Super Fast Retransmit: A Proposal to Improve the Performance of Short-Lived TCP Connections

A.J.Tarr and B.V.Ghita

Network Research Group, University of Plymouth, Plymouth, United Kingdom
e-mail: research@alextarr.co.uk; info@network-research-group.org

Abstract

This paper presents a method of improving the performance of TCP (Transfer Command Protocol) based short-lived connections, such as web transfers. This paper presents the methods used to examine the characteristics experienced while communicating with a number of Internet sites, and finds that constancy in the RTT (Round Trip Time) and throughput can be observed for periods extending to a number of days. 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, it was found that use of the Super Fast Retransmit algorithm provided enhanced performance showing the average duration of a connection was at least 8% lower compared to TCP Reno.

Keywords

IP Network Performance, Short-Lived TCP Connections

1. Introduction

While network connections have experienced exponential increases in performance, Internet usage has also grown to become the major source of traffic over IP networks. Web transfers, *i.e.* HTTP connections, can be characterised by the small amount of data they transfer, with the majority of connections just a few kilobytes in size; these transfers therefore typically last for less than a second and are termed "short-lived connections". Although TCP and its component algorithms perform well, the increased demand to detect and utilise larger path capacities over ever shrinking time periods leaves considerable room for improvement within TCP.

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 from this investigation will then be used as the basis to propose an algorithm to enhance TCP which will improve the performance of connections, especially those of a short-lived nature. The proposal that results from this paper will therefore be an important part of improving the performance of the Internet, especially when used for web-browsing activities, with the analysis adding to the understanding of the dynamics of the Internet.

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 short-lived connections, with section 4 detailing the analysis of the characteristics that occurred during the captured traces. In section 5 the Super Fast Retransmit 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 ideas for future work.

2. Related Work

The notion of path constancy was introduced in (Zhang, 2001), where the research focused on how well past values can be used to predict future path characteristics. The research 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 and constancy when examining the collected traces. The 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; the current investigation re-examines the levels of constancy seen in 2005.

Modern TCP implementations are based upon the algorithms presented in (Stevens, 1997). One of the algorithms, Slow Start, controls the initial transmission in the period before the first loss occurs; the algorithm exponentially increases the amount of in-flight data until the network capacity is detected. 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.

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 in 2005?" 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.

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* (Wget, 2005) tool at one minute intervals, with the packet streams captured using *tcpdump* (Tcpdump, 2005), 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.

The sites used were referenced by IP Address rather than DNS host name; this allowed the specification of a single end-point and avoid the effects of load-sharing, which might otherwise bias the results. 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 *ping*. The times returned by *ping* matched closely with the RTT measured from the HTTP requests, with the majority of connections exhibiting a ratio between TCP and *ping* RTTs of between 1.0 and 1.1.

To ensure the results reflected a wide range of situations, over 250,000 traces were collected between April and June 2005. With the traces collected from two locations, both running Linux and both locations were on the campus network at the University of Plymouth, which is connected to the Internet via JANET. The first location was a workstation in a student laboratory, while the second was a monitoring station on the network backbone.

4. Analysis of Collected Traces

Based upon the collected data, the *tcptrace* (Osterman, 2005) 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

To asses the constancy of a path, the behaviour of two key characteristics: RTT and throughput, were examined; these are critical in affecting the overall performance which an end-user will experience. The cumulative distribution of the RTT and throughput for four of the sites monitored is presented in Figure 1.

As can be observed in Figure 1 (left), 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.

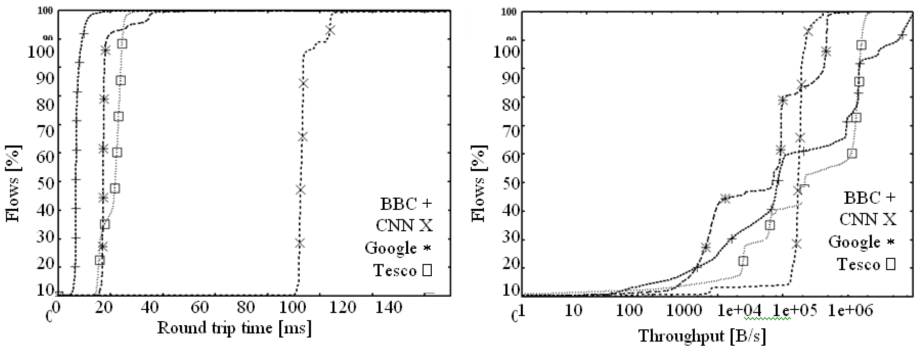

Figure 1: Cumulative distribution of four sites taken from data sets ω_2 to ω_4 showing: left - RTT; right - throughput.

The throughput, as shown in Figure 1 (right), 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 focussed on the Yahoo UK site which exhibited 11 main throughput levels causing a layered appearance over time, see Figure 2 (left).

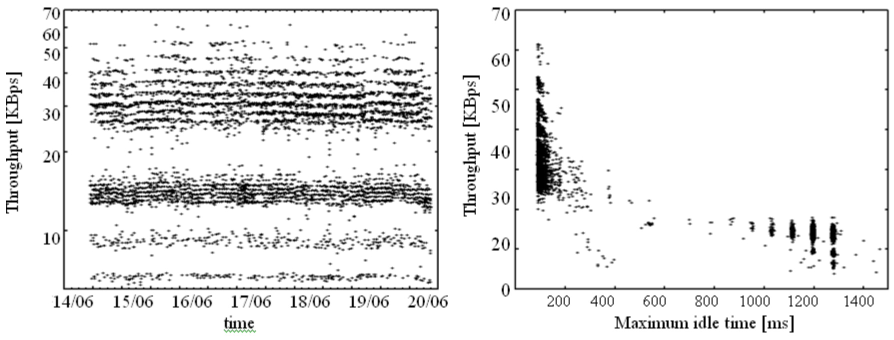

Figure 2: throughput for Yahoo UK shown: left - over time; right - against the maximum idle-time for the connection.

