並列に動作するサブコネクションを用いた超高速アプリケーション向け TCP 輻輳制御方式の提案

加藤 聰彦　　井上 広喜　　伊藤 秀一
電気通信大学　大学院　情報システム学研究科

1. はじめに

近年、ネットワークの広帯域化が進み、超 Gbps クラスのネットワークの構築が行われている。これに伴い、高エネルギー・核融合科学などの科学技術研究において、超広帯域ネットワークを用いて、テラバイトを越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いる。このような要求を満足させるためには、超大容量データ転送アプリケーションのための TCP 輻輳制御が重要な課題となる。TCP では、送信側で連続的に送信できるデータ量を管理する輻輳ウィンドウを導入し、パケットロスが生じそのパケットの再送が必要となった時点で輻輳が生じたと判断し、輻輳ウィンドウの値を減少させ、その後徐々に輻輳ウィンドウを増加させるという方法を用いている。この輻輳制御は、広帯域ネットワークにおいても、1 つ 1 つの TCP コネクションの利用帯域はそれほど大きくなく、多数の TCP コネクションが交互に輻輳ウィンドウを増減させながら、全体でネットワークの帯域を活用することを前提としている。しかし 1 つの TCP コネクションで超大容量のデータ転送を行うアプリケーションでは、輻輳が生じた場合に大量のパケットロスが生じ、輻輳ウィンドウの増加速度が極端に低下し、スループットが上がらないという問題が起こる。このような TCP 輻輳制御については、IETF においてもいくつかドラフトが提案されている[1,2]が、新たな方式の確立にはいたっていない。また本稿の提案と同様な目的を有する方式が、MulTCP として提案されている[3]。しかし MulTCP では、1 つの TCP コネクションの中に仮想的に複数のサブコネクションがあった場合の輻輳ウィンドウの変化を模擬する近似的な方法を用いているため、複数の TCP コネクションの動作を正確に反映できず、高速な再送がサブコネクションのうち 1 つでしか行われないため性能が上がらない。そこで本研究では、実際に複数のサブコネクションを用いた、超大容量データ転送のための新たな TCP 輻輳制御のアルゴリズムの提案を行う。

2. 概要

(1) 1 つの TCP コネクションのデータ転送を、複数のサブコネクションに分ける。これにより、データ転送レートの低い、多数の TCP コネクションが存在する状況を等価的に実現する。

(2) TCP の有するデータの順序制御、フロー制御、再送制御の各機能は、TCP コネクション全体で行うこととし、TCP ヘッダの有するシーケンス番号と受信確認シーケンス番号を使って行う。一方、輻輳制御は、各サブコネクションで独立に行う。

(3) サブコネクションの導入に当たって、TCP ヘッダに追加の情報は導入しない。また、サブコネクションごとの輻輳制御は、送信側のみで実現する。

(4) 送信側はサブコネクションごとに輻輳ウィンドウを管理し、輻輳ウィンドウが送信を許可するサブコネクションをラウンドロビンに使用してデータを送信する。送信後はそのパケットのシーケンス番号とサブコネクションを記録し、送信したデータを再送用に保持する。

(5) 再送制御および輻輳制御は、通常の TCP と同様な方式を採用することとする。すなわち、再送については、重複 ACK を 3 つ受信すると対応するデータを再送する Fast Retransmit と、タイムアウト再送の 2 種類に対応する方法を用いる。また輻輳制御については、Fast Retransmit の後は、輻輳ウィンドウを 1/2 に減少し Congestion Avoidance により徐々に輻輳ウィンドウを増加させ、タイムアウト再送の後は、輻輳ウィンドウを 1 パケット分とし、スロースタートと Congestion Avoidance を行う。

(6) 再送制御を受信確認シーケンス番号を用いて単純に行うと、パケットロスのうち最も小さいシーケンス番号のものだけしか Fast Retransmit ができなくなる。すなわち複数のサブコネクションでパケットが損失しても、Fast Retransmit が行われるのはその内 1 つのサブコネクションのみで、他はタイムアウト再送となり、全体の性能が低下する。そこで、TCP の選択的受信確認 (Selective Acknowledgment: SACK)オプションを用いて、非連続的に受信された番号を確認し、それにより重複 ACK が受信されないサブコネクションに対しても、そのサブコネクションで受信されていない最小のシーケンス番号を持つパケットを検出し、Fast Retransmit と同様な処理を実現する。

