情報処理学会論文誌 Vol.58 No.2 427–438 (Feb. 2017)

IEEE 802.11n無線LANのアップロードTCP通信における Bufferbloat問題に対するアクセスポイントでの対応方式

野元 祐孝[1,a)] 策力木格[1,b)] 大坐畠 智[1,c)] 加藤 聰彦[1,d)]

受付日 2016年5月20日, 採録日 2016年11月1日

概要：IEEE 802.11n 無線アクセスポイントを経由する TCP 通信では，低い MAC レベルデータレートが使用される場合に，往復遅延が増加する Bufferbloat 問題が発生し，同じキューを共通する他の通信の遅延時間が増加し，ユーザレベルの品質の悪化となる．この対処として，アクティブキュー管理方式，筆者らが先に示した無線 LAN の最大再送回数を制限する方法，遅延増加時に TCP 通信を停止する方法などが提案されている．しかしこれらはすべて，データの送信側に実装する必要がある．このため，端末からアクセスポイントへのアップロード転送においては，端末のすべてにこのような方式を導入しなければならない．そこで本稿では，データの受信側であるアクセスポイントにおいて，低い MAC レベルデータレートで無線 LAN フレームの最大再送回数を制限したと同様な効果を与えることにより，TCP の輻輳ウィンドウを減少させ，Bufferbloat 問題に対応する方法を提案する．パソコンを用いたアクセスポイントに提案方式を実装することにより，Bufferbloat 問題に対応していない端末で，低い MAC レベルデータレートでの往復遅延を減少させられること，さらに Bufferbloat 問題に対する方法を実装している端末においても TCP 通信のスループットを悪化させることがないことを明らかにした．

キーワード：無線 LAN，IEEE802.11n，TCP，動的レート切替え，Bufferbloat 問題

Resolving Bufferbloat Problem at Access Point for Upload TCP Communication Over IEEE802.11n Wireless LAN

Masataka Nomoto[1,a)] Celimuge Wu[1,b)] Satoshi Ohzahata[1,c)] Toshihiko Kato[1,d)]

Received: May 20, 2016, Accepted: November 1, 2016

Abstract: IEEE 802.11n wireless LANs provides a Bufferbloat problem such that the round-tirp delay increases in TCP communications when the MAC level data is low such as 6.5 Mbps. This problem degrades user level quality of service for communications sharing the transmission queue with the TCP communications. In order to resolve this problem, several schemes are proposed such as a scheme based on active queue management, our previous scheme reducing the retry-out limit at low data rate, and a scheme stopping sending TCP level data when the queue contains more than a specific number of packes. However, those schemes need to be implemented at the sender side. In this paper, we propose a new scheme resolving a Bufferbloat problem in the upload TCP communications by providing an effect at a receiving side access point similar to reducing the retry-out limit when the MAC level data rate is low. This decreases the congestion window size in the sending side and can reduce the delay. This paper also shows the results of the performance evaluation by implementing the proposed method in a PC based access point. According to the results, the proposed method decreases the round-trip time in the low MAC data rate and does not reduce the TCP throughput. It does not give any influences on the TCP thoughput for the terminals supporting the conventional schemes for Bufferbroat problems.

Keywords: wireless LAN, IEEE802.11n, TCP, dynamic rate switching, bufferbloat problem

1. はじめに

現在広く普及しているIEEE 802.11n [1] 準拠の無線LANでは，従来の規格に対して高速なデータ通信を行うための手順が導入されている．複数のアンテナやチャネル帯域を利用するMIMOおよびチャネルボンディングにより，MACレベルのデータレート（MACデータレート）として300 Mbpsといった高い値が実現可能となる．また複数フレームを1つに集約するフレームアグリゲーションや複数のデータフレームの受信確認をまとめて行うブロックACKにより，MACレベルのオーバヘッドを減少させ，信頼性の高い誤り回復が可能となっている．

その一方で従来の無線LANと同様に，複数のデータレートをサポートし，端末とアクセスポイント（AP）間でそれぞれに最適なMACデータレートを選択する動的レート切替えを行う．このため，端末がAPの近隣に位置する場合には300 Mbpsといった高いMACデータレートを利用できるが，端末がAPから離れた場合は6.5 Mbpsなどの大幅に低下したMACデータレートを使用することになる．

低いMACデータレートの使用時にTCPによるバルクデータ転送を行うと，送信側に数百ミリ秒から数秒のキューイング遅延が発生し，送信キューを共有する他の通信がその影響を受け，ユーザレベルの通信品質が劣化する [2], [3]．これはBufferbloat問題と呼ばれ，広く議論が行われている [4], [5]．

Bufferbloat問題を解決するためにいくつかの方法が提案されている [2], [3], [6], [7]．いずれもバルクデータ転送を行うTCPトラヒックのレートを減らすことを目的としており，2つのグループに分けられる．

第1が，キューイング遅延が発生するような状況において，あえてパケットロスを発生させ，TCPの輻輳ウィンドウを減少させるというものである．CoDel [6] は，アクティブキュー管理（Active Queue Management：AQM）に分類される方法で，送信キューに一定時間以上パケットがとどまる状況が発生すると，キューの中の1パケットを廃棄する．また筆者らが先に提案した方法 [2], [3] では，低いMACデータレートが用いられた場合に，TCPセグメントを含むMACフレームの最大再送回数を減らし，MACレベルでのフレーム廃棄を発生しやすくするというものである．

第2が，TCPのデータ送信時にMACレベルの送信キューに多くのパケットが蓄積された場合は，TCPでデータ送信を停止するというものである．Linuxオペレーティングシステムのバージョン3.6以降に組み込まれているTCP small queue [7] がその例である．

無線LANにおけるBufferbloat問題は，APから端末へのダウンロード通信でも，端末からAPへのアップロード通信でも発生する．ダウンロード通信におけるBuferbloat問題はAPにおけるMAC送信キューで発生する．APがエンドノードではないため第2の方法は使用できず，CoDelのようなAQM方式か，筆者らの方式 [2], [3] をAPに実装する必要がある．いずれかの方式をAPに実装することにより，すべての端末に対するBufferbloat問題を解決できる．

一方，アップロード通信におけるBufferbloat問題は，ある端末においてバルクデータ転送を行うアプリケーションがあった場合，その端末内の別のアプリケーション，たとえばWebブラウジングやリモートログインが，遅延の影響を受けるというものである．この問題を防ぐには，個々の端末に上記のいずれかの方法を実装する必要がある．Linuxオペレーティングシステムでは CoDel または TCP small queues が実装されているため，すでに対応済みであるといえる．しかし他のオペレーティングシステムを用いるパソコン，スマートフォン，タブレットなど，端末によってはこれらをサポートしない場合もあり，対応が必要である．実際4.2節で述べるように，Windows 10やMac OSにおいてはBufferbloat問題への対応が行われていないと考えられる．

