特別研究報告

題目

無線 LAN 環境における
遅延に基づく輻輳制御を用いた TCP の性能評価

指導教員
中野 博隆 教授

報告者
橋本 匡史

平成 20 年 2 月 19 日

大阪大学 基礎工学部 情報科学科

平成 19 年度 特別研究報告

無線 LAN 環境における
遅延に基づく輻輳制御を用いた TCP の性能評価

橋本 匡史

内容梗概

近年，インターネットが様々なネットワーク技術により大規模化・高速化するとともに，無線 LAN によるインターネットアクセスが一般化している．無線 LAN 規格には，無線伝送速度が最大 11 [Mbps] である IEEE 802.11b や無線伝送速度が最大 54 [Mbps] である IEEE 802.11a などが標準化され一般に利用されている．さらに，無線伝送速度が最大 300 [Mbps] を越える IEEE 802.11n も標準化が進み，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．

一方，高速・高遅延ネットワークを対象に従来の TCP Reno に替わる TCP 改良手法が数多く提案されている．それらの提案手法の中には，TCP コネクションのラウンドトリップ時間 (RTT) をネットワークの輻輳の指標として用いる手法 (delay-based 手法) がある．Delay-based 手法はパケット廃棄の発生を指標として用いる手法に比べ，早期に輻輳を検出することにより高いスループットを得ることができる．しかし，そのような手法は，無線ネットワークのような RTT がネットワークの輻輳とは関係なく変動する環境では正しく動作しないことが予想される．しかし，無線端末においてネットワーク環境に応じ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く，無線ネットワーク環境が今後も高速化される傾向にあるため，delay-based 手法がそのまま利用される可能性が十分ある．また，無線 LAN は上りと下りでネットワーク帯域を共有するため，特に無線端末が TCP 送信側になる場合には，アクセスポイントが輻輳を起こすことが考えられる．しかし，この問題が TCP の性能に与える影響はこれまでほとんど検証されていない．

そこで本報告では，無線 LAN 環境において，RTT をネットワークの輻輳の指標として用いる delay-based 手法の性能評価をシミュレーションによって行う．その結果，無線端末が送信側である場合において，無線端末のバッファにデータパケットが蓄積することや TCP 改良手法が実装レベルで RTT 情報をフィルタリング処理することによって， RTT の変動の影響を抑えることができることを示す．また，無線アクセスポイントに ACK パケットが蓄積することが原因となり TCP コネクション間に深刻な不公平性が発生することを明らかにする．

目 次

図目次

表 目 次

1 はじめに

近年のネットワークの高速化にともない，Transmission Control Protocol (TCP) [1] 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 [2]．TCP Reno は輻輳ウィンドウサイズの増加幅がラウンドトリップ時間 (RTT) ごとに 1 パケットと小さいにもかかわらず，パケット廃棄を検出した際の輻輳ウィンドウサイズの減少幅が輻輳ウィンドウの 1/2 以上と大きいため，帯域遅延積の大きい高速・高遅延ネットワークでは輻輳ウィンドウサイズが十分に大きくならない．さらに，帯域遅延積の大きなネットワークのリンク帯域を十分使うためには非常に低いパケット廃棄率である必要がある．このため，高速・高遅延ネットワークにおいて TCP Reno は十分なスループットを得ることができない．

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 [2–7]．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，ネットワークの輻輳の指標として RTT を用いる delay-based 手法，およびそれらを組み合わせた手法の 3 つに大別される [8]．Loss-based 手法に分類される TCP 改良手法としては High-Speed TCP [2] や Scalable TCP [7] などがある．また，delay-based 手法に分類される TCP 改良手法には TCP Vegas [9] や FAST TCP [3] などが挙げられる．Loss-based 手法と delay-based 手法を組み合わせた手法に分類される TCP 改良手法には Compound TCP (CTCP) [4] や TCP-Adaptive Reno (AReno) [5] などがある．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 [10].

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となってきている．無線 LAN 規格として，IEEE によって標準化された IEEE 802.11b [11] や IEEE 802.11a [12] 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 では無線伝送速度が最大 300 [Mbps] を越えるなど，さらなる高速化が行われている．それらの無線 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，搬送波を検出するランダムな時間である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プロトコルである．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．

無線ネットワークは有線ネットワークに比べ，インフラ整備が簡単である反面，通信の信頼性が低いことや遅延ジッタが大きいことが問題となっている．このような無線ネットワーク環境において，TCP を利用すると，その輻輳制御方式の特徴により，リンクエラーによる

パケット廃棄をネットワークの輻輳として認識してしまうという問題がある [13]．この問題に対しても，さまざまな TCP 改良手法が提案されている [14, 15]．しかし，無線 LAN 環境における遅延ジッタが TCP の輻輳制御に与える影響については，ほとんど検証されておらず，前述の高速・高遅延ネットワークのための TCP 改良手法の多くに対しても，検証は行われていない．これは，delay-based 手法が比較的新しい TCP 改良手法であり，高速・高遅延ネットワークを対象に考案されているためだと考えられる．

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 が変動すると，ネットワークの輻輳状態を正しく得ることができないことが予想される．しかし，無線端末において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 手法の評価を行うことは重要である．

また，無線 LAN は上りと下りでネットワーク帯域を共有するため，特に無線端末が TCP 送信側になる場合には，アクセスポイントが輻輳を起こすことが考えられる．しかし，この問題が TCP の性能に与える影響は検証されていない．

そこで本報告では，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 を用いる delay-based 手法の性能評価をシミュレーションによって行う．無線 LAN 規格として IEEE 802.11b および IEEE 802.11a を利用し，ボトルネックリンクが無線区間にある場合と有線区間にある場合の両方について，ns-2 [16] を用いてシミュレーションを行った結果を示す．シミュレーション結果から，無線端末が送信側であるようなネットワーク環境では，無線端末のバッファにパケットが蓄積することや各 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って，無線 LAN の遅延変動の影響を抑えることができることを示す．さらに，無線端末が送信側となった場合に，無線アクセスポイントに ACK パケットが蓄積されることによって，TCP コネクション間の公平性に深刻な問題が引き起こされることを述べる．

以下，2 章では輻輳制御に RTT を用いる loss-based 手法や delay-based 手法，およびそれらを組み合わせた手法の輻輳制御方式，および観測した RTT の輻輳制御への利用の仕方について簡単に述べる．3 章においては，ボトルネックリンクが無線区間にある場合および有線区間にある場合のそれぞれについてシミュレーションを行った結果を示し，無線 LAN 環境がさまざまな TCP 改良手法に与える影響について考察する．最後に，4 章で本報告のまとめと今後の課題について述べる．

2 RTT を利用する TCP 改良手法

本章では，RTT を利用する TCP 改良手法の輻輳制御方式のうち，輻輳ウィンドウサイズの更新アルゴリズムについて述べる．また，観測した RTT 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．本報告では delay-based 手法として TCP Vegas および FAST TCP，loss-based 手法と delay-based 手法を組み合わせた手法として CTCP および AReno，そして，loss-based 手法として CUBIC を取り上げる．

どの delay-based 手法においても，方式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 が利用されることは記述されている．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，本報告では 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 TCP Vegas [9]

TCP Vegas は高速・高遅延向けの TCP 改良手法ではないが，多くの delay-based 手法の基本となる輻輳ウィンドウ更新アルゴリズムを使用している．TCP Vegas は以下の式によって，ネットワーク上に蓄積されているパケット数を推測する．

$$\begin{aligned} Diff &= (Expected - Actual) \cdot baseRTT \\ Expected &= cwnd/baseRTT \\ Actual &= cwnd/RTT \end{aligned} \tag{1}$$

なお，$cwnd$ 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，[9] には明記されておらず，Linux 実装においてはラウンドトリップ時間で計測された最小の 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，以下の式にしたがって輻輳ウィンドウサイズを RTT に 1 度増減させる．

$$cwnd \leftarrow \begin{cases} cwnd + 1 & (Diff < \alpha_v) \\ cwnd & (\alpha_v \leq Diff \leq \beta_v) \\ cwnd - 1 & (Diff > \beta_v) \end{cases} \tag{2}$$

ここで，α_v と β_v は制御パラメータである．式 (2) は，TCP Vegas がボトルネックリンクに蓄積されているパケット数が α_v 以上 β_v 以下になるように輻輳ウィンドウサイズを制御することを意味する．

2.2 FAST TCP [3]

FAST TCP は TCP Vegas と同様に，今までに観測された最小の 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サイズの増減を RTT に 1 回行う．

$$cwnd \leftarrow \min\left\{2cwnd,\ (1-\gamma_f)\,cwnd + \gamma_f\left(\frac{baseRTT}{avgRTT}cwnd + \alpha_f\right)\right\} \tag{3}$$

ここで，$avgRTT$ は以下の式のように輻輳ウィンドウサイズの 3 分の 1 の重みで指数移動平均化された RTT である．なお，$avgRTT$ は [3] においては明確に指定されておらず，ns-2 のシミュレーションコード [17] においては以下の式のように定義されている．

$$avgRTT = ((cwnd/3 - 1)\cdot avgRTT + RTT)\,/\,(cwnd/3) \tag{4}$$

ここで，RTT は観測された現在の RTT である．FAST TCP はネットワークの帯域遅延積が大きい場合でもすばやく輻輳ウィンドウを増加することができるため，TCP Vegas に比べネットワークの利用率を向上できる．

2.3 Compound TCP (CTCP) [4]

TCP Vegas や FAST TCP のような純粋な delay-based 手法は，TCP Reno などの loss-based 手法と混在するネットワーク環境において， loss-based 手法に比べてスループットが低下するという問題がある [18]．この問題に対して，loss-based 手法が混在する環境においても，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することができる手法として，loss-based 手法と delay-based 手法を組み合わせた手法が提案されている．

CTCP は loss-based 手法と delay-based 手法を組み合わせた手法の 1 つである．CTCP は TCP Vegas と同様に以下の式により，ネットワーク上に蓄積されているパケット数を推測する．

$$\begin{aligned} Diff &= (Expected - Actual)\cdot baseRTT \\ Expected &= cwnd/baseRTT \\ Actual &= cwnd/RTT \end{aligned} \tag{5}$$

ここで，$baseRTT$ はいままで観測された最小の RTT，$Diff$ はネットワーク上で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．なお，式 (5) で利用している RTT をどう設定するかは [4] には明記されておらず，Linux における実装ではラウンドトリップ時間で計測された最小の RTT を $minRTT$ とし，式 (5) の RTT として用いている [19]. 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 度増減させる．

$$\begin{aligned} cwnd &\leftarrow cwnd_{reno} + dwnd \\ dwnd &\leftarrow \begin{cases} dwnd + \max(\alpha_c \cdot cwnd^k - 1,\ 0) & (Diff < \gamma_c) \\ \max(dwnd - \zeta_c \cdot Diff,\ 0) & (Diff \geq \gamma_c) \end{cases} \end{aligned} \tag{6}$$

ここで，$cwnd_{reno}$ 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，α_c，ζ_c および γ_c は制御パラメータである．CTCP は式 (6) により，ネットワーク帯域が使いきれていないと判断した場合は，ネットワークに蓄積されているパケット数が γ_c 程度になるように輻輳ウィンドウサイズを増加させ制御する．さらに，ネットワーク帯域が使い切られ輻輳が発生した状態になると，$dwnd$ が 0 になり，最終的に輻輳ウィンドウサイズの増加速度が TCP Reno と同じになる．

2.4 TCP-Adaptive Reno (AReno) [5]

AReno は CTCP と同様に loss-based 手法と delay-based 手法を組み合わせた手法である．AReno の CTCP との相違点は，CTCP が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リンク帯域を推定し，それを用いて輻輳ウィンドウサイズの増減幅を決定する点である．

AReno は以下の式によってネットワークの輻輳レベルを推定する．

$$\begin{aligned} c &= \min\left(\frac{sRTT - RTT_{min}}{RTT_{cong} - RTT_{min}},\ 1\right) \\ RTT_{cong} &= (1-a)RTT_{cong} + aRTT \end{aligned} \tag{7}$$

ここで，c はネットワークの輻輳レベル，RTT_{min} はいままで観測された最小の RTT，RTT_{cong} は式 (8) 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 (7) 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 $sRTT$ を利用している．

AReno は c 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の増減を ACK パ

ケットを受信するごとに行う．

$$
\begin{aligned}
cwnd &\leftarrow w_{base} + w_{probe} \\
w_{base} &\leftarrow w_{base} + 1\ MSS/cwnd \qquad (8) \\
w_{probe} &\leftarrow \max(w_{probe} + W_{inc}/cwnd,\ 0) \qquad (9) \\
W_{inc}(c) &= W_{inc}^{max}/e^{\alpha_a c} + \beta_a c + \gamma_a \\
W_{inc}^{max} &= B/M{\cdot}MSS \\
\beta_a &= 2W_{inc}^{max}(1/\alpha_a - (1/\alpha_a + 1)/e^{\alpha_a}) \\
\gamma_a &= 1 - 2W_{inc}^{max}(1/\alpha_a - (1/\alpha_a + 1/2)/e^{\alpha_a})
\end{aligned}
$$

ここで，MSS は最大セグメントサイズ，B は TCP Westwood [14] と同様に ACK の到着間隔から求められるボトルネックリンク帯域の推定値，M はスケーリング係数である．また，式 (8) は TCP Reno に相当するウィンドウサイズであり，式 (9) は delay-based 手法に相当するウィンドウサイズである．AReno は c によりネットワーク帯域が使い切れていないと判断すれば w_{probe} の増加量を大きくし，ネットワーク帯域を使い切るように動作する．さらに，c に連動して w_{probe} を変化させることにより輻輳ウィンドウを一定に保つように動作し，最終的には TCP Reno と輻輳ウィンドウサイズの増加速度が同じになる．

2.5 CUBIC [6]

CUBIC はパケット廃棄が発生してからの経過時間を用いて，以下の式にしたがって輻輳ウィンドウサイズを RTT とは独立な一定時間ごとに 1 回更新する．

$$
cwnd \leftarrow C(t - \sqrt[3]{W_{max}\beta_{cubic}/C})^3 + W_{max} \qquad (10)
$$

ここで，C はスケーリング係数，t はパケット廃棄が発生してからの経過時間，β_{cubic} は倍数減少係数，W_{max} はパケット廃棄が発生する直前の輻輳ウィンドウサイズである．CUBIC は式 (10) のように上述の TCP 改良手法とは異なり，輻輳ウィンドウサイズの更新間隔や増減量が RTT に依存しない．このため，RTT が変動しても輻輳ウィンドウサイズの増減に影響がない．

3 シミュレーション評価

本章では，ns-2 [16] を用いて，2 章で紹介した，純粋な delay-based 手法である TCP Vegas および FAST TCP，loss-based 手法と delay-based 手法を組み合わせた手法である CTCP および AReno，および loss-based 手法である CUBIC を用いて無線端末が TCP の送信側である場合のシミュレーションを行い，無線 LAN 環境におけるそれらの手法の性能について考察する．

シミュレーションにおいて用いる無線 LAN 規格は，IEEE 802.11b および IEEE 802.11a である．それぞれのパラメータを表 1 および表 2 に示す．

