Studies of TCP's Retransmission Timeout Mechanism

Nabil Seddigh
Tropic Networks
nseddigh@tropicnetworks.com

Mike Devetsikiotis
North Carolina State University
mdevets@eos.ncsu.edu

Abstract

This paper focuses on the initial value of TCP's Retransmission Timeout (RTO) timer. We make a three-fold contribution to the work in this area. Firstly, to motivate this work, we conduct a study of Internet traffic to determine loss rates for those packets that rely on timeouts with the initial RTO value to recover. Secondly, we experiment with TCP stacks in nine different operating systems to understand the initial RTO values used. Finally, we conduct simulations with HTTP traffic models to study the impact of different initial RTO values on the transfer delay of web objects. The experiments show that average transfer delay can be impacted by as much 70% on bottleneck links with loss rates above 6%. Further, TCP flows with timer granularity below 100ms or Round Trip Times (RTT) below 250ms can suffer similar performance degradation. The results suggest that TCP's initial RTO value can be reduced from its current recommended value of 3s to between 500ms and 1s.

1. Introduction

In the absence of ECN (Explicit Congestion Notification), the TCP (Transmission Control Protocol) sender's primary mechanism to infer packet loss is by observing three duplicate acknowledgements (*dupack*) in the stream of acknowledged packets. As a safeguard for cases where the three-*dupack* scheme does not trigger, TCP employs a retransmission timeout (RTO) mechanism to infer packet loss. Inference of packet loss via RTO results in performance degradation since the sender cannot send any packets but has to wait for the RTO to expire.

There are two complementary approaches to reducing the performance impact of RTO in TCP: (i) modify the TCP flow control scheme so that the *dupack* scheme is triggered more often and there is less reliance on RTO to recover from packet loss – a recent large-scale Internet Traffic study by Balakrishnan et al [9] showed that approximately 50% of all packet losses require a timeout to recover (ii) improve the TCP RTO mechanism so that less time is spent waiting for the timer to expire. Proposals such as "Increased Initial Window Size" [11] are based on the former approach. In addition, motivated by the latter approach mentioned above, there has been renewed research into the TCP RTO mechanisms. The current scheme for TCP's RTO estimation was first presented in [1] and is codified in [15]. The scheme combines a smoothed estimate of the RTT with a mean deviation of the RTT to obtain the RTO estimate.

Recent work investigating the TCP RTO mechanism and proposing improvements can be found in publications by Allman and Paxson [14], Ludwig and Slower [16], Allman and Griner [17], Aron and Druschel [10]. The studies focus on the effectiveness of the RTO mechanism after a certain number of packets have been transmitted by the TCP flow.

In this work, we investigate whether the choice of initial RTO value impacts the performance of HTTP flows in routers with RED buffer management. The authors are unaware of any other publications on this theme. To motivate the work, we perform a passive analysis of Internet traffic traces to determine the prevalence of timeout with duration of initial RTO. We then study nine different operating systems to determine the initial RTO values utilized. The actual research on the performance impact is carried out using simulation studies.

This paper is organized as follows: Section 2 discusses related work. Section 3 determines situations in which the initial RTO value affects performance. Section 4 contains results of the loss rate study of Internet Traffic. Section 5 presents results of the initial RTO experiments with the TCP implementations. Sections 6 and 7 present the simulation results. Finally, conclusions and recommendations can be found in section 8.

2. Related Work

Selecting an appropriate initial RTO value is important. A large initial RTO value will be problematic for flows with small RTT as it may result in performance degradation in the case of *early packet drop* –dropping of any of the first few packets of a flow. Too small an initial RTO will be problematic for flows with large RTT as it may cause needless retransmissions due to bad timeouts. RFC 2525 [13] documents examples where low initial RTO values caused needless performance degradation via early retransmissions. RFC 1122 [2] states that the initial RTO should be set to 3 seconds.

