MP-TCP：マルチホーム環境下において帯域集約を実現するトランスポート層プロトコルの提案

五十嵐 健[†,††] 重野 寛[†] 井原 武[††]
三浦 章[††] 岡田謙一[†]

多様な無線アクセス技術の登場とともに，複数の無線インタフェースを持ったマルチホーム端末が一般的になりつつある．しかし，マルチホームの効果的な利用法の 1 つである帯域集約を実現するためには，集約されたパスの Round Trip Time の違いによってセグメントが受信側に順序どおりに到着しないため，トランスポート層プロトコルとして TCP が用いられた場合，Fast Retransmit and Fast Recovery が起動されスループットが低下してしまうという問題を解決する必要がある．本稿では，トランスポート層においてセグメントの順序違いを吸収し，帯域集約を実現する MP-TCP を提案する．MP-TCP では，集約する各パスごとに TCP Control Block を用意し，送信バッファを改良することによって TCP/IP ヘッダのみを用いて，複数のパスを独立して管理することを可能にする．さらに Delayed ACK の改良や，複数の ACK パスを利用することによって，集約される各パスについて，既存の TCP と同等の制御を可能にする．MP-TCP をコンピュータシミュレーションによって評価した結果，MP-TCP は既存の TCP と同様に ACK 数を 50%程度削減できるとともに，集約される各パスにおいて既存の TCP との公平性を保って帯域集約を実現できることが示された．

MP-TCP: Transport Layer Protocol for Network Bandwidth Aggregation under Multi-homing Environment

Ken Igarashi,[†,††] Hiroshi Shigeno,[†] Takeshi Ihara,[††]
Akira Miura[††] and Kenichi Okada[†]

Emerging of various wireless technologies, multi-homing hosts which have multiple wireless interfaces become general. One of the effective usage of multi-homing is bandwidth aggregation; however, the difference of Round Trip Time of aggregated paths causes segments misorder and throughput is degraded, because Fast Retransmit and Fast Recovery is activated if TCP is used as a transport layer protocol. MP-TCP is proposed to solve the problem. MP-TCP can maintain several TCP Control Blocks and the improvement of the send buffer makes possible to manage each path independently only using TCP/IP header. In addition, adoption of Delayed ACK and using several ACK paths enable MP-TCP to achieve same control as TCP for each aggregated path. To evaluate MP-TCP by computer simulation, the results show that MP-TCP can achieve ideal bandwidth aggregation, while MP-TCP can reduce the number of ACKs by 50%, which is the same level as conventional TCP. Moreover, MP-TCP can maintain fairness to conventional TCP for each path, when it performs bandwidth aggregation.

1. はじめに

モバイルインターネットユーザの増加にともない，多様な無線アクセス技術が開発されている．たとえば WLAN: Wireless Local Area Network の分野においては IEEE 802.11 a/b/g，WPAN: Wireless Personal Area Network の分野では IEEE 802.15.1（Bluetooth），IEEE 802.15.4（ZigBee），WWAN: Wireless Wide Area Network の分野においては GPRS/GSM，1xRTT/CDMA 等が登場し，今日ではそれら無線インタフェースを複数持った，マルチホーム端末が一般的になりつつある．

マルチホームを効率的に扱うことによって，次のような機能の実現が期待される．

シームレスハンドオフ：無線アクセス技術によって異なるカバーエリアの違いを利用し，ハンドオーバ時の通信を継続する．

† 慶応義塾大学大学院開放環境科学専攻
Graduated School of Science and Technology, Keio University
†† 株式会社 NTT ドコモネットワーク研究所
Network Laboratories, NTT DoCoMo, Inc.

対障害性の向上：ある無線リンクの状態が悪化した場合に，他の無線リンクへ迂回する．

帯域集約（負荷分散）：複数のインタフェースが提供する帯域を集約し，1 つのアプリケーションに対して集約した帯域を提供する．

トランスポート層でマルチホームを扱った技術としては，SCTP [14),15)] や Migrate-Permitted TCP オプション[13)] が提案されている．しかし，これらの技術は，対障害性の向上やシームレスハンドオフの実現を目指して提案されたものであり，帯域集約の実現を目指したものではない．

トランスポート層より下の層においてマルチホームを扱った技術としては Mobile IP（MIP）[1)] や HIP [3)] が提案されている．これらの技術を用いることによって SCTP や Migrate-Permitted TCP と同様に，対障害性の向上やシームレスハンドオフを実現することができる．一方，帯域集約を実現するためには，上位層であるトランスポート層で用いられるプロトコルを考慮する必要がある．特に，インターネット上で最も使われている[2)] TCP [4)] が用いられた場合には MIP や HIP による帯域集約の実現は非常に困難となる．

TCP は信頼性のあるセグメントの配送を実現するために用いられるトランスポート層のプロトコルであり，ロストしたセグメントの再送機マルチホーム環境下において能のほかに，フロー制御機能，輻輳制御機能を備えている．TCP では 2 つのトリガを利用してセグメントロスを検出し，セグメントロスをネットワークの輻輳と判断し，輻輳制御を行う．セグメントロスを検出する 1 つ目の手段は，一定期間 ACK を受け取らなかった際にセグメントを再送する Retransmission Timeout（RTO）[5)] である．もう一方の手段が Duplicate Acknowledgment（D-ACK）である．D-ACK は受信側で順序違いのセグメントを受け取った際に，受信側が本来受信を期待するセグメントのシーケンス番号を示す ACK を返すことによって生成され，通常は 3 つの D-ACK を受信すると Fast Retransmit and Fast Recovery（F-Retran）[6)] に従ってセグメントが再送される．

帯域集約を行う場合，集約されるそれぞれのパス☆の帯域幅，混雑度合，伝播遅延の違いにより，各パスにおける Round Trip Time（RTT）が異なることが大きな問題を引き起こす．RTT が異なる場合，送信されたセグメントが受信側に順序どおりに到着しない．その結果，受信側で D-ACK が生成され，セグメントロスが発生していないにもかかわらず F-Retran が起動され，スループットが低下してしまう．

トランスポート層より下の層の技術によってこの問題を解決するためには，受信側へセグメントを順序どおりに到着させるための何らかの機能が必要となる．たとえば RTT の違いを考慮してパケットの送出順を決定する等の方法が考えられるが，1 つのパスにおいても混雑度合いや，無線リンクで用いられる再送機構等により RTT が揺らぐため，セグメントの到着時間を予想することは難しく，トランスポート層プロトコルを変更せずに，トランスポート層より下の層の技術だけでは帯域集約を実現することは非常に困難である（詳細は 2.1 節参照）．

本稿では，トランスポート層において集約されるパスの RTT の違いに起因するスループットの低下を防いで帯域集約を実現する Multi-Path TCP（MP-TCP）を提案する．MP-TCP では，パスごとに TCP Control Block（TCP-CB）を用意することで複数のパスを独立して扱うことを可能にする．さらに，複数のセグメントの受信報告を 1 つの ACK でまとめて行うことにより ACK 数を削減する Delayed ACK（Del-ACK）[25)] の改良や，複数の ACK パスを利用することによって，集約される各パスに対して，既存の TCP と同等の制御を実現する．また，送信バッファの改良によって，既存の TCP/IP ヘッダのみを用いてこれらの制御を実現するため，受信側が MIP のようにネットワーク層においてマルチホームを扱うことができる技術を備えている場合，受信側の TCP へ変更を加えることなく帯域集約を実現することができる（詳細は 3.5 節参照）．

本稿の構成は次のとおりである．2 章において帯域集約を実現するための課題および関連研究の紹介を行う．3 章では，提案方式である MP-TCP についての説明を行い，4 章においてコンピュータシミュレーションによって MP-TCP を評価した結果について報告する．そして，5 章でまとめおよび今後の課題について述べる．

2. 帯域集約の実現

帯域集約を行う場合，集約されるそれぞれのパスの帯域幅，混雑度合い，伝播遅延の違いにより，各パスにおける RTT が異なることが大きな問題を引き起こす．さらに集約されるパスに無線リンクが含まれる場合，無線区間における誤りの影響を軽減し，高信頼度のパケット伝送を実現するためにデータリンク層に

