

# 無線 LAN 環境における 複合型 TCP 輻輳制御手法の性能評価とその改善手法

橋本 匡史<sup>†</sup> 長谷川 剛<sup>††</sup> 村田 正幸<sup>†</sup>

<sup>†</sup> 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

<sup>††</sup> 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: <sup>†</sup>{m-hasimt,murata}@ist.osaka-u.ac.jp, <sup>††</sup>hasegawa@cmc.osaka-u.ac.jp

あらまし 本稿では、近年高速・高遅延環境向けに提案されており、ネットワーク輻輳の指標としてパケット廃棄の発生やラウンドトリップ時間(RTT)、およびその両方を複合的に用いる TCP 輻輳制御手法の、IEEE 802.11 無線 LAN 環境における性能をシミュレーションにより評価し、無線 LAN 環境において発生する遅延時間の変動が TCP 性能に与える影響を評価する。また、従来指摘されている上下フロー間だけでなく、上りフロー間にも深刻な不公平が発生することを明らかにする。さらに、不公平性を改善するための手法として、ACK パケットの損失に対して輻輳制御を行う手法を提案し、提案手法が上りフロー間の不公平に対して有効であるだけでなく、上下フロー間の不公平に対しても一定の改善を行うことができることを示す。

キーワード 無線 LAN, 複合型 TCP, 公平性, 輻輳制御

## Performance Evaluation and Improvement of Hybrid TCP Congestion Control Mechanisms in Wireless LAN Environment

Masafumi HASHIMOTO<sup>†</sup>, Go HASEGAWA<sup>††</sup>, and Masayuki MURATA<sup>†</sup>

<sup>†</sup> Graduate School of Information Science and Technology, Osaka University 1-5, Yamadaoka, Suita, Osaka, 560-0871 Japan

<sup>††</sup> Cybermedia Center, Osaka University 1-32, Machikaneyama, Toyonaka, Osaka 560-0043 Japan

E-mail: <sup>†</sup>{m-hasimt,murata}@ist.osaka-u.ac.jp, <sup>††</sup>hasegawa@cmc.osaka-u.ac.jp

**Abstract** In this report, we evaluate the performance of recent TCP variants for high-speed and long-delay networks in IEEE 802.11 wireless LAN environment. We show that some of them have well-known TCP unfairness among up-link and down-link flows, and there is another unfairness problem among up-link flows. We then proposed an end-to-end-basis modification to TCP congestion control mechanisms to alleviate the unfairness problems, which activates congestion control when detecting ACK packet losses. We present that the proposed method is effective not only for improving the TCP fairness among up-link flows but also for the TCP fairness among up-link and down-link flows.

**Key words** wireless LAN, hybrid TCP, fairness, congestion control

### 1. はじめに

近年のネットワークの高速、大規模化にともない、エンドホスト間のネットワークパスの利用できるネットワークの帯域遅延積が増大している。このような高速・高遅延ネットワークにおいて、従来の TCP として一般に利用されている TCP Reno は、その輻輳制御方式が原因となり、十分なスループットが得られないことが知られている [1]。

この問題に対して、さまざまな TCP 改良手法の提案が行われている [1-6]。それらの TCP 改良手法は、ネットワークの輻輳の指標としてパケット廃棄の発生を用いる loss-based 手法 [1, 2]、ネットワークの輻輳の指標としてラウンドトリップ時間 (RTT) を用いる delay-based 手法 [3, 7]、およびそれらを組み合わせた hybrid 手法 [4, 5] の 3 つに大別される [8]。これらの多くは主に

有線ネットワークを対象に検証されている [9]。

一方、無線ネットワーク技術の向上により、ノート PC や携帯電話などの端末が無線 LAN や無線 WAN などを通してインターネットにアクセスすることが一般的になりつつある。無線 LAN 規格として、IEEE によって標準化された IEEE 802.11b [10] や IEEE 802.11a [11] などがある。無線伝送速度は IEEE 802.11b の場合は最大 11 [Mbps] であり、IEEE 802.11a の場合は最大 54 [Mbps] である。さらに、近年標準化が進んでいる IEEE 802.11n は無線伝送速度が最大 300 [Mbps] であるなど、さらなる高速化が行われている。これらの無線 LAN 規格ではアクセス制御方式に CSMA/CA を用いている。CSMA/CA は、搬送波を検出するためのバックオフ時間の間、フレームの送信を待ち、通信中の端末が存在しないことを確認することによって、フレームの衝突をできるだけ避けようと試みる [12]。このことが無線 LAN において伝送遅延時間が変動する一つの要因である。

ネットワークの輻輳の指標として RTT を用いる delay-based 手法のような TCP 改良手法においては、輻輳に関係なく RTT が変動する環境においては、ネットワークの輻輳状態を正しく知ることができないことが予想される。しかし、無線ネットワークインターフェイスを持った端末において、接続されているネットワーク環境によってトランスポート層プロトコルを切り替えるのは現実的ではない。さらに、無線ネットワーク環境は今後も高速化される傾向にあるため、高速・高遅延ネットワーク向けの delay-based 手法や hybrid 手法がそのまま利用されることは十分考えられる。したがって、無線 LAN 環境において、delay-based 手法および hybrid 手法の評価を行うことは重要である。

また、無線 LAN 環境において、TCP フロー間のスループットに関して不公平が生じることが指摘されている [13]。この問題に対しても、さまざまな手法が提案されている [13-17]。これらの手法は、アクセスポイントにおいてその MAC プロトコルのパラメータやキュー管理機構を変更することで TCP フロー間の不公平を改善している。しかし、通常無線アクセスポイントの MAC 機能はハードウェア化されており、変更には大きなコストが伴う。また、キュー管理機構の変更による改善手法においては、アクセスポイントを通過しているフロー数や各フローのスループットの推定を行う必要がある。

