社団法人 情報処理学会　研究報告
IPSJ SIG Technical Report

2004－MBL－28　(27)
2004／3／5

TCPを拡張した複数経路通信における再送制御に関する検討

殿内雅晴[†1]　峰野博史[†2]　石原進[†3]
高橋修[†4]　水野忠則[†2]

筆者らはこれまでに性能や品質の異なる複数の経路を利用する通信方式において，TCP を拡張することで適切な輻輳制御や再送制御を行うプロトコル DMTCP(Dynamic Multipath TCP) を提案している．しかしこれまでの検討では再送時の DMTCP の振る舞いを考慮していなかった．

本稿では，経路ごとに再送タイマと輻輳ウインドウと送信セグメントのリストを設け，最初に送信した経路以外にもう 1 本経路を使用して再送を行う方式を提案している．また再送が失敗した場合にも通信が破綻しないような手法についても説明する．さらに提案した方式に関して机上でシミュレーションを行い有効性や信頼性を確認する．

A Study on retransmission control of multipath-extended TCP

Masaharu Tonouchi,[†1] Hiroshi Mineno,[†2] Susumu Ishihara,[†3]
Osamu Takahashi[†4] and Tadanori Mizuno [†2]

We have proposed DMTCP(Dynamic Multipath TCP) that controls congestion and retransmission in multipath transmission by extended TCP. But we have not consider retransmission control.

In this paper, we propose retransmission control method that has retransmission timer, congestion window, transmitted segments list for every path and retransmits using one more path in addition to the path transmitted first. We illustrate this method keeps transmission when it fails retransmission. In addition, We check effectivity and reliability of this method by simulating on paper.

1. はじめに

近年，無線通信インフラの整備に伴い，携帯電話やPHS などモバイル端末を用いた無線通信で大容量のコンテンツを鑑賞することが可能となっている．しかし，モバイル端末を用いた無線通信には，通信速度が遅い，信頼性が低い，通信が不安定であるなどの問題が存在している．有線通信においても DSL・FTTH などによって広帯域化が進む一方で，科学技術研究の分野やグリッドコンピューティングではテラバイトを超える大容量のデータ転送を行いたいという要求が生じており，ネットワークの信頼性やスループットのさらなる向上が期待されている．

†1 静岡大学大学院情報学研究科
Graduate School of Information, Shizuoka University
†2 静岡大学情報学部
Faculty of Information, Shizuoka University
†3 静岡大学工学部
Faculty of Engineering, Shizuoka University
†4 NTT ドコモ
NTT docomo

このような要求を満足させるために，複数経路を用いた通信の研究が注目を集めている．例えば，複数のインタフェースでネットワークに接続するマルチホーミングや，複数の移動端末が集まり各端末が持つ回線を共有する通信回線共有方式 SHAKE(SHAring multiple paths procedure for cluster networK Environment)[1] が挙げられる．

筆者らはマルチホーミングや SHAKE のような環境を想定し，トランスポート層で複数経路通信に適した輻輳制御や再送制御を行う方式として Dynamic Multi Link TCP(以下 DMTCP とする) を提案している．筆者らはこれまで DMTCP の機能の設計と，シミュレーションモデルのプロトタイプを用いて DMTCP の輻輳制御機構の簡易評価を行っている[2]．本稿では送信セグメントを喪失した時に DMTCP が行う再送制御方式を提案し，その有効性について検討する．

2. DMTCP

2.1 複数経路を用いた通信方式

複数の経路を用いた通信方式として，インターネッ

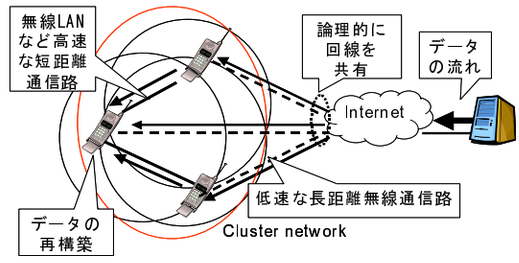

図 1 通信回線共有方式

トを用いた複数経路データ伝送方式[3] がある．これはインターネットにおいてネットワーク障害に対する信頼性やスループットの向上を目的とし，専用ゲートウェイを用いて複数経路に IP トンネリングを行うことで実現されている．この方式ではトランスポート層での送信制御までは提案されていない．MulTCP[4] では大容量のデータ転送を行う際に，輻輳ウインドウサイズの極端な変化を抑え安定した送信を行うことを目的に，トランスポート層において一つのコネクションの中に複数のサブコネクションを用いる通信に適応した送信制御が提案されている．しかしこの方式は複数経路の通信を想定したものではない．通信中の経路の増減や各サブコネクションの経路品質の差を考慮に入れた送信制御などは提案されておらず，複数経路通信の制御には適していない．

