A Preliminary Analysis of Loss Synchronisation between Concurrent TCP Flows

Philip Jay, Qiang Fu[*], Grenville Armitage
Centre for Advanced Internet Architectures
Swinburne University of Technology
Melbourne, Australia
{pjay, qfu, garmitage}@swin.edu.au

Abstract— **This paper introduces a preliminary study of synchronisation between packet loss events across concurrent TCP flows. Loss synchronisation is explored over four routing paths – one inside Australia and three international paths (Australia to Asia, Europe and North America respectively). We confirm that loss synchronisation does exist but the level varies from path to path. We outline further work that is required to improve our understanding of loss synchronisation.**

I. Introduction

Understanding Internet traffic could give significant insight into networking mechanism design. Most of the current Internet measurements are focused on the traffic characteristics within a single flow, such as revealing the relationship between packet losses, Round-Trip Times (RTTs) and congestion window sizes. As a result, they are not able to explicitly reveal how the traffic characteristics within a given flow are related to the ones within other competing (concurrent) flows.

This paper is an effort to reveal the correlation level of traffic characteristics between concurrent TCP flows. In particular, we focus on the level of loss synchronisation between concurrent TCP flows. Perfect loss synchronisation (global synchronisation) assumes that in the presence of congestion all the competing flows experience packet losses at the same time (thus, packet losses are 'synchronised' between competing flows). The synchronised loss model has been used in related work in [1] and [2] and observed by some early work [3]. Unpublished sources from SLAC [4] and other researchers also show the signs of loss synchronisation.

The recent development on high-performance TCP variants has resulted in TCP variants such as Scalable TCP [5], HSTCP [6], HTCP [7], BIC [1], CUBIC [8], etc. These variants increase their congestion window more aggressively in a round-trip time (RTT), release a smaller portion of bandwidth in response to loss events than standard TCP and thus improve overall bandwidth utilisation in high bandwidth-delay-product (BDP) networks. It is conjectured that in the presence of congestion the increased aggressiveness of high-performance TCP variants would cause large synchronised packet losses. The work in [2] shows the long convergence time between a newly established high-performance TCP flow and an already established flow to achieve the fair bandwidth share. Ideally the two flows are expected to achieve the fair bandwidth share as soon as possible.

The long convergence time observed in [2] is obviously a negative effect of loss synchronisation and needs to be tackled. On the other hand, loss synchronisation could be utilised. The work in [9] proposed a "blind" method to improve the performance of high-performance TCP with random losses, based on the idea that serious congestion tends to create loss synchronisation while packet losses tend to be random between concurrent TCP flows in the presence of transit congestion or random packet losses (such as wireless losses). The double sides of loss synchronisation (negative and positive effects) motivate us to explore its nature and its potential impact on networking mechanism design.

In this paper, we report some of the early findings from this loss synchronisation study: a glance at loss synchronisation between concurrent TCP flows. Four paths were studied, namely, Melbourne to Adelaide (Australia), Beijing (China), Miami (USA) and Berlin (Germany). We found that loss synchronisation between concurrent TCP flows does exist but the level varies from path to path. More detailed analysis is being carried out and will be available for future report.

This paper is organised as follows. First, we describe data collection procedure in Section II. Then in Section III, we present definition and analysis method used in this paper. The measurement analysis is presented in Section IV. Section V concludes this paper and presents a discussion of future work.

II. Data Collection

Data collected for this study was generated in week-long trials comprising of tests at regular intervals. The data was collected at the bulk-sending side of the TCP flow. It is easier to detect loss events at the sending end and thus the local host collected the data. A number of TCP flows were initiated concurrently in each test.

