PREEMPTIVE RETRANSMISSION - IMPROVING THE PERFORMANCE OF SHORT-LIVED TCP CONNECTIONS

Alex J. Tarr, Bogdan V. Ghita
Network Research Group
University of Plymouth
Plymouth, UK
email: alex@alextarr.co.uk, bghita@plymouth.ac.uk

ABSTRACT
This paper presents a method of improving the performance of TCP (Transfer Command Protocol) based short-lived connections, such as web transfers. The characteristics (throughput and round trip time) experienced while communicating to a selection of Internet sites were examined, revealing that path constancy can be observed over extended periods of time. The research identified a temporal correlation between the three-way handshake and subsequent measurements of the RTT; by using this correlation, the algorithm is able to detect the signs of a packet loss and retransmit the affected segment before a standard implementation of TCP is able to recover the lost segment. Following the proposal of the algorithm, simulations were performed to study the performance of the proposed algorithm. In summary, during these simulations it was found that use of the Preemptive Retransmission algorithm provided enhanced performance, with the average duration of a connection 8% shorter than when compared to TCP Reno.

KEY WORDS
Networking, TCP Congestion Control Mechanisms, Short-Lived TCP Connections

1 Introduction

Web transfers, i.e. HTTP connections, can be characterised by the small amounts of data they transfer. With the average HTTP connection transferring less than 100 kBytes, these transfers typically last for less than a second and are known as "short-lived connections". TCP is already adept at handling the wide variety of network conditions which may be experienced. However, with the increasing volume of short-lived connections, the pottential to improve the performance for this type of connection within TCP should be explored.

The research presented in this paper aims to investigate the current state of the Internet by collecting a set of traces for web transfers and establishing the trends which can be observed. The findings of this investigation will form the basis of an algorithm to enhance the performance of TCP connections, especially those of a short-lived nature. Therefore, the proposal that results from this paper can contribute to improving the performance of the Internet, especially for web-browsing activities.

The paper is organised as follows: section 2 presents the background literature and current algorithms in TCP. Section 3 details the method used to collect traces of real network short-lived connections, with section 4 detailing the analysis of the characteristics that occurred during the captured traces. In section 5 the Preemptive Retransmission algorithm is proposed. Section 6 introduces the tools that will be used to validate the proposal, with section 7 presenting the findings of the simulations revealing the performance improvements offered by the proposed algorithm. Finally, section 8 presents the overall conclusions and potential directions for future work.

2 Related Work

The TCP behaviour is very closely linked to the Internet characteristics. Throughout the history of TCP, the protocol-related variables were adjusted to reflect current conditions of traffic and network performance. A landmark study performed to establish long-term network behaviour was performed in [13], where the research focused on how well past values can be used to predict future path characteristics. The paper also investigated the concept of path constancy, in order to indicate how such characteristics evolve over time. Continuing the earlier experiments in [10], Zhang's study explored three definitions of constancy: mathematical, operational and predictive; whereby a path could display any combination of these, e.g. appearing statistically non-constant while being entirely predictable in its future performance. The current work uses the concept of mathematical constancy when examining the collected traces. Zhang's work, based upon data collected between 1999 and 2001, identified constancy which existed for period in excess of several minutes; throughput displayed change free regions of up to 20 minutes, with delay constant for periods up to 30 minutes. In order to verify the efficiency of the proposed approach, this study will be based on the levels of constancy, as seen in 2005.

Although heavily influenced by the network parameters, the core TCP functionality relies upon the algorithms presented in [1]. One of the algorithms, Slow Start, exponentially increases the amount of in-flight data until the

congestion window (cwnd) exceeds the Slow Start threshold (ssthresh), after which congestion avoidance is invoked. A TCP connection therefore starts by using Slow Start until the network capacity is reached, as indicated by a lost segment. TCP detects losses using a timeout that occurs when a segment has not been acknowledged within a period of time determined by the observed RTT; to reduce the time taken to detect a pack loss, Fast Retransmit uses the receipt of three duplicate acknowledgements to signal that a retransmission is required. Importantly, Fast Retransmit therefore requires at least three packets to be transferred following the lost segment.

