モバイルコンピューティングと
ワイヤレス通信 24−18
（2003．3．7）

スプリアス・タイムアウト検出と輻輳ウインドウ制御アルゴリズムに関する研究

三宅 基治，稲村 浩，高橋 修

NTT ドコモ　マルチメディア研究所
〒239-8536 神奈川県 横須賀市 光の丘 3-5
Tel: 046-840-3812, FAX: 046-840-3364
e-mail: {miyake, inamura, osamu}@mml.yrp.nttdocomo.co.jp

あらまし　TCP ストリームを使用した移動通信では，無線区間の遅延ジッタ増加や，移動端末の一時的な圏外などに起因するスプリアス・タイムアウト（以下，STO と呼ぶ）によって go-back-N 再送となり，送信済みのセグメントが再送されるスプリアス再送問題が発生する．本論文では，不要なデータ・セグメント再送と輻輳ウインドウ縮小によるスループット低下抑制のため，go-back-N 再送期間を含めた RTT (Round Trip Time) 監視と，再送セグメントを越えるシーケンス番号をもつ ACK 監視による STO 検出機能および，輻輳ウインドウ制御機能を具備するアルゴリズムを提案する．提案方式では，サーバへの実装のみで動作し，TCP オプションなどの付加情報によるオーバーヘッドがないため，移動通信網への適用が期待される．*ns2* を用いたシミュレーションでは，提案方式による STO 検出および，輻輳ウインドウ制御によりスプリアス再送とスループット低下の抑制が適切に行われることを明らかにする．

キーワード　TCP，スプリアス・タイムアウト，スプリアス再送，輻輳ウインドウ，移動通信

A Spurious Timeout Detection and Congestion Window Control Algorithm

Motoharu Miyake, Hiroshi Inamura, Osamu Takahashi

NTT DoCoMo, Multimedia Laboratories
3-5 Hikarinooka, Yokosuka, Kanagawa, 236-8536, Japan
Tel: +81-46-840-3812, Fax: +81-46-840-3364
e-mail: {miyake, inamura, osamu}@mml.yrp.nttdocomo.co.jp

Abstruct　A spurious timeout (STO) and a spurious retransmission caused by a delay-jitter of wireless links and passage of an outside of service area occur a negative impact on the TCP connection's performance. In this paper, we propose a STO detection and congestion window control algorithm for the suppression both the wasteful go-back-N retransmission and the degradation of throughput. To detect a STO, the proposed algorithm with a sender-side modification observes a RTT (Round Trip Time) after the timeout and an ACK that covers 2 full-size segments and up. This method is suitable for mobile communications, because it dose not require either of additional information (or TCP options). The simulations in *ns2* show the proposed algorithm can suppress both the spurious retransmission and the loss throughput sufficiently.

Keywords　TCP, Spurious timeout, Spurious retransmission, Congestion window, Mobile communications

1 はじめに

バックボーン・ネットワークが広帯域化している昨今，移動通信網を含む End-End 通信では，無線区間の特性が TCP ストリームを用いたデータ伝送に与える影響は大きい．第 3 世代移動通信 (IMT-2000) を構成する W-CDMA (Wideband-Code Division Multiple Access) 方式では，無線区間の通信品質保証に RLC (Radio Link Control) による自動再送機構 (ARQ: Automatic Repeat Request) を用いている [1].

無線状態の悪化により，この RLC レイヤでの再送が繰り返される場合，TCP ストリームにおける遅延ジッタの急激な増加となり，不要な RTO (Retransmission Timeout) を引き起こすため，FreeBSD では RFC2988 [2] によってタイムアウトの発生を抑えている [3]．しかし，ビルやトンネルなどの遮蔽物による数秒から数十秒にわたる比較的長時間の通信中断に対しては，十分な抑制効果は期待できないため，RTO 後，スロー・スタートによるデータ・セグメント（以下では，セグメントと呼ぶ）再送が行われる．この際，サーバでは，オリジナル・セグメントに対する ACK を再送セグメントに対するものと判断し，go-back-N 再送によって後続の再送セグメントを送信するスプリアス再送問題が発生する [4].

