2005－EVA－13（5）
2005／6／24

IP ヘッダの ID フィールドを用いた TCP 通信中の RTT のパッシブ測定による推定法

大坐畠 智 †　　鈴木 秀章 ‡　　川島 幸之助 †

† 東京農工大学 大学院 共生科学技術研究部 システム情報科学部門
‡ 東京農工大学 大学院 工学教育部 情報コミュニケーション工学専攻

概要　TCP 通信において Round Trip Time (RTT) は輻輳制御に不可欠であり, その長さは通信性能に大きな影響を与える. TCP 通信の性能評価において, 通信中の RTT を知ることは重要なことであるが, パッシブ測定による通信中 RTT の計測は困難であった. そこで, 本論文では同一輻輳ウインドウ内に複数のデータ, ACK セグメントがやりとりされている場合でも, 連続した IP ヘッダの ID フィールド値を TCP によるバースト転送と定義し，バースト間隔を測定する．このバースト間隔とサーバで測定した RTT とを比較することにより，その間隔が RTT として扱えるか検討，考察をする．

An Estimation Method for RTT on TCP Communication with IPID by Passive Measurement using IPID

Satoshi Ohzahata†, Hideaki Suzuki‡ and Konosuke Kawashima†

† Institute of Symbiotic Science and Technology, Tokyo University of Agriculture and Technology
‡ Graduate School of Engineering, Tokyo University of Agriculture and Technology

Abstract　In TCP communication, Round Trip Time (RTT) is indispensable for the congestion control and gives large impact for the communication performance. When we evaluate TCP performance, information of RTT during the communication is an important factor but it is difficult to measure by the passive measurement. In this paper we propose a passive measurement method for estimating RTT in communication by using IP's ID field. We define the back-to-back IPID number as burst traffic of TCP and measure the intervals. By comparing the inervals and measured RTT in the server, we discuss the relations between them.

1　はじめに

Transmission Control Protocol (TCP) は，インターネットにおけるエンド・トゥー・エンド制御による信頼性のあるコネクション型のプロトコルである．TCP 通信のトラヒック制御はウインドウフロー制御により行われ, RTT (Round Trip Time) はその性能に大きな影響を与える.

RTT は TCP 送信側がデータセグメントのヘッダ部分のシーケンス番号に対する ACK (Acknowledgment) セグメントを受信者から受け取ることにより, その間隔の時間として測定される．この測定方法はデータ送信者によるアクティブな測定により行われ, 通信経路上でのパッシブ測定による RTT の計測は難しく，ごく限られた場合での RTT の値の推定のみが可能であり，通信中の RTT の変化を把握することは不可能であった.

パッシブ測定による RTT 測定法はいくつか検討されている. 参考文献 [1] では TCP 通信を始める際のコネクションを確立する部分である 3 ウェイハンドシェイク部分, 及び, スロースタート部分に着目し, データセグメント, ACK セグメントのやりとりを測定することにより, RTT を推測している．この手法では, 1

回のコネクションにつき, 数回しか RTT を測定することができず, 定常的な TCP 通信中の RTT 測定は不可能であった.

文献 [2] では, TCP 通信中の RTT を推測するため, 通信経路上の測定地点において, それぞれの TCP コネクションの送信側の TCP 遷移状態を保持することにより, 輻輳ウインドウサイズを推測する. これにより, どの ACK セグメントにより, どのデータセグメントが送られたのかを推測することが可能になり, RTT を推測可能になる. しかしながら, 送信側の TCP の遷移状態を推測するには, TCP のバージョン (Tahoe, Reno, New Reno, Sack) をセグメントのやりとりで分ける必要があるが, これを見分けることが難しく, 正確な推定は困難である.

文献 [4] において，TCP のタイムスタンプオプション [5] を用いて，ACK とそのデータセグメントを対応づけて，RTT を推定する方式が提案されている．通信中でも RTT を推定可能な方式だが，タイムスタンプオプションはサーバ，クライアントの両方で対応していなければならなく，現在，広くは用いられていない．また，文献 [4] で，TCP のバーストトラヒックが “self-clocking” することに着目し，相関をとることにより，RTT を推定している．“self-clocking” 方式は，片方向のトラヒックのみの測定で推定可能である．しかしながら，RTT が安定した場合かつ，同一タイムスタンプが押される期間の 2 倍以上の RTT の場合のみに推定可能である.

