情報処理学会論文誌 Vol. 50 No. 1 121–132 (Jan. 2009)

複数経路環境における非対称なTCP通信での不要な再送抑制手法

荻野秀岳[†1,*1] 石原進[†2]

筆者らは，Mobile IP 環境において，複数の移動端末が協調し，各端末が持つ長距離低速リンクを同時に利用することで，高速な通信を実現する Mobile IP SHAKE（SHAring multipath procedure for a cluster networK Environment）を提案している．Mobile IP SHAKE における TCP 通信では，各端末のリンク品質の違いのためにセグメントの順序が逆転して受信端末に到着するために，重複 ACK が発生し，その結果不要な再送が頻発する．本論文ではその解決手法として HA 不要 ACK 破棄法および RAH（RTT between AL and HA）経路選択法を提案する．HA 不要 ACK 破棄法は，Mobile IP SHAKE においてトラフィックの分配・集約を行う Home Agent（HA）が，セグメントの順序逆転を検知し，受信端末から送信される不要な重複 ACK を破棄する手法である．RAH 経路選択法は，送信元の移動端末（Alliance Leader: AL）と HA 間の各経路の RTT を測定し，HA にセグメントが最も早く到着すると予想される経路を，AL の TCP からの送信経路に選択する手法である．シミュレーションの結果，両方式によってパケットの順序逆転の抑制および不要な再送の抑制が行えることが確認できた．

A Scheme for Avoiding Unnecessary Retransmissions of Asymmetric TCP Flows on Multipath Environment

Hidetake Ogino[†1,*1] and Susumu Ishihara[†2]

We have developed a system that increases the communication speed of mobile hosts, called Mobile IP SHAKE (SHAring multipath procedure for a cluster networK Environment). In Mobile IP SHAKE, a mobile node uses both its own link and the links of other hosts in the local network simultaneously and disperses the traffic to multiple links between the local network and the external host. Because of the different delay of links between the external nodes and the nodes in the local network, arrival of the packets at the receiver tends to be out of order. This cause unnecessary duplicate TCP ACKs and unnecessary retransmissions. We propose two schemes to solve this problem. *HA unnecessary ACK dropping method* allows a home agent which manages multiple paths between the internet and mobile hosts drop unnecessary duplicate ACKs by monitoring forwarded segments. *RAH (RTT between AL and HA) path selection method* allows TCP of a mobile host (Alliance Leader: AL) select a route according to the measured RTT of multiple paths between the AL and the HA. Simulation results show that the schemes are useful to reduce unnecessary retransmissions and out of order arrival of packets.

1. はじめに

筆者らは，無線通信環境において高速な通信を実現する手法として通信回線共有方式（SHAKE：SHAring multipath procedure for a cluster networK Environment）を提案している．SHAKE では，複数のインタフェースを搭載した移動端末が，近接する移動端末と短距離無線リンクを用いて一時的にローカルネットワークを構築し，同盟関係（Alliance）を結ぶ．Alliance 内の各移動端末の長距離低速リンクを同時に利用し，各リンクにトラフィックを分散させ，ネットワーク資源を有効に活用することで高速な通信を実現する．SHAKE の実現方法として，端末の移動透過性を保障する Mobile IP を応用することで，移動透過性を保障しながら SHAKE を用いた通信を実現する Mobile IP SHAKE が提案・実装されている[1),2)]．

Mobile IP SHAKE 環境での TCP 通信では，Alliance 内の移動端末が持つ性質が必ずしも同じでない長距離無線リンクを同時に利用する可能性がある．そのため，各経路の状態によってセグメントの順序が逆転して受信端末に到着してしまい，重複 ACK の発生にともなう無駄な再送が頻発する．この結果，TCP の輻輳ウィンドウが増加しにくくなり，高いスループットを得ることができない．本論文では，Mobile IP SHAKE の Alliance から Alliance 外部への上り通信において，セグメントの順序逆転によって発生する不要な重複 ACK にともなう再送の抑制方法を提案する．

複数経路を用いた TCP 通信に関しては，多くの試みがなされている．たとえば，Rojviboonchai らは，各経路のボトルネック部の帯域幅を一定時間内の確認応答から算出

†1 静岡大学大学院工学研究科
Graduate School of Engineering, Shizuoka University
†2 静岡大学創造科学技術大学院
Graduate School of Science and Technology, Shizuoka University
*1 現在，ヤマハ株式会社
Presently with Yamaha Corporation

し，各経路の輻輳ウィンドウがボトルネック部の帯域幅より大きくなった時点で，輻輳回避段階へと移行する輻輳制御手法 R-M/TCP を提案している[3]．また，板谷らは，文献 3) を FreeBSD へ実装し，通信端末間の経路が 2 本の場合に帯域幅を変化させて性能評価を行っている[4]．Lee らは，送信端末が利用する複数経路の数に応じて，再送処理へ移行するまでの重複 ACK 数を決定し，受信端末では Delayed ACK において再送パケットに対してのみ即座に ACK を返信する手法を提案している[5]．Zhang らは，各経路の受信端末へ配送途中のデータ量と輻輳ウィンドウから次に送信する経路を決定し，タイムアウトが発生した場合には，調査パケットを用いることで再送パケットを別の経路から送信する MTCP を提案している[6]．Kim らは，移動端末のグループが各長距離無線リンクを用いて 1 つのデータを集約するような状況を想定し，送信端末にプロキシを設けることで，利用する各経路の情報管理や ACK の再配列や TCP ヘッダの拡張，再送処理を行うとともに輻輳制御を行う手法を提案・実装評価している[7]．殿内らは，複数経路対応の TCP として，経路ごとのシーケンス番号管理と輻輳ウィンドウ管理を行い，再送時には輻輳発生経路以外で輻輳ウィンドウが最大の経路を送信経路とする手法を提案している[9]．

Gerla らは，無線環境に適した TCP 実装としてパケットの送信間隔および ACK の受信間隔によって，利用する経路の帯域幅を測定することで ssthresh および再送タイムアウト時間を決定する TCP Westwood を提案しているが，それを TCP Westwood を複数経路に対応させる試みも行っている[8]．

Iyengar らは，SCTP において，確認応答が未受信の送信済データ量と Delayed ACK を用いて不要な高速再送を抑制し，重複 ACK においても輻輳ウィンドウを増加させる手法を提案している[10]．また，Casetti らは，複数経路環境での SCTP 通信において，TCP Westwood で用いられるボトルネックリンクの帯域幅予測を用いた送信経路選択手法を提案している[11]．

以上の方式は，両エンドの TCP が複数経路に対応する必要がある．一方，本論文で提案する方式は，Mobile IP SHAKE 環境での TCP 通信の効率化を対象とし，移動端末および中継端末（Mobile IP の Home Agent（HA））以外では複数経路を用いた通信に対応した拡張がされていることを前提としないという特徴がある．つまり，移動端末側の TCP は複数経路に対応した設定である一方で，通信相手の TCP は複数経路に対応しない通常の TCP である非対称な TCP 通信を前提とする．この環境での通信の効率化のために HA への比較的軽量な機能追加によって対処する．

