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.

エッジルータにおける改造 TCP の検出・制御手法の提案

丸山　純一† 　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: †{j-maruyama,murata}@ist.osaka-u.ac.jp, ††hasegawa@cmc.osaka-u.ac.jp

あらまし　本稿では，エッジルータにおいて利己的な挙動をする tampered-TCP コネクションを検出・制御することによって，通常の TCP コネクションを保護し，TCP コネクション間の公平性を実現する手法を提案する．提案手法は，エッジルータにおいて TCP パケットを観測することにより，TCP コネクションのウィンドウサイズ，あるいはスループットを推測する．さらに，その値をもとに TCP コネクションのタンパリング性を判断し，必要に応じてパケットを意図的に廃棄する．シミュレーション評価により検証した結果，提案手法によって tampered-TCP コネクションを高い確率で検出し，TCP Reno コネクションとのスループット比をほぼ 1 に保つことができることを示す．
キーワード　TCP，改造 TCP，エッジルータ，ウィンドウサイズ，ネットワーク監視，公平性

Detection and Control Mechanisms of Tampered-TCP at Edge Routers

Junichi MARUYAMA†, 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: †{j-maruyama,murata}@ist.osaka-u.ac.jp, ††hasegawa@cmc.osaka-u.ac.jp

Abstract　In this report, we propose a new mechanism which detects tampered-TCP connections and keeps the fairness among TCP connections at edge routers. The proposed mechanism monitors TCP packets at an edge router and estimates the window size or throughput of each TCP connection. By using estimation results, the proposed mechanism judges whether each connection is tampered or not and drops packets intentionally if necessary to improve the fairness among connections. From the results of simulation experiments, we exhibit that the proposed mechanism can identify tampered-TCP connections at high probability and regulate throughput ratio between tampered-TCP connections and competing TCP Reno connections to around 1.
Key words　TCP, Tampered-TCP, Edge router, Window Size, Network Monitoring, Fairness

1. はじめに

Transmission Control Protocol (TCP) [1] はエンド端末で動作するプロトコルであるため，特に Linux などのオープンソースのオペレーティングシステムにおいては，その挙動を容易に変更することが可能である．そのため、現在のインターネットには, [2, 3] に示されるように悪意のあるユーザによって改造された TCP が存在する．このような改造 TCP を，本稿では tampered-TCP と呼ぶ．

一般に，TCP の輻輳制御アルゴリズムに対する変更を提案する場合，既存の TCP と提案する TCP が共存する環境における性能を確認することで，既存環境との親和性が評価される [4–7]．しかし，悪意あるユーザは既存の TCP より高いスループットを獲得することだけを考え，利己的に TCP の挙動を変更する．そのため，ネットワーク内に tampered-TCP コネクションが増加すると，tampered-TCP コネクションが多くのネットワーク帯域を占領し，既存の TCP コネクションが十分なスループットを獲得できないことが予想される．

我々は [8] において，ウィンドウサイズの上げ幅と下げ幅を変更し，SACK オプション [9] を用いない tampered-TCP がネットワークに与える影響の評価を行い，その有効領域が極めて狭いことを明らかにした．これは，tampered-TCP が既存の TCP よりも積極的にデータ転送を行うことで，逆に自身のスループットを低下させ，自滅することを意味する．しかし，悪意あるユーザが tampered-TCP に対して SACK オプションを用いないことは考えにくい．また，近年は SACK オプションが標準的に用いられているオペレーティングシステムも多い [10–12]．そこで，本稿では SACK オプションを有効にした tampered-TCP がネットワークに与える影響を評価し，広いパラメータ領域において非常に有効であることを示す．

このような tampered-TCP から通常の TCP コネクションを保護するためには，ネットワーク内での制御が必要となる。そこで本稿では，エッジルータで TCP コネクション間の公平性を実現する手法を提案する．コアルータではなく，エッジルータで実現することを想定する理由は 2 点ある．1 点目は，各ルータが監視するコネクション数を減少させることができるため，ルータへの負荷を抑え，ルータ本来の機能であるルーティングへの影響を小さくすることができる点である．2 点目は，tampered-TCP コネクションの利己的な挙動を，コアネットワークに到達する前に制御できる点である．

提案手法は，エッジルータで TCP パケットを観測すること

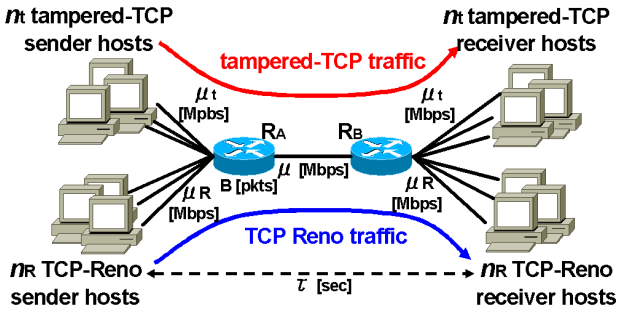


図 1　評価モデル

で，各 TCP コネクションのウィンドウサイズ，あるいはスループットを推測し，その値をもとに，各 TCP コネクションのタンパリング性の有無を判断する．タンパリング性を持つと判断されたコネクションに対しては，そのスループットが TCP Reno と同等になるようにパケット廃棄率を設定し，エッジルータにおいてパケットを意図的に廃棄する．