A. Remote Hosts

During the study, four remote hosts at educational institutions were used to receive the data generated by the concurrent TCP flows. Given that these hosts are at educational institutions, the network paths that the packets traverse include research networks that may have different characteristics to commercial networks. It is assumed that this does not adversely affect the study. The network paths to these hosts are shown below:

[*] Corresponding author

ISBN/ISSN 0977586103

- Adelaide, Australia:
  12ms round trip time (RTT), 11 hops.
  01 136.186.229.3
  02 136.186.254.213
  03 203.21.130.33 (vic-gw-atm2-0-5.vrn.edu.au)
  04 202.158.200.1 (ge-1-0-3.bb1.a.mel.aarnet.net.au)
  05 202.158.194.17 (so-2-0-0.bb1.a.adl.aarnet.net.au)
  06 202.158.199.170
  (gigabitethernet0.er1.unisa.cpe.aarnet.net.au)
  07 202.158.199.122 (unisa-er1.gw.aarnet.net.au)
  08 130.220.200.48 (gi5-1.core-b.ce.unisa.edu.au)
  09 130.220.201.52 (po15.core-b.ml.unisa.edu.au)
  10 130.220.166.1 (dist-e1-16.ml.unisa.edu.au)
  11 130.220.166.65 (docker.levels.unisa.edu.au)
- Beijing, China: 170ms RTT, 22 hops.
  01 136.186.229.3
  02 136.186.254.213
  03 202.0.98.161
  (ATM8-0-0-1.edge1.vic.grangenet.net)
  04 202.0.98.41 (Vlan52.edge1.nsw.grangenet.net)
  05 202.0.98.98
  06 202.158.194.118 (ge-1-0-7.bb1.b.syd.aarnet.net.au)
  07 202.158.194.33 (so-2-0-0.bb1.a.mel.aarnet.net.au)
  08 202.158.194.17 (so-2-0-0.bb1.a.adl.aarnet.net.au)
  09 202.158.194.5 (so-0-1-0.bb1.a.per.aarnet.net.au)
  10 202.158.194.138 (so-0-0-2.bb1.a.sin.aarnet.net.au)
  11 202.179.249.25 (sg-ge-02-1g.bb-v4.noc.tein2.net)
  12 202.179.241.13
  (hk-so-02-622m.bb-v4.noc.tein2.net)
  13 202.179.241.9 (bj-so-01-622m.bb-v4.noc.tein2.net)
  14 202.179.241.26 (cn.pr-v4.noc.tein2.net)
  15 202.112.53.17
  16 202.112.53.181
  17 202.112.53.178
  18 202.112.61.254
  19 219.243.208.2
  20 219.243.208.6
  21 219.243.215.253
  22 219.243.215.196
- Miami, USA: 230ms RTT, 18 hops.
  01 136.186.229.3
  02 136.186.254.213
  03 202.0.98.161
  (ATM8-0-0-1.edge1.vic.grangenet.net)
  04 202.0.98.41 (Vlan52.edge1.nsw.grangenet.net)
  05 202.0.98.98
  06 202.158.194.82 (pos2-0.bb1.a.suv.aarnet.net.au)
  07 202.158.194.86 (pos1-0.bb1.b.hnl.aarnet.net.au)
  08 202.158.194.90 (pos2-0.bb1.b.sea.aarnet.net.au)
  09 207.231.242.8
  (abilene-1-lo-std-707.sttlwa.pacificwave.net)
  10 198.32.8.50 (dnvrng-sttlng.abilene.ucaid.edu)
  11 198.32.8.14 (kscyng-dnvrng.abilene.ucaid.edu)
  12 198.32.8.80 (iplsng-kscyng.abilene.ucaid.edu)
  13 198.32.8.78 (atlang-iplsng.abilene.ucaid.edu)
  14 198.32.252.237 (abilene-i2-flr-10g.ampath.net)
  15 198.32.252.250 (fiu-wr2-v103.ampath.net)
  16 131.94.192.25 (fastgate.cs.fiu.edu)
  17 131.94.134.129 (sagwa.cs.fiu.edu)
  18 131.94.130.45 (hpc.cs.fiu.edu)
- Berlin, Germany: 340ms RTT, 23 hops.
  01 136.186.229.3
  02 136.186.254.213
  03 202.0.98.161
  (ATM8-0-0-1.edge1.vic.grangenet.net)
  04 202.0.98.41 (Vlan52.edge1.nsw.grangenet.net)
  05 202.0.98.98
  06 202.158.194.82 (pos2-0.bb1.a.suv.aarnet.net.au)
  07 202.158.194.86 (pos1-0.bb1.b.hnl.aarnet.net.au)
  08 202.158.194.90 (pos2-0.bb1.b.sea.aarnet.net.au)
  09 207.231.242.8
  (abilene-1-lo-std-707.sttlwa.pacificwave.net)
  10 198.32.8.50 (dnvrng-sttlng.abilene.ucaid.edu)
  11 198.32.8.14 (kscyng-dnvrng.abilene.ucaid.edu)
  12 198.32.8.80 (iplsng-kscyng.abilene.ucaid.edu)
  13 198.32.8.76 (chinng-iplsng.abilene.ucaid.edu)
  14 198.32.8.83 (nycmng-chinng.abilene.ucaid.edu)
  15 198.32.11.51
  16 62.40.112.133 (so-7-0-0.rt1.ams.nl.geant2.net)
  17 62.40.112.57 (so-6-2-0.rt1.fra.de.geant2.net)
  18 62.40.124.34 (dfn-gw.rt1.fra.de.geant2.net)
  19 188.1.18.53 (cr-berlin1-po1-0.x-win.dfn.de)
  20 188.1.20.85 (ar-tuberlin2-po6-0.x-win.dfn.de)
  21 195.37.76.33 (funnel.fokus.fraunhofer.de)
  22 193.175.134.17
  23 193.175.134.50