そこで本稿では、無線 LAN 環境において、ネットワークの輻輳の指標として RTT 情報を利用する delay-based 手法および hybrid 手法の性能評価をシミュレーションにより行う。delay-based 手法や hybrid 手法が RTT に関する情報をどう用いるかについては、提案されている論文には明確に記述されていないものがほとんどである。そのため、本稿における評価では、公開されている Linux の実装コードおよび ns-2 [18] のシミュレーションコードを基に、各 TCP 改良手法における RTT の利用方法を推定し、評価を行う。シミュレーション結果から、各 TCP 改良手法が実装レベルで実装している RTT 情報のフィルタリング処理により、無線 LAN における遅延変動による影響が抑えられることを示す。さらに、loss-based 手法や hybrid 手法を用いた場合、アクセスポイントにおける ACK パケットの多量の廃棄が原因となり上りフロー間や上下フロー間で深刻な不公平が発生するが、delay-based 手法ではそのような不公平が発生しないことを示す。

しかし、delay-based 手法には、loss-based 手法と混在する環境において著しくスループットが低下するという問題 [19] が存在する。このことは、TCP Vegas をはじめとする delay-based 手法が実際のインターネットにおいて広く用いられていない要因の 1 つとなっている。そこで本稿では、不公平の原因となっている ACK パケットの損失に対して輻輳制御を行うことにより、不公平性を軽減する手法を提案する。シミュレーション結果から、提案手法を loss-based 手法に組み合わせることにより、TCP フロー間の公平性が改善できること、また、アクセスポイントでの改善手法に比べて柔軟性が高いことを示す。

## 2. 関連研究

### 2.1 RTT 情報を利用する TCP 改良手法

本節では、高速・高遅延ネットワーク向けの TCP の輻輳制御方式のうち、輻輳ウィンドウの更新アルゴリズムについて述べる。また、観測した RTT 情報が輻輳ウィンドウサイズの更新アルゴリズムにどのように利用されるかについても述べる。

本節において紹介する delay-based 手法や hybrid 手法は、その手法が提案されている論文などにおいて、輻輳ウィンドウの更新に RTT が利用されることが記述されている。しかし、RTT をどのように計測および統計処理を行い、輻輳ウィンドウの更新がどのような間隔で行われるかについてはほとんど記述されていない。このため、本稿では Linux 上の実装コード (kernel 2.6.22) およびシミュレーションソフトウェアである ns-2 のシミュレーションコードから、各 TCP 改良手法における RTT 情報の利用方法を推定する。

#### 2.1.1 TCP Vegas

TCP Vegas [7] は高速・高遅延ネットワーク向けの TCP 改良手法ではないが、多くの delay-based 手法および hybrid 手法の基本となる輻輳ウィンドウ更新アルゴリズムを利用している。TCP Vegas は以下の式によって、ボトルネックルータのパッファ内に蓄積されているパケット数を推測する。

$$Diff = (w/baseRTT - w/RTT) \cdot baseRTT \quad (1)$$

なお、 $w$  は輻輳ウィンドウサイズ、 $baseRTT$  は今までに観測された最小の RTT、 $Diff$  はネットワーク上に蓄積されているパケット数の推定値、 $RTT$  は最新の RTT の値である。ここで、式 (1) で利用している  $RTT$  をどう設定するかは [7] に明記されていないが、Linux 実装においてはラウンドトリップ時間に計測された最小の RTT を  $minRTT$  とし、式 (1) の  $RTT$  として用いている。TCP Vegas は  $Diff$  の値に基づき、輻輳ウィンドウサイズを RTT に 1 度更新する。具体的には制御パラメータ  $\alpha_v$  および  $\beta_v$  を用いて、 $\beta_v < Diff < \alpha_v$  となるように輻輳ウィンドウサイズを 1 ずつ増減させることで輻輳ウィンドウサイズを調節する。

#### 2.1.2 FAST TCP

FAST TCP [3] は TCP Vegas と同様に、今まで観測された最小の RTT である  $baseRTT$  を用いて、以下の式にしたがって輻輳ウィンドウの更新を RTT に 1 度行う。

$$w \leftarrow \min \left\{ 2w, (1 - \gamma_f)w + \gamma_f \left( \frac{baseRTT}{avgRTT} w + \alpha_f \right) \right\}$$

ここで、 $avgRTT$  は [3] において、明確に指定されておらず、ns-2 のシミュレーションモジュール [20] においては、指数移動平均化された RTT が用いられている。

#### 2.1.3 Compound TCP

TCP Vegas や FAST TCP は純粋な delay-based 手法であるのに対し、Compound TCP (CTCP) [4] は delay-based 手法と loss-based 手法が組み合わされた hybrid 手法である。hybrid 手法は loss-based 手法との公平性を保ちながらネットワーク帯域を有効に利用できる TCP 改良手法である。CTCP は TCP Vegas と同様の式 (1) を利用して、ボトルネックルータのパッファ内に蓄積されているパケット数を推測する。式 (1) における  $RTT$  として、Linux 実装においては、TCP Vegas と同様に  $minRTT$  が用いられている。CTCP は  $Diff$  の値に基づき、以下の式にしたがって輻輳ウィンドウサイズを RTT に 1 度更新する。

$$w \leftarrow w_{reno} + dwnd$$

$$dwnd \leftarrow \begin{cases} dwnd + \max(\alpha_c \cdot w^k - 1, 0) & (Diff < \gamma_c) \\ \max(dwnd - \zeta_c \cdot Diff, 0) & (Diff \geq \gamma_c) \end{cases}$$

