1T－7

複数経路データ転送におけるコネクション指向プロトコルの提案

星谷 直哉　相田 仁　齊藤 忠夫

東京大学工学部

1 はじめに

インターネットのようなベストエフォート型通信を提供する分散型ネットワークではその信頼性は低く、電子商取引、電子決済、行政上の各種証明・申請手続き等、高い信頼性やある程度のリアルタイム性を必要とする作業を行うのは困難であった。そこで、インターネットの信頼性を向上する手法としてあらかじめ複数の経路を選択してデータを転送する複数経路データ転送方式を取り上げる。本稿ではその通信プロトコルを提案し、その性能について述べる。

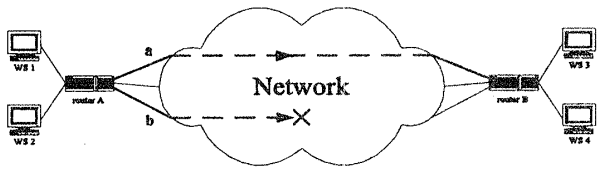

図 1: 複数経路データ転送の概念

2 複数経路データ転送

複数経路データ転送は、通信を行う際に送信元から受信先へ至る転送経路を複数設定し、それぞれの経路にデータを送出する方法である。複数経路データ転送の概念を表したものを図1に示す。

WS1,2 から WS3,4 へデータを送信する際に、経路 a および b を確保してそれぞれにデータを送信すれば、例えば経路 b の伝送路上で何らかの障害が発生して通信が出来なくなったとしても、経路 a によってそれを補うことができ通信を中断することなく続けることが出来る。

複数の経路におけるデータ転送形式としては、それぞれの経路全てに同一データを複製したものを送出する方式、全てではなくランダムに選んだ幾つかの経路に送出する方式、障害・輻輳等の経路状態に合わせて各経路にデータを分配する方式などが考えられる。複数の経路に同一データを送信する方式では、受信側で1番最初に到着するパケットを採用することにすると、単一の経路で送るよりもスループットの向上も望めると思われる。一方、複数の経路に同一のデータを送ることで無駄なトラフィックを作ることになるが、特定の重要な通信のみに本手法を適用することによって悪影響の発生を抑える。

3 TCP に準じた複数経路データ転送プロトコルの提案

複数経路データ転送の機能をトランスポート層で実現することにより、効率的な輻輳・フロー制御とうまく組み合わせることができる。そこで、実際にインターネットで広く使用されている TCP を拡張する形で新しいコネクション指向のプロトコルを提案する。

3.1 ヘッダ

図 2に新しいプロトコルにおけるヘッダの構成を示す。ほぼ TCP と共通であるが、順序番号は各経路ごとに独立の値を持つこととする。これらの他に RTT やタイムアウトを調べるために TCP オプションの形で各経路ごとにタイムスタンプをつける。

Source Port								Destination Port	
Sequence Number									
Acknowledgement Number (common)									
Data Offset	Reserved		URG	ACK	PSH	RST	SYN	FIN	Window Size
Checksum								Urgent Pointer	
Option-Kind		Option-Length(=11)						Route ID	
Timestamp Value									
Timestamp Echo Reply									Padding

図 2: 提案プロトコルのヘッダ

3.2 コネクション管理

コネクションの確立は、TCP と同じくスリーウェイハンドシェイクを用いる。コネクションを張る際には受信側が全経路からのセグメントを全て受け取って初めて ACK を全経路に返すこととし、送信側も ACK を全経路から受け取ることで次の状態へ移行するものとする。

Proposal of Connection Oriented Protocol on Multi-path Data Transmission

Naoya Hoshiya, Hitoshi Aida, Tadao Saito

Faculty of Engineering,The University of Tokyo

何度か再送を行なっても全てのセグメントが到着しない場合は、RST のフラグをセットしたセグメントを送信して初期状態に戻す。

コネクションの解放時は、FIN や ACK のフラグを立てたセグメントを全経路に送るが、そのうちの一つが届いた時点ですぐにそれに対する ACK を返信して次の状態へ移行してもかまわない。ただし、最後の 2MSL タイムアウトに関してはそのコネクションにおける全てのセグメントがネット上からなくなることを保証するため、一番遅い経路に合わせて設定する必要がある。このように、コネクションを張る際の動作は、複数経路からのセグメントを待つ必要があること以外は TCP の場合と大きく違うところはない。