これらの問題を解決するため, 本論文では, TCP の輻輳制御により，IP ヘッダの ID フィールド部分の番号が連続にふられていることに着目する．TCP ヘッダ部分のシーケンス番号はコネクション毎に連続にふられるが, IP ヘッダの ID は送信ノードがコネクションに依存せず, 送信の際に連続に割り当てられる．このことを用いて, データセグメント転送のバースト間隔を測定し，その間隔とサーバで測定した RTT と比較することにより，その間隔が RTT として扱えるか検討，考察をする.

本論文は以下のように構成される．2 章で RTT 推定方法を説明する．3 章でトラヒック測定方法, 4 章で解析結果を述べ, 5 章で本論文をまとめる.

2 RTT 推定方式

TCP 通信における RTT は送信側がデータセグメントに対する ACK セグメントを受信することにより，計測されている．よって，送信者以外が RTT を測定することは一般には難しい．この節で，パッシブ測定による RTT 推定方式を議論する.

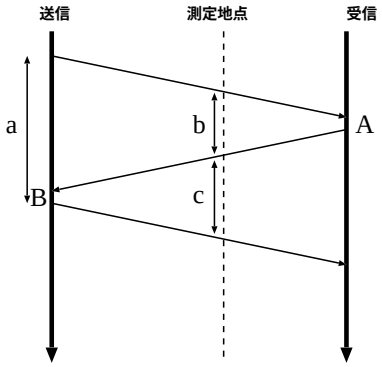

図 1: RTT 推定方式 1.

2.1 パッシブ測定による RTT 推定方式

図 1 に RTT 推定法を示す．TCP の輻輳制御に用いる RTT 計測は送信側で行われ, 送信したデータセグメントに対する ACK を受信するまでの時間である (図 1 期間 a)．しかしながら, この RTT 計測はデータ送信者のみでしか行うことができず, 通信経路上のトラヒック測定地点, 及びデータ受信側では計測できない.

そこで, 通信経路上でのパッシブ測定による RTT 推定方法が提案されている [1]. TCP 通信確立時に行われる 3 ウェイハンドシェイク, データセグメント送信開始時のスロースタートフェイズでは, 同一の輻輳ウインドウ内に 1 もしくは数セグメントのみを送信する時がある．この時, 測定ポイントにおける送信セグメント通過時から, その ACK セグメントの返信が通過するまでの時間 (図 1 の期間 b) と ACK セグメントを受け取って新しいデータセグメントが測定ポイントを通過するまでの時間 (図 1 の期間 c) を加えたものを RTT と定義している．1 コネクション当たり, この ACK，データセグメント間の対応関係が明らかなスロースタートフェイズの数回のみの RTT 計測を可能としている [1]．TCP 通信中は同一輻輳ウインドウ内に複数のデータセグメント, ACK セグメントのやりとりを行うので, この方法で TCP 通信中の RTT を推定するのは不可能である.

図 2 に示すような輻輳回避フェイズの通信中でも，送信側がどの ACK セグメントに対してデータセグメントを送信しているか特定できれば，図 1 と同じく, RTT を推定することが可能である．しかしなが

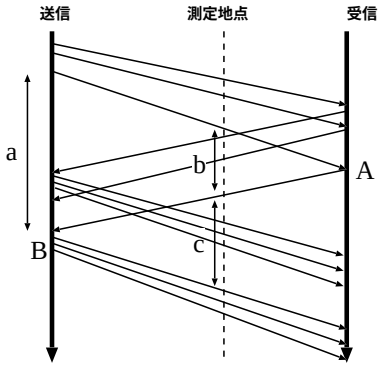

図 2: RTT 推定方式 2.

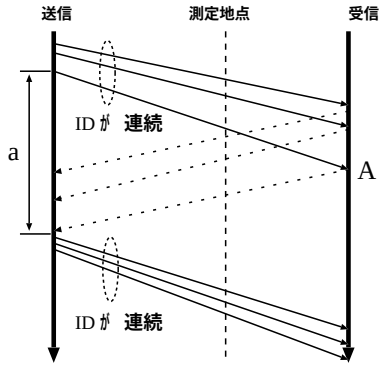

図 3: 提案バースト間隔解析方式 1.

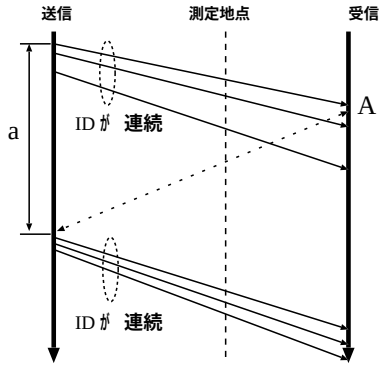

図 4: 提案バースト間隔解析方式 2.