TCP Vegas，CTCP，AReno，および CUBIC は NS-2 TCP-Linux [20] を用いることにより，Linux 上の実装コードを ns-2 上で実行できるように変換を行い，シミュレーションを行った．FAST TCP については，公開されているシミュレーションモジュール [17] を用いた．

3.1 ボトルネックが無線区間にある場合

本節では，ボトルネックが無線区間にあるネットワーク環境においてシミュレーションを行い，無線 LAN 環境がボトルネックとなる場合の各 TCP 改良手法への影響について考察を行う．

3.1.1 シミュレーション環境

無線 LAN 規格として IEEE 802.11b を使用し，ボトルネックが無線区間にあるネットワークモデルを図 1 に示す．複数の無線端末が送信側 TCP として 1 つの無線アクセスポイントを共有し，無線アクセスポイントから受信端末までは有線リンクにより接続されている．無線端末から無線アクセスポイントまでは IEEE 802.11b を利用しているため，無線伝送速度は最大 11 [Mbps] の無線リンクである．また，無線アクセスポイントと受信端末までは，帯域が 100 [Mbps] であり，片道伝搬遅延が 200 [msec] の有線リンクで接続されている．このため，このネットワーク環境では無線端末から無線アクセスポイントまでの無線区間がボトルネックとなる．また，無線アクセスポイントのバッファサイズを 100 [pkts] に設定し，無線アクセスポイントから受信端末までの有線リンクにおけるバッファサイズは十分大きな値に設定した．無線端末の送信バッファについても，無線 LAN 環境に影響を与えないように十分大きな値に設定した．なお，無線アクセスポイントから受信端末までの有線リンクのバッファには，キュー管理機構として DropTail を用いた．

また，無線 LAN 規格として IEEE 802.11a を使用し，ボトルネックが無線区間にあるネットワークモデルを図 2 に示す．図 2 は図 1 において，無線 LAN 規格を IEEE 802.11b から

表 1: IEEE 802.11b における設定パラメータ

項目	値
伝送速度	11 [Mbps]
プリアンブル長	144 [μs]
PLCP ヘッダ長	48 [bit]
スロットタイム	20 [μs]
SIFS	10 [μs]
DIFS	50 [μs]
CWmin	31
CWmax	1023

表 2: IEEE 802.11a における設定パラメータ

項目	値
伝送速度	54 [Mbps]
プリアンブル長	16 [μs]
PLCP ヘッダ長	4 [μs]
スロットタイム	9 [μs]
SIFS	16 [μs]
DIFS	34 [μs]
CWmin	15
CWmax	1023

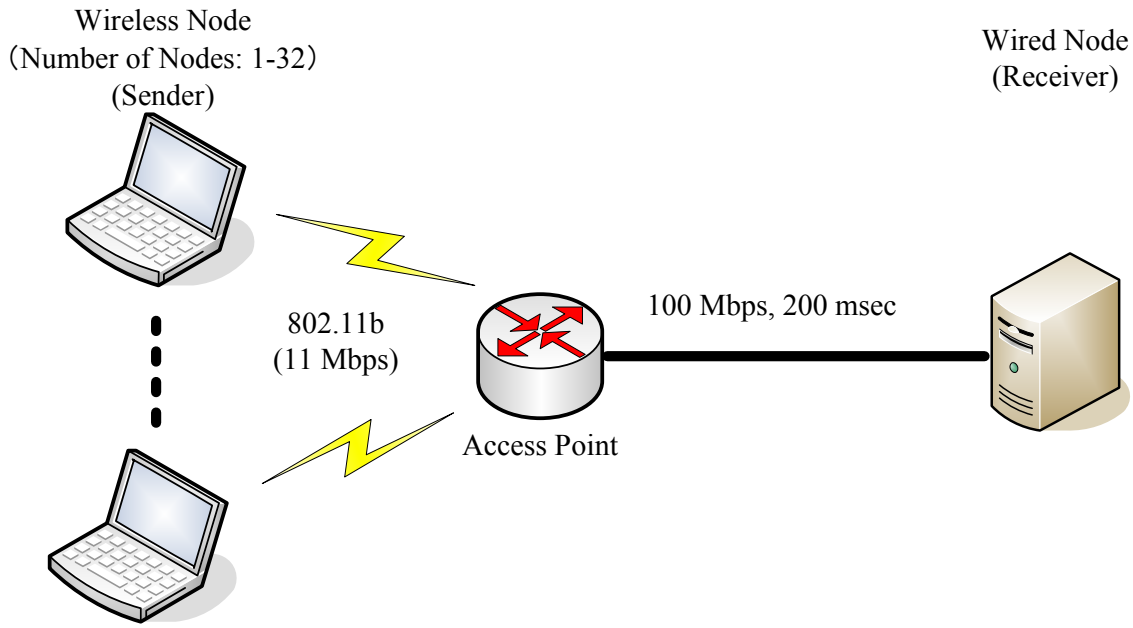


図 1: IEEE 802.11b を用いた場合のボトルネックが無線区間にあるネットワークモデル

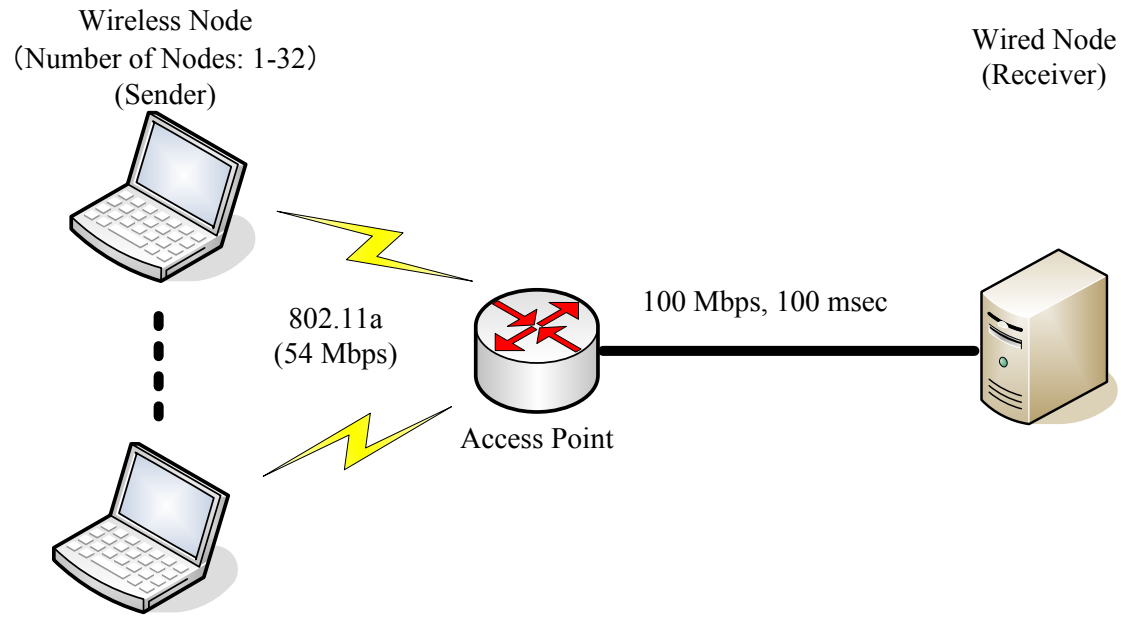


図 2: IEEE 802.11a を用いた場合のボトルネックが無線区間にあるネットワークモデル

IEEE 802.11a に変更し，片道の伝搬遅延を 100 [msec] に変更したネットワーク環境である．さらに，無線アクセスポイントのバッファサイズを 500 [pkts] に変更している．

シミュレーションは図 1 および図 2 のネットワーク環境において，無線端末数を 1 台から 32 台まで変化させ，各無線端末から上り方向の TCP コネクションを無線端末 1 台につき 1 本生成することにより行う．つまり，無線端末数が 2 台であるとき，TCP コネクション数は合計で 2 本である．TCP には，TCP Vegas，FAST TCP，CTCP，AReno，および CUBIC を用いて，それぞれの場合について各無線端末の TCP に同一の TCP 改良手法を設定した．各 TCP コネクションはシミュレーション開始後，0.01 秒間隔で，データ転送を開始した．各無線端末は生成した TCP コネクションを用いて，アプリケーション層プロトコルとして FTP を用いて，無限大のサイズを持つデータ転送を行う．データパケットのサイズは 1500 [pkts] に設定している．

また，各 TCP 改良手法のパラメータは，それぞれ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 $\alpha_c = 0.125, \gamma_c = 30, \zeta_c = 1$

AReno $\alpha_a = 16, \beta_a = 1, \gamma_a = 10$

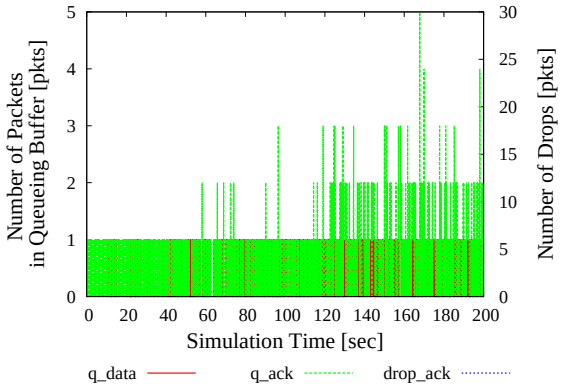
CUBIC 設定項目なし

3.1.2 シミュレーション結果と考察

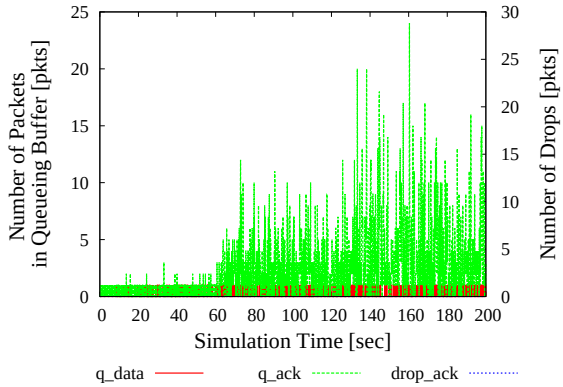
図 1 のネットワークモデルにおいて，コネクション数を 1 から 8 まで増やした場合の無線アクセスポイントのバッファサイズの変化，および無線アクセスポイントにおけるバッファ溢れによるパケット廃棄数の変化を，TCP Vegas の場合を図 3 に， AReno の場合を図 4 に，CUBIC の場合を図 5 にそれぞれ示す． なお，図 3-5 中の q_data は無線アクセスポイントのバッファに蓄積されているデータパケットの数，q_ack は無線アクセスポイントのバッファに蓄積されている ACK パケットの数，drop_ack は無線アクセスポイントのバッファ溢れによって発生したパケット廃棄の数である．

図 3 から図 5 をみると，無線アクセスポイントのバッファにデータパケットではなく ACK パケットが蓄積されていることがわかる．また，コネクション数が増加すると，どの TCP 改良手法においても，無線アクセスポイントに蓄積される ACK パケット数が増加していることがわかる．しかし，図 3 から TCP Vegas の場合は，無線アクセスポイントに蓄積される ACK パケットがコネクション数によってほぼ一定の値になり，ACK パケットの廃棄は発生していない．これは，無線アクセスポイントに ACK パケットが蓄積されることによって RTT が増加するため，delay-based 手法である TCP Vegas は輻輳ウィンドウサイズを制御するためだと考えられる．また，図 5 より，loss-based 手法である CUBIC では，コネクション数が増えると 無線アクセスポイントにおいて ACK パケットの廃棄が発生していることがわかる．図 4(c) より，loss-based 手法と delay-based 手法を組み合わせた手法である AReno は，delay-based 手法で動作している 20 秒までは無線アクセスポイントに蓄積されている ACK パケット数が 20 [pkts] 程度に保たれている．しかし，20 秒以降に loss-based 手法に切り替わるため，CUBIC と同様に，無線アクセスポイントに蓄積されているパケット数が増加し，最終的には ACK パケットの廃棄が発生している．

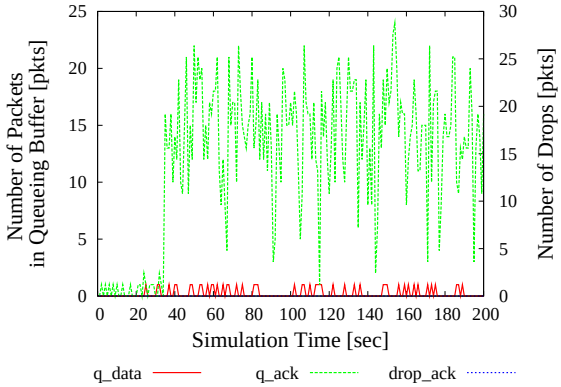
コネクション数が増加すると，どの TCP 改良手法においても，無線アクセスポイントに ACK パケットが蓄積される理由として以下が考えられる．CSMA/CA において，無線端末と無線アクセスポイントの送信機会は平等である．このため，無線端末が増えコネクション数が増加すると，コネクション数が 1 である場合は無線アクセスポイントの送信機会が $1/2$ であったのが，コネクション数が 2，4 と増加すると送信機会が $1/3$，$1/5$ と減少する．無線アクセスポイントにおいて送信機会が減少しても，図 1 のネットワーク環境下の帯域遅延積は変化がないため，無線アクセスポイントに到着する ACK パケット数に変化はない．このため，無線アクセスポイントに ACK パケットが到着する間隔より，無線アクセスポイントから各無線端末へ ACK パケットが送信されるまでの間隔が長くなり，次第に無線端末のバッファに ACK パケットが蓄積されたと考えられる．これは，CSMA/CA を用い，かつ上りと下りで帯域を共有する無線 LAN 環境に特有の現象であるといえる．