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

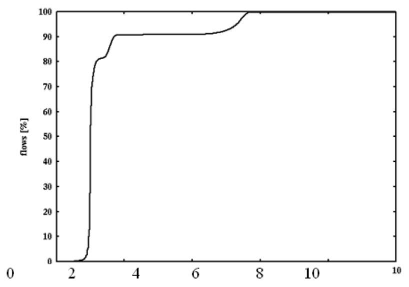

Figure 3: ratio between three-way handshake RTT and subsequent RTT for set ω_2.

Figure 3 shows 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.

5. Super Fast Retransmit Algorithm

The proposed algorithm improves the performance of short-lived connections by detecting a loss more efficiently than either Fast Retransmit or the standard TCP timeout mechanism. The proposed algorithm uses the relationship between the three-way handshake RTT and subsequent RTTs as a basis to estimate the period in which a packet should be acknowledged; if an acknowledgement has not been received, the first unacknowledged segment is retransmitted.

The Super Fast Retransmit algorithm functions as follows:

Step 1:	Establish the three-way handshake RTT to determine the timeout period.
Step 2:	When sending a segment a timer set to expire after the timeout period is started.
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.

It is obviously unrealistic to expect all packets to be acknowledged within a single RTT, therefore the proposed algorithm includes a parameter to determine the algorithms timeout period, known as the "multiplication factor". The largest ratio between the three-way handshake RTT and the maximum RTT was found to be three, therefore this value is used. 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 Super Fast Retransmit 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.

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 (Ghita, 2003), the algorithm uses a cut-out limit of 80kBytes. Following the successfully acknowledgement of the 80kBytes of data, the algorithm is disabled and can generate no further retransmissions.

6. Validation Environment

To asses the improvement in performance offered by the proposed algorithm, a number of simulations were performed using the *ns-2* network simulation tool (ns-2, 2005). Within *ns-2,* the *PackMime* (PackMime, 2005) 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 Super Fast Retransmit algorithm was implemented by modifying the implementation of TCP Reno (named Full-TCP), included with *ns-2.* 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 Super Fast Retransmit 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; while when Super Fast Retransmit was used, the average connection duration dropped to 268ms. 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 Super Fast Retransmit 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 4 (left), 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 Super Fast Retransmit, 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 network with a delay of 60ms can be explained as the timeout used by Super Fast Retransmit 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 Super Fast Retransmit 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 4 (right), 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 Super Fast Retransmit 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 Super Fast Retransmit was 23%, and 38% and 8% faster for utilisations of 25%, 50% and 75% respectively.

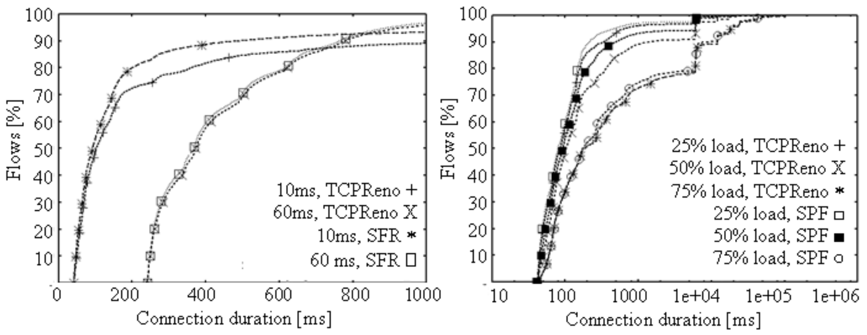

Figure 4: The effect upon the performance improvement of Super Fast Retransmit under varying: left - network delays; right - traffic levels.

7.2 Algorithm Parameters

The proposed algorithm contains two configuration parameters, the multiplication factor and the cut-out limit; the setting of these critically determines how well the algorithm functions. 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 Super Fast Retransmit.

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 80kBytes represents the optimum value based upon the average web page size.

7.3 Summary

In 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 here indicate the algorithm reduces the average duration of a connection by more than 8%; and, importantly, the performance is never lower than as for TCP Reno.

8. Conclusions and Future Work

This paper used the analysis of a large collection of trace data to establish the characteristics of short-lived TCP connections. This identified that current TCP implementations are unable to recover from a lost packet when less than three packets follow the packet which is lost, 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; critically to this research, the relationship between the three-way handshake and subsequent RTT values has been identified.

Based upon these findings, an addition to the current TCP algorithms, named Super Fast Retransmit, which causes the first unacknowledged segment to be retransmitted following a period equal to three-times the original three-way handshake RTT. To validate the proposed algorithm, a series of simulations were performed under the *ns-2* 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 Super Fast Retransmit 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.

9. References

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 ns-2, (2005) "The Network Simulator - ns-2", http://www.isi.edu/nsnam/ns/ (Accessed 17 July 2005)

Osterman, S., (2005) "Tcptrace Home Page", http://www.tcptrace.org/ (Accessed 21 January 2005)

PackMime, (2005) "Web Traffic Generation in NS-2 with PackMime-HTTP", http://www-dirt.cs.unc.edu/packmime/ (Accessed 27 July 2005)

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

Stevens, W. (1997) "TCP Slow Start, Congestion Avoidance, Fast Retransmit and Fast Recovery Algorithm", RFC2001

Tcpdump, (2005) "Tcpdump Public Repository", http://www.tcpdump.org/ (Accessed 21 January 2005)

Wget, (2005) "GNU Home Page", http://www.gnu.org/software/wget/ (Accessed 21 January 2005)

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