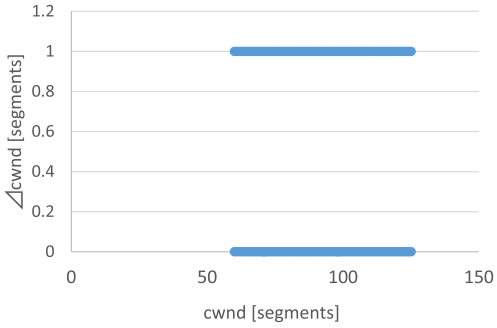
4.3 個別の輻輳制御アルゴリズムへの適用結果

(1) TCP Reno

TCP Reno の実験結果を**図 3** に示す．使用したトレースはブリッジと受信側の接続にイーサネットを使用したも

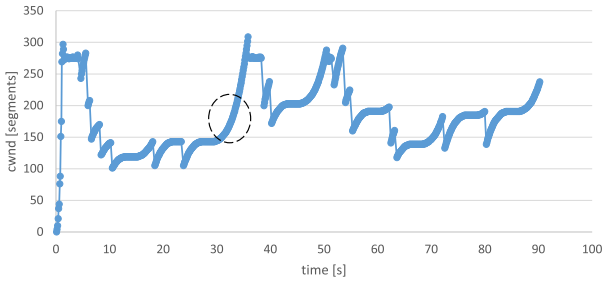


(a) 推定された cwnd の時間変化

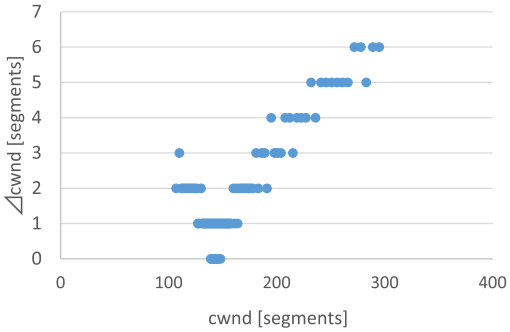


(b) ⊿cwnd と cwnd の対応（(a)の丸印の部分）

図 3 TCP Reno の結果（双方向パケットトレース）
Fig. 3 Results of TCP Reno with bidirectional trace.



(a) 推定された cwnd の時間変化



(b) ⊿cwnd と cwnd の対応（(a)の丸印の部分）

図 4 CUBIC TCP の結果（双方向パケットトレース）
Fig. 4 Results of CUBIC TCP with bidirectional trace.

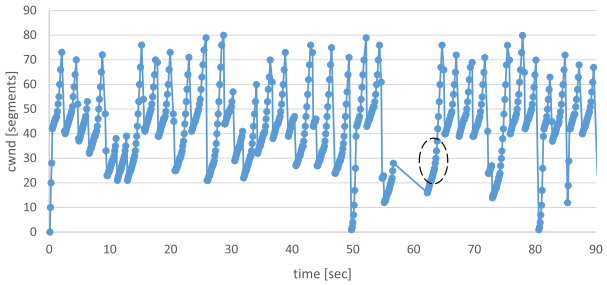
のである．トレースに含まれるデータセグメントと ACK セグメントのヘッダ中のシーケンス番号と応答確認番号から，cwnd の値を推定した結果を図 3 (a) に示す．このうちで，丸印で示した，cwnd が増加し続ける部分に対して，(cwnd, Δcwnd) の関係をグラフしたものが図 3 (b) である．前章で示したとおり，Δcwnd が 0 と 1 を交互にとっていることが分かる．図 3 (b) の他の部分に対しても同様な結果を得た．

(2) CUBIC TCP

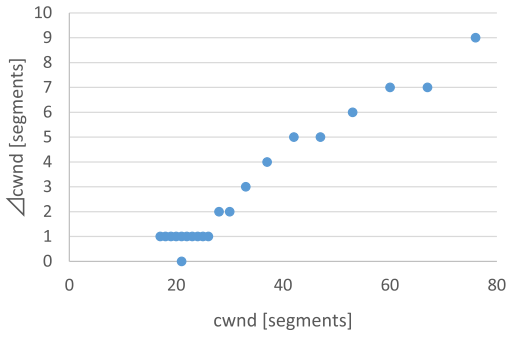
CUBIC TCP の実験結果を**図 4** に示す．使用したトレースはイーサネット接続を使用したものである．図 4 (a) がトレースの情報から推定した cwnd の時間的変化である．図 4 (b) が丸印で示した部分に対する (cwnd, Δcwnd) のグラフである．この結果は，式 (3) に対応していると判断できる．すなわち，cwnd = 150 付近で Δcwnd が 0 となり，それを中心に Δcwnd が対称なグラフを示している．また Δcwnd のグラフは上に凸の形になっている．(cwnd, Δcwnd) のグラフがこのような形をとる場合は，CUBIC TCP が使用されていると判断できる．