Finally, in terms of implementations, TCP evolved a long way. After the early (1983) BSD 4.2, which led to "congestion collapse", TCP Tahoe followed in 1988 with congestion control mechanisms and congestion detection through packet loss, then TCP Reno in 1990 improved congestion control through the recovery phase. TCP Reno was followed in the mid-1990 by TCP New Reno, which includes all four congestion algorithms [6] . More recently, TCP Vegas [4] has been proposed to estimate the maximum transmission rate while avoiding any packet losses, but it has yet to gain acceptance by the Internet community. Current OS implementations use Reno-based implementations of TCP, with NewReno [2] and SACK [5] being the typical candidates.

3 Trace data collection

To investigate the behaviour of TCP connections a collection of traces was obtained upon which further analysis could be performed. The analysis will answer two important questions: "What level of constancy is experienced in the Internet (at the moment of the collection)?" and "Are there any correlations between the characteristics of a connection?"; by answering these questions an improved understanding of the behaviour of short-lived TCP connections can be gained. As the main focus of the research is TCP improvement, the evaluation of the Internet path constancy will not constitute a second aim of the study, but only a vehicle for introducing the proposed changes.

The collection technique aimed to recreate a realistic usage scenario; this was achieved by monitoring HTTP transfers with using automated requests which provided a predictable stream of connections, and the ability to examine the behaviour of a site over a prolonged period. The homepage of a selection of sites were requested using the wget [12] tool at one minute intervals, with the packet streams captured using `tcpdump` [11], and the resulting files were stored for post-processing. The selected sites represented a range of genres e.g. shopping, news etc. in the UK and US. The sites used were: BBC, BBC News, CNN, DigitalSpy, Exeter University, Google UK, Tesco, The Register and Yahoo UK. Although the selected sites do not capture the complexity and variety of the Internet, they do however provide an indication of the behaviour of Internet paths.

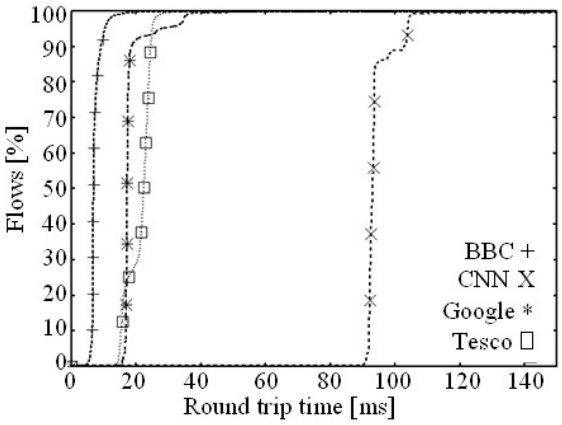

Figure 1. Cumulative distribution of the RTT for the collected data sets

The HTTP requests were sent to the servers IP Addresses rather than the host name of the web site. Although an end user typically connects to a web site using the host name, the DNS look-up was considered an external system and outside the scope of this investigation. The rational for using the IP Address was made to avoid potential distortion of the results which may occur if using a host name; for example if load-sharing or differing server configurations existed. To ensure the RTTs observed were accurate and had not been skewed by the presence of a caching proxy server, the RTTs were also sampled using ICMP ping. The times returned by ping correlated closely with the RTT measured from the HTTP requests, with the majority of connections exhibiting a ratio between TCP and ping RTTs of 1.0 to 1.1. Finally, to ensure the results reflected a wide range of situations, more than 250,000 connections were collected between April and June 2005. The traces were collected from two locations, both Linux systems located on the university campus network, which is connected to the Internet via the UK Joint Academic Network (JANET).