3.3 データ転送管理

データ転送には前述したように幾つかの方法が考えられるが、受信側では複数の経路のうちもっとも早く届いたセグメント内のデータを有効とする方法を用いる。送信側では全経路にデータを転送する方法と経路状態によって送信経路を選択する方法を考える。どちらの方法でもヘッダは共通のものでよく、受信側の動作も変える必要はない。トランスポート層の重要な役割の一つであるフロー制御も TCP と同様に輻輳ウィンドウと受信ウィンドウによって行う。受信側は各経路共通の一つの受信ウィンドウによって制御するが、送信側は各経路の状態がそれぞれ異なるため、輻輳制御は経路ごとに独立で行う。ヘッダの順序番号が各経路で独立としたのもこのためである。送信側では、タイムアウト処理などに必要な RTT も各経路ごとに計算する必要があるが、ACK 番号は全経路で共通であるためそれぞれの経路は自分の送信したセグメントに対する ACK であるか否かの区別がつかず、RTT を計算することができない。そこで、RFC1323 で定義されているタイムスタンプオプションに更に経路を区別する ID を付け加えることによって各経路ごとにセグメントの往復時間を計測することを可能とし、それぞれの RTT を更新する。送信経路を選択して負荷を分散する方法では、RTT の値を比較するなどの方法で経路に順序付けを行い、上位から送信したい経路数だけ選択してその経路のみで送信処理を行う。受信側は、到着したセグメントの順序制御、データ重複の回避を TCP と同様に行う。

ここで、ある経路の回線速度が大きく、その経路で送信されたセグメントの ACK が残りの経路の snd_nxt よりも大きい値となった場合を考える。この場合、速度の遅い経路で送信しようとしているデータは既に受信側で受信されていることになり、このまま送信するのは無意味である。そこで、遅い経路の snd_nxt をデータを送信することなく ACK の指す値まで更新する。これによって、速度の遅い経路が無駄にデータを送信することを防ぐ。また、こうすることによって遅い経路の snd_nxt が速い経路のそれにすぐに追い付くことができるので、その後ネットワーク状況が変化し回線速度が逆転してしまうような場合でも、すぐに速くなった経路がそれまで最も速かった経路を追い越すことができる。これはちょうど複数の経路の最も速度の速いものを常に選択して送信するような形に近く、全体としての速度向上が見込まれる (図 3)。

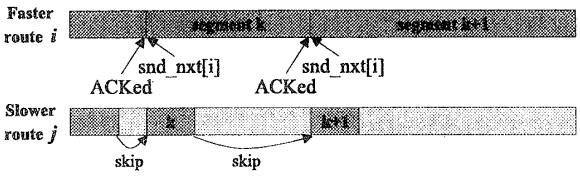

図 3: 複数経路データー転送方式における速度向上

4 まとめ

本稿では、インターネットにおいてより信頼性の高い通信を行う方法として複数経路データ転送を提案し、この方式をトランスポート層レベルで実行することを考えたコネクション指向のプロトコルを TCP の拡張として提案し、その動作概要を示した。今後の研究課題として、今回提案したプロトコルを疑似的に実装し、実験を行ってその評価を行う予定である。そのための評価方法についても考察する必要がある。

参考文献

[1] J.Postel, “Transmission Control Protocol - DARPA Internet Program Protocol Specification”, RFC 793, September 1981.

[2] V.Jacobson,R.Braden,D.Borman, “TCP Extensions for High Performance”, RFC 1323, May 1992.

[3] P. Miller, 苅田幸雄 監訳, “マスタリング TCP/IP”, オーム社,1998.

[4] A.S.Tanenbaum, “Computer Networks Third Edition”,Prentice Hall International Editons,1996.

[5] D.Bertsekas,R.Gallager, 八星禮剛 監訳, “データネットワーク”, オーム社,1990.

[6] D.Comer, 村井純・楠本博之 訳, “bit 別冊 TCP/IP によるネットワーク構築”, 共立出版,1990.

[7] 山口英, “カーネルを読もう (10) ～ (12)”,UNIX MAGAZINE 1997.6 ～ 8,ASCII