一方無線による複数経路を束ねて通信する方式として SHAKE がある．SHAKE では，複数の移動端末が短距離高品質リンクを用いて相互に接続し，一時的にネットワークを構成することで (これをクラスタネットワークと呼ぶ)，ある端末がクラスタ外のホストと通信する時は，それ自身が持つ通信路のみならずクラスタ内の他の端末の外部への通信路も用いて通信を行う方式である (**図 1**)．無線を利用した通信は，その特性上，有線通信に比べて帯域が狭く，また伝送誤りが生じる割合が高いため，バースト的なデータロス，スループットの低下が起こるという課題も存在するが，SHAKE では，通信の高速化を実現するだけでなく，品質の変化しやすい無線通信においても代替経路を利用することで信頼性を向上させることができる．

現在様々な階層からのアプローチでこの SHAKE の研究が進められており，トランスポート層において SHAKE に適した送信制御を行う方式[5] も提案されているが，SHAKE 以外の通信方式や通信途中の経路数の増減への対応は考えられていない．

DMTCP はマルチホーミングなど SHAKE 以外の通信方式において性能や品質の異なる通信に適した送信制御を行い，これまで検討されてこなかった通信途中の経路数の増減への対応も取り入れている．

2.2 DMTCP の概要

DMTCP では一つのコネクションで大容量のデータ転送を行う場合での利用を想定している．例えば，FTP や HTTP や P2P のアプリケーションで大容量のデータをダウンロードする場合などが考えられる．このような場合に，DMTCP はエンドエンド間の複数経路を一つのコネクションとみなし，経路が複数あることを活かして通信の効率を向上させるために，以下のような送信制御を行う．

経路ごとの輻輳制御によるトラフィック分配

TCP における輻輳ウインドウとは，通信経路の輻輳状態に応じて送信可能なデータ量を調整する機構である．通信経路が複数あるにも関わらず輻輳制御をまとめて行った場合，帯域やロス率など各経路の品質に応じた効率的な輻輳制御を行うことはできない．例えば一つの経路で輻輳が起きた場合には全ての経路で送信量を落とすことになってしまう．DMTCP では経路ごとに輻輳ウインドウと再送タイマを設置することにより，この輻輳制御を経路ごとに行う．これによって送信エラーの起きた経路では輻輳ウインドウを縮小し送信量を減らし，順調に確認応答が返ってくる経路では輻輳ウインドウを増加させ送信量を増やす．これは送信セグメントを性質の異なる経路に適切に振り分けることになる．DMTCP では輻輳制御の仕組みを利用して各経路の状況に応じた動的なトラフィック分配を実現する．

複数経路を活用した再送制御

送信エラーが輻輳によるものであれば，その経路で再送を行っても再び送信エラーを起こしてしまう可能性は非常に高い．複数経路を用いた通信であっても，経路ごとにコネクションを確立した場合は再送制御は経路ごとになってしまい，輻輳を起こした経路で再送を行うことになってしまう．DMTCP では，エンドエンド間で一つのコネクションを形成することにより，元々送信した経路以外の経路を用いて再送を行う．これによって信頼性の高い再送制御を実現する．

ネットワーク層とのインタフェース

DMTCP では，複数の経路を明示的に使い分けることのできるように拡張したネットワーク層を前提としている．1 台の端末が複数のネットワークインタフェースで異なる経路を使う通信方式であれば，経路とネットワークインタフェースの対応付けを行う機能があればよく，SHAKE のようにクラスタ内の他の端末の通信路を利用する通信方式であれば，経路と中継ホスト