(3) Hamilton TCP

Hamilton TCP の実験結果を**図 5** に示す．使用したトレースはブリッジと受信側を無線 LAN で接続した場合のものである．図 5 (a) がトレースの情報から推定した cwnd の時間的変化で，図 5 (b) が図の (a) の丸印の部分に対する (cwnd, Δcwnd) のグラフである．この結果では，cwnd が 17 から 26 セグメントの間は，Δcwnd が 1 または 0 となっており，それ以降は緩やかに増加している．増加部分が cwnd の 2/3 乗であるかどうかは判断できないが，最初は Reno と同様で，その後増加するという前章で述べた Hamilton TCP の特徴に対応していると考えられる．

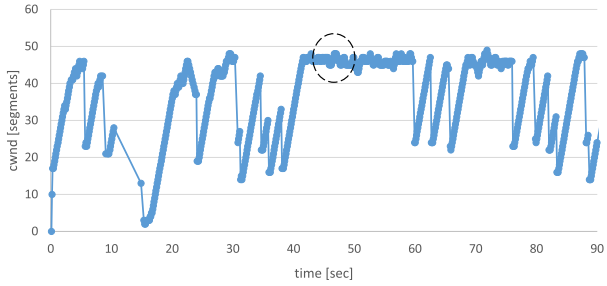


(a) 推定された cwnd の時間変化

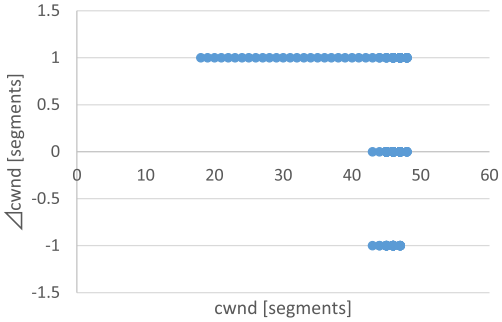


(b) ⊿cwnd と cwnd の対応（(a)の丸印の部分）

図 5 Hamilton TCP の結果（双方向パケットトレース）
Fig. 5 Results of Hamilton TCP with bidirectional trace.

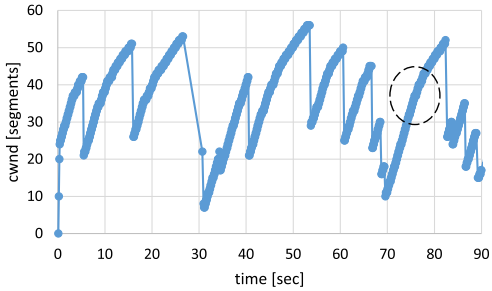


(a) 推定された cwnd の時間変化

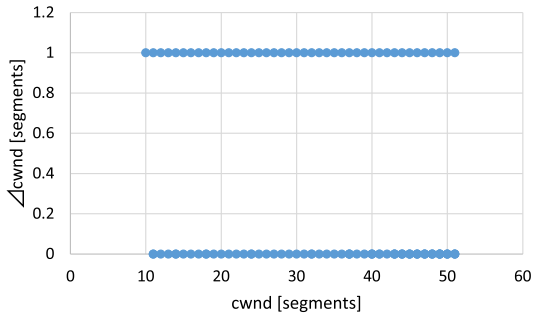


(b) ⊿cwnd と cwnd の対応（(a)の丸印の部分）

図 6 TCP Vegas の結果（双方向パケットトレース）
Fig. 6 Results of Hamilton TCP with bidirectional trace.



(a) 推定された cwnd の時間変化



(b) ⊿cwnd と cwnd の対応（(a)の丸印の部分）

図 7 TCP Veno の結果（双方向パケットトレース）
Fig. 7 Results of TCP Veno with bidirectional trace.

(4) TCP Vegas

TCP Reno の実験結果を**図 6** に示す．使用したトレースは無線 LAN による接続を用いたものである．図の (a) が cwnd の時間的変化，(b) が $(\mathrm{cwnd}, \Delta\mathrm{cwnd})$ のグラフである．(b) の結果では，$\Delta\mathrm{cwnd} = 1, 0, -1$ が混在しているため，前章で示したとおりの結果となった．ただし，この結果は，図 6 (b) において cwnd が一定の値にとどまっている部分に対して調査を行ったために得られたものである．cwnd が単に増加しているだけの部分を選んだ場合は，TCP Reno と同様に，Δcwnd が 1 と 0 を交互にとるという結果しか得られない．このため，TCP 通信によっては Reno と区別できない場合があることに注意する必要がある．