☆ 本稿では送信元 IP アドレス，宛先 IP アドレスの組を 1 つのパスと定義する．

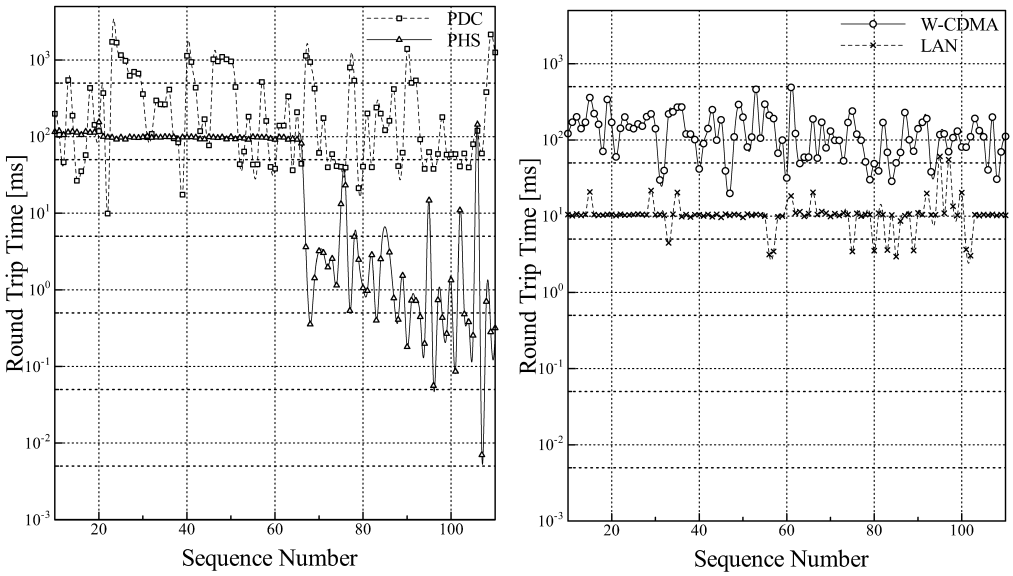

図 1 無線アクセス技術ごとの RTT の揺らぎ
Fig. 1 RTT fluctuation of each radio access technology.

おいて，再送機構として Automatic Repeat reQuest（ARQ）[17] が用いられた場合，無線リンクの状態によって大きく RTT が揺らぐことが予想される．

図 1 に無線アクセス技術ごとに計測した RTT の揺らぎを示す．RTT の揺らぎ（RTT_{fluc}）は TCP Vegas [29] と同様，計測された RTT（$RTT_{measure}$）と，計測された最小の RTT（RTT_{base}）を用いて式 (1) のように定義した．

$$RTT_{fluc} = RTT_{measure} - RTT_{base} \quad (1)$$

計測に用いた無線アクセス技術は Personal Handyphone System（PHS），Personal Digital Cellular（PDC），Wideband Code Division Multiple Access（W-CDMA）であり，無線アクセス以外の影響を最小マルチホーム環境下において限にするために，*traceroute* を用いて特定された最初のルータとの間で RTT を計測した．

図 1 の結果から，無線アクセス技術によって RTT が大きく異なるとともに，RTT が大きく揺らいでいることが分かる．RTT の揺らぎが最も大きな PDC において約 2.5 秒，最も小さな PHS においても約 155 ミリ秒程度の RTT の揺らぎが計測された．通常はさらに有線を経由して通信が行われるため，有線ネットワークの混雑による揺らぎ（図 1 中の LAN 部分参照☆）が加わることになる．帯域集約を実現するためには，このようなパスごとの特性の違いに対応する機能が必要となる．

2.1 関 連 研 究

TCP に変更を加えずに帯域集約を実現する技術として SHAKE [18] が提案されている．SHAKE では RTT を予測し，受信側へセグメントが順序どおりに届くように，遅延を加えてセグメントを送信する．しかし，無線リンクが用いられた場合，RTT は大きく揺らぐため（図 1 参照），セグメントの到着時刻を正確に予測することは困難であり実現は難しい．

他の技術としては，アプリケーション層において複数の TCP コネクションをまとめることによって帯域集約を実現する技術が提案されている[7),10),☆☆]．しかし，アプリケーション層で帯域集約を行う場合，集約されるパスにスループットの低いパスが含まれている際には期待される集約効果を得ることができない．

TCP によって達成されるスループットは，Maximum Segment Size（MSS）に比例して，RTT とパケットロス率（p）に反比例することが知られ，定数 C を用いて式 (2) のように表すことができる[19]．

$$B \leq \frac{MSS}{RTT}\frac{C}{\sqrt{p}} \quad (2)$$

n 本の帯域集約を行った場合，期待されるスルー

☆ 計測は 100Base-TX のイーサネットを用いて，PHS の計測に用いたルータと www.mos.ics.keio.ac.jp の間で計測した．

☆☆ これらの方法マルチホーム環境下においては WEB ブラウザや GridFTP [11] 等において，同一パスに複数の TCP コネクションを張ることによるスループットの向上を目的としているが，その性質上，帯域集約にも適用することができる．

プットは式 (3) で表される．

$$B_{agg} \leq \left[\frac{MSS}{RTT_1} \frac{C}{\sqrt{p_1}} + \cdots + \frac{MSS}{RTT_n} \frac{C}{\sqrt{p_n}} \right] \quad (3)$$

送信したいデータ量を D_{total} として，式 (3) を利用すると，送信に必要となる時間 (T_{snd}) は式 (4) で表される．

$$T_{snd} = \frac{D_{total}}{B_{agg}} \quad (4)$$

しかし，アプリケーション層において帯域集約を実現する場合，あらかじめデータを割り当てることによって，独立した各 TCP コネクションの送信バッファを満たしておく必要がある．たとえば，集約される n 本の TCP コネクションに対してデータを均等に割り当てた場合，送信に必要となる時間は式 (5) のようになり，最もスループットの低い TCP コネクションがデータ送信を完了する時間に，通信時間が依存してしまう．

$$T_{snd} = \frac{D_{total}/n}{Min\left[\frac{MSS}{RTT_1} \frac{C}{\sqrt{p_1}} + \cdots + \frac{MSS}{RTT_n} \frac{C}{\sqrt{p_n}} \right]} \quad (5)$$

この問題を解決したのが pTCP [8),9)] である．pTCP では TCP の Congestion Window（CWND）の変化を監視し，集約している TCP コネクション内に，セグメントロス等により CWND が大きく減少したコネクションを発見すると，余剰となったセグメントを他の TCP コネクションに振り替えることで，データ送信を完了するまでの時間を，最もスループットの低いコネクションに依存させないようにしている．しかし，pTCP では複数の TCP コネクションを集約するために pTCP 用のヘッダを必要とするため，ヘッダ長が増加するとともに pTCP を装備していない受信側との通信においては，帯域集約を実現することができない．また，セグメントを再送する際に，コネクションごとに用意された送信バッファ間でセグメントの複製/削除が必要となる．

mTCP [12)] はトランスポート層で帯域集約を実現する技術である．mTCP では，TCP の SACK [20)] オプションを利用して，受信側に順序が異なって届いたセグメントを把握することによって，既存の TCP/IP ヘッダのみを用いての帯域集約が可能である．しかし，mTCP では ACK の順序違いに対応していないため，ACK パスについては RON [21),☆] によって選択されたプライマリパスのみを利用する．ACK パスを 1 本しか利用できないことは，3 つの問題を引き起こす．

1 つ目の問題は，対障害性の低下である．プライマリパスが切断された場合，すべてのパスから ACK が返らなくなるため，プライマリパスの切断は，他のすべてのパスへ影響を与えてしまう☆☆．