の対応付けを行う機能が必要となる．

3. DMTCP のウインドウ制御

3.1 DMTCP の輻輳制御

DMTCP で経路ごとに行う輻輳制御では，現在の TCP で広く普及している TCP Reno[6] の輻輳制御アルゴリズムを採用する．さらに，セグメント喪失の検出方法も TCP Reno と同様に再送タイムアウトと重複確認応答を用いて行う．またこの輻輳ウインドウや 3.3 章で述べる送信セグメントリストなどの経路ごとに必要なデータ構造は，全て PCB(Path Control Block) と呼ばれるデータ領域に格納する．経路ごとに行う輻輳制御のアルゴリズムについて以下で説明する．

データ転送開始時

データ転送の開始時には，各経路の輻輳ウインドウの大きさを 1 に設定し，スロースタートアルゴリズムにより輻輳ウインドウの値を指数関数的に増加させていく．各スロースタート閾値には，この TCP コネクションで許容できる最大の値が設定される．

再送タイムアウト

再送タイムアウトによりセグメント喪失を検出した経路は，スロースタート閾値を現在の輻輳ウインドウの大きさの 1/2 に設定し，輻輳ウインドウの大きさを 1 まで減少させて，スロースタート段階へ移行する．

高速リカバリアルゴリズム

重複確認応答によってセグメント喪失を検出した場合は，その経路の輻輳ウインドウとスロースタート閾値の大きさを現在の輻輳ウインドウの大きさの 1/2 に設定する．セグメント喪失の検出後に行う再送制御は 3.3 で述べる．

3.2 ローカルシーケンス番号

輻輳制御と再送制御を各経路で独立に行うには，どの経路でどのセグメントの喪失があったのかを知る必要がある．そこで，DMTCP では送信セグメントに対し経路ごとにシーケンス番号を付ける．これをローカルシーケンス番号と呼ぶ．また DMTCP は TCP を拡張したプロトコルであり，TCP で行うウインドウ制御も同時に行う．TCP で用いられる通常のシーケンス番号を，ローカルシーケンス番号と区別するためにグローバルシーケンス番号と呼び併用する．ローカルシーケンス番号を用いることで，各経路でセグメントが順序通り受信されているか，どのセグメントが喪失したかを把握することが可能となる．

DMTCP におけるセグメントのヘッダはローカルシーケンス番号を管理するために TCP ヘッダに経路 ID，再送経路 ID，ローカル ACK の情報を追加する．この情報は TCP ヘッダのオプションフィールドに，12 バイトの長さを持つ DMTCP オプションとして設定することで，TCP との互換性を持たせている (**図 2**)．種類と長さのフィールドは TCP ヘッダのオプションとして必要なもので，このヘッダが DMTCP ヘッダオプションであることを示す．経路 ID は送信セグメントがどの経路で送信され，どの経路の送信ウインドウにより管理されているかを示す．再送経路 ID は，再送の際に再送セグメントが実際にはどの経路で送信されたかを通知するためのものである．DMTCP ではセグメントを受信した経路を使って ACK を返すことで，各経路の RTT を測定する．また送信側は ACK が返ってくることによりその経路が通信可能であることを知ることができる．再送時も同様に再送セグメントを受信した経路で ACK を返す．前述したように DMTCP では再送時に最初に使用した経路とは異なる経路で再送することが可能である．受信側の DMTCP はこのフィールドを見ることによって，実際に再送に使われた経路で ACK を返すことができる．また，再送時にはこのオプションに加えて再送経路のローカルシーケンス番号とローカル ACK の情報も追加する．これについては次項で詳細を述べる．

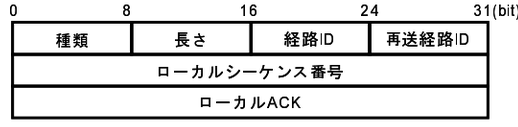

図 2 DMTCP ヘッダオプション

3.3 DMTCP の再送制御

DMTCP のセグメント喪失の検出方法は TCP Reno と同様に，タイムアウトと重複確認応答の 2 通りある．タイムアウトによりセグメントの喪失を検出した場合は，コネクションが現在使用している複数の経路のうち 2 本の経路を併用して再送を行う．重複確認応答によってロスを検出した場合は，ロス発生後に 3 つ以上のセグメントが受信側ホストに到着しているため，輻輳の度合いはさほど大きくないものと見ることができる．この考え方に基づき，重複確認応答によってロスを検出した場合はタイムアウト発生時のように 2 本の経路を使用せず，ロスを起こした経路だけで再送を行う．一方タイムアウト発生時には重度の輻輳が発生している考えられるため，ロスを実際に起こした経路に加え，その時点で輻輳ウインドウサイズがスロースタート閾値より大きく，かつ輻輳ウインドウの送信可能なサイズが最大の経路も併用して再送を行う．輻輳