ら, 測定地点でのこの対応関係を把握することは難しい. これは, TCP がウインドウフローコントロール (go-back-N, N: 最大輻輳ウインドウサイズ) により, トラヒックが制御されていることに起因する. これにより, ある ACK セグメントを受け取ると n ($n \leq N$) セグメントが一度にネットワークに送出される. これにより, 1 つの ACK セグメントに対して複数のデータセグメントが送られることになり, 前述の RTT 推測方法が適用できない原因となる.

そこで文献 [3] では，アプリケーションレベルで用いられる 3 ウェイハンドシェイク部分に着目し，TCP 通信中の RTT を推定を可能としている．文献 [2] では, TCP 通信中の RTT を推測するため, 測定ポイントにおいて, それぞれの TCP コネクションの送信側の TCP 遷移状態を保持することにより, 輻輳ウインドウサイズを推測する. これにより, どの ACK セグメントにより, どのデータセグメントが送られたのかを推測することが可能になり, 上記 RTT での推測が可能となる. (図 2 の A，B を特定) しかしながら, 送信側の TCP の遷移状態を推測するには, TCP のバージョン (Tahoe, Reno, New Reno, Sack) をセグメントのやりとりからそれぞれのアルゴリズムを見分けるか, TCP ヘッダの Sack オプション等で見分ける必要があるが, これらバージョンを見分けることが難しく, 正確な推定は困難である.

文献 [4] では，TCP のタイムスタンプオプション [5] を用いて，ACK とそのデータセグメントを対応づけて，RTT を推定する方式が提案されている．高速回線で大きなデータを通信する際，TCP のシーケンス番号が一周し，重複する可能性がある．これを防ぐためタイムスタンプオプションは一定期間，同一番号をオプションとして付加してデータセグメントを送信する．そして，タイムスタンプを受け取った側はその番号をエコーバックすることによって，シーケンス番号が重複なのかどうかを見分けることを可能にする．これを用いることにより，図 2 の A–B 間の ACK セグメントとデータセグメントの対応関係の識別をタイムスタンプが変わる時点で可能としている.

2.2 提案バースト間隔解析方式

文献 [4] で，TCP のバーストトラヒックの起きる間隔が “self-clocking” することに着目し，その間隔の相関をとることにより，RTT を推定している．しかしながら，RTT が安定した場合かつ，同一タイムスタンプが押される期間の 2 倍以上の RTT の場合のみに推定可能という制限がある．そこで本論文では, 同一ホストから IP データグラムが送信される際, IP ヘッダの ID フィールドの値は連続した値がふられることに着目する. (近年この事実を用いてトラヒックの解析が行われている [6], [7]) 本論文では，TCP のバーストトラヒックを連続した IPID 値で定義，その値の不連続期間を測定し，送信側で測定した RTT との比較を行う.

測定解析を行うにあたり，2 種類のバースト間隔を定義を提案する．提案バースト間隔解析方式 1(図 3) では，バーストの最後と次のバーストの最初をバースト間隔と定義し，提案バースト間隔解析方式 2(図 4) では，バーストの最初と次のバーストの最初をバースト間隔と定義する．どちらの方式でも図の中の A からの ACK セグメントをうけて，次のデータセグメントのバースト転送が起きることを仮定していることになる．両方式とも “次のバースト” が連続 n セグメント以上あった場合のバースト間隔を採用するものとする．

本提案方式の制限事項として，下記があげられる．サーバでは複数セッションの通信を行っていなければならない．本来，IPID が連続な部分のセグメントロスがあると，バースト間隔を誤ることがある．本方式は IPID 値の設定の実装方式, 及び, TCP 層から IP 層にセグメントが送られる際のセグメントの転送する際の各 OS の実装方式に依存し, 適用できない場合がある．

3 トラヒック測定，解析方法

測定は，LAN 環境 (ping による RTT が 1ms 以内)，WAN 環境 (ping による RTT が数十 ms) で行った．HTTP サーバ (FreeBSD 4.11 Stable, Apache 2.0) に 100MB のファイルを用意し，クライアント PC(FreeBSD 5.3 Stable) から wget コマンドを用いてダウンロードする．IPID を不連続にするため，測定 PC にて同時に 2 回 wget コマンドを実行した．測定は，サーバ，及び，クライアントで Snort[8] を用いて行った．