スプリアス再送を抑制するため，TCP オプションまたは，統計情報を用いたアルゴリズムが提案されている [4, 5]．Ludwig らは，TCP のタイムスタンプ・オプションを用いることによりスプリアス・タイムアウト（以下，STO とする），スプリアス・ファースト・リトランスミットへの耐性をもつ Eifel アルゴリズム [4] を，Allman らはタイムアウト直後の RTT (Round Trip Time) による STO 検出方法をそれぞれ提案した [5]．これとは別に，Floyd らは TCP の SACK オプションを拡張し，重複セグメントを検出したクライアントは，SACK ブロックを用いてサーバに通知する提案を行った [6]．しかし，Eifel アルゴリズムでは，すべてのセグメントおよび ACK の TCP オプション部分にタイムスタンプが付加するためオーバーヘッドとなる．また，Allman のアルゴリズムや DSACK では，STO 検出のための ACK 検出時間が限定されるため，移動通信へ適用しても，十分な効果を期待することは困難である．

本論文では，移動通信におけるスプリアス再送抑制のため，従来研究されてきた IP 網での STO 検出と，無線網に有効な STO 検出機能を兼ね備えるアルゴリズムを提案する．提案手法では STO 検出後，従来法と同様に輻輳ウインドウをタイムアウト前の値に復元し，スロー・スタートと輻輳回避アルゴリズムによるスループット低下を防止する．

提案方式の STO 検出には，タイムアウト発生直後は Allman のアルゴリズム同様に RTT を，go-back-N 再送期間では連続的に到着する ACK の RTT をそれぞれ監視する．更に，RTT によるあいまいさを排除し，STO 検出精度を高めるため，再送セグメントを超えるシーケンス番号をもつ ACK 監視もあわせて行う．

この結果，提案方式はサーバへの実装のみでよく，TCP オプションなどの付加情報を一切必要としないためオーバーヘッドは発生しない．また，タイムアウト発生から go-back-N 再送を含めた期間での STO 検出を可能としているため，Allman のアルゴリズムのよりも検出期間が長く，DSACK よりも早いタイミングとすることが可能である．以上から，従来法で適用が困難であった従量制課金をもつ移動通信網への適用が容易といった特徴をもつ．

最後に，*ns2* を用いたシミュレーションによって，提案方式による STO 検出および，輻輳ウインドウ制御によりスプリアス再送の抑制が適切に行われることを明らかにする．

2 通信路モデルとスプリアス再送

はじめに，本論文で想定する通信路モデルを図 1 に示す．このうち，クライアントは W-CDMA 方式による無線網と，サーバは IP 網とそれぞれ接続しており，TCP ストリームによるデータ通信がクライアント・サーバ間でなされるものとする．

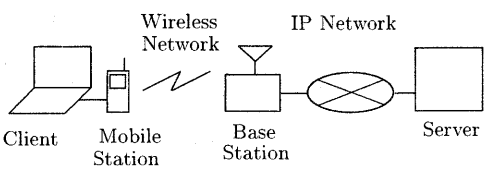

図 1: 通信路モデル

TCP ストリームを使用したデータ通信において，サーバは，各コネクションの RTO タイマの適応的な学習を行う．しかし，移動通信のように無線区間の遅延ジッタ増加や，移動端末の一時的な圏外などによる RTT の急激な変化に対しては十分な効果が得られないため STO が発生し，スロー・スタートによるセグメント再送が行われる [7]．この際，サーバ

からのオリジナル・セグメントを蓄積している基地局では無線状態の回復を待って移動端末に送信を再開する．この結果，クライアントにはオリジナル・セグメントと再送セグメントが重複して届き，それぞれに対して ACK を返信する．

TCP は，スロー・スタート開始後に連続的に到着するこれらの ACK が，オリジナル・セグメントと再送セグメントのいづれに対するかの判断機能を備えていない．従って，オリジナル・セグメントに対する ACK を，再送セグメントに対するものとサーバでは判断し，実際にはパケットロストが発生していない場合でも go-back-N 再送により，送信済みのセグメント再送が繰り返されるスプリアス再送問題が発生する [4]．

W-CDMA 方式による無線網は，IP 網に比べ遅延が大きいため，十分なスループットを得るためには帯域幅遅延積 [8] を考慮したウインドウサイズに設定することが文献 [9] において推奨されている．しかし，RTO によるスロー・スタート，輻輳回避アルゴリズムの適用により，輻輳ウインドウサイズ回復に時間を必要とするため，スループット低下といった問題が発生する．更に，従量制課金による移動通信では，スプリアス再送により無駄なセグメント再送による課金の増加が懸念される．