以下，2 章で Mobile IP SHAKE の概要について述べ，TCP 通信における順序逆転にともなう不要な再送の問題について明らかにする．次に 3 章で，本論文で提案する手法の設計方針について述べた後，4 章，5 章で Mobile IP SHAKE の移動端末側からの通信を対象とした不要な再送の抑制方法，HA 不要 ACK 破棄法および RAH（RTT between AL and HA）経路選択法をそれぞれ提案する．6 章でシミュレーションによる両手法の評価について述べ，7 章で本論文をまとめる．

2. Mobile IP SHAKE

2.1 Mobile IP SHAKE の概要

Mobile IP SHAKE は，Mobile IP[12),13)] において移動端末の移動情報を管理する Home Agent（HA）および移動端末の IP 層にトラフィックの分配機構を追加することで，複数の経路を同時に利用した通信を実現する．Alliance 内の移動端末が持つ長距離無線リンクを利用して実際に通信を行う移動端末である Alliance Leader（AL）は，AL に通信帯域を提供する他の移動端末である Alliance Member（AM）の Care of Address（CoA）を HA に通知する．HA は，AL の Home Address（HoA）と CoA の対応および AL の HoA と AM の CoA の対応を保持する．HA は，この対応に基づいて通信相手（CN: Correspondent Node）が AL に送信するパケットを HA と AL 間および HA と AM 間を結ぶ複数の経路へ分配する．また，AL が CN へパケットを送信する場合，AL は自身の外部リンクおよび AM の外部リンクを用いた経路へパケットを分配する．

2.2 Mobile IP SHAKE での TCP 通信の問題

Mobile IP SHAKE で用いる複数の経路，すなわち AL および各 AM の外部リンクの通信帯域や遅延は一般にそれぞれ異なり，さらにそれぞれ変動する．したがって，パケットの送信経路の選択に適切な手法を用いたとしても，パケットの順序が逆転して CN に到着してしまうことが避けられない．TCP 通信では，累積確認応答を行うため，セグメントの順序が逆転して CN に到着してしまうと，重複 ACK が AL へ送信される．TCP Reno 等多くの実装では，重複 ACK を連続して 3 つ受信した AL は，ロスしていないセグメントを無駄に再送し，かつ輻輳ウィンドウを縮小する．このため，スループットが低下してしまう．**図 1** の例では，AL から遅延の異なる複数の経路にセグメントが振り分けられて送信された結果，2 番目のセグメントよりも先に，3，5，7 番目のセグメントが到着するために重複 ACK が 3 つ発生している．この結果，2 番目のセグメントが実際にはロスしていないにもかかわらず，そのセグメントの再送が行われてしまう．

通常の TCP では，無駄な再送の原因である不要な重複 ACK の発生を避けることができ

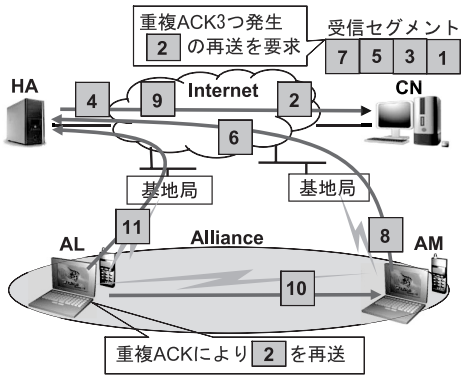

図 1　Mobile IP SHAKE における TCP 通信の問題
Fig. 1　Problem of TCP on Mobile IP SHAKE.

ない．ただし，CN の TCP が複数経路を利用した通信に対応したものであれば，ACK の返信経路を 1 本に限定することで，使用する各経路の片側 RTT を測定し，送信経路の選択を行ったり[5),6)]，Delayed ACK や SACK[14)] の使用によってこのような問題を防止したりすることがある程度可能である．しかし，Mobile IP SHAKE では，AL の通信相手として通常の TCP を備えた任意の端末を想定するため，CN の TCP による不要な重複 ACK の送信を抑制できない．AL においては，複数経路に対応した TCP を備えることが可能であるが，そのような TCP であっても，CN が送信する重複 ACK が不要であるかを AL 単体では判別することができない．

3. 設 計 方 針

Mobile IP SHAKE は，HA と AL の IP 層に特殊な機能を追加することで実現される．そこで，HA と AL において，不要な再送を抑制する機能の追加を考える．

Mobile IP SHAKE においてセグメントの順序逆転の多くは AL-HA 間のリンクで発生するため，下り通信では，HA および AL はセグメントの順序逆転を検知することができない．一方，上り通信においては，HA は CN に到着する順番とほぼ同じ順番でセグメントを受信することが期待できるため，HA で多くのセグメントの順序逆転を検知することができる．すでに IP 層に特殊な機能を追加している AL では，TCP 層に複数経路対応の機能を追加することは許容可能と考える．一方，CN は通常，任意のサーバであり，その TCP に特殊機能の追加を行うことは現実的ではない．Mobile IP SHAKE に対応している HA において TCP のために機能追加を行うことは許容できると考えられるが，HA は複数の移動端末の移動情報を集中して管理するため，TCP 性能向上のために追加される機能は軽量である必要がある．1 つの方法として，HA にスプリット型の TCP を導入し，さらに複数経路対応を行う方法が考えられるが，各フローの状態管理にともなう負荷が大きくなると予想される．

そこで本論文では，AL から CN への上り通信に対して，HA と AL で動作し，かつ HA 上での処理が軽量である不要な再送を抑制する手法として，i) HA 不要 ACK 破棄法，ii) RAH 経路選択法を提案する．なお，前提として，AL の TCP は，Mobile IP SHAKE で提供される複数の経路を識別でき，各経路に明示的にセグメントを振り分け可能であるものとする．一方，CN の TCP は通常のものであり，SHAKE に関連した機能追加は行わないものとする．

4. HA 不要 ACK 破棄法

HA 不要 ACK 破棄法では，HA は AL から CN へ中継するセグメントの TCP ヘッダ情報を用いて，セグメントの順序逆転またはロスの発生を検知し，不要な重複 ACK を破棄することで，AL での不要な再送を抑制する．HA は，重複 ACK を受信した時点で，その重複 ACK がロスによるものでなく，セグメントの順序逆転によって発生したと判別すると，その重複 ACK は不要であると見なして破棄する．

HA は，複数経路を用いた通信におけるセグメントの到着順序逆転で発生する不要な重複 ACK を検知するため，以下の 4 つの変数を利用する．

(1)　$M_{\mathrm{AL}}(i)$：経路 i から以前に受信した最大シーケンス番号
(2)　$M_{\mathrm{HA}}(i)$：経路 i から受信した最大シーケンス番号
(3)　R：AL から欠落なく受信したデータの最大シーケンス番号
(4)　A：CN から受信した最大の ACK 番号

さらに，HA はセグメントの到着順序逆転で生ずるシーケンス番号空間上の穴を管理するための HoleList を持つ．Holelist の詳細は後述する．

AL は，各経路で送信した最大シーケンス番号である $M_{\mathrm{AL}}(i)$ を保持し，セグメント送信時に，$M_{\mathrm{AL}}(i)$ と経路の識別子を送信セグメントの TCP ヘッダに付加する．HA は，$M_{\mathrm{AL}}(i)$，$M_{\mathrm{HA}}(i)$，R を用いてセグメントの順序逆転またはロスがどの経路で発生しているかを検知する．

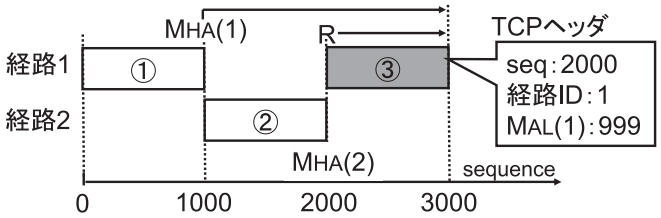

図 2　HA が正しい順序でセグメントを受信した場合の処理
Fig. 2　On receiving a segment in the correct order.