(7) 送信側は ACK パケットを受信すると、以下を行う。
- 受信確認が取れていなかった最も古いものから ACK パケットの受信確認シーケンス番号に対応するデータパケットまでに対して、すべて ACK パケットが受信されたものとして、再送用のデータを解放し、輻輳ウィンドウを更新する。
- 重複 ACK が 3 つ受信された場合、Fast Retransmit の手順によりその番号に対するデータパケットを再送し、Congestion Avoidance の手順を行う。
- SACK オプションが存在する場合は、SACK オプションにより 3 回連続で受信確認が行われない場合は、その最小のシーケンス番号を持つデータパケットに対して、重複 ACK が 3 つ連続で受信されたと解釈し、そのデータパケットを再送し、Congestion Avoidance の手順を行う。

3. 通信シーケンス

送信側は、i 番目のサブコネクションに対して、輻輳ウィンドウ cwnd[i]、スロースタートスレッショルド ssthresh[i]、そのサブコネクションで送信したデータのう

"Proposal on TCP Congestion Control for Ultra High Speed Application Using Parallelized Subconnections"
Toshihiko Kato, Hiroki Inoue and Shuich Itoh
University of Electro-Communications

ち、受信確認が取れていない最小シーケンス番号 una[i]、サブコネクションごとの重複 ACK の受信回数 dup_ack[i] などの変数を管理する。サブコネクションが 4 本の場合(1 から 4)の通信シーケンスの例を図 1 に示す。

最初に 4 つの cwnd はそれぞれ 1 に初期化される(①)。これに従って、1 から 4 までのデータがサブコネクション 1 から 4 で転送される。

受信側は 2 つのデータを受信すると Ack により受信確認を行う。Ack 3 を受信すると、送信側は Data 1 と Data 2 を送信したサブコネクションに対して Ack を受信したと解釈し、それぞれの cwnd を広げる(②)。続いて、Data 5 から Data 8 までがサブコネクション 1 と 2 により転送される。このとき Data 5 がサブコネクション 1、Data 6 がサブコネクション 2 というように、ラウンドロビンにサブコネクションが使用される。またこの例では Data 5 と Data 8 が輻輳などにより紛失されるとしている。

受信側は Data 3 と 4 を受信すると Ack 5 を返す。これによりサブコネクション 3 と 4 の cwnd が 2 に増加し、una[1]と una[2]がそれぞれ 5 と 6 になる(③)。cwnd の増加に応じて、Data 9 から Data 12 までがサブコネクション 3 と 4 で転送される。このとき Data 10 と 11 が紛失されると想定している。

受信側は Data 6 を受信すると、Data 6 を受信したことを示す SACK オプションをつけた Ack 5 を返す。この受信で、dup_ack[1]が 1 に設定されるとともに、サブコネクション 2 の una が 8 となり、またサブコネクション 3 と 4 の una も設定される(④)。さらに Data 7 に対する Ack により、dup_ack[1]が 2 に増加する。

次に Data 9 が受信側に届くと、4 までの受信と 6/7 および 9 の選択的受信を示す Ack が返される。これはシーケンス番号 5 の 3 つ目の重複 ACK であるため、サブコネクション 1 上で Data 5 が再送される。また 9 の選択的受信確認によりサブコネクション 3 の una が 11 となり、Data 8 が受信されていないことがわかるため、サブコネクション 2 の dup_ack が 1 になる(⑥)。さらに Data 5 が再送された後、Fast Recovery の手順により、ssthresh[1]が 1 に、cwnd[1]が ssthresh[1]+3=4 になり、Data 13 と 14 が新たに送信される(⑦)。

受信側は Data 12 を受信すると、4 までの受信と 6/7, 9, 12 の選択的受信を通知する Ack を返す。これにより、サブコネクション 1 の cwnd は 1 増加し、Data 15 が送出され、また Data 8 の受信が通知されないため、サブコネクション 2 の dup_ack が 2 となる。さらに、Data 10 と 11 が受信されていないことがわかるため、サブコネクション 3 と 4 の dup_ack がそれぞれ 1 となる(⑧)。