4 Analysis of Collected Traces

Based upon the collected data, the `tcptrace` [8] tool was used to perform per-flow analysis, with the resulting characteristics of each connection extracted into a CSV file. The output from tcptrace was then used to establish the period for which constancy could be observed, and presence of any trends or correlations in the characteristics of the traces.

4.1 Path Constancy

Constancy of a path was assessed through the behaviour of RTT and throughput, as they are critical in affecting the overall performance which an end-user will experience. The cumulative distributions of the RTT and throughput for four of the sites monitored are presented in Figure 1- 2

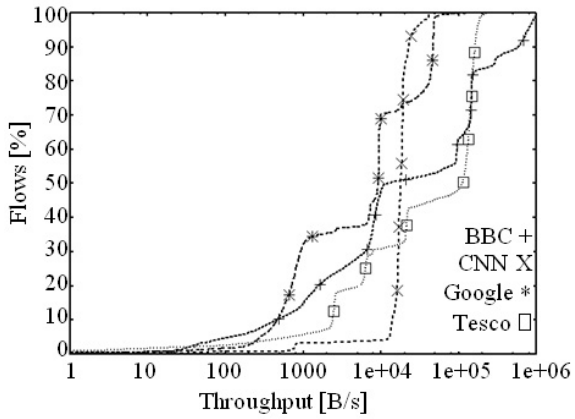

Figure 2. Cumulative distribution of the throughput for the collected data sets

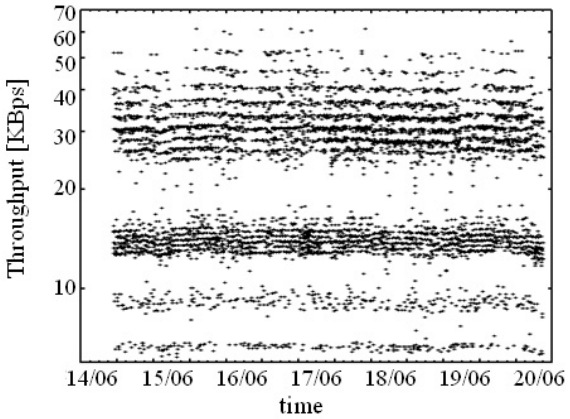

Figure 3. Throughput for **yahoo.co.uk** over time

As can be observed in Figure 1, the majority of connections experienced relatively constant RTTs, the level of constancy can be confirmed with standard deviations of 1.8ms, 5.5ms, 3.63ms and 3.6ms for BBC News, CNN, Google and Tesco respectively; these values illustrate the small variation in RTTs which the paths experienced. In all cases, the RTTs were almost identical for the duration of the trace collection with few outlying results observed.

The throughput, as shown in Figure 2, experiences increased variability over time when compared to that experienced for the RTT. Most sites demonstrated clustering of throughput around a number of distinct levels over the observation period, with these levels occurring concurrently for the entire length of the trace, rather than at different throughputs for disjoint periods. The high levels of variation can be confirmed by observing the standard deviations of the sites, the values returned were: 241kBytes/s, 6kBytes/s, 19kBytes/s and 68kBytes/s respectively for BBC News, CNN, Google, and Tesco.

4.2 Throughput Analysis

As observed in the previous section, the throughput of the captured traces occurred at a number of levels for the duration of the data collection sessions, in this section the cause of these different levels are investigated. The investigation focused on the Yahoo UK site which exhibited 11 main throughput levels causing a layered appearance over time, as presented in Figure 3

Examining the number of packets tcptrace reported the workstation as receiving identified variation between 22 and 32 packets. Part of this variation is due to a fluctuating page size, which accounts for between 22 and 25 packets being observed. Examining the throughput, a connection receiving 25 or fewer packets achieved between 20 to 60kBps; while connections receiving more than 25 packets experienced throughputs of less than 20kBps. Therefore, if more than 25 packets were received, some of these had been retransmitted. The characteristics from tcptrace revealed that each connection sent, on average, nine duplicate acknowledgements to the server; this illustrates the high level of either re-ordering or loss of the studied path.