サーバでアクティブに測定した RTT，クライアント PC で測定されたログをパッシブ測定されたものとして同一フローのログの解析を行った．パッシブ測定による解析方式として，まず，前節の提案方式 1, 2, そして FreeBSD 4.11，FreeBSD 5.3 はタイムスタンプオプションに対応しているため，タイムスタンプを用いた RTT 推定法 [4] を用いても解析，比較を行う．両提案方式では，“次のバースト” が連続 8 セグメント以上あった場合のバースト間隔を採用するものとした．サーバで測定した RTT，タイムスタンプを用いた方式では，パケットロスが生じた場合の RTT の値は除外した．

4 解析結果

4.1 LAN 環境

図 5–8 に LAN 環境における提案バースト間隔解析結果を示す．サーバで測定した RTT をみると，LAN 環境においても RTT は大きく変動していることがわかる．これは TCP 制御によるバースト転送により，自ら輻輳をつくり出していることが考えられる．データセグメントの転送間隔から，IPID 値が連続となる小さなバースト転送 (ms オーダ) と，更に大きな周期のバースト転送 (s オーダ) があることがわかる．

図 5，6 より，提案バースト間隔解析方式 1 による解析では，バースト間隔が数 ms から数秒まで開いていることがわかる．0.008 s 付近で提案バースト間隔解析方式 1 とタイムスタンプを用たいた方式と類似の傾向が見られる．しかしながら，RTT の変動には対応しておらず，他の数値付近では，一致している箇所が少ない．

図 7，8 より，提案バースト間隔解析方式 2 は提案方式 1 に比べ，RTT が大きな場合サーバでの測定値と一致するが，RTT が小さな場合，ほとんど一致しないことがわかる．

タイムスタンプを用いた RTT 推定法は RTT が数 ms 程度の小さな値を除いて，比較的よく RTT を推定できている．推定できていない範囲は，同一タイムスタンプが用いられる期間に依存していることが考えられる．

4.2 WAN 環境

図 9–12 に WAN 環境における提案バースト間隔解析結果を示す．サーバで測定した RTT をみると，WAN 環境においては LAN 環境に比べ，RTT の変動幅が小さくなっていることがわかる．これは最小 RTT が大きくなり，TCP 制御によるバースト転送がおこりにくくなり，自ら輻輳をつくり出すことが少なくなっていることが考えられる．

図 9–10 から，提案バースト間隔解析方式 1 は，RTT とは関係のない小さな期間をあけて転送され，一致する箇所が少なくなっている．しかしながら，図 11–12 により，今度は，提案バースト間隔解析方式 2 が比較的 RTT と同じ値をあけて転送されていることがわかる．

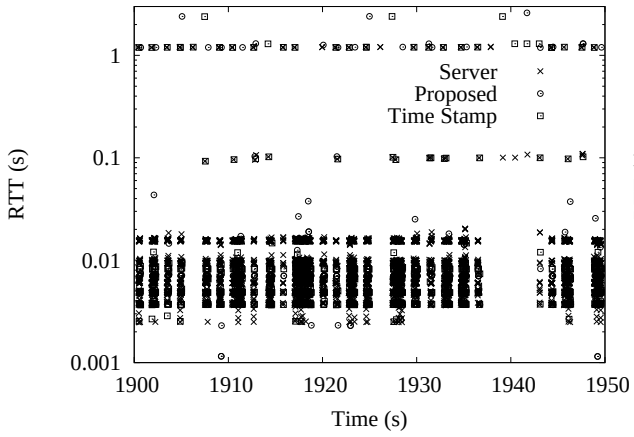

図 5: 提案バースト間隔解析方式 1 (LAN, 1900–1950 s).

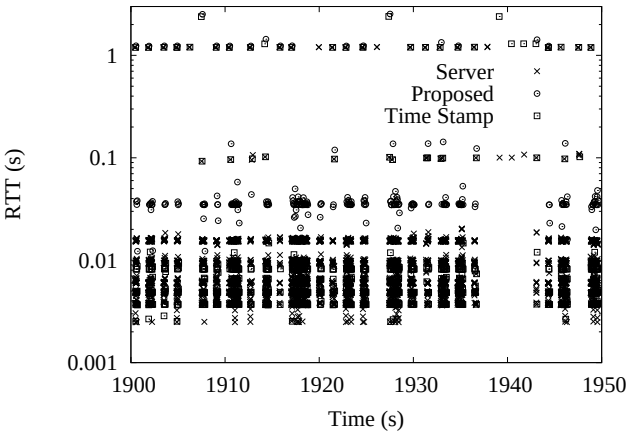

図 7: 提案バースト間隔解析方式 2 (LAN, 1900–1950 s).