一例として，*ns*2 [10] を使用し，図 2 に示すシミュレーションモデルによるスプリアス再送問題について明らかにする．移動端末 (MS: Mobile Station) と基地局 (BS: Base Station) 間のデータ伝送帯域幅を 384 kbps, RTT は 300 ms とし，上下対称となるように 150 ms に均等に配置した．バックボーン回線は高速であると仮定し，基地局とサーバ (SV: Server) 間はデータ伝送帯域幅 10 Mbps, RTT は 20 ms とした．また，バルク転送には TCP ストリームを使用し，サーバから移動端末方向にセグメントが伝送される．なお，基地局には十分なキューがあり，パケット破棄は行わないものとする．

図 3 に，サーバ側で取得したスプリアス再送を含む TCP タイム・シーケンスグラフを示す．なお，混同の恐れのない限り以下のタイム・シーケンスグラフの説明では，サーバ側で取得したデータを使用するものとする．図 3 において x 軸は時刻を，y 軸は TCP ストリームのシーケンス番号を表し，太線はセグメント，細線は ACK のシーケンス番号および，広告ウインドウサイズをそれぞれ表す．同図から，9:00:08 に送信したセグメント ⓐ （シーケンス番号 134321 ）に対する ACK 未受信により，サーバは 9:00:10.6 および，9:00:13.5 に同一セグメント ⓑ の再送 (図中 R 記号) を行っている．その後，9:00:15.1 の ACK ⓒ 到着を契機として，go-back-N 再送となり，合計で 42 セグメントが再送されている．更に，クライアントからの広告ウインドウサイズ (64 kB) まで広がっていた輻輳ウインドウは，スロー・スタート，輻輳回避の影響により，大幅に縮小したことが分かる．

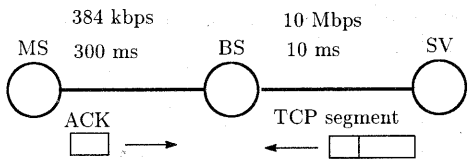

図 2: シミュレーションモデル

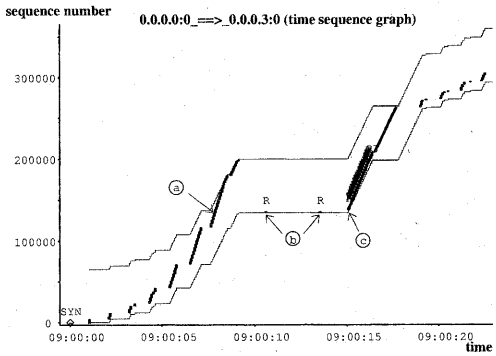

図 3: スプリアス再送の例

3 関連研究

本章では，スプリアス再送検出と輻輳ウインドウ制御を行う従来提案されたアルゴリズムについて概観する．そして，これら従来法を移動通信に適用する場合の問題点について述べる．

3.1 Eifel アルゴリズム

Ludwig らの Eifel アルゴリズム [4] では，タイムスタンプ・オプションを利用し，セグメント再送後に届く ACK がタイムアウト前のオリジナル・セグメントに対するものか，再送によるものかを，タイムスタンプから一意に判別する．タイムスタンプ・オプションは RFC1323 [11] として標準化され，多くの OS でサポートされているため，サーバ側に Eifel アルゴリズムを実装することでスプリアス再送の抑制が可能である．ただし，タイムスタンプ使用には TCP オプションの 12 バイトを必要とするた

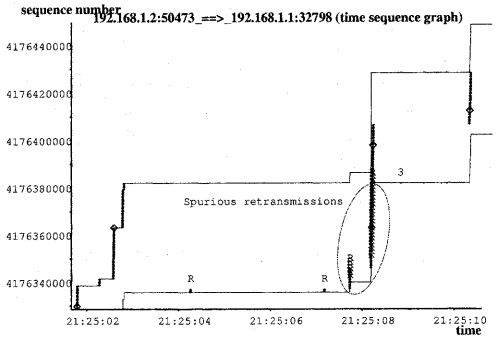

図 4: STO 未検出例（ Eifel アルゴリズム）

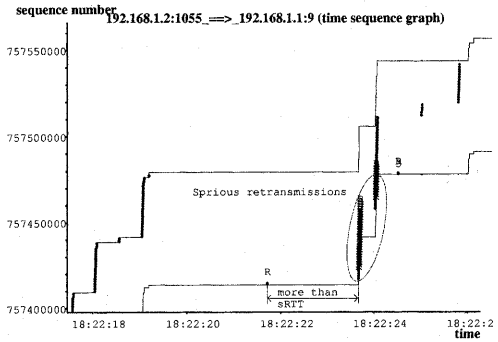

図 5: STO 未検出例（Allman のアルゴリズム）

め，$MSS = 1460$ のフルセグメント送信時には，約 1%， ACK では 30% ものオーバーヘッドとなる．従って，従量制課金をもつ移動通信網への導入には，STO やスプリアス・ファースト・リトランスミットにともなう再送発生率との間のトレードオフの関係を，予め見極めておく必要がある．