By examining the maximum idle-time (the longest period between two packets in the same direction), it can be observed that a connection generally achieves a lower throughput the longer the maximum idle-time, see Figure 4. When a loss occurs, the maximum idle-time represents the time taken to detect, and resend the segment. It could therefore be observed that a large number of connections experienced an idle-time greater than 500ms, therefore indicating that a timeout had triggered the retransmission rather than Fast Retransmit. As a connection which experiences a timeout achieves significantly lower throughput, the detection of lost packets is therefore a key factor in determining the performance of a connection.

These investigations illustrated how TCP and the Fast Retransmit mechanism can fail when insufficient packets exist to maintain the return data flow. The Fast Retransmit algorithm requires three duplicate acknowledgements to be received to trigger a retransmission; therefore, if less than three packets are sent following a lost packet, Fast Retransmit is unable to recover the loss and a timeout must be used to correct the loss.

4.3 Three-Way Handshake Observations

Examining the RTT statistics reported by tcptrace, a link between the RTT for the three-way handshake, and the subsequent RTTs experienced over the connection was observed. The ratio between these values was computed and plotted in a cumulative distribution, as shown in Figure 3; analysing the ratios reveals that more than 80% of connections demonstrated how the three-way handshake value accurately predicted the average RTT within 10%. Further, 98% of the connections experienced a maximum RTT of three times or less than the original three-way handshake predictor. It was therefore concluded that the three-way

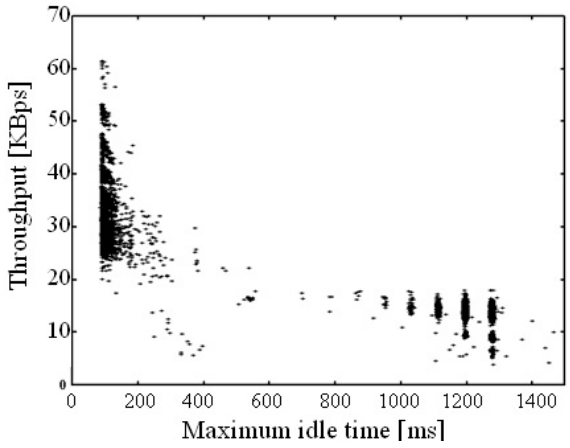

Figure 4. Throughput for **yahoo.co.uk** against the maximum idle period for the connection

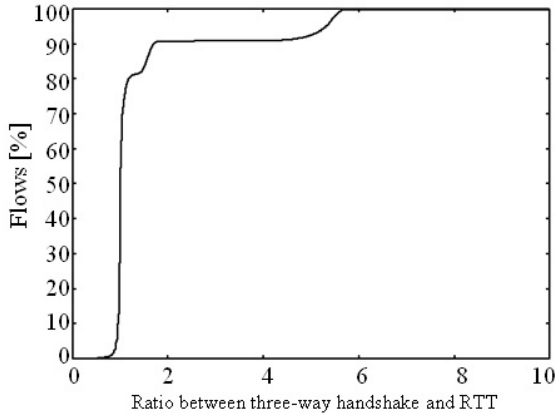

Figure 5. Cumulative distribution of the ratio between the three-way handshake RTT and subsequent RTT.

handshake could be successfully used to predict the subsequent RTTs that a connection will experience.

Figure 5 exhibits a number of connections with a ratio between five and six; these values all originated from the DigitalSpy site which contains a forum, and were the result of a large (more than 15 times the other RTTs for that connection) delay between the HTTP request and response. As a result, this site was excluded from further investigations as it was not typical of the values obtained for the other sites and the large variations are likely to have been caused by implementation issues on the server.