なお，TCP ヘッダへの情報追加は TCP をオプションを使って行うことになる．オプションの内訳は，オプション種別，オプション長，経路識別子は各 1 バイト，$M_{\mathrm{AL}}(i)$ は 4 バイトとなるので，オプション長は 7 バイトである．実際には，パディングを含み 8 バイトのヘッダ長増加となる．

以下では，経路の数を 2 までに限定して議論する．

4.1　データセグメント受信時の処理

HA は，AL から経路 i を介してセグメント（シーケンス番号：seq，セグメント長：L）を受信すると，seq と R を比較し，そのセグメントを正しい順序で受信できたか判別する．

4.1.1　$seq = R + 1$ の場合

HA は，正しい順序でデータセグメントを受信している．$M_{\mathrm{HA}}(i)$ および R に $seq + L - 1$ を代入し，そのセグメントを CN へと転送する．

このときの HA の処理を**図 2** に沿って説明する．簡単のため，以下，セグメント長はすべて 1,000 とする．この図は，先頭シーケンス番号が 2,000 の seg#3 を HA が経路 1 から受信した状況を示している．seg#3 を受信する前には，先頭シーケンス番号が 0 の seg#1 を経路 1 から，先頭シーケンス番号が 1,000 の seg#2 を経路 2 から受信している．HA は，経路 1 から seg#1 を受信した時点で $M_{\mathrm{HA}}(1)$ と R（ともに初期値 0）を seg#1 の最大シーケンス番号である 999 まで更新し，経路 2 から seg#2 を受信した時点で $M_{\mathrm{HA}}(2)$ と R を seg#2 の最大シーケンス番号である 1,999 まで更新している．そして seg#3 を受信した時点では，seg#3 の最大シーケンス番号 2,999 まで AL から欠落なくデータを受信しているため，$M_{\mathrm{HA}}(1)$ と R を 2,999 まで更新する．

4.1.2　$seq < R + 1$ の場合

HA が受信したセグメントは，再送セグメントである．HA が CN へと転送したセグメントが HA-CN 間の経路でロスした場合には，$R+1$ より小さなシーケンス番号のセグメント

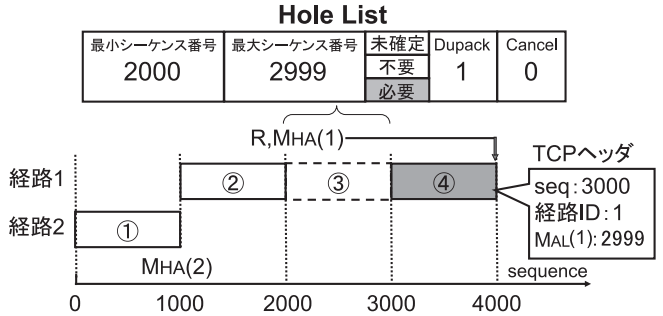

図 3　未受信シーケンス番号空間（穴）を検知した場合（$seq > R + 1$，$M_{\mathrm{HA}}(i) \neq M_{\mathrm{AL}}(i)$）
Fig. 3　Unreceived sequence number space (a hole) is found ($seq > R + 1$, $M_{\mathrm{HA}}(i) \neq M_{\mathrm{AL}}(i)$).

が HA に到着する．HA は受信した再送セグメントを CN へと転送し，R および M_{HA} は更新しない．

4.1.3　$seq > R + 1$ の場合

HA は，セグメントの欠落もしくは順序逆転が発生していることを検知する．**図 3** の例では，HA は seg#2 を経路 1 から受信したあとに，続けて経路 1 から seg#4 を受信している．このとき HA は，自身の持つ R（= 1,999）と受信した seg#4 の seq（= 3,000）とを比較し，$seq \neq R + 1$ であることから，シーケンス番号 2,000〜2,999 の空間が未受信であることを検知する．

HA は未受信のシーケンス番号空間（穴）を管理する HoleList を持つ．HA は，1 つの連続した未受信シーケンス番号空間に対して，1 つの HoleList 要素を作成する．セグメント到着によって，新たに穴ができる場合，あるいは分割される場合には，新しく HoleList に要素が加えられる．

HoleList 要素は，以下の 5 項目から構成される．

- 未受信シーケンス番号空間の最小シーケンス番号
- 未受信シーケンス番号空間の最大シーケンス番号
- ACK 受信時の処理を決定するモード（$Mode$）
  未確定，不要，必要の 3 モードがある．未受信シーケンス番号空間がロスまたは遅延により遅れているのか判断できない状況では未確定モード，未受信シーケンス番号空間を埋めるセグメントを受信した状況では不要モード，未受信シーケンス番号空間に対するセグメントがロスした状況では必要モードとなる．

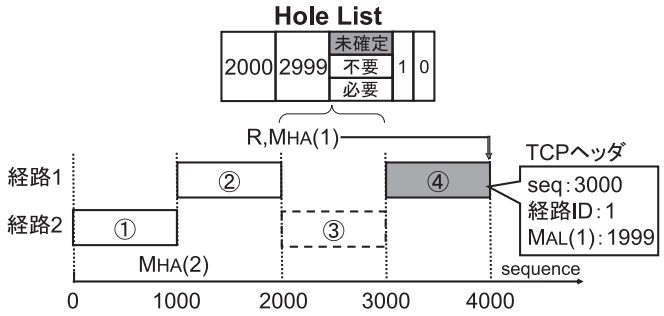

図 4　$M_{\mathrm{HA}}(i) = M_{\mathrm{AL}}(i)$ の場合
Fig. 4　$M_{\mathrm{HA}}(i) = M_{\mathrm{AL}}(i)$.

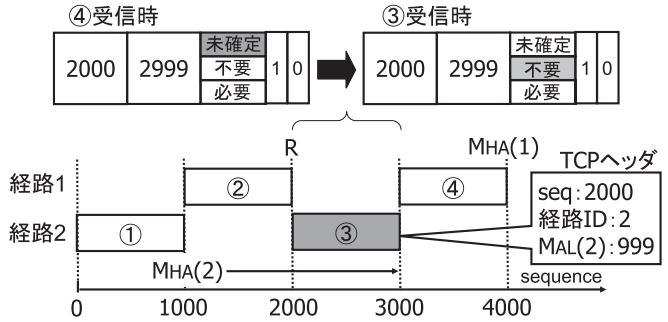

図 5　HoleList に対応するセグメントを受信した場合
Fig. 5　On receiving a segment corresponding to a HoleList item.