本稿では，802.11n無線LANでのアップロードTCP通信におけるBufferbloat問題を，APのみで対応する方法を提案する．具体的には，筆者らが先に提案した送信側で最大再送回数を減らす方式と同等の効果を，フレームを受信するAPにおいて実現するという方式を用いる．すなわち，低いMACデータレートに対しては送信側よりも低い再送回数の上限を設定し，フレームを受信するAP上で再送回数を推定し，回数が上限を超えた場合はそのフレームをロスしたと見なす．提案方法をAPに実装するだけで，アップロード通信におけるBufferbloat問題に対処することが可能となる．本稿ではパソコンによるAP上に提案方式を実装し，その性能評価結果を示す．結果として，提案方式がBufferbloat問題に対応していない端末に対して遅延を減少させるとともに，CoDelまたはTCP small queuesを実装した端末に対してもTCPによるバルクデータ転送のスループットをほとんど低下させないことを示す．

本稿の構成は次のとおりである．2章で802.11n無線LANにおけるフレームアグリゲーションとブロックACKの手順と関連研究について述べる．3章では提案する方式の詳細を示す．4章では性能評価の結果を示し，5章において結論を述べる．

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

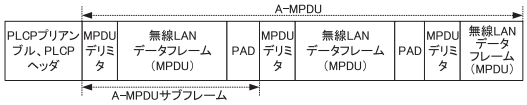


図 1 A-MPDU のフォーマット
Fig. 1 Format of A-MPDU.

2. 802.11n の手順と関連研究

2.1 802.11n におけるフレームアグリゲーションとブロック ACK の手順

802.11n では，データフレームの送信の効率化を図るために，データフレーム（MAC Protocol Data Unit：MPDU と呼ぶ）を複数個まとめて送信することを可能としている（**図 1** 参照）．フレーム全体は A-MPDU（Aggregated MPDU）と呼ばれ，MPDU デリミタ，MPDU 本体，パディングを含む A-MPDU サブフレームの集合となる．MPDU デリミタには，MPDU の長さやデリミタ自身の誤りを検出する CRC などが含まれる．またパディングは 0 から 3 バイトの長さで，A-MPDU サブフレームを 4 バイトの倍数になるように調整する．

また，802.11n では HT-immediate Block Ack という受信確認方式を採用している．A-MPDU を受信すると，各 MPDU のシーケンス番号に対応する受信・未受信を示す Block Ack Bitmap を持つ Block Ack フレームを返信する．Bitmap は 64 個分の MSDU（MAC Service Data Unit：MAC サブレイヤの上位から与えられるデータ）の受信・未受信を指定できるようになっている．送信側はその情報をもとに，MPDU を再送する．なお，A-MPDU 全体が紛失した場合は Block Ack 自身が返送されないため，A-MPDU 全体をタイムアウト再送する．

これらの送受信処理は，データの送信側と受信側の無線 LAN チップと無線 LAN インタフェースのデバイスドライバで実現される．それぞれの処理は**図 2** のようになると考えられる．データ送信側では，デバイスドライバにおいて，アグリゲーションを行う MPDU を選択する．その中には，Block Ack フレームの Bitmap から判断した再送分も含まれる．また一定回数の再送が行われた場合は，MPDU ごとにリトライアウトを判断する．これに対して，無線 LAN チップでは，実際に A-MPDU を作成し送出する処理，A-MPDU 全体のタイムアウト再送の処理，BlockAck フレームの Bitmap のデバイスドライバへの通知などの処理を行う．

一方データ受信側では，無線 LAN チップにおいて，受信した MPDU のデバイスドライバへの通知，BlockAck フレームの送信などを行う．なお MPDU の通知機能では，CRC エラーとなった MPDU に関する情報も知らされる．それに対して，受信側のデバイスドライバでは，MPDU の順序管理やリトライアウトの判断などを行う．受信側でのリトライアウトは，再送回数を計測するのではなく，一定時間受信できないとリトライアウトされたと判断するという処理に基づく．

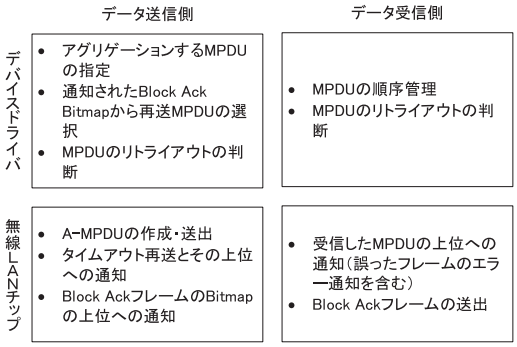


図 2 A-MPDU の送受信処理の分担
Fig. 2 Send/Receive processing of A-MPDUs.

2.2 関連研究

(1) CoDel

CoDel [6] は，送信キューでのパケットの滞在時間を監視する AQM の一手法である．パケットがデキューされるたびに，キューに滞在した時間（キューイング時間）を測定し，別途定めたインターバル時間（初期値のデフォルトは 100 ミリ秒）ごとにキューイング時間の最小値を算出する．インターバル時間内の最後のパケットがデキューされた時点で，その間のキューイング時間の最小値が一定値（デフォルト値は 5 ミリ秒）以上であった場合，次の 1 パケットを破棄し，インターバル時間を短い値（破棄したパケット数の平方根に反比例）に設定する．もし最小キューイング時間が 5 ミリ秒より小さい場合，インターバル時間は初期値に戻る．

(2) 無線 LAN の最大再送回数を制御する方式

筆者らは，802.11n 無線 LAN における TCP 通信の Bufferbloat 問題の原因の 1 つが，HT-immediate Block Ack という受信確認方式と，それによる MPDU の選択的再送が強力なことであると考えている．そこで，低い MAC データレートで送信される TCP セグメントを含む MPDU に対しては，最大再送回数を小さくすることにより，パケット損失の発生頻度をあえて増加させるという方法を提案した [2], [3]．これにより，TCP レベルでの輻輳ウィンドウを減少させる機会を増加させる．先の論文 [2], [3] では，TCP 通信のスループットを低下させずに，RTT（Round-trip Time）の増加を抑制できることを報告した．

(3) TCP small queues

通常，TCP 送信ソケットバッファに格納されたアプリケーションのデータは，輻輳ウィンドウまたは受信側が広告したウィンドウで規定されるサイズまで TCP セグメントとして下位層に出力される．これに対して TCP small queues [7] は，送信端末内の Linux のスケジューラ（qdisc）