2 つ目の問題は，ACK 数の増大である．TCP ではフォワードパス（データパス）のセグメントロスに対しては輻輳制御を行うが，リバースパス（ACK パス）のセグメントロスに対しては輻輳制御を行わない．そのため，ACK 数の増加はネットワークの輻輳を引き起こす大きな要因となる [24)]．通常の TCP では Del-ACK を用いて ACK 数を削減することができる．しかし，RFC2581 [25)] ではセグメントの順序違いが発生している最中には，セグメントロスの検出を遅らせないために Del-ACK を利用することを奨励していない．そのため，帯域集約によってセグメントの順序違いが発生する場合には Del-ACK による ACK 数の削減効果は低下する．さらに，ACK はすべてプライマリパスに返されるため，集約するパスの数が増加するとともに，プライマリパスに返る ACK 数が増大する．

3 つ目の問題は，TCP では ACK の受信を契機にセグメントの送信を行うことで *self-clock* を保ってのセグメント送信を可能にしているが，通常とは異なるパスを利用して ACK が返された場合，*self-clock* が崩れてしまい，同一のパスを通る他の TCP コネクションとの公平性が保たれなくなるということである．特に大きな問題となるのは，式 (2) からも明らかなように，ACK パスが短くなることによって RTT が短縮された場合，通常よりも多くのセグメントを送信することが可能となり，他の TCP コネクションの帯域を奪ってしまう場合である．この問題を解決する種々の Active Queue Management 技術が提案されているが [22),23)]，現状ではその複雑さのために，あまり普及しているとはいえない．

3. Multi-Path TCP

MP-TCP では，Del-ACK の改良や複数の ACK パスを利用することによって，mTCP がもたらす問題を解決し（2.1 節参照），集約される各パスにおいて通常の TCP と同等の制御（送信レート，ACK 数）を行いながら帯域集約を実現する．また，pTCP のように，帯域集約のために TCP/IP ヘッダ以外のヘッダを

☆ RON では，*PROBE_INTERVAL* ごとに送信する probe によって，各パスの RTT やパケットロス率を測定し，式 (2) を利用して最もスループットが得られるパスをプライマリパスとして選択する．

☆☆ mTCP では，プライマリパス切断時に他のパスへの影響を取り除く方法を紹介しているが，プライマリパスで RTO が発生するまで通信が行えないことや，RTO の間に他のパスで発生したセグメントロスが無視されるという問題がある．

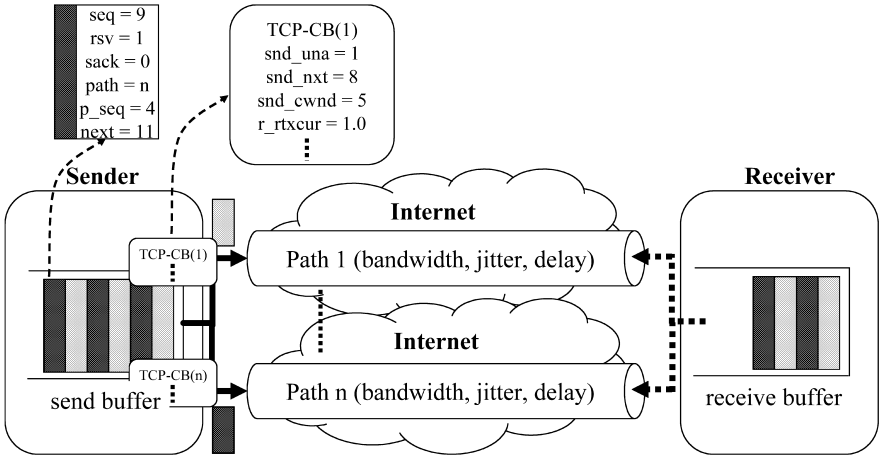

図 2 MP-TCP の概要
Fig. 2 MP-TCP overall.

利用しないため，受信側が MIP のようにネットワーク層においてマルチホームを扱うことができる技術を備えている場合，受信側の TCP へ変更を加えずに帯域集約を実現することができる（詳細は 3.5 節参照）．

3.1 MP-TCP の概要

図 2 に MP-TCP の概要を示す．MP-TCP では，通常の TCP と同様，最初は 1 つのパスを用いて 3 Way Handshake[4] を通して TCP コネクションを確立する．その他のパスは，最初に確立されたパスを利用して SCTP のソケット API[16] のようにマルチホーム向けに拡張されたソケット API を用いて，ユーザが制御を行う．新たなパスを追加する場合には，新たにアクティブとなったインタフェースの IP アドレスを取得し，取得した IP アドレスを *bind* システムコールによりソケットに追加する．そして，追加した IP アドレスを送信元に指定して *connect* システムコール☆を発呼することにより，Migrate-Permitted TCP 同様に SYN セグメントが送信され，新たに追加されたパスを管理する TCP-CB が新たなパス番号を付与され作成される．

このように，MP-TCP では通常 1 つの TCP コネクションに対して 1 つしか用意されない TCP-CB を，帯域集約を行うパスごとに用意することによって，特性（帯域幅，混雑度合い，伝播遅延等）が異なる複数のパスについて CWND（snd_cwnd）や RTT（r_rtxcur）等を独立して管理することにより，パスごとに輻輳制御および再送制御を行うことを可能にする．

パスの切断は，パスの追加と同様に任意のタイミングで行うことができる．パスを切断する場合，切断したいパスを指定し，*close* システムコールを通して利用可能なパスから FIN メッセージが送信され，TCP-CB や送信バッファで予約されているセグメントが RTO 発生時と同様に開放され（詳細は 3.4 節参照），パスが切断される．

MP-TCP では，TCP/IP ヘッダのみを用いて帯域制御を実現するために，送信バッファにおいて次のような情報とともにセグメントの管理を行う．

シーケンス番号（seq）：セグメントのシーケンス番号☆☆

予約フラグ（rsv）：セグメントが送信済み，またはいずれかのパスに予約されていることを表すフラグ

SACK パス番号（sack）：SACK を受信したパスの番号

パス番号（path）：予約フラグを設定したパスの番号

パスシーケンス番号（p_seq）：パスごとに割り当てられるセグメントのシーケンス番号

次セグメントシーケンス番号（next）：次セグメントへのシーケンス番号

受信側では，順序違いで受信したセグメントを，受信したパスの番号とともに管理する．送信側に ACK を返す場合，管理されているパスの番号を参照して，セグメントを受信したパスに対して ACK を返すことによって複数の ACK パスを利用する．さらに，セグメントの順序違いが発生している最中でも，同一のパ

☆ *connect* システムコールにおいて指定する宛先 IP アドレスは SCTP ソケット API における *sctp_getpaddrs* と同様のシステムコールを利用して取得する．

☆☆ TCP ではバイトオーダが用いられるが，本稿では簡略化のためにシーケンスオーダを用いる．

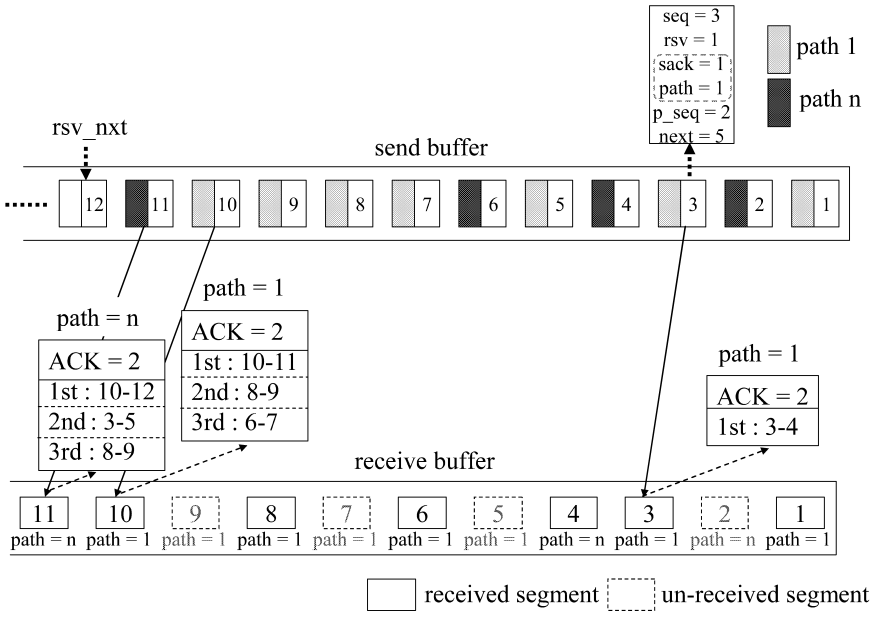

図 3 セグメントの送受信
Fig. 3 Send/Receive segment.