- 破棄可能であると推定される重複 ACK 数（*Dupack*）
- 破棄された重複 ACK 数（*Cancel*）

4.2　順序逆転とロスの判定

$seq > R + 1$ の場合，HA は，受信したセグメントの TCP ヘッダに含まれる $M_{\mathrm{AL}}(i)$ と $M_{\mathrm{HA}}(i)$ を比較することで，セグメントの順序逆転またはロスがどの経路で発生しているかを判別し，対応する HoleList 要素の更新を行う．

4.2.1　$M_{\mathrm{HA}}(i) \neq M_{\mathrm{AL}}(i)$ の場合（図 3）

経路 i でセグメントロスが発生したと推定する．図 3 の例では，経路 1 から seg#4 を受信した HA は，TCP ヘッダ内の $M_{\mathrm{AL}}(1)$（= 2,999）と $M_{\mathrm{HA}}(1)$（= 1,999）を比較し，$M_{\mathrm{HA}}(1) \neq M_{\mathrm{AL}}(1)$ であることから，経路 1 で seg#3 よりも前に送られた別のセグメントがあることを知る．同一経路上での順序逆転はないと見なすと，先に送られたセグメントはロスしていることになる．

HA はロスしたことが分かったシーケンス番号を含む HoleList 要素の *Mode* を必要モードにし，$M_{\mathrm{HA}}(1)$ に $seq + L - 1$ を代入後，到着したセグメントを CN へ転送する．

4.2.2　$M_{\mathrm{HA}}(i) = M_{\mathrm{AL}}(i)$ の場合（図 4）

経路 i 以外の経路でロスが発生した，もしくは遅延によりセグメントが遅れている状況である．図 4 の例では，seg#4 を受信した HA は，TCP ヘッダ内の $M_{\mathrm{AL}}(1)$ を参照し，$M_{\mathrm{AL}}(1) = M_{\mathrm{HA}}(1)$ であることから，i) seg#4 より前に経路 1 から送信されたセグメントはすでに HA へ到着していること，ii) 新たに発生したシーケンス番号空間上の穴は経路 1 以外の経路（経路 2）で送信されたセグメントがロスしているか，HA への到着が遅れているために発生したことを知る．

この場合，HA は HoleList 要素の *Mode* を未確定モードにする．また，このセグメントか CN に到着することによって発生する重複 ACK は ACK 受信時の seg#3 の受信状況によっては破棄可能である．そこで，HoleList の中から，最小シーケンス番号が最小の要素の *Dupack* を 1 増加させる．そして $M_{\mathrm{HA}}(1)$ に $seq + L - 1$ を代入し，到着したセグメントを CN へ転送する．

4.3　*Mode* の変更

HA は，HoleList 要素に対応するセグメントを受信した場合，あるいはロスしたことが分かった場合に，HoleList 要素の *Mode* を変更する．

4.3.1　HoleList 要素に対応するセグメントを受信した場合

新規に受信したセグメントにより，HoleList 要素に対応するシーケンス番号空間上の穴全体が満たされる場合，HoleList 要素先頭のシーケンス番号に対応する重複 ACK はすべて不要となる．なぜなら，このセグメントが CN に到着することによって，穴の末尾よりも大きなシーケンス番号まで CN で受信されたことになるからである．したがって，このとき HA は HoleList 要素の *Mode* を不要モードにセットする．**図 5** の例では，HA は，経路 1 から seg#4 を受信後，経路 2 から seg#3 を受信している．HA が seg#3 に対応する HoleList 要素の *Mode* を不要モードにすることで，CN が seg#3 受信時に発生する重複 ACK（ACK 番号：2,000）は破棄されるようになる．

新規に受信したセグメントにより，HoleList 要素に対応する穴の先頭部分が埋まり，残りがまだ穴として残る場合，まず，4.2 節での処理に基づいて，HoleList 要素の更新をする．

その後，穴として残った部分に対応する新たな HoleList 要素を作成する．新たな HoleList 要素では，*Mode* のみを引き継ぎ，*Dupack*，*Cancel* の値は，それぞれ 0 とする．元々の HoleList 要素は，*Mode* を不要モードとして，残しておく．

新規に受信したセグメントにより，HoleList 要素に対応する穴の中央部が埋まり，先頭，末尾に分割される場合，まず，4.2 節での処理に基づいて，HoleList 要素の更新をする．さらに，穴が埋められた部分の前後に HoleList 要素を分割する．このとき分割してできる元の穴の先頭側の HoleList 要素の属性は，最小，最大シーケンス番号以外の属性を引き継ぐものとする．末尾側の HoleList 要素の属性は，*Mode* のみを引き継ぐ．*Dupack*，*Cancel* の値は，それぞれ 0 とする．

新規に受信したセグメントにより，HoleList 要素に対応する穴の末尾が残りがまだ穴として残る場合，まず，4.2 節での処理に基づいて，HoleList 要素の更新をする．さらに，埋められたシーケンス番号空間に従って，HoleList 要素の最大シーケンス番号を更新する．

4.3.2　HoleList 要素に対応するセグメントがロスした場合

HA への到着セグメントが，HoleList に対応するシーケンス番号空間上の穴を埋める場合，埋めない場合にかかわらず，セグメント到着時に 4.2 節の処理によって，HoleList 要素に対応するセグメントがロスしたかどうかが判定できる．

HoleList に対応するセグメントがロスしていることが検知された場合，HoleList に対応する重複 ACK は，セグメントのロスを検知するために必要な重複 ACK である．HA は，HoleList 要素の *Mode* を必要モードにセットする．

図 6 の例では．HA は，経路 1 から seg#4 を受信後，経路 2 から seg#5 を受信している．HA が seg#3 に対応する HoleList 要素の *Mode* を必要モードにすることで，重複 ACK（ACK 番号：2,000）を破棄することなく，AL へ転送する．

4.4　ACK 受信時の処理

CN から ACK（シーケンス番号：*ack*）を受信した HA は，HA が保持する受信した最大 ACK 番号である *A* と *ack* を比較することで，重複 ACK であるかを判別する．

4.4.1　$ack > A$ の場合

受信した ACK は重複 ACK ではないので，*A* に *ack* を代入した後，ACK を AL へ転送する．HA は，不要になった HoleList 要素，すなわち最大シーケンス番号が *ack* よりも小さい要素を削除する．

4.4.2　$ack \leq A$ の場合

受信した ACK は重複 ACK である．HA は，*ack* に対応する HoleList 要素の *Mode* によって ACK の処理を決定する．

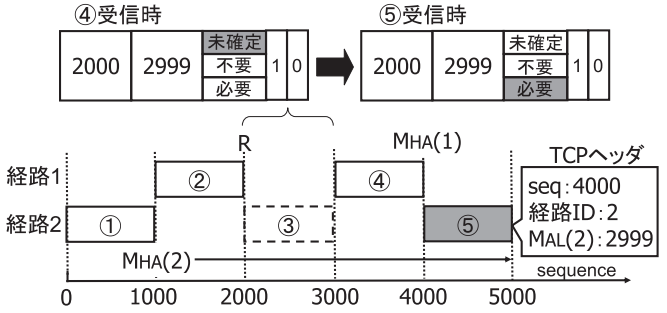

図 6　HoleList に対応するセグメントがロスした場合
Fig. 6　A segment corresponding to a HoleList item is lost.