提案手法は，ns-2 [13] を用いたシミュレーションを行うことで評価する．その結果から，提案手法を用いることで tampered-TCP コネクションを高い確率で検出し，TCP Reno コネクションと tampered-TCP コネクションのスループット比をほぼ 1 に保つことができることを示す．

以下，2 章では，SACK オプションを有効にした tampered-TCP がネットワークに与える影響を評価する．3 章ではエッジルータで TCP コネクション間の公平性を実現する手法の提案を行う．4 章ではシミュレーションによって提案手法の性能を評価する．最後に 5 章で本稿のまとめと今後の課題を述べる．

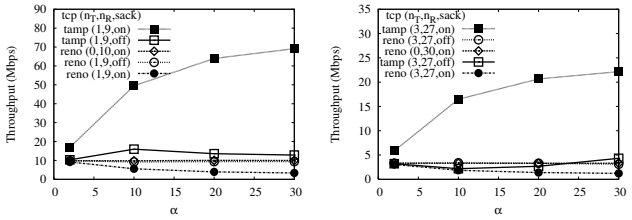
2.　tampered-TCP の有効性評価

本章では，SACK オプションが tampered-TCP コネクションに与える影響について評価を行う．

図 1 に，本章における評価に用いるネットワークモデルを示す．ネットワークモデルは，n_{T} 本の tampered-TCP コネクションの送信側/受信側端末，n_{R} 本の TCP Reno コネクションの送信側/受信側端末，Droptail バッファを持つ 2 つのルータ，およびそれらを接続するリンクから構成されている．ルータ R_{A} とルータ R_{B} 間のリンクの帯域を μ Mbps，ルータ R_{A} のバッファサイズを B packets，送信側/受信側端末間の伝搬遅延時間を τ sec，tampered-TCP コネクションの送信側/受信側端末とルータ間のリンクの帯域を μ_{T} Mbps，TCP Reno コネクションの送信側/受信側端末とルータ間をつなぐリンクの帯域を μ_{R} Mbps とする．各コネクションの送信側端末には常に送信するデータが存在し，ウィンドウサイズが許す限りデータを送信し続けると仮定する．

評価の対象とするのは，輻輳ウィンドウサイズの上げ幅 α を変更した tampered-TCP とし，SACK オプションを用いるものとする．なお，パケット廃棄検出時の輻輳ウィンドウサイズの下げ幅 β は 0.5 に固定している．各パラメータの設定は，$\mu_{\mathrm{R}} = \mu_{\mathrm{T}} = 100$ Mbps，$\mu = 100$ Mbps，$\tau = 20$ msec，$B = 667$ packets，パケット長は 1500 byte である．また，シミュレーション時間は 60 sec とする．

図 2 に，全コネクション数が 10 本，30 本の場合それぞれにおいて，tampered-TCP コネクションが存在しない場合，SACK オプションを無効にした tampered-TCP コネクションが 10 % 存在する場合，および SACK オプションを有効にした tampered-TCP コネクションが 10 % 存在する場合のそれぞれにおける，輻輳ウィンドウサイズの上げ幅 α の変化に対するスループットの変化を示す．図から，tampered-TCP コネクションが存在しない場合はコネクション間の公平性が保たれていることが分かる．また，SACK オプションが無効である tampered-TCP コネクションは，共存する TCP Reno コネクションに比べてそれほど大きなスループットを獲得できていない．しかし，SACK オプションを有効にした tampered-TCP コネクションは，α が大きくなるにつれ大きなスループットを獲得し，TCP Reno コネクションのスループットを圧迫している．



(a) コネクション数 10 本　　(b) コネクション数 30 本

図 2　改造 TCP と通常 TCP 間のスループット ($\beta = 0.5$)

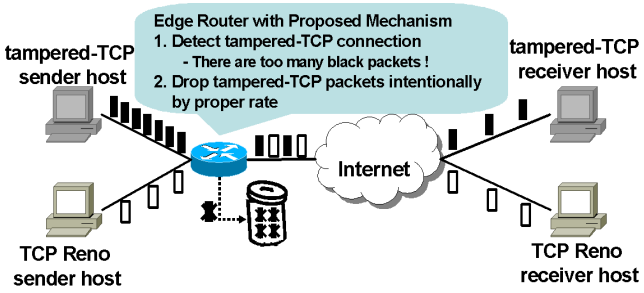


図 3　tampered-TCP の検出・制御手法の動作

したがって，このような tampered-TCP コネクションから通常の TCP コネクションを保護するためには，ネットワーク内における対策が必要となる。

3.　提 案 手 法

本章では，エッジルータで TCP コネクション間の公平性を実現する手法を提案する．提案手法では，図 3 に示すように，エッジルータ上で TCP パケットを観測することで，各 TCP コネクションのウィンドウサイズあるいはスループットを推測し，その結果を用いて tampered-TCP コネクションを検出する．さらに，検出した tampered-TCP コネクションに対して適切なパケット廃棄率を設定してパケットを意図的に破棄することにより，tampered-TCP コネクションのスループットを低下させ，TCP Reno コネクションを保護する．

以降では，TCP コネクションのウィンドウサイズ，およびスループットの推測値を用いて TCP コネクション間の公平性を保つ手法を述べる．なお本稿では，前者をウィンドウサイズ監視手法，後者をスループット監視手法と呼ぶ．