(5) TCP Veno

TCP Reno の実験結果を**図 7** に示す．使用したトレースは無線 LAN 接続を用いたものである．図の (a) の丸印に対応する $(\mathrm{cwnd}, \Delta\mathrm{cwnd})$ のグラフを図の (b) に示している．この結果は図 3 (c) の TCP Reno の結果と同様であるが，cwnd が 39 セグメントよりも小さい範囲では Δcwnd が 1 と 0 の割合が 1 : 1 で，それ以降では 1 : 3 となっている．このため，全体を通じてその割合が 1 : 1 となっている TCP Reno と区別することができる．

以上のように，双方向のパケットトレースに基づいて，輻輳制御アルゴリズムを識別する手法の評価を行った．3 章で述べたように，$(\mathrm{cwnd}, \Delta\mathrm{cwnd})$ のグラフを観測することにより，アルゴリズムごとの特徴を見い出すことが可能であることを示した．

5. 片方向パケットトレースへの適用方法と結果

次に，片方向のパケットトレース（データセグメントのキャプチャのみを含む）に提案手法を適用する方法とその結果について述べる．

5.1 これまでの検討結果

この課題に対して筆者らはこれまでに 2 つの検討を行った．1 つは，ある時間間隔ごとに，その間に転送されたバイト数を cwnd と見なし，その値の列に対して 3 章で示した手法を適用するものである [24]．これを 4 章の実験で収集した双方向のパケットトレースからデータセグメントのみを抜き出した片方向トレースに適用した．その際の時間間隔は，ブリッジで挿入した往復 100 ミリ秒の遅延と同じとした．その結果，TCP Reno などイーサネット経由のトレースを用いた場合は比較的良い結果が出たが，TCP Vegas などに無線 LAN 経由のトレースを用いた場合については Δcwnd と cwnd の関係がうまく導出できなかった．その理由は，前者では実際の RTT が 100 ミリ秒でほぼ安定していたのに対し，後者では 100 から 140 ミリ秒の間でばらつきがあったためである．

そこで，ピリオドグラム法 [26] を用いて，上記の片方向パケットトレースから RTT を推定することを試みた [27]．その結果，ある程度の推定は可能なものの，100 ミリ秒程

度の RTT に対して数十ミリ秒の推定誤差が生じた．推定結果は，イーサネット経由のトレース（実際の RTT は安定している）ではばらつきが大きく，逆に無線 LAN 経由のトレース（実際の RTT はばらついている）ではほぼ 100 ミリ秒に近くで安定した値を推定してしまった．いずれにせよ，片方向トレースから RTT を正確に推定することは困難であるという結論を得た．

5.2 アプローチ

そもそも，cwnd は時間 vs. シーケンス番号のグラフを時間で微分したようなものである．さらに筆者らの用いる Δcwnd は時間 vs. cwnd のグラフを微分したようなものである．このため，もともとの時間 vs. シーケンス番号のグラフが離散的に定義されており，かつなめらかでない場合は，そのグラフの微分係数を計算する際に大きな誤差が生ずる．

一方シーケンス番号の時間変化について以下の考察が得られる．TCP Reno では cwnd の変化量が定数であるため，cwnd は時間の一次式，シーケンス番号は時間の 2 次式となる．また CUBIC TCP では cwnd が時間の 3 次関数であるため，シーケンス番号は時間の 4 次式となると予想できる．このように，既存の輻輳制御アルゴリズムでは，シーケンス番号の時間変化を時間の多項式で近似できると考えられる．

そこで，片方向トレースから輻輳制御アルゴリズムを推定する手法として，以下の方法を採用することとした．

- シーケンス番号の時間変化に対して，再送が行われていない範囲（シーケンス番号が単調増加している範囲）を選ぶ．
- その範囲のデータを，時間の 1 次から 4 次の範囲の関数として，最小二乗法を用いて近似する．
- その際，途中から近似関数が変更される場合も考慮する．具体的には，一定の時間（本論文では 0.5 秒）の刻みで，近似関数の境界を仮定し，その前後で異なる関数を用いることにより，より誤差の少ない近似ができるかどうかを確認する．
- 得られた近似関数を時間で微分した関数を Δseq，さらにそれを再度時間で微分した関数を Δ^2seq とする．
- Δ^2seq を Δseq の関数としてグラフ化する．Δseq と Δ^2seq が 3 章の手法における cwnd と Δcwnd に対応すると想定し，その形態から 3.2 節と同様な方法で，輻輳制御アルゴリズムを推定する．具体的な方法は以下のとおりである．
  - TCP Reno：RTT ごとに推定された Δcwnd では，0 また 1 セグメントとなったが，片方向トレースからの識別では，それらが平均化され，一定値となる．Δ^2seq の値は一定値をとるものの，3.2 節 (1) に示したように MSS の値に相当するわけではない．これは片方向トレースに対する方法が，RTT を考慮していないためである．
  - CUBIC TCP：Δ^2seq の値は特定の Δseq の値で 0 となり，その値を中心として線対称な形態をとる．また Δ^2seq は Δseq の 2/3 乗の関係になる．
  - Hamilton TCP：2 種類の関数に分かれて近似されると予想される．前半は Reno と同様に Δ^2seq が一定値をとり，後半は Δ^2seq が Δseq の 2/3 乗で増加する部分となる．
  - TCP Vegas：Δ^2seq は Δseq に対して一定値となる．ただし，輻輳の発生状況により異なる関数で近似されると考えられ，Δ^2seq が異なる値を持つ．特に cwnd が変更されない状況では，Δ^2seq が 0 となる．
  - TCP Veno：Δ^2seq は Δseq に対して一定値となるが，ボトルネックルータにおけるバッファサイズが大きい場合，そうでない場合に対して，Δ^2seq が半分となると考えられる．