ここで、 $w_{reno}$  は TCP Reno の場合の輻輳ウィンドウサイズ、 $dwnd$  は遅延ウィンドウサイズ、 $\alpha_c$ 、 $\zeta_c$  および  $\gamma_c$  は制御パラメータである。

#### 2.1.4 TCP AReno

TCP-Adaptive Reno (AReno) [5] は CTCP と同様に hybrid 手法である。CTCP は輻輳ウィンドウサイズの増減幅を現在の輻輳ウィンドウサイズによって決定するのに対し、AReno はボトルネックリンクの帯域を推定し、それを用いて輻輳ウィンドウの増減幅を決定する。AReno は以下の式により、RTT 情報を用いてネットワークの輻輳レベルを推定する。

$$c = \min \left( \frac{sRTT - RTT_{min}}{RTT_{cong} - RTT_{min}}, 1 \right) \quad (2)$$

$$RTT_{cong} = (1 - a)RTT_{cong} + aRTT \quad (3)$$

ここで、 $c$  はネットワークの輻輳レベル、 $RTT_{min}$  は今まで観測された最小の RTT、 $RTT_{cong}$  は式 (3) により求められるパケット廃棄が発生する直前の RTT、 $sRTT$  は smoothed RTT である。なお [5] では式 (2) において、使用される  $RTT$  の詳細が明確に指定されておらず、Linux 実装では  $RTT$  として通常の TCP が管理している平滑化 RTT である  $sRTT$  が用いられている。AReno は  $c$  の値に基づき輻輳ウィンドウサイズの更新を ACK パケットを受信するたびに行う。

## 2.2 無線 LAN 環境における

### TCP の公平性問題に対する改善手法

無線 LAN 環境における TCP コネクション間で発生する問題に対して様々な手法が提案されている [13-17]。[13] では、TCP 受信側から返信される ACK パケット内の広告ウィンドウを書き換えることにより、上下フロー間の不公平を改善している。

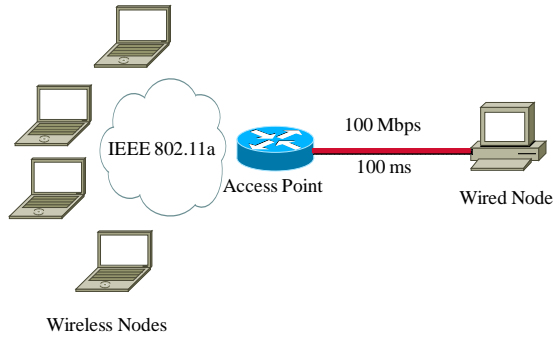


図1 ネットワークトポロジ

しかし、この手法では各フローの RTT が異なる場合において不公平の改善効果が小さくなることや無線帯域が有効に利用できない場合があることが指摘されている。

[14]では、アクセスポイントの MAC プロトコルのパラメータのみを変更し、通常 Distributed InterFrame Space (DIFS) 期間行うキャリアセンスの期間を DIFS 期間より短い Point InterFrame Space (PIFS) 期間に変更することで、上下フロー間および上りフロー間の不公平を改善している。

また [15]では、アクセスポイントのバッファをデータパケットを格納するためのバッファと ACK パケットを格納するためのバッファの2つに分割する。各バッファに格納されたパケットは、以下の式にしたがい確率  $p$ ,  $q$  ( $q = 1 - p$ ) を求め、データパケット用のバッファからは確率  $p$ , ACK パケット用のバッファからは確率  $q$  によってどちらかのパケットが転送される。

$$p = \begin{cases} \frac{mw_r}{mw_r + B_A} & \text{if } B_A < nw_r/b \\ \frac{mb}{mb+n} & \text{if } B_A \geq mw_r/b \end{cases}$$

$$q = \begin{cases} \frac{B_A}{mw_r + B_A} & \text{if } B_A < nw_r/b \\ \frac{n}{mb+n} & \text{if } B_A \geq mw_r/b \end{cases}$$

なお、 $m$  は下りフロー数、 $n$  は上りフロー数、 $w_r$  は広告ウィンドウサイズ、 $B_A$  は ACK パケット用のキューサイズ、 $b$  は1つの ACK パケットが確認応答できるデータパケット数である。

[16]では、アクセスポイントにおいて ACK パケットをフィルタリングすることにより、上りフロー間の公平性を改善している。この手法は TCP が輻輳制御に累積 ACK を用いていることを利用し、アクセスポイントのバッファに蓄積される ACK パケットの数を減らすことで実現している。

しかし [13, 15, 16]の手法は、アクセスポイントにおいて TCP ヘッダを解析する必要があり、さらに [16]の手法の場合は各フローの情報を保持する必要がある。

[17]では、上下フロー間においてそれぞれの合計スループットが公平になるように上りフローに対してアクセスポイントあるいはその周辺ルータがレート制御を行うことにより、上下フロー間の公平性を改善している。この手法では、下りフローの合計スループットを用いてレート制御するため、それを推定する必要がある。

### 3. 無線 LAN 環境が TCP 改良手法に与える影響

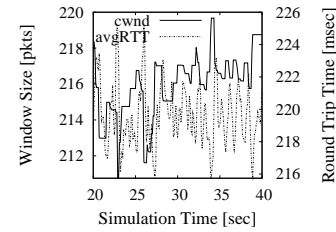
本章では、無線 LAN 環境が 2.1 節で紹介した TCP 改良手法に与える影響について、ns-2 を用いてシミュレーションを行い、明らかにする。