3.1　ウィンドウサイズ監視手法

ウィンドウサイズ監視手法は，エッジルータを通過する各 TCP コネクションのパケットを監視し，輻輳ウィンドウサイズを継続的に推測する．さらに推測結果を用いて，送信側 TCP が用いている輻輳ウィンドウサイズの上げ幅 α および下げ幅 β を推測する．その結果，TCP Reno よりも高いスループットを不当に獲得していると考えられる場合には，そのコネクションを tampered-TCP コネクションであると判断し，適切なパケット廃棄率を設定して制御を行う．以降では，具体的なアルゴリズムを説明する．

（1）　ウィンドウサイズの推測

TCP は 1 ウィンドウ単位でパケットをバースト的に送信する．そのため，あるウィンドウの最後尾のパケットと次のウィンドウの先頭のパケットの間の間隔は，他の連続する 2 パケットの間隔と比較して大きくなる．その大きな間隔によって区切られる 2 つのウィンドウの境界を検出することにより，1 ウィンドウ内に送信側 TCP によって送信されたパケット数をエッジルータ上で観測し，ウィンドウサイズの推測を行う．

そのため，各 TCP コネクション内の連続する 2 パケットの到着間隔を記録し，到着間隔の変化量を観察する．変化量の観察にあたり，観測値の急激な変化を検知する手法を提案した文献 [14] において定義されている以下の式を用いる．

$$g_k = (1-\delta)g_{k-1} + \delta(y_k - \bar{y})^2$$

上式は，観測値 y_k とその平均値 $\bar{y}$ の差を二乗した値について，平滑化パラメータ δ を用いて指数移動平均値を求めている．その値が閾値 h を超えた場合に，そのときの観測値がそれ以前

の観測値と比較して急激に変化したと判断する．この式において，y_k を k 番目のパケット間隔，$\bar{y}$ をパケット間隔の平均値とすることでウィンドウの境界を検出し，ウィンドウサイズの推測を行う．これにより，1 RTT に約 1 回，ウィンドウサイズの推測値が得られると考えられる．

一方, [15] において提案されている手法を用いてルータで RTT を推測し，ウィンドウサイズを推測する方法もある．この方法では各コネクションの RTT を高い精度で推定する必要があるため，本稿では採用していない．本節で提案しているウィンドウサイズ監視手法は，ルータにおいて各コネクションの RTT の推測が不要なことが特徴のひとつである．

（2） α, β の推測

パケット廃棄によって送信側 TCP のウィンドウサイズが減少した場合，エッジルータにおけるウィンドウサイズの推測値も減少する．ここで，あるパケット廃棄によってウィンドウサイズの推測値が減少した直後から，次のパケット廃棄によってウィンドウサイズの推測値が減少する直前までの区間をサイクルと呼び，あるサイクル c の j RTT 後のウィンドウサイズの推測値を $W_e(c, j)$ と定義する．

α の推測値を算出するため，連続する 2 つのウィンドウサイズの推測値の増加量 $\alpha_e(c, j)$ を

$$\alpha_e(c, j) = W_e(c, j) - W_e(c, j-1)$$

と算出する．この値に関して，1 サイクル分の平均値を次式で求める．

$$\overline{\alpha_e}(c) = \frac{\sum_{j=1}^{l(c)} \alpha_e(c, j)}{l(c)}$$

ここで，$l(c)$ はサイクル c におけるウィンドウサイズの推測値のサンプル数である．上式によって得られた α の推測値に対して平滑化を行う．平滑化を行った α の推測値を $\overline{\alpha_e}$ とすると，$\overline{\alpha_e}$ は平滑化パラメータ γ_α を用いて，以下のように導出する．

$$\overline{\alpha_e} = (1 - \gamma_\alpha)\overline{\alpha_e} + \gamma_\alpha \overline{\alpha_e}(c)$$

サイクル c における β の推測値 $\beta_e(c)$ は，次式で求めることができる．

$$\beta_e(c) = \frac{W_e(c, 1)}{W_e(c-1, l(c-1))}$$

上式によって得られた β の推測値に対して平滑化を行う．平滑化を行った β の推測値を $\overline{\beta_e}$ とすると，$\overline{\beta_e}$ は平滑化パラメータ γ_β を用いて，以下のように求める．

$$\overline{\beta_e} = (1 - \gamma_\beta)\overline{\beta_e} + \gamma_\beta \beta_e(c)$$

（3） パケット廃棄率の推測

パケット廃棄率は，提案手法を実装したエッジルータ上の MIB (Management Information Base) [16] で管理されている情報を用いて算出する．通常 MIB では，ルータを通過したパケット数やルータで破棄されたパケット数を保存している．そのため，提案手法を実装したエッジルータがボトルネックになっていると仮定すると，MIB が管理する情報によって算出したパケット廃棄率が，ルータを通過する TCP コネクションのパケット廃棄率とほぼ等しくなる．一方，提案手法を実装したエッジルータ以外のルータがボトルネックになっている場合，この方法で算出したパケット廃棄率は，各 TCP コネクションが経験するパケット廃棄率より小さくなる．すると，後述する提案手法は適切に動作しない可能性がある．この場合，たとえば [17] で提案されている，個々の TCP コネクションのパケット廃棄率をルータにおいて推測する手法を用いることで改善することが考えられる．MIB を用いる方法と [17] で用いられている方法の性能や，ルータの処理負荷の比較は今後の課題としたい．