や無線 LAN などのデバイス上でのキューイング状況を考慮し，TCP から送出されるセグメントの合計バイト数を制限する方式である．具体的には，qdisc およびデバイスのキューに，指定量（デフォルトでは 128 KB）以上のパケットが蓄積された場合，TCP は新たに生成されたデータセグメントを下位層に渡さずに送信ソケットバッファ上でキューイングを続ける．下位のキューが空いた場合にはじめて，対応するセグメントを下位層に渡す．その結果，たとえば無線 LAN のキューに多くのデータが蓄積された場合は，送信ソケットバッファに未送信データが蓄積され，アプリケーションから TCP へのデータ転送が停止する．これにより，TCP セグメントを破棄することなしに送信端末内の Bufferbloat 問題を抑えることができる．

3. 提案方式

3.1 基本方針

本稿は，802.11n 無線 LAN を対象として，端末から AP へのアップロード TCP 通信において端末内の送信キューで発生するバッファリング遅延を解消する方法を提案する．具体的には，MAC データレートが低い場合は，端末から送信する TCP セグメントを含む MPDU に対して，最大再送回数を減少させたと同様な効果を，受信側である AP において実現することを目指す．これにより，パケット損失の頻度が増加し，TCP での輻輳ウィンドウの減少につながる．この方式を実現するにあたり，以下のような基本方針を採用することとした．

前述のように，無線 LAN では，その手順を実現するために，無線 LAN チップとそのデバイスドライバとに機能分担させる．そこで第 1 の方針として，無線 LAN チップの機能は変更せずに，デバイスドライバの変更のみで対応することとした．これは提案方式を現実的な方法で実装するためである．

MPDU の再送と Block Ack フレームによる受信確認は無線 LAN チップにより実装される．したがって，送信側が決められた最大再送回数まで再送を行う動作を，受信側の AP で変更することができない．そこで，MAC データレートが低い場合には，受信側のデバイスドライバにおいて，小さい再送回数であたかもリトライアウトが生じたように振る舞い，その次のデータの受信処理を進めるというアプローチをとる．この回数をリトライアウトインデックスと呼ぶ．

第 2 に，無線 LAN チップからデバイスドライバに通知される情報を最大限活用するという方針を用いることとした．前述のように，受信側のデバイスドライバには，MPDU デリミタまたは MPDU において CRC エラーとなった MPDU のシーケンス番号などの情報も通知される．CRC エラーである場合，このような情報は誤っている可能性があるが，提案方式の実現においては，これらの情報も矛盾のない限りは利用することとする．

表 1 リトライアウトインデックス
Table 1 Retry out index.

平滑化データレート [Mbps]	～25	25～50	50～100	100～
リトライアウトインデックス	2	5	8	適用外

3.2 詳細設計

以下に，提案方式の詳細な実装方法を示す．

(1) リトライアウトインデックスを，MAC データレートに応じて**表 1** に示すように定める．データレートが 100 Mbps 未満でもともとの最大再送回数よりも低い値をリトライアウトインデックスとして使用する．この値は，AP が管理する端末ごとに管理する．また，値を定めるために用いるデータレートは，個別の A-MPDU を受信した際のデータレートを指数移動平均により平滑化したものを使用する．その際の平滑化係数は 0.25 とする．

(2) 受信側の AP において，MPDU ごとに再送回数を推定する．MPDU が TCP セグメントを含み，かつ推定再送回数がその時点のリトライアウトインデックスを超えた場合は，その MPDU は紛失したとして処理し，次の MPDU の受信に移行する．次の MPDU を受信している場合は上位に通知する．ただし，送信側は本来の最大再送回数まで再送を行うため，紛失の処理を行った MPDU を受信する場合もあることに注意する必要がある．

(3) 再送回数の推定は，CRC エラーの MPDU の通知により行う．すなわち，CRC エラーの MPDU が到着したとき，そのシーケンス番号が正しいとして，対応するシーケンス番号の MPDU の再送回数を加算する．なお，この場合の推定された再送回数は，Block Ack フレームによる選択的再送と，A-MPDU 全体のタイムアウト再送の両方を含むことになる．

(4) エラーを含まない正しい MPDU が到着した場合，紛失の処理を行っていないものに対しては規定の動作を行う．すなわち，順序どおりに到着した場合は上位に通知し，期待されるシーケンス番号より大きい場合はバッファリングする．また，紛失の処理をした MPDU を受信した場合は無視する．

3.3 動作例

図 3 に提案方式による AP における処理の例を示す．図に示された表の第 1 行は受信した A-MPDU を，第 2 行は各 A-MPDU に含まれる MPDU のシーケンス番号をそれぞれ示す．また MPDU の番号の上の斜線は CRC エラーであったことを意味する．たとえば，A-MPDU(1) は 5 つの

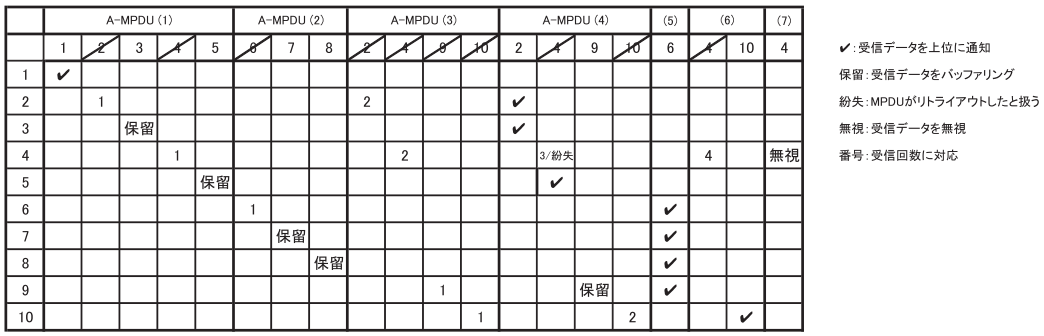


図 3 提案方式による AP の処理の例
Fig. 3 Example of AP side processing in proposed scheme.