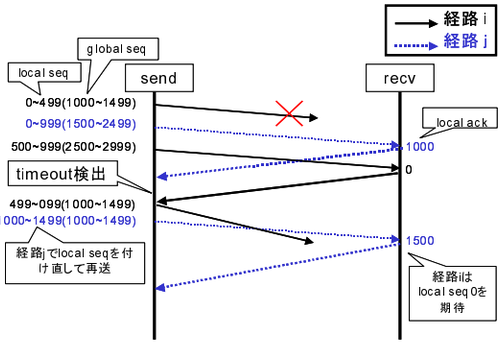

図 3 再送時ローカルシーケンス番号を新たに付け直した場合

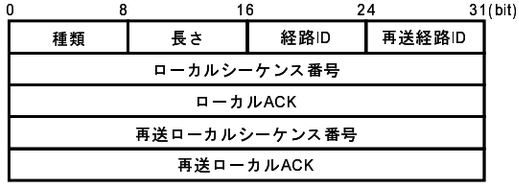

図 4 DMTCP 再送ヘッダオプション

ウインドウサイズがスロースタート閾値より大きいということは，経路の状況が安定しているとと解釈できる．また，輻輳ウインドウの送信可能なサイズが最大の経路を使う理由は，この経路は新たにセグメントを送信できるネットワーク上の許容量に最も余裕があると考えられるからである．利用可能な全ての経路を用いて再送する方法も考えられるが，トラフィックの無駄な増加を抑えるために DMTCP ではこの 2 本のみを用いて再送する．

ここで，タイムアウトの発生した経路を i とし，経路 i と共に再送を行うもう片方の経路を j とする．PCB 内のフラグ変数 rexmt は，これが立っていると他の経路で再送して欲しい状態であることを示す．経路 i でタイムアウトが発生したら rexmt[i] を立て，経路 i で次に ACK が返ってくるか，経路 i の再送が完了するまで立ち続ける．

再送時に最初に使用した経路とは異なる経路で送信した場合，再送セグメントのローカルシーケンス番号を新たに付け直してしまうと問題が発生する．受信ホストは最初の経路上の欠けたローカルシーケンス番号を持ったセグメントを待ち続けてしまうのである．この様子を**図 3** に示す．この例では，経路 i の再送が再び失敗した場合，経路 j で再送したグローバルシーケンス番号 1000〜1499 を経路 i は待ち続けることになる．一方で，経路 j で再送したセグメントに対して経路 j のローカルシーケンス番号を付けない場