5.3 個別の輻輳制御アルゴリズムへの適用結果

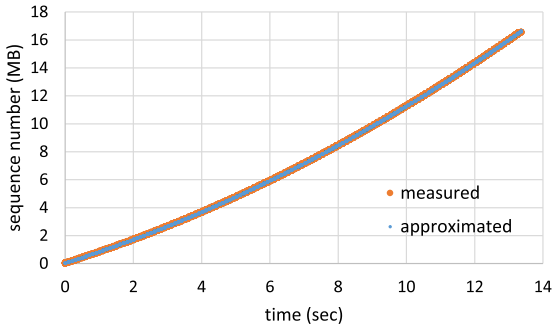
4.3 節で使用した双方向のパケットトレースから，データセグメントに関する部分だけを取り出し，片方向パケットトレースを導出した．これに対して，前節で示した手法を適用した結果を示す．なお，対象とするシーケンス番号の範囲は，各アルゴリズムについて，4.3 節で (cwnd, Δcwnd) のグラフを計算した範囲とする．すなわち双方向トレースに対する手法と同じ範囲を用いて片方向トレースに対する手法を評価する．

(1) TCP Reno

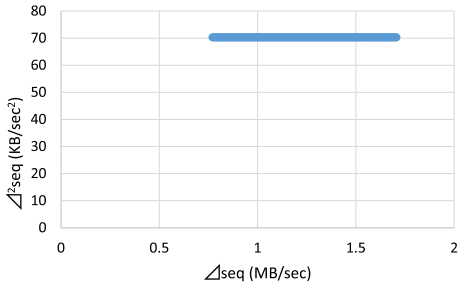
図 8 に TCP Reno の実験結果を示す．この実験では，図 3 (a) に示した cwnd の時間変化において，丸印を付けた部分に対応するシーケンス番号の時間変化を対象として，多項式近似を行った．図 8 (a) はその結果を示す．この図ではオレンジのグラフがトレースからのシーケンス番号の時間変化，青のグラフがそれに対する多項式近似である．なお時間およびシーケンス番号の値は，0 からの相対値としている．この例では 2 次式による近似が最適となっており，図に示すように類似性の高い近似が可能となっている．

近似した多項式から，その一階微分（Δseq）と二階微分（Δ^2seq）の対応を示したものを図 8 (b) に示す．結果は Δ^2seq が Δseq によらず一定値となっており，前節で示した TCP Reno の特徴を示している．このように Δ^2seq が一定値をとる場合は，その TCP フローが Reno を採用している可能性が高いと判断できる．

なお，この手法は RTT の推定値を用いていないことに注意する必要がある．また，Δ^2seq の値が 70 K バイト/秒^2 となっているが，RTT が 0.1 秒程度であることを考えると RTT 間のシーケンス番号の増加量は約 700 バイトとなる．これは 1 MSS の半分に近い値であり，したがって，RTT

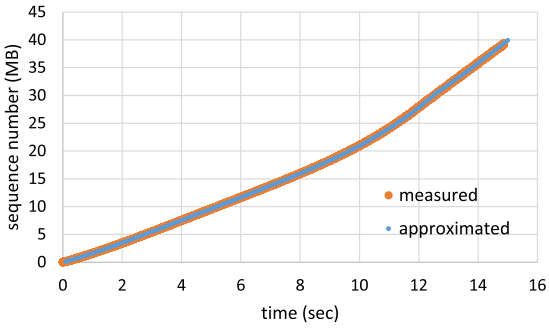


(a) シーケンス番号の多項式近似

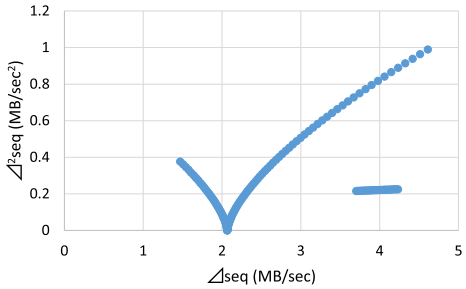


(b) Δ^2seq と Δseq の対応

図 8 TCP Reno の結果（片方向パケットトレース）
Fig. 8 Results of TCP Reno with unidirectional trace.