スにおいてシーケンス番号が連続するセグメントを受信した場合には Del-ACK の生成を可能にすることによって，帯域集約を行う場合においても ACK 数を削減することを可能にする．

3.2 セグメントの送受信

セグメントの送信には，受信側から返される ACK に含まれる SACK ブロックの情報を用いる．**図 3** を用いて，セグメントの順序違いが発生した場合における制御について説明する．セグメントの順序違いによって，複数のセグメントの抜け落ち（点線で示されたセグメント）が発生している状況において，シーケンス番号 3 のセグメントを受信した場合，TCP ヘッダ内の ACK としては 2，SACK の第 1 ブロックには 3–4 を記入した ACK が返される．送信側では，受信した ACK に含まれる SACK ブロックの情報に従い，シーケンス番号 3 のセグメントの SACK パス番号として，ACK を受信したパスの番号（sack = 1）を設定する．シーケンス番号 3 のセグメントが，パス 1 の TCP-CB において管理されている ACK を受け取っていない最初のセグメント（snd_una）のときには，SACK パス番号とパス番号が一致した場合（セグメントの送信に用いたパスと同一のパスから ACK を受信した場合），パス 1 の TCP-CB で管理される CWND に従って，新たなセグメントを送信する．このように，MP-TCP では受信側に順序どおりに到着しないセグメントを SACK ブロックの情報を用いて判別し，パスごとに輻輳制御を行う．また，MP-TCP では ACK パスについても複数のパスを利用するため，SACK パス番号とパス番号が一致しない場合がある☆．セグメントを送信したパス以外のパスで受信した ACK に含まれる SACK ブロックの情報に基づいてセグメントを送信した場合 *self-clock* が崩れてしまうため（2.1 節参照），SACK パス番号とパス番号が一致しない場合には CWND を変更せず，新たなセグメントを送信しない☆☆．

3.2.1 SACK ブロックの生成

MP-TCP では，複数の ACK パスを効率的に扱うために新たな SACK ブロックの生成規則を用いる．RFC2018 [20] に従った場合，SACK ブロックは次のように生成される．

第 1 ブロック：ACK を生成する元となったセグメントを含むブロック

第 2 ブロック以降：最近生成された SACK ブロック（ただし，他のブロックと重複させない）

MP-TCP では，次のような SACK ブロックの生成規則を用いる．

第 1 ブロック：ACK を生成する元となったセグメントを含むブロック

第 2 ブロック以降：セグメントを受け取ったパスにお

☆ 他のパスからの ACK によって，セグメントの到着が報告された場合．
☆☆ 再送セグメントを含む．

いて，最近生成された SACK ブロック

空きブロック：当該パス以外のパスを含めて最近生成された SACK ブロック

ただし，各ブロックは RFC2018 と同様，お互いに重複することのないように設定する．

図 3 を用いて MP-TCP における SACK ブロックの生成規則を説明する．シーケンス番号 10 のセグメントを受信した場合，SACK の第 1 ブロックにはこの ACK を生成する元となったシーケンス番号 10 のセグメントを含む SACK ブロック 10–11 を用いる．第 2 ブロックには，パス 1 において前回 SACK ブロックを生成した際に第 1 ブロックに用いた，SACK ブロック 8–9 を用いる．第 3 ブロックについても同様に，パス 1 で前々回，第 1 ブロックに用いた，SACK ブロック 6–7 を用いる．次にシーケンス番号 11 のセグメントを受信した場合について説明する．第 1，第 2 ブロックについては先ほどと同様に SACK ブロック 10–12, 3–5 を用いる．しかし，パス n で受信したセグメントでは，これ以上 SACK ブロックを生成することができないため，第 3 ブロックには他のパスで最近生成された SACK ブロックとなる，SACK ブロック 8–9 を用いる☆．

MP-TCP では *self-clock* を保つために ACK を受信したパスが，セグメントを送信したパスと一致しない限りセグメントの送信を行わない．そのため，受信したセグメントをパスごとに管理し，同一パスのセグメントを優先的に SACK ブロックへ含めることによって，1 つのセグメントが同一パスにおいて複数の ACK に含まれる可能性を高めることができる．このような SACK ブロックの生成規則を用いることによって，1 つの ACK がロストした場合でも，他の ACK によって，セグメントの到着を送信側へ伝えることが可能となり，ACK ロスに対する耐性を高めることができる．

たとえば図 3 においてシーケンス番号 4 のセグメントに注目した場合，RFC2018 に従って SACK ブロックを生成した場合，シーケンス番号 11 のセグメント受信時には各 SACK ブロックは，第 1 ブロック 10–12, 第 2 ブロック 8–9，第 3 ブロック 6–7 となり，シーケンス番号 4 のセグメントは SACK ブロックに含まれない．一方，MP-TCP の SACK ブロックの生成規則に従った場合は，同一のパスで受信されたセグメント

☆ 最近生成された SACK ブロックとしては，シーケンス番号 10 のセグメント受信時に生成された第 1 ブロックである SACK ブロック 10–11 になるが，すでに第 1 ブロックに含まれているため SACK ブロック 8–9 を用いる．

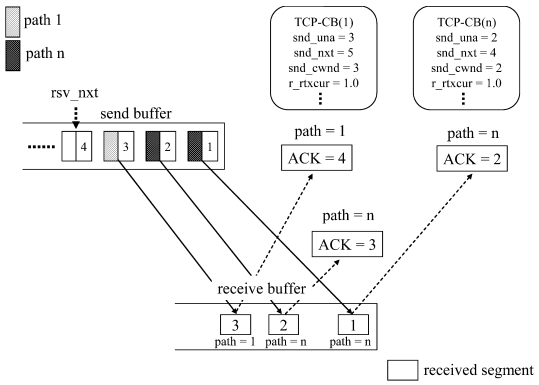

図 4 ACK の順序違い
Fig. 4 Miss-ordering of ACK.

が優先されるため，シーケンス番号 11 のセグメントを受信した際に生成される ACK の SACK ブロックに，シーケンス番号 4 のセグメントを含むことができる．さらに，SACK ブロックに空きが生じた際には，他のパスの情報を含めて SACK ブロックを生成することによって，無線の切断等によって 1 つのパスが切断された場合に，無駄なセグメントの再送を防ぐことができる☆☆．

3.2.2 送信済みセグメントの消去

MP-TCP では複数の ACK パスを利用するため，送信バッファから送信済みセグメントを消去する際に ACK の順序違いを考慮する必要がある．通常の TCP では，ACK を受けたセグメントは送信バッファから削除することができる．しかし，ACK の順序違いが発生する場合，他のパスを経由して先に届いた ACK によって snd_una に該当するセグメントが削除されてしまう可能性がある．snd_una に該当するセグメントが削除された場合，後に届く ACK によって snd_una に該当するセグメントの到着が報告されたときに，セグメント送出の可否を判断することができなくなってしまう．

図 4 を用いて，ACK の順序違いが発生した場合の動作について説明する．シーケンス番号 2 のセグメントの受信によって，パス n から ACK = 3 が返される．同様にして，シーケンス番号 3 のセグメント受信時に，パス 1 から ACK = 4 が返される．ここで，パス 1 の RTT がパス n の RTT よりも短いために，パス 1 から返される ACK（ACK = 4）がパス n から返される ACK（ACK = 3）よりも先に送信側に到着した場合，送信バッファからシーケンス番号 3 までの