0-7803-7097-1/01/$10.00 ©2001 IEEE

In Comer and Lin's [4] 1994 study of 5 operating systems, none of the surveyed TCP implementations conformed to the 3-second initial RTO. The initial RTO values ranged from 200-milliseconds to 1.5 seconds. In a subsequent study published in 1997, Dawson et al [8] actively probed the behaviour of 6 commercial TCP implementations and found that 5 of the implementations had an initial RTO of 1 second while one implementation had an initial RTO of 300-milliseconds.

Paxson undertook a passive analysis study [7] of 20,000 Internet traces. The traces include data transfers from 8 different TCP implementations including some operating systems previously reported on as part of the studies in [4] and [8]. His study confirms that Solaris 2.3 and 2.4 maintain an initial RTO value of 300 msec and suggests that this low RTO value coupled with difficulties in adapting the RTO value to changing RTTs, results in "broken retransmission behaviour". He further finds that the Linux version 2.x TCP implementation uses high-resolution RTT measurements with more frequent RTT sampling than once per RTT.

3. When is Initial RTO Value Important?

This section performs packet-by-packet analysis to understand situations in which the TCP RTO timer expires with its value set to the initial RTO. The experiment was performed on a company Intranet. The simple network setup is shown in Figure 1. In the experiment, the client H1 retrieves a 10Kbyte file from the HTTP Web Server H2. The total transmission delay between client and server is on the order of milliseconds.

Figure 1: Network Setup

We repeated the 10Kbyte data transfer numerous times each time forcing a particular packet from the transfer to be dropped. Specific packet drops were engineered by adding filters to router R based on the source or destination IP address and packet number in the flow. The key observations can be summarized as follows:

1. Dropping of the SYN Packet caused timeout with initial RTO value – 3 seconds for the Linux client
2. Dropping of the SYNACK Packet caused a timeout with the initial RTO value – 6 seconds for the VxWorks server. The SYN was also retransmitted after an RTO of 3 seconds.
3. Dropping of the HTTP GET packet caused a timeout with initial RTO value – 3 seconds.
4. Dropping of a single packet just before or during the FIN handshake caused timeout with value based on the RTO estimation algorithm.

From the above, we note that dropping the SYN, SYNACK or the GET packet results in a timeout with initial RTO value. This caused a 1 second data transfer to be completed in around 4 seconds. In the case of the dropped SYNACK, the total data transfer time was almost 6.7 seconds. This clearly underscores the importance of the initial RTO value. In the next section, we investigate the likelihood of losing the SYN and SYNACK packets.

The GET timeout duration is set to initial RTO because the Linux client, despite obtaining its first RTT sample on receipt of the SYNACK, does not use this sample to update the smoothed RTT, the smoothed RTT variance or the RTO value. In reality, the first RTT sample that affects the RTO value does not occur until the server ACKs the GET packet – this was verified by examining the Linux TCP source code.

4. Internet Traffic – Passive Analysis

The objective of this section is to study the loss rates of TCP's SYN and SYNACK packets in real networks today. In a passive analysis of trace data from the mid-nineties [5], Stevens found that almost 10% of SYN packets were retransmitted.

4.1 Network Topology and Data Set

The study utilized vBNS Internet Traffic Traces collected by the National Laboratory for Applied Network Research (NLANR) [18].

Table 1: Packet Trace Details

Dump Sites	# Traces	Trace Duration (s)	Trace Length (N packets)
MaeWest-AIX	5	$10 \le T \le 186$	$167568 \le T \le 1564405$
Argone NatLab	6	$103 \le T \le 181$	$24637 \le T \le 107417$
APAN-(APN) STARTAP	6	$170 \le T \le 191$	$95601 \le T \le 386898$
Florida U (FLA)	5	$160 \le T \le 177$	$132914 \le T \le 258688$
Front Rng Gig	2	$89 \le T \le 184$	$375236 \le T \le 539934$
Mich (MRT) U	6	$183 \le T \le 193$	$630449 \le T \le 1784119$
NCAR – Boulder	6	$35 \le T \le 101$	$25553 \le T \le 306946$
North Carolina U	6	$53 \le T \le 169$	$65114 \le T \le 339699$
Old Dominion U	6	$122 \le T \le 178$	$120886 \le T \le 313395$
Ohio State U	5	$97 \le T \le 189$	$86162 \le T \le 889180$
SanDiego State	6	$172 \le T \le 192$	$290846 \le T \le 767300$
Tel Aviv Univ	6	$85 \le T \le 177$	$105580 \le T \le 554169$
Texas Univs	6	$68 \le T \le 81$	$108486 \le T \le 164870$