The local host is located at the Centre for Advanced Internet Architectures (CAIA), Swinburne University of Technology (SUT), Melbourne, Australia. Connection to the SUT network is provided over 100 megabit per second (Mb/s) Ethernet. SUT has a 155Mb/s connection to its upstream provider. The connection to GrangeNet [10] is allocated 55Mb/s of this upstream, whereas the connection to AARNet [11] is allocated 100Mb/s.

B. *Software*

To generate concurrent TCP flows the OpenSSH [12] based utility *scp* [13] was utilised. *scp* was used to transfer a 10 megabyte file from CAIA to the remote hosts. Between 2 and 64 instances of *scp* were run concurrently. *scp* encrypts the data payload during transfers. AES encryption is the *scp* default, and is more processor intensive than Blowfish encryption [14][15]. To ensure the lightest possible load on the CPUs of the hosts Blowfish encryption was used. *tcpdump* [16] was used to capture the TCP/IP headers of the packets generated by these data transfers. Each test was coordinated by a UNIX shell script. Scheduling the tests to occur at regular intervals was performed using the UNIX *cron* [17] utility.

The TCP stack of the FreeBSD [18] host at CAIA used in the study was modified to insert the TCP stack's current RTT estimate into the header of outgoing TCP packets. We used an unallocated TCP option number, 30, [19] to insert the RTT estimate. FreeBSD stores the RTT estimate in the *t_srtt* kernel variable. Outputting the RTT estimate in the TCP headers allows the RTT estimate to be captured by *tcpdump*. An assumption of this study is that inserting a custom option into the TCP header will not materially affect the behaviour of the TCP flow. (Routers do not process TCP options, but the receiving TCP stack will expend a few extra clock cycles deciding if the option is understood and needs processing.)

ISBN/ISSN 0977586103

All hosts used a TCP-SACK [20] compatible variant of TCP with timestamps [21] enabled and the default maximum window size (64kb). TCP-SACK allows selective acknowledgement of lost packets to reduce spurious retransmissions and avoid unnecessary timeouts. This is useful to the study since it allows accurate detection of lost packets. Only lost packets will be retransmitted.

C. Trials

Each trial lasted a week and consisted of a number of tests with constant intervals. During each test, a number of concurrent TCP flows were established and each of the flows transferred a 10 megabyte file from CAIA to the remote hosts. During the week-long trial, the intervals between the tests were either one or two hours. Table I summaries the trials in the study.