4.4.2.1　*Mode* が未確定モードの場合

対応する HoleList 要素のシーケンス番号空間のセグメントが，経路の遅延差によって遅れて配送されているのか，セグメントのロスが発生しているのか判別できない状況である．そこで，経路の遅延差によってセグメントの到着が遅れている可能性を考え，HA は受信した ACK を破棄する．そして最小シーケンス番号が最も小さい HoleList 要素の属性値 *Cancel* を 1 増加する．*Cancel* は破棄した重複 ACK の数をカウントする変数である．なお，セグメントがロスしている場合には，重複 ACK を AL へ転送する必要があることを想定し，*Cancel* に対して，閾値 θ_c を設ける．*Cancel* の値が閾値 θ_c を超えた場合，HA はその ACK の TCP ヘッダに再送用にフラグをセットし，AL へ転送する．AL はこのフラグがセットされた ACK を受け取ると，高速再送処理を行う．

4.4.2.2　*Mode* が不要モードの場合

対応する HoleList 要素のシーケンス番号空間のセグメントを HA がすでに受信した状況である．重複 ACK によるセグメントの再送は望ましくないため，その ACK を破棄するとともに，HoleList 要素が保持する破棄可能な重複 ACK 数 *Dupack* を 1 減少させる．*Dupack* が 0 の場合，重複 ACK を破棄することなく AL へ転送する．HA では，セグメントを受信した時点で HoleList 要素に対する ACK 数を *Dupack* を用いてカウントすることで，破棄する ACK 数に制限を設ける．これにより，HA-CN 間でセグメントがロスした場合であっても，必要以上に重複 ACK を破棄しないことで，AL がセグメントのロスを検知することができる．

4.4.2.3　*Mode* が必要モードの場合

対応する HoleList 要素のシーケンス番号空間のセグメントは，ロスしたことが明白である．したがって，HA は受信した ACK を AL へと配送する．

4.5　SACK との関係

今日多くの OS の TCP で Selective ACK（SACK）が利用可能となっている．SACK を用いる場合，受信側から明確に欠落したパケットを送信側に通知可能になるが，依然として受信側で検出した欠落パケットが再送不要であるかどうかは送信側では検知できない．受信側で一時的に欠落されたと見なされていても，HA ではそのパケットの受信に成功している場合には，送信側が再送ならびに輻輳ウィンドウの縮小をする必要はない．したがって，HA 不要 ACK 破棄法は SACK を用いる場合でも有効と考えられる．

5.　RAH 経路選択法

AL における経路選択を適切に行うことでセグメントの順序逆転を防ぐ方法として，AL-HA 間 RTT および送信キューでの待ち時間に基づく RAH（RTT between AL and HA）経路選択法を提案する．文献 15) では，複数経路を利用した TCP 通信において，End-to-End での RTT と送信キューでの送信待ち時間を利用した送信経路選択手法が提案されている．本提案は，文献 15) で提案された Arrival-Time maching Load-Balancing（ATLB）法を，送信端末である AL とセグメント中継端末である HA 間に応用したものである．

概要を**図 7** に沿って説明する．HA は，AL からセグメントを受信すると，AL へ擬似 ACK を送信する．AL は，データセグメントの送信から擬似 ACK の受信までの時間によって，利用する各経路ごとに AL-HA 間 RTT を測定する．さらに，AL がセグメントを送信する時点で，送信キューにある送信待ちデータ量と，一定時間内に受信した擬似 ACK から算出されるスループットから，各経路の送信キューでの送信待ち時間を算出する．そして AL は，利用する各経路の AL-HA 間 RTT および送信キューでの送信待ち時間の合計が最小の経路，つまり HA にセグメントが最も早く到着すると推測される経路を送信経路として選択する．これにより，AL-HA 間におけるセグメントの順序逆転の発生頻度を可能な限り減少させることができる．

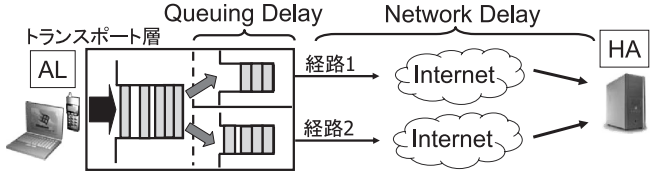

図 7　RAH 経路選択法
Fig. 7　RAH path selection method.

5.1　AL-HA 間の RTT 測定

AL は，セグメントを送信する際，TCP のオプションを使って，送信する経路の ID と送信時刻のタイムスタンプを TCP ヘッダに付加する．セグメントを受信した HA は，AL に返信する擬似 ACK の TCP ヘッダに含まれる経路 ID フィールドおよびタイムスタンプオプションに，受信したセグメントの TCP ヘッダに含まれる経路 ID およびタイムスタンプをセットし，AL へ送信する．HA 不要 ACK 破棄法で増加する 7 バイトのオプションに加え，10 バイトのタイムスタンプオプションの追加にともない，TCP ヘッダサイズは合計 17 バイト増加する．実際にはパディングを含み 20 バイトの増加となる．AL では，受信した擬似 ACK の経路 ID およびタイムスタンプオプションを用いることで，利用する各経路の AL-HA 間 RTT を測定することができる．また，AL-HA 間 RTT は，リンク状態によって大きく変動する可能性がある．そこで式 (1) を用いて平滑化を行う．

$$srtt(i) = \alpha \cdot srtt(i-1) + (1-\alpha) \cdot rtt(i) \quad (i > 0) \qquad (1)$$
$$srtt(0) = rtt(0)$$

$rtt(i)$ は，経路 i で送信したセグメントに対する擬似 ACK を受信した際に測定した AL-HA 間 RTT である．$srtt(i-1)$ は，以前に経路 i で送信したセグメントに対する擬似 ACK を受信した際に算出された AL-HA 間 RTT を用いて平滑化された RTT である．α の値には多くの TCP での実装と同様に 7/8 を用いた．

5.2　送信キューでの送信待ち時間の測定

AL は，セグメントを送信する際，利用する各経路の送信キューに溜まっている送信待ちデータ量 $Q(i)$ をチェックする．そして $Q(i)$ と各経路ごとに測定するスループット $G(i)$ を用いることで，各経路における送信キューでの送信待ち時間（$Q(i)/G(i)$）を得る．ここで，各経路ごとに測定するスループットとは，一定時間内に受信する擬似 ACK によって確認応答されたデータ量である．各経路ごとに測定するスループットにおいても AL-HA 間 RTT 同様に式 (2) を用いて平滑化を行う．

$$G(i) = \beta \cdot G(i-1) + (1-\beta) \cdot TPUT(i) \quad (i > 0) \qquad (2)$$
$$G(0) = TPUT(0)$$

β の値は文献 15) と同様に 1/2 とした．$TPUT(i)$ は，経路 i で一定時間ごとに算出されるスループットである．シミュレーションでは 1 秒ごとに算出した．