MPDU を含み，そのうちシーケンス番号 2 と 4 の MPDU が CRC エラーであったことを示している．

表のセルは，第 2 行で示された MPDU を受信した場合の各 MPDU （表の第 1 列の番号に対応）の処理の様子を示している．「✓」は受信したデータを上位に通知することを意味し，「保留」と「無視」は受信したデータをバッファリングまたは無視することをそれぞれ示す．またセル中の数字はそのシーケンス番号の MPDU の受信回数を示す．これは推定再送回数＋ 1 を意味する．なお，この例ではリトライアウトインデックスを 2 と仮定する．

上述のように，最初の A-MPDU(1) は 5 つの MPDU を含み，そのうち 2 と 4 が誤っている．このため，1 のデータは上位に通知し，3 と 5 はバッファリングする．2 と 4 に対しては受信回数を 1 とする．次に，A-MPDU(2) に対して，同様に，6 の受信回数を 1 とし，7 と 8 をバッファリングする．

次に，A-MPDU(3) は A-MPDU 全体が誤った場合である．この場合も誤った MPDU のシーケンス番号はデバイスドライバに通知されることは実験的に確認済みである．そこで，2，4，9，10 の MPDU に対して受信回数を設定する．次に A-MPDU(4) で前の A-MPDU がタイムアウト再送される．ここで 2 の MPDU が正しく受信されるため，2 とバッファリングされていた 3 を上位に通知する．また 4 と 10 の受信回数を増加させ，9 を保留とする．ここで，MPDU4 に対しては，受信回数が 3 となり，リトライアウトインデックス分だけ再送されたことになる．このため MPDU4 は紛失した処理を行う．このとき MPDU5 のデータがバッファリングされているため，そのデータが上位に通知される．続いて，6 の MPDU が単独で受信され，6 およびそれに続く 7 から 9 の MPDU のデータを上位に通知する．さらに，A-MPDU(6) において，MPDU4 が誤って，MPDU10 が正しく受信される．MPDU4 は紛失処理が行われているため単に受信回数を増加させる．また MPDU10 は上位に通知する．最後に，A-MPDU(7) において MPDU4 が正しく受信されるが，これに対しては，無線 LAN チップで Block ACK を返答するものの，デバイスドライバではそのデータを無視する．

このようにして，送信側の最大再送回数よりも小さいリトライアウトインデックスを受信側で設定し，早めに MPDU の紛失処理を行う．これにより，本来の 802.11n 手順にはない MPDU の紛失が発生し，TCP において輻輳制御が起動され，無線 LAN への送信負荷の軽減につながることが期待される．

4. 評価実験

提案方式の有効性を示すために，Bufferbloat 問題が発生する環境下において TCP 通信と Ping（ICMP Echo Request/Reply）通信の実験を行い，TCP 通信のスループットと Ping 通信の RTT を評価した．具体的には，TCP 通信と送信キューを共有する Ping 通信の RTT が抑制されて Bufferbloat 問題が解決しているかどうか，また提案方式によるパケット損失によって TCP 通信のスループット性能が悪化していないかどうかについて確認を行う．あわせて，すでに Bufferbloat 問題への対策を導入している端末の場合，提案方式を AP に導入した場合に性能悪化がないかどうかも確認する．

4.1 実験条件

図 4 の環境で性能評価実験を行った．建物は木造家屋 2 階建てであり，サーバと AP は 2 階に設置した．1 台の端末は様々な地点に設置し，その場所で端末から AP に接続されたサーバに対して TCP 通信と Ping 通信を並行して行う．

実験に用いた機器の仕様を**表 2** に示す．端末としては，Linux，Windows 端末，Mac OS 端末の 3 種類を利用した．また AP には Atheros 社製チップを搭載した無線 LAN PC カードを装備した Lenovo 社のパソコンを用いた．AP には hostapd ソフトウェアをインストールしている．

なお，Linux 端末に対しては，Bufferbloat 問題の対策を施していない端末，TCP small queues を実装した端末，CoDel を適用した端末の 3 種類のバージョンを用意した．TCP small queues を適用した端末は Linux 3.13 をそのま

表 2 AP と端末の仕様
Table 2 Specification of AP and terminal.

	AP	Linux 端末	Windows 端末	Mac OS 端末
機種	Lenovo ThinkPad X61		Apple MacBook Air (Mid 2012)	
Kernel	Linux 3.2.38	Linux 3.13.0	Windows 10	Mac OS 10.11
無線 LAN MAC	IEEE 802.11n（5 GHz）			
TCP 輻輳制御	-	Cubic TCP [9]	OS のデフォルト設定のバージョン	
その他の TCP の設定	-	ECN，RED などの機能は使用せず， 送受信ソケットバッファは OS のデフォルト設定を利用		
AP ソフトウェア	hostapd 0.7.3	-		
NIC	NEC Aterm WL300NE-AG		Mac 内蔵（Broadcom）	
Driver	ath9k [11]		OS 標準ドライバ	

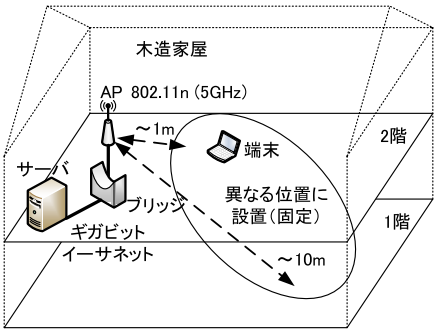


図 4 実験環境
Fig. 4 Experimental environment.

ま利用する．Bufferbloat 問題の対策を施していない端末は Linux 3.13 の TCP small queues の制限を大きな値（10,000 パケット）にしたものを用いる．CoDel を適用した端末は，Linux 3.13 で TCP small queues の制限を大きくしたうえで CoDel を実装したものを利用する．なお，使用した Linux では，後述の図 12 (a) に示すように，輻輳ウィンドウは 1,000 パケット付近に上限が設けられている．このため，TCP small queues の制限を 10,000 パケットに設定することにより，その機能を使用しないようにできる．また Linux 端末の場合は，TCP 通信における輻輳ウィンドウの値や，送信キューに接続されたパケット数（送信キュー長）など，詳細な情報が入手可能である．

表 3 に評価のための通信実験の詳細を示す．この表に示すように，端末の種類および位置に対して，それぞれ 60 秒間の通信を行っている．

4.2 全体的な実験結果

本節では全体的な実験結果，すなわち端末の種類と位置を変えた 60 秒間の通信実験における各性能の平均値について示す．ここで，端末の位置の指標として，その場所での 60 秒間の通信において転送されたすべての A-MPDU の MAC データレートの平均値を使用することとする．

まず，予備的な検討結果として，**図 5** に提案方式を AP に導入していない場合の，端末とアクセスポイントの間の MAC レベルの平均誤り率を示す．この値は，Linux 端末

表 3 1 回の通信実験の詳細
Table 3 Specification of one experimental run.

