

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

殿内雅晴[†] 鄭宇新[†] 峰野博史^{††} 石原進^{†††} 水野忠則^{††}

[†] 静岡大学大学院情報学研究科 ^{††} 静岡大学情報学部 ^{†††} 静岡大学工学部

Evaluation of transport control method for using multiple paths

Masaharu Tonouchi[†] Ushin Tei[†] Hiroshi Mineno^{††} Susumu Ishihara^{†††}
Tadanori Mizuno^{††}

[†] Graduate School of Information, Shizuoka University ^{††} Faculty of Information, Shizuoka University
^{†††} Faculty of Engineering, Shizuoka University

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。

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現在さかんに行われている。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。

本稿では、性能や品質の異なる複数の経路を利用し通信の高速高信頼化を目指す通信方式において、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 Dynamic Multi Link TCP(以下 DMTCP とする) の機能の詳細な設計とこれまでに開発したシミュレーションモデルのプロトタイプを用いて DMTCP の効果について簡易評価する。

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、インターネットを用いた複数経路データ伝送方式 [2] 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。この方法では特別に専用ゲートウェイを設置する必要がある上にトランスポート層での送信制御までは考慮されていない。MulTCP[3] では大容量のデータ転送を行う際に、輻輳ウィ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、一つの接続の中にも複数のサブ接続を用いる通信に適した送信制御が提案されている。しかしこの方法は複数経路の通信を想定し

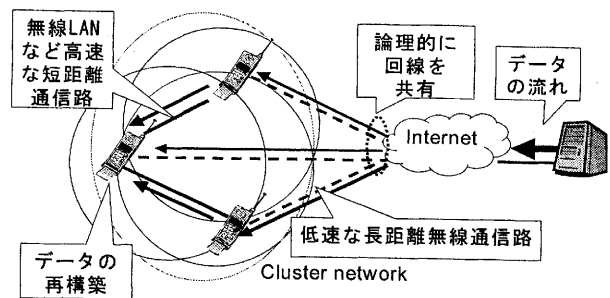


図 1: 通信回線共有方式

たものではなく、通信中の経路の増減や、各サブ接続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った課題も存在するが、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いる。トランスポート層において TCP の機能を拡張し、SHAKE に適したものに拡張したプロトコルとして筆者らは TCP SHAKE[4] を提案している。トランスポート層では通信の信頼性の保証と、再送制御や輻輳制御などの送信制御が行われている。複数経路通信の送信制御をトランスポート層で行うことで、経路の状況に応じた柔軟で動的なトラフィック分配や、どの経路でどれだけのデータを喪失したかを判断した上での再送が可能になる。TCP SHAKE ではトランスポート層において SHAKE に適した送信制御を行うことによる効率的な通

信手法を提案している。DMTCP は TCP SHAKE に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を取り入れ、マルチホーミングなど SHAKE 以外の通信方式の環境も想定したものである。

2.2 DMTCP の特徴

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できればよく、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。

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 における輻輳ウィンドウとは、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場合に輻輳ウィンドウが一つでは、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る。DMTCP では経路ごとに輻輳ウィンドウ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ィ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ィ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 複数の経路を利用した通信の場合、送信エラーの発生により再送を行う際に、送信エラーの発生した経路で再び再送を行う必要はない。つまり、再送時により適切な経路を使って再送することができる。例えば、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そのセグメントが失われてしまう可能性が非常に高い。再送時により適切な経路を使って再送することによって再送の信頼性を向上させることが可能となる。しかし、DMTCP のようにエンドエンド間で一つのコネクションを形成するのであれば、複数経路であることを活かして元々送信した以外の経路で再送を行うことができる。

DMTCP は輻輳制御を各経路で独立に行う。その方式は TCP Reno と同様なものを採用する。タイムアウトによるセグメント喪失検知の場合は、輻輳ウィンドウを 1 セグメント分とし、スロースタートと輻輳回避を行い、重複 ACK による喪失検知の場合は、輻輳ウィンドウを 1/2 に減少させる。輻輳ウィンドウ制御用変数など経路ごとに必要な情報は、経路の数だけ PCB(Path Control Block) と呼ばれるデータ領域を用意し管理する。後述

