動的に確立するサブコネクションを用いた超高速 TCP 通信のための輻輳制御方式

堀内 律之　　加藤 聰彦　　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、さまざまな科学技術研究において、超広帯域ネットワークの帯域を活用し、テラバイトを越える大容量のデータを高速で転送し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、超高速 TCP 通信のための輻輳制御が重要な課題となる[1]。これに対して、筆者らは、1 つの TCP コネクションの中に、輻輳制御を分割して行う仮想的なサブコネクションを用いた TCP 輻輳制御のアルゴリズムを検討している[2]。本論文では、これまで検討した方式に、サブコネクションの本数を動的に変更する機能を追加するとともに、詳細なアルゴリズムを検討した結果について述べる。

2. 設計方針

(1) 1 つのコネクションで Gbps クラスのトラヒックを生成する TCP 通信に対して、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。図 1 に概念図を示す。
(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 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、TCP コネクション全体で行う。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、データの送信側のみで実現する。すなわち、サブコネクションは送信側が仮想的に管理するもので、明確な確立解放手順やヘッダ情報は持たない。
(4) TCP コネクションの最初は、サブコネクション数が 1 で開始し、ACK を受信するごとに、事前に指定した最大数まで、サブコネクション数を増加させることとする。またタイムアウト再送によりスロースタートが行われた後に、タイムアウト再送が行われた場合は、そのサブコネクションを使用しないこととする。これにより動的にサブコネクションの数を調整する。
(5) 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。
(6) 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。
(7) 複数のサブコネクションで転送したパケットが損失した場合に、各サブコネクションで Fast Retransmit を行わせるために、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。
(8) 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、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。
- 再送については、対応するセグメントを送信したサブコネクションについて行われるものと想定する。

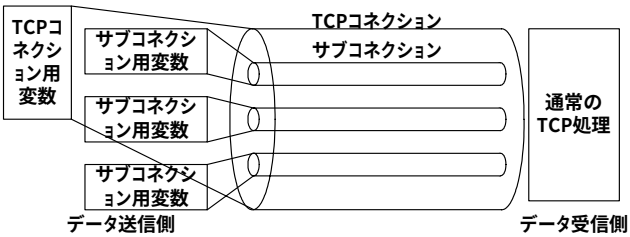

図 1　概念図

3. 詳細アルゴリズム

3．1　制御用データ構造

本方式を実現するために、送信側で管理するデータ構造を図 2 に示す。TCP コネクション用の変数としては、送信したセグメントのシーケンス番号を管理する snd_nxt/ snd_max/ snd_una、通信相手から通知されたウィンドウサイズ win、その時点で使用しているサブコネクション数 subcons、使用可能なサブコネクションの最大値 max_subcons、次にデータを送信すべきサブコネクション

"TCP Congestion Control for Ultra High Speed Application Using Dynamically Established Subconnections"
Noriyuki Horiuchi, Toshihiko Kato and Shuich Itoh
University of Electro-Communications

