

無線 LAN 環境における TCP の再送方式の提案と実装

田村 陽介

戸辺 義人

徳田 英幸

慶應義塾大学 政策・メディア研究科 慶應義塾大学 SFC 研究所 慶應義塾大学 環境情報学部

本論文では無線 LAN 上における TCP の性能を分析し、効率的な TCP の再送方式を提案する。無線 LAN 上における通信において頻繁なセグメントの欠損は輻輳制御をまねき送信量を制御する。このような状態の時、従来の TCP の再送方式は効果的に動作せず、再送タイマの不必要なタイムアウトが通信性能の低下を引き起こしている。TCP の Fast Retransmit のタイミングとなる重複した ACK を受けとる数値を TCP のウィンドウの大きさに合わせて動的に変更することにより従来の TCP と比べて 10-15% スループットが向上した。

Efficient Retransmit Scheme for TCP in Wireless LANs

Yosuke Tamura[†], Yoshito Tobe^{††} and Hideyuki Tokuda[‡]

[†]Graduate School of Media and Governance, Keio University

^{††}Keio Research Institute at SFC, Keio University

[‡]Faculty of Environmental Information, Keio University

This paper looks at TCP performance in wireless LANs, and finds that TCP coarse (retransmission) timeouts are the main cause of poor end to end performance. To fix this, we propose changing the number of duplicate acks required before triggering a fast retransmit, and more importantly, making it a dynamically calculated value that ranges between 1 and 3. This is found to improve TCP performance by 10-15%, particularly in the case of TCP sources with small windows (less than 5 packets).

1 はじめに

近年、無線 LAN は、その高い移動性や敷設の容易さ故に、大学のキャンパス LAN や企業 LAN など普及しはじめている。ほとんどの無線 LAN 環境は、有線と無線とのブリッジとなるアクセスポイントが存在し、そのアクセスポイントに対して複数の計算機がアクセスするように設置されている。同一の無線媒体を複数の計算機で共有するような状態では、その使用率が上がるに従って高い確率でセグメントが欠損が起こる。この高欠損率の要因は無線媒体上のセグメントが無線通信特有の干渉・マルチパ

スフェージングといった影響によるビットエラーや、物理層でのフレームの衝突が検知できない為、上位層での再送が必要となるということが挙げられる。

信頼性を保証する通信プロトコルである Transmission Control Protocol (TCP)[12] は、WWW, FTP[13], Telnet[14] など現在のインターネット上で多くの通信アプリケーションで使われている。しかし TCP は本来有線ネットワークを意識して設計されたプロトコルである為、無線ネットワーク上で実現した場合、その再送処理、輻輳制御などが効率的に動作しているとは言えない。無線通信上で効率

的な通信環境を実現する為には既存の TCP をさらに改良することが重要である。

頻繁なセグメントの欠損は、TCP の輻輳制御を引き起こし、ウィンドウを縮小させる。その為、送信側の送信量が少なく制限されてしまい、欠損が起こった時に Fast Retransmit[8] による再送をするのに十分な ACK を受けとることができない状態が起こり得る。このような場合、従来の TCP では送信側は存在しない重複した ACK を待ち続け、結果的に再送タイマがタイムアウトを起こし、再送することとなる。これは大きなタイムロスとなり通信性能を低下させている。

本論文では、TCP の送信量を考慮し Fast Retransmit を行なうタイミングを動的に変更させるアルゴリズムを提案する。我々はこのアルゴリズムを 2.4GHz の WaveLAN 上で実験、評価をし、無線通信のように高い欠損率のネットワークで有効であることを実証する。

2 章では、TCP の再送処理を解析した結果と、問題点を述べる。3 章では我々の提案する新しい再送方式の設計と実装について述べ、4 章で現在の TCP である TCP-Reno との比較を行ないながら性能評価を行なう。5 章で関連研究について述べ、最後に今後の課題とまとめを述べる。