TABLE I. TRIALS CONDUCTED IN THE STUDY

	Remote host location							
	Adelaide, Australia		Beijing, China		Miami, USA		Berlin, Germany	
No. of concurrent TCP flows	8	2, 4, 16, 32	16	2, 4, 8, 32	32	64	32	64
Interval between tests (hours)	1	2	1	2	1	2	1	2

The UNIX shell script that launched each test staggered the *scp* transfers at one second intervals if there were to be 32 or less TCP flows. In the case of a 64 flow test the shell script would launch two *scp* transfers at each one second interval. Due to this staggering the data is only considered valid between the start of the last *scp* transfer (last flow) to begin, and the end of the first *scp* transfer (first flow) to finish. This can be easily detected by finding the last synchronise (SYN) flagged packet, and the first finish (FIN) flagged packet of the concurrent TCP flows. The 64 flow test to the Miami remote host produced no valid data. This is because the CPU in the local host is not powerful enough to perform the encryption required, when connecting with more than approximately 50 flows. The consequence of this is that no more than approximately 50 *scp* transfers start until the CPU load goes below 100%. Hence the last SYN packet is after the first FIN packet.

III. DEFINITION AND ANALYSIS METHOD

A. Custom Software

A custom Java [22] program was written to analyse the *tcpdump* trace files. It utilises the 3rd party Java library *jpcap* [23] to interface to *libpcap* [16] which can read the *pcap* format files created by *tcpdump*.

B. Definition

When analysing the data we needed to define a loss event. The loss event start time is the time at which the first lost packet of any number of subsequent lost packets is sent. The event finish time is the time at which the first duplicate acknowledgement (ACK) arrives back at the sender. The ACK will subsequently be duplicated, and therefore signal a loss event. Since all hosts are using TCP-SACK we consider all the retransmitted packets to be lost.

If a lost packet sending time is not during the current loss event time, then it must signal a new loss event. This definition of loss events is verified by multiplying the length (in time) of a loss event by 2, and by 3. In both cases the number of loss events detected in trace files does not change by more than 10%. This indicates that the time-based definition of loss events is accurate for our purposes.

A correlated loss event is signalled if there is any overlap (in time) of loss events between two or more flows and all the involved flows are considered correlated with each other. For instance, if a loss event in flow 1 is correlated with a loss event in flow 2 and there are no loss events from the other concurrent flows which are correlated with these two correlated loss events in flow 1 and 2, then we will regard that only flow 1 and flow 2 are correlated with each other, that is, this is a two-flow based correlation. However, if a loss event in flow 3 is correlated with the loss event in flow 2 but not the loss event in flow 1 (that is, by the definition above flow 1 is not correlated with flow 3), we will still regard that flow 1, 2 and 3 are correlated with each other (three-flow based correlation) because of the mutual correlation with flow 2.

C. Method

Synchronisation levels between the concurrent TCP flows are calculated by inspecting the loss events in the order that they occurred. If there are correlated loss events in other flows then a counter is incremented. This counter indicates how many flows are synchronised with the loss event.

When there are no more loss events to inspect, an array containing a counter for each loss event is available for analysis. Loss event synchronisation is calculated as the number of loss events that had *x* number of flows or more divided by the total number of loss events. Synchronisation is expressed as a percentage.

IV. MEASUREMENT ANALYSIS

In this section, we focus on the measurement analysis of loss synchronisation between concurrent TCP flows. Fig. 1 ~ 4 shows the results of the data traces to the remote hosts in Adelaide, Beijing, Miami and Berlin, respectively. In the figures, the y-axis represents the correlation percentage between the concurrent TCP flows while the x-axis represents the least number of flows that are correlated with each other. For instance, when x=2, y=92.97% in Fig. 1 it means that 92.97% of the total loss events involves *at least* 2 correlated flows within the total 32 concurrent flows. For another example, when x=2, y=44.17% in Fig. 1 it means that 44.17% of the total loss events involves *at least* 2 correlated flows within the total 4 concurrent flows, that is, it includes the cases of 2-flow, 3-flow and 4-flow based correlation.