5.3　各経路における HA までの到着時間の算出

AL は，送信キューでの送信待ち時間の算出結果（$Q(i)/G(i)$）と AL-HA 間 RTT の測定結果から推測する片側遅延（$srtt(i)/2$）の和として，HA までの到着時間 $SCORE(i)$ を各経路ごとに算出する．そして AL は $SCORE(i)$ が最小の経路を送信経路として選択する．

$$SCORE(i) = \frac{Q(i)}{G(i)} + \frac{srtt(i)}{2} \tag{3}$$

6. シミュレーション評価

HA 不要 ACK 破棄法および RAH 経路選択法の性能評価を行うため，ns2.31 を用いてシミュレーションを行った．この結果に基づき，以下の 3 つの送信経路選択方法における HA 不要 ACK 破棄法の効果を比較する．

- Round Robin 方式
- 遅延比方式（利用する 2 経路の遅延の比に応じた比率で送信機会を提供する方式）
- RAH 経路選択方式

シミュレーションに用いたネットワークトポロジを**図 8** に示す．

AL-HA，AM-HA 間のリンクの特性は，3.5 世代携帯電話の上りパケット通信を想定し，帯域幅 384 kbps，遅延 150 ms，パケットロス率を 0.1%とした．また，AM-HA 間の遅延を 150〜290 ms，パケットロス率を 0.1〜1.0%の範囲で変動させた．AL-AM 間のリンクには IEEE802.11b 無線 LAN を想定し，帯域幅 11 Mbps，遅延 1 ms，パケットロス率は 0%とした．通常無線 LAN のパケットロスが完全に 0%となることはないが，AL-AM 間のパケットロスによって AM 経由の AL-HA 間の経路全体のパケットロスを表現しているといえるので，今回のシミュレーションでは AL-AM 間の無線 LAN リンクのパケットロスは 0%としている．なお，今回想定したパケットロス率は ARQ による再送を行う同環境としては大きい値と考えられる．しかし，HA 不要 ACK 破棄法は，パケットロスが発生しているときに必要な重複 ACK を破棄することで，性能の悪化を招く可能性を持つため，パケットロス率が比較的大きい状況での評価結果を示すことで同方式の有効性を示すことができると考える．シミュレーションでは，100 秒間のファイル転送を行い，20 回の平均値を評価に用いた．なお，4.1 節で示した，HA での再送要請までの閾値 θ_{C} は 5 としている．AL の TCP のモデルは，TCP-Reno に SHAKE による複数経路への分配機能と，HA 不要 ACK 破棄法で必要なタイムスタンプと経路 ID，$M_{\mathrm{AL}}(i)$ のヘッダへの追加処理機能を追加したものとした．また，CN の TCP のモデルは TCP-Reno 相当のものである．データパケットのサイズは TCP，IP のヘッダを含め 1,500 バイトとし，経路識別子，M_{AL}，タイムスタンプの追加に必要となる TCP ヘッダサイズ増加の影響を考慮してグッドプットを計算した．

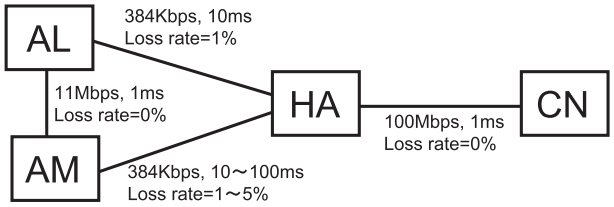

図 8　ネットワークトポロジ
Fig. 8　Network topology.

6.1　リンクの遅延差の影響

AM-HA 間のパケットロス率を 0.1%で固定し，遅延を 150〜290 ms で変化させたときのシミュレーション結果を**図 9** に示す．送信経路選択方法に RAH 経路選択法を用いた場合，Round Robin 方式および遅延差方式に比べて，高いグッドプットを得ることができた．さらに，各送信経路選択方法において，HA 不要 ACK 破棄法を併用することで，各送信経路選択方法のみ用いた場合に比べて，より高いグッドプットを得ることができた．AM-HA 間の遅延が 210 ms 以下の状況では，再送が発生するほどの複数のセグメントをまたいだ順序

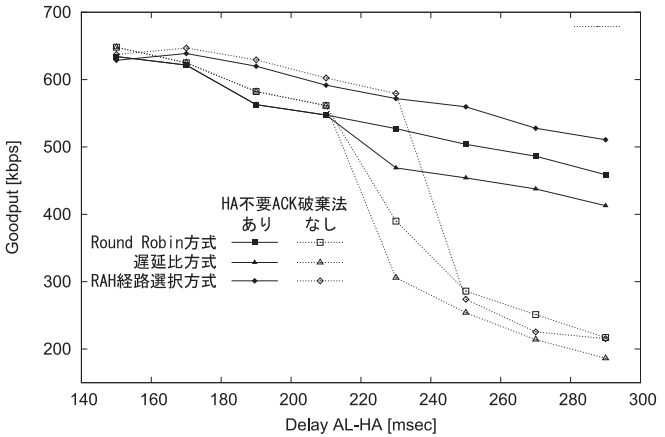

図 9　リンクの遅延差とグットプットの関係
Fig. 9　Goodput vs. difference of delay between routes.

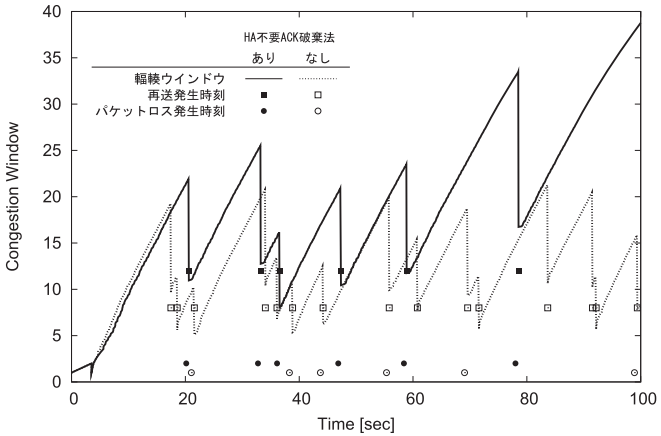

図 10　AL-HA 間の遅延が 230 ms の場合の輻輳ウィンドウの推移の一例
Fig. 10　Transition of congestion window. Delay of route 2 is 210 ms.

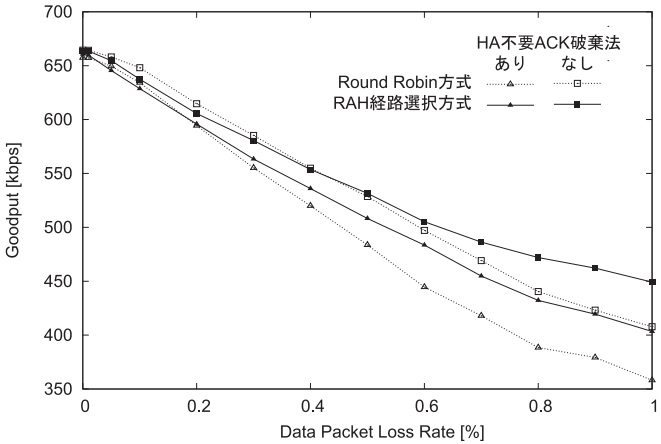

図 11　パケットロス率とグッドプットの関係（HA 不要 ACK 破棄法）
Fig. 11　Goodput vs. packet loss rate (HA unnecessary ACK dropping method).