Eifel アルゴリズムは， Linux Kernel 2.4 系では標準機能として実装されている．そこで，Red Hat Linux 7.3 と文献 [7] に示された W-CDMA エミュレータを用い，図 2 と同様となるように通信路モデルを設定し，平均 BLER 5%，MAXDAT 25，`Timer_Status_Prohibit` 150 ms として動作確認を行う．この際， RLC レイヤでの再送制御が比較的長時間働くようにエラーパターンを通信途中に挿入し，遅延ジッタ増加に起因するタイムアウトを発生させている．この結果，図 4 に示すように，2 度の同一セグメント再送後に ACK が届く場合には，go-back-N 再送となり，Red Hat に実装された Eifel アルゴリズムでは STO 検出が適切に実行されていないことが分かる．ただし，上記事象は，セグメント送信時刻の保持方法に起因する原因であり，この部分の修正により，解決されるものと考えられる．従って，以下では，図 4 の場合でも STO 検出が可能であるとして議論を進める．

3.2 Allman のアルゴリズム

Allman らは，インターネット区間における STO の解析を行い，オリジナル・セグメントに対する ACK の多くはタイムアウト後のセグメント再送から $\frac{1}{2}RTT_{min}$ 時間以内に到着していることを発見した．ここで， RTT_{min} は TCP コネクション確立後に計算された最小の RTT とする．そして，この sweet point と呼ぶしきい値を用いて， STO を判定するアイディアを提案している [5]．

彼らのアルゴリズムは FreeBSD 4.3 以降で実装され， $sRTT$ (smooth RTT) をしきい値とし，タイムアウト後から，$sRTT$ 以内に ACK が到着した場合，Eifel アルゴリズム同様に輻輳ウインドウを STO 発生前の値に戻し，後続のオリジナル・セグメントから送信を継続する．しかし，図 4 と同様に W-CDMA エミュレータを用いた結果， $sRTT$ よりも ACK 到着に時間がかかる場合にはスプリアス・タイムアウトの検出が行えないことが図 5 から分かる．更に，2 回以上のセグメント再送後または， go-back-N 再送期間にも動作しないといった問題がある．

3.3 DSACK

DSACK (Duplicate Selective Acknowledgement) は TCP オプションの 1 つである SACK (Selective Acknowledgement) [12] の拡張として Floyd らにより提案され， RFC2883[6] として標準化された．

クライアントに同一シーケンス番号をもつセグメントが重複して届いた場合， SACK ブロックとしてサーバに通知することで，STO，スプリアス・ファースト・リトランスミットの判定をサーバが行うことができる．ただし，これら判定には少なくとも 1 RTT 時間が必要となる．このため， DSACK を適用したとしても，スプリアス再送抑制を行うことはできない．

図 6 に，図 4 と同様に W-CDMA エミュレータを用いた結果を示す．同図から，クライアントからの重複セグメント受信を示す SACK ブロックは，go-back-N 再送がすべて完了した後にサーバへ届くことが分かる．

4 提案する STO 検出アルゴリズム

TCP ストリームを使用した移動通信では，従来研究されてきた IP 網での STO 検出と，無線網に有効

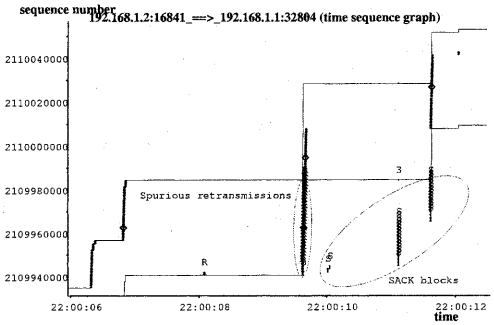

図 6: DSACK による重複セグメント検出

な STO 検出機能を兼ね備えることが必要不可欠である.