ウィンドウサイズの上げ幅を大きくした tampered-TCP と通常の TCP Reno コネクションが存在する場合，tampered-TCP コネクションが原因となりルータにおけるパケット廃棄率が高くなる．我々のこれまでの研究 [8] から，tampered-TCP コネクションのパケット廃棄率はその輻輳ウィンドウサイズの上げ幅 α に比例して大きくなることがわかっている．すなわち，本提案手法においては，ルータを通過する全てのコネクションが通常の TCP Reno コネクションであると仮定した場合のパケット廃棄率を推測し，その値を基に tampered-TCP コネクションに対して設定すべきパケット廃棄率を算出する必要がある．

エッジルータにおける廃棄パケット数を n_d，全通過パケット数を n_a，各 TCP コネクションの $\overline{\alpha_e}$ の平均値を $\overline{A_e}$ とすると， パケット廃棄率の推測値 p は次式で求める．

$$p = \frac{\frac{n_d}{n_a}}{\overline{A_e}}$$

本提案手法では，この式から得られる p に対して以下のように平滑化を行った値を後述する制御手法において用いる．平滑化したパケット廃棄率の推測値を $\overline{p}$ とすると，$\overline{p}$ は平滑化パラメータ γ_d を用いて，

$$\overline{p} = (1 - \gamma_d)\overline{p} + \gamma_d p$$

と求める．なお，最新のパケット廃棄率の推測値 p は後述の目標パケット廃棄率を更新する際に算出し，同時に $\overline{p}$ も更新する．

（4） タンパリング性の判定

文献 [4] では，文献 [18] で提案されている TCP Reno コネクションのスループットを算出する式に対し，輻輳回避フェーズにおけるウィンドウサイズの上げ幅 α および下げ幅 β を一般化した式を提案しており，以下の式が満たされる場合，TCP Reno と同等のスループットが得られるとしている．

$$\alpha = \frac{4(1 - \beta^2)}{3}$$

そこで提案手法においては，$\overline{\alpha_e}$，$\overline{\beta_e}$ が次式を満たす場合に，そのコネクションが tampered-TCP コネクションであると判断する．

$$\frac{4(1 - \overline{\beta_e}^2)}{3\overline{\alpha_e}} < (1 - \gamma_w)$$

γ_w $(0 < \gamma_w < 1)$ は，$\overline{\alpha_e}$，$\overline{\beta_e}$ の推測誤差を許容するためのパラメータである．タンパリング性を持たないと判断されたコネクションに対しては，r_w 個のパケットがルータに到着する毎にタンパリング性を再判定する．なお，r_w は正の整数 k_w を用いて次式で定義する．

$$r_w = \frac{k_w}{\overline{p}}$$

（5） 目標パケット廃棄率の算出

タンパリング性を持つコネクションのスループットが TCP Reno と同等になるよう，意図的にパケットを破棄するための目標パケット廃棄率 p' を算出する．p' の設定の際には，ルータを通過する TCP コネクションが全て通常の TCP Reno であると仮定した場合の TCP Reno コネクションのウィンドウサイズの挙動に着目する．ここで，その TCP Reno コネクションを仮想 TCP Reno コネクションと呼ぶ．p' は，仮想 TCP Reno コネクションの 1 サイクルあたりのスループットと，制御後の tampered-TCP コネクションの 1 サイクルあたりのスループットが等しくなるように設定する．

図 4 に，仮想 TCP Reno コネクションおよび目標パケット廃棄率を用いてパケットを意図的に廃棄したときの tampered-TCP コネクションの，ウィンドウサイズの時間的変化を模式的に示す．仮想 TCP Reno コネクションが 1 サイクル中に送信するパケット数は，3. 1(3) で求めた，ルータを通過する TCP コネクションが全て通常の TCP Reno であると仮定した場合のルータにおけるパケット廃棄率の推測値 $\overline{p}$ を用いて，$\frac{1}{\overline{p}}$ と表すことができる．これは，図 4(a) の斜線部の面積に相当するため，サイクル開始時の仮想 TCP Reno コネクションのウィンドウサイズの推測値を W_R とすると，

$$\frac{1}{2} \cdot (W_R + \frac{1}{2}W_R) \cdot \frac{1}{2}W_R = \frac{1}{\overline{p}} \qquad (1)$$

という関係が成り立つ．一方，図 4(b) に示す tampered-TCP コネクションについても同様に，

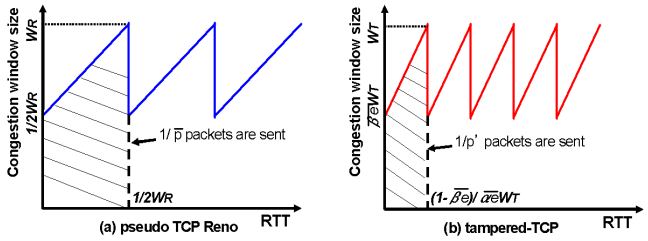


図 4　ウィンドウサイズ監視手法における目標パケット廃棄率の設定

$$\frac{1}{2}\cdot(W_T+\overline{\beta_e}W_T)\cdot\frac{(1-\overline{\beta_e})}{\overline{\alpha_e}}W_T=\frac{1}{p'} \qquad (2)$$