逆転の発生頻度が低いため，HA 不要 ACK 破棄法の効果は認められず，グッドプットは用いない場合よりも若干小さくなった．これは，本来必要な ACK の破棄にともなう性能悪化と考えられる．

HA 不要 ACK 破棄法の効果を確認するため，送信経路選択方法に Round Robin 方式を用いた場合における AL-HA 間の遅延が 230 ms の場合の輻輳ウィンドウの推移の一例を**図 10** に示す．HA 不要 ACK 破棄法を用いない場合，シミュレーション時間内において頻繁に再送が発生し，輻輳ウィンドウが 5〜20 程度を推移している．一方，HA 不要 ACK 破棄法を用いた場合，HA 不要 ACK 破棄法を用いなかった場合に比べて，再送が頻発することなく，輻輳ウィンドウの縮小を抑制していることが分かる．HA 不要 ACK 破棄法を用いない場合，15 回の再送が発生している．しかし，実際にロスし，再送されるべきセグメントは，5 個である．一方 HA 不要 ACK 破棄法を用いた場合，実際にロスが発生したときにのみ再送が行われている．

6.2　パケットロス率の影響

AM-HA 間の遅延を 10 ms で固定し，パケットロス率を 0.1〜1.0%で変化させたときのシミュレーション結果を**図 11** に示す．パケットロス率が上がるにつれて，HA 不要 ACK 破棄法を用いた場合のグッドプットが，HA 不要 ACK 破棄法を用いない場合を大きく下回ってしまうことが確認できる．さらに，送信経路選択方法として Round Robin 方式を用いた場合，パケットロス率が上がるにつれて，HA 不要 ACK 破棄法を用いた場合は，用いない場合に比べてグッドプットの低下が著しい．

主な理由として，再送タイムアウトの増加が考えられる．HA 不要 ACK 破棄法では 4.1 節で述べたように，HA が重複 ACK を受信した状況では，対応する HoleList 要素の *Mode* が未確定モードの場合，HoleList 要素に対応するセグメントが経路の遅延によって遅れて配送されている可能性を考え，その重複 ACK を破棄する．しかし，HoleList 要素に対応するセグメントが実際にロスした場合，HA が重複 ACK を破棄してしまった分，AL でロスを検知するまでの時間が長くなってしまう．ロスを検知するまでの時間が長くなると，再送タイムアウトが発生してしまう．TCP では，高速再送でロスが検知されると，輻輳ウィンドウの値を半分にする．一方で，再送タイムアウトでロスが検知されると輻輳ウィンドウの値が 1 まで減少してしまう．HA 不要 ACK 破棄法を用いた場合，高速再送の発生頻度を抑制することができても，再送タイムアウトの発生頻度が増加することで，高いグッドプットを得ることができない．

この問題を解決するため，HA 不要 ACK 破棄法に改良を加える．HA において不要と判別され，破棄される重複 ACK の中で，再送が発生しない重複 ACK の上限である 2 つまで破棄せず AL へ転送する．以下，この方法を TDAR（Two Duplicate ACK Relay）方式

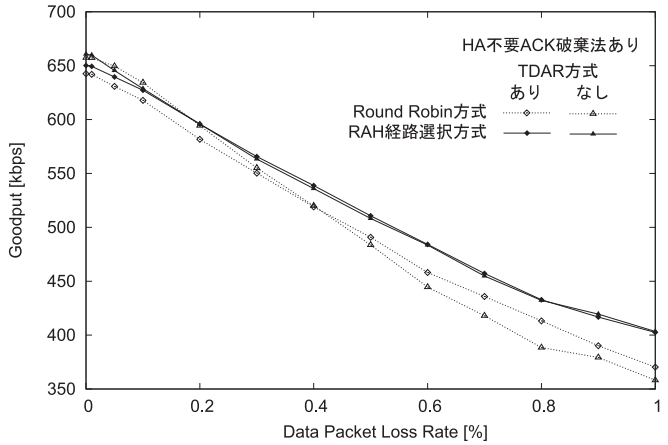

図 12　パケットロス率とグッドプットの関係（TDAR 方式）
Fig. 12　Goodput vs. packet loss rate (TDAR Method).

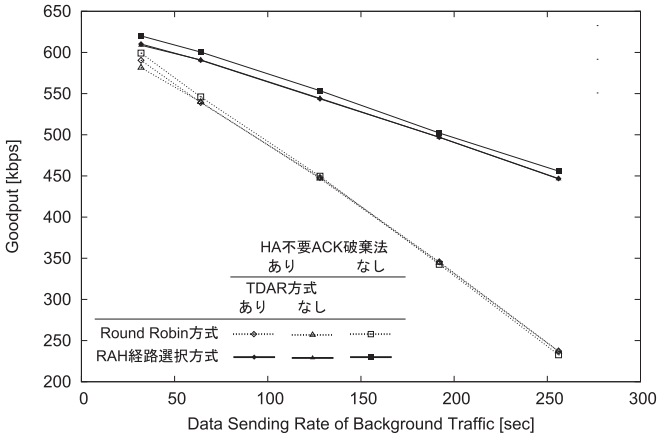

図 13　バックグラウンドトラフィックの影響
Fig. 13　Effect of background traffic.

と呼ぶ．

TDAR 方式を用いる場合，HoleList の属性として不要 ACK を通過させた数を示すカウンタ c_p を設ける．c_p は HoleList 作成時に 2 に初期化される．HA が ACK を受信したとき，4.4.2 項で述べたように $ack \leq A$ の場合に ACK の破棄が行われる可能性がある．

まず，Mode が未確定モードの場合（4.4.2.1）を考える．対応する HoleList 要素の属性値 *Cancel* が θ_c 以下ならば，TDAR 方式を使用しない場合には ACK を破棄し，HoleList 要素の *Cancel* をインクリメントする．一方，TDAR 方式を使用する場合，HoleList において $c_p > 0$ なら ACK を破棄せず転送し，c_p を 1 減少させる．$c_p \leq 0$ の場合には，TDAR 方式を使用しない場合と同じ処理を行う．なお *Cancel* の値は，ACK を破棄しない限り増加させない．

次に Mode が不要モードの場合を（4.4.2.2）考える．TDAR を使用しない場合は ACK を破棄し，HoleList 要素の属性値 *Dupack* を 1 減少させる．一方 TDAR を使用する場合には，$c_p > 0$ なら ACK を破棄せず転送し，*Dupack* と c_p を 1 減少させる．これ以外の処理は TDAR を用いない場合と同じである．

AM-HA 間のパケットロス率を変化させた場合の TDAR 方式のシミュレーション結果を**図 12** に示す．経路選択に Round Robin を用いた場合には，TDAR 方式を用いるとパケットロス率が 0.5%以上のときに，5〜20 kbps 程度のグッドプットの向上がみられる．

6.3　バックグラウンドトラフィックの影響

AM-HA 間の遅延とパケットロス率を AL-HA 間と同じにし，AM から CN へ CBR トラフィック（パケットサイズ：1,000 bytes，送信レート：64〜256 kbps）を流した場合のシミュレーション結果を**図 13** に示す．送信経路選択として RAH 経路選択方式を用いることで，大幅なグッドプット向上が得られている．