The vBNS is a nation-wide network in US that supports high-performance, high-bandwidth research applications. The network connects 101 institutions including 94 US universities, 4 supercomputer centres and 3 collaborative institutions. A total of 71 traces were utilized in the study.

Table 1 summarizes the 71 individual traces representing traffic from 13 different sites across the US. In total, the traces represented 23 million packets, 1.6 million flows and 9600 seconds worth of data. The total number of packets in each trace ranges from tens of thousands to millions while the trace duration ranges from 9 seconds to 190 seconds. The traces were collected during Jan-June 2000.

4.2 Analysis Methodology

Determining loss rate without access to the intermediate routers is a difficult task. Instead, we rely on observing retransmissions as an approximation to the loss rate. Details on the analysis method can be found in [19]. In this section, we simply highlight some of the key issues for which heuristics had to be developed:

- Recognize that packets in a flow may take multiple paths to the destination and thus, leave holes in the TCP sequence number trail. Be concerned with packet reordering that may occur.
- Approximate loss rate not by retransmissions but by "retransmission bursts" which may occur due to "go-back-N" schemes.
- Distinguish between retransmitted SYN(ACK)s due to router drops and retransmitted SYN(ACK)s due to bad destination IP addresses.
- Detect broken TCP implementations and exclude them from the analysis.

We are confident that the analysis accurately reflects data packet and SYN, SYNACK loss for the traces studied.

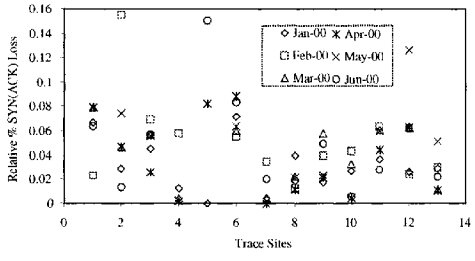

Figure 2: SYN and SYNACK Relative Loss Rates

4.3 SYN & SYNACK Loss

In this section, we present the results of the Internet loss rate study for SYN and SYNACK packets. The analysis summary for all 67 files indicates that 5% of packets were retransmitted. The retransmitted packets consisted of retransmission bursts that we use to approximate loss. The estimated overall loss rate is 1.3%. The relatively low loss rate is expected since the VBNS is a high-performance network. We note that other studies [6] have shown that on expensive WAN links, packet loss of 10-15% is not uncommon.

For the purposes of motivating the evaluation of TCP's initial RTO value, it is not the overall loss rate that is of interest but rather, the relative loss rates of SYN, SYNACK and HTTP GET packets. The study found that 4% of all packet loss were SYN packets while 2% of dropped packets were SYNACKs. Loss rate for HTTP GET packets was left for future study. Figure 2 depicts the average relative loss rates of SYN and SYNACK packets as a percentage of total loss for each trace. The x-axis corresponds to the 13 sites listed in Table 1 in descending order. Majority of the traces experienced SYN(ACK) relative loss rates under 10%. A few traces experienced relative SYN(ACK) loss rates from 10 to 20%.