本章では, Allman により提案された ACK 監視アルゴリズムの go-back-N 再送期間への拡張と, 再送セグメントのシーケンス番号を超える 2 セグメント ACK 監視による STO 検出機能および, 輻輳ウインドウ制御機能を具備するアルゴリズムを提案する. 提案手法と従来法は, いずれも STO 検出とスループット低下抑制のための輻輳ウインドウ制御機能を有し, 表 1 に示す分類となる. 以下では, 提案手法で採用する 2 つの STO 検出アルゴリズムについて説明を行う.

4.1 ACK 到着時刻監視による STO 検出

第 3 世代移動通信方式におけるパケット通信方式では一時的な移動機の圏外や, 無線区間での遅延ジッタの増加時にも, 基地局においてセグメントが十分な時間保持されるため, インターネット上のボトルネック・ルータのような輻輳やパケットロストの発生はきわめて低い. ただし, 無線状態の回復時には, 図 3 に示すように, 複数のセグメントがクライアントに連続的に届き, go-back-N 再送を引き起こすといった問題がある.

そこで, 提案するアルゴリズムでは, タイムアウト発生直後は IP 網での輻輳と考え, Allman のアルゴリズム同様に RTT 監視を, それ以降は無線網による影響と考え, go-back-N 再送期間に連続的に到着する ACK の RTT 監視をそれぞれ行う. 具体的には, ① 最初の再送セグメントに対する ACK 受信時は Allman のアルゴリズムをもとにした FreeBSD と同様に $sRTT$ をしきい値とし, ② go-back-N 再送期間中は複数セグメント ACK を考慮して

$$\texttt{t\_now(ackno)} - \texttt{t\_rexmt(highest\_ack)} \leq \alpha \times sRTT \quad (1)$$

をしきい値とする RTT 監視を行い, STO 検出をそれぞれ行う. ここで, `t_now(ackno)` はシーケンス番号 `ackno` をもつ ACK の到着時刻, `t_rexmt(highest_ack)` は直前に受信した ACK と同じシーケンス番号 `highest_ack` をもつ再送セグメント送信時刻, α は送信セグメントに対する ACK セグメント数, セグメントサイズなどを考慮した定数とする. また, 複数セグメント ACK とは, Delayed ACK における 2 セグメント ACK や, ACK ロストによる 2 セグメント以上のシーケンス番号をもつ ACK を意味する. たとえば, サーバによる 2 セグメント分のデータ送信後, Delayed ACK による 2 セグメント ACK を受信した場合, `ackno` と `highest_ack` の間には, $\texttt{ackno} - \texttt{highest\_ack} = 2 \times \texttt{segment\_size}$ (たとえば, 2920 バイト) の関係が成り立つ.

4.2 2 セグメント ACK 監視による STO 検出

無線区間の遅延ジッタ増加時や移動端末の一時的な圏外時に送信された TCP セグメントは, 基地局で一時的に保持される. そして, 無線状態の回復時には, 複数のセグメントが連続的に到着するため, クライアントは 2 セグメント ACK [*1] をサーバに送信することが予測される. 一方, タイムアウト判定を行ったサーバでは, 輻輳ウインドウサイズを 1 とし, スロー・スタートによる セグメントの再送が行われる. このため, STO 発生時には, 1 セグメントの再送後, 前述のようにオリジナル・セグメントに対する 2 セグメント ACK がクライアントから送信される可能性が高い.

そこで, 提案するアルゴリズムではタイムアウト後の再送セグメントのシーケンス番号を超える ACK 到着の監視を行い, この ACK がオリジナル・セグメントに対する 2 セグメント ACK か否かの判定を通して STO 検出を行う. 更に, go-back-N 再送期間中には, ACK ロストに起因する 2 セグメント以上のシーケンス番号をもつ ACK を対象とした監視を同様に実施する.

これにより, 従来, Allman のアルゴリズムでは検出不能であった複数回の同一セグメントの再送後でも, STO 検出を行うことができる. また, ACK 到着時刻だけでは判断が難しいあいまいさを, 本方式によって排除できるため, 常に正確な STO 検出

[*1] 一般的な TCP の実装では 200 ms のタイマをもち, piggyback によって ACK へ便乗するデータの有無を確認するための Delayed ACK 機能をもつ [8]. ただし, フルサイズのセグメントを 2 個受信した場合には ACK を即座に送信するよう規定されている [13].