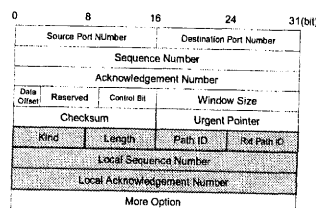


図 2: DMTCP ヘッダオプション

する送信データサイズ決定のために使う変数や、送信セグメントリストもここに格納する。

輻輳制御を各経路で独立に行うには、どの経路で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、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 TCP で用いられる通常のシーケンス番号も併用し、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ぶこととする。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 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、ローカルシーケンス番号、ローカル ACK の情報を追加する。これらは TCP ヘッダのオプションフィールドに 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)。

経路 ID は送信セグメントがどの経路で送信され、どの経路の送信ウィ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。

2.3 送信データサイズ

TCP は輻輳ウィンドウによって輻輳制御を実現している。輻輳制御はネットワークの輻輳の状態に応じて送信データサイズを調節する。輻輳制御では、ネットワークの能力のみを考慮して送信量を変化させるが、それでは受信ホストの能力が低い場合に処理しきれない量のデータを送信する可能性がある。これを避けるために TCP は受信ホストの性能に応じた送信制御機構であるフロー制御を同時に行う。フロー制御は受信ホストから通知される広告ウィンドウサイズによって実現される。つまり、送信側はフロー制御によって受信ホストの受信可能なデータサイズを判断し、輻輳制御によって各経路の処理可能なデータサイズを判断している。

送信側の TCP は、受信側の TCP から通知される広告ウィンドウと内部で保持している輻輳ウィンドウの大きさを比較し、

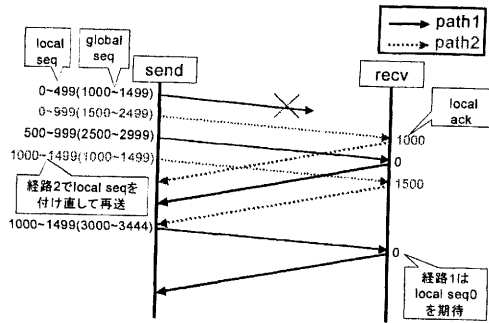


図 3: 再送時ローカルシーケンス番号を付け直した場合

小さな方の値を新しい送信ウィンドウサイズとして決定する。実際に送信するデータサイズは、送信ウィンドウサイズと送信バッファの小さな方の値から、送信後まだ確認応答のとれていないデータサイズを引いて求める。

広告ウィンドウサイズを adv_wnd 、輻輳ウィンドウサイズを $cwnd$ 、送信ウィンドウサイズを win 、送信後まだ確認応答のとれていないデータサイズを off 、送信バッファサイズを buf 、送信データサイズを len とすると、送信データサイズは以下の式のように表すことができる。

$$win = MIN(adv_wnd, cwnd) \quad (1)$$

$$len = MIN(buf, win) - off \quad (2)$$

DMTCP でも TCP と同様にフロー制御と輻輳制御によって送信データサイズを調節する。DMTCP がデータを送信する際は、最初に送信を行う経路として、輻輳ウィンドウに最大の空きを持った経路を探す。その後送信経路に応じて送信データサイズを決定する。しかし、送信後まだ確認応答のとれていないデータサイズが、コネクション全体と各経路では異なるため、送信データサイズを求める方法は TCP より少し複雑になる。