2 無線通信における TCP の解析

本章では、無線通信においてどのように再送が行なわれているか解析し、問題点を示す。図 1 が実験環境である。実験には、2Mbps の帯域を持つ AT&T の WaveLAN に繋がれた無線計算機 MH1, MH2, MH3, MH4 (Dynabook SSR-590, Pentium 90MHz, RAM 16M, FreeBSD2.2.1-R) の 4 台と、10Mbps の帯域を持つ Ethernet に繋がれた CH (PC-AT, Pentium Pro 200MHz, RAM 32M, FreeBSD2.2.1-R) の 1 台を用いた。MH から CH に対して TCP を用いてデータを送信し、どの実験においても、CH のキューイングのオーバーフローは起こらなかった。通信の挙動をネットワーク解析ツールである tcpdump と東陽テクニカ Sniffer を用いて解析した。

2.1 再送処理

MH1, MH2, MH3 の 3 台の無線計算機から CH に向けて 1Mbyte のデータを同時に送信した。ここでは、図 2, 図 3, 図 4 は、それぞれ再送が行なわれた一部分を拡大した図である。

現在の TCP では、欠損したセグメントは Fast Retransmit, もしくは再送タイマのタイムアウトに

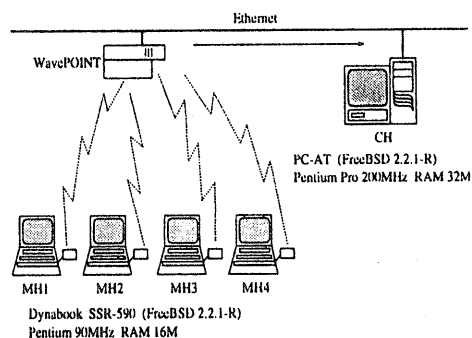


図 1: 実験環境

より再送される。図 2 は、再送タイマのタイムアウトによる再送を示し、図 3 は Fast Retransmit による再送が起こった時のセグメントのシーケンス番号の推移を示したグラフである。再送タイマによる再送では、欠損したセグメントが最初に送られてから再送されるまで 2 秒近くかかっているのに対し、Fast Retransmit では、その 10 分の 1 の 0.2 秒で再送が行なわれている。このように Fast Retransmit は再送タイマのタイムアウトを待たずに素早い再送ができる為、TCP の性能を大幅に効率化している。

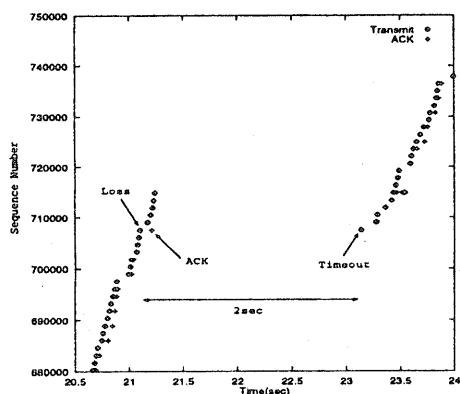


図 2: Timeout による再送

Fast Retransmit が有効となる前提として、送信側は重複した ACK を十分受信できる状態ということが挙げられる。即ち送信側はある程度のウィンドウの大きさを保持して、欠損セグメントよりも後のセグメントを送っている必要がある。無線通信など欠損率の高い通信や、TCP のコネクション接続時における TCP ウィンドウの小さい時など、送信量が

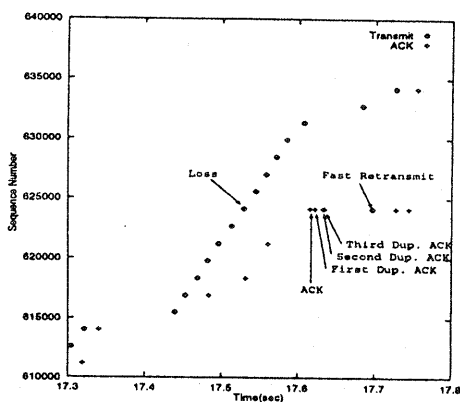


図 3: Fast Retransmit による再送

少なく制限される為、Fast Retransmit が有効とならない場合がしばしば起こる。図 4は、その時の様子を示したグラフである。送信側では、ウィンドウが小さい為、欠損セグメント以降のセグメントを 2 つしか送ることができない。最大でも 1 つの重複した ACK を受け取ることしかできないにもかかわらず、その ACK を受け取っても欠損セグメントは再送されていない。結果的に再送タイマのタイムアウトにより再送が行われている。現在の TCP では、このようにウィンドウが小さな状況では、Fast Retransmit によって再送をすることが不可能となる。