(a) シーケンス番号の多項式近似



(b) Δ^2seq と Δseq の対応

図 9 CUBIC TCP の結果（片方向パケットトレース）
Fig. 9 Results of CUBIC TCP with unidirectional trace.

が推定できればこの方法で cwnd の推定も可能となることを示している．

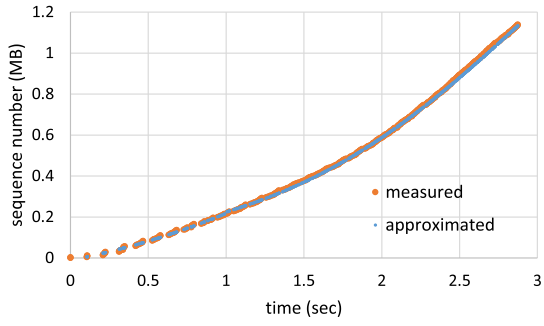
(2) CUBIC TCP

図 9 に CUBIC TCP の実験結果を示す．この実験でも図 4 (a) の丸印に対応するシーケンス番号の時間変化を対象とした．図 9 (a) が多項式近似の結果である．この場合は 12.5 秒までが 4 次式で，それ以降が 3 次式で近似された．その結果を微分することにより得られた Δseq と Δ^2seq の対応関係を図 9 (b) に示す．この結果は 2 つの部分から構成される．1 つは，Δseq が 2 M バイト/秒付近で 0 となり，それを中心に線対称となるグラフである．これは 4 次多項式で近似されたシーケンス番号の変化に対応するもので，3.2 節 (2) で述べた CUBIC TCP の特徴を反映している．ほかは，Δ^2seq の値が約 0.2 M バイト/秒^2 でほぼ一定となっているもので，これは 3 次多項式で近似された部分に対応する．後者については，実際の通信で cwnd が増加した結果，RTT の間に cwnd のすべてが送信できなくなった部分に相当する．すなわち，片方向トレースによる手法では，cwnd が大きい場合に正しく輻輳制御アルゴリズムの振舞いを推定できない場合があることを示している．いずれにせよ，前者のグラフにより，この TCP フローが CUBIC TCP を採用していることは推定可能である．

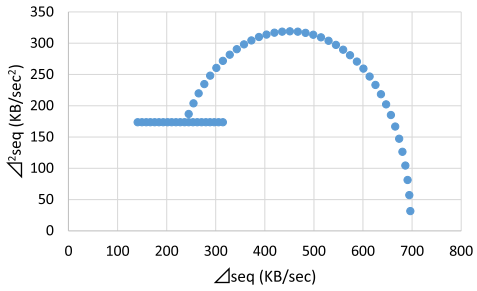
(3) Hamilton TCP

図 10 に Hamilton TCP の実験結果を示す．図 10 (a) がシーケンス番号の多項式近似の結果である．この場合も 2 つの関数で近似され，1 秒までが 2 次式で，それ以降が 4 次式で近似された．図に示すように Hamilton TCP に対しても高い類似度で近似することができている．

その結果を微分することにより，図 10 (b) に示すような Δseq と Δ^2seq の対応関係が得られた．Δseq が 120 から 310 K バイト/秒の範囲で，Δ^2seq が一定となっている部分

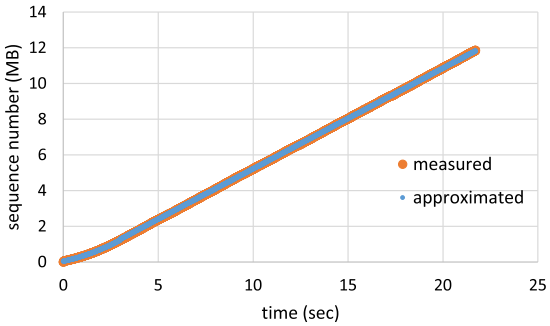


(a) シーケンス番号の多項式近似

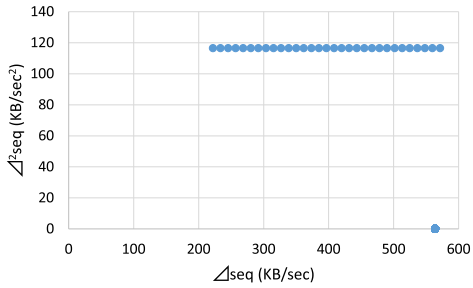


(b) Δ^2seq と Δseq の対応

図 10 Hamilton TCP の結果（片方向パケットトレース）
Fig. 10 Results of Hamilton TCP with unidirectional trace.