受信側は次に、再送された Data 5 を受信する。これに対して次に受信を期待するシーケンス番号 8 と、9 と 12 の選択的受信を通知する Ack を返す。これは送信側に対して新たな Ack であるため、サブコネクション 1 に対して、dup_ack が 0 に戻され cwnd が ssthresh の値 1 に設定されるとともに、その una の値が更新される。また Data 8 の受信が通知されないため、dup_ack[2]が 3 となり Data 8 が再送される。さらに Data 10 と 11 が受信されていないことが通知されるため、dup_ack[3]と[4]がともに 2 となる(⑨)。続いて⑦の場合と同様にサブコネクション 2 の cwnd が増加し、Data 16 から 18 が転送される(⑩)。

続いて Data 13 を受信すると対応する Ack が送信される。これによりサブコネクション 1 の una の更新、サブコネクション 2 の cwnd の増加が行われるとともに、サブコネクション 3 と 4 の dup_ack が 3 となるため、Data 10 と 11 が再送される(⑪)。

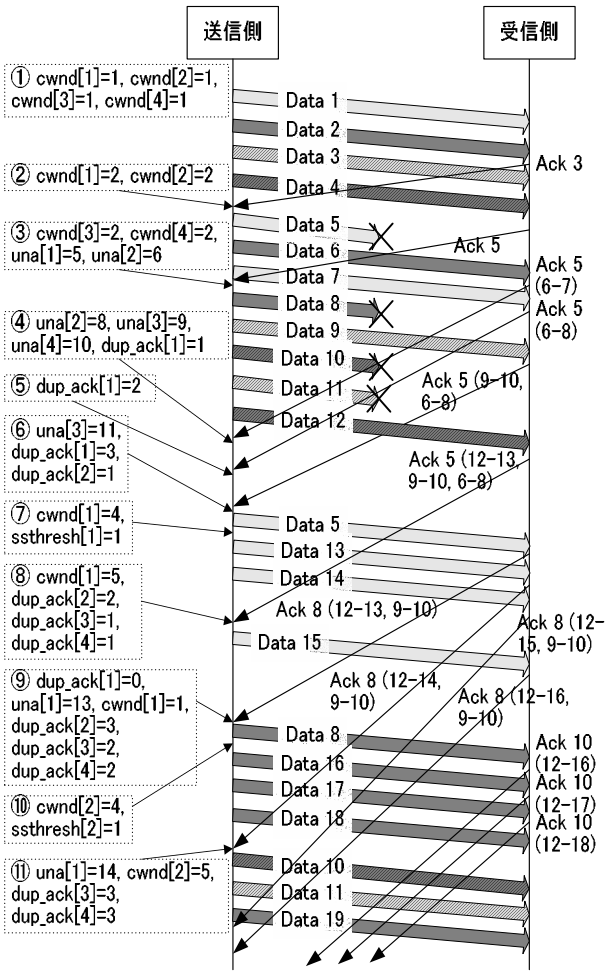

図 1　通信シーケンスの例

以上のようにして、サブコネクションごとに独立の TCP コネクションに近い形で Fast Retransmit ならびに輻輳制御を行うことができる。

4. おわりに

本稿では、超高速ネットワークを単一のアプリケーションで専有する場合の TCP 通信のための、新たな輻輳制御方式の提案を行った。この方式では 1 つの TCP 通信にサブコネクションを導入し、それぞれのサブコネクションで独立の輻輳制御を実現し、さらに SACK オプションにより、効率的な Fast Retransmit を可能としている。今後は本方式の性能評価を進める予定である。

参考文献

[1]: S. Floyd, “High Speed TCP for Large Congestion Windows,” INTERNET DRAFT, available at draft-floyd-tcp-highspeed-00.txt, June 2002.

[2]: S. Floyed, “Limited Slow-Start for TCP with Large Congestion Windows,” INTERNET DRAFT, available at draft-floyd-tcp-slowstart-01a.txt, August 2002.

[3]: P. Gevros, et al., “Analysis of a Method for Differential TCP Service,” available at http://www-mice.cs.ucl.ac.uk /multimedia/publications/gi99.ps.gz, December 1999.