という関係が成り立つ．ここで，W_T はサイクル開始時の tampered-TCP コネクションのウィンドウサイズの推測値である．また，図 4 の 2 つの斜線部分のスループットが等しくなるためには，

$$\frac{\frac{1}{\bar{p}}}{\frac{1}{2}W_R}=\frac{\frac{1}{p'}}{\frac{(1-\overline{\beta_e})}{\overline{\alpha_e}}W_T} \qquad (3)$$

という関係が成り立つ必要がある．

式 (1)-(3) を用いることにより，目標パケット廃棄率を

$$p'=\frac{(1+\overline{\beta_e})}{3(1-\overline{\beta_e})}\overline{\alpha_e}\bar{p}$$

と設定することができる．目標パケット廃棄率は，ネットワークの状況や α，β の変化に追従して設定する．そのため，u_w 個のパケットがルータに到着する毎に再設定を行う．なお，u_w は次式で定義する．

$$u_w=\frac{1}{p'}$$

3.2　スループット監視手法

スループット監視手法では，制御区間を導入し，各制御区間ごとに，sFlow [19] や NetFlow [20] といったトラヒック監視ツールを用いて各 TCP コネクションのスループットの観測値を取得する．さらに，文献 [18] で提案されている TCP コネクションのスループットを算出する式から，各 TCP コネクションが TCP Reno コネクションであると仮定した場合に獲得できるスループットの推測値を算出する．観測値と推測値を比較し，観測値の方が大きい場合，tampered-TCP コネクションと判断し，観測値と推測値の値に合わせて適切なパケット廃棄率を設定して制御を行う．以降では，具体的なアルゴリズムを説明する．

（1）　制御区間の設定

制御区間は次式で定義される $n_I(i)$ 個のパケットがルータに到着するのにかかる時間とする．

$$n_I(i)=\frac{k_t}{p(i)}$$

ここで，$p(i)$ は制御区間 i 開始時のパケット廃棄率の推測値，k_t は正の整数とする．

（2）　スループットの観測値の算出

トラヒック監視ツールでは，エッジルータを通過したパケットの総バイト数とコネクションの監視時間を保存している．制御区間 i におけるバイト数を $b(i)$，制御区間 i の長さを $t(i)$，スループットの観測値を $T_o(i)$ とすると，$T_o(i)$ は次式で求めることができる．

$$T_o(i)=\frac{b(i)}{t(i)}$$

（3）　スループットの推測値の算出

文献 [18] で提案されている TCP コネクションのスループットを推測する式は，パラメータとしてパケットサイズ，delayed ACK オプションの設定値，ラウンドトリップ時間 (RTT)，再送タイムアウト時間，パケット廃棄率を用いる．制御区間 i において，ある TCP コネクションが TCP Reno コネクションであると仮定した場合に獲得できるスループットの推測値 $T_e(i)$ を算出するためには，各パラメータの推測値をエッジルータ上で算出する必要がある．そこで以降では，各パラメータの推測方法を述べる．

- パケットサイズ

sFlow や NetFlow などのトラヒック監視ツールは，前述のエッジルータを通過したパケットの総バイト数とコネクションの監視時間に加え，ルータを通過したパケット数を保存している．そこで，制御区間 i 開始時にトラヒック監視ツールで管理されているパケット数を $n(i)$，パケットサイズの推測値を $s_e(i)$ とすると，$s_e(i)$ は次式を用いて推測することができる．

$$s_e(i)=\frac{b(i)}{n(i)}$$

- delayed ACK オプションの設定値

制御区間 i において，$(j-1)$ 番目に観測された ACK パケットの ACK 番号を $a(i,j-1)$，j 番目に観測された ACK パケットの ACK 番号を $a(i,j)$ とする．この 2 つの ACK パケットの ACK 番号の差を delayed ACK オプションの設定値の推測値とする．その値を $del_e(i,j)$ とすると，$del_e(i,j)$ は次式で定義される．

$$del_e(i,j)=a(i,j)-a(i,j-1)$$

上式で算出される delayed ACK オプションの推測値に関して，制御区間 i 内の平均値 $\overline{del_e}(i)$ を以下のように求める．

$$\overline{del_e}(i)=\frac{\sum_{j=1}^{n_b}del_e(i,j)}{n_b}$$

ここで，n_b は制御区間 i で得られた delayed ACK オプションの設定値のサンプル数とする．なお，重複 ACK を観測した場合およびその直後は，推測値の算出対象としない．この場合は連続する 2 つの ACK パケットの ACK 番号の差が 0，または delayed ACK オプションの設定値を上回る値になり，推測誤差の原因となるためである．

- RTT

過去の文献において，様々な RTT の推測手法が提案されているが，本稿では文献 [15] で提案されている，タイムスタンプを用いた RTT 推測手法を用いる．この手法では，ルータで各 TCP コネクションのパケットを監視し，データパケット dp_1 のタイムスタンプ ts_1 およびルータに到着した時間 m_1 を記憶する．次に，ts_1 をエコーとして持つ ACK パケット ap_1 を観測すると，ルータは ap_1 が dp_1 に対して送信された ACK パケットであると認識し，そのタイムスタンプ ts_2 を記憶する．さらに，ts_2 をエコーに持つデータパケット dp_2 を観測すると，ルータは送信側端末が ap_1 を受信したことによって dp_2 を送信したと認識し，その到着時間 m_2 を記憶する．この dp_1 の到着時間 m_1 および dp_2 の到着時間 m_2 を用いて，制御区間 i における j 番目の RTT の推測値 $rtt_e(i,j)$ を次式で定義する．