#### 3.1 シミュレーション評価環境

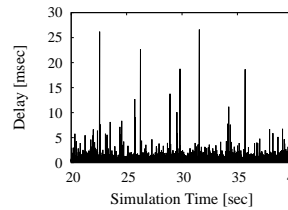
無線 LAN 規格として IEEE 802.11a を用いたネットワークポロジを図 1 に示す。複数の無線端末が 1 つのアクセスポイントと共有し、アクセスポイントから有線端末までは有線リンクにより接続されている。全ての無線端末はアクセスポイントから 4 [m] の位置に配置している。有線リンクの帯域は 100 [Mbps] であり、伝搬遅延時間は 100 [msec] である。また、アクセスポイントのバッファサイズを 100 [pkts] に設定し、有線リンクのバッファサイズ、無線端末の送信バッファサイズおよび広告ウィンドウサイズは十分大きい値を設定した。なお、アクセスポイントおよび有線リンクのキュー管理機構には Drop Tail を用いた。IEEE 802.11a のパラメータは表 1 のとおり

表 1 IEEE 802.11a 設定パラメータ

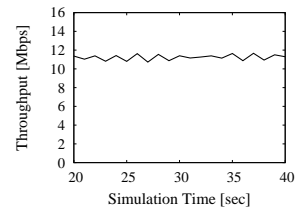
Data Rate	54 [Mbps]
Slot Time	9 [ $\mu$ s]
SIFS	16 [ $\mu$ s]
DIFS	34 [ $\mu$ s]
CW <sub>min</sub>	15
CW <sub>max</sub>	1023



(a) 輻輳ウィンドウと RTT



(b) 無線区間における伝送遅延



(c) スループットの変化

図 2 FAST TCP の場合 (上りフロー数: 2, 伝搬遅延: 100 [msec])

定した。TCP として、TCP Vegas, FAST TCP, CTCP および AReno を用い、各 TCP のパラメータは以下のように設定した。

TCP Vegas  $\alpha_v = 2, \beta_v = 4$

FAST TCP  $\alpha_f = 20, \gamma_f = 0.5$

CTCP  $\alpha_c = 0.125, \gamma_c = 2, \zeta_c = 1$

AReno  $a = 0.5$

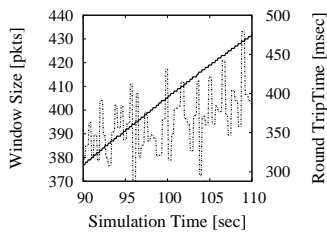
なお、シミュレーション時間は 200 [sec] と設定し、各フローは無線端末 1 台につき 1 本発生させ、シミュレーション開始時に同時にデータ転送を開始させた。ここで、上りフローとは無線端末から有線端末へのデータ転送を行う TCP フローであり、下りフローとはその逆方向のデータ転送を行う TCP フローである。

#### 3.2 シミュレーション評価結果

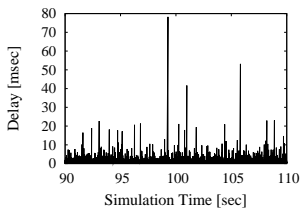
##### 3.2.1 無線区間の遅延変動による影響

まず、TCP として FAST TCP を用いた場合の結果を図 2 に示す。なお、本シミュレーションにおいては上りフローを 2 本発生させた。図 2(a) は輻輳ウィンドウの変化およびそのときの  $avgRTT$  であり、図 2(b) は無線端末からアクセスポイントまでの通信において発生する遅延の変動の変化である。図 2(c) は 1 [sec] ごとに平均したスループットの変化である。また、それぞれの結果は、無線区間の遅延変動の影響が見やすいように、各 TCP 改良手法の動作が安定した 20 [sec] から 40 [sec] の間の結果を示している。

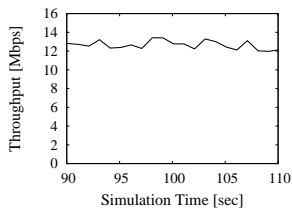
図 2(b) から、無線区間の遅延は、約 2 [msec] の遅延が最も多く、10 [msec] を超えるような遅延は少ないことがわかる。特に遅延変動が大きくなっている 23 [sec] 付近に注目すると、図 2(a) から、その時点において  $avgRTT$  が増加し、約 4 [pkts] 分の輻輳ウィンドウサイズが低下していることがわかる。一方、TCP として TCP Reno を用いた場合のシミュレーション結果を図 3 に示す。図 3(a) と図 3(b) から、遅延変動が 80 [msec] 程度発生したとしても、輻輳ウィンドウサイズは単調に増加し続けており、遅延変動の影響を受けていないことがわかる。つまり、TCP Reno のような loss-based 手法の場合は、遅延変動の影響を受けない。このため、図 3(c) の TCP Reno のスループットのは、図 1 のネットワークポロジにおいて、TCP の輻輳制御が無線区間の遅延変動の影響を受けない場合の結果と考えることができる。TCP Reno のスループットは約 1 [Mbps] 変動していることから、この環境においては、輻輳制御手法に関係なく、約 1 [Mbps] のスループットの変動が発生するといえる。図 2(c) より、FAST TCP のスループットの変動の範囲は約 1 [Mbps] であることがわかる。これは、FAST TCP の場合は無線区間の遅延