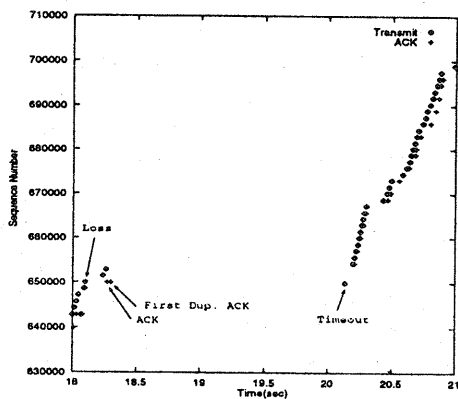


図 4: ウィンドウの大きさが小さい時の再送

2.2 ウィンドウ

送信側が存在しない ACK を待つ状態は、ウィンドウが小さく制御されている時に起こる。図 5, 図 6, 図 7は、3 台の無線計算機で同時にデータ転送を

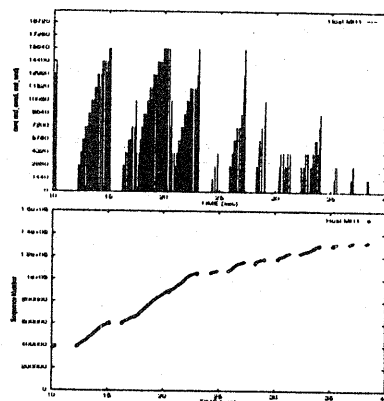


図 5: MH1 のウィンドウサイズ (上) とシーケンス番号 (下) の変化

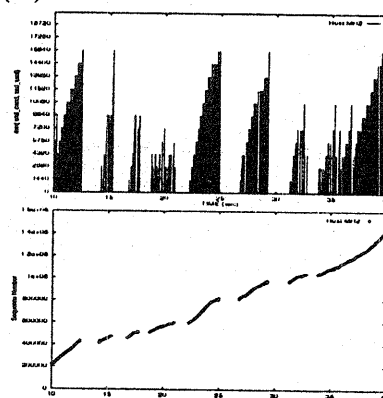


図 6: MH2 のウィンドウサイズ (上) とシーケンス番号 (下) の変化

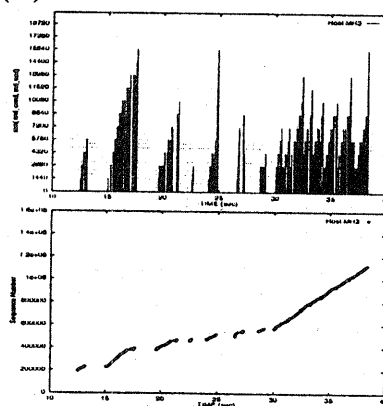


図 7: MH3 のウィンドウサイズ (上) とシーケンス番号 (下) の変化

した時の TCP のウィンドウの大きさとシーケンス番号の推移を示したグラフである。頻繁にウィンドウの大きさが変化していることが分かる。影になっている部分が、ウィンドウの大きさが4 セグメント以下の部分であり、この時にセグメントの欠損が起こると Fast Retransmit が有効とならずに再送タイマのタイムアウトが必ず引き起こされることとなる。

このようにウィンドウが頻繁に収縮するのは、TCP の頻繁な再送が輻輳制御を導いているからである。同じ実験を帯域 10Mbps の Ethernet で行なったグラフが図8である。有線 LAN では一般的に欠損率も低く欠損が起こったとしてもすぐにウィンドウの大きさは回復する。そして送信側は、Fast Retransmit が動作するのに十分なセグメントを送信することができ、以上のように無線と有線との通信では明らかに特徴が異なる。既存の TCP の再送方式を無線通信にも適応できるように改良することが重要である。

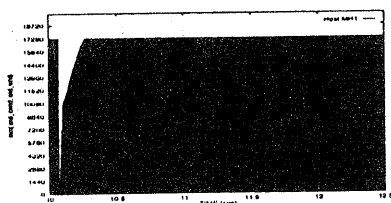


図 8: 有線ネットワークでのウィンドウサイズの変化

3 Efficient Fast Retransmit

本章では前章で述べた問題点をもとに我々の提案する Efficient Fast Retransmit (EFR) という新しい TCP の再送方式の設計と実装について述べる。