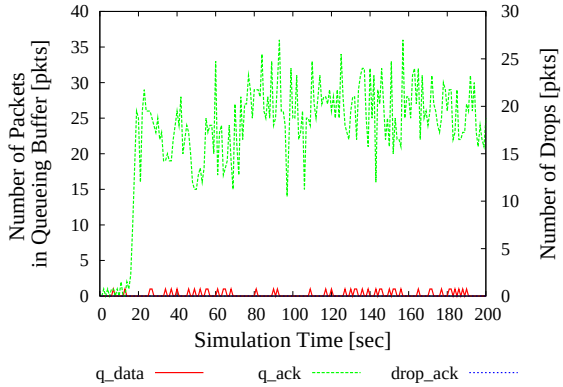
(a) コネクション数が 1 である場合



(b) コネクション数が 2 である場合

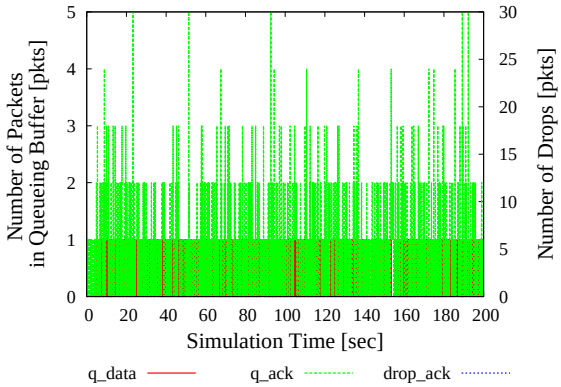


(c) コネクション数が 4 である場合

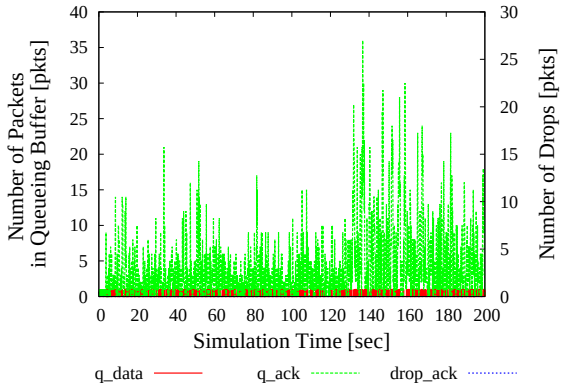


(d) コネクション数が 8 である場合

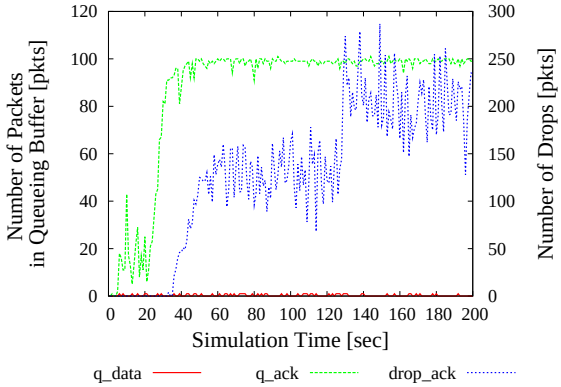
図 3: 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄数の変化 (TCP Vegas, 伝搬遅延: 200 [msec])



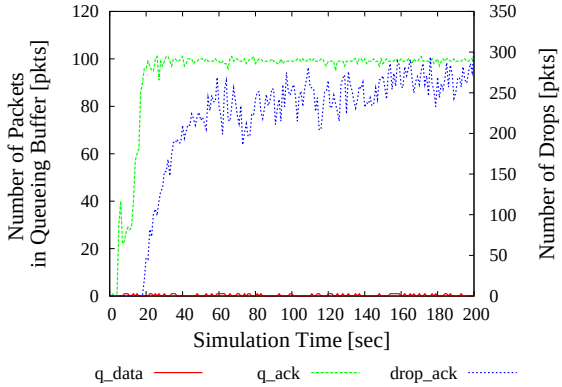
(a) コネクション数が 1 である場合



(b) コネクション数が 2 である場合

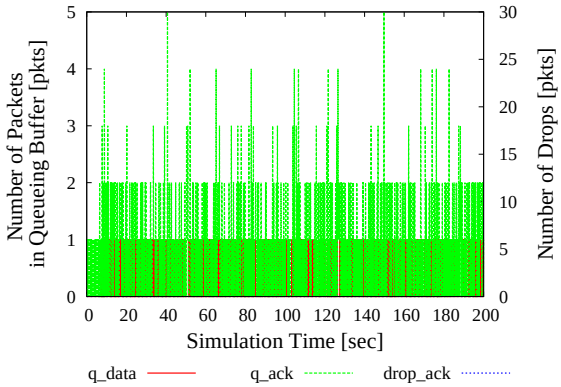


(c) コネクション数が 4 である場合

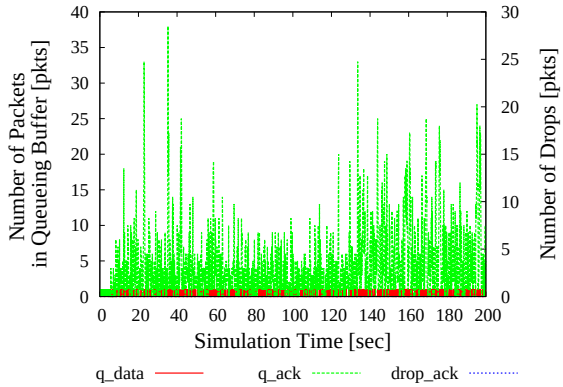


(d) コネクション数が 8 である場合

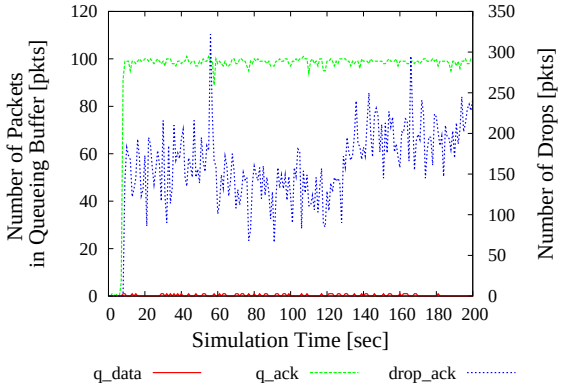
図 4: 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄数の変化 (AReno, 伝搬遅延: 200 [msec])



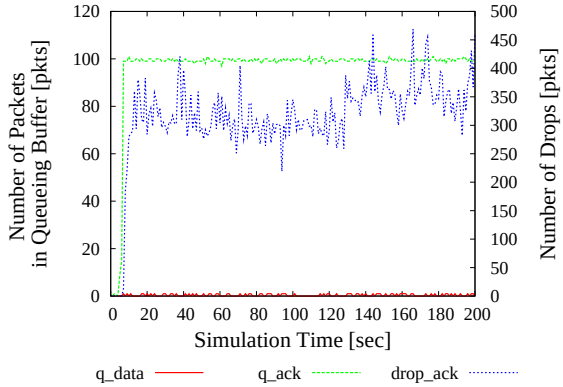
(a) コネクション数が 1 である場合



(b) コネクション数が 2 である場合



(c) コネクション数が 4 である場合



(d) コネクション数が 8 である場合

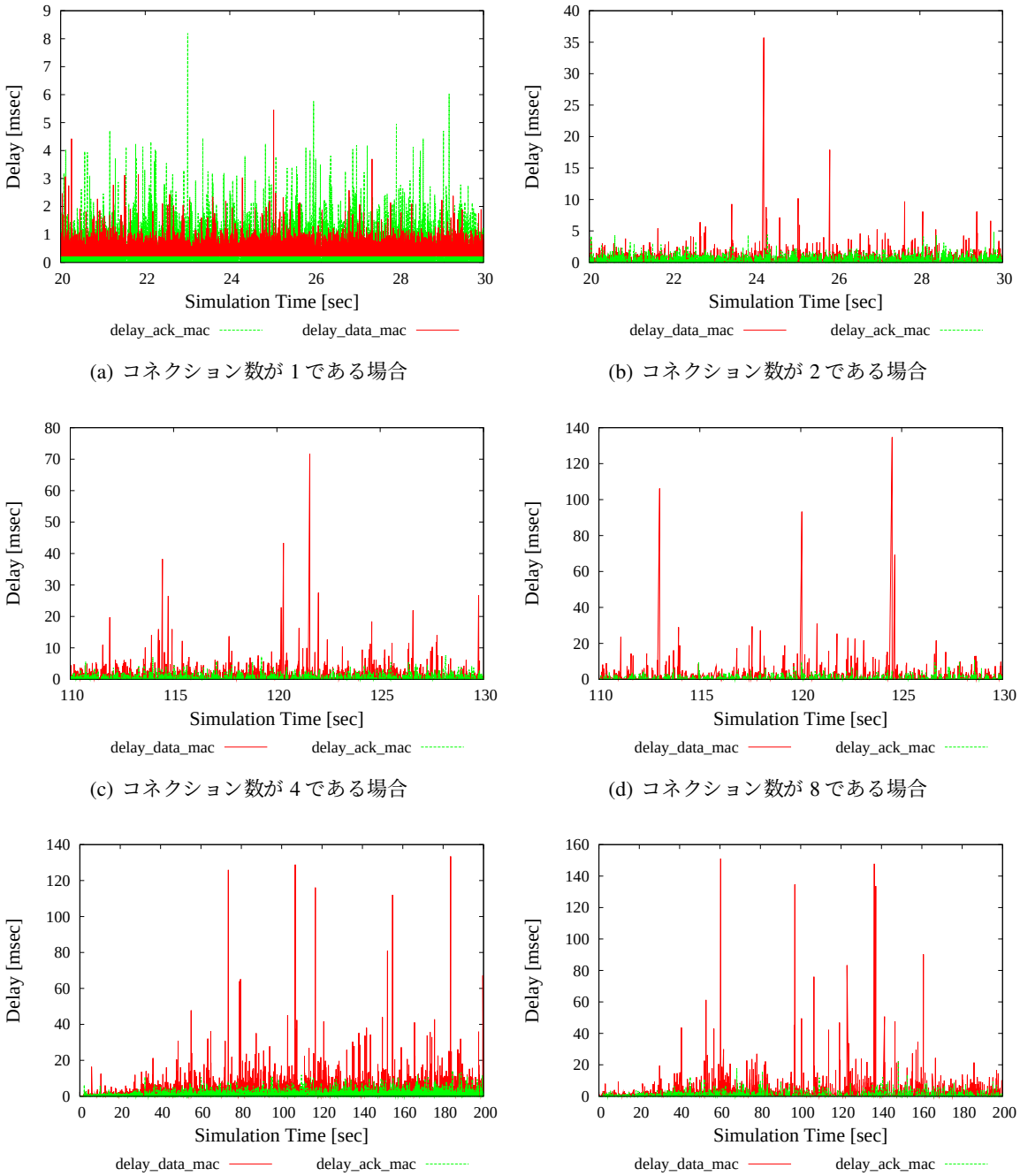
図 5: 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄数の変化 (CUBIC, 伝搬遅延: 200 [msec])

次に，図 2 のネットワーク環境において，AReno を用いた場合の無線区間の遅延変動を図 6 に，無線区間に送出されるパケットの総数の変化を図 7 に，無線アクセスポイントにおけるキュー長の変化およびパケット廃棄数の変化を図 8 にそれぞれ示す．なお，図 6 において，コネクション数が 2 以上の場合においては，無線端末が複数あるが，より無線区間の遅延変動が大きい端末の遅延変動を表している．また，コネクション数によって，図 6 のシミュレーション時間が異なるが，各コネクションの場合においてより遅延変動が大きい時間を選んでいる．図 6 中の delay_data_mac は無線端末からフレームが送信されて無線アクセスポイントにおいて受信されるまでの時間，delay_ack_mac は無線アクセスポイントからフレームが送信されて無線端末において受信されるまでの時間である．

図 6 より，コネクション数が増加しても，delay_ack_mac の変動幅が増加していないのに対し，コネクション数が増加すると delay_data_ack の変動幅が増加していることがわかる．例えば，図 6(a) では，delay_ack_mac の変動幅が最大で 8 [msec] 程度であり，図 6(f) では最大で 20 [msec] 程度であるが，delay_data_mac は図 6(a) では変動幅が最大で 5 [msec] 程度であり，図 6(f) では変動幅が最大で 150 [msec] 程度である．これは，ACK パケットサイズが 40 [pkts] と小さいのに対し，データパケットサイズが 1500 [pkts] であるために，データパケットの方が ACK パケットより伝送にかかる時間が長く，フレームの衝突が発生しやすいためだと考えられる．

図 7(a) をみると，無線区間に送出されるデータパケットの総数と ACK パケットの総数がほぼ同数であることがわかる．図 7(b) のように，コネクション数が増加すると，ある時点でデータパケットの総数が増加し， ACK パケットの総数が減少し始めることが分かる．ここで，図 8 から，コネクション数が 1 である場合はシミュレーション時間内では，無線アクセスポイントにおいて ACK パケットが廃棄されていないが，コネクション数が大きくなると ACK パケットが廃棄されていることがわかる．図 7 と図 8 より，ACK パケットが廃棄され始める時間と無線区間に送出されるデータパケットの総数が増大し ACK パケットの総数が減少する時刻が一致することがわかる．例えば，図 7(c) において，無線区間に送出されるパケットの総数が変動する時間は 40 秒付近であり，図 8(c) より ACK パケットが廃棄され始める時間が 40 秒付近と一致している．このことから，無線区間に送出されるデータパケットの総数が増加し ACK パケットの総数が減少する現象が，無線アクセスポイントにおいて ACK パケットが廃棄されることの原因であるといえる．

この問題が引き起こされる理由は以下によるものだと考えられる．無線アクセスポイントにおいて，ACK パケットが廃棄されることによって，無線端末では，その ACK パケットに対応するデータパケットが受信端末で正しく受信されたという情報を，そのデータパケットが受信端末で受信された以降のデータパケットに対応する ACK パケットを受信することによって知ることになる．TCP は，ACK パケットのヘッダに記載された，データパケットの



(a) コネクション数が 1 である場合

(b) コネクション数が 2 である場合

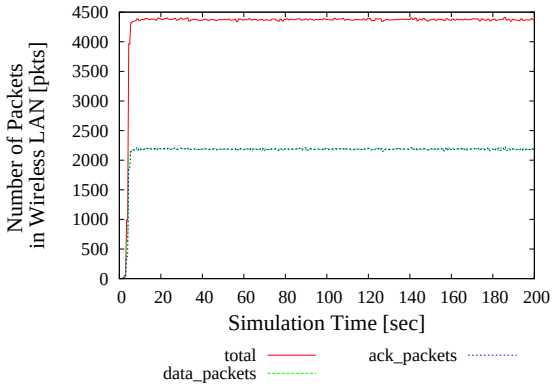
(c) コネクション数が 4 である場合

(d) コネクション数が 8 である場合

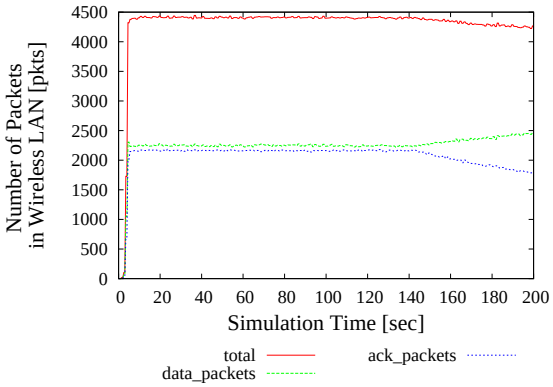
(e) コネクション数が 16 である場合

(f) コネクション数が 32 である場合

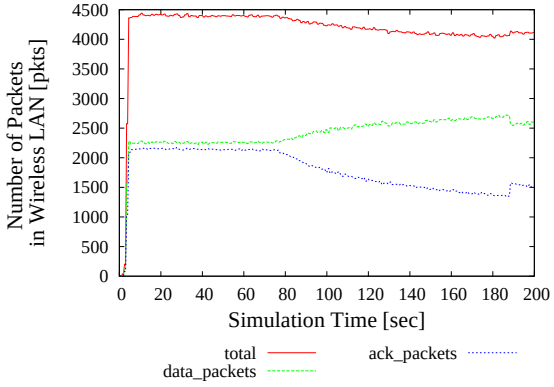
図 6: 無線区間にボトルネックがある場合の無線区間の遅延変動 (AReno, 伝搬遅延: 100 [msec])