(a) 輻輳ウィンドウとRTT

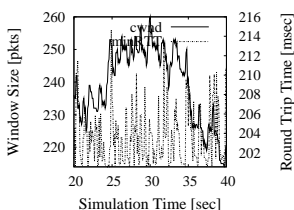


(b) 無線区間における伝送遅延

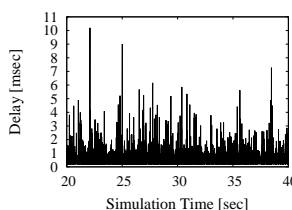


(c) スループットの変化

図3 TCP Reno の場合 (上りフロー数: 2, 伝搬遅延: 100 [msec])



(a) 輻輳ウィンドウとRTT



(b) 無線区間における伝送遅延

図4 CTCP の場合 (上りフロー数: 2, 伝搬遅延: 100 [msec])

変動の影響を受けて輻輳ウィンドウサイズを変化させるが、その変動が無線端末の送信バッファに蓄積されたデータパケット数の変動に対してのみ影響を与えたため、結果的にスループットに対して影響が現れなかったと考えられる。

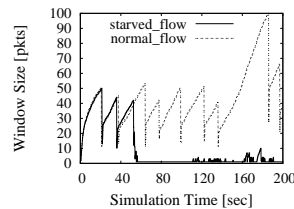
次に、TCP として CTCP を用いた場合の結果を図 4 に示す。図 4(b) より、無線区間の遅延変動が、1 [msec] 程度の変動が最も多く、5 [msec] を超えるような変動は少ないことがわかる。特に、遅延変動が 10 [msec] と大きくなっている 22 [sec] 付近に注目すると、図 4(a) において  $minRTT$  は 5 [msec] と、無線区間の遅延変動より小さいことがわかる。これは、 $minRTT$  は 1 RTT 中に観測された RTT であり、無線区間の遅延変動のような瞬間的に増加した遅延はフィルタリングされるためだと考えられる。したがって、ここで発生している  $minRTT$  の増加は、データパケットが無線端末の送信バッファに蓄積されることによるものであると考えられる。

また、TCP Vegas や AReno についても、FAST TCP や CTCP と同様に、実装レベルで実装されているフィルタリング処理によって RTT 情報がフィルタリングされることや、無線端末の送信バッファによって、無線区間で生じる遅延変動の影響がほとんど現れないという結果を得た。

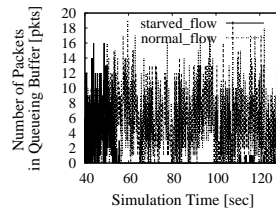
以上から、無線区間の遅延変動による影響は、delay-based 手法および hybrid 手法に対して大きな影響を与えないといえる。

### 3.2.2 上りフロー間の公平性

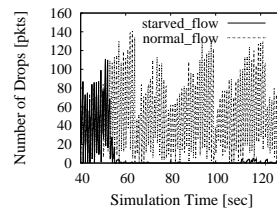
Loss-based 手法や hybrid 手法を用いた場合において、上りフロー数を増加させると、輻輳ウィンドウサイズが大きくなり、無線帯域を占有するフローと輻輳ウィンドウサイズがほとんど増加せず、スループットが非常に小さくなるフローが生じるという現象が現れる。TCP として CTCP を用いた場合において、正常に通信できているフロー (normal\_flow) と通信ができていないフロー (starved\_flow) のシミュレーション結果を図 5(a) に示す。なお、シミュレーションにおいては有線リンクの伝搬遅延を 100 [msec] に設定し、上りフロー数を 16 本として行った。図 5(a) から、normal\_flow は輻輳ウィンドウサイズを増減させ正常に通信できているが、starved\_flow は 55 [sec] 付近から 110 [sec] 付近まで輻輳ウィンドウサイズが 1 から増加していないことがわかる。starved\_flow が通信できていない期間に注目すると、図



(a) 輻輳ウィンドウ



(b) アクセスポイントにおける ACK パケット蓄積数



(c) アクセスポイントにおける ACK パケット廃棄数

図5 CTCP において不公平が生じる場合 (上りフロー数: 16, 伝搬遅延: 100 [msec])

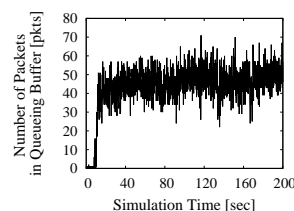


図6 TCP Vegas の場合のアクセスポイントにおいて蓄積される ACK パケット数 (上りフロー: 16, 伝搬遅延: 100 [msec])

5(b) と図 5(c) より、その期間において、このフローの ACK パケットがアクセスポイントのバッファに入ることなく、廃棄されていることがわかる。また、ACK パケットが廃棄されてから次の ACK パケットが廃棄されるまでの間隔が増加していることもわかる。

以上から、上りフロー間で深刻な不公平が生じた原因は以下によるものであると考えられる。図 1 のようなネットワーク環境において上りフロー数が増加すると、無線端末とアクセスポイント間の無線ネットワーク部分がボトルネックとなる。この時、アクセスポイントが輻輳状態となり、ACK パケットがアクセスポイントのバッファに蓄積され、廃棄が発生する。また、TCP はデータパケットが 1 つでも損失した場合においては輻輳制御によって輻輳ウィンドウサイズを減少させる。しかし、ACK パケットが損失した場合は輻輳制御が行われない。このため、アクセスポイントが輻輳しているにもかかわらず、輻輳ウィンドウが増加し続ける。最終的に、1 ウィンドウ内のすべての ACK パケットが損失すると、タイムアウトが発生し輻輳ウィンドウサイズが 1 [pkt] となる。このとき、輻輳状態は解消されていないため、アクセスポイントのバッファは一杯であり、新たに有線端末から送信された ACK パケットはアクセスポイントでのバッファ溢れによって廃棄されるため無線端末までほとんど到達しない。以上により、一旦小さくなった輻輳ウィンドウサイズの回復が行われず、深刻な不公平を生じたと考えられる。