通信時間	TCP トラヒック	Ping トラヒック
60 秒	iperf [10] を 60 秒間使用．転送データ量は TCP スループットに依存．たとえば，6 Mbps，60 Mbps のスループットの場合は，それぞれ 45 M バイト，450 M バイト．	1 秒間に 1 回 ICMP エコー要求を送信．データは 64 バイト．1 回の通信実験で 60 回の Ping を実行．

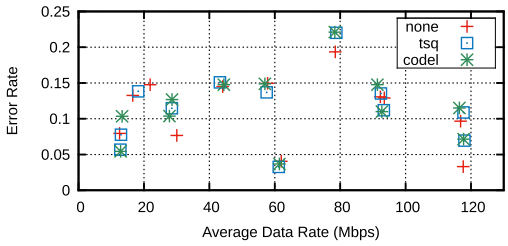
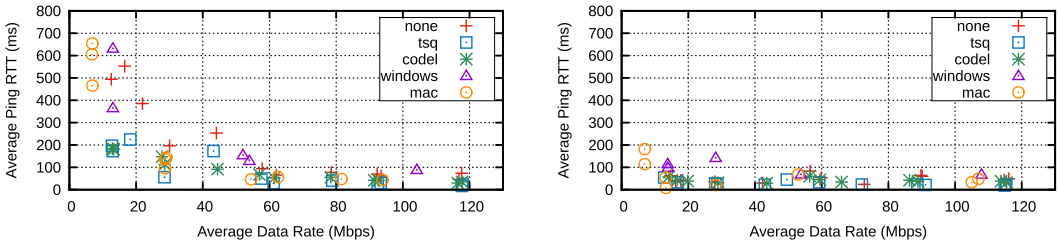


図 5 平均データレートと平均誤り率の対応
Fig. 5 Average error rate vs. average data rate.

から TCP 通信を行った場合の，転送された全 MPDU 数（MAC レベルの再送も含む）と AP により受信確認された MPDU 数から算出したものである．図 5 は，Linux 端末で 3 種類のバージョンを用いた場合の，測定における平均 MAC データレートと，平均誤り率の対応を示している．図で，none，tsq，codel はそれぞれ Bufferbloat 問題の対策を適用していない端末，TCP small queues を適用した端末，CoDel を適用した端末の結果を示す．

この結果から，MAC レベルの平均誤り率は 0.05 から 0.2 程度の範囲で分散していること，同じような位置（同程度の平均 MAC データレート）では Linux 端末のバージョンにはあまり依存しないことなどが分かる．

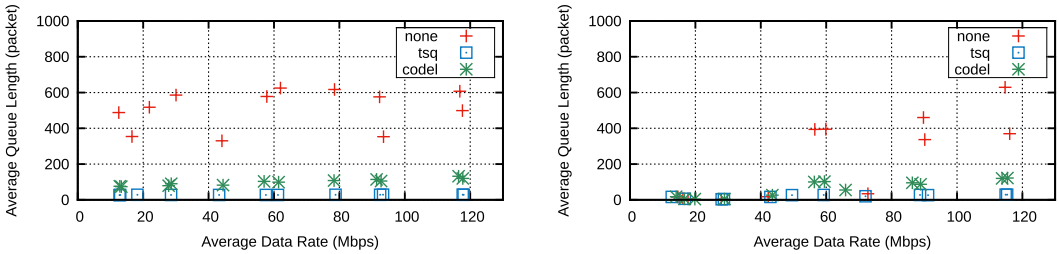
次に，**図 6**，**図 7**，**図 8** に具体的な測定結果を示す．各図の横軸は図 5 と同様に 60 秒間の測定において端末から送信された A-MPDU の MAC データレートの平均値である．図 6 は平均 MAC データレートと Ping 通信の平均



(a) 提案方式なし　(b) 提案方式あり

図 6　平均データレートと Ping の平均 RTT の対応

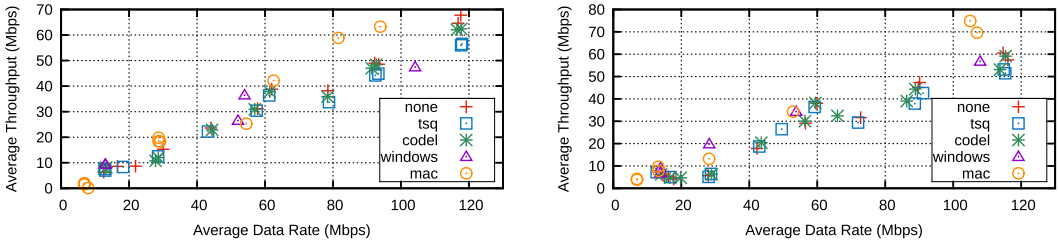
Fig. 6　Average RTT in Ping vs. average data rate.



(a) 提案方式なし　(b) 提案方式あり

図 7　平均データレートと平均送信キュー長の対応

Fig. 7　Average send queue length vs. average data rate.



(a) 提案方式なし　(b) 提案方式あり

図 8　平均データレートと平均 TCP スループットの対応

Fig. 8　Average TCP throughput vs. average data rate.

RTT の対応，図 7 は端末の送信キューの平均長との対応，図 8 は TCP 通信の平均スループットとの対応をそれぞれ示す．各図で (a) が提案方式を用いていない場合，(b) が提案方式を AP に実装した場合である．図の 1 つ 1 つのプロットは，1 回の測定値に対応する．また，none，tsq，codel は図 5 と同様であり，windows と mac はそれぞれ Windows 端末と Mac OS 端末の結果を示す．なお，図 7 の送信キューの平均長については Linux 端末のみに対して測定している．

図 6 (a) では，提案方式を用いていない場合において，Bufferbloat 問題に対応していない Linux 端末の Ping 平均 RTT が，平均 MAC データレートが 40 Mbps 程度より小さい範囲で 200 ミリ秒を超えている．特に 20 Mbps 以下では 500 ミリ秒より大きくなっている．図には示していないが，TCP 通信の平均 RTT（データセグメントと対応する ACK セグメントの時間間隔の平均値）も測定しており，Ping 通信の平均 RTT の約 2 倍となっていることを確認している．このため，Echo Request パケットの送信間隔によっては平均 RTT が悪化することも考えられる．この結果から，端末から AP 方向への通信において，Bufferbloat 問題が発生していると判断できる．また，Windows 端末や Mac OS 端末においても，平均 MAC データレートが 20 Mbps 以下で，Ping 平均 RTT が 350 ミリ秒から 700 ミリ秒程度に増大している．このため，Windows 10 や Mac OS 10.11 のデフォルト設定の TCP においても Bufferbloat 問題が発生すると考えられる．これに対して，同図の tsq および codel は，平均 RTT を下げており，Bufferbloat 問題を抑制していることが分かる．図 6 (b) では，提案方式により，Linux 端末の 3 種類と，Windows および Mac OS の各端末において，50 Mbps 以下の範囲で平均 RTT が抑えられている．特に Bufferbloat 問題に対応していない場合の Linux 端末および Windows と Mac OS の各端末おいて改善が顕著である．これによって，提案方式を AP に実装することにより Bufferbloat 問題に対して有効に対応可能であるといえる．