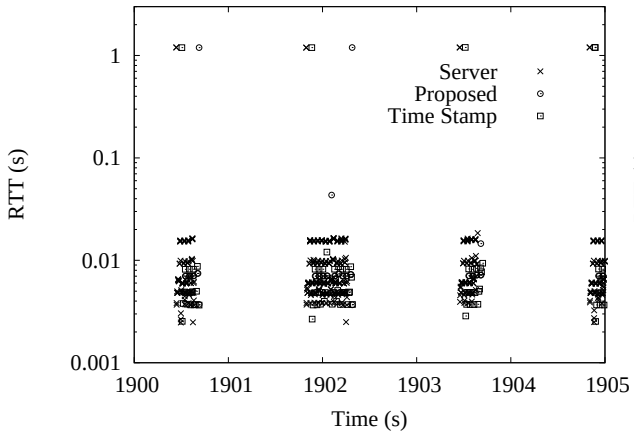

図 6: 提案バースト間隔解析方式 1 (LAN, 1900–1905 s).

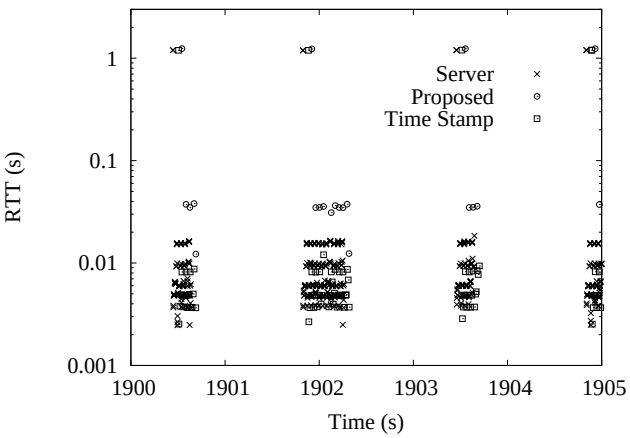

図 8: 提案バースト間隔解析方式 2 (LAN, 1900–1905 s).

5 まとめ

本論文でパッシブ測定を用いた TCP 通信中におけるバースト間隔解析方式を提案した. 実機を用いたシミュレーション実験により，提案した方式とサーバで計測した RTT と比較を行った．その結果，LAN 環境では提案バースト間隔解析方式 1 が，WAN 環境では提案バースト間隔解析方式 2 がサーバで計測された RTT に比較的近い値を示した．このことから，IPID を解析することにより，TCP 通信中の RTT 推定の可能性を示した.

今回，提案方式では，“次のバースト” が連続 8 セグメント以上あった場合のバースト間隔を採用するものとしたが，この値の妥当性も評価する必要がある．

今回提案した方式を改良し，RTT を見積もることで，滞留セグメント，輻輳ウインドウサイズが可能になり，TCP 通信の品質評価が可能になる．更に実時間トラヒック測定ツールの開発, TCP 通信の性能評価, トラヒック制御方式の開発なども今後の課題として考えられる.

参考文献

[1] H. Martin, A. McGregor, and J. Cleary, *“Analysis of Internet Delay Times,”* Proc. of Passive and Active Measurement Workshop ’00 (PAM2000), 2000.

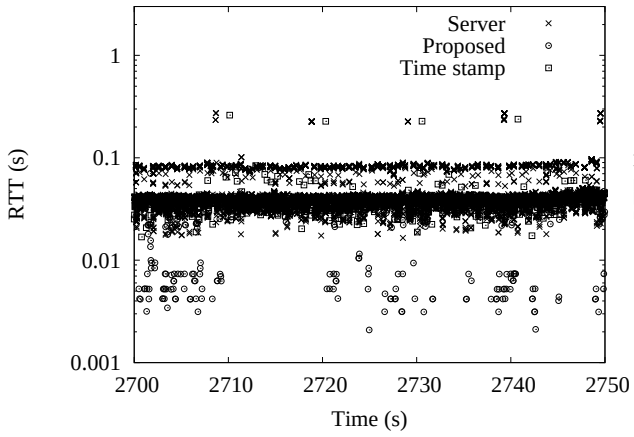

図 9: 提案バースト間隔解析方式 1 (WAN, 2700–2750 s).