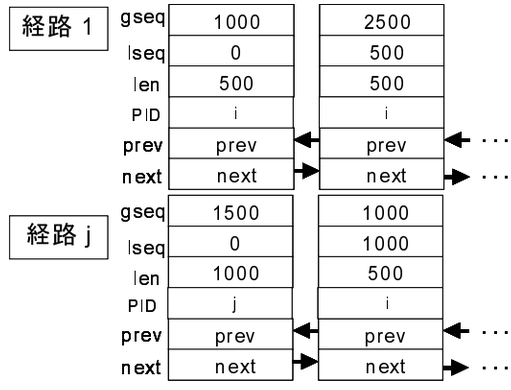

図 5 送信セグメントリストの例

合，経路 j は再送セグメントに対する輻輳制御・再送制御を行うことができなくなるという問題が発生する．経路 j の再送において，DMTCP はその時点で送信可能なサイズ一杯の再送セグメントを経路 j へ流す．この再送セグメントを無視すると，経路 j の輻輳制御において推測されるネットワーク能力の限界を超えた負担が掛かるため，輻輳を引き起こしてしまう．このため，経路 j の再送セグメントに対しても輻輳制御から外すわけにはいかない．そこで経路 j における再送セグメントのヘッダには，最初に使用した経路の経路 ID とローカルシーケンス番号を付け，さらに再送経路 ID とローカルシーケンス番号を付けることで問題を解決する．経路 i で行う再送には図 2 で示した DMTCP ヘッダを用い，経路 j で行う再送には**図 4** で示す DMTCP 再送ヘッダを用いる．DMTCP 再送ヘッダには DMTCP ヘッダに加えて，再送ローカルシーケンス番号として経路 j のローカルシーケンス番号，再送ローカル ACK として経路 j のローカル ACK の情報を追加する．DMTCP 再送ヘッダの長さは 20 バイトとなる．

再送ヘッダに経路 i の初回の送信で使ったローカルシーケンス番号を乗せるには，経路 i でどのセグメントを送信し，ローカルシーケンス番号はいくつであったか知っていなくてはならない．そこで，送信ホストは送信セグメントのローカルシーケンス番号とグローバルシーケンス番号を対応付けしたリストを経路ごとに保持し，再送に備える．これによって再送時には初回の送信時と同じローカルシーケンス番号とグローバルシーケンス番号の組み合わせで送信することができる．送信セグメントリストの例を**図 5** に示す．送信済みセグメントブロックのグローバルシーケンス番号の先頭を表す Gseq，それに対応するローカルシーケンス番号の先頭を表す Lseq，このブロックのデータ長

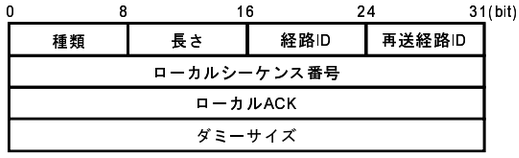

図 6 DMTCP 再再送ヘッダオプション

を表す len，このブロックを最初に送信した経路の経路 ID を表す PID，をメンバとしてこのリストは持つ．送信済みセグメントブロックとは，連続したグローバルシーケンス番号を持った送信済みセグメントの集まりである．この例では経路 i でグローバルシーケンス番号 1000〜1499 と 2500〜2999 という 2 つのセグメントブロックを送信し，経路 j では 1500〜2499 のセグメントブロックと経路 i で送信したグローバルシーケンス番号 1000〜1499 の再送を行っている．このリストは確認応答されたセグメントから削除されるため通信終了時には全てのリストは削除される．

また，経路 j の再送セグメントが再送タイムアウトによって喪失した場合，経路 j で再送セグメントを再再送することはしない．これは，元々経路 j は信頼性が高くネットワークの許容量に余裕があるから経路 i の再送を補助していたのであって，経路 j で輻輳が起きた場合は他の経路のセグメントを再送する余裕が無いからである．経路 j は送信セグメントリストの PID を見ることで，セグメントブロックが経路 i の再送セグメントかどうか判断することができる．図 5 では経路 j の 2 つ目のセグメントブロックの PID が i になっている．これは，このセグメントブロックが元は経路 i で送信されたものであることを意味しているため，もしこのセグメントブロックが再送タイムアウトを起こしても経路 j は再送を行わない．経路 j は再再送を行わないが，再送セグメントに対しても送信制御を行っているため経路 j の受信側は再送セグメントを待ち続けてしまう．そこで DMTCP では経路 j で再送セグメントが再送タイムアウトによって喪失した場合は，再送セグメントを送信リストから切り捨て，輻輳ウインドウも再送セグメントの分だけ進める．さらに，再送セグメントの切り捨てを経路 j の受信側にも知らせるために，次に送信するセグメントのヘッダに，切り捨てる再送セグメントのサイズの情報を載せる．このヘッダを DMTCP 再再送ヘッダと呼び，**図 6** に示す．このヘッダのダミーサイズフィールドに切り捨てる再送セグメントサイズを格納する．

表 1 再送ケース一覧

	Gseq1 送信	Gseq1 再送	ACK
Case1			
Path i	×	○	○
Path j	-	○	○
Case2			
Path i	×	○	○
Path j	-	○	×
Case3			
Path i	×	○	○
Path j	-	×	-
Case4			
Path i	×	○	×
Path j	-	○	○
Case5			
Path i	×	○	×
Path j	-	○	×
Case6			
Path i	×	○	×
Path j	-	×	-
Case7			
Path i	×	×	-
Path j	-	○	○
Case8			
Path i	×	×	-
Path j	-	○	×
Case9			
Path i	×	×	-
Path j	-	×	-