図 7 (a) の平均送信キュー長の結果より，Bufferbloat 問

表 4 提案方式ありの場合の TCP スループットの割合
Table 4 Ratio of TCP throughput when proposed method used.

	none	tsq	codel
スループット割合	97.6%	102.4%	98.3%

題に対応していない Linux 端末は 500 から 600 パケット程度が，端末の送信キューにバッファリングされていることが分かる．これに対して，TCP small queues の Linux 端末と CoDel の Linux 端末における滞留パケット数は 150 パケット以下に抑えられており，送信キューで滞留するパケット数が減少することによってキューイング遅延を小さくして Bufferbloat 問題を抑制することができていると考えられる．図 7 (b) において，平均 MAC データレートが 50 Mbps 以下の範囲では，いずれの種類の端末に対しても，キュー長が 50 パケット以下に抑えられており，その結果，平均 RTT が抑えられたといえる．50 Mbps を超える範囲では，Bufferbloat 問題に対応していない端末において，送信キュー長の減少があまり見られないが，RTT の増加は見られないため問題はないと考えられる．

図 8 (a) は，TCP 通信の平均スループットが端末の種類にはあまり依存していないことを示す．また図 8 (b) と図 8 (a) を比較すると，提案方式を導入した場合に，3 種類の Linux 端末および Windows 端末と Mac OS 端末における平均スループットは，ほぼ同様の結果を示している，

提案方式の有無における TCP 通信の平均スループットを数値的に比較する．図 8 (a) と (b) の結果は，同じ平均 MAC データレートに対して得られているわけではないので，各端末の結果を直線で近似しその傾きの違いをもって評価することとしたい．提案方式なしの傾きに対する提案方式ありの傾きの割合で測定したスループットの割合を**表 4** に示す．なおこの結果は測定回数の多い Linux 端末の 3 種類について示している．この結果からも提案方式を導入することによる TCP スループットの低下はほぼないといえる．

本節の最後に，各測定における測定値の時間的変化について言及したい．各測定は TCP 通信と Ping 通信を開始してから 60 秒間継続させたもので，TCP コネクション確立の動作なども含まれている．そこで，例として，平均 MAC データレートが約 13.5 Mbps と約 110 Mbps の場合における Ping 通信の RTT と TCP スループットの時間変化を**図 9** と**図 10** にそれぞれ示す．両図では提案方式がない場合とある場合（without / with proposal）の双方を示している．これらから，Ping 通信 RTT も TCP スループットも測定の開始直後から平均値に近づいていることが分かる．このため，図 6 から図 8 で得られた 60 秒間の平均値は，十分安定した値であると考えられる．

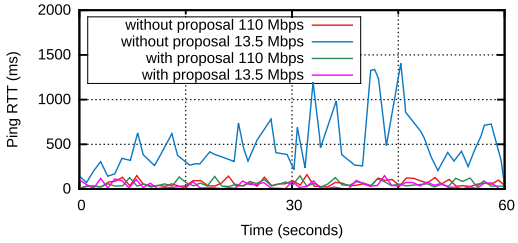


図 9 Ping RTT の時間変化
Fig. 9 Time variation of Ping RTT.

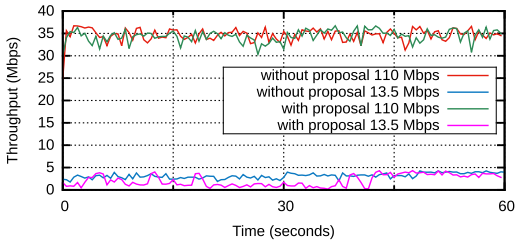
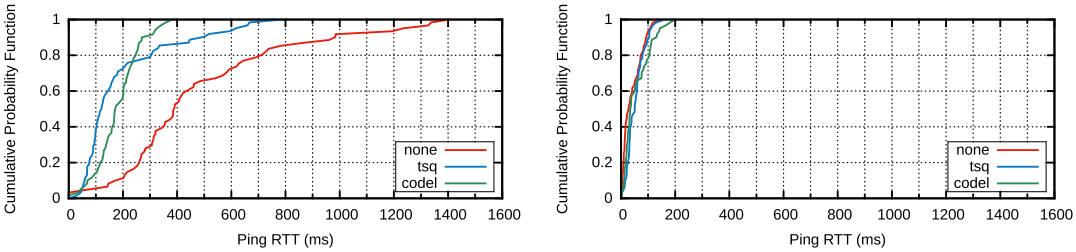


図 10 TCP スループットの時間変化
Fig. 10 Time variation of TCP throughput.

4.3 個別の測定の評価

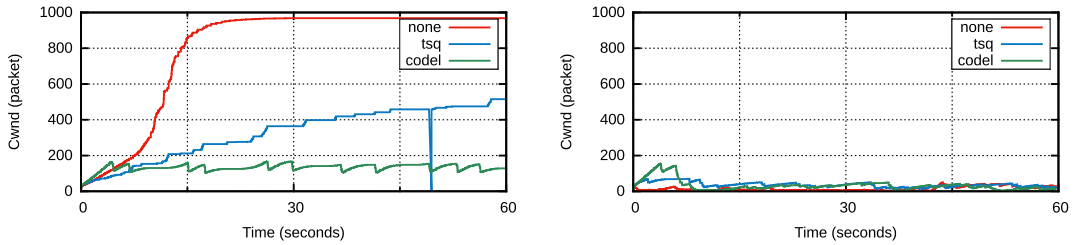
次に本節では，4.2 節で示した 60 秒間の個別の測定の詳細について，Linux 端末の通信を対象として述べる．**図 11** に平均 MAC データレートが約 12 Mbps だった場合の Ping 通信での個々の RTT の累積分布を示す．(a) が提案方式を用いていない場合，(b) が提案方式を AP に実装した場合である．提案方式を用いない場合，この測定では，Bufferbloat 問題に対応していない端末，TCP small queues を適用した端末，CoDel を適用した端末で，Ping RTT の平均値がそれぞれ約 500 ミリ秒，約 200 ミリ秒，約 200 ミリ秒であった．その詳細を見ると，Bufferbloat 問題に対応していない端末では 0 ミリ秒から 1,400 ミリ秒まで分布していた．TCP small queues と CoDel の場合でも，それぞれ最大値が 800 ミリ秒と 400 ミリ秒であった．提案方式を AP に実装しない場合は，いずれも Ping 通信の RTT が平均値の 2 から 4 倍の値まで広がっていた．これに対して，提案方式を AP に実装した場合は，いずれの場合も Ping RTT の平均値が 100 ミリ秒以下で，分布の最大値も 200 ミリ秒よりも小さい．このためユーザレベルの応答性の悪化を解消できることが期待できる．