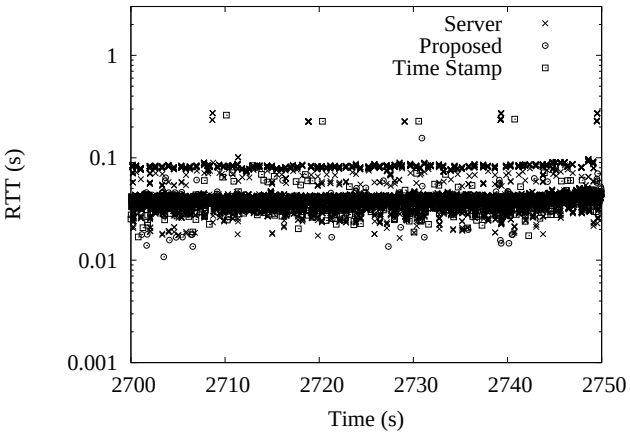

図 11: 提案バースト間隔解析方式 2 (WAN, 2700–2750 s).

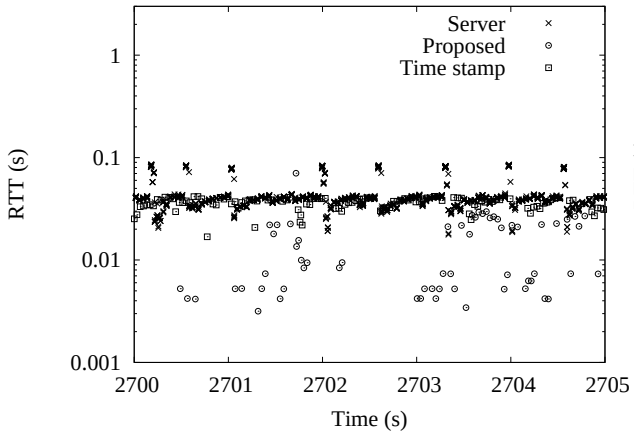

図 10: 提案バースト間隔解析方式 1 (WAN, 2700–2705 s).

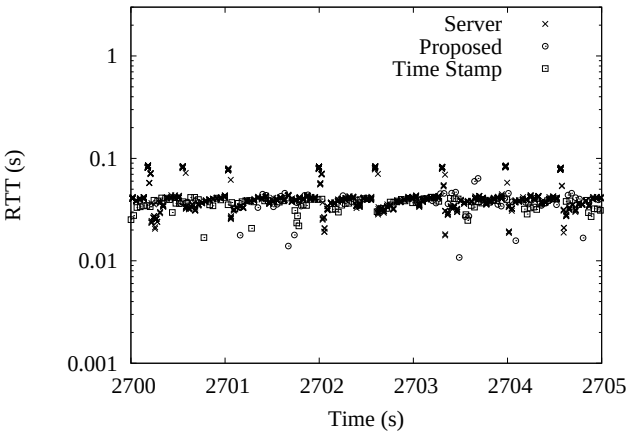

図 12: 提案バースト間隔解析方式 2 (WAN, 2700–2705 s).

[2] S. Jaiswal, G. Iannaccone, C. Diolt, J. Kurose and D. Towsley, *"Inferring TCP Connection Characteristics Through Passive Measurements,"* Proc. of IEEE INFOCOM'04, 2004.

[3] H. Jiang and C. Dovrolis, *"Passive Estimation of TCP Round-Trip Times,"* ACM SIGCOMM Computer Communications Review, Vol. 32, Num. 3, 2002.

[4] B. Veal, K. Li and D. Lowenthal, *"New Methods for Passive Estimation of TCP Round-Trip Times,"* Proc. of Passive and Active Measurement'05 (PAM2005), pp. 121–134, 2005.

[5] V. Jacobson, R. Braden and D. Borman, *"TCP Extensions for High Performance,"* RFC 1323, 1992.

[6] S. M. Bellovin, *"A Technique for Counting NATed Host,"* Proc. of ACM IMW'02, pp. 267–272, 2002.

[7] W. Chen, Y. Huang, B. F. Ribeiro, K. Suh H. Zhang E. S. Silva, J. Kurose and D. Towsley, *"Exploiting the IPID field to infer network path and end-system characteristics,"* Proc. of Passive and Active Measurement'05 (PAM2005), pp. 108–120, 2005.

[8] Snort, http://www.snort.org.