また、この現象は、アクセスポイントにおいて ACK パケットが廃棄されるほどバッファにパケットが蓄積されているために生じている。このため、既にこのような輻輳が発生している無線 LAN 環境に新たにフローが参加しようとした場合でも、同様な現象が生じると考えられる。

次に、TCP Vegas を用いた場合の、アクセスポイントに蓄積される全 ACK パケットの数の変化を図 6 に示す。図 6 から、アクセスポイントのバッファにおいて、ACK パケットの総数が約 45 [pkts] に保たれており、廃棄が発生していないことがわかる。TCP Vegas のような delay-based 手法は、ボトルネックルータ内に蓄積されるパケット数を一定にするよう動作するため、上述の loss-based 手法や hybrid 手法のように上りフロー間の

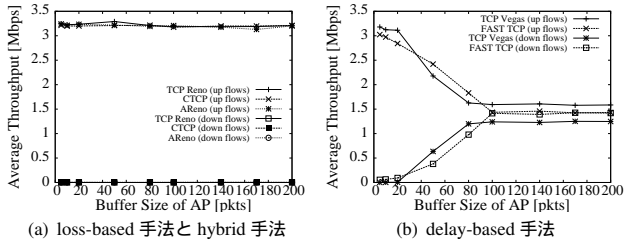


図7 平均スループット(上りフロー数: 8, 下りフロー数: 8, 伝搬遅延: 100 [msec])

不公平が発生しないといえる。

以上から, loss-based 手法や最終的に loss-based 手法として動作する. hybrid 手法では, アクセスポイントにおいて ACK パケットが大量に廃棄されることに起因して, 上りフロー間に深刻な不公平が生じることがいえる。

### 3.2.3 上下フロー間の公平性

次に, 上下フローが混在する環境における評価結果を示す. 各 TCP 改良手法を用いた場合について, 上下フローをそれぞれ 8 本ずつ発生させた場合の, loss-based 手法および hybrid 手法における上下フローの平均スループットを図 7(a), delay-based 手法における同様の結果を図 7(b) に示す. ここでは, アクセスポイントのバッファサイズを変化させた場合の結果を示している. なお, 平均スループットは, TCP の動作が安定した時点以降を見るため, シミュレーション後半の 160 [sec] の間におけるスループットを用いている。

図 7(a) から, loss-based 手法および hybrid 手法においては, フロー数やアクセスポイントのバッファサイズによらず, 上下フロー間に深刻な不公平が生じていることがわかる. これは以下の理由による. 図 1 のようなネットワーク環境下において, アクセスポイントが混雑すると, 上りフローにおいてはアクセスポイントで ACK パケットが廃棄されるが, 下りフローにおいてはアクセスポイントでデータパケットが廃棄される. このとき, データパケットが損失した下りフローは輻輳制御を行うが, ACK パケットが損失した上りフローは輻輳制御を行わない. したがって, 上りフローは輻輳ウィンドウを減少させないが, 下りフローは輻輳制御により輻輳ウィンドウを減少させる. このため, 上下フロー間に深刻な不公平が生じる。

また, 図 7(b) から, delay-based 手法の場合においては, バッファサイズが 100 [pkts] 未満の場合には, 上下フロー間に不公平が生じているが, 100 [pkts] を越えると公平になることがわかる. これは, アクセスポイントのバッファが小さい場合においては, delay-based 手法のパラメータがその環境に対して不適切であったため, loss-based 手法や hybrid 手法と同様な原因で, 上下フロー間に不公平が生じたと考えられる。

## 4. ACK の損失に対する輻輳制御手法

本章では, loss-based 手法や delay-based 手法に対して, 3. で示した不公平の原因となっている 大量の ACK パケットの損失に対して輻輳制御を行うことにより不公平性を軽減する手法を提案する。

### 4.1 提案手法

3. 章で示した TCP フロー間の不公平性は, ACK パケットが大量に廃棄される環境においても, TCP が輻輳ウィンドウを大きくし続けることが主な原因である. そこで, 提案手法では, ACK パケットの損失に対して輻輳制御を行う. 通常のデータパケットの損失に対する輻輳制御は, 3 つの重複 ACK の受信あるいはタイムアウトの発生によって行われる. これに対して, ACK パケットの損失に対する輻輳制御は, 送信側 TCP が受信する ACK パケットのシーケンス番号を監視することで行う. 受信した ACK パケットのシーケンス番号が 1 パケット分ずつ増加する場合は ACK パケットは廃棄されていないと判断し, また, シーケンス番号の増加が 1 以上である場合は ACK パケットが損失したと判断する. このとき, シーケンス番号の増加幅を廃棄された ACK パケット数として計上する. さらに, 1 RTT 中に観測された ACK パケットの損失数が閾値 `thresh_ack_losses` 以上である場合に, 輻輳が発生していると判断する. このとき,

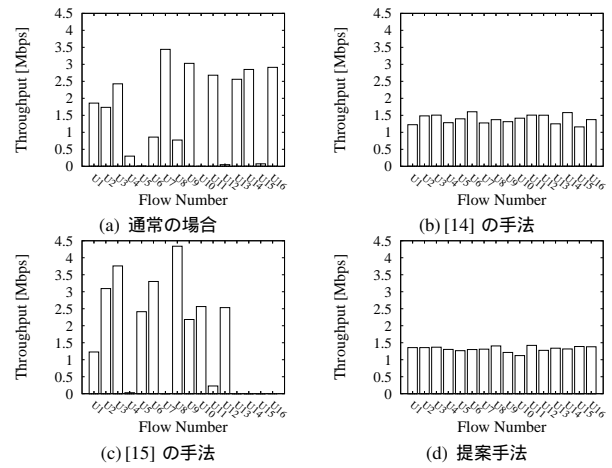


図8 平均スループット(上りフロー数: 16, 伝搬遅延: 100 [msec])