5 The Preemptive Retransmission Algorithm

The proposed algorithm improves the performance of short-lived connections by detecting a loss prior to either Fast Retransmit or the standard TCP timeout mechanism. The value for the Preemptive Retransmission Timeout is measured between the "SYN" message of the three-way handshake and its corresponding acknowledgement. By using the relationship identified between the three-way handshake RTT and subsequent RTTs of the connection, an unacknowledged segment can be assumed lost prior to the standard TCP timeout mechanism. When the Preemptive Retransmission Timer expires due to an acknowledgement having not been received, the first unacknowledged segment is retransmitted.

It is unrealistic to expect all packets to be acknowledged within the same RTT as identified during the three-way handshake RTT; therefore, the proposed algorithm includes a "multiplication factor" parameter to control the sensitivity of the algorithms timeout period. The largest ratio between the three-way handshake RTT and the maximum RTT of the captured traces was three, therefore this value is suggested. The three-way handshake RTT will be multiplied by the multiplication factor to determine the timeout used to generate a retransmission. For example, the Yahoo traces had a mean RTT of 84ms, thus the Preemptive Retransmission algorithm would set the timeout to three times 84ms, thus 252ms; this would therefore at least halve the maximum idle-time experienced by a connection compared to the standard TCP retransmission mechanism.

As the algorithm can potentially generate unnecessary retransmissions if the RTT should alter over the life of a connection, another parameter determines a maximum volume of data that the proposed algorithm functions over, this is known as the "cut-out limit". Based upon the average page size of 72kBytes observed in [3], the algorithm uses a cut-out limit of 80kB. Following the successfully acknowledgement of 80kByte's of data, the algorithm is disabled and generates no further retransmissions.

In summary, the Preemptive Retransmission algorithm functions as follows:

- **Step 1** - Measure the RTT during the three-way handshake phase to determine the timeout period based upon the multiplication factor.
- **Step 2** - When sending a segment a timer is started for the calculated timeout period, while less than 80kBytes of data have been acknowledged.
- **Step 3** - For each new acknowledgement, the timer is either cancelled if outstanding data is cleared, or restarted if unacknowledged segments still exist.
- **Step 4** - If the timer expires, the first unacknowledged segment is retransmitted. The timer is not restarted until a new segment is transmitted.

6 Validation Environment

To asses the improvement in performance offered by the proposed algorithm, a number of simulations were performed using the `ns2` network simulation tool [7]. Within ns2, the `PackMime` [9] traffic generation tool was used

to produce aggregate Internet packet traffic over a link. PackMime uses a stochastic model to control the connection initialisation rate and a heavily-tailed HTTP response size distribution. This provides a simulation environment similar to that experienced on the Internet. The simulations were all based upon an underlying 5Mbps bottleneck link, chosen to allow realistic volumes of aggregate traffic yet providing suitable simulation durations; the delay was set to either 10ms or 60ms representing UK and US sites respectively, with the delays having been verified through traceroute. PackMime starts connections at a specified rate where the selection of value is used to control the utilisation seen over the bottleneck link. The Preemptive Retransmission algorithm was implemented by modifying the implementation of TCP Reno (named Full-TCP), included with ns2. This allowed observation of the difference in performance between the modified implementation and the standard TCP Reno provided with ns-2 to establish the effect to the proposed algorithm on performance.

7 Simulation Results

The first simulation performed aimed to observe the general effects of utilising the Preemptive Retransmission on an environment with a 10ms link delay, and a utilisation of 50% over the bottleneck link, chosen to represent a moderately congested link. The simulations were performed twice, once using the original TCP Reno from ns-2, then with the modified TCP Reno implementation; therefore allowing the performance to be compared. The results of the first simulations proved extremely encouraging: the average connection duration under TCP Reno was 297ms, figure that dropped to 268ms when Preemptive Retransmission was used. This equates to an 8% improvement in performance when using the proposed algorithm. Further simulations confirmed the behaviour of the algorithm under a variety of scenarios and to the validity of the parameters, i.e. cut-out limit and multiplication factor, which control the behaviour of the algorithm.