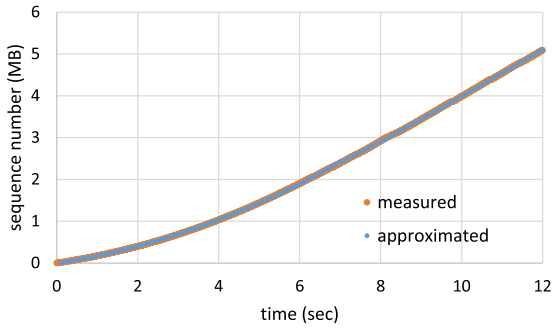


(a) シーケンス番号の多項式近似

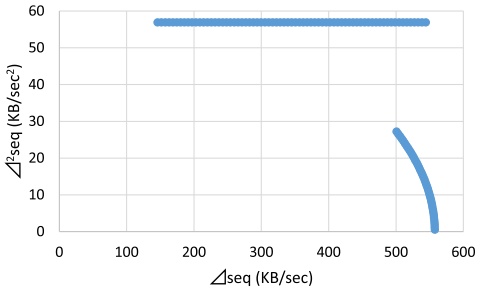


(b) Δ^2seq と Δseq の対応

図 11 TCP Vegas の結果（片方向パケットトレース）
Fig. 11 Results of TCP Vegas with unidirectional trace.



(a) シーケンス番号の多項式近似



(b) Δ^2seq と Δseq の対応

図 12 TCP Veno の結果（片方向パケットトレース）
Fig. 12 Results of TCP Veno with unidirectional trace.

は，1 秒までの 2 次式で近似された部分に対応し，Reno に対応する動作を行う部分である．また上に凸の曲線になった部分は，1 秒以降の 4 次式で近似された部分に対応する．本来はこの部分では cwnd が増加し続けるため，Δ^2seq が減少するグラフにはならないはずである．ここでも cwnd の増加にともない，片方向のトレースではその増加を正しく推定できていないと考えられる．いずれにせよ，Δ^2seq が一定値と増加部分を持つことから Hamilton TCP と推定することは可能である．

(4) TCP Vegas

TCP Vegas の実験結果を**図 11** に示す．図 11 (a) がシーケンス番号の時間変化の多項式近似で，3 秒までが 2 次式，それ以降が 1 次式で近似される．この場合も類似性は高い．その結果から求めた Δseq と Δ^2seq の対応関係を図 11 (b) に示す．

3 秒以降は 1 次関数で近似されているため，その 2 階微分は 0 となる．それが，Δseq が 560 K バイト/秒付近で Δ^2seq が 0 となっている点である．図 6 に示した双方向トレースの実験では，RTT ごとの推定が可能であったため，Δcwnd が 1，0，−1 セグメントに分散していた．この部分が片方向トレースでは 1 つにまとまった結果となっている．

しかしこの結果は他の状況でも起こりうる．たとえば，TCP Reno において cwnd が広告ウィンドウよりも大きくなった場合も同様な結果が得られる．このため，図 11 (b) のようなグラフが得られた場合は，TCP Vegas の可能性があると判断できるのみである．

(5) TCP Veno

TCP Veno の実験結果を**図 12** に示す．図 12 (a) のシーケンス番号の時間変化については，7 秒前までと 7 秒後で異なる関数で近似されている．前者が 2 次式，後者が 3 次式となった．近似自身は高い類似性を保っている．

図 12 (b) が近似された関数を微分することにより得られた Δseq と Δ^2seq の対応関係である．Δseq が小さいうちは Δ^2seq が一定値を持つ．これは TCP Reno と同様である．しかし，Δseq が 500 K バイト/秒より大きい範囲では，Δ^2seq が減少するグラフとなった．本来はこの部分は，Δ^2seq が前者の半分の値で一定値をとるはずである．Δseq が 500 K バイト/秒の時点では半分の値であるが，その後は Δ^2seq の値が減少している．これも CUBIC TCP や Hamilton TCP と同様に，cwnd の増加を正しく推定できていないためと考えられる．

また，図 12 (b) の結果は，cwnd が増加した場合にその増加の割合が減少したことを意味しているので，TCP Vegas でもこのような形をとりうる．したがってこの結果からは Vegas か Veno の可能性があると考えられる．

6. おわりに

本論文では，双方向および片方向（データセグメントのみ）のパケットトレースから，TCP フローごとの輻輳制御アルゴリズムを推定する手法を示した．基本手法は，RTT ごとの輻輳ウィンドウ（cwnd）とその増加分（Δcwnd）を

計算し，(cwnd, Δcwnd) のグラフの形状から判断するというものである．双方向のトレースに対しては，TCP Reno，CUBIC TCP，Hamilton TCP，TCP Vegas，TCP Veno に対して，識別可能であることを示した．