3.1 EFR の設計

EFR の目的は、送信側がセグメントの再送処理時に存在しない ACK を待ち続けて再送タイマのタイムアウトを引き起こすという状態を避けることである。現在、Fast Retransmit のタイミングとなる重複した ACK を受け取る数はスタティックに 3 と定められている [15]。EFR ではこの値を TCP のウィンドウの大きさを考慮しながら最適な値に動的に変更して効率的な再送を実現する。

図 9 は、EFR と TCP-Reno の再送処理の違いを示している。図では送信側は 4 セグメント分の大きさのウィンドウを持っている。1 から 4 までのセグメントを ACK を待たずに送信する。1 が欠損した場合、送信側は 1 を要求する 2, 3, 4 のセグメント

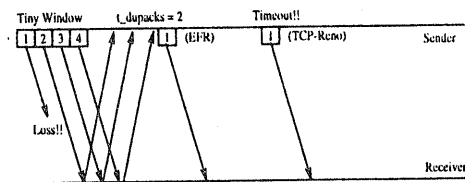


図 9: EFR と TCP-Reno の比較

の ACK を 3 つ受け取る。この時 1 に対する重複した ACK は 2 つである為、従来の TCP-Reno では、Fast Retransmit は起こらず再送タイマがタイムアウトして再送することになる。EFR では、4 の ACK を受け取った時点で、これ以上 ACK が存在しないことを予測し Fast Retransmit で再送を試みる。

EFR では、最初に重複した ACK を受け取った時点で受信可能な ACK の数を計算する。ここではウィンドウの大きさが 4 であるので EFR では、2 つの重複した ACK までは受け取れることを計算し、2 つ受け取った時点で再送を行う。この処理によりタイムアウトを待つ時間を浪費せずに再送を行うことが可能となる。

3.2 誤った再送問題

TCP の下位プロトコルである Internet Protocol (IP) は、IP パケットに信頼性のある通信を保証しない。その為、TCP セグメントは、順序が間違っただけ可能性がある。このような時、受信側では順序が後になったセグメントが欠損していると仮定し重複した ACK を返す。Fast Retransmit を行なうタイミングを早めれば早める程、誤った再送を引き起こす可能性が高くなる。誤った Fast Retransmit は不要な輻輳制御 (Fast Recovery) を導き現在の輻輳ウィンドウの大きさを半分にしてしまう。この為、単に Fast Retransmit のタイミングを 1 や 2 に設定することが良いとは言えない。EFR では、ウィンドウが最大でも 4 セグメントの大きさの時にしか動作をしないので、誤った輻輳制御が起こったとしても通信性能に与える影響は少なく、全体的に利点の方が高くなる。

3.3 EFR の実装

TCP では、通信の状態を保持する為の変数が TCP Control Block (TCB) として *tcp_var.h* の中で定義されている。この中に新しく、ウィンドウの大きさを保持しておく変数 (*snd_cur*) と Fast Retransmit のタイミングとなる重複した ACK の数のしきい値を格納す

る変数($tcp\text{prexmtthres}$)を追加した。 $tcp\text{prexmtthres}$ の値は、それぞれのセグメントに対して最初に重複したACKを受けとった時点で、 snd_cur の値を基に動的に変更される。 $tcp\text{prexmtthres}$ の値は3を越えないように設定される。