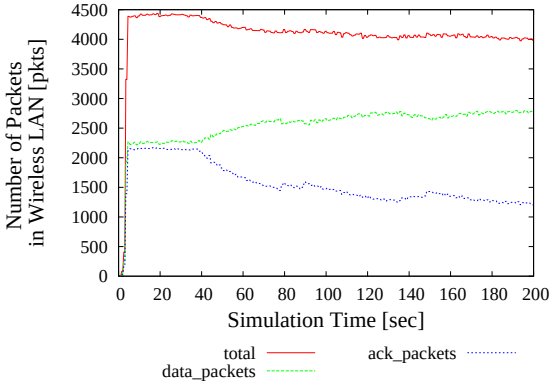
(a) コネクション数が 1 である場合



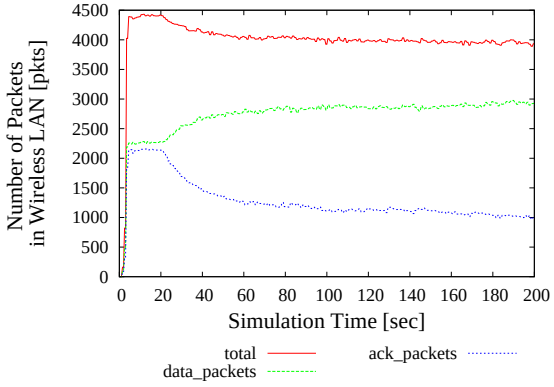
(b) コネクション数が 2 である場合



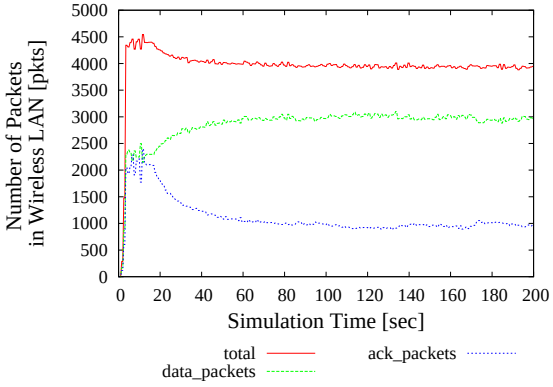
(c) コネクション数が 4 である場合



(d) コネクション数が 8 である場合

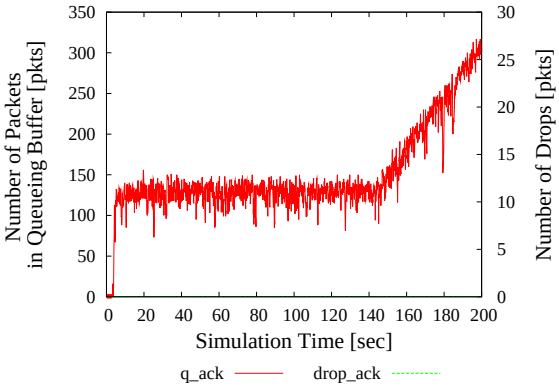


(e) コネクション数が 16 である場合

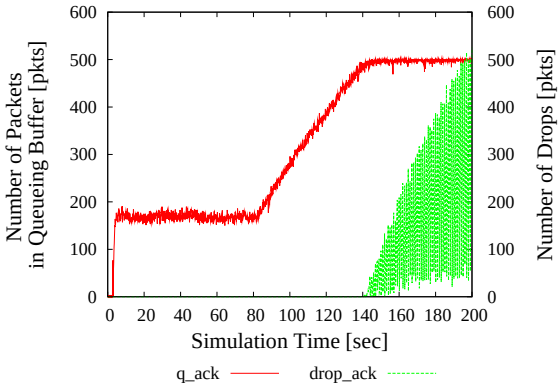


(f) コネクション数が 32 である場合

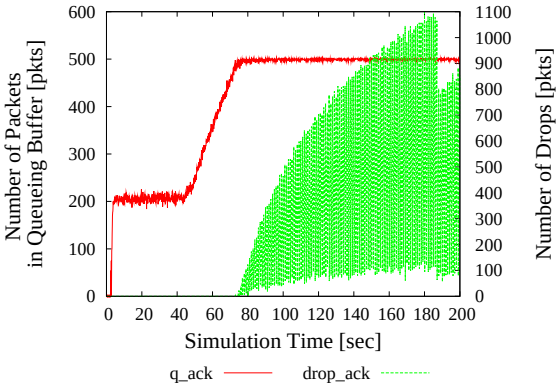
図 7: 無線区間にボトルネックがある場合の無線区間に送出されるパケットの総数の変化 (AReno，伝送遅延: 100 [msec])



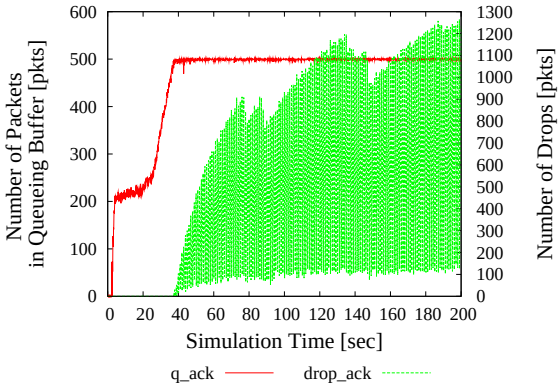
(a) コネクション数が 1 である場合



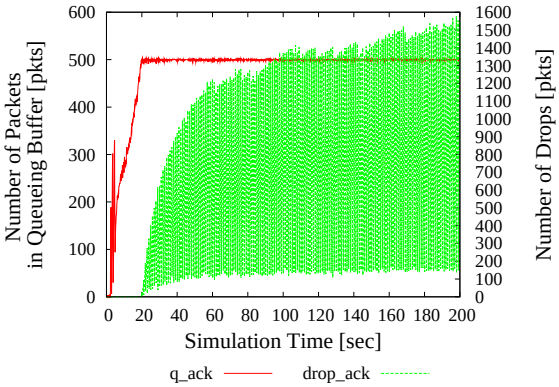
(b) コネクション数が 2 である場合



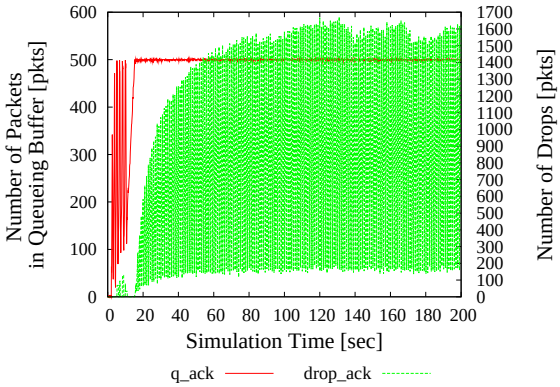
(c) コネクション数が 4 である場合



(d) コネクション数が 8 である場合



(e) コネクション数が 16 である場合



(f) コネクション数が 32 である場合

図 8: 無線区間にボトルネックがある場合の無線アクセスポイントにおけるキュー長の変化およびパケット廃棄数の変化 (AReno，伝送遅延: 100 [msec])

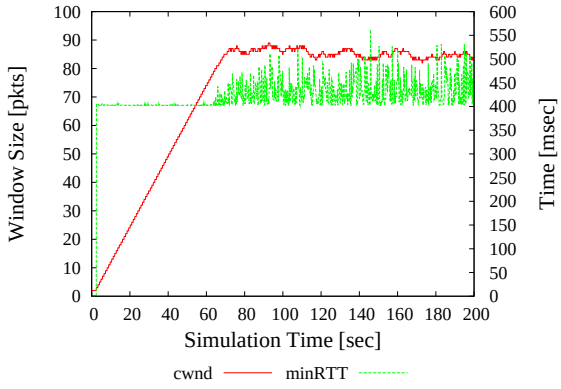
最大シーケンス番号を管理することによって，ネットワーク上にデータパケットを送出するかを決定する．TCP は現在の輻輳ウィンドウサイズのデータパケットをネットワーク上に送出すると，ACK パケットを受信しない限り，新たにデータパケットをネットワーク上に送出しない．データパケットおよび ACK パケットが廃棄されない場合は，ACK パケットに記載されている最大シーケンス番号は連続であるため，ACK を 1 つ受信するたびにデータパケットが 1 つ送信可能となる．しかし，図 8(b) から図 8(f) のように，無線アクセスポイントにおいて ACK パケットが損失すると，最大シーケンス番号が不連続となる．このため，無線端末が ACK パケットを受信すると，送信可能となるデータパケット数が 1 ACK パケットにつき複数になり，無線区間において送出されるデータパケットの総数が急激に増加するため，無線 LAN において衝突が増加する．このため，無線区間において送出される ACK パケットの総数が減少したと考えられる．

データパケットは前述のとおり ACK パケットより衝突しやすいため，さらなる衝突を誘発すると考えられる．このため，ACK パケットが無線アクセスポイントに蓄積されてからネットワークに送出するまでの時間が増加し，さらに，無線アクセスポイントにおいて ACK パケットの廃棄が増大すると考えられる．

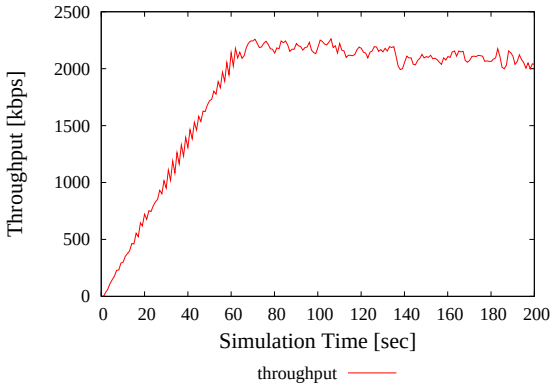
これを考慮し，図 6(e) と図 7(e)，図 6(f) と図 7(f) をそれぞれ比較すると，無線区間に送出されるデータパケットの総数と ACK パケットの総数がほぼ同数である場合より，無線区間に送出されるデータパケットの総数が ACK パケットの総数より大きい場合の方が，無線区間の遅延変動が大きくなっていることがわかる．以上から，無線アクセスポイントにおいて ACK パケットが廃棄されることによって，無線 LAN 内の輻輳がより悪化することがいえる．

図 1 のネットワークモデルにおけるシミュレーション結果について，コネクション数が 2 であり，TCP として TCP Vegas を用いた場合の結果を図 9 に，FAST TCP を用いた場合の結果を図 10 に示す．また，CTCP を用いた場合の結果を図 11 に AReno を用いた場合の結果を図 12 に，CUBIC を用いた場合の結果を図 13 に示す． 各図の (a) は輻輳ウィンドウの変化と 各 TCP 改良手法が輻輳ウィンドウの制御に用いる RTT 情報の変化を表し，(b) はスループットの変化を表す，また，(c) は無線区間における遅延変動，すなわち，TCP においてデータパケットが発生してから無線アクセスポイントにおいて受信されるまでの時間 (delay_data)，無線端末からフレームが送信されて無線アクセスポイントにおいて受信されるまでの時間 (delay_data_mac) ，および無線アクセスポイントからフレームが送信されて無線端末において受信されるまでの時間 (delay_ack_mac) を表す．

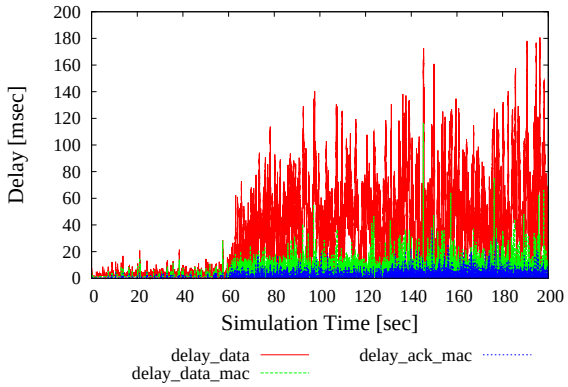
図 9 から図 12 の (c) より，純粋な delay-based 手法である TCP Vegas および FAST TCP の場合は，delay_data がそれぞれ 60 [msec] を中心に 60 [msec] の幅，100 [msec] を中心に 100 [msec] の幅で変動し，遅延が一定になるように動作していることがわかる．しかし，loss-



(a) 輻輳ウィンドウサイズの変化および RTT の変化

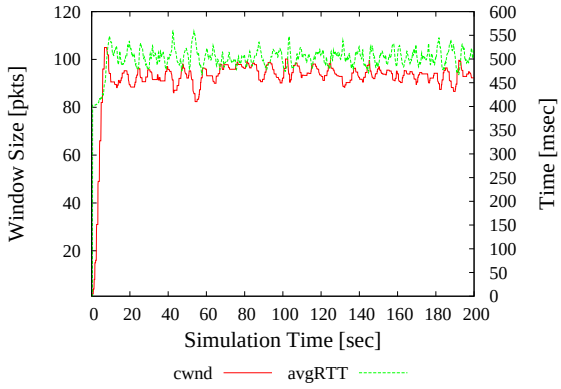


(b) スループットの変化

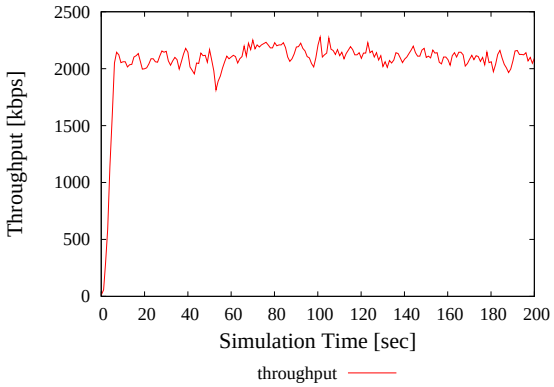


(c) 無線区間の遅延変動

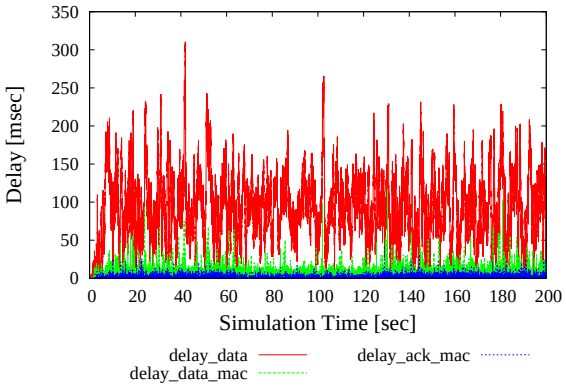
図 9: 無線区間にボトルネックがあるネットワークにおけるシミュレーション結果 (TCP Vegas, コネクション数: 2，伝搬遅延: 200 [msec])