これらの結果は，**図 12** に示した TCP 輻輳ウィンドウ（congestion window: cwnd）の時間変化から説明できる．なおこの図の結果は，端末において tcpprobe [12] を用いて測定したものである．提案方式を導入していない場合（図 12 (a)），Bufferbloat 問題に対応していない端末と TCP small queues を適用した端末では，パケットロスが発生せず輻輳ウィンドウが上昇し続ける．このため，TCP からの送信のレートが時間とともに増加し，それが内部の送信キューに蓄積され，Ping 通信の RTT 増加につながったと考えられる．CoDel を用いた端末ではそのアルゴリズムに



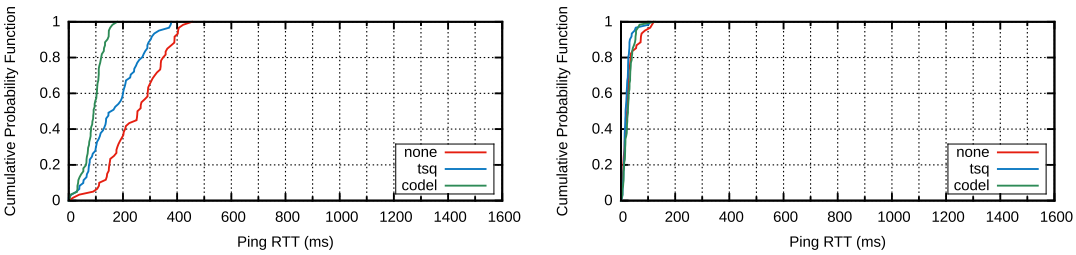
(a) 提案方式なし　(b) 提案方式あり

図 11 平均 MAC データレートが 12 Mbps の場合の Ping 通信の RTT の累積分布
Fig. 11 Cumulative probability function of Ping RTT when average MAC data rate is 12 Mbps.



(a) 提案方式なし　(b) 提案方式あり

図 12 平均 MAC データレートが 12 Mbps の場合の TCP 通信の輻輳ウィンドウの時間変化
Fig. 12 Time variation of TCP congestion window size in TCP when average MAC data rate is 12 Mbps.



(a) 提案方式なし　(b) 提案方式あり

図 13 平均 MAC データレートが 45 Mbps の場合の Ping 通信の平均 RTT の累積分布
Fig. 13 Cumulative probability function of Ping RTT when average MAC data rate is 45 Mbps.

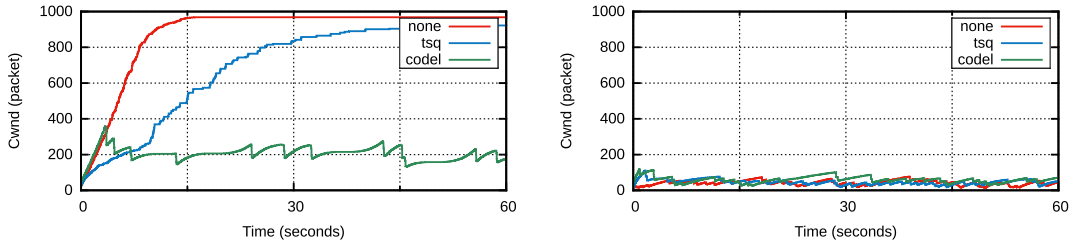
よってパケットロスが発生し，3 種類のうちでは輻輳ウィンドウが小さく抑えられた．これに対して提案方式を導入した場合は，図 12 (b) のように 3 種類ともに輻輳ウィンドウは低いままとなっている．

平均 MAC データレートが約 45 Mbps だった場合の Ping 通信の RTT の累積分布を**図 13** に示す．提案方式を使用しない場合（図 13 (a)），Bufferbloat 問題に対応していない端末，TCP small queues を適用した端末，CoDel を適用した端末で，Ping RTT の平均値がそれぞれ約 250 ミリ秒，約 180 ミリ秒，約 100 ミリ秒であった．分布を見るとその最大値が，Bufferbloat 問題に対応していない端末と TCP small queues を適用した端末では，双方とも 400 ミリ秒まで，CoDel を用いた端末では 200 ミリ秒程度までそれぞれ広がっている．この例でも最大の RTT は，平均値の 2 倍程度となっていることが分かる．これに対して，図 13 (b) に示すように，提案方式を AP に導入する場合は，いずれの場合も Ping RTT の平均値および最大値を低く抑えることが可能となっている．また**図 14** に平均 MAC データレートが 45 Mbps の場合の TCP 輻輳ウィンドウの時間変化を示す．これは 12 Mbps の場合と同様な結果を得ている．

これらの結果から，提案方式は輻輳ウィンドウを低く保つことで Bufferbloat 問題に対処することができているといえる．

4.4 リトライアウトインデックスの影響

提案方式においては，リトライアウトインデックスの値を表 1 のように定めた．これら値は筆者らの先の研究 [2], [3] を参考にして決定している．この研究では，送信側で故



(a) 提案方式なし　(b) 提案方式あり

図 14 平均 MAC データレートが 45 Mbps の場合の TCP 通信の輻輳ウィンドウの時間変化
Fig. 14 Time variation of congestion window size in TCP when average MAC data rate is 45 Mbps.

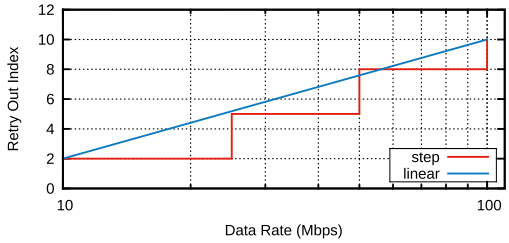


図 15 リトライアウトインデックスの決定方法
Fig. 15 Determination of retry out index.

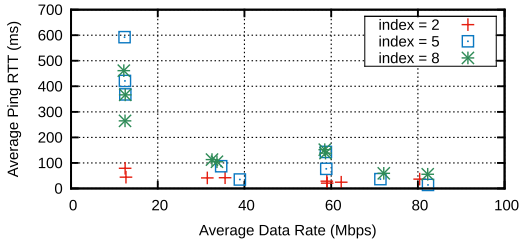


図 16 リトライアウトインデックスを変更した場合の Ping の平均 RTT
Fig. 16 Average RTT in Ping for different retry out index.

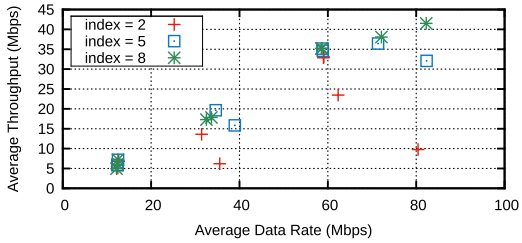


図 17 リトライアウトインデックスを変更した場合の平均 TCP スループット
Fig. 17 Average TCP throughput for different retry out index.