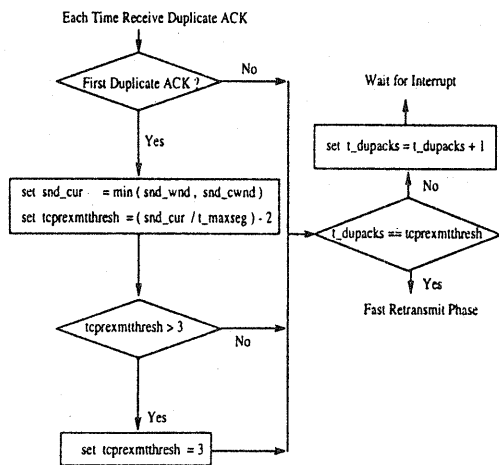


図 10: EFR アルゴリズム

図 10は、EFR のアルゴリズムを示した図である。重複した ACK を受け取ると、それがそのセグメントに対する最初の重複した ACK かどうかを調べる。最初であった場合、現在のウィンドウの大きさを、受信側からのウィンドウ通知により設定されている受信バッファの空き容量 (snd_wnd) と送信側の輻輳ウィンドウ (snd_cwnd) の最小値を snd_cur に格納する。その値は byte で示されている為、送信セグメントの大きさ (t_maxseg) で割り、さらに、欠損セグメントと重複していない最初の ACK の分の 2 セグメント分を引く。この値が、送信側が受信可能な最大重複 ACK 数となり、 $tcp\text{prexmtthres}$ に格納される。EFR の目的は、ウィンドウが小さい時に存在しない ACK を待つ状態を避けることなので、ウィンドウが大きくて $tcp\text{prexmtthres}$ の値が 3 よりも大きくなった場合は、本来の値である 3 に変更する。重複した ACK の数は、 $t_dupacks$ に格納されている。この値が $tcp\text{prexmtthres}$ に一致した時点で Fast Retransmit による再送が行なわれる。

4 評価

我々は EFR を FreeBSD2.2.1R 上に実装し、AT&T WaveLAN 環境で評価を行なった。無線計算機の数

ブリッジである WavePOINT を介して繋がれている Ethernet 上の計算機に対してデータを送信し、その性能を TCP-Reno と比較、評価した。実験環境は、図 1 で示されている。本章では、再送処理の特徴の比較と、転送性能の比較を行う。

4.1 転送特徴

無線計算機を 2 台から 4 台まで増やしそれぞれの計算機から有線につながれた計算機に対して一斉にデータを送信した。50Mbyte のデータを送信した時の 1 台の再送処理の様子を TCP-Reno と比較、解析した。図 11 は、Fast Retransmit による再送と再送タイマのタイムアウトによる再送の回数を示したグラフである。EFR を用いると本来タイムアウトになっていた欠損が Fast Retransmit により再送され、さらに Fast Recovery により輻輳ウィンドウを半分の大きさに戻す。この為、より多くのセグメントがネットワーク上に送信され、その分セグメントの衝突の率が TCP-Reno と比べて増加する。しかし前章で述べたように再送タイマは、Fast Retransmit に比べて数十倍の時間のオーバーヘッドとなっている。その為タイムアウトの数が少なくなっていることが全体として性能の向上につながる。図 12 は、その割合を示した図である。TCP-Reno と比べて EFR は再送タイマのタイムアウトによる再送の割合が増加している。

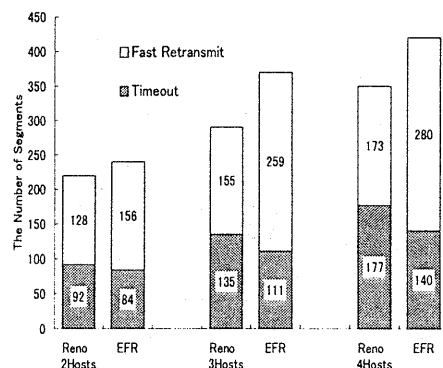


図 11: Fast Retransmit と 再送タイマによる再送の数

図 13 は、再送タイマがタイムアウトした時に送信側が受け取っていた重複した ACK の数を TCP-Reno と EFR とで比較したグラフである。重複した ACK を 3 つ以上受け取っている状態とは、Fast

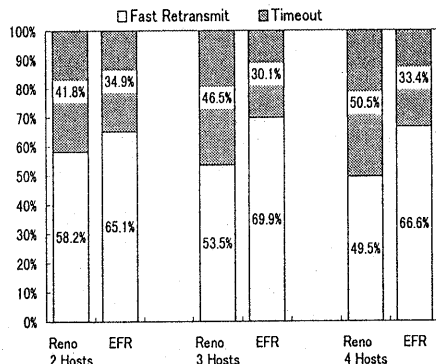


図 12: Fast Retransmit と 再送タイマによる再送の割合

Retransmit で再送したセグメントが再び欠損し、タイムアウトが起こったという状態である。EFR の目的は 1 つもしくは 2 つの重複した ACK をすべて受け取っているにもかかわらずタイムアウトを起こしているという割合を減らすことである。図 13 では明らかにその割合が減少している。その数が 0 とならないのは、一つのウィンドウ内で複数の欠損が起こった時には EFR は重複した ACK を最大数受け取ることができずに再送ができないからである。