(a) 輻輳ウィンドウサイズの変化および RTT の変化

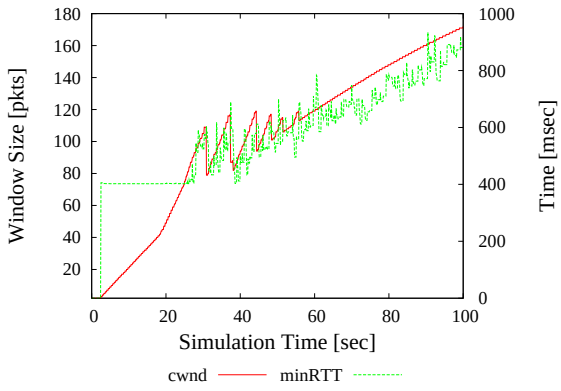


(b) スループットの変化

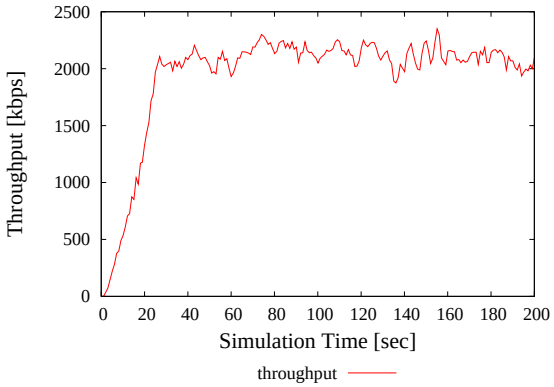


(c) 無線区間の遅延変動

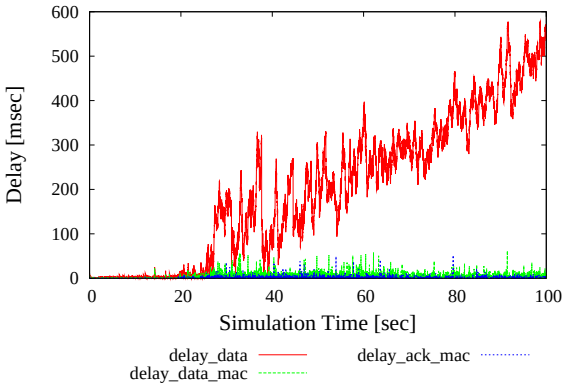
図 10: 無線区間にボトルネックがあるネットワークにおけるシミュレーション結果 (FAST TCP, コネクション数: 2，伝搬遅延: 200 [msec])



(a) 輻輳ウィンドウサイズの変化および RTT の変化

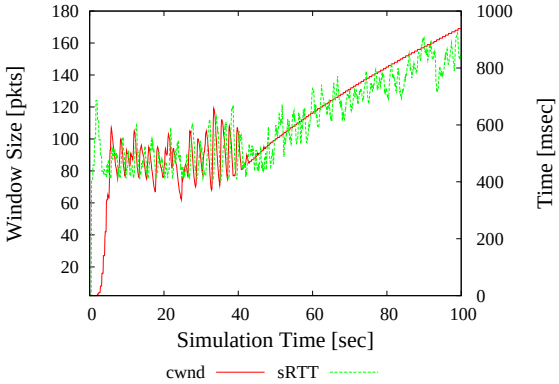


(b) スループットの変化

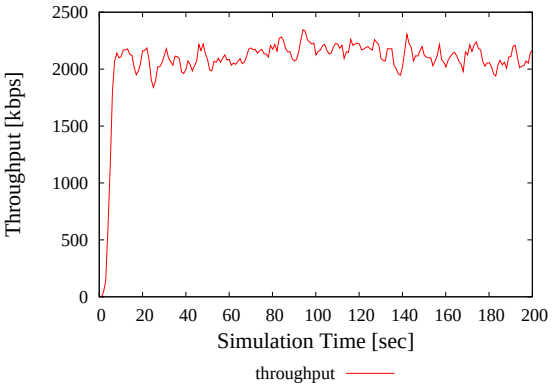


(c) 無線区間の遅延変動

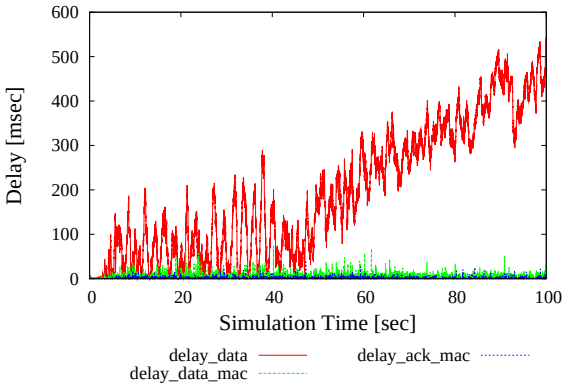
図 11: 無線区間にボトルネックがあるネットワークにおけるシミュレーション結果 (CTCP, コネクション数: 2，伝搬遅延: 200 [msec])



(a) 輻輳ウィンドウサイズの変化および RTT の変化

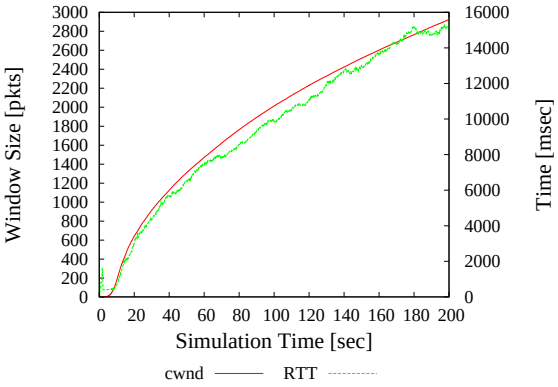


(b) スループットの変化

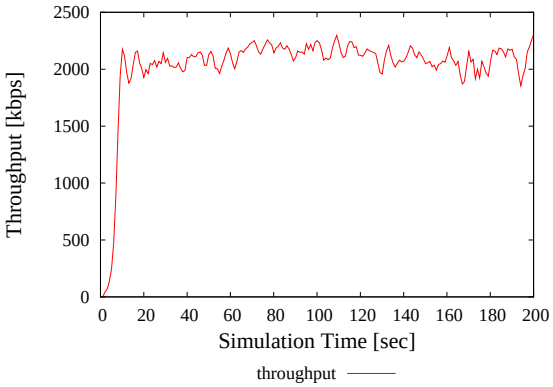


(c) 無線区間の遅延変動

図 12: 無線区間にボトルネックがあるネットワークにおけるシミュレーション結果 (AReno, コネクション数: 2，伝搬遅延: 200 [msec])



(a) 輻輳ウィンドウの変化および RTT の変化



(b) スループットの変化

図 13: 無線区間にボトルネックがあるネットワークにおけるシミュレーション結果 (CUBIC, コネクション数: 2，伝搬遅延: 200 [msec])

based 手法と delay-based 手法を組み合わせた手法である CTCP および ARenoの場合は，シミュレーション開始時から 50 秒経過した辺りから delay_data が変動しながら単調増加していることがわかる．これは，CTCP および ARenoがシミュレーション開始時から 50 秒経過するまでは，delay-based 手法による動作によって TCP Vegas や FAST TCP のように delay_data の値を一定に保つように動作しているが，50 秒移行は TCP Reno の輻輳制御によって求められる輻輳ウィンドウサイズが 各方式の delay-based 手法による動作に求められる輻輳ウィンドウサイズより大きくなり，輻輳ウィンドウの増加速度をそれぞれの手法が TCP Reno と同じにしたためだと考えられる．

また，CTCP や ARenoが TCP Reno に対抗するための loss-based 手法の動作に切り替わった後において，delay_data が増加しつづけている．これは，前述のように図 1 のネットワーク環境下においては，ACK パケットが無線アクセスポイントにおいて廃棄されるが，データパケットが廃棄されないことに起因していると考えられる．TCP はデータパケットの損失に対しては輻輳制御を行い，輻輳ウィンドウサイズを減少させるが，ACK パケットの損失に対しては 1 RTT 内のすべての ACK パケットが損失しない限り輻輳制御を行わない．このため，loss-based 手法の輻輳制御による輻輳ウィンドウサイズの増加がとまらず，データパケットが無線端末内の送信バッファに蓄積されることにより，delay_data，すなわち，TCP においてデータパケットが発生してから無線アクセスポイントにおいて受信されるまでの時間が増加しつづけていると考えられる．

このことと前述の ACK パケットが廃棄されることによって無線 LAN の輻輳状態が悪化してしまうということから，loss-based 手法や loss-based 手法と delay-based 手法を組み合わせた手法では，最終的に無線アクセスポイントにおいて ACK パケットの廃棄が発生するため，今回検討したような環境においては，純粋な delay-based 手法のように遅延の増加によって制御するべきだといえる．

TCP として TCP Vegas を用いた場合は，図 9 (a) より，TCP Vegas が式 (2) によって輻輳ウィンドウサイズを増加させると，輻輳ウィンドウサイズがシミュレーション開始から 70 秒で本シミュレーション環境下での帯域遅延積に相当するパケット数になっていることがわかる．また，図 9(c) より，60 秒以降から，delay_data_mac は 10 [msec] から最大 60 [msec] 程度の幅で変動し，delay_ack_mac は 20 [msec] から最大 120 [msec] 程度の幅で変動していることがわかる．これらの無線区間の遅延変動にともない，図 9(a) より，$minRTT$ の値が変動しているが，輻輳ウィンドウサイズの変化が 5 [pkts] 程度と変動幅が小さいことがわかる．これは，式 (2) より，TCP Vegas の輻輳ウィンドウサイズの増減幅が 1 であるために，無線区間の遅延変動により $minRTT$ の値が大きく変動したとしても，輻輳ウィンドウサイズに対して影響がほとんど現れないためだと考えられる．

TCP として FAST TCP を用いた場合は， 図 10(c) より，delay_data_mac が 10 [msec] から

最大 100 [msec] 程度変動しており，delay_ack_mac に関しては 30 [msec] から最大 200 [msec] 程度変動していることがわかる．この無線区間の遅延変動によって，図 10(a) より，FAST TCP が輻輳制御に用いる RTT 情報である $avgRTT$ が 100 [msec] 程度の幅で変動していることがわかる．さらに，$avgRTT$ の変動にともなって，輻輳ウィンドウサイズの増減が 15 [pkts] 程度の幅で変動していることがわかる．これは，FAST TCP が式 (3) のように，現在の輻輳ウィンドウサイズと $avgRTT$ の値に応じて，輻輳ウィンドウサイズの増減幅を決定するため，TCP Vegas より輻輳ウィンドウが大きく変動していると考えられる．

TCP として CTCP を用いた場合は，図 11(a) より，CTCP は 20 秒付近から delay-based 手法の動作によって輻輳ウィンドウサイズが大きくしており，50 秒付近から TCP Reno と同じ増加速度で輻輳ウィンドウサイズが大きくなっていることがわかる．また，図 11(c) より，delay_data_mac が 10 [msec] から 50 [msec] 程度変動しており，delay_ack_mac は 20 [msec] から 50 [msec] 程度の幅で変動していることがわかる．図 11(a) と図 11(c) から，CTCP が delay-based 手法の動作によって輻輳制御を行っている間，無線区間の遅延変動による影響はほとんど現れず，無線端末の送信バッファにパケットが蓄積されることによる遅延の変動に応じて，$minRTT$ が変動し，それにしたがって輻輳ウィンドウが変動していることがわかる．

TCP として AReno を用いた場合は，図 12(a) より，AReno は 40 秒付近まで delay-based 手法の動作によって輻輳ウィンドウサイズを増減させており，40 秒以降は TCP Reno と同じ増加速度で輻輳ウィンドウサイズを増加させていることがわかる．また，図 12(c) より，delay_data_mac が 10 [msec] から 50 [msec] 程度変動しており，delay_ack_mac は 20 [msec] から 20 [msec] 程度変動していることがわかる．図 12(a) と図 12(c) から，送信バッファにパケットが蓄積されることによって発生する遅延の変動と無線区間における遅延変動によって，AReno は輻輳ウィンドウサイズを 90 [pkts] 程度を中心に 30 [pkts] 程度変動させていることがわかる．

TCP として CUBIC を用いた場合は，図 12(a) から，CUBIC は，シミュレーション時間内において常に輻輳ウィンドウサイズを増加していることがわかる．これは，前述のとおり，図 1 のネットワーク環境下において，ACK パケットは廃棄されるがデータパケットが廃棄されないために，loss-based 手法である CUBIC はネットワークの輻輳を検出できないためだと考えられる．

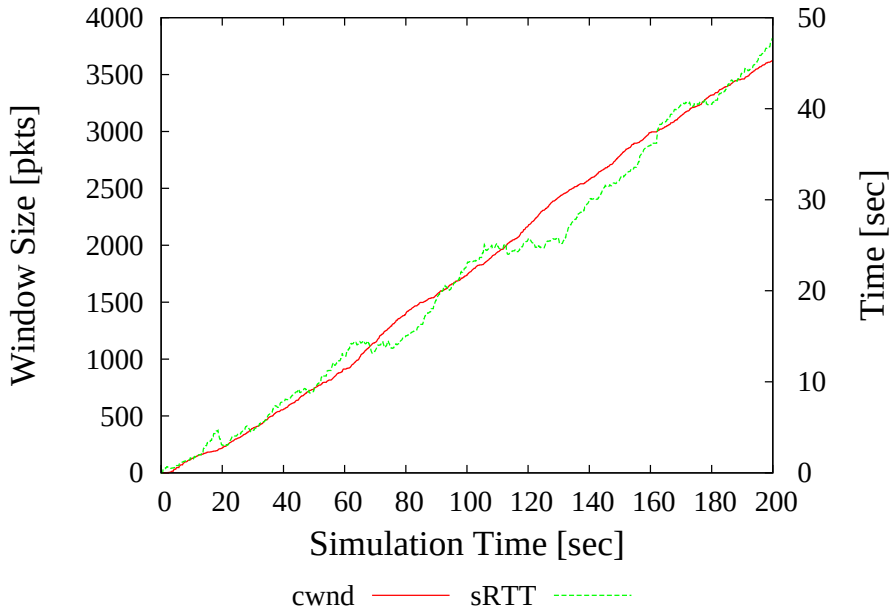
2.5 節で述べたとおり，CUBIC は輻輳ウィンドウサイズの制御に RTT を用いないため無線区間の変動による影響は受けないと考えられる．このため，CUBIC のスループットの変化は図 1 によって表されるネットワーク環境下では無線区間の遅延変動がない場合の結果であると考えることができる．図 13(b) より，CUBIC のスループットは 200 [kbps] 程度変動していることから，無線区間の遅延の影響とは無関係に 200 [kbps] 程度スループットが変動す

ることがわかる．図 9(b) から図 12(b) より，どの TCP 改良手法に置いてもスループットの変動幅が 200 [kbps] 以内であることがわかる．これは，FAST TCP や AReno の場合は輻輳ウィンドウを大きく変化させるが，その変動が無線端末の送信バッファ内に蓄積されたデータパケット数の変動にしか影響を与えないため，スループットまでは影響が出なかったと考えられる．