4. 再送のシナリオ

ここでは前章で提案した DMTCP の再送制御方式について，通信が途中で破綻せずに再送を完了することを示すために机上でシミュレーションを行う．今，利用可能な経路が i と j の 2 本あるとする．最初送信側ホストは経路 i を使ってグローバルシーケンス番号 (以下 Gseq)1 を送信し，そこでタイムアウトが発生し Gseq1 を喪失する．次に経路 i と経路 j を使って Gseq1 を再送する．受信側ホストはどちらかの経路から再送セグメントを受け取ることができればそれに対して ACK を送信側ホストに返す．この時の起こりうる状況をまとめると表 1 のようになる．以下で，全ての送信が成功した場合，途中でセグメントが失われて再送が完了しない場合，一つの経路が完全にダウンしてしまった場合の代表的なケースについて説明する．

4.1 Case1

Case1 は経路 i，j ともに再送が成功した場合である．この時のシーケンスを**図 7** に示し，DMTCP の再送制御の手順について以下で述べる．また初期状態は次のようになっている．輻輳ウインドウに関して，経路 i，j ともに輻輳ウインドウサイズ cwnd=1，送信

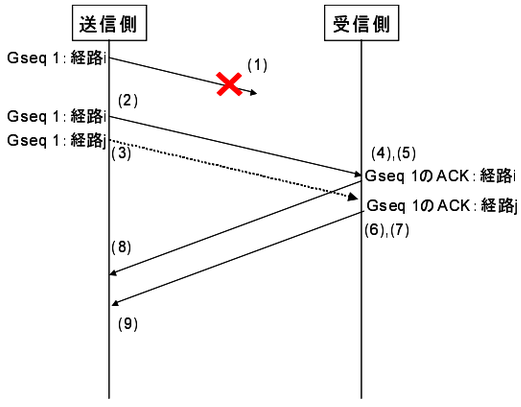

図 7 Case1

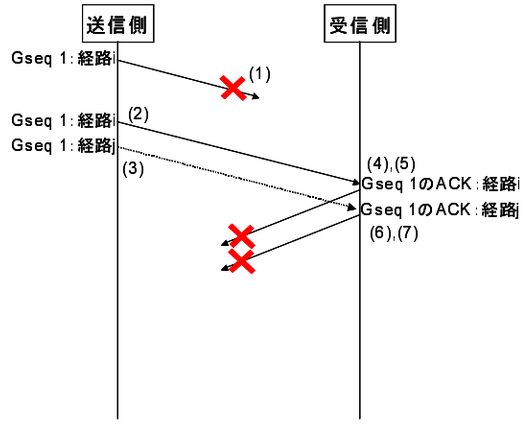

図 8 Case5

したうち受信確認応答がとれていない最小のローカルシーケンス番号 una=1，次に送信する予定のローカルシーケンス番号 nxt=2，送信したうち最大のローカルシーケンス番号の次の番号を示す max=2，次に受信を期待するローカルシーケンス番号 rcv_nxt=1 となっている．経路 i の cwnd であれば cwnd[i] のように表すことにする．経路 i は次に送信する予定のローカルシーケンス番号を示す nxt の他に，経路 j で次に送信するローカルシーケンス番号を示す変数 rexnxt[i] を持っており，una[i] と等しく rexnxt[i]=1 となっている．また経路 i の送信セグメントリストには Gseq=1, Lseq=1，len=1 というブロックが登録されている．

(1) 経路 i で再送タイムアウト発生を検出
rexmt[i]=1 として，経路 i で再送が必要な状態だということを示す．タイムアウトが発生したので cwnd[i]=1 とし，nxt[i] は una[i] まで戻して nxt[i]=1 とする．経路 j で次に再送するセグメントを示す rexnxt[i] も una[i] まで戻して rexnxt[i]=1 とする．