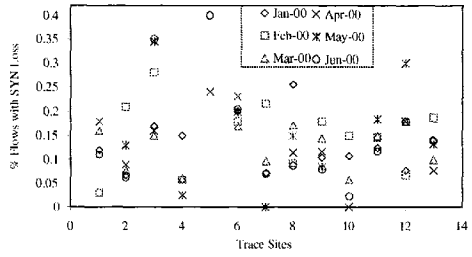

Figure 3: Lossy Flows with SYN Loss

Figure 3 shows the trace summary for lossy flows that experience SYN and SYNACK loss. The results indicate that on average, 16% of all flows with loss experienced a SYN or SYNACK packet drop. This result points to a lockout phenomenon in the routers – if a flow experiences loss, it is quite likely to happen on flow startup because of full router queues. We note that these statistics would be higher if the studies also considered HTTP GET loss patterns. Further discussion on the implication of these results can be found in section 8.

5. TCP Implementation Study

In this section we test TCP implementations on nine different operating systems to determine their initial RTO value. Unlike previous studies that used modified active probes or coordinated collection of distributed traces from around the Internet, we install filters on intermediate routers to drop TCP SYN packets. The end-host then retransmits the packet after the initial timeout. The measurements were repeated 10 times to gain confidence in the results. The average initial RTO values are presented in Table 2.

The results show that three operating systems - Win98, WinNT and Linux - have initial RTO values that are virtually identical to what is suggested as a SHOULD in the TCP standards. Four of the operating systems - SunOS 4.1.4, IRIX 6.2, VxWorks 5.4 and NetBSD - have initial RTO values that are almost *twice* the recommended value in the standards. The

HP-UX initial RTO value fluctuated greatly through the various tests with a deviation almost twice the average value of 4.2 seconds. The Solaris initial RTO was found on average to be around 3.3 seconds. Previous work [4] found this value to be around 300 milliseconds. Since the previous studies reviewed earlier in this paper, TCP implementations appear to have modified their initial RTO value. However, there still remain a number of implementations with initial RTO values that are clearly too high and will result in performance degradation of short flows because of unnecessary idle time in the case of early packet drop. In subsequent sections, via simulation and live network experimentation, we try to understand the impact of choosing such initial RTO values.

Table 2: OS Average Initial RTO Values

Operating System	Measured Init RTO (Avg)
Win98	2.97s
WinNT 4.0 – Service Pack 3	2.98s
Linux 2.2.12-20	3.0s
Sun Solaris (SunOS 5.7)	3.37s
SunOS 4.1.4 1	5.7s
Irix 6.2	5.82s
VxWorks 5.4 – SENS	5.86s
NetBSD 1.0A	5.81s
HP-UX B.10.20	4.22s

In addition to the above, tests on the exponentially backed off RTO values show that 3 operating systems (Win98, WinNT and Linux) have values that match the expected results. Three operating systems (NetBSD, SunOS 4.1.4 & SunOS 5.7[Solaris]) exhibit RTO values that double with each backoff – even though the initial RTO does not match the standard. Three operating systems (HP-UX, VxWorks and Irix) appear to use a larger backoff multiplier than the standard value of two.

6. Simulation Setup

In this section, we utilize a simulation study to investigate the effect of the initial RTO value on the performance of short TCP New Reno flows in a RED-based router network. We argue that short TCP flows continue to dominate Internet traffic and thus, deserve special attention to improve their performance.

6.1 Network Topology

The ns-2 simulator is used for the simulation study. The network topology is depicted in Figure 4. The topology consists of 10 web servers and 10 web clients. The web clients access the web server through a network with a bottleneck link of C_2 Mbps and transmission delay of T_2 ms. The bottleneck link emulates a WAN link. The leaf routers (L_1 and L_2) emulate routers inside an Intranet or MAN. The routers bounding the bottleneck links use RED for buffer management while the leaf routers and other nodes use drop tail. We ensure that there is no packet loss on the access links between the web servers and the leaf routers or between the leaf routers and the bottleneck link routers. Relevant simulation parameters can be found in Table 3. The parameters were used for all experiments unless explicitly stated otherwise.