$$rtt_e(i,j)=m_2-m_1$$

さらに，制御区間 i における平均 RTT $\overline{rtt_e}(i)$ を次式より求める．

$$\overline{rtt_e}(i)=\frac{\sum_{j=1}^{n_r}rtt_e(i,j)}{n_r}$$

なお，n_r は制御区間 i で得られた RTT のサンプル数とする．

- 再送タイムアウト時間

文献 [21] では，RTT の 4 倍を再送タイムアウト時間の推測値とすることが推奨されている．本稿でも，この推測方法を用いて再送タイムアウト時間の推測値 $rto_e(i)$ を得る．すなわち，以下の式を用いる．

$$rto_e(i)=4\overline{rtt_e}(i)$$

- パケット廃棄率

パケット廃棄率の算出方法に関しては，ウィンドウサイズ監視手法とほぼ同様である．ただし，スループット監視手法では各 TCP コネクションの輻輳ウィンドウサイズの上げ幅 α の推測

値を取得することができない．そのため，ルータを通過する各 TCP コネクションがすべて TCP Reno であると仮定した場合のパケット廃棄率を推測することができない．よって，制御区間 i 内のエッジルータにおける廃棄パケット数を $n_d(i)$，全通過パケット数を $n_a(i)$，パケット廃棄率の推測値を $p(i)$ とすると，次式で $p(i)$ を導出する．

$$p(i) = \frac{n_d(i)}{n_a(i)}$$

ルータを通過するコネクション数が少ない場合，上式で求めたパケット廃棄率の推測値は，TCP コネクションが経験するパケット廃棄率より大きくなる．しかし，エッジルータを通過するコネクション数が増加し，tampered-TCP コネクションの数が相対的に少なくなると，その影響は小さくなると考えられる．したがって，コネクション数が多い場合には，上式を用いて算出されるパケット廃棄率の推測値を用いることにより，後述の制御手法によって適切な目標パケット廃棄率を設定することができると考えられる．提案手法においては，上式から得られる $p(i)$ を次式によって平滑化した値を用いて制御を行う．

$$\overline{p}(i) = (1 - \gamma_l)\overline{p}(i-1) + \gamma_l p(i)$$

（4） タンパリング性の判定
本提案手法では，制御区間 i におけるスループットの観測値 $T_o(i)$ と推測値 $T_e(i)$ が次式を満たす場合に，そのコネクションが tampered-TCP コネクションであると判断する．

$$\frac{T_o(i)}{T_e(i)} > (1 + \gamma_t)$$

なお，γ_t $(0 < \gamma_t)$ は，$T_o(i)$，$T_e(i)$ の真値からの誤差を許容するためのパラメータである．タンパリング性の有無が判断されたコネクションは，後続の制御区間においても継続的にタンパリング性の判定を行う．これにより，一時的な推測誤差による誤判断の影響を小さくすることができる．

（5） 目標パケット廃棄率の算出
制御区間 i の目標パケット廃棄率 $p'(i)$ を設定するにあたり，TCP コネクションのスループットがパケット廃棄率の平方根に反比例する，という性質を利用する．この性質から，$p'(i)$ を次式で定義することができる．

$$p'(i) = \left(\frac{T_o(i-1)}{T_e(i-1)}\right)^2 p'(i-1)$$

目標パケット廃棄率は，ネットワークの状況に追従して設定するため，各制御区間ごとに再設定する．

4. 提案手法のシミュレーション評価

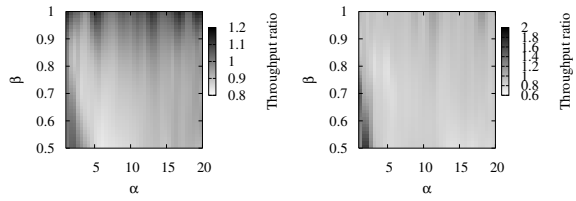
本章では，3. 章で提案した手法をシミュレーションにより評価する．ウィンドウサイズ監視手法のパラメータは，$\delta = 0.6$，$h = 0.0001$，$\gamma_\alpha = 0.6$，$\gamma_\beta = 0.6$，$\gamma_d = 0.6$，$\gamma_w = 0.1$，$k_w = 4$ とする．スループット監視手法のパラメータは，$\gamma_l = 0.6$，$\gamma_t = 2$，$k_t = 4$ とする．

シミュレーションには図 1 に示すネットワークモデルを用いる．各パラメータの設定は，$\mu_{\mathrm{R}} = \mu_{\mathrm{T}} = 100$ Mbps，$\mu = 50$ Mbps，$\tau = 20$ msec，$B = 333$ packets，$n_T = 1$，$n_R = 20$，パケット長は 1500 byte である．シミュレーション時間を 60 sec とし，各 TCP コネクションは SACK オプションを有効にする．この環境において，tampered-TCP コネクションの輻輳ウィンドウサイズの上げ幅 α を [1，20]，下げ幅 β を [0.5，1] の範囲で変化させた場合の提案手法の性能を評価する．

性能指標には，次式で定義されるスループット比を用いる．

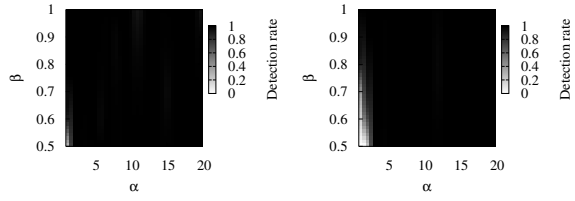
$$\text{スループット比} = \frac{\text{(tampered-TCP のスループット)}}{\text{(TCP Reno のスループット)}}$$