一方，片方向トレースでは，RTT を厳密に求めることが困難なことから，時間 vs. シーケンス番号のグラフを，1 次関数から 4 次関数の範囲で多項式近似し，そのグラフの 1 階微分と 2 階微分（Δseq と Δ^2seq）の対応から，輻輳制御アルゴリズムを識別する手法を考案した．この手法を，双方向トレースからデータセグメントの情報を抜き出して作成した片方向トレースに適応した結果を示した．(Δseq, Δ^2seq) のグラフは (cwnd, Δcwnd) と同様な傾向を示した．このためある程度の推定は可能であることを明らかにした．しかし，TCP Reno，TCP Vegas，TCP Veno といった cwnd の増加が一定となる場合は，区別が困難な場合もあることが明らかとなった．

提案した手法は，cwnd に相当するデータが転送されることを前提としている．しかし，広告ウィンドウが小さい，多数の TCP トラヒックが存在し個々のフローのデータ転送量が制限される，アプリケーションがデータ転送量を制限しているなどの理由で，この条件が満足されない場合がある．また上記のように，片方向トレースからの推定についてはうまく推定できない場合もある．このためより正確に推定するためには，さらに検討が必要である．たとえば，機械学習による輻輳制御アルゴリズムの推定などが考えられる．

謝辞 本研究は JSPS 科研費 JP16K06341 の助成を受けたものである．

参考文献

[1] Javobson, V.: Congestion Avoidance and Control, *ACM SIGCOMM Comp. Commun. Review*, Vol.18, No.4, pp.314–329 (1988).

[2] Floyd, S., Henderson, T. and Gurtov, A.: The NewReno Modification to TCP's Fast Recovery Algorithm, IETF RFC 3728 (2004).

[3] Afanasyev, A. et al.: Host-to-Host Congestion Control for TCP, *IEEE Commun. Surveys & Tutorials*, Vol.12, No.3, pp.304–342 (2010).

[4] Floyd, S.: HighSpeed TCP for Large Congestion Windows, IETF RFC 3649 (2003).

[5] Ha, S., Rhee, I. and Xu, L.: CUBIC: A New TCP-Friendly High-Speed TCP Variant, *ACM SIGOPS Operating Systems Review*, Vol.42, No.5, pp.64–74 (2008).

[6] Kelly, T.: Scalable TCP: Improving performance in high-speed wide area networks, *ACM Comp. Commun. Review*, Vol.32, No.2, pp.83–91 (2003).

[7] Leith, D. and Shorten, R.: H-TCP: TCP for high-speed and long distance networks, *Proc. Protocols for Fast Long Distance Networks* (*PFLDnet*), pp.1–16 (Mar. 2004).

[8] Mascolo, S. et al.: TCP Westwood: Bandwidth estimation for enhanced transport over wireless links, *Proc. ACM MobiCom '01*, pp.287–297 (July 2001).

[9] Brakmo, L. and Perterson, L.: TCP Vegas: End to End Congestion Avoidance on a Global Internet, *IEEE J. Selected Areas in Commun.*, Vol.13, No.8, pp.1465–1480 (1995).

[10] Fu, C. and Liew, S.: TCP Veno: TCP Enhancement for Transmission Over Wireless Access Networks, *IEEE J. Selected Areas in Commun.*, Vol.21, No.2, pp.216–228 (2003).

[11] Liu, S., Basar, T. and Srikant, R.: TCP-Illinois: A loss- and delay-based congestion control algorithm for high-speed networks, *Performance Evaluation*, Vol.65, No.6-7, pp.417–440 (2008).

[12] Tan, K. et al.: A Compound TCP Approach for High-speed and Long Distance Networks, *Proc. INFOCOM 2006*, pp.1–12 (Apr. 2006).

[13] Paxson, V.: Automated Packet Trace Analysis of TCP Implementations, *ACM Comp. Commun. Review*, Vol.27, No.4, pp.167–179 (1997).

[14] Kato, T. et al.: Design of Protocol Monitor Emulating Behaviors of TCP/IP Protocols, *Proc. 10th Int. Workshop on Testing of Communicating Systems* (*IWTCS '97*), pp.416–431 (Sep. 1997).

[15] Jaiswel, S. et al.: Inferring TCP Connection Characteristics Through Passive Measurements, *Proc. INFOCOM 2004*, pp.1582–1592 (Mar. 2004).

[16] 茂木重憲，渡邊 晶：輻輳制御パラメータのリアルタイム推定，情報処理学会論文誌，Vol.52, No.3, pp.1308–1322 (2011).

[17] 大塩純平，阿多信吾，岡 育生：クラスター分析に基づく各種 TCP バージョンの識別手法，信学技報，Vol.108, No.457, NS2008-179, pp.205–210 (2009).