通常の TCP と同様に送信バッファサイズを buf 、コネクション全体で送信後まだ確認応答の取れていないデータサイズを off 、広告ウィンドウサイズを adv_wnd とする。経路 i の輻輳ウィンドウサイズを $cwnd[i]$ 、経路 i で送信後まだ確認応答の取れていないデータサイズを $off[i]$ 、経路 i で送信可能なサイズを $adv[i]$ 、受信ホストが受信可能なサイズを adv 、送信ホストが送信可能なデータサイズを $able$ 、送信バッファに残ってる未送信のデータサイズを $want$ 、これから送信する送信データサイズを len とすると、送信データサイズは以下の式のように表すことができる。

$$want = buf - off \quad (3)$$

$$able[i] = cwnd[i] - off[i] \quad (4)$$

$$adv = adv_wnd - off \quad (5)$$

$$able = MIN(able[i], adv) \quad (6)$$

$$len = MIN(want, able) \quad (7)$$

ここで、式 (3) における $want$ は送信バッファ中に残っている未送信データサイズであり、現在送信したいデータサイズを表している。式 (4) における $able[i]$ は経路 i の処理可能なデータサイズであり、経路 i の輻輳ウィンドウサイズから経路 i で送信後まだ確認応答のとれていないデータサイズを引くことで算出される。式 (5) の adv は受信ホストの受信可能なデータサイ

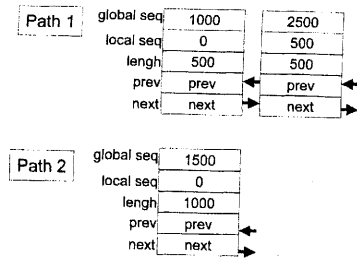


図 4: 送信セグメントリストの例

ズであり、広告ウィンドウサイズからコネクション全体で送信後まだ確認応答のとれていないデータサイズを引いて算出する。式 (6) によって現在送信可能なデータサイズ $able$ を、フロー制御と輻輳制御の結果である (4) と (5) の最小値を取ることで求めている。最後に式 (7) によって、現在送信したいデータサイズと現在送信可能なデータサイズを比べ、小さい方の値を経路 i における送信データサイズ len としている。

2.4 再送制御

2.2 で述べたように、DMTCP では再送時に用いる経路は最初の経路とは限らず異なる経路を用いることができる。再送に使う経路は、RTT 最小の経路、輻輳ウィンドウサイズが最大の経路など様々な方法が考えられる。

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再送経路で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ることになる。この様子を図 3 に示す。この例では経路 2 で再送したグローバルシーケンス番号 1000~1499 を経路 1 は待ち続ける。

DMTCP では再送セグメントには最初に使用した経路と同じ経路 ID とローカルシーケンス番号を付けることでこの問題を回避する。そのために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の対応付けを保持し、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。この対応付けのためのリストの例を図 4 に示す。このリストは、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、それに対応するローカルシーケンス番号の先頭、そのブロックのデータ長をメンバとして持つ二重連結リストで構成される。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 1 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 2 では 1500~2499 のセグメントブロックを送信している。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。

3 DMTCP の評価

3.1 シミュレーション

以上の設計に基づき、DMTCP の特徴である経路ごとの輻輳制御によるトラフィック分配の効果についてシミュレーションにより簡易評価を行った。シミュレータは BSD の実装を基にプロトコルスタックを階層的にモデル化している GloMoSim[5] を用い、TCP を修正することで DMTCP の機能の一部を実装した。現在実装の終了しているのは、各経路毎にローカル輻輳ウィンドウを用意し、ローカル輻輳ウィンドウに空きがあれば 3.2 で述べた算出式によって送信データサイズを算出し送信するという部分と、確認応答のあった経路のローカル輻輳ウィンドウサイズを増加させる部分である。各経路毎に送信セグメントリストを管理する部分は未実装であるため、セグメント喪失時にどの経路でどのセグメントが喪失したかを検知できず、セグメントの喪失した経路の輻輳ウィンドウを TCP Reno のアルゴリズムに従って調整する機能や 3.3 で述べた再送制御は実装できていない。そのため、実装済みの部分で DMTCP の有効性を簡易評価するために、セグメント喪失が発生しない状況で、送信側の輻輳ウィンドウサイズが理想の値に達するまでの推移の様子を評価した。