表 1: 提案アルゴリズムと従来法の比較

Algorithms	Eifel	Allman	DSACK	Proposed
Detection	Timestamp	Rapid ACK*	SACK block	Rapid ACK, ACK cover 2 segments
Recovery	Congestion window controls (Restore original *cwnd* and *ssthresh*)			

* 1st retransmist segment only

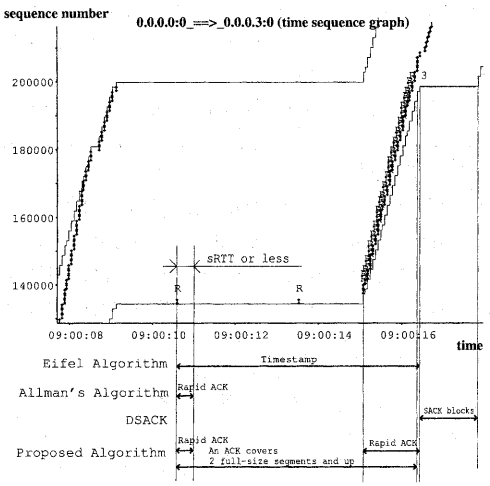

図 7: 従来法と提案方式の検出タイミングの比較例

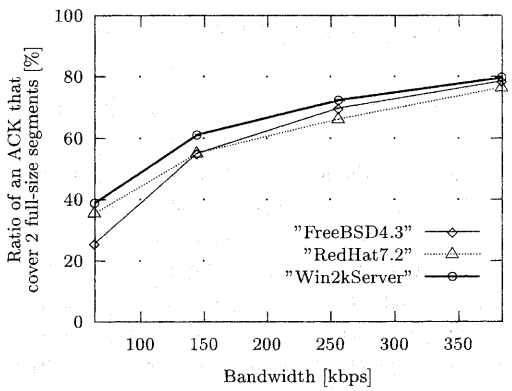

図 8: データ伝送帯域と 2 セグメント ACK 発生確率

が可能となる.

4.3 提案手法による適用範囲と 2 セグメント ACK 発生頻度

図 3 に示した STO とスプリアス再送の例をもとにして, 各アルゴリズムの検出方法の違いを図 7 に示す. 同図より, Allman のアルゴリズムと DSACK は, セグメント再送後の $sRTT$ 以内と go-back-N 再送後と適用範囲が限定される. これに対して, Eifel アルゴリズムはタイムアウト発生から go-back-N 再送終了までと検出期間も長く, STO 検出には最も優れていることが分かる. 一方, 提案手法による STO 検出は, Allman の提案した ACK 監視の go-back-N 再送期間への拡張に加え, 再送セグメントを超えるシーケンス番号をもつ ACK 監視を組み合わせることにより, Eifel アルゴリズム同様にタイムアウト発生から go-back-N 再送終了までの期間に対して適用可能であることが分かる.

次に, 提案手法において, STO 検出契機の一つである 2 セグメント ACK の発生頻度とデータ伝送帯域の関係を明らかにする. Delayed ACK のタイマの実装は OS により異なる [14]. そこで, FreeBSD 4.3, RedHat 7.2, Windows2000 Server の 3 種類を対象として, 各 OS の 2 セグメント ACK 発生確率を 10 回の 1 MB のバルク転送の平均値から判断する. 想定する移動通信網は, 図 2 と同様とし, タイムアウト発生のエラーパターンを排除した. 図 8 から, データ伝送速度が 64 kbps 付近では OS による若干の違いが見られるものの, データ伝送帯域の拡大時には, セグメント当たりの転送時間が短縮され, いずれの OS にいおいても 2 セグメント ACK の発生頻度は高くなる. 更に, データ伝送帯域と 2 セグメント ACK の発生傾向に OS による大きな違いは無いことが確認できる.

以上から, より高速なデータ伝送を実現する移動通信網に提案手法を適用する場合, 再送セグメントを超えるシーケンス番号をもつ ACK 監視による検出確率が向上するといった特徴をもつことが分かる.

