L-008

TCP パケットフローからの輻輳ウィンドウの推定
Estimation of Congestion Window from TCP Packet Flow

加茂 亮平† 平中 幸雄† 武田 利浩†
Ryouhei Kamo Yukio Hiranaka Toshihiro Taketa

1. はじめに

近年の情報化社会に伴い、より効率の良いネットワークインフラの研究、開発が盛んである。その中には、TCP プログラムの開発、改良も含まれる。TCP（Transmission Control Protocol）は、インターネットで標準的に使われているプロトコルである。TCP は受信側からの確認応答により、送信に確実性があり、またフロー制御や輻輳制御等の機能により、効率的な送信を実現し、信頼性の高いプロトコルとなっている。TCP プログラムの開発では、より効率的な輻輳制御アルゴリズムが検討されている。開発した輻輳制御が、正しく、設計者の期待通りに動いているかを確かめる必要がある。輻輳制御動作で最も重要なのは、輻輳ウィンドウである。そこで、本研究では TCP パケットフローから輻輳ウィンドウを推定する方法を検討した。

2. TCP

1.1 TCP パケット

TCP はデータを送信する際に、データを複数のパケットに分割して送信を行う。パケットには送信したいデータのほかに、宛先などの送信時に扱うデータが記述されている。加えて TCP では、送信者がパケットを送信すると、受信者はパケットが届いたかどうかの確認応答を行う。この確認応答で返信されるパケット信号を ACK パケットという。

本研究では、送受信間のデータと ACK のパケットの流れを利用する。

1.2 輻輳ウィンドウ

送信者は送信を効率良く行うために、ACK の受信を待たずに、一度に複数のパケットを送信する方式が取られている。一度に送信できるパケットの量をウィンドウサイズという。送信者はウィンドウサイズを増減することによって、送信する量を調節する。輻輳ウィンドウ（cwnd）はネットワークの状態によって増減するウィンドウサイズの値である。

1.3 輻輳制御

ネットワークにアクセスが集中し、ネットワークの回線容量を超えると、通信が処理しきれなくなる場合がある。この状態を輻輳という。輻輳はネットワークの効率を下げる原因となる。

TCP では、輻輳が起こらないように、輻輳制御を行っている。輻輳制御では主に、回線が空いているときは通信の量を増やし、混雑しているときは通信の量を減らすように、輻輳ウィンドウを増減していく。

スロースタートは TCP 輻輳制御の基礎的な方法である。スロースタートでは、ネットワークの状況が分からない通信開始時は、輻輳ウィンドウを低く設定して、通信量を少なくする。そして、受信者から送信成功の確認応答を受け取り、回線に余裕があることを確認すると、ウィンドウサイズを徐々に増やしていく。

スロースタート時、送信側は基本的に次のように動作する。

1) 輻輳ウィンドウが 0 以上のときは、データの送信を行う。
2) データを一つ送信すると、輻輳ウィンドウを 1 減らす。
3) ACK を受信したら、輻輳ウィンドウを 2 増やす。

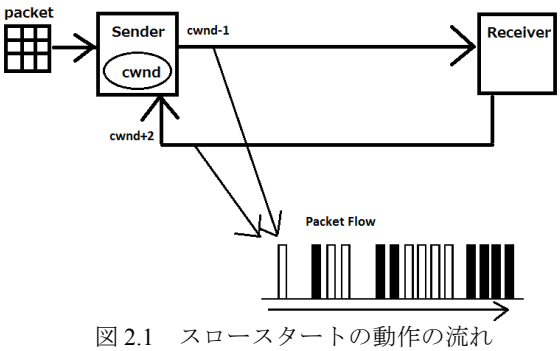

図 2.1 スロースタートの動作の流れ

実際に送信を行う場合、パケットフローと輻輳ウィンドウは図 2.2 のようになると考えられる。

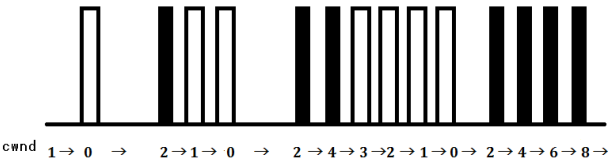

図 2.2 パケットフローとその輻輳ウィンドウの推移

図はデータを 7 パケットだけ送信したときのパケットフローである。白が送信データのパケット、黒が ACK のパケットを表す。輻輳ウィンドウは 1 から始まり、データを送信すると 1 減り、0 になるとそれ以上は送信を行わない。ACK を受信すると、輻輳ウィンドウが 2 増え、再度データを送信する。送信するデータがなくなれば送信を終える。当然、輻輳ウィンドウが 0 より大きくても送信は行わない。

本研究では、このスロースタートの条件により得られたパケットフローを対象に輻輳ウィンドウの推定を行った。

† 山形大学 Yamagata University

3. アルゴリズムの検討

本研究では、逆方向的に輻輳ウィンドウの推定する方法をとった。逆方向の推定では、パケットフローを時間の後から順に調べ、動作や条件を通常とは逆にして、輻輳ウィンドウの推定を進めていく。図 2.1 のスロースタートの動作の方向や計算式を反転した場合、図 3.1 のようになり、輻輳ウィンドウの推定はこの図に沿って行う。