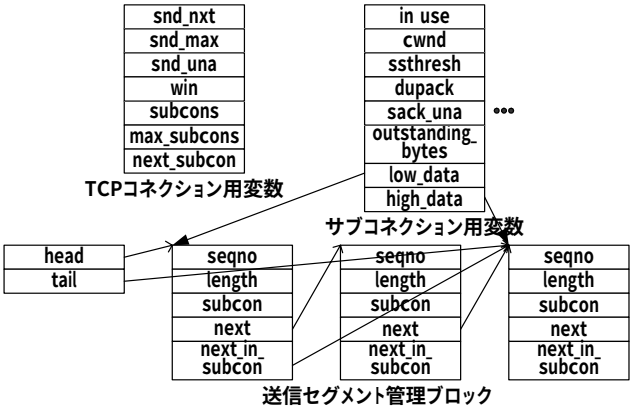

図 2　制御用データ構造

next_subcon を含む。また、サブコネクションごとには、そのサブコネクションが使用中かどうか(in_use)、輻輳制御関係のパラメータ cwnd/ssthresh、重複 ACK の受信回数(dupack)、ACK または SACK により受信応答が取れていない最小のシーケンス番号(sack_una)、そのサブコネクション中でどれだけのバイト数がアウトスタンディングであるか、そのサブコネクションで送信したセグメントに対する管理ブロックへのポインタを用意する。これらの情報はサブコネクションの番号に対する配列で用意される。また、送信し応答を受信していないセグメントの情報を管理する送信セグメント管理ブロックを導入する。ここでは送信したセグメントごとに対応するサブコネクションの番号を示すとともに、同一のサブコネクションで送信されたセグメントのリストを構成している。これらの構造体により、サブコネクションから送信したセグメントへ、また、シーケンス番号からそのセグメントを送信したサブコネクションへ対応付けることが可能となっている。

3．2　データ送信時の手順

データを送信する場合は、まず相手から通知されたウィンドウの範囲でデータが送信可能か(snd_max − snd_una < win)かどうかをチェックする。可能であれば、次の送信すべきサブコネクション(next_subcon)のサブコネクションを調べ、そのサブコネクションが使用中で(in_use = 1)でかつ輻輳ウィンドウの範囲で送信可能か(outstanding_bytes < cwnd)どうかを調べる。

送信可能である場合は、セグメントを送信し、その情報を送信セグメント管理ブロックに記録する。この管理ブロックは管理ブロック内全体のリスト next、同一サブコネクション内のリスト next_in_subcon の 2 種類のリストにリンクされ、サブコネクション用変数の high_data によってもポイントされる。また outstanding_bytes に送信したセグメントのバイト数が追加される。さらに next_subcon が次のサブコネクションを指すようにする。

一方、次のサブコネクションでは送信できない場合は、さらに次のサブコネクションを調査する。現在確立されているすべてのサブコネクションで送信できない場合は、送信を待つ。

3．3　ACK 受信時の手順

SACK オプションを含む ACK を受信した場合は、まず確認シーケンス番号 ack が新たにセグメントの受信を通知するものである場合(ack > snd_una)は、サブコネクションの本数を 1 増加する。また snd_una = ack とする。

また、ack と、SACK オプションのパラメータ(left、right で代表して表す)を用いてすべてのサブコネクションに対して以下の処理を行う。

- まず、次の処理により、ack および SACK オプションの値から重複 ACK の検出を行う。
  - ack = sack_una ならば、dup_ack++とする。
  - dupack = 0 かつ sack_una がすべての left から right − 1 の範囲に入っていないならば、dupack++とする。
  - dupack > 0 かつ(ack < sack_una または sack_una がすべての left から right − 1 の範囲に入っていない）ならば、dupack++とする。

  dupack の値により通常の Fast Retrasmit と Fast Recovery の処理を、以下のようにサブコネクションごとに行う。
  - dupack が 0 から 2 となった場合は何もしない。
  - dupack が 3 となった場合は、sack_una のパケットを再送し、ssthresh をこれまでの min(cwnd, win)の値の半分とし、cwnd を ssthresh+3MSS とする。
  - dupack が 4 以上となった場合は、cwnd を MSS 分増加し、新たに送信可能となったデータパケットを送信する。

  新たに送信したデータパケットの ACK を受信したこと(すなわち再送したパケットが受信されたこと）は以下のようにして検出する。
  - ack > sack_una となった場合。
  - sack_una が left から right − 1 の範囲に入っている SACK オプションがある場合。

  その場合は以下の処理を行う。
  - dupack = 0 とする。
  - cwnd を ssthresh の値に戻す。

3．4　タイムアウト発生時の手順

タイムアウトが発生した場合は、snd_nxt を snd_una の値にもどして、サブコネクションの cwnd のみを 1 パケットとし、そこから再送する。また、その時点に cwnd がすでに 1 であった場合、そのサブコネクションを用いない処理を行う。

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を想定した、超高速 TCP 通信のための輻輳制御方式についての検討結果を述べた。この方式では 1 つの TCP 通信に、動的に確立される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ている。

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]: 加藤 他, “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,” 情処第 65 回全大, 6H-2, March 2003.