4.4 TCP セグメントロスト対策

図 1 の IP 網としてインターネットを利用した場合, データ・セグメントと ACK がロストする確率

は高くなる．検出期間の拡大に伴い，輻輳ウインドウ制御機能への TCP セグメントロストの影響が懸念されるため，より正確な STO 検出が求められる．

このため提案方式では，TCP セグメントのロストが考えられる Duplicate ACK または，Full ACK[*2] 受信時には，通常のスロー・スタートと輻輳回避アルゴリズムを継続するものとする．また，再送セグメントを越える Partial ACK [*3] 受信時には，4.2 節の手法に従い，輻輳ウインドウを STO 発生前の値に戻す．その後，サーバで受信されることが予想される Duplicate ACK に関しては，通常の TCP 同様にファースト・リカバリ，輻輳回避アルゴリズムを適用することとなる．本動作は，Allman, Eifel アルゴリズムによる STO 検出，輻輳ウインドウ制御がなされた場合と同様である．

図 9 は，TCP セグメントロストにより引き起こされる Full ACK 受信により，提案方式を用いた STO 検出を中止する例を示す．同図の左列から行番号，データ・セグメントのシーケンス番号 (seqno)，TCP セグメント送信方向，ACK のシーケンス番号 (ackno) を表す．また，dropped, received はセグメントのロスト，受信をそれぞれ表す．1, 3, 5 行目の TCP セグメントロストによるタイムアウトの後，サーバによるセグメント再送が 7 行目で実施されている．この結果，クライアントでは送信済みの全てのセグメントに対する Full ACK をサーバ側に返す．一方，サーバではセグメントロストを Duplicate ACK 等により知ることはできないものの，輻輳ウインドウの制御誤りを防ぐため，提案方式による STO 検出を中止し，通常のスロー・スタート，輻輳回避アルゴリズムを継続する．

5 シミュレータによる提案手法評価

本章では，提案方式によるスプリアス検出と輻輳ウインドウ制御機能を，*ns*2 バージョン 2.1b9 [10] へ実装し，シミュレーションにより検証する．

5.1 ACK 到着時間監視による STO 検出確認

はじめに，提案したアルゴリズムを図 3 と同様に 2 度の同一セグメント再送が発生する通信路モデルに適用し，go-back-N 再送期間での $\alpha \times sRTT$ 時間以内に到着する ACK 監視により，STO 検出が

```
  Server(seqno)              Client(ackno)
=========================================
 1: 500-999     ----------->  (dropped)
 2: 1000-1499   ----------->  (received)
 3: (dropped)   <-----------  500
 4: 1500-1999   ----------->  (received)
 5: (dropped)   <-----------  500
 6:               [Timeout]
 7: 500-999     ----------->  (received)
 8: (received)  <-----------  2000
```

図 9: TCP セグメントロストにより引き起こされる Full ACK 例

適切に行われることを明らかにする．以下では，通信路の帯域幅遅延積から決定したウインドウサイズ (window_size = 64 kB) によって，

$$\alpha = 1 + (\text{ackno} - \text{highest\_ack})/\text{window\_size}$$

としてシミュレーションを行うものとする．

図 10 において，9:00:8.9 および，9:00:12.4 のセグメント再送後，9:00:14.1 に 1 セグメント分の ACK がサーバに到着している．この段階では，オリジナルまたは，再送セグメントかの判定は行えないため，通常のスロー・スタートの動作に従って 輻輳ウインドウサイズ を 1 増やし，後続の 2 セグメントの再送が行われる．しかし，そのわずか 100 ($< sRTT$) ms 後に ACK が届いたことで，サーバは STO と判定し，セグメント再送を中止している．そして，輻輳ウインドウをタイムアウト直前の値に戻し，後続のオリジナル・セグメントを送信している．従って，本方式は go-back-N 再送期間においても，ACK 監視による STO 検出を行い，輻輳ウインドウ制御が適切になされていることが確認できる．

5.2 2 セグメント ACK 監視による STO 検出確認

次に，図 11 では，再送セグメントのシーケンス番号を超える ACK 監視による STO 検出および，輻輳ウインドウ制御が適切に行われることを明らかにする．

図 11 において，9:00:8.9 および，9:00:12.4 のセグメント再送後，9:00:14.6 に 2 セグメント ACK が到着している．この 2 セグメント ACK は，再送セグメントのシーケンス番号を超える ACK であるためサーバでは STO と判定している．そして，輻輳ウインドウをタイムアウト直前の値に戻し，後続のオリジナル・セグメントを送信している．従って，本方式は，複数回の同一セグメント再送後であって