☆☆ 再送時，SACK パス番号が設定されているセグメント（SACK ブロックによって受信側への到着が確認されたセグメント）は再送されない．

セグメントが消去される．そのため，パス n から後に到着する ACK（ACK = 3）を受け取った場合，すでにシーケンス番号 3 のセグメントは削除されているため，新たなセグメントの送出可否を判断することができない．そのため MP-TCP では，新たな ACK を受信した際に，その他のパスの snd_una を検査し，最も小さな snd_una までのセグメントのみを消去することにより ACK の順序違いに対応する．パス 1 から返る ACK（ACK = 4）を受信した場合，パス 1 以外のパスの snd_una を参照し，最も小さな snd_una となるパス n の snd_una（seq = 2）までのセグメントを送信バッファから消去することによって，ACK の順序違いが発生した場合でも，セグメント送出の可否の判断を可能にする．

3.3 Delayed ACK の生成

MP-TCP では，セグメントの順序違いが発生している最中でも ACK 数の削減を可能にするために次のような機能を提供する．送信側ではセグメントを送信する際に，1 つのパスで送信されるセグメントのシーケンス番号が連続するように，セグメントを送信する前にシーケンス番号が連続する複数のセグメントの送信予約を行う．予約されたセグメントの数は ACK を受信するごとに確認され，予約されているセグメント数が 1 つの ACK を受信した際に送信することができる最大のセグメント数（max-burst）を下回った場合，予約されたセグメント数が max-burst もしくは CWND の最大値になるよう，再度セグメントの予約を行う．しかし，RFC2581 ではセグメントの順序違いが発生している場合にはセグメントロスの検出を遅らせないために Del-ACK を利用することを奨励していない．そのため，セグメントの順序違いが頻発するような環境では，このような機能を用いた場合でも Del-ACK による ACK 数の削減効果は限られてしまう．そこで MP-TCP では，セグメントの順序違いが発生している最中でも，同一のパス内でシーケンス番号が連続するセグメントを受信した場合には Del-ACK を生成できるように受信側へ変更を加えた．セグメントの順序違いが発生している最中でも Del-ACK を生成可能にすることによって，ACK 数が減りセグメントロスの検出が遅れることが懸念されるが，MP-TCP では D-ACK の数ではなく，D-ACK に含まれる SACK ブロックの情報を用いてセグメントロスを検出するため，ACK 数が減少することによる影響を小さくすることができる（詳細は 3.4 節参照）．

3.4 セグメントの再送

ロストしたセグメントは各パスの TCP-CB で管理されている再送タイマ（t_rtxcur）の満了または，D-ACK に含まれる SACK 情報によって検出される．**図 5** を用いて，パス 1 で送信されたシーケンス番号 3 のセグメントがロストし，シーケンス番号 4–9 の SACK ブロックを含む ACK を受信した場合を例に用いてセグメントの再送手順を説明する．

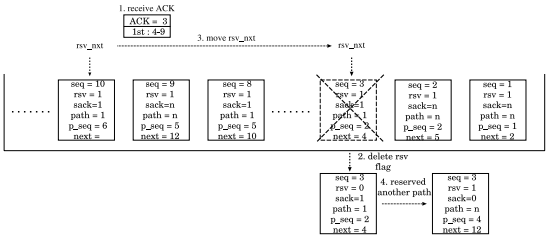

図 5 セグメントの再送
Fig. 5 Segment retransmission.

(1) シーケンス番号 8 の SACK ブロックを含む ACK を受信した場合，シーケンス番号 8 のセグメントのパスシーケンス番号（p_seq = 5）を参照する．パスシーケンス番号を参照することで，ACK または SACK ブロックによって受信側への到着が確認されていないシーケンス番号 3（p_seq = 2）のセグメントとの間に 3 つ以上のセグメントがあることが分かるので，シーケンス番号 3 のセグメントがロストしたと判断する．

(2) CWND の減少が予想されるため，未送信の予約済みセグメントの予約フラグを消去するとともに，ロストしたと判断されたシーケンス番号 3 のセグメントの予約フラグを消去する☆．

(3) 予約されていない最初のセグメントを示す，予約済みセグメント番号（rsv_nxt）をシーケンス番号 3 のセグメントへ移動する．

(4) セグメントの予約は，予約済みセグメント番号に示されているセグメントから開始されるため，次にセグメントの予約を行うパスによって，シーケンス番号 3 のセグメントが予約される．この際に，予約済みのセグメント内でシーケンス番号に基づきソートを行うことで，再送セグメントが遅延することを防ぐ☆☆．さらに，シーケンス番号 3 のパス番号（path = 1）で示されたパス以外で予約を行うことによって，セグメントロスが発生していないパスを用いてセグメントを再送する．

☆ RTO によってセグメントロスを検出した場合は，送信済みで ACK または SACK ブロックによって受信側への到着が確認されていないすべてのセグメントの予約フラグを消去する．

☆☆ 仮にシーケンス番号 9 のセグメントが送信されていない場合，シーケンス番号 3 のセグメントを先に送出する．

このように，MP-TCP では，pTCP がパスを切り替えての再送を行う際に送信バッファ間でのセグメントの複製/削除といった作業を必要とするのに対して，すべてのパスが送信バッファを共有しているため，予約済みセグメント番号に示されているセグメントを変更するだけで，パスを切り替えてのセグメントの再送を容易に実現することができる．

3.5 他方式との連係

MP-TCP では，TCP/IP ヘッダのみを用いて帯域集約のための制御を行うため，受信側がネットワーク層においてマルチホームを扱うことができる技術を備えている場合，受信側の TCP へ変更を加えることなく帯域集約を実現することができる．たとえば，すでに標準化が終了し導入が進められている MIP を受信側が備えている場合，MIP によって複数のパスから到達したセグメントを，Home Address [1] でまとめることによって，複数のパスを単一の TCP コネクションとして扱う機能が提供される．

送信側に MP-TCP，受信側に MIP を用いて帯域集約を行う場合，新たなパスは Binding Update（BU）[1] によって追加することができる．さらに，送信セグメントへつねに BU を付加することによって，受信側が ACK を返す際の IP アドレスを指定することができるため，複数の ACK パスを利用しての帯域集約を実現することができる☆．しかし，受信側の TCP の変更を行わない場合，3.2.1 項で述べた MP-TCP 独自の SACK ブロックの生成規則は適用されず，RFC2018 に従った通常の SACK ブロックが生成される．さらに，3.3 節で述べた Del-ACK の生成機能についても，受信側の変更に関する機能は提供されない．

4. 性能評価

MP-TCP をシミュレータ [27] 上に実装して評価を行った．**図 6** に，シミュレーションに用いたネットワークモデルを示す．送信側にはマルチホーム端末（src-1）および，通常の端末（src-2，src-3，src-4）を用意し，それぞれを受信端末（sink-1，sink-2，sink-3，sink-4）へ接続した．マルチホームは 3 本のパス（path-1，path-2，path-3）を束ねることにより実現され，各パスの特性を変化させるために受信側のアクセスリンクの帯域幅および遅延をそれぞれ path-1：(1 Mbps，100 ms)，path-2：(2 Mbps，100 ms)，path-3：(5 Mbps，50 ms）とした．また，

☆ 受信側の送信アドレスについては送信側から制御することができないため，パスではなく，セグメントを受信した IP アドレスから SACK パス番号を判断する必要がある．

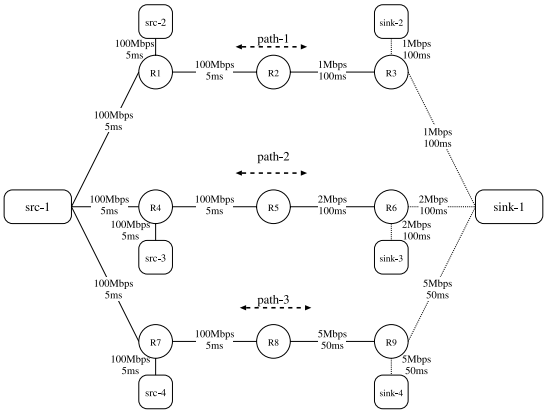

図 6 シミュレーションに用いたネットワークモデル
Fig. 6 Simulation network model.

