輻輳制御パラメータの自動解析を行うTCP通信解析ツールの研究

茂木 重憲† 渡辺 晶†
明星大学大学院 情報学研究科 情報学専攻‡

1 はじめに

TCPは, 輻輳制御パラメータであるcwndやssthreshを用いて1度に送出するデータ量や輻輳動作を制御している. cwndとssthreshの値が分かれば送出可能なデータ量や輻輳制御動作が把握できる. しかし, cwndとssthreshの値はネットワーク上に送出されないため, 輻輳制御動作の解析にはcwndとssthreshを推定する必要がある. tcpdumpを用いて再送パケットやその原因を特定するには輻輳制御動作の解析が必要となり, cwndやssthreshの値の推定を手作業で行うことになる. このため, 通信トレースからTCP実装の輻輳制御動作に基づいて自動的にcwndとssthreshの値を推定するツールを開発した.

2 設計

輻輳制御アルゴリズムは多数のバージョンが存在する. 本研究では, FreeBSD 7.0に実装されているTCPのバージョンであるReno, NewReno, SACK, オプションであるタイムスタンプオプション, ウインドウスケールオプション, RFC3042オプションを解析対象とした. 解析方法は, tcpdumpの通信トレースからTCP実装の輻輳制御手順に基づく解析を行ってcwndとssthreshの値を推定する方法と取る. しかし, TCPの仕様通りにTCPが実装されているとは限らない為, 輻輳制御手順はFreeBSDのソースファイルを参照して解析方法を考え, 実装した. また, 図1, 図2の様に視覚的に問題点を発見しやすいようにTCPの通信をプロットして表示する機能と表形式でパケットのcwndやssthreshの値などの詳細情報や再送などの問題発生箇所に色を付けて表示する機能を実装した.

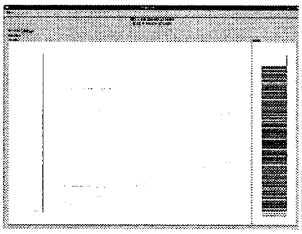

図 1: プロット表示

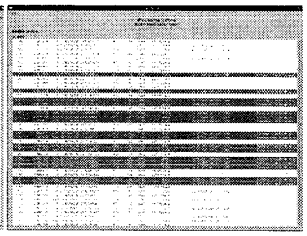

図 2: 表形式

†Shigenori Mogi, Akira Watanabe
‡Department of Infomation, Graduate School of Informatics,MEISEI University

2.1 解析方法

- NewReno
  本研究ではFreeBSD 7.0のTCP実装がRFC 2582を参照しているので, 以下のように実装した.
  - recoverという変数に今まで送信した最大シーケンス番号を入れる.
  - ssthreshをcwndとウィンドウサイズの小さい方の半分, cwndをssthresh+3MSSに設定する.

  以降, recoverの値以上のACKを受信した場合はcwndをssthreshに設定し再送を終了, それ以外はpartial ACKとして一時的にウインドウがACKされたパケットから始まるようにcwndを設定する. 再送が終了するとcwndの値を再送前の値に戻す.

- SACK
  SACKが有効になっている場合, TCPコネクションを確立する際にSack-Permittedオプションを付加したSYNパケットを送信する. 本ツールではこれを検知し, SACKを有効とした解析を行うようにした. SACKオプションは再送動作を行わないので, SACKを検査し, SACKブロックにかかれたシーケンス番号で抜けているシーケンス番号のパケットが再送されているか検査する.

- On Estimating End-to-End Network Path Properties
  On Estimating End-to-End Network Path Properties[1]は再送が1回で終了した場合, その輻輳は軽微だと考え, cwndを直前のcwnd +MSSに戻す方法を提案している. 本ツールでは再送が発生した場合, 次に送出されるパケットが通常のパケットであればcwndを直前のcwnd+MSSに設定するように実装した.

- タイムアウトの検出
  TCPは送出したパケットのRTTがRTOの値を越える場合, そのパケットは損失したとしてcwndを1MSSに設定し, そのパケットを再送する. 最初のタイムアウトによる再送時に再送すべきパケットが複数ある場合, パケットを連続再送する. 連続再送中にACKを受信すると, cwndの値をACKを受信した回数×MSSに設定する. duplicate ACKによる再送, 新し

いパケットが送出された時は処理を終了する. 本ツールで上記の動作を解析できるように実装した. しかし, 本研究ではタイムスタンプから RTT の値を取得し, RTO の算出を行おうとしたが, RTO の推定精度が低いため, 多少の誤差がでている.