[*2] 送信済みの複数のオリジナル・セグメントすべてに対するシーケンス番号をもつ ACK

[*3] 送信済みのオリジナル・セグメントの一部を含むシーケンス番号をもつ ACK

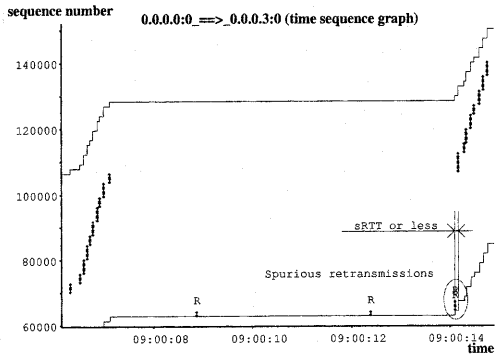

図 10: go-back-N 再送中の ACK 到着時刻監視による STO 検出例

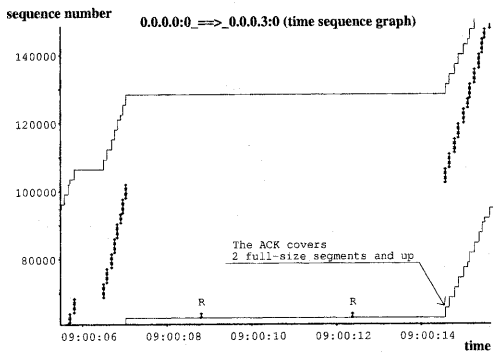

図 11: 2 セグメント ACK 監視による STO 検出例

も，オリジナル・セグメントに対する 2 セグメント ACK を契機として STO 検出を行い，輻輳ウインドウ制御が適切になされていることが分かる．

6 まとめ

本論文で提案したアルゴリズムは，従来法を移動通信網に適用する際の STO 検出期間の問題や，TCP オプションによるオーバーヘッド問題を解決し，タイムアウト発生から go-back-N 再送までの期間を対象とする STO 検出および，輻輳ウインドウ制御が適切に行われることを，*ns*2 シミュレータへの実装・評価を通して明らかにした．

今後は，様々なセグメントロストを含む環境下での評価を通して，提案方式の有効性について確認を進めたい．更に，SACK オプション，DSACK 利用による STO 検出精度の改善も期待されるため，導入評価を行うことも今後の課題の一つと考える．

謝辞

本研究を進めるにあたりご協力頂いた大菅大吉氏に感謝いたします。

参考文献

[1] 3GPP. 3G TS 25.322 v.3.5.0. RLC Protocol Specification, 2000.

[2] V. Paxson and M. Allman. Computing TCP's Retransmission Timer. RFC2988, Nov. 2000.

[3] 稲村 浩，石川太郎，高橋 修. W-CDMA 網でのリンク層 ARQ と TCP の性能評価. 情処学論, Vol. 43, No. 12, pp. 3859–3868, Dec. 2002.

[4] R. Ludwig and K. Sklower. The Eifel Retransmission Timer. *SIGCOMM Computer Communication Review*, Vol. 30, No. 3, pp. 17–27, July 2000.

[5] M. Allman and A. Falk. On the Effective Evaluation of TCP. *SIGCOMM Computer Communication Review*, Vol. 29, No. 5, Oct. 1999.

[6] S. Floyd, J. Mahdavi, M. Mathis, and M. Podolsky. An Extension to the Selective Acknowledgement (SACK) Option for TCP. RFC2883, July 2000.

[7] 稲村 浩，石川太郎，高橋 修. W-CDMA 網での TCP トラフィック特性. 情処学 MBL 研報, Vol. MBL18-33, pp. 245–251, Sep. 2001.

[8] W. Richard Stevens 著（橘 康雄 訳、井上 尚司 監訳）. 詳解 TCP/IP Vol.1 プロトコル. ピアソン・エデュケーション, 2000.

[9] H. Inamura et al. TCP over Second (2.5G) and Third (3G) Generation Wireless Networks. draft-ietf-pilc-2.5g3g-12.txt, Dec. 10 2002.

[10] The Network Simulator - ns-2. http://www.isi.edu/nsnam/ns/.

[11] V. Jacobson, R. Braden, and D. Borman. TCP Extensions for High Performance. RFC1323, May 1992.

[12] M. Mathis, J. Mahdavi, S. Floyd, and R. Romanow. TCP Selective Acknowledgment Options. RFC2018, Oct. 1996.

[13] M. Allman, V. Paxson, and W. Stevens. TCP Congestion Control. RFC2581, April. 1999.

[14] A.M. Costello and G. Varghese. Redesigning the BSD Timer Facilities. *Software–Practice and Experience*, Vol. 28, No. 8, pp. 883–896, July 1998.