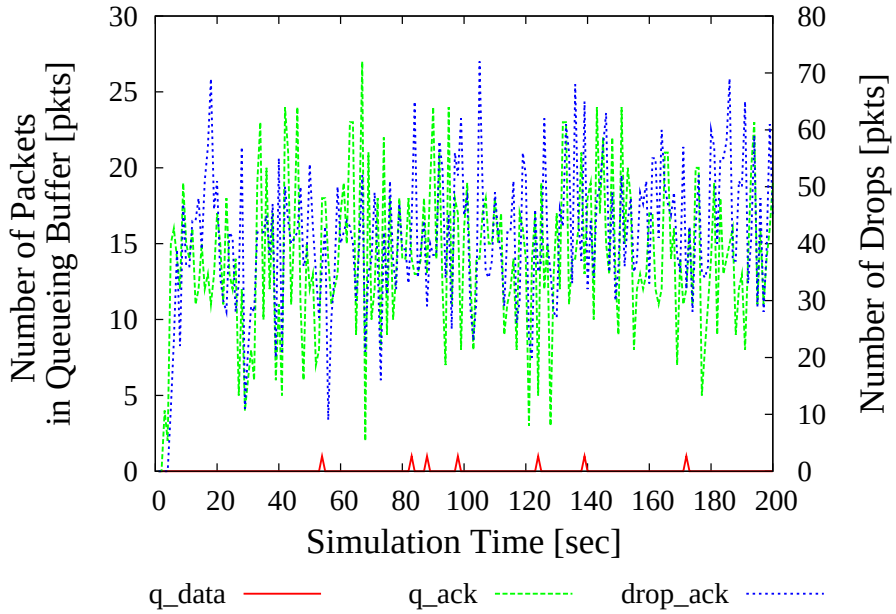
以上のことから，無線端末が TCP の送信側で無線区間がボトルネックとなる場合においては，無線区間の遅延変動は delay-based 手法の動作に影響を与えないということができる．

図 1 のネットワーク環境において，コネクション数が多くなった場合，輻輳ウィンドウが大きくなるコネクションが存在する反面，輻輳ウィンドウサイズがほとんど増加しないコネクションが出現するという現象が現れる．TCP として AReno を用いた場合の正常なコネクションのシミュレーション結果を図 14 に，問題のあるコネクションのシミュレーション結果を図 16 に，また，TCP として CUBIC を用いた場合の正常なコネクションのシミュレーション結果を図 15 に，問題のあるコネクションのシミュレーション結果を図 17 に示す．図 14(a) および図 15(a) より，それらの TCP コネクションでは輻輳ウィンドウサイズが十分大きくなっているのに対し，図 16(a) および図 17(a) のような TCP コネクションの場合は輻輳ウィンドウサイズが 1 [pkts] からほとんど増加していないことがわかる．図 16(b) および図 17(b) から，輻輳ウィンドウが 1 [pkts] からほとんど増加しないコネクションでは，無線アクセスポイントにほとんど ACK パケットが蓄積せず，ACK パケットが廃棄されていることがわかる．それらに対し，正常に輻輳ウィンドウサイズを増加させているコネクションは図 14(b) および図 15(b) のように，無線アクセスポイントにおいて ACK パケットが廃棄されているが，無線アクセスポイントのバッファ内にも ACK パケットが保持されていることがわかる．

これらのことから，このような TCP コネクション間で深刻な不公平性が発生した原因は，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下では，前述のとおり，コネクション数が増加すると，ACK パケットが無線アクセスポイントに蓄積し，ACK パケットが廃棄される．また，TCP はデータパケットが 1 つでも廃棄され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 パケットが廃棄される場合においては，TCP は 1 ウィンドウ中のすべての ACK パケットが廃棄されなければ輻輳制御が動作せず，輻輳ウィンドウサイズを減少させない．このため，図 1 のようなネットワーク環境のような ACK パケットが廃棄される環境では，TCP はネットワークの輻輳を 1 ウィンドウ中のすべての ACK が廃棄されることによって発生するタイムアウトによってのみ認識し，輻輳ウィンドウサイズを 1 に減少させる．しかし，このとき，無線アクセスポイントのバッファは一杯であり，ACK パケットが無線端末までほとんど到達しない．以上により，輻輳ウィンドウサイズの回復が行われず，深刻な不公平性を生じたものだと考えられる．

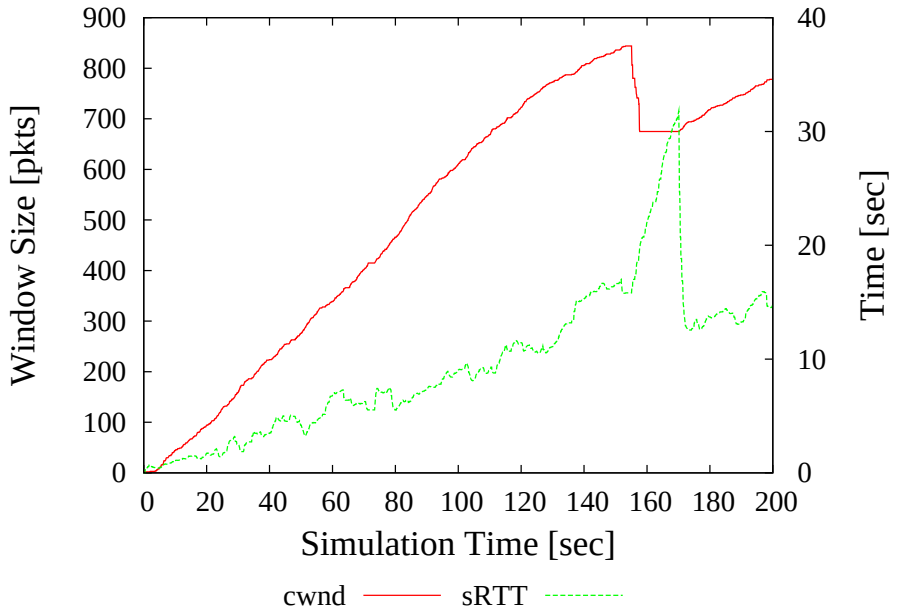


(a) 輻輳ウィンドウサイズの変化

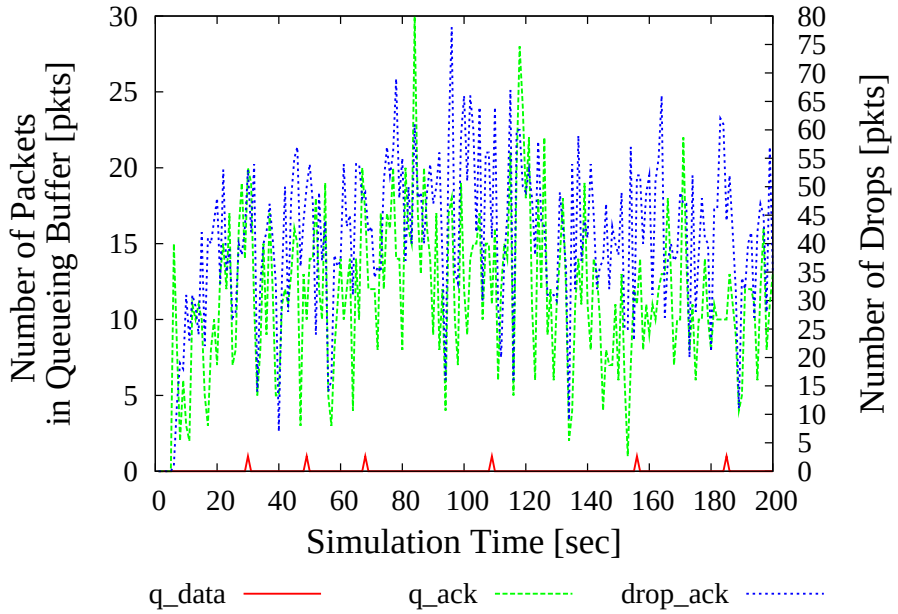


(b) 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄率の変化

図 14: 不公平が発生する場合の正常なコネクションのシミュレーション結果 (AReno，コネクション数: 16，伝搬遅延: 200 [msec])

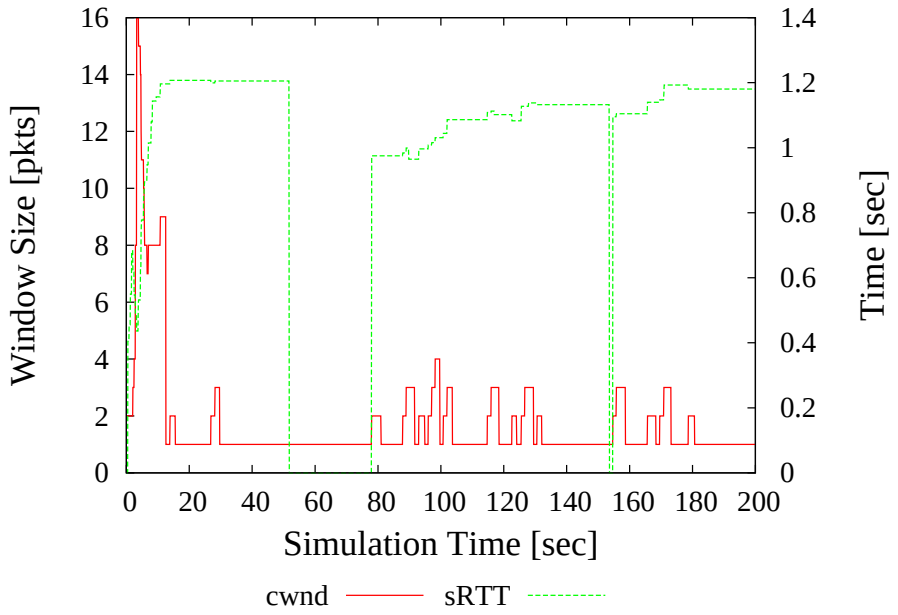


(a) 輻輳ウィンドウサイズの変化

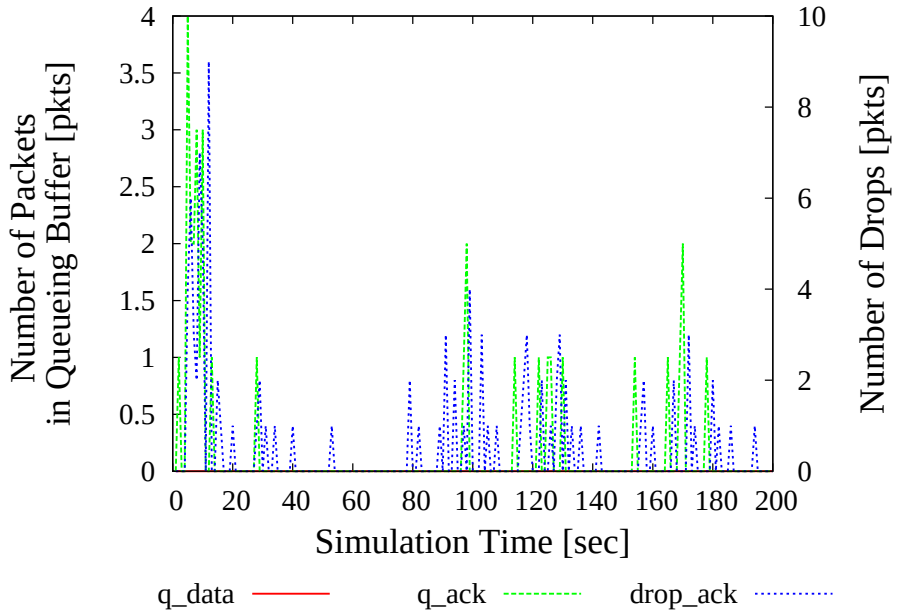


(b) 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄率の変化

図 15: 不公平が発生する場合の正常なコネクションのシミュレーション結果 (CUBIC，コネクション数: 16，伝搬遅延: 200 [msec])

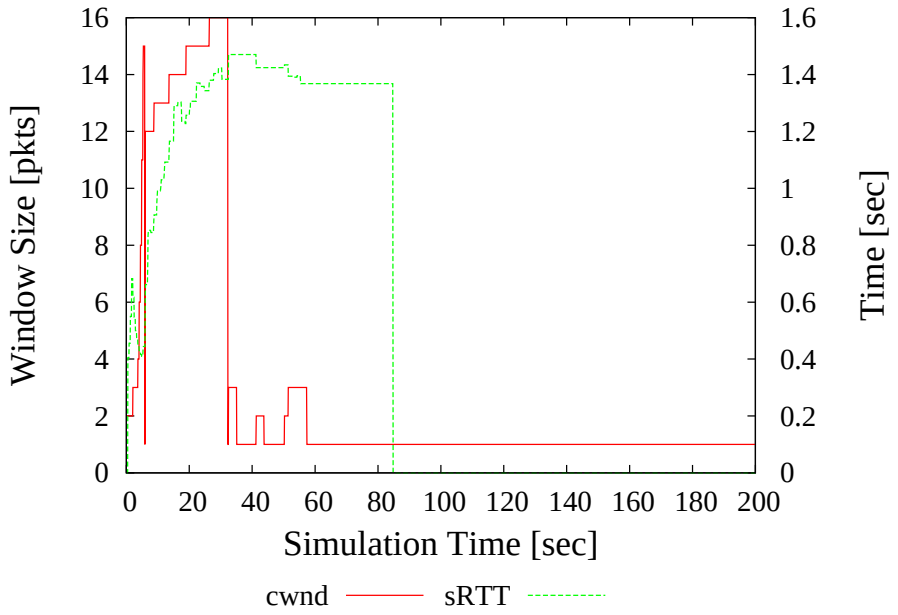


(a) 輻輳ウィンドウサイズの変化

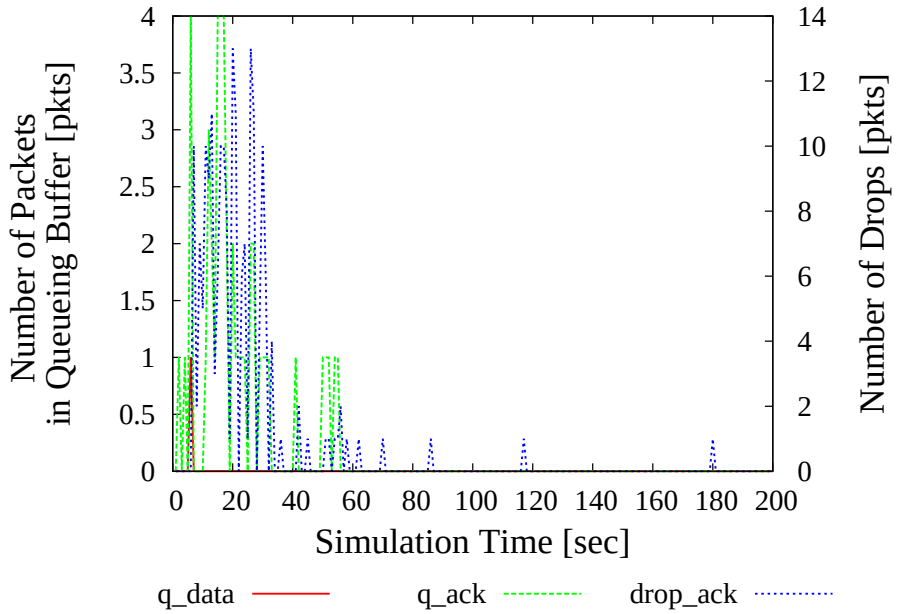


(b) 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄率の変化

図 16: 不公平が発生する場合の問題のあるコネクションのシミュレーション結果 (AReno，コネクション数: 16，伝搬遅延: 200 [msec])