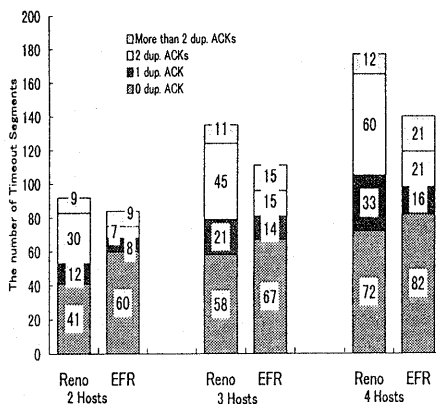


図 13: 再送タイマのタイムアウト時の重複した ACK の数の割合

4.2 転送性能

図 14 は、無線計算機の数 を 2 から 4 まで増加し、1Mbyte, 2Mbyte, 4Mbyte, 8Mbyte を 50 回繰り返

し送った時の平均と最小値, 最大値を示したグラフである。データの大きさが大きくなる程分散が小さくなる。さらに平均して EFR は TCP-Reno よりも高いスループットが計測された。

図 15 は、すべての無線計算機のスループットの合計を示している。EFR は TCP-Reno に比べて総使用帯域数でも明らかに高い性能を示している。このようにタイムアウトは非常に大きなタイムロスとなる為、EFR では欠損率が増加しても結果的に通信性能は良くなる。

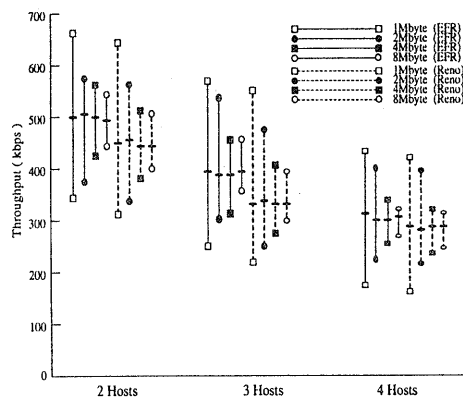


図 14: スループットの比較

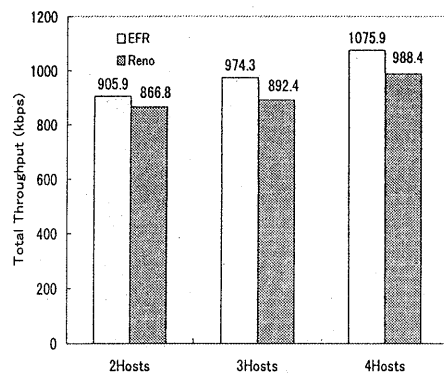


図 15: 合計スループットの比較

5 関連研究

今日まで無線ネットワーク上での TCP の改良を目的とした多くの研究がなされている [2, 3]. この章では、EFR と関連のある TCP の性能向上に関するいくつかの研究を挙げる。

5.1 TCP Vegas

TCP-Vegas[4, 1] は, TCP-Reno の挙動を X-Kernel 上に実装され, その挙動を解析することにより再送アルゴリズムや, ウインドウ更新アルゴリズムなどに大幅な修正を加えたものである. TCP-Vegas の再送アルゴリズムは, 粒度の細かなタイマにより測定された RTT を基本として, 再送を行なう.

Vegas の提案においても, ウインドウが小さな時には FR が有効にならないということが述べられている. しかし Vegas では, RTT を基本としている為, 遅延の変化が起きた場合など正確にセグメントの欠損を判断することが難しく, 新しいタイマの追加など実装も複雑である. EFR ではウインドウを基本として送出量を計算している為, 正確である. また簡単な実装で効率的な再送を実現できる.

5.2 NewReno, SACK and FACK

NewReno[7] は TCP の Fast Retransmit に変更を加え, 一つのウインドウ内で複数の欠損が起こった時に, 2 番目以降のセグメントの欠損に対してタイムアウトを招かず効率的に再送を行なうアルゴリズムである. SACK[9] は一つのウインドウ内で複数の欠損が起こった時に受信側のバッファの状態を TCP のオプションとして付け加え, 送信側に知らせ素早い再送を可能とするアルゴリズムである. さらに SACK の情報から効率的な輻輳制御を行なう FACK[11] なども提案されている.