7.1 Network Conditions

In the simulated network conditions, the effect of different network characteristics upon the performance improvement offered by Preemptive Retransmission were investigated. The impact of delay was initially examined, by considering two scenarios to represent a connection to a UK and US sites, using one-way delays of 10ms and 60ms respectively. As can be seen in Figure 6-top, the algorithm improved performance under both scenarios, although a larger improvement was observed for the 10ms delay. Both simulations occurred with 25% utilisation. Comparing the simulations using TCP Reno and Preemptive Retransmission, the proposed algorithm improves the average connection duration by 39% for a delay of 10ms, and 8% for a delay of 60ms. The significantly lower improvements for a

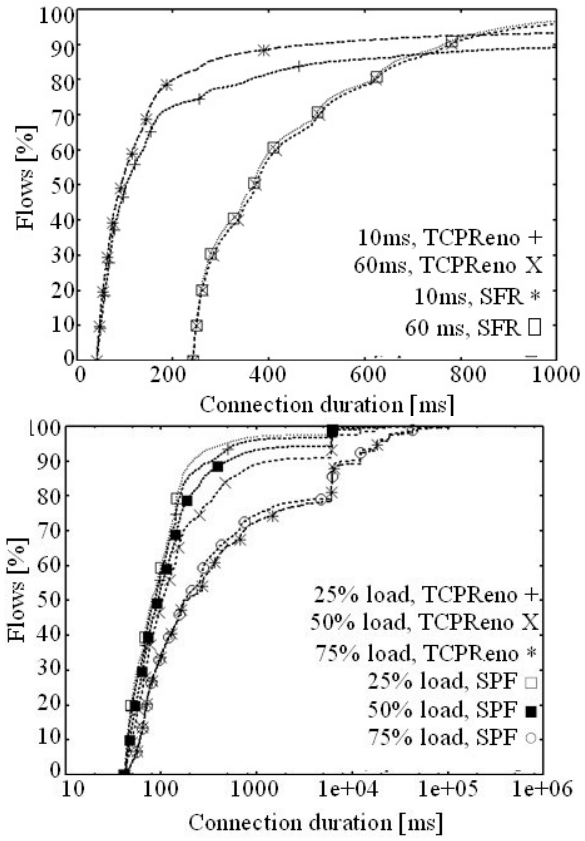

Figure 6. The effect upon the performance improvement of the Preemptive Retransmission algorithm under varying: top - network delays; bottom - traffic levels

network with a delay of 60ms can be explained as the timeout used by Preemptive Retransmission is set at 360ms, compared to 60ms on a network with a delay of 10ms. Therefore, on the 60ms network, the proposed algorithms timeout period is closer to the standard timeout mechanism used by TCP, this reduces the time benefit of using the proposed algorithm compared to TCP Reno.

The effect of background traffic levels on the performance of Preemptive Retransmission was also investigated by generating utilisation levels of 25%, 50% and 75% over the backbone link in the simulation. The results, as shown in Figure 6-bottom, indicated that the algorithm improves the performance of a connection under all scenarios. The largest improvement in performance was observed for 50% utilisation; where the connections are experiencing numerous losses due to congestion, but critically the link is not completely saturated thus by the time Preemptive Retransmission retransmits the lost segment, sufficient capacity exists to allow the segment to transmit successfully. The resulting improvements in performance, based upon the average duration of a connection, showed Preemptive Retransmission was 23%, and 38% and 8% faster for utilisations of 25%, 50% and 75% respectively.

7.2 Algorithm Parameters

The proposed algorithm contains two configuration parameters: the multiplication factor and the cut-out limit. Simulations were therefore performed to identify the optimum configuration of the algorithm. The results identified that the highest performance increase occurred when a multiplication factor of two was used. This finding is the result of lower variations than normally occur due to the absence of application layer interactions within ns-2 and PackMime. Surprisingly, the simulation reported the second best value as a multiplier of four with practically identical average connection durations but considerably fewer retransmissions caused by Preemptive Retransmission. The cut-out limit responded as expected, with a larger limit allowing more connections to benefit for their entirety from the proposed algorithm. Importantly, as the cut-out limit increased, the relative performance improvement decreased, therefore it was concluded that a limit of 80kB represents the optimum value based upon the average web page size.