送信セグメントのサイズは，すべてのパスにおいてパケットが 1,500 Bytes で送信されるように設定し，広告ウィンドウの最大値は通信速度を低下させないよう 1,000 パケットとした．

4.1 帯域集約

最初に帯域集約について評価を行った．シミュレーションは 60 秒間行い，比較方式として src-2〜src-4 へ通常の TCP として FACK [28] を配置した．FACK に関しては，2.1 節で述べたような問題を発生させないために，それぞれを独立して動作させている．src-1 には MP-TCP のほかに mTCP を用意し，すべてのパスの組合せについてスループットの計測を行った．

図 7 に計測された各方式のスループットを示す．FACK と MP-TCP を比較した場合，最も違いが出た場合のスループットは次のようになった．

path-1：MP-TCP（1，3）のときに 0.3%増加
path-2：MP-TCP（2，3）のときに 0.4%減少
path-3：MP-TCP（1，3）のときに 0.2%減少

MP-TCP では，集約される各パスにおいて TCP と同等の輻輳制御を行っているため，理想値となる FACK とほぼ同様なスループットを得ることができる．

FACK と mTCP を比較した場合，最も違いが出た場合のスループットは次のようになった．

path-1：mTCP（1，2，3）のときに 2.5%増加
path-2：mTCP（1，2，3）のときに 4.9%増加
path-3：mTCP（1，2，3）のときに 0.7%減少

mTCP では，path-1 と path-2 において FACK よりもスループットが増加した．これは 2.1 節で述べたように，すべてのパスにおいて ACK がプライマリパ

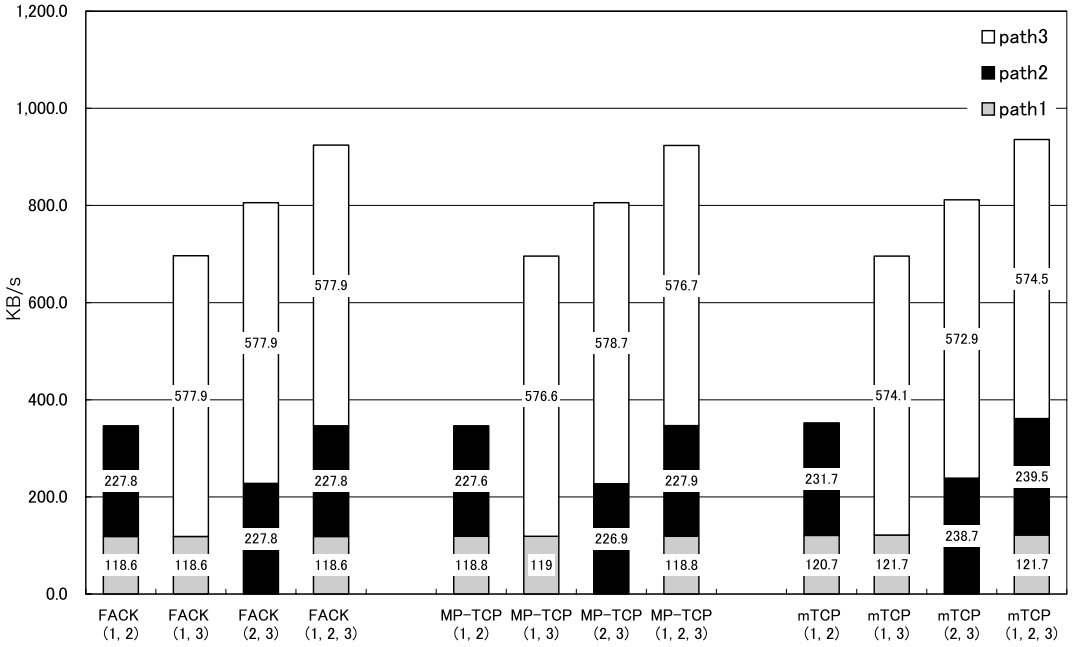

図 7 帯域集約時のスループット
Fig. 7 Throughput of network bandwidth aggregation.

スとなる path-3 で返されるため☆☆，path-1 と path-2 の RTT が通常よりも短くなるためである．

4.2 公 平 性

次に，既存の TCP との公平性についての検証を行うために，path-1〜path-3 をすべて利用して，MP-TCP と FACK，mTCP と FACK をそれぞれ同時に 60 秒間動作させた場合のデータ送信量を計測した．

図 8 に MP-TCP/mTCP と FACK 混在時のデータ送信量を示す．MP-TCP の場合，path-1〜path-3 までのデータ送信量の合計は 26.7 MB となり，FACK のデータ送信量（27.2 MB）と比較した場合，1.8%の減少にとどまった．MP-TCP では複数の ACK パスを利用することによって *self-clock* を保ってのセグメントの送信が行われるため，既存の TCP との公平性を保ちながら帯域集約を実現できていることが分かる．

mTCP の場合，path-1〜path-3 までのデータ送信量の合計は 32.4 MB となり，FACK のデータ送信量（22.7 MB）と比較した場合 42.7%増加する．特に 4.1 節に示したように，path-1，path-2 については mTCP が FACK よりも多くのセグメントを送出するため，MP-TCP の場合と比較すると FACK のデータ送信量は，path-1 では 32.4%，path-2 では 26.5%減少する．mTCP では ACK パスとしてプライマリパスのみを利用するため *self-clock* が崩れてしまい，既存の TCP と同時に用いた場合，公平性を保つことができないことが分かる．

4.3 Del-ACK

送信セグメントに対する ACK 数を計測した結果を**表 1** に示す．計測は 4.1 節のシミュレーション中に行った．

mTCP では送信セグメントに対して，平均で 91.5%の ACK が返された．帯域集約を行った場合は受信側にセグメントが順序どおりに届かないため，通常の Del-ACK では ACK 数をあまり削減することができないことが分かる．

MP-TCP では，すべての場合において送信セグメントに対する ACK 数の割合は 50.1%となった．これは FACK における送信セグメントに対する ACK 数の割合の平均 50.9%とほぼ同等の値となっており，3.3 節で述べた機能によって，帯域集約を行った場合においても ACK 数を削減できていることが分かる．

表 2 に MP-TCP の機能ごとの送信セグメントに対する ACK 数を示す（path-1，2，3 の場合）．3.3 節で述べたように，MP-TCP では，1 つのパスで送信されるセグメントのシーケンス番号が連続するようにセグメントの送信予約を行う送信側の機能と，セグメントの順序違いが発生している最中でも，同一のパス内でシーケンス番号が連続するセグメントを受信した

☆☆ path-1 と path-2 の組合せの場合，プライマリパスは path-2.

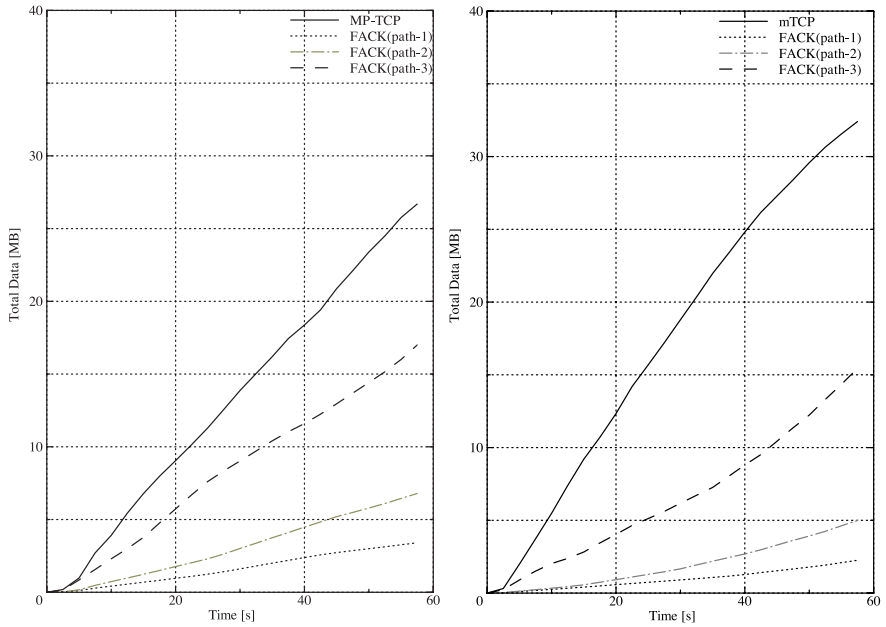

図 8 FACK，MP-TCP/mTCP 混在時のデータ送信量
Fig. 8 Total data of coexistence with MP-TCP/mTCP and FACK.