(a) 輻輳ウィンドウサイズの変化



(b) 無線アクセスポイントにおけるキュー長の変化およびパケット廃棄率の変化

図 17: 不公平が発生する場合の問題のあるコネクションのシミュレーション結果 (CUBIC, コネクション数: 16，伝搬遅延: 200 [msec])

また，この現象は，無線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 ACK パケットが蓄積されているために生じているため，既にこのような輻輳が発生している無線 LAN 環境に新たに無線端末が参加しようとした場合でも，同様な現象が発生すると考えられる．

以上の結果から，無線端末が TCP の送信側であり無線区間がボトルネックとなるネットワーク環境においては，delay-based 手法に対して無線区間の遅延変動による影響が抑えられるため問題はなく，loss-based 手法や最終的に loss-based 手法として動作する loss-based 手法と delay-based 手法を組み合わせた手法では，ACK パケットが無線アクセスポイントに蓄積されることに起因して，無線 LAN の輻輳を悪化させ，TCP コネクション間に不公平性を生じる．そのため，輻輳制御を遅延のみに基づいて行う方が良いといえる．

3.2 ボトルネックが有線区間にある場合

本節では，ボトルネックが有線区間にあるネットワーク環境においてシミュレーションを行い，無線 LAN 環境がボトルネックとなる場合の各 TCP 改良手法への影響について考察する．さらに，ボトルネックが無線区間にある場合のネットワーク環境でのシミュレーション結果を踏まえ，無線 LAN 環境において無線端末が送信側である場合の各 TCP 改良手法への影響について考察する．

3.2.1 シミュレーション環境

無線 LAN 規格として IEEE 802.11a を使用し，ボトルネックリンクが有線区間にある場合のネットワークモデルを図 18 に示す．図 18 は，図 1 と同様に，複数の無線端末が送信側 TCP として 1 つの無線アクセスポイントを共有し，無線アクセスポイントから受信端末までが有線リンクにより接続されている構成である．無線端末から無線アクセスポイントまでは IEEE 802.11a を利用しているため無線伝送速度は最大 54 [Mbps] の無線リンクである．また，無線アクセスポイントから受信端末まではリンク帯域が 22 [Mbps] であり，片道の伝搬遅延が 50 [msec] である．このため，このネットワークモデルでは，無線アクセスポイントから受信端末までの有線区間がボトルネックリンクとなる．なお，無線アクセスポイントトから受信端末までの有線リンクのバッファには，キュー管理機構として DropTail を用いた．

シミュレーションは 3.1.1 節で記述した内容と同様に，無線端末数を 1 台から 32 台まで変化させ，各無線端末から有線端末に対して TCP コネクションを 1 本生成する．TCP には CTCP および AReno を用いて，それぞれの場合について各無線端末の TCP に同一の TCP 改良手法を設定した．また，有線ボトルネックリンクのバッファサイズは，TCP に CTCP を用いた場合は 40 [pkts] に，TCP に AReno を用いた場合は 20 [pkts] に設定した．これは，有線ボトルネックリンクのバッファサイズを小さくすることにより，無線区間の遅延変動の影響を相対的に大きくするためである．各 TCP コネクションはシミュレーション開始後，0.01 秒間隔で，データ転送を開始した．各無線端末は生成した TCP コネクションを用いて，アプリケーション層プロトコルとして FTP を用いて無限大のサイズを持つデータの転送を行う．データパケットのサイズは 1500 [pkts] に設定した．

また，AReno に関しては，図 18 のネットワークモデルにおいて，有線ボトルネックのリンク帯域を 100 [Mbps] に，有線ボトルネックリンクのバッファサイズを 500 [pkts] にそれぞれ設定し，前述のシミュレーションを行った．

各 TCP 改良手法のパラメータは，それぞれ以下のように設定した．

CTCP $\alpha_c = 0.125$, $\gamma_c = 30$, $\zeta_c = 1$

AReno $\alpha_a = 16, \beta_a = 1, \gamma_a = 10$

3.2.2 シミュレーション結果と考察

図 18 のネットワーク環境において，有線ボトルネックのリンク帯域を 100 [Mbps]，有線ボトルネックのバッファサイズを 500 [pkts] に設定し，TCP として AReno を用いた場合のシミュレーションにおける，無線区間の遅延変動を図 19 に，無線区間に送出されるパケットの総数の変化を図 20 に示す．なお，図 19 において，図 6 と同様に，コネクション数が 2 以上の場合においては，無線端末が複数あるが，より無線区間の遅延変動が大きい端末の無線区間の遅延変動を示している．

3.1.2 節で述べたとおり，無線区間にボトルネックがある場合はコネクション数が増加するにつれ，無線区間の遅延変動が大きくなるという結果であったが，有線区間にボトルネックがある場合では，図 19 より，コネクション数が増加しても遅延変動がほとんど増加しないことがわかる．また，図 20 をみると，コネクション数が増加しても，無線区間に送出されるデータパケットの総数と ACK パケットの総数がほぼ同数であることがわかる．このことから，有線区間にボトルネックがある場合における，無線区間の遅延変動が無線区間にボトルネックリンクがある場合と比べて小さいのは以下の理由によるものだと考えられる．図 7 および図 20 より，無線区間の送出されるパケット総数が，無線区間がボトルネックである場合は約 4400 [pkts] であるのに対して，有線区間にボトルネックがある場合は約 3500 [pkts] と小さい．これは，有線区間にボトルネックがある場合は無線区間のネットワークには余裕があるということを意味する．したがって，フレームが衝突する確率が，無線区間にボトルネックがある場合に比べて低いと考えられる．このため，無線区間の遅延変動が小さくなると考えられる．さらに，無線区間がボトルネックである場合のように，ACK パケットが廃棄されることによる遅延変動の増大も発生しない．

また，コネクション数が増加すると無線区間に送出されるパケットの総数が変動しているのがわかる．これは，データパケットが有線ボトルネックリンクにおいて廃棄されることによって，TCP が輻輳制御を行った結果，輻輳ウィンドウサイズが減少したためだと考えられる．

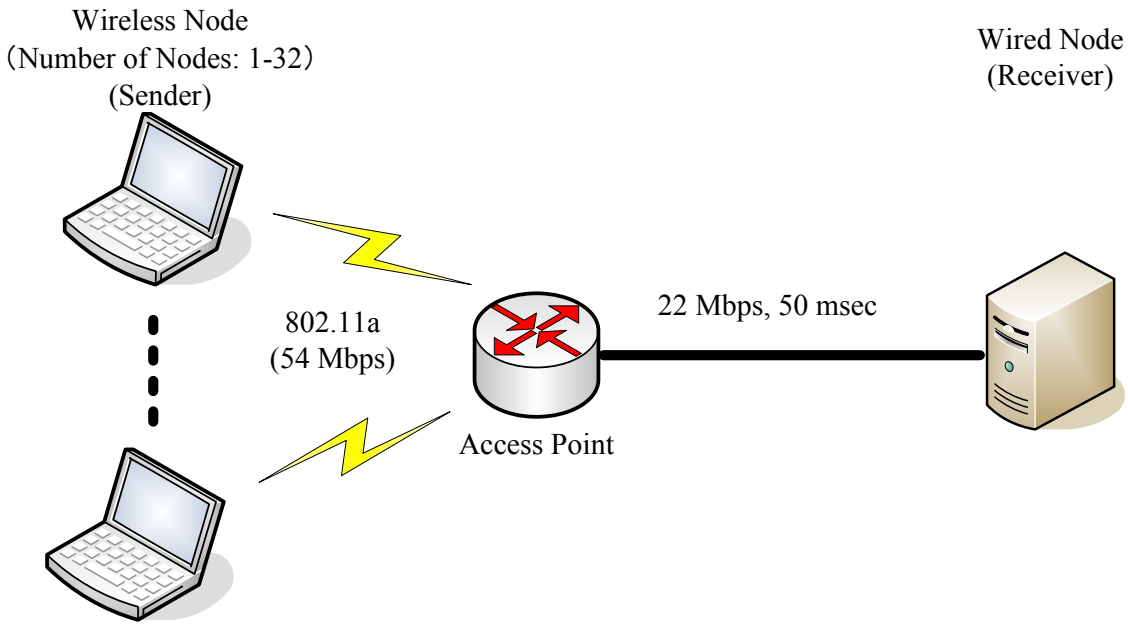
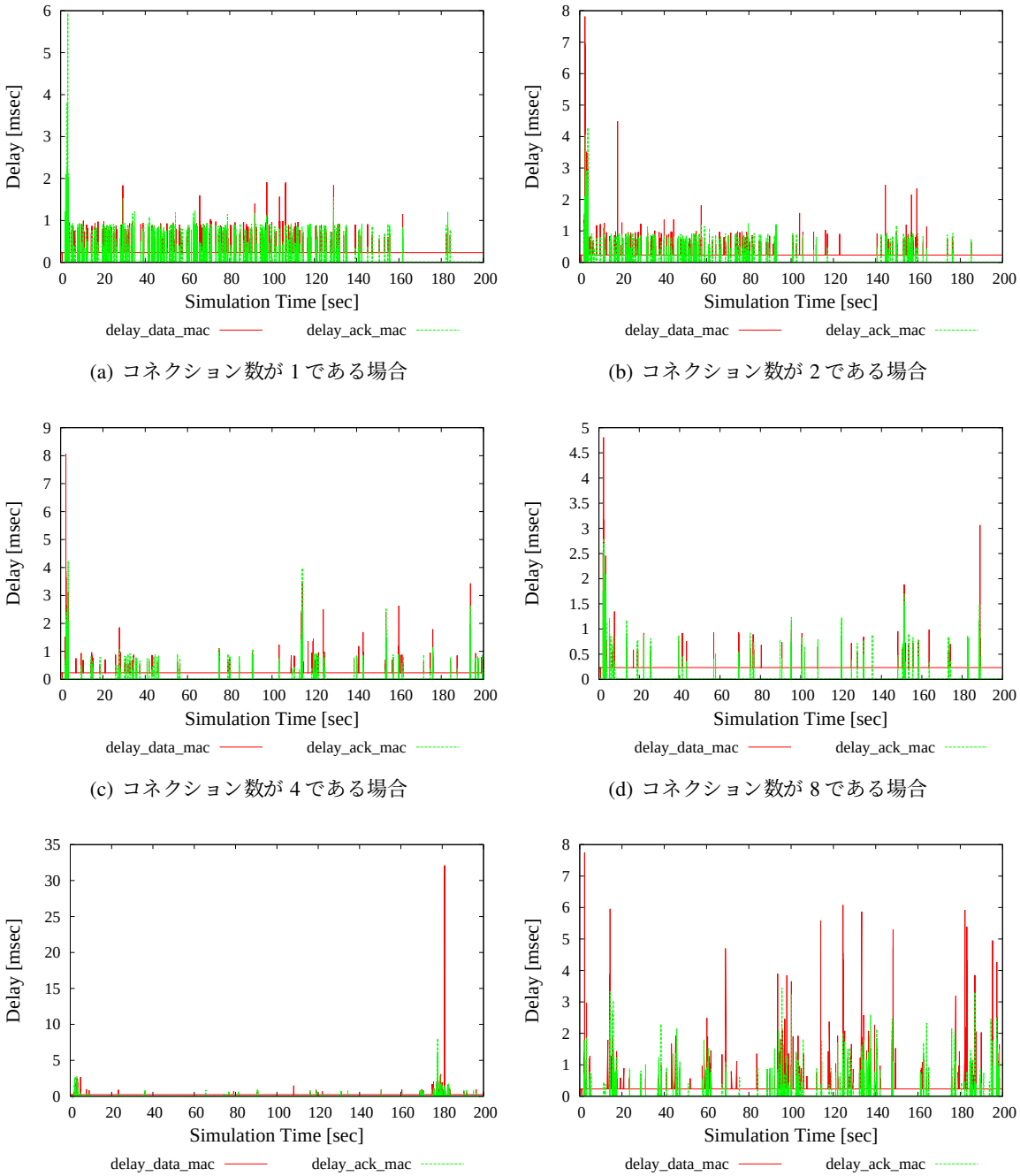


図 18: IEEE 802.11a を用いた場合のボトルネックが有線区間にあるネットワークモデル



(a) コネクション数が 1 である場合

(b) コネクション数が 2 である場合

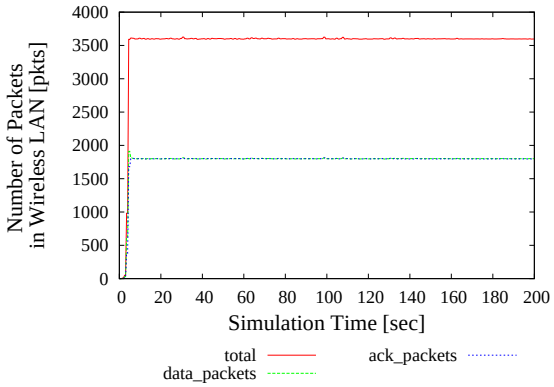
(c) コネクション数が 4 である場合

(d) コネクション数が 8 である場合

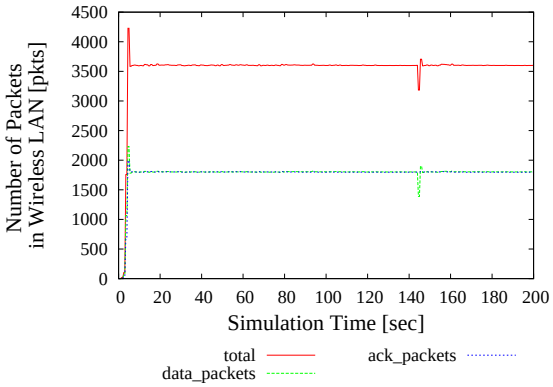
(e) コネクション数が 16 である場合

(f) コネクション数が 32 である場合

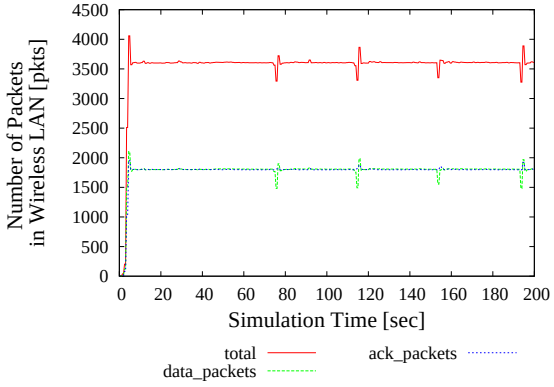
図 19: 有線区間にボトルネックがある場合の無線区間の遅延変動 (AReno，伝搬遅延: 100 [msec])