7.3 Summary

The proposed algorithm significantly improves the performance of short-lived connections, even under situations such as long delay links and high traffic loadings. The results presented indicate the algorithm reduces the average duration of a connection by more than 8% and, importantly, the performance is never lower than TCP Reno's.

8 Conclusions and Future Work

This paper proposed an addition to the current TCP algorithms, named Preemptive Retransmission, which causes the first unacknowledged segment to be retransmitted following a period equal to three-times the original three-way handshake RTT. The proposed change was based on the results obtained from a set of packet traces, collected in order to establish the characteristics of short-lived TCP connections. The packet traces revealed that current TCP implementations are typically unable to recover from a lost packet when less than three packets follow the missing packet, therefore insufficient information is returned which would allow timely recovery from the loss. The analysis of the collected traces revealed that constancy is evident over extended time periods, which extend to several days, suggesting that current Internet backbones have sufficient capacity to avoid congestion occurring; critical for these findings, a relationship between the three-way handshake and subsequent RTT values was also established.

To validate the proposed algorithm, a series of simulations were performed under the ns2 network simulator to examine the algorithms performance under a variety of network conditions. The results of these experiments indicated a decrease in the average connection duration that was in excess of 8%, with a maximum observed improvement of 39% over a moderately loaded link.

Future work should aim to improve the proposed algorithm when multiple packet losses have occurred, a scenario which is currently handled poorly by the proposed algorithm; for example this problem is particularly evident in long lived connections where halve the congestion window is lost at the end of slow start. Also, additional examination of the improvements in performance the Preemptive Retransmission algorithm provides must be performed over a live network; this will allow the results of the simulations, as presented in this paper, to be confirmed under a real scenario.

References

[1] Allman, M., Paxson, V., Stevens, W. (1999), "TCP Congestion Control", RFC2581, April 1999

[2] Floyd, S., Henderson, T., Gurtov, A., (2004), "The NewReno Modification to TCP's Fast Recovery Algorithm", RFC3782, April 2004

[3] Ghita, B.V., Furnell, S.M., Lines B.M. and Ifeachor, E.C. (2003) , "Endpoint study of Internet paths and web pages transfers", *Campus-Wide Information Systems*, Vol. 20, Issue 3, pp90-97, 2003

[4] Hengartner, U., Bollinger, J., and Gross, T. (2000), "TCP Vegas Revisited" *Proceedings of IEEE Infocom 2000*, pp1546-1555, 2000

[5] Mathis, M., Mahdavi, J., Floyd, S., Romanow, A., (2006), "TCP Selective Acknowledgments", RFC 2018, October 2006.

[6] Noureddine, W., and Tobagi, F. (2002), "The Transmission Control Protocol", http://citeseer.ist.psu.edu/noureddine02transmission.html

[7] ns2, (2007), "The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/

[8] Osterman, S., (2007), "Tcptrace Home Page", http://www.tcptrace.org/

[9] PackMime, (2007), "Web Traffic Generation in NS-2 with PackMime-HTTP", http://www-dirt.cs.unc.edu/packmime/

[10] Paxson, V., (1999), "End-to-End Internet Packet Dynamics" *IEEE Transactions of Networking*, Vol. 7, Issue 3, pp272-292, 1999

[11] Tcpdump, (2007), "Tcpdump Public Repository", http://www.tcpdump.org/

[12] wget, (2007), "wget GNU home page", http://www.gnu.org/software/wget/

[13] Zhang, Y., Duffield, N., Paxson, V., and Shenker, S., (2001), "On the Constancy of Internet Path Properties", *Proceedings of ACM SIGCOMM Internet Measurement Workshop*, November 2001