表 1 送信セグメントに対する ACK 数
Table 1 The number of ACKs to segments.

TCP	セグメント数 ACK 数	ACK の 割合[%]	MP-TCP	セグメント数 ACK 数	ACK の 割合[%]	mTCP	セグメント数 ACK 数	ACK の 割合[%]
(1, 2)	13,817 7,102	51.4	(1, 2)	13,855 6,942	50.1	(1, 2)	14,092 12,519	88.8
(1, 3)	27,859 14,123	50.7	(1, 3)	27,822 13,931	50.1	(1, 3)	27,831 25,915	93.1
(2, 3)	32,186 16,327	50.7	(2, 3)	32,176 16,107	50.1	(2, 3)	32,463 28,553	88.0
(1, 2, 3)	36,931 18,776	50.8	(1, 2, 3)	36,930 18,504	50.1	(1, 2, 3)	37,423 35,914	96.0

表 2 MP-TCP における送信セグメントに対する ACK 数
Table 2 The number of ACKs to segments in MP-TCP.

	MP-TCP	MP-TCP (w/o receiver mod)	MP-TCP (w/o sender mod)
セグメント数	36,930	36,922	37,209
ACK 数	18,504	34,244	20,017
ACK の割合 [%]	50.1	92.7	53.8

場合には Del-ACK を生成する受信側の機能に分けることができる．送信側だけを変更した場合，送信セグメントに対する ACK 数の割合が 92.7%であったのに対して，受信側だけを変更した場合，送信セグメントに対する ACK 数の割合は 53.8%となった．この結果から，ACK 数の削減に対しては受信側の機能が大きく貢献することが分かる．しかし，既存の TCP と同等まで ACK 数を削減するためには，受信側だけではなく，送信側の機能も必要となる．

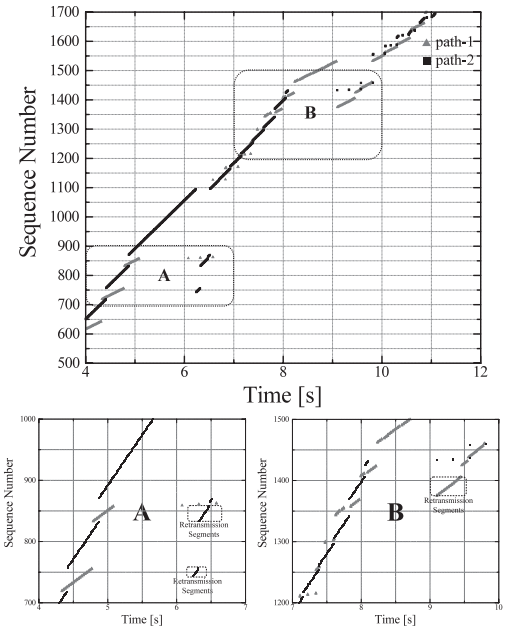

図 9 セグメントの再送
（再送パスの切替えあり）
Fig. 9 Segment retransmission (Path Switching).

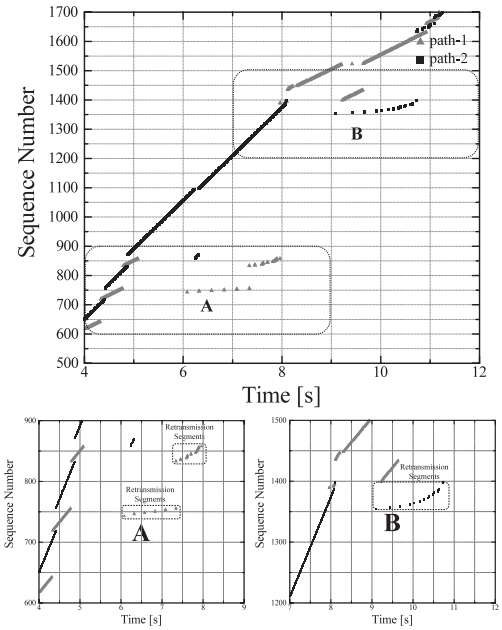

図 10 セグメントの再送
（再送パスの切替えなし）
Fig. 10 Segment retransmission (w/o Path Switching).

4.4 セグメントロス

最後に，セグメントロス発生時の動作について検証を行った．今回のシミュレーションでは path-1，path-2 のみを利用し，受信側のアクセスリンクを切断することでセグメントロスを強制的に発生させた．path-1 は 4.5 秒から 500 ミリ秒間，path-2 は 8.0 秒から 500 ミリ秒間リンク断を発生させ，ロストしたセグメントをロスが発生したパスとは異なるパスを用いて再送する場合と，ロスが発生したパスを用いて再送する場合について計測を行った．

シミュレーション結果を**図 9** と**図 10** に示す．再送パスを切り替えた場合，path-1 が切断されている間にロストしたセグメント（シーケンス番号 743〜756，833〜858）は 6.1 秒に path-1 の RTO によって検出される．ロストしたセグメントは path-2 を用いて 6.2 秒から再送され，6.5 秒までに path-1 でロストしたすべてのセグメントが再送される．path-2 の場合，ロストしたセグメント（シーケンス番号 1375〜1432）は 9.1 秒に path-2 の RTO によって検出される．ロストしたセグメントは 9.1 秒から path-1 で再送され，9.5 秒までにロストしたすべてのセグメントが再送される．

再送パスを切り替えない場合，path-1 が切断されている間にロストしたセグメント（シーケンス番号 743〜756，833〜858）は 6.1 秒に path-1 の RTO によって検出され，ロストしたすべてのセグメントは path-1 を用いて 8.0 秒までに再送される．path-2 の場合，ロストしたセグメント（シーケンス番号 1355〜1398）は 9.1 秒に path-2 の RTO によって検出され，ロストしたすべてのセグメントは path-2 を用いて 10.7 秒までに再送される．

セグメントのロスを経験したパスの CWND は 1 になるため，パスの切替えを行わずにセグメントの再送を行った場合，パスの切替えを行った場合に比べて path-1 のセグメントロスに対しては 1.6 秒，path-2 のセグメントロスに対しては 1.2 秒余計に再送時間を必要とした．ロストしたセグメントをロスが発生したパスとは異なるパスを用いて再送した場合，ロスを経験したパスよりも大きな CWND を持つパスによってセグメントが再送されるため，再送時間が短縮され，ロストしたセグメントによって通信が完了する場合には通信時間についても短縮することが可能となる．

5. ま と め

本稿では，マルチホーム環境を利用して帯域集約を実現するトランスポート層のプロトコルとして MP-TCP の提案を行った．トランスポート層プロトコルとして TCP が用いられた場合，帯域集約を実現するために

は，集約されるパスの RTT が異なることによって，セグメントが受信側に順序どおり届かないことにより Fast Retransmit and Fast Recovery が起動され，スループットが低下する問題を解決する必要がある．

MP-TCP では，集約する各パスごとに TCP Control Block を用意して，複数のパスを独立して管理することによって帯域集約を実現した．さらに Delayed ACK の改良や，複数の ACK パスを用いることによって，帯域集約時においても集約される各パスについて既存の TCP と同等の制御を可能にした．

MP-TCP をコンピュータシミュレーションによって評価した結果，MP-TCP は既存の TCP と同様に ACK 数を 50%程度まで削減できるとともに，既存の TCP との公平性を保って帯域集約を実現できることが示された．今後は実装を行い，TCP Control Block を複数持つことや，送信バッファの改良によるオーバヘッドを明らかにし，新たなトランスポート層プロトコルとしての実現可能性を調査していく予定である．