Overall, the figures show that loss synchronisation between the concurrent TCP flows appears to exist. For the same remote host, different numbers of concurrent TCP flows have the similar loss synchronisation pattern (Fig. 1, 2 and 4). For instance, in Fig. 1 and 2 the pattern for 32 concurrent flows is similar to the pattern for 16 concurrent flows. The level of synchronisation varies depending on the network path. TCP flows to Beijing exhibited the highest level of synchronisation (Fig. 2), whereas TCP flows to Berlin had the lowest (Fig. 4). The Miami host and the Berlin host shared network paths until somewhere inside the USA, and TCP flows to these hosts exhibited a similar loss synchronisation pattern (Fig. 3 and 4). Levels of loss synchronisation were stronger in the Miami trials (Fig. 3), compared to the Berlin trials. This indicates that the traffic between USA and Germany makes packet losses less synchronised. Trials involving the Adelaide host exhibited stronger synchronisation than trials involving either the Miami or Berlin host, but not as strong as the Beijing host.

The RTT to the Adelaide host is 12ms while the RTT to the Miami host is 230ms and to the Berlin host is 340ms, and correspondingly the number of hops is 11, 18 and 23. It seems that the level of loss synchronisation is correlated with RTT as well as number of hops: longer RTT and more hops all contribute to less loss synchronisation. More hops could mean more random background traffic and thus make packet losses between concurrent TCP flows appear to be more random. Longer RTT would mean larger bandwidth-delay product (BDP) and thus require a bigger TCP congestion window to fully utilise available bandwidth or cause packet losses. As mentioned earlier, the maximum TCP window size is set to the default value (64kb). This would indicate that in the presence of congestion longer RTT would have more flows than shorter RTT which have reached their maximum window size. As a result, shorter RTT would have more flows that increase their window size during the congestion and have more chance to face synchronised packet losses between these flows. Further investigation is needed to determine the correlation between RTT, number of hops and loss synchronisation.

Interestingly the loss synchronisation pattern for the Beijing host is distinct from the others (exponential decrease for the others). The front part of the pattern is not similar to the others having a logarithmic decrease curve. However its rear part curve is similar to the others (exponential decrease). This indicates stronger loss synchronisation than the other hosts. The Beijing host has longer RTT and more hops than the Adelaide host and more hops than the Miami host. However, it has the strongest loss synchronisation. Again, this is an area that will require further investigation. At this stage we consider that a couple of factors might contribute to this observation.

One factor could be the distinct local traffic in China. Asian countries such as China, Korea and Japan usually have a substantial quantity of local content which is only available in local language. The content is usually retrieved by the local users rather than the users overseas and the local users generally request local content. On the other hand, the rapid deployment of broadband Internet access in households and the

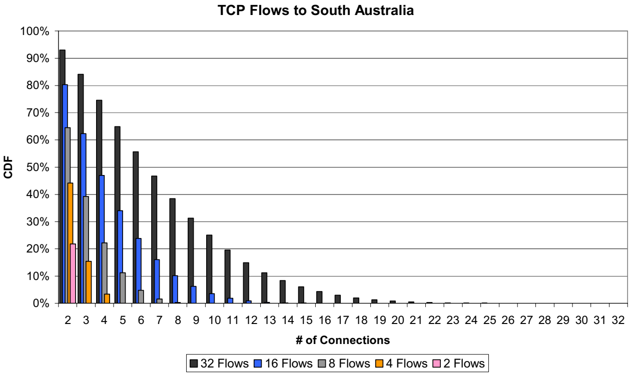

Fig 1. Percentage of synchronisation between concurrent TCP flows to Adelaide, Australia

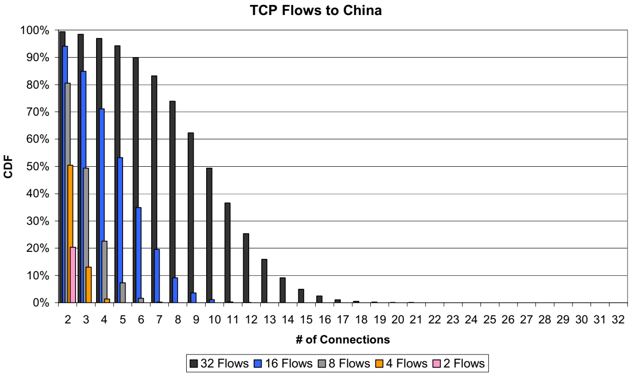

Fig 2. Percentage of synchronisation between concurrent TCP flows to Beijing, China

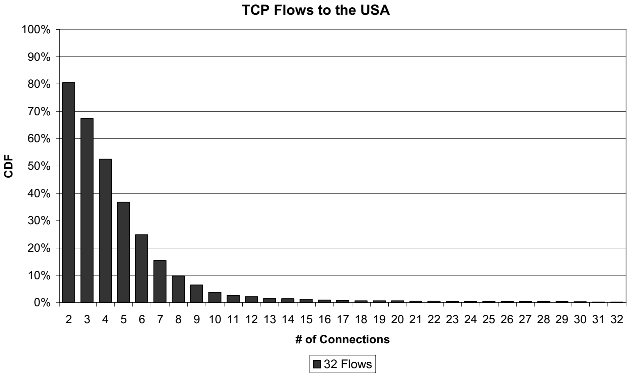

Fig 3. Percentage of synchronisation between concurrent TCP flows to Miami, USA

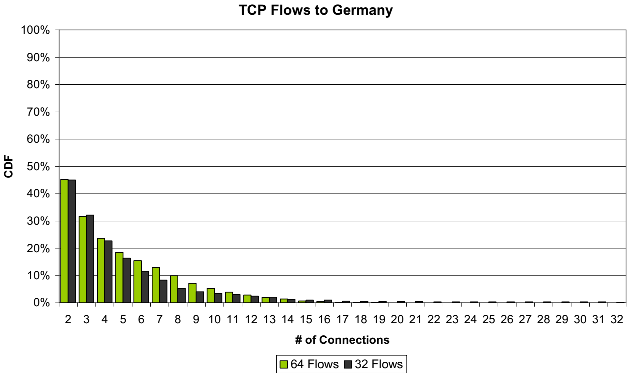

Fig 4. Percentage of synchronisation between concurrent TCP flows to Berlin, Germany

ISBN/ISSN 0977586103

rapid increase of Internet users in China could also contribute the uniqueness of its local traffic.

Another factor could be the use of queue management such as Random Early Detection (RED). RED causes packet losses between concurrent flows not to be synchronised due to its stochastic nature of dropping packets. The work in [1] presents a brief analysis of loss synchronisation in a simple simulation environment. It shows that in the simulation with RED turned on the loss synchronisation pattern is exponential decrease while with RED turned off the pattern is logarithmic decrease. (Note that our study is very different from [1] because of different analysis method and environment.) Since the turnoff of RED in [1] has a partially similar pattern to the Beijing host and the use of RED in [1] has a similar pattern to the other three remote hosts. RED could be a reason to cause the difference. However, it has been suggested that RED is usually turned off in practice.

V. Conlusions and Future work

In this paper, we have tentatively studied the correlation of loss events (loss synchronisation) between concurrent TCP flows over three international routing paths from Australia to Asia, Europe and North America and a domestic routing path from Melbourne to Adelaide. The measurement results we have presented in this paper suggest that loss synchronisation between concurrent TCP flows does exist but the level varies from path to path. The local internet traffic might also have unique impact on loss synchronisation. For instance, the path to China exhibits a loss synchronisation pattern distinct from the others. It could be due to the frequent local traffic caused by accessing local internet content and the behaviour of local Internet users. The existence of loss synchronisation indicates that the design of networking mechanisms such as TCP should take this issue into account. Some designs would prefer eliminate the effects of loss synchronisation while some might be able to utilise it. The presented results are preliminary and further analysis is certainly required for better interpretation and understanding.

We plan to study the relationship between the number of lost packets and the level of synchronisation. Through the analysis we would like to find out how the average number of lost packets is related to the synchronised loss events. It is also interesting to increase the maximum window size in TCP to study loss synchronization. The maximum window size in the study was set to its default value (64kb). For a long haul link such as the one to Germany, the window size could not be sufficient to fully utilise available bandwidth or to cause packet losses. On the other hand, it is worthwhile to do the study with recently developed high-performance TCP variants such as HSTCP to compare loss synchronisation between standard TCP and high-performance TCP variants. It is conjectured that high-performance TCP variants would cause large and synchronised packet losses. The future work would also be able to give some insights into this issue.

Acknowledgement

We are grateful to the following people for providing computer accounts at the remote sites: Dr. Steven Gordon (ITR, University of South Australia), Ke Xu (Tsinghua University, China), (Chi Zhang, Florida International University) and Frank Burkhardt (FOKUS, Germany).

References

[1] L. Xu, K. Harfoush and I. Rhee, "Binary Increase Congestion Control for Fast Long-Distance Networks", *INFOCOM* 2004, Hong Kong, China, Mar. 2004.

[2] Y. Li, D. Leith and R. Shorten, "Experimental Evaluation of TCP Protocols for High-Speed Networks", *IEEE/ACM Trans. on Networking* (to appear).

[3] S. Shenker, L. Zhang, and D. Clark, "Some observations on the dynamics of a congestion control algorithm", *SIGCOMM* 1990, Philadelphia, USA, Sep. 1990.

[4] "FAST TCP for Multi-Gbps WAN: Experiments and Applications", http://www.slac.stanford.edu/grp/scs/net/talk/fast-i2-apr03.ppt, viewed 9th August, 2006.

[5] T. Kelly, "Scalable TCP: Improving performance in highspeed wide area networks", *Computer Communication Review* 32(2), April 2003.

[6] S. Floyd, "HighSpeed TCP for Large Congestion Windows", *IETF RFC 3649*, Dec. 2003.

[7] R. Shorten and D. Leith, "H-TCP: TCP for high-speed and long-distance networks", *PFLDnet* 2004, Argonne, USA, Feb. 2004.

[8] I. Rhee and L. Xu, "CUBIC: A New TCP-Friendly High-Speed TCP Variant", *PFLDnet* 2005, Lyon, France, Feb. 2005.

[9] Q.Fu, G.Armitage "A Blind Method towards Performance Improvement of High Performance TCP with Random Losses", *WWIC* 2006, Berne, Switzerland, May 2006

[10] "GrangeNet", http://www.grangenet.net/, viewed 3rd August, 2006.

[11] "AARNet home", http://www.aarnet.edu.au/, viewed 3rd August, 2006.

[12] "OpenSSH", http://www.openssh.com/, viewed 3rd August, 2006.

[13] "UNIX man pages: scp (1)", http://unixhelp.ed.ac.uk/CGI/man-cgi?scp+1, viewed 3rd August, 2006.

[14] H. Xie, L. Zhou, and L. Bhuyan, "Architectural Analysis of Cryptographic Applications for Network Processors", *8th International Symposium on High-Performance Computer Architecture*, Cambridge, USA, Feb. 2002.

[15] "Setting up SSH", http://pierre.mit.edu/compfac/ssh.html, viewed 7th August, 2006.

[16] "TCPDUMP public repository", http://www.tcpdump.org/, viewed 3rd August, 2006.

[17] "Newbie: intro to cron", http://www.unixgeeks.org/security/newbie/, viewed 3rd August, 2006.

[18] "The FreeBSD project", http://www.freebsd.org/, viewed 3rd August, 2006.

[19] "TCP option numbers", Internet Assigned Numbers Authority, California, USA, Jul. 2006.

[20] M. Mathis, J. Mahdavi, S. Floyd and A. Romanow, "TCP Selective Acknowledgment Options", *IETF RFC 2018*, Oct. 1996.

[21] V. Jacobson, R. Braden and D. Bormanm, "TCP Extensions for High Performance", *IETF RFC 1323*, May 1992.

[22] "Java technology", http://java.sun.com/, viewed 3rd August, 2006.

[23] "Jpcap – Java package for packet capture", http://netresearch.ics.uci.edu/kfujii/jpcap/doc/, viewed 3rd August, 2006.

ISBN/ISSN 0977586103