この値が 1 より大きい場合は tampered-TCP が効果を発揮しており，1 より小さい場合は効果を発揮していないと考えることができる．また，スループット比に加え，tampered-TCP コネクションの検出確率と検出にかかる時間，TCP Reno コネクションを tampered-TCP コネクションと誤認する確率を評価指標として用いる．



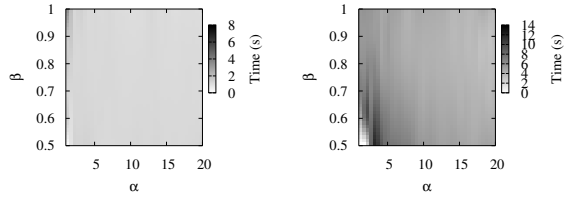
(a) ウィンドウサイズ監視手法　　(b) スループット監視手法

図 5　スループット比の変化



(a) ウィンドウサイズ監視手法　　(b) スループット監視手法

図 6　tampered-TCP コネクションの検出確率



(a) ウィンドウサイズ監視手法　　(b) スループット監視手法

図 7　tampered-TCP コネクションの検出時間

4.1 スループット比

ウィンドウサイズ監視手法およびスループット監視手法を用いた場合のスループット比の変化を図 5 に示す．図 5(a) から，ウィンドウサイズ監視手法を用いることでほぼすべてのパラメータ領域においてスループット比を 1 程度に保っていることがわかる．

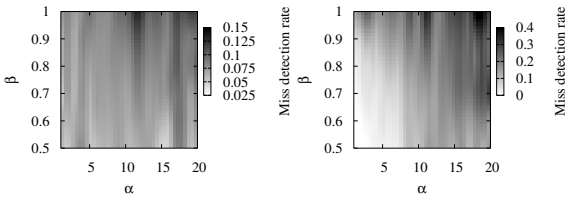
また，図 5(b) から，スループット監視手法を用いた場合には，タンパリング性が弱い領域においてスループット比が 1 より大きくなっていることがわかる． これは，γ_t を用いて推測結果の誤差を許容しているため，タンパリング性が弱いコネクションを tampered-TCP コネクションと判断できないためである．しかし，それ以外の広いパラメータ領域では，スループット比を 1 程度に保つことができている．

4.2 tampered-TCP の検出確率と検出時間

図 6 および図 7 に，tampered-TCP コネクションの検出確率および検出時間の変化を示す．

図 6 から，tampered-TCP コネクションの検出確率については，スループット監視手法，ウィンドウサイズ監視手法ともに類似した傾向を示していることがわかる．通常の TCP Reno と同じパラメータである $(\alpha, \beta) = (1, 0.5)$ 付近では検出確率が 0 に近い値となっている．これは，提案方式が通常の TCP Reno と同じ動作をする TCP コネクションを tampered-TCP と判断していないことを意味する．また，タンパリング性が弱い領域においては，検出確率が低くなっている．これは，タンパリング性を判断する際に，γ_w および γ_t を用いて推測値の誤差を許容しているため，タンパリング性が弱いコネクションを tampered-TCP コネクションと判断できないことがあるためである．しかし，それ以外の広いパラメータ領域において，ほぼ 100 % の確率で tampered-TCP を検出している．

図 7 より，tampered-TCP コネクションの検出時間は，スループット監視手法は約 5 秒，ウィンドウサイズ監視手法は約 2 秒であることがわかる．タンパリング性が弱い領域では，やや長い検出時間を要しているものの，その場合でも 10 秒程度である．現在，ISP などが高レートのフローを検出する際には，MIB の情報が主に用いられているが，MIB の情報更新間隔が通常 5 分であることを考えると，提案手法の検出時間は非常に



(a) ウィンドウサイズ監視手法　　(b) スループット監視手法

図 8　TCP Reno コネクションの誤認確率

表 1　誤認された TCP Reno と誤認されなかった TCP Reno のスループットとスループット比

ウィンドウサイズ監視手法				
α	β	誤認 Reno (Mbps)	Reno (Mbps)	スループット比
10	0.7	2.614	2.329	1.122
20	0.9	2.330	2.391	0.975

スループット監視手法				
α	β	誤認 Reno (Mbps)	Reno (Mbps)	スループット比
10	0.7	2.464	2.358	1.050
20	0.9	2.209	2.405	0.926

短いといえる．

以上から，スループット監視手法，ウィンドウサイズ監視手法ともに，タンパリング性が弱い領域においては検出確率，検出時間という両方の観点から精度が低下するものの，その他の広いパラメータ領域では，数秒の観測時間の後，100 % の確率で tampered-TCP コネクションを検出することができることが明らかとなった．

4.3　TCP Reno の誤認確率

本節では，提案手法が TCP Reno コネクションを tampered-TCP コネクションと誤って判断する確率を評価する．図 8 に，ウィンドウサイズ監視手法およびスループット監視手法における誤認確率を示す．