参 考 文 献

1) Johnson, D., Perkins, D. and Arkko, J.: Mobility Support in IPv6, RFC3775 (June 2004).
2) Fraleigh, C., Moon, S., Lyles, B., Cotton, C., Khan, M., Moll, D., Rockell, R., Seely, T. and Diot, C.: Packet-level traffic measurement from the sprint IP backbone, *IEEE Network Magazine* (Nov. 2003).
3) Nikander, P., Arkko, J. and Henderson, T.: End-Host Mobility and Multi-Homing with the Host Identity Protocol, Internet-Draft, draft-ietf-hip-mm-01 (Feb. 2005).
4) Wright, R. and Stevens, R.: *TCP/IP Illustrated*, Vol.2, Addison Wesley (2001).
5) Postel, J.: Transmission Control Protocol, RFC793 (Sep. 1997).
6) Jacobson, V.: Congestion Avoidance and Control, *Proc. ACM SIGCOMM* (1988).
7) Sivakumar, H., Bailey, S. and Grossman, R.: PSockets: The Case for Application-level Network Striping for Data Intensive Applications using High Speed Wide Area Networks, *Supercomputing* (*SC*), US (Nov. 1999).
8) Hsieh, H., Kim, K. and Sivakumar, R.: An End-to-End Approach for Transparent Mobility across Heterogeneous Wireless Networks, *Mobile Networks and Applications*, Kluwer Academic Publishers (2004).
9) Hsieh, H. and Sivakumar, R.: A Transport Layer Approach for Achieving Aggregate Bandwidths on Multi-homed Mobile Hosts, *ACM MOBICOM* (Sep. 2002).
10) Hacker, T. and Athey, B.: The End-to-End Performance Effects of Parallel TCP Sockets on a Lossy Wide-Area Network, *IEEE IPDPS* (Apr. 2002).
11) Allcock, B., Bester, J., Bresnahan, J., Chervenak, A., Foster, I., Kesselman, C., Meder, S., Nefedova, V., Quesnel, D. and Tuecke, S.: Data management and transfer in high-performance computational grid environments, *Parallel Computing*, Vol.28, No.5 (2002).
12) Zhang, M., Junwen, L., Krishnamurthy, A., Peterson, L. and Wang, R.: A Transport Layer Approach for Improving End-to-End Performance and Robustness Using Redundant Paths, *Proc. USENIX* (June 2004).
13) Snoeren, A. and Balakrishnan, H.: An End-to-End Approach to Host Mobility, *MOBICOM* (Aug. 2000).
14) Stewart, R., et al.: Stream Control Transmission Protocol, RFC2960 (Oct. 2000).
15) Stewart, R., et al.: Stream Control Transmission Protocol (SCTP) Dynamic Address Reconfiguration, Internet-Draft, draft-ietf-tsvwg-addip-sctp-11 (Feb. 2005).
16) Stewart, R., et al.: Sockets API Extensions for Stream Control Transmission Protocol (SCTP), Internet-Draft, draft-ietf-tsvwg-sctpsocket-11.txt (Sep. 2005).
17) Chase, D.: Code combining — A maximum-likelihood decoding approach for combining an arbitrary number of noisy packets, *IEEE Trans. Commun.*, Vol.33 (1985).
18) Koyama, K., Ito, Y., Mineno, H. and Ishihara, S.: Evaluation of Performance of TCP on Mobile IP SHAKE, *IPSJ Journal*, Vol.45 (Oct. 2004).
19) Mathis, M., Semke, J., Mahdavi, T. and Ott, M.: The Macroscopic Behavior of the TCP Congestion Avoidance Algorithm, *ACM Computer Communication Review*, Vol.27 (Nov. July 1997).
20) Mathis, M., Mahdavi, J., Floyd, S. and Romanow, A.: TCP selective acknowledgement options, RFC 2018 (Oct. 1996).
21) Andersen, D., Balakrishnan, H., Kaashoek, F. and Morris, R.: Resilient overlay networks, *Proc. ACM SOSP* (Oct. 2001).
22) Stoica, I., Schenker, S. and Zhang, H.: Core-stateless fair queueing: Achieving approximately bandwidth allocations in high speed networks, *Proc. ACM SIGCOMM* (Sep. 1998).
23) Lin, D. and Morris, R.: Dynamics of random

early detection, *Proc. ACM SIGCOMM* (Sep. 1997).
24) Paxson, V.: End-to-end Internet packet dynamics, *IEEE/ACM Transactions on Networking*, Vol.7 (1997).
25) Allman, M. and Stevens, W.: TCP Congestion Control, RFC 2581 (Apr. 1999).
26) Fall, K. and Floyd, S.: Simulation-based Comparisons of Tahoe, Reno and SACK TCP, *Computer Communication Review*, Vol.26 (July 1996).
27) The Network Simulator, ns-2. http://www.isi.edu/nsnam/ns/
28) Mathis, M. and Mahdavi, J.: Forward Acknowledgment: Refining TCP Congestion Control, *Proc. ACM SIGCOMM* (Aug. 1996).
29) Brakmo, L., O'Malley, S. and Peterson, L.: Vegas: New techniques for congestion detection and avoidance, *Proc. ACM SIGCOMM* (Aug. 1994).

(平成 17 年 5 月 19 日受付)
(平成 17 年 11 月 1 日採録)

五十嵐 健

1998 年慶應義塾大学理工学部計測工学科卒業．2000 年同大学大学院理工学研究科修士課程修了．同年 NTT DoCoMo に入社．以来，モビリティマネジメントおよびトランスポート層プロトコルに関する研究に従事．2004 年より慶應義塾大学理工学研究科博士課程に在籍．

重野 寛（正会員）

1990 年慶應義塾大学理工学部計測工学科卒業．1997 年同大学大学院理工学研究科博士課程修了．1998 年同大学理工学部情報工学科助手（有期）．現在，同大学理工学部情報工学科助教授．工学博士．計算機ネットワーク・プロトコル，モバイル・コンピューティング，マルチメディア・アプリケーション等の研究に従事．著書『ネットワーク・ユーザのための一無線 LAN 技術講座』（ソフト・リサーチ・センター），『コンピュータネットワーク』（オーム社）等．電子情報通信学会，IEEE，ACM 各会員．

井原 武

1990 年東京大学工学部機械工学科卒業．1992 年同大学大学院修士課程修了．同年日本電信電話株式会社通信網総合研究所に入所．以来，PHS 交換システム，ATM 交換システムの研究開発，ATM 信号方式の国際標準化活動に従事．2001 年に NTT DoCoMo に転籍．以来，ネットワーク研究所にて，次世代移動通信システムにおける移動管理技術，トランスポート技術，トラフィック制御技術の研究に従事．

三浦 章

1976 年東京工業大学理学部情報科学科卒業．1979 年同大学大学院修士課程修了．同年日本電信電話公社電気通信研究所入所．以来，ファクシミリ蓄積交換システム，パケット・回線複合交換機，インテリジェントネットワーク将来方式等，インテリジェントネットワーク用サービス制御ノードの研究開発に従事．2000 年に NTTDoCoMo に転籍．以来，PDC-P ネットワーク開発，次世代移動通信方式およびそれにかかわるトラフィック研究に従事．

岡田 謙一（フェロー）

慶應義塾大学理工学部情報工学科教授，工学博士．専門は，CSCW，グループウェア，ヒューマン・コンピュータ・インタラクション．『ヒューマンコンピュータインタラクション』（オーム社），『コラボレーションとコミュニケーション』（共立出版）をはじめ著書多数．情報処理学会誌編集主査，論文誌編集主査，GW 研究会主査等を歴任．現在，情報処理学会 MBL 研究会運営委員，BCC 研究グループ幹事，日本 VR 学会 CS 研究会副委員長．情報処理学会論文賞（1996 年，2001 年），情報処理学会 40 周年記念論文賞，日本 VR 学会サイバースペース研究賞，IEEE SAINT '04 最優秀論文賞を受賞．情報処理学会フェロー，IEEE，ACM，電子情報通信学会，人工知能学会会員．