(2) 経路 i で Gseq1 を再送
この時の DMTCP ヘッダは経路 ID=i，再送経路 ID=i，Gseq=1，Lseq=1 となっている．nxt[i] を進めて nxt[i]=2．経路 i の再送タイマをセットする．セットする値は RTT は関係なく，前回の値を 2 倍したものになる．

(3) 経路 j で Gseq1 を再送
rexnxt[i]=1 を見て再送するセグメントを決める．DMTCP ヘッダは経路 ID=i，再送経路 ID=j，Gseq=1，Lseq=1，rLseq=1 となる．Lseq は i のローカルシーケンス番号，rLseq は j のローカルシーケンス番号を表している．nxt[j] を進めて nxt[j]=2．j の送信リストは Gseq=1,Lseq=1，len=1，PID=i というブロックを新たに登録．j の再送タイマは，現在停止しているならセットし直し，他のセグメントを計測中ならセットし直さない．

(4) 経路 i で Gseq1 を受信
rcvnxt[i]=2 となる．

(5) 経路 i で ACK を送信
この時の DMTCP ヘッダは経路 ID=i，再送経路 ID=i，Gack=2，Lack=2 となっている．

(6) 経路 j で Gseq1 を受信
rcv_nxt[j]=2 となる．

(7) 経路 j で ACK を送信
DMTCP ヘッダは経路 ID=i，再送経路 PID=j，Gack=2，Lack=2，rLack=2 となっている．Lack は経路 i のローカルシーケンス番号，rLack は経路 j のローカルシーケンス番号を表している．

(8) 経路 i で ACK を受信
経路 i の輻輳ウインドウサイズはスロースタート状態なので cwnd[i]=2．una[i]=2，nxt[i]=2，max[i]=2．経路 i の送信リストから Gseq1 のブロックを消す．経路 i で ACK が返ってきたので rexmt[i]=0 にして，他の経路での再送を中止する．rex_nxt[i]=2 とする．

(9) 経路 j で ACK を受信
経路 j の輻輳ウインドウサイズは輻輳回避状態なので cwnd[j]=2．una[j]=2，nxt[j]=2，max[j]=2
経路 j の送信リストから Gseq1 のブロックを消す．これで経路 i，j ともに再送を完了する．

4.2 Case5

この時のシーケンス図を**図 8** に示す．これは初期

状態から (7) までは Case1 と同じである．このケースでは経路 i,j ともに再送セグメントが到着しているが，どちらもそれに対する ACK が返ってこないためまだ再送が完了していない状態である．ここからさらに Case5.1 と Case5.2 の 2 つの状態に分けることができる．

4.2.1 Case5.1

経路 j の輻輳ウインドウはスロースタート閾値以上あり，Gseq1 以外にもセグメントを送信しているため，Gseq1 に対する ACK 以外の ACK が返ってくる可能性がある．このケースでは経路 j から Gseq1 に対する ACK 以外の ACK が返ってくる場合である．この ACK は Gseq1 は既に到着済みであることを示すため，ここで Gseq1 の再送は完了する．

4.2.2 Case5.2

Case5.1 と異なり，経路 j からも ACK が一つも返ってこない場合である．この時，経路 j も再送タイムアウトが発生していることになる．この時の処理を**図 8** の (7) の続きから以下に示す．

(8) 経路 j で再送タイムアウト発生を検出
経路 j でタイムアウトが発生したので cwnd[j]=1 まで減少させる．cwnd[j] はスロースタート閾値より小さくなったので，経路 j は経路 i の再送をもう手伝わない．経路 i は経路 j 以外に，cwnd がスロースタート閾値以上の経路を探し，もし見つかればその経路と一緒に Gseq1 を再再送する．
また経路 j の送信リストには Gseq1 が登録されているが，これは元は経路 i の再送セグメントである．経路 j ではこのセグメントの再再送を行わない．送信リストから Gseq1 を切り捨て，nxt[j]=2 まで進める．次に経路 j で送信するセグメントのヘッダには Gseq1 を切り捨てた分のダミーサイズの情報が追加されることになる．

4.3 Case7 の後経路 i がダウンした場合

この時のシーケンス図を**図 9** に示す．これは (5) の経路 j で ACK を送信するところまで Case1 と同じである．このケースでは経路 j では Gseq1 の再送に成功しているが，経路 i はダウンしてしまいその後何も送信できなくなっている．(6) からの処理を以下で示す．