- 通信トレースを取得する場所
  例えば, ホスト A, B 間の通信をその間にあるホスト C で取得した場合, C-B 間で損失したパケットは C のトレースには残っているが, B には届いていないため, C の通信トレースと B の通信トレースが一致しない. B が再送を促す可能性があり, 正確な輻輳制御動作の解析ができなくなる可能性がある. このような問題を回避するため, シーケンス番号の検査行う. 受信したはずのシーケンス番号のパケットが送出されている場合, 同じシーケンス番号のパケットの中で最も新しいパケットを採用し, それ以外のパケットは無視し, cwnd の値を修正する方法を取る. ただし, ACK の損失に関しては Delay ACK などのオプションがあるため, 現時点では修正の対象にはしていない.

3 評価

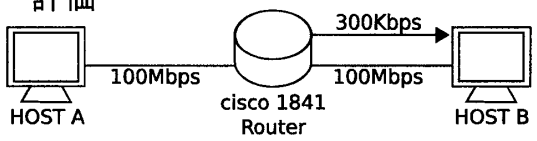

図 3: 実験環境

本研究ではツールの動作を確認するための簡易的な評価を行った. 実験環境を図 3 に示す. FreeBSD 7.0 の端末 2 台を用いる. また, A のカーネルを改造し, TCP のパケットに cwnd と ssthresh の値を挿入できるようにした. 解析対象の TCP は SACK を用い, タイムスタンプ, ウインドウスケールオプションを有効にした FTP 通信を使用する. A-B 間に設置したシスコ社製ルータ 1841 の機能を使って, ルータから B への帯域を 300Kbps に制限する. A から B へデータを送信し, A で tcpdump による通信キャプチャを行い, 解析を行った. cwnd と ssthresh の推定値と実測値をグラフ化し評価する. グラフの x 軸はパケット数, y 軸はそれぞれの値 (byte) である. 図 4 と図 5 は推定値と実測値の cwnd と ssthresh の比較を行っている. 図 4 を見ると, 前半部分では推定値と実測値が一致している. 200 番目のパケットのあたりでタイムアウトによる再送が発生している. タイムアウトによる再送中は cwnd と ssthresh の実測値に特殊な値が設定されるため, 推定値と一致しない. しかし, タイムアウトが発生すると cwnd が 1MSS から slow start を行うので, 推定値が正しい. 証拠に, 220 番目あたりでタイムアウトによる再送が終了し, 推定値と実測値が一致していることから推定値が正しいと言える. その後, 300 番目のパケットのあたりで RTO の誤差によってタイムアウトが発生したという間違った判定をしている. 以降, RTO の誤差が累積してしまい, 推定値と実測値が一致しない. 図 5 の始めで推定値と実測値が一致しないのは, 実測値をパケットに入れることができなかっただけで, 推定値が合っている. また, 200 番目のパケットのあたりでタイムアウトによる再送が発生しているため, ssthresh の値が変化している. その後は ssthresh の値が一致している. また, 300 番目のパケットのあたりで間違ってタイムアウトと判定しているのに推定値と実測値が一致しているのは偶然一致しただけである.

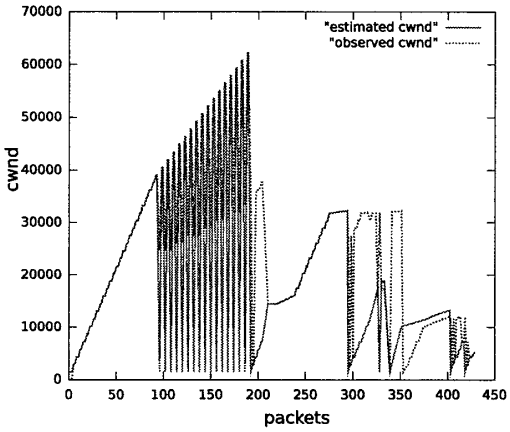

図 4: cwnd

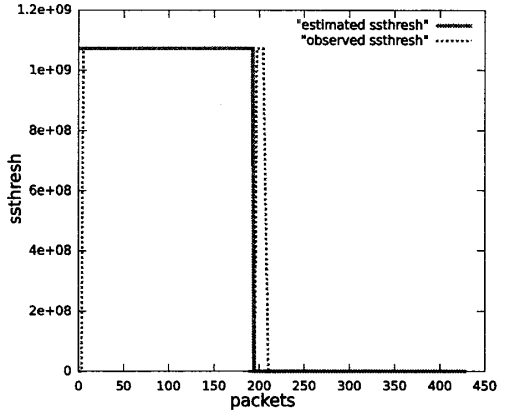

図 5: ssthresh

4 まとめと今後の課題

本研究では輻輳制御動作の自動解析を行い, cwnd と ssthresh を推定するツールを開発した. 実測値と推定値を比較し, RTO の誤差が発生しない場合, 輻輳制御動作の推定が正しく行われていることを確認した. 今後の課題としては, 実際のネットワーク上で実験を行うこと, RTO の推定精度を向上させ, 判定精度を上げることなどが挙げられる.

参考文献

[1] M.Allman V.Paxson,[http://www.icir.org/mallman/papers/estimation-la.pdf] "On Estimating End-to-End Network Path Properties" ACM SIGCOMM(2001)