[18] Oshio, J., Ata, S. and Oka, I.: Identification of Different TCP Versions Based on Cluster Analysis, *Proc. 18th Int. Conf. Computer Communications and Networks* (*ICCCN 2009*), pp.1–6 (Aug. 2009).

[19] Qian, F., Gerber, A. and Mao, Z.: TCP Revisited: A Fresh Look at TCP in the Wild, *Proc. Internet Measurement Conference 2009* (*IMC '09*), pp.76–89 (Nov. 2009).

[20] Yang, P. et al.: TCP Congestion Avoidance Algorithm Identification, *Proc. 2011 International Conference on Distributed Computing Systems* (*ICDCS '11*), pp.310–321 (June 2011).

[21] Kato, T. et al.: Inferring TCP Congestion Control Algorithms by Correlating Congestion Window Sizes and Their Differences, *Proc. 9th Int. Conf. Systems and Networks Communications* (*ICSNC 2014*), pp.42–47 (Oct. 2014).

[22] 小田 淳，策力木格，大坐畠智，加藤聰彦：輻輳ウィンドウとその増加値の対応関係と減少パラメータの推定による TCP 輻輳制御方式の特徴付け，信学技報，Vol.114, No.209, CQ2014-69, pp.171–176 (2014).

[23] Kato, T. et al.: Comparing TCP Congestion Control Algorithms Based on Passively Collected Packet Traces, *Proc. 10th Int. Conf. Systems and Networks Communications* (*ICSNC 2015*), pp.135–141 (Nov. 2015).

[24] Kato, T. et al.: A Study on How to Characterize TCP Congestion Control Algorithms from Unidirectional Packet Traces, *Proc. 11th Int. Conf. on Internet Monitoring and Protection* (*ICIMP 2016*), pp.23–28 (May 2016).

[25] Fisk, M. and Feng, W.: Dynamic Right-Sizing, *Proc. Los Alamos Computer Science Institute Symposium*, pp.1–7 (Oct. 2001).

[26] Carra, D. et al.: Passive Online RTT Estimation

for Flow-Aware Routers Using One-Way Traffic, *NETWORKING 2010*, LNCS6091, pp.109–121 (May 2010).
[27] Kato, T. et al.: Applying Lomb Periodogram to Round-trip Time Estimation from Unidirectional Packet Traces with Different TCP Congestion Controls, *Proc. 13th Int. Conf. Internet Monitoring and Protection* (*ICIMP 2018*), pp.1–6 (July 2018).

加藤 聰彦 （正会員）

昭和 53 年東京大学工学部電気工学科卒業．昭和 58 年同大学大学院博士課程修了．同年国際電信電話（現，KDDI）（株）入社．平成 14 年まで同社研究所および（株）KDDI 研究所において，OSI，ATM，インターネット等の研究に従事．平成 14 年より電気通信大学大学院情報システム学研究科助教授．現在，同大学教授．インターネットやアドホックネットワーク等の通信プロトコルの研究に従事．工学博士．

小田 淳

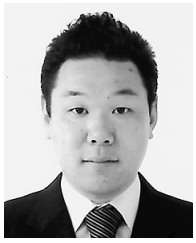
平成 25 年電気通信大学電気通信学部卒業，平成 27 年同大学大学院情報システム学研究科博士前期課程修了．同年日本電気通信システム（株）入社．電気通信大学大学院在学中に，双方向のパケットトレースからの TCP 輻輳制御アルゴリズムの推定手法について研究を進める．

厳 笑凡

平成 30 年電気通信大学大学院情報理工学研究科博士前期課程修了．同年 SMK（株）入社．電気通信大学大学院在学中に片方向のパケットトレースからの TCP 輻輳制御アルゴリズムの推定手法について研究を進める．

山本 嶺

平成 19 年芝浦工業大学システム工学部電子情報システム工学科卒業，平成 21 年同大学大学院電気電子情報工学専攻修士課程修了．平成 25 年早稲田大学大学院博士課程修了．平成 22 年～平成 26 年同大学国際情報通信研究科助手を経て，現在，電気通信大学大学院情報理工学研究科助教，博士（国際情報通信学）．主に無線アドホック・センサネットワークに関する研究に従事．平成 26 年～早稲田大学国際情報通信研究センター招聘研究員．平成 22 年電子情報通信学会学術奨励賞受賞．



大坐畠 智 （正会員）

平成 15 年筑波大学大学院博士課程工学研究科電子・情報工学専攻修了，同年東京農工大学工学部情報コミュニケーション工学科助手，平成 19 年同大学助教，平成 21 年電気通信大学大学院情報システム学研究科准教授．ネットワーク性能評価の研究に従事．平成 17 年度電子情報通信学会学術奨励賞受賞，IEEE，ACM，IEICE 各会員．博士（工学）．