図から，ウィンドウサイズ監視手法，スループット監視手法ともに，tampered-TCP コネクションのタンパリング性が強くなるにしたがって，TCP Reno を tampered-TCP と誤認する確率が高くなっていることがわかる．これは以下の理由による．本稿で対象としている tampered-TCP コネクションは，タンパリング性が強くなると急激にウィンドウサイズを大きくするため，それらと競合し影響を受ける TCP Reno コネクションのウィンドウサイズおよびスループットの変動が不安定になる．その結果，提案方式がエッジルータ上で算出する α，β の推測値，およびスループットの観測値と推測値の誤差が大きくなるためであると考えられる．

しかし，誤認された場合でも，誤認された TCP Reno コネクションが受ける影響は小さい．そのことは，誤認された TCP Reno コネクションと，誤認されなかった TCP Reno コネクションのスループットおよびスループット比を示した表 1 から分かる．以上のことから，提案手法は TCP Reno コネクションを tampered-TCP と誤認する場合があるものの，誤認によって TCP Reno コネクションのスループットが低下することはほとんどないことが分かる．

5.　ま　と　め

本稿では，エッジルータで tampered-TCP コネクションを検出・制御することで，TCP コネクション間の公平性を実現する手法を提案した．提案方法は，エッジルータ上で各 TCP コネクションのウィンドウサイズ，あるいはスループットを推測し，推測結果に基づいてタンパリング性の判断，および目標パケット廃棄率を設定し， tampered-TCP コネクションの制御を行う．シミュレーションによる性能評価から，提案手法を用いることにより高い確率で tampered-TCP コネクションを検出し，TCP Reno コネクションとのスループット比を，ほぼ 1 に保つことができることを示した．

今後の課題としては，提案手法を実装したエッジルータがボトルネックになっていない場合のパケット廃棄率を推測する方法を検討することが挙げられる．また，実ネットワーク環境における提案手法の性能評価を行うことも検討したい．

文　　献

[1] M. Allman, V. Paxson, and W. Stevens, “TCP congestion control,” *RFC2581*, Apr. 1999.

[2] S. Savage, N. Cardwell, D. Wetherall, and T. Anderson, “TCP congestion control with a misbehaving receiver,” *ACM SIGCOMM Computer Communications Review*, vol. 29(5), pp. 71–78, Oct. 1999.

[3] M. Baldi, Y. Ofek, and M. Yung, “Idiosyncratic signatures for authenticated execution of management code,” in *Proceedings of DSOM 2003*, Oct. 2003.

[4] Y. R. Yang and S. S. Lam, “General AIMD congestion control,” in *Proceedings of ICNP 2000*, Nov. 2000.

[5] L. Mamatas and V. Tsaoussidis, “Protocol behavior : More effort, more gains?,” in *Proceedings of PIMRC 2004*, Sept. 2004.

[6] K. Tan, 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.

[7] H. Shimonishi, M. Sanadidi, and T. Murase, “Assessing interactions among legacy and high-speed TCP protocols,” in *Proceedings of PFLDnet 2007*, Feb. 2007.

[8] J. Maruyama, G. Hasegawa, and M. Murata, “Is tampered-TCP really effective for getting higher throughput in the Internet?,” in *Proceedings of ATNAC 2006, pp. 167-171*, Dec. 2006.

[9] E. Blanton, M. Allman, K. Fall, and L. Wang, “A conservative selective acknowledgment (SACK)-based loss recovery algorithm for TCP,” *RFC3517*, Apr. 2003.

[10] J. Padhye and S. Floyd, “On inferring TCP behavior,” *ACM SIGCOMM Computer Communication Review*, vol. 31(4), pp. 287–298, Aug. 2001.

[11] K. Pentikousis and H. Badr, “Quantifying the deployment of TCP options - a comparative study,” *IEEE Communications Letters*, vol. 8(10), pp. 647–649, Oct. 2004.

[12] M. Mellia, R. L. Cigno, and F. Neri, “Measuring IP and TCP behavior on edge nodes with Tstat,” *Computer Networks*, vol. 47(1), pp. 1–21, Jan. 2005.

[13] “The Network Simulator - ns-2.” available at `http://www.isi.edu/nsnam/ns/`.

[14] M. Basseville and I. Nikiforov, *Detection of abrupt changes: Theory and application*. Prentice-Hall,Inc, 1993.

[15] B. Veal, K. Li, and D. K. Lowenthal, “New methods for passive estimation of TCP round-trip times,” in *Proceedings of PAM 2005*, pp. 121–134, Mar. 2005.

[16] K. McCloghrie and M. Rose, “Management information base for network management of TCP/IP-based Internets: MIB-II,” *RFC1213*, Mar. 1991.

[17] P. Benko and A. Veres, “A passive method for estimating end-to-end TCP packet loss,” in *Proceedings of IEEE GLOBECOM 2002*, Nov. 2002.

[18] J. Padhye, V. Firoiu, D. Towsley, and J. Kurose, “Modeling TCP throughput: A simple model and its empirical validation,” in *Proceedings of ACM SIGCOMM ’98*, Sept. 1998.

[19] P. Phaal, S. Panchen, and N. McKee, “InMon corporation’s sFlow: A method for monitoring traffic in switched and routed networks,” *RFC 3176*, Sept. 2001.

[20] “NetFlow.” available at `http://www.cisco.com/japanese/warp/public/3/jp/product/hs/ios/nmp/prodlit/pdf/iosnf_ds.pdf`.

[21] M. Handley, S. Floyd, J. Pahdye, and J. Widmer, “TCP friendly rate control (TFRC),” *RFC3448*, Jan. 2003.