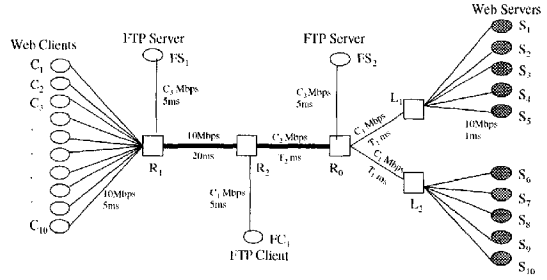

Figure 4: Network Topology

6.2 Traffic Model - Background FTP Flows

To create congestion for both the HTTP GET requests and the HTTP data, bi-directional background traffic is generated in the form of infinitely long-lived FTP flows. The HTTP data shares the same congested link (R_0-R_2) as FTP data flowing from FS_2 to FC_1 – the FTP client. The HTTP GET requests share the same congested link (R_1-R_2) as FTP data from FS_1 to FC_1. Depending on the particular testcase, the number of background FTP flows is varied. The start times of the FTP flows are uniformly distributed between 0 and 1s.

Table 3: Simulation Parameters

Parameter	Value
TCP Sender & Receiver	New Reno; Delayed ACK
RED Parameters (min_{th}, max_{th}, max_p, w_q)	40pkts, 80pkts, 0.1, 0.002
DropTail Q Sizes (Bottleneck, Access Links)	100 pkts, 200 pkts
Packet Size	1000bytes
TCP Timer Granularity	500ms
TCP Receiver Max Window	500 pkts

6.3 Traffic Model - HTTP Flows

This section describes the ns-2 web traffic model utilized in the first experiment of the section. The traffic model uses a TCP where the three-way opening handshake is omitted and data can only be transferred uni-directionally. To more closely emulate the real world, the ns-2 implementation utilizes two separate flow-pairs. A single flow-pair is used to send a GET from client to server for which it receives an ACK. As soon as the ACK arrives at the client, the server begins the data transfer. For each new page, the client randomly selects a server from the pool of servers. All objects in a page are sent from the same server to the client. Each object starts a new TCP flow. Sessions are started 50s after the long-lived FTP flows are started. The inter-session start times are uniformly distributed between 0 and 2s. The same random seed was utilized for all runs of the experiment. Table 4 presents more details.

Table 4: HTTP Traffic Model for 1st Experiment

Parameter	Value
# Sessions	10
# Pages per session	400
Inter-page generation time	Exponential Distrib (avg = 2s)
# Objects per page	Exponential Distrib (avg = 2)
# packets per object	Constant size of 8
Inter-object generation time	Exponential Distrib (avg = 0.01s)
Simulation Duration	1300 s
Total # objects	approximately 10,000

7. Experiments And Results

In this section we present results of simulations to assess the performance impact of the initial RTO value on transfer delay of short flows. We consider the performance impact in the presence of three factors: different loss rates, TCP timer granularity and RTT size. The initial RTO values considered reflect the values currently used in various TCP stacks deployed on the Internet today – section 5

7.1 Performance Metrics

To compare results between various simulations we utilize a set of metrics related to the transfer time for an object. We define Transfer Time T_i for object i to be: $T_i = T_{LAST\_ACK} - T_{GET}$; where T_{GET} is the time when the client sends the HTTP GET packet to the server and T_{LAST_ACK} is the time when the server receives the ACK for the last packet of data for object i. We include the transfer delay of the GET request/response because it may play a key role in affecting the end-user's gauge of transfer delay. Three performance indicators are computed to assist in evaluating the effect of various initial RTO values: (i) the average transfer time T_{avg} for N objects (ii) the standard deviation of the transfer times (iii) the fairness index (FI) as presented in [3]: The FI lies between 0-1 and is computed as follows: $FI = \frac{\left(\sum_{i=1}^{N} T_i\right)^2}{N \sum_{i=1}^{N} T_i^2}$. A higher value for FI indicates greater fairness or lesser variability amongst object transfer times.