シミュレーションにおけるネットワークポロジは、クライアントとサーバを 2 本の経路で接続した単純な構成とし、経路 1 は帯域 1Mbps で固定遅延 20msec、経路 2 は帯域 0.5Mbps で固定遅延 20msec を設定した。トラフィックモデルとして、GloMoSim の標準アプリケーションモデルとして用意されている FTP/GENERIC を用い、クライアントからサーバに 500kbyte のデータを送信する。各経路の振り分け方式として、DMTCP とラウンドロビンでシミュレーションを行った。この 2 つの方式において輻輳ウィンドウサイズの変化の様子を評価した。輻輳ウィンドウサイズの理想値は帯域と経路の積から算出することができる。使用する 2 本の経路の帯域を足し合わせたと考えると、帯域 1.5Mbps で固定遅延 20msec の経路の RTT を計測し、ここから理想の輻輳ウィンドウサイズ 12,105byte を求めた。

送信側は通信を開始すると徐々に輻輳ウィンドウサイズを増加させていくが、理想値に達するのが早いほどスループットが向上すると考えられる。

3.2 結果と考察

図 5 に DMTCP で振り分けた場合の経路 1 と経路 2 における各ローカル輻輳ウィンドウサイズ、この 2 つを足し合わせることでコネクション全体の輻輳ウィンドウサイズとしたもの、ラウンドロビンで振り分けた場合の輻輳ウィンドウサイズの変化の様子を示す。この図から通信開始直後に DMTCP の方が早く輻輳ウィンドウサイズの理想値の 12,105byte に達していることが分かる。また通信開始直後だけでなく、セグメント喪失時に輻輳ウィンドウサイズが減少した状態からの復帰も早くなると考えられる。

また、DMTCP の場合、各経路の帯域に合うようにローカル輻輳ウィンドウサイズが変化していることが分かる。これによってラウンドロビンより DMTCP の方がセグメントの到着の順序の入れ替わりを抑えられると考えられる。セグメント到着順序の入れ替わりが減ることで、重複 ACK の発生を抑制し、

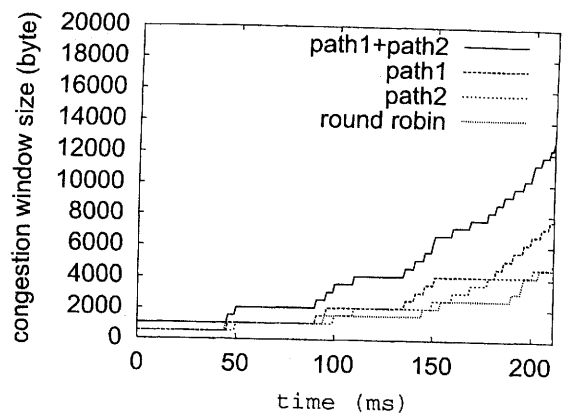


図 5: 輻輳ウィンドウサイズの変化

高速再送アルゴリズムによる無駄な再送と、無駄な輻輳ウィンドウ減少を防ぐことができる。以上のことから、DMTCP の方がラウンドロビンよりもスループットが出ると考えられる。

4 まとめ

移動無線通信環境における複数経路を用いた通信において複数経路を一つのコネクションとみなし適切な送信制御を行う方式 DMTCP の輻輳制御方式、送信データサイズの決定、再送制御方式を検討した。また現時点で実装している部分で示すことのできる DMTCP の効果を簡易的に評価した。今回は再送制御部分の実装が間に合わなかったため、再送制御の有効性や、再送処理を含んだ通信終了時のスループットに関する評価は省略している。今後は再送制御部分を実装し、再送が発生した場合の DMTCP の有効性について評価を行う。

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] 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, Vol.J84-B, No.3, 2001.
- [3] P.Gevros, F.Risso and P.kirstein, “Analysis of a Method for Differential TCP Service”, GLOBECOM’99, Dec. 1999.
- [4] 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.
- [5] GloMoSim, <http://pcl.cs.ucla.edu/profects/gloimosim/>.