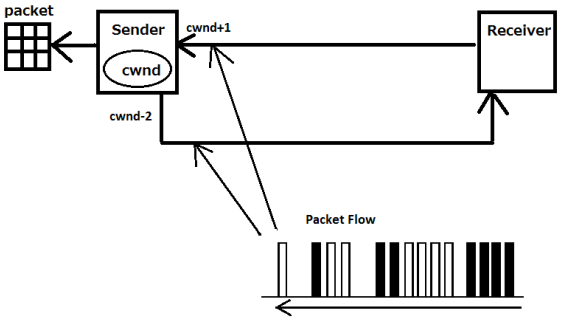

図 3.1　パケットフローから輻輳ウィンドウの推定

通常時とは逆に、ACK の検知から始まり、ACK 毎に輻輳ウィンドウを 2 減らす。そして、データ毎に輻輳ウィンドウを 1 増やす。

さらに推定結果を絞るために、いくつか条件を追加する。通常の送信では、輻輳ウィンドウが 1 以上のときにデータの送信を行う。このとき、ウィンドウサイズは 0 以上であり、送信するデータがあれば長い時間で 1 以上になることはないと言える。この二つの条件を推定に加える。

また、逆方向の推定では、輻輳ウィンドウの初期値が不明であるため（通常の送信での最後の確認応答時点での輻輳ウィンドウは送信パケットの数次第であり、様々なケースが考えられる）、全てのケースについて推定していく必要がある。輻輳ウィンドウに限界値を定めて、限界までのそれぞれのケースについて、推定を行う。

まとめると、次のような条件を使って、輻輳ウィンドウを推定する。

1) データ送信一つ毎に、輻輳ウィンドウを 1 増やす。
2) ACK 受信一つ毎に、輻輳ウィンドウを 2 減らす。
3) 輻輳ウィンドウは常に 0 以上でなければならない。
4) 送信終了時を除いて、送信間隔 d を超えて、輻輳ウィンドウが 1 以上になってはならない。
5) 推定開始時のウィンドウサイズの値は、可能性のある値の範囲で全て試す。

結果として、推定開始時の輻輳ウィンドウの値を含めて、ある時点での輻輳ウィンドウが一意に定まることが期待される。

4. シミュレーション結果

前章で検討した条件により、パケットフローから輻輳ウィンドウを推定するシミュレーションを行うとき、図 4.1 のような結果となった。

7 パケットだけ送信したときのパケットフローに対してシミュレーションを行ったとき、輻輳ウィンドウの初期値が 8 として一意に与えられ、そのときの輻輳ウィンドウの推移が推定できる。

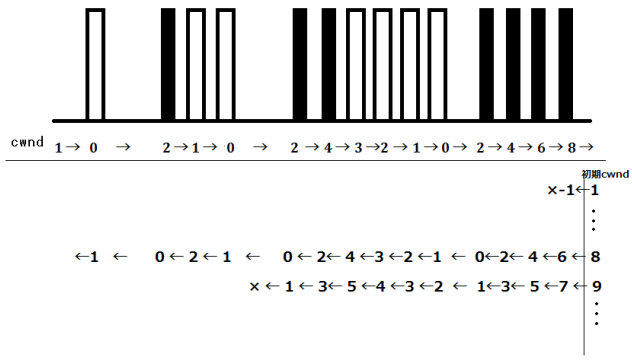

図 4.1　シミュレーション結果

輻輳ウィンドウの初期値が 8 以外であるとき、条件を満たさないので、結果は不可となる。例えば初期値が 1 のとき、輻輳ウィンドウが途中で負の値になり、条件を満たさない。また初期値が 9 のときは、送信間隔を超えて、輻輳ウィンドウの値が 1 以上になる箇所があるので、同じく条件を満たさない。実際にプログラムし、シミュレーションを行うと、図 4.2 のような結果となる。

def	cwnd	TorF
1		false
2		false
3		false
4		false
5		false
6		false
7		false
8		true
9		false
10		false
11		false
12		false
13		false
14		false
15		false
16		false
17		false
18		false
19		false

図 4.2　シミュレーション結果一覧

図は、左の値が推定開始時の輻輳ウィンドウの値、右の値がそのときの可能性の有無である。このシミュレーションの結果から、輻輳ウィンドウの初期値が一意に定まり、輻輳ウィンドウの推移を推定することができた。

5. おわりに

パケットフローから輻輳ウィンドウを推定する方法を検討し、シミュレーションを行った。

今後の課題として、TCP 輻輳制御アルゴリズムについて、輻輳ウィンドウを推定する方法を検討することが挙げられる。

参考文献

[1] Yukio Hiranaka, Toshihiro Taketa, Shinichi Miura, “Case Branching Backward Simulator for Integer Factorization”, 8th EUROSIM Congress on Modelling and Simulation (2013).
[2] 加茂亮平, “TCP パケットフローの逆方向シミュレーションモデル”, 山形大学卒業論文(2013).