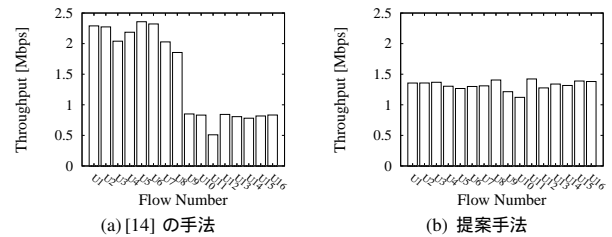


図9 アクセスポイントからの距離を変更した場合の平均スループット(上りフロー数: 16, 伝搬遅延: 100 [msec])

1 RTT の間, 輻輳を誤検出していないか待機する. 具体的には, 現在受信している最大シーケンス番号より小さいシーケンス番号の ACK パケットを受信した場合には, パケット順序の入れ替えなどによる誤検出があったと認識し, ACK パケットの損失数から 1 パケット分差し引く. また, データパケットの損失を検出した場合においては, その後の 1 RTT の間は, ACK パケットの損失数の計上を行わない. 輻輳を検出すると, 現在の輻輳ウィンドウサイズを半減する。

### 4.2 シミュレーション評価

提案手法の性能評価を ns-2 によるシミュレーションによって行う. 本稿における評価では, 提案手法は TCP Reno に対して適用する. また, 性能比較のために, アクセスポイントにおける既存の改善手法として, MAC プロトコルのパラメータを変更する手法 [14] およびキュー管理機構を変更する手法 [15] を用いた場合のシミュレーション結果を合わせて示す. なお, [15] の手法においては広告ウィンドウサイズを 160 [pkts] に設定し, ACK パケット用およびデータパケット用のバッファとして, 50 [pkts] を割り当てた. 提案手法の閾値 `thresh_ack_losses` はリンクエラーを考慮し, 2 とした. ネットワークトポロジ等の他の設定は 3. 章と同じである。

上りフローを 16 本発生させた場合の各フローの平均スループットを図 8 に示す. 図 8 から, 改善手法を適用していない場合においては, 深刻な不公平が発生していることがわかる. これに対して, 提案手法や [14] の手法では上りフロー間の公平性が大幅に改善されていることがわかる. しかし, [15] の手法においては, 改善手法を適用していない場合と同様に深刻な不公平が発生している. これは, [15] の手法がアクセスポイントのバッファをデータパケット用と ACK パケット用の 2 つに分割するが, ACK パケットのバッファにおいて, 大量に ACK パケットが廃棄される状況が改善されないため, 深刻な不公平が発生したと考えられる。

次に, [14] の手法および提案手法において, アクセスポイントから 1 [m] 離れた位置 (U1 から U8) と 10 [m] 離れた位置 (U9 から U16) にそれぞれ 8 台の無線端末を配置し, 上りフローを発生させた場合の平均スループットを図 9 に示す. 図 9 から, 提案手法においては上りフロー間の公平性を維持しているが, [14] の手法においてはアクセスポイントに距離に近いフローと遠い

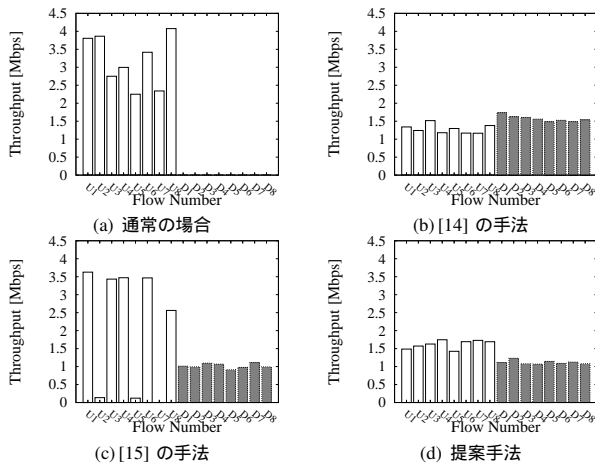


図 10 平均スループット (上りフロー数: 8, 下りフロー数: 8, 伝搬遅延: 100 [msec])

フローとの間に大きな不公平が生じていることがわかる。これは以下の理由によるものだと考えられる。[14]の手法においては、ACK パケットの損失が発生しても輻輳制御が行われないために、輻輳ウィンドウサイズが増加し続ける。その結果、無線 LAN 部分の輻輳状態が悪化する。その際、アクセスポイントから遠い端末がより不利となるため、その輻輳状態の影響を受けやすく、無線区間における伝送遅延が増加する。このため、アクセスポイントから遠い端末ほどスループットが低下し、上りフロー間に不公平が発生したと考えられる。提案手法においては、輻輳制御により無線 LAN 内の輻輳が抑えられるため、このような不公平は発生しない。