これらのアルゴリズムは, 一つのウインドウ内で複数の欠損が起こった時に有効である [5]. 我々の提案するアルゴリズムは, ウインドウの小さい状態での欠損に対して効率的である. これらを組み合わせた実装によりさらに性能の向上が期待される.

6 今後の課題

今回我々は, EFR のアルゴリズムを, 無線 LAN 環境で測定, 評価した結果について述べた. しかし我々が指摘した TCP の問題点は, 有線通信や, WAN 環境でも起こり得る. 特に WAN 環境における WWW アクセスなど, 欠損率が高く, しかも遅延が大きな通信状態では EFR は非常に効果的であると考えられる. また NewReno や SACK と組み合わせることにより複数の欠損に対しても対応のできるようなアルゴリズムへの拡張など今後の課題である.

7 まとめ

本論文では無線 LAN 上における TCP の性能を分析し, 効率的な TCP の再送方式を提案した. 無

線 LAN 上における通信において頻繁なセグメントの欠損は輻輳制御をまねき送信量を制御する. このような状態の時, 従来の TCP の再送方式は効果的に動作せず, 再送タイマの不必要なタイムアウトが通信性能の低下を引き起こしている. TCP の Fast Retransmit のタイミングとなる重複した ACK を受けとる数値を TCP のウインドウの大きさに合わせて動的に変更することにより従来の TCP-Reno と比べて 10-15% スループットが向上することを実証した.

謝辞

本研究を進めるにあたり, 慶応義塾大学次世代マイクロカーネルプロジェクトの皆様にご多大な御助言・御協力を頂きました. ここに感謝の意を表します.

参考文献

- [1] J. S. Ahn, P. B. Danzig, Z. Liu, and L. Yan. Evaluation of TCP Vegas: Emulation and Experiment. In *Proceedings of ACM SIGCOMM '95, Oct. 1995. TCP Performance over Wireless*
- [2] H. Balakrishnan, V.N. Padmanabhan, S. Seshan, and R.H. Katz. A Comparison of Mechanisms for Improving TCP Performance over Wireless Links. In *Proceedings of ACM SIGCOMM '96, Aug. 1996.*
- [3] H. Balakrishnan, S. Seshan, and R.H. Katz. A Improving TCP/IP Performance over Wireless Networks. In *Proceedings of ACM MOBICOM '95, Nov. 1995.*
- [4] L. S. Brakmo, S. W. O'Malley, and L. L. Peterson. TCP Vegas: New Techniques for Congestion Detection and Avoidance. In *Proceedings of ACM SIGCOMM '94, May. 1994.*
- [5] K. Fall and S. Floyd. Simulation-based Comparisons of Tahoe, Reno, and SACK TCP. *ACM SIGCOMM, vol.26, no.3, pp.5-21, July 1996.*
- [6] J. C. Hoe. Improving the Start-up Behavior of a Congestion Control Scheme for TCP. In *Proceedings of ACM SIGCOMM '96 Aug. 1996.*
- [7] V. Jacobson. Congestion Avoidance and Control. In *Proceedings of ACM SIGCOMM '88 Aug. 1988.*
- [8] V. Jacobson and R. Braden. TCP Extensions for Long-Delay Paths. *Network Working Group, Request for Comments: 1072 Oct. 1988.*
- [9] M. Mathis and J. Mahdavi. Forward Acknowledgment: Refinding TCP Congestion Control. In *Proceedings of ACM SIGCOMM '96, Aug. 1996.*
- [10] J. Postel. Transmission control protocol. *Network Working Group, Request for Comments: 793, Sep. 1981.*
- [11] J. Postel and J. Reynolds. FILE TRANSFER PROTOCOL(FTP). *Network Working Group, Request for Comments: 765, Sep. 1985.*
- [12] J. Postel and J. Reynolds. TELNET PROTOCOL SPECIFICATION. *Network Working Group, Request for Comments: 854, Sep. 1983.*
- [13] W. R. Stevens. TCP/IP Illustrated, Volume1 The Protocols(p.312). Addison-Wesley Publishing Company. 1994.