(6) 経路 j で ACK を受信
ここで Gseq1 の再送は完了する．経路 j は輻輳回避状態なので cwnd[j]=2．una[j]=2，nxt[j]=2，max[j]=2 とする．経路 i の輻輳ウインドウサイズは増やさずにウインドウを右に一つずらし，una[i]=2，nxt[i]=2，max[i]=2．経路 i と経路 j の送信リストから Gseq1 のブロックを消す．rexnxt[i] もひとつずらし rexnxt[i]=2 とする．

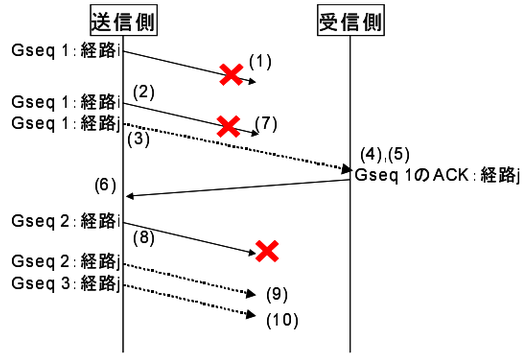

図 9 Case7 の後経路 i がダウンした場合

(7) 経路 i でタイムアウト発生し Gseq1 を喪失
タイムアウト発生により cwnd[i]=1 に減少．

(8) 経路 i で Gseq2 を送信
cwnd[i]=1，nxt[i]=2 となっているので Gseq2 を送信．もう再送するセグメントは無いが，再送が必要な状態であることを示す rexmt はそのままにしておく．

(9) 経路 j で Gseq2 を再送
rexmt[i]=1 なので，rexnxt[i]=2 を見て Gseq2 を送信．

(10) 経路 j で Gseq3 を送信
経路 j の輻輳ウインドウにまだ空きがあるので，Gseq3 を送信．

この後も経路 i で ACK が返ってくるまでは，経路 j でも経路 i と同じセグメントを同時に送信する．また経路 i でタイムアウトがある設定回数以上発生した場合，経路 i は使用不能になったと判断し，以後は使用しない．

この章では，提案した DMTCP の再送制御方式について有効性と信頼性を確認するために机上でシミュレーションを行い，全ての送信が成功した場合，2 本の経路で行った再送が両方失敗し再送が完了しない場合，一つの経路が完全にダウンしてしまった場合の代表的な 3 つのケースについて説明した．この結果，提案した再送制御方式を用いた通信が途中で破綻しないことを示した．

5. ま と め

本稿では，複数経路通信において各経路にウインドウ，再送タイマ，送信セグメントリストを設け，再送時には 2 本の経路を用いてセグメントを送信する方式を提案した．そして提案した再送制御方式の有効性と信頼性を机上シミュレーションで確認した．現在，提案した DMTCP の再送制御機構の部分をこれまでに作成した DMTCP のシミュレーションモデルに組み込んでおり，今後は検討した再送制御方式を組み込んだ DMTCP の性能評価を行う予定である．

参 考 文 献

1) 峰野, 青野, 太田, 井手口, 水野, “クラスタ型ネットワークにおける通信回線共有方式の提案と評価”, 情報処理学会論文誌, Vol.41, No.2, pp.354–362, 2000.
2) 殿内，鄭，峰野，石原，水野,“ 複数経路を用いた通信のためのトランスポート制御機構の評価 ”, 情報学ワークショップ 2003.
3) 林, 山崎, 森田, 相田, 武市, 土居, “インターネットを用いた複数経路データ伝送方式の性能評価”, 電子情報通信学会論文誌, VolJ84-B, No.3, 2001.
4) P.Gevros,F.Risso and P.kirstein,“Analysis of a Method for Differential TCP Service”, GLOBECOM’99, Dec. 1999.
5) 飯田，石原，水野,“ 複数無線リンク上でのコネクション型通信手法の性能評価 ”, 情報処理学会研究報告 2001–DPS–102, Vol.2001, No.29, マルチメディア通信と分散処理 102–2, pp.127–132, 2001.
6) Allman, M., V. Paxson, W.Stevens, “TCP Congestion Control”, RFC2581, April 1999.