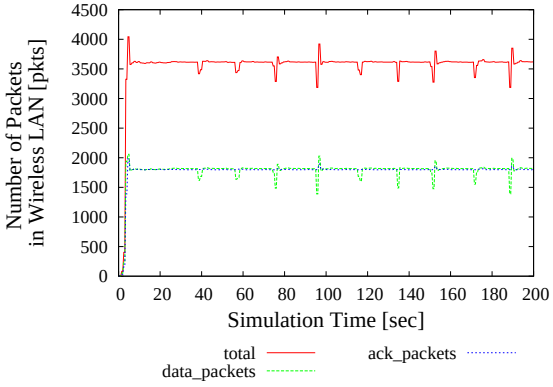
(a) コネクション数が 1 である場合



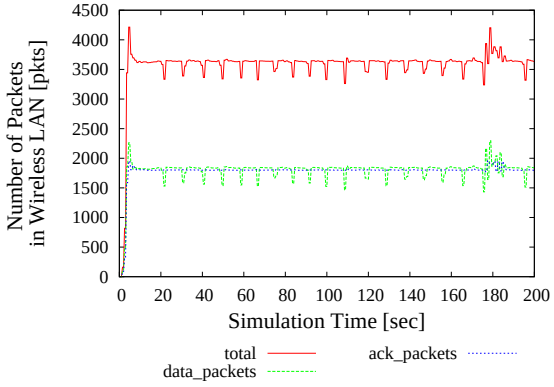
(b) コネクション数が 2 である場合



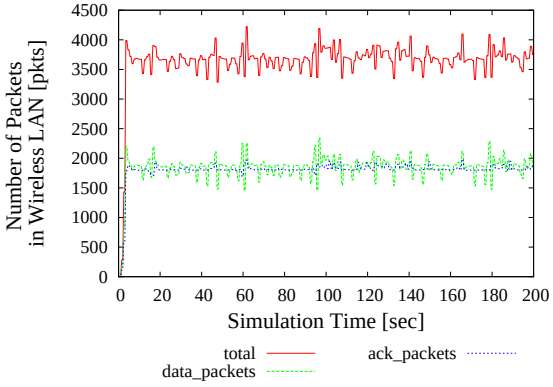
(c) コネクション数が 4 である場合



(d) コネクション数が 8 である場合



(e) コネクション数が 16 である場合



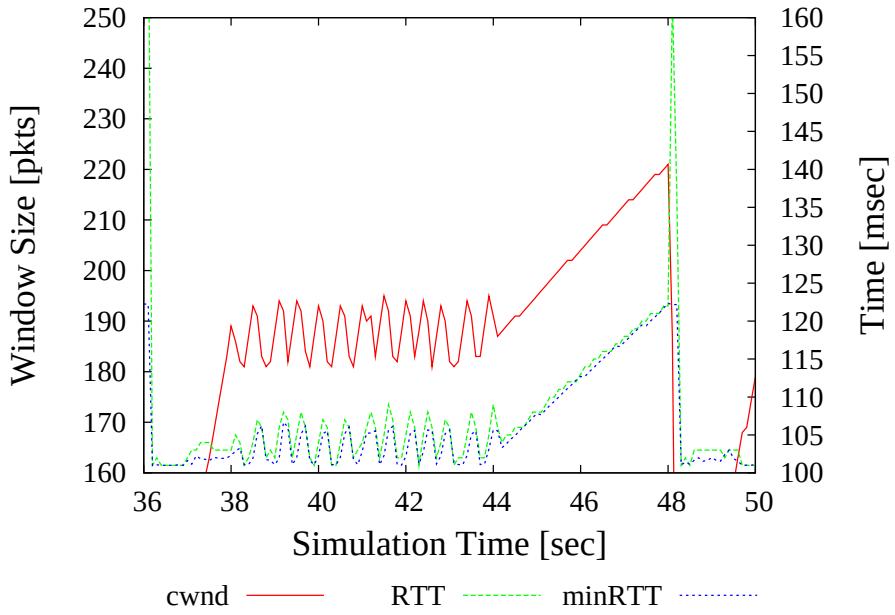
(f) コネクション数が 32 である場合

図 20: 有線区間にボトルネックがある場合の無線区間に送出されるパケットの総数の変化 (AReno，伝送遅延: 100 [msec])

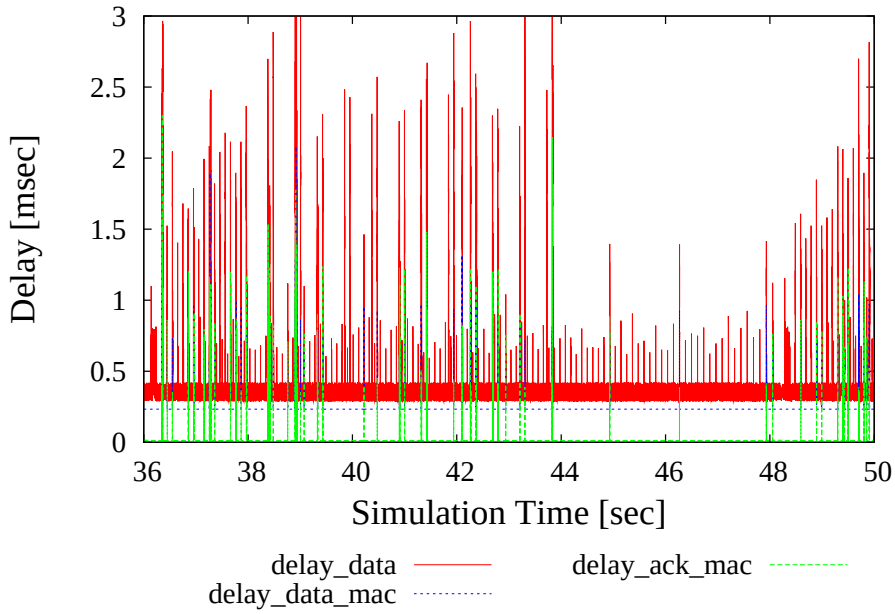
次に，図 18 のネットワーク環境において，CTCP のシミュレーション結果を図 21 に，AReno のシミュレーション結果を図 21 に示す．図 21 および図 22 をみると，前述のとおり有線区間にボトルネックがある場合は無線区間における遅延変動が小さいことがわかる．また，2.3 節で述べたように，CTCP は式 (5) の RTT 情報として，1 RTT 内に観測した最小の RTT である $minRTT$ を用いる．このため，1 RTT 以上の間，無線区間の遅延変動による RTT の増加が継続されなければ，$minRTT$ に影響を与えない．このため，図 21(a) のように RTT には無線区間の遅延変動による影響が現れているが，$minRTT$ には影響はなく，さらには輻輳ウィンドウサイズの増減に対しても影響がないといえる．

2.4 節で述べたとおり，AReno は式 (8) に用いる RTT 情報として，TCP が管理している平滑下 RTT である $sRTT$ を用いているが，図 22(b) のように，無線区間の遅延変動が 最大で 2.5 [msec] と $sRTT$ の値より相対的に小さく，それをさらに平滑化するため，図 22(a) において，輻輳ウィンドウサイズの増減に対してほとんど影響がないといえる．また，FAST TCP の場合は，AReno より重みが大きな式 (4) を用いているため，AReno のよりさらに影響が出にくいと考えられる．

以上の結果と，3.1 節の結果から，無線端末が TCP の送信側であるときには，無線区間にボトルネックがある場合は，無線端末の送信バッファに蓄積されるパケットにより無線区間における遅延変動による影響が抑えられ，有線区間にボトルネックがある場合は，無線区間における遅延変動の幅が無線区間がボトルネックである場合より小さく，各 TCP 改良手法で用いられている RTT 情報のフィルタリング処理により遅延変動の抑えられるため，TCP スループットに与える影響は小さい，ということが明らかとなった．

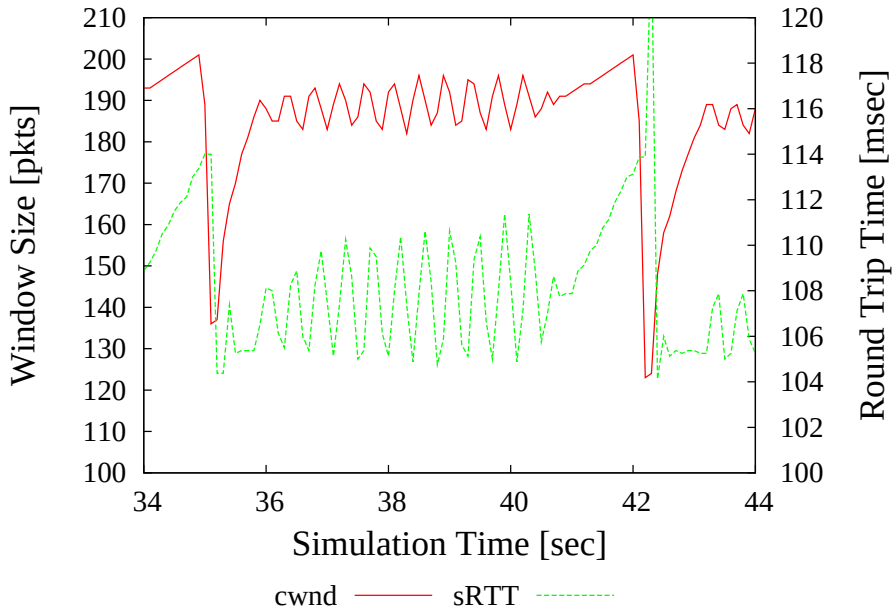


(a) 輻輳ウィンドウサイズの変化

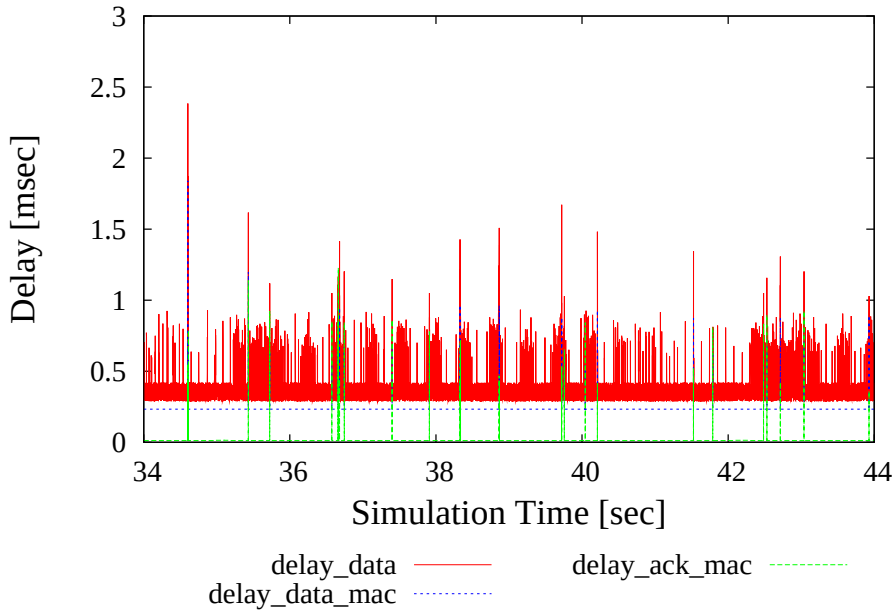


(b) 無線区間における遅延変動

図 21: IEEE 802.11a を用いた場合におけるシミュレーション結果 (CTCP，コネクション数: 1，伝搬遅延: 50 [msec])



(a) 輻輳ウィンドウサイズの変化



(b) 無線区間における遅延変動

図 22: IEEE 802.11a を用いた場合におけるシミュレーション結果 (AReno，コネクション数: 1，伝搬遅延: 50 [msec])

4　まとめと今後の課題

本報告では，ns-2 を用いたシミュレーションを行うことにより，無線 LAN 環境がさまざまな TCP 改良手法へ与える影響を検証するとともに，RTT を輻輳の指標として用いる TCP 改良手法の，無線 LAN 環境における有効性を評価した．

シミュレーション結果から，無線区間がボトルネックとなる場合においては，無線区間の遅延変動が大きくなるものの，TCP スループットに対してほとんど影響を与えないことがわかった．また，有線区間がボトルネックとなる場合においては，無線区間の遅延変動が小さくなるため，各 TCP 改良手法で用いられている RTT 情報のフィルタリング処理により無線区間の遅延変動の影響が抑えられ，輻輳ウィンドウおよびスループットにほとんど影響を与えないことがわかった．以上から，本報告において対象にしたネットワーク環境においては，RTT を輻輳の指標として用いる TCP 改良手法は，無線 LAN 環境下でも正常に動作するということがいえる．

一方で，無線区間がボトルネックとなる場合においては，無線端末が増え，TCP コネクションが増加すると，無線アクセスポイントにおいて ACK パケットがバッファ溢れを引き起こし，TCP コネクション間に深刻な不公平が発生することが明らかとなった．

今後の課題として，無線端末が受信側となる場合および上りと下りのコネクションが混在する場合における性能評価や，本報告において指摘した，各 TCP コネクション間に深刻な不公平が発生する問題に対しての対策の提案が挙げられる．

謝辞

本報告を終えるにあたり，御指導，御教授を頂きました中野博隆教授，村田正幸教授に深く感謝致します．また，本報告を作成するにあたり，日頃から熱心かつ的確な指導および助言をして頂きました長谷川剛准教授に心より感謝致します．最後に，日頃の研究生活を送るにあたり，相談に答えて頂きました平岡祐一朗氏，児玉瑞穂氏および堀江拓郎氏をはじめとする中野研究室の皆様方，また，津川知朗氏をはじめとする村田研究室の皆様方に心より御礼申し上げます．

参考文献

[1] W. R. Stevens and G. R. Wright, *TCP/IP Illustrated, Volume 2 : The Implementation*. Addison-Wesley, 1995.

[2] S. Floyd, "HighSpeed TCP for large congestion windows," *Request for Comments 3649 (Experimental)*, Dec. 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] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks," *ACM SIGCOMM Computer Communication Review*, vol. 33, pp. 83–91, Apr. 2003.

[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, Jan. 2007.

[9] 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.

[10] 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.

[11] 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.

[12] 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.

[13] V. Tsaoussidis and I. Matta, "Open issues on TCP for mobile computing," *Wireless Communications and Mobile Computing*, vol. 2, pp. 3–20, Feb. 2002.

[14] C. Casetti, M. Gerla, S. Mascolo, M. Y. Sanadidi, and R. Wang, "TCP Westwood: Bandwidth estimation for enhanced transport over wireless links," in *Proceedings of ACM Mobicom 2001*, pp. 287–297, July 2001.

[15] K. Xu, Y. Tian, and N. Ansari, "TCP-Jersey for wireless IP communications," *IEEE JSAC*, vol. 22, pp. 747–756, May 2004.

[16] "The network simulator ns-2." available at `http://www.isi.edu/nsnam/ns/`.

[17] "FAST TCP simulator module for ns-2." available at `http://www.cubinlab.ee.mu.oz.au/ns2fasttcp/`.

[18] 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.

[19] "Compound TCP in Linux." available at `http://netlab.caltech.edu/lachlan/ctcp/`.

[20] D. X. Wei and P. Cao, "A Linux TCP implementation for NS2." available at `http://www.cs.caltech.edu/~weixl/technical/ns2linux/index.html`, May 2006.