6.4　遅延変動の影響

3.5 世代携帯電話網のリンクでは，パケットロス時に ARQ による再送のために，一時的に遅延が増加する．この影響を確かめるため，平均 5 秒の指数乱数による間隔で，指数乱数で平均 0〜1.0 秒間 AL-HA および AM-HA 間のリンクでパケット転送を休止させた場合のシミュレーションを行った．AM-HA 間の遅延とパケットロス率は AL-HA 間と同じである．結果を**図 14** に示す．休止時間が長くなるにつれてグッドプットは減少するが，RAH 経路選択法を用いる場合，休止時間が長い場合にグッドプットが改善されている．一方，休止時間が比較的短い場合は，Round robin 方式の方が高いグッドプットが得られている．休止時間が比較的短い場合，RAH 経路選択法では過去に観測したスループットの算出値の影響によって，一時的に休止しているリンクを選択する頻度が高くなったために，グッドプットが低下したと考える．これは，スループットの算出値頻度を細かくすることで防止できる

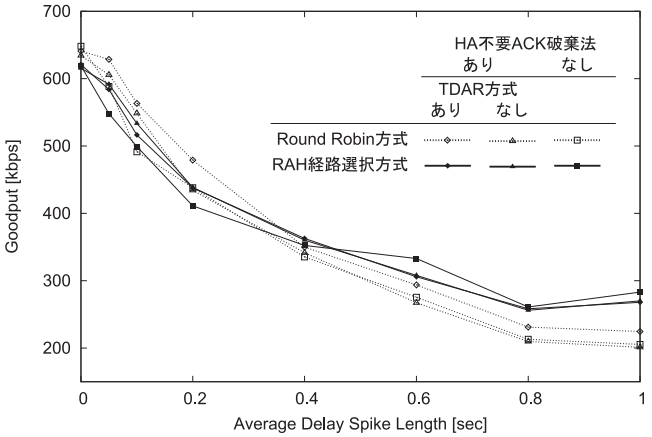

図 14　遅延変動の影響
Fig. 14　Effect of delay jitter.

と考えられる．なお，経路選択に Round Robin 方式を用いる場合，HA 不要 ACK 破棄法と TDAR 方式を使うことによって 5〜10%程度のグッドプットの改善が確認できる．

7. ま と め

本論文では，Mobile IP SHAKE における TCP 通信において，利用する各端末の外部リンクの性質がそれぞれ異なることで，セグメントの順序が逆転して受信端末に到着し，不要な重複 ACK の発生にともなう無駄な再送や輻輳ウィンドウの縮小が頻発するという問題を示し，その対策として，HA 不要 ACK 破棄法および RAH 経路選択法を提案した．シミュレーションの結果，RAH 経路選択法を用いることで，Round Robin 方式および遅延比方式に比べ，高いグッドプットを得ることが確認できた．また，経路間の遅延差が大きい場合には，各送信経路選択に HA 不要 ACK 破棄法を併用することで，不要な重複 ACK による再送が抑制され，輻輳ウィンドウの低下・グッドプットの減少を避けることができることが分かった．HA 不要 ACK 破棄法を用いた場合，パケットロス率が増加すると，HA 不要 ACK 破棄法を用いない場合に比べてグッドプットが著しく低下することが観察されたが，AL で再送が発生しない上限まで重複 ACK を AL へ転送する TDAR 方式を導入することで，グッドプットの低下を避けることができた．今後の課題として，HA における AL への ACK 転送経路の選択手法の検討，HA で破棄するセグメント数の閾値 θ_C の評価があげられる．

参 考 文 献

1) Koyama, K., Ito, Y., Mineno, H. and Ishihara, S.: Performance evaluation of TCP on Mobile IP SHAKE, *IPSJ Journal*, Vol.45, No.10, pp.2270–2278 (2004).
2) 舛田知広，大木一将，峰野博史，石原　進：Mobile IPv6 を用いた通信回線共有方式の実装，情報処理学会論文誌，Vol.46, No.9, pp.2214–2225 (2005).
3) Rojviboonchai, K., Osuga, T. and Aida, H.: R-M/TCP: Protocol for Reliable Multi-path Transport over the Internet, *Proc. AINA 2005*, Vol.1, pp.801–806 (2005).
4) 板谷俊輔，相田　仁：複数経路通信プロトコル M/TCP の FreeBSD への実装の性能向上，電子情報通信学会技術報告，Vol.105, No.628, pp.213–218 (2006).
5) Lee, Y., Park, I. and Choi, Y.: Improving TCP Performance in Multipath Packet Forwarding Networks, *Journal of Communications and Networks*, Vol.4, No.2, pp.148–157 (2002).
6) Zhang, M., Lai, J., Krishnamurthy, A., Peterson, L. and Wang, R.: A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths, *Proc. 2004 USENIX Annual Technical Conference*, pp.1–14 (2004).
7) Kim, K. and Shin, K.G.: Improving TCP Performance over Wireless Networks with Collaborative Multi-homed Mobile Hosts, *Proc. MobiSys '05*, pp.107–120 (2005).
8) Gerla, M., Lee, S.S. and Pau, G.: TCP Westwood Simulation Studies in Multiple-path Cases, *2002 International Symposium on Performance Evaluation of Computer and Telecommunication Systems*, pp.287–297 (2002).
9) 殿内雅晴，水野忠則：複数経路通信向けトランスポートプロトコルの性能評価，情報学ワークショップ 2004，U1-3, pp.226–231 (2004).
10) Iyengar, J.R., Amer, P.D. and Stewart, R.: Concurrent Multipath Transfer Using SCTP Multihoming Over Independent End-to-End Path, *IEEE/ACM Trans. Networking*, Vol.15, No.5, pp.951–964 (2006).
11) Casetti, C. and Gaiotto, W.: Westwood SCTP: load balancing over multipaths using bandwidth-aware source scheduling, *Proc. VTC2004-Fall*, Vol.4, pp.3025–3029 (2004).
12) Perkins, C.: IP mobility support for IPv4, RFC 3344 (2002).
13) Johnson, D., Perkins, C. and Arkko, J.: Mobility support for IPv6, RFC 3775 (2004).
14) Floyd, S., Mahdavi, J., Mathis, M. and Podolsky, M.: An extension to the selective acknowledgement (SACK) option for TCP, RFC 2883 (2000).

15) Hasegawa, Y., Yamaguchi, I., Hama, T., Shimonishi, H. and Murase, T.: Improved data distribution for multipath TCP communication, *Proc. IEEE GLOBECOM'05*, Vol.1, pp.271–275 (2005).

(平成 20 年 3 月 31 日受付)
(平成 20 年 10 月 7 日採録)

荻野　秀岳

平成 18 年静岡大学工学部システム工学科卒業．平成 20 年同大学大学院工学研究科修士課程修了．同年ヤマハ（株）入社．会議システム関連業務に従事．

石原　　進（正会員）

平成 6 年名古屋大学工学部電気工学科卒業．平成 11 年同大学大学院工学研究科博士後期課程修了．平成 10 年日本学術振興会特別研究員．平成 11 年静岡大学情報学部助手．平成 13 年同大学工学部助教授．現在，静岡大学創造科学技術大学院准教授．博士（工学）．平成 9 年電気通信財団テレコムシステム技術学生賞．モバイルコンピューティング，無線環境用 TCP/IP，モバイルアドホックネットワークに関する研究に従事．電子情報通信学会，IEEE，ACM 各会員．