次に、上下フローをそれぞれ 8 本ずつ発生させた場合の各フローの平均スループットを図 10 に示す。図 10 から、改善手法を適用していない場合においては、上下フロー間だけでなく上りフロー間においても深刻な不公平が生じていることがわかる。これに対して、[14]の手法においては、上下フロー間の不公平が改善されていることがわかる。[15]の手法においては、下りフロー間では不公平を改善しているが、上りフロー間に関しては深刻な不公平が発生している。これは図 8 と同様な原因によるものだと考えられる。提案手法においては、上下フロー間の公平性を改善しているものの、下りフローは上りフローの約 2/3 倍のスループットでも深くなることがわかる。これは、以下の理由による。提案手法においては、リンクエラーによる ACK パケットの廃棄の発生などの、フロー間の公平性に無関係な要因による誤動作を防止するため、閾値  $\text{thresh\_ack\_losses}$  として 2 を設定した。このため、アクセスポイントの輻輳によって、1 RTT の間に ACK パケットが 1 つ廃棄された場合においては輻輳の検出ができない。このため、上下フロー間のスループットが完全に公平にならなかったと考えられる。しかし、提案手法は上下フロー間のスループットの改善に関して [14] と同程度の改善効果があることがわかる。

## 5. おわりに

本稿では、高速・高遅延ネットワーク環境向けに近年提案されている、RTT 情報を輻輳制御に用いる TCP 改良手法の、無線 LAN 環境における性能をシミュレーションにより評価した。

シミュレーション結果から、無線区間の遅延変動が各 TCP 改良手法が実装レベルで実装している、RTT 情報のフィルタリング処理や、無線端末の送信バッファにおけるデータパケットのバッファリング効果によって、TCP スループットに対してその影響が小さいことがわかった。また、loss-based 手法および hybrid 手法においては、上りフロー間や上下フロー間に深刻な不公平が発生するが、delay-based 手法においては、ボトルネックルータ内においてパケットを一定に保つため、そのような不公平が発生しないことがわかった。

さらに本稿では、loss-based 手法および hybrid 手法において発生するフロー間の不公平に対して、ACK パケットの損失に対して輻輳制御を行う手法を提案し、その評価を行った。シミュ

レーション結果から、提案手法によって上りフロー間の不公平を改善できるだけでなく、上下フロー間の不公平に対しても一定の改善効果があることがわかった。

本稿において、無線 LAN 環境として無線リンクエラーが発生しない理想的な環境を想定してシミュレーションを行っていた。今後の予定として、無線リンクエラーを考慮したシミュレーション評価を行いたい。また、さらなる公平性の改善についても今後の課題としたい。

## 文 献

- [1] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments 3649 (Experimental)*, Dec. 2003.
- [2] T. Kelly, "Scalable TCP: Improving performance in high-speed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, Apr. 2003.
- [3] C. Jin, D. X. Wei, and S. H. Low, "FAST TCP: Motivation, architecture, algorithms, performance," in *Proceedings of IEEE INFOCOM 2004*, Mar. 2004.
- [4] 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.
- [5] H. Shimonishi, T. Hama, and T. Murase, "TCP-Adaptive Reno: Improving efficiency-friendliness tradeoffs of TCP congestion control algorithms," in *Proceedings of PFLDnet 2006*, Feb. 2006.
- [6] I. Rhee and L. Xu, "CUBIC: A new TCP-friendly high-speed TCP variant," in *Proceedings of PFLDnet 2005*, Feb. 2005.
- [7] L. S. Brakmo and L. L. Peterson, "TCP Vegas: End to end congestion avoidance on a global Internet," *IEEE Journal on Selected Areas in Communications*, vol. 13, no. 8, pp. 1465–1480, 1995.
- [8] 長谷川 剛, 村田 正幸, "高速・高遅延ネットワークのためのトランスポート層プロトコル," 電子情報通信学会技術研究報告 (IN2006-169), vol. 106, pp. 41–46, Feb. 2007.
- [9] H. Shimonishi, M. Sanadidi, and T. Murase, "Assessing interactions among legacy and high-speed TCP protocols," in *Proceedings of PFLDnet 2007*, pp. 91–96, Feb. 2007.
- [10] IEEE 802.11b, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications: Higher-Speed Physical Layer Extension in the 2.4 GHz Band*. IEEE, Feb. 1999.
- [11] IEEE 802.11a, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) specifications High-speed Physical Layer in the 5 GHz Band*. IEEE, Feb. 1999.
- [12] IEEE 802.11-2007, *Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications*. IEEE, June 2007.
- [13] S. Pilosof, R. Ramjee, D. Raz, Y. Shavitt, and Prasun, "Understanding TCP fairness over wireless LAN," in *Proceedings of IEEE INFOCOM 2003*, 2003.
- [14] Y. Fukuda and Y. Oie, "Unfair and inefficient share of wireless LAN resource among uplink and downlink data traffic and its solution," *IEICE Transactions on Communications*, vol. E88-B, pp. 1577–1585, Apr. 2005.
- [15] J. Ha and C.-H. Choi, "TCP fairness for uplink and downlink flows in WLANs," in *Proceedings of IEEE Global Telecommunications Conference 2006*, pp. 1–5, Nov. 2006.
- [16] F. Keceli, I. Inan, and E. Ayanoglu, "TCP ACK congestion control and filtering for fairness provision in the uplink of IEEE 802.11 infrastructure basic service set," in *Proceedings of IEEE International Conference on Communications*, June 2007.
- [17] N. Blefari-Melazzi, A. Detti, I. Habib, A. Ordine, and S. Salsano, "TCP fairness issues in IEEE 802.11 networks: Problem analysis and solutions based on rate control," in *Proceedings of IEEE Transactions on Wireless Communications*, vol. 6, pp. 1346–1355, Apr. 2007.
- [18] "The network simulator ns-2," available at <http://www.isi.edu/nsnam/ns/>.
- [19] 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.
- [20] "FAST TCP simulator module for ns-2," available at <http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/>.