意に無線 LAN の最大再送回数を制限する場合，6.5 Mbps や 13.5 Mbps の MAC データレートにおいては制限値を 2 回とするのが適当なことを実験的に確認している．一方，MAC データレートが 100 Mbps においては最大再送回数として 10 を用いている．これを参考に提案方式では，受信側で使用するリトライアウトインデックスを以下のように定めた．

- 10 Mbps 以下の MAC データレートに対しては，リトライアウトインデックスを 2 とする．
- MAC データレート 100 Mbps に対しては，インデックスを 10 とする．
- 10 Mbps と 100 Mbps の間に対しては，**図 15** の点線のように，ログスケールの MAC データレートに対して 2 と 10 の間を補間する直線を想定する．その直線を参考にして，図の実線のような階段状の値を採用する．

しかし，図の直線補間とそれに基づく階段状の値の設定は，理論的な背景があるわけではない．そこで，リトライアウトインデックスを表 1 の値から変更した場合の性能への影響を評価した．

具体的な評価方法は次のとおりである．4.1 節で示した実験環境において，AP に提案方式を導入し，リトライアウトインデックスを 2，5，8 の値で固定する．Bufferbloat に対応していない Linux 端末を異なる場所に設置し通信実験を行い，Ping 通信の RTT と TCP スループットを測定する．

図 16 と**図 17** に測定結果を示す．図の表記は 4.2 節に示したものと同様である．図 16 では，平均 MAC データレートが 20 Mbps 以下かつリトライアウトインデックスが 5 と 8 の場合，300 ミリ秒から 600 ミリ秒と Ping 通信の RTT が増加している．30 Mbps から 60 Mbps の平均 MAC データレートでは，リトライアウトインデックスが 8 の場合の Ping RTT が他に比べて若干大きくなっている．

一方，図 17 では，30 Mbps 以上の平均 MAC データレートの場合において，リトライアウトインデックスが 2 の場合スループットが小さい．顕著な例では，リトライアウトインデックスが 2 かつ平均データレート 80 Mbps の場合，10 Mbps という著しく低いスループットを記録している．また平均 MAC データレートが 80 Mbps を超えると，リトライアウトインデックスが 5 の場合も若干低いスループットを記録している．

以上の結果から，100 Mbps までの MAC データレートの範囲において，表 1 に示したように階段状にリトライアウトインデックスを決定するのが有効であると考えられる．固定的に値を設定する場合よりも，高いデータレートでは

スループットを落とさず，かつ低いデータレートにおいて遅延を減少させることができる．

5. おわりに

本稿では，IEEE802.11n 無線 LAN の端末からアクセスポイントへのアップロード TCP 通信において，動的な MAC データレート切替えに起因する TCP 通信の遅延増加（Bufferbloat 問題）にアクセスポイントのみで対応する方法を提案した．具体的には，以下のアプローチをとる．(1) データの受信側であるアクセスポイントにおいて MPDU の再送回数を推定する．(2) 低い MAC データレートで転送された TCP を含む MPDU に対しては，本来の最大再送回数よりも小さなリトライアウトインデックス回の再送があった時点で，紛失したとして処理し，次の MPDU の受信処理に移行する．これにより，低いデータレートにおいてパケット損失の頻度が増加し，TCP での輻輳ウィンドウの減少につながる．アクセスポイントに提案方式を実装するだけで，無線 LAN に収容されたすべての端末で発生する Bufferbloat 問題に対処することが可能となる．

筆者らはパソコンを用いたアクセスポイントに提案方式を実装し，アップロード TCP 通信のスループットと，並行して実行した Ping 通信の往復遅延時間を測定した．その結果，低い MAC データレート使用時の遅延時間を減少できること，TCP のスループットの大きな低下を引き起こさないこと，Bufferbloat 問題に対応する他の方法を実装した端末の通信にも影響を与えないことなどを明らかにした．

参考文献

[1] IEEE Standard for Information technology, Local and metropolitan area networks Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications (2012).

[2] 野元祐孝, 加藤聰彦, 策力木格, 大坐畠智：IEEE 802.11n 無線 LAN 上の TCP 通信における大幅なレート変更時の遅延増加の改善手法，信学技報，Vol.113, No.208, CQ2013-40, pp.75–78 (2013).

[3] Nomoto, M., Kato, T., Celimuge, W. and Ohzahata, S.: Resolving Bufferbloat Problem in 802.11n WLAN by Weakening MAC Loss Recovery for TCP Stream, *Proc. IASTED PDCN 2014* (Feb. 2014).

[4] Gettys, J. and Nichols, K.: Bufferbloat: Dark Buffers in the Internet, *ACM Queue* (Nov. 2011).

[5] Allman, M.: Comments on Bufferbloat, *ACM SIGCOMM Computer Communication Review*, Vol.43, No.1 (2013).

[6] Nichols, K. and Jacobson, V.: Controlling Queue Delay, ACM Queue, Networks, Vol.10, No.5, pp.1–15 (2012).

[7] Dumazet, E.: [PATCH v3 net-next] tcp: TCP Small Queues (July 2012), available from ⟨http://article.gmane.org/⟩.

[8] Floyd, S. and Jacobson, V.: Random Early Detection Gateways for Congestion Avoidance, *IEEE/ACM Trans. Networking*, Vol.1, No.4 (1993).

[9] Rhee, I. and Xu, L.: CUBIC: A new TCP-friendly high-speed TCP variant, *SIGOPS Operating Systems Review*, Vol.42, No.5 (2008).

[10] iperf, available from ⟨http://iperf.sourceforge.net/⟩.

[11] ath9k Linux Wireless, available from ⟨http://wireless.kernel.org/en/users/Drivers/ath9k⟩.

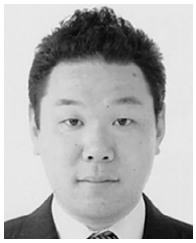
[12] Linux foundation, tcpprobe, available from ⟨http://www.linuxfoundation.org/collaborate/workgroups/networking/tcpprobe⟩.

野元 祐孝

平成 18 年電気通信大学電気通信学部情報通信工学科卒業．平成 28 年同大学大学院情報システム学研究科博士後期課程単位取得退学．無線ネットワークの通信プロトコルに関する研究に従事．

策力木格 （正会員）

平成 22 年電気通信大学大学院情報システム学研究科情報ネットワークシステム学専攻博士課程修了．同年より電気通信大学大学院情報システム学研究科助教．平成 27 年より同大学准教授．アドホックネットワーク，通信プロトコル等の研究に従事．博士（工学）．



大坐畠 智 （正会員）

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

加藤 聰彦 （正会員）

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