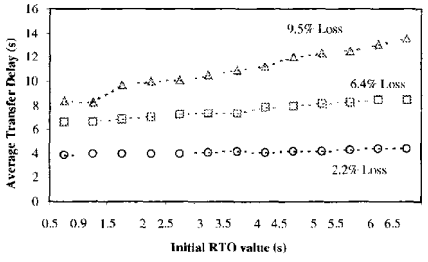

Figure 5: Initial RTO Effect for Different Loss – Avg Delay

7.2 Impact of Loss Rate

In this section we investigate the effect of the initial RTO value in impacting the transfer delays of web traffic under low, medium and high loss rates. We select loss rates of around 2.2%, 6.4% and 9.5% to reflect low, medium and high respectively. Background FTP flows are utilized to increase congestion on the link to sufficient levels that can realize the desired loss rates – 8, 25 and 40 FTP flows were utilized for the increasing loss rates respectively. For this experiment, C_1, C_2, C_3, T_1, and T_2 were 10Mbps, 10Mbps, 100Mbps, 5ms, and 10ms respectively. The TCP timer granularity is 500ms.

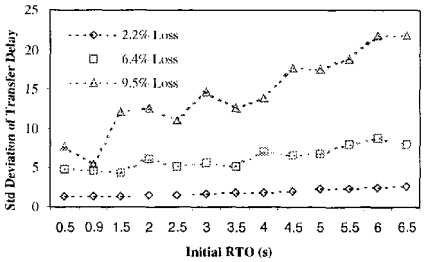

Figure 6: Initial RTO Effect for Different Loss – Std Dev

Figures 5, 6 and 7 illustrate the effect of initial RTO values on the average, standard deviation and fairness of transfer delay respectively – for three different loss rates. The key observations from the above Figure include:

- ✓ For a loss rate of 2.2%, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 3.92s to 4.32s (a 10% increase); the standard deviation of the transfer delay increases from 1.45 to 2.68 – (an 84% increase); the fairness index decreases from 0.88 to 0.73 – (a decrease of 20% decrease).
- ✓ For a loss rate of 6.4%, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 6.63s to 8.43s – (a 27% increase); the standard deviation of transfer delay increases from 4.9 to 8.01 – (a 63% increase); the fairness index decreases from 0.65 to 0.52 – (a 20% decrease).

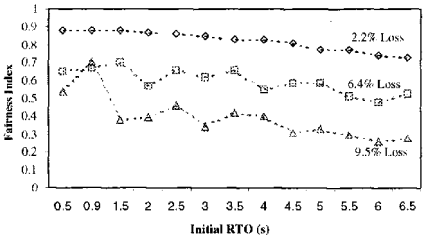

Figure 7: Initial RTO Effect for Different Loss Rates – Fairness

- ✓ For a loss rate of 9.5%%, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 8.4s to 13.6s – (a 61% increase); the standard deviation of transfer delay increases from 7.8 to 21 – (a

169% increase); the fairness index decreases from 0.54 to 0.28 – (a 48% increase).

The results show that for higher loss rates, the average transfer time and fairness are affected by the choice of initial RTO value.

7.4 TCP Timer Granularity

In this section we investigate the effect of the initial RTO value on performance for different granularities of the TCP timer.

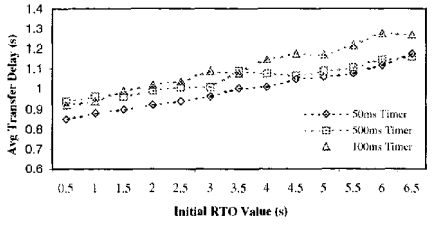

Figure 8: Impact of Initial RTO for Different Timer Granularity– Avg

The traffic model used for this experiment differs slightly from the previous section. We mimic 10 HTTP sessions continually retrieving objects with each object utilizing a distinct TCP flow. All objects have a fixed size of 8 packets. New objects for a session are retrieved only after the current object for that session has been completely retrieved. To reduce synchronization effects, we introduce a small time between the end of one transaction and start of the next – the duration of this time is uniformly distributed between 0 and 1second. A total of 10,100 objects are generated. The sources and destination nodes remain the same as in the previous experiment. For this experiment, C_1, C_2, C_3, T_1, and T_2 were 10Mbps, 10Mbps, 100Mbps, 5ms and 10ms respectively. The RTT for the HTTP flows is around 75ms, reflecting common RTT values used on the Internet [12]. The loss rate on the bottleneck link is approximately 2.45% with no loss on other links. For the test with 500ms, the loss rate was low at 1.74%.

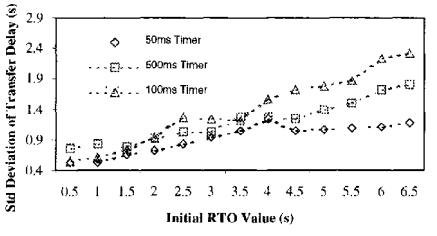

Figure 9: Initial RTO Impact for Different Timer Granularity–St Dev

The test is repeated for TCP timer granularities of 50ms, 100ms and 500ms. Most TCP implementations today utilize a 500ms timer. However, a number of implementations such as Linux utilize more fine-grained timers. Also, 25 FTP flows were used for the test with 100ms timer while 20 FTP flows were used for the test with 50ms and 500ms timers. Figures 8, 9 and 10 present the results of the experiments with the different timer granularities. The key observations are:

- ✓ For a granularity of 50ms, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 0.85 to 1.21 – (an increase of 39%); the standard deviation of the transfer delay increases from 0.54 to 2.11 (a 290% increase); the fairness index decreases from 0.71 to 0.24 (a 66% decrease).
- ✓ For a granularity of 100ms, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 0.92 to 1.27 – (an increase of 38%); the standard deviation of the transfer delay increases from 0.55 to 2.33 (a 320% increase); the fairness index decreases from 0.73 to 0.23 (a 68% decrease).
- ✓ For a granularity of 500ms, as the initial RTO increases from 0.5s to 6.5s, the average transfer delay increases from 0.94 to 1.16 – (an increase of 23%); the standard deviation increases from 0.76 to 1.81 (a 138% increase); the fairness index decreases from 0.61 to 0.29 (a 52% decrease).

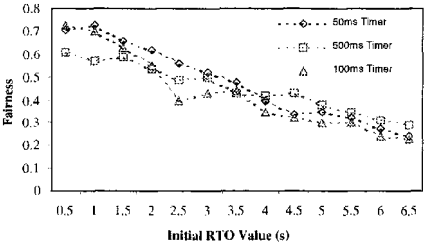

Figure 10: Initial RTO Impact for different Timer granularities- FI

7.5 Impact of RTT Size

In this section, we investigate the effect of the initial RTO for TCP flows with different RTTs.

We first conduct a test where the HTTP flows have a common internet RTT of 80ms [12] and TCP timer granularity of 50ms. The traffic model is the same one used for the test with TCP timer granularity. There are 40 background FTP flows. A 500ms timer granularity is used to reflect common TCP implementations today. For this experiment, C_1, C_2, C_3 and T_2 were 10Mbps, 10Mbps, 100Mbps and 10ms respectively.

The results are presented in Figures 11 and 12. We see a marked increase in average delay and its standard deviation as the initial RTO value is increased from 0.5 to 6.5. The average delay increases by 71% while the standard deviation increases by 460%. Further we note that the fairness index decreases from 0.63 to 0.14 – a marked decrease of 77%.

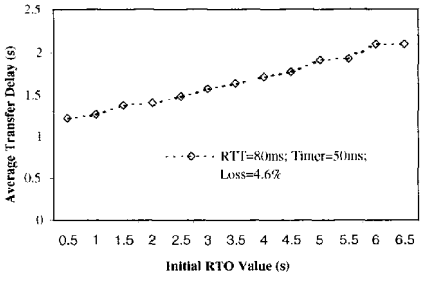

Figure 11: Average Delay – Small RTT; Small Timer

A second RTT experiment was also performed to gauge the effect when the initial RTO value is smaller than the RTT for a TCP flow. The experimental details can be found in [19]. Here, the results are summarized:

- There is at most a 10% difference in average transfer delays for any two initial RTO values for the large RTT values of 1s, 3s and 5s.
- For the case of small initial RTO and large RTT, the increase in unnecessary timeouts translates into a 10% increase in average delay and has no impact on fairness or standard deviation.

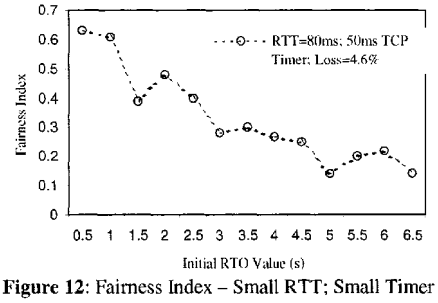

Figure 12: Fairness Index – Small RTT; Small Timer

8. CONCLUSION

This paper evaluates the effect of initial RTO value on the transfer delay of short flows. Analysis showed that dropping of SYN, SYNACK or HTTP GET packets causes timeout with initial RTO value. Examination of Internet traces revealed sufficient loss rates for SYN and SYNACK packets to motivate a study on the effect of the initial RTO value.

Simulation studies indicated that for loss rates above 5%, TCP timer granularity of 100ms or less, and for RTTs below 500ms, initial RTO values of 3-6s can result in average transfer delays increasing by as much as 70%. Fairness in transfer delays suffered a large proportional decrease in the above cases. Further, we have found that the performance gain by using reduced initial RTO values more than compensates the borderline case of unnecessary retransmissions of SYN and SYNACK due to long RTTs or a receiver who is not present – the performance degradation in terms of reduced average transfer delay is less than 10% when the RTT is 5s.

We recommend that the initial RTO value for TCP be reduced from 3seconds to either 1seconds or 0.5seconds – the latter is particularly recommended if a TCP utilizes a timer granularity of 100ms or less.

9. ACKNOWLEDGEMENTS

We acknowledge contributions from Biswajit Nandy, Pete Pieda, Fulu Li, Jamal Hadi Salim, Manuel Oliveira and Rupinder Makkar on TCP, ns-2, NLANR trace data and Linux-related investigation and discussion. We are also grateful to NLANR for making the Internet traces available.

10. REFERENCES

[1] Jacobson V., *"Congestion Avoidance and Control"*, In Proceedings of SIGCOMM '88, Stanford, CA, August 1988, ACM

[2] Braden R, *"Requirements for Internet Hosts -- Communication Layers"*, Internet RFC 1122, October 1989

[3] Jain R, "The Art of Computer Systems Performance Analysis", John Wiley & Sons, New York, 1991

[4] Comer D and Lin J, "Probing TCP Implementations", Usenix 1994.

[5] Stevens W. R, *"TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP and the UNIX Domain Protocols"*. Addison-Wesley, 1994

[6] Paxson V, *"End-to-End Internet Packet Dynamics"*, Proceedings of SIGCOMM 97, France, September 1997

[7] Paxson V, *"Automated Packet Trace Analysis of TCP Implementations"*, Proceedings of SIGCOMM 97, France, September 1997

[8] Dawson S, Jahanian F, and Mitton T, *"Experiments on Six Commercial TCP Implementations Using a Software Fault Injection Tool"*, In Proceedings of Software Practice and Experience, Vol 1(1); 1997

[9] Balakrishan H, Padmanabhan V, Seshan S, Stemm M and Katz M, *"TCP Behaviour of a Busy Internet Server: Analysis and Improvements"*, Proceedings of INFOCOMM 98, San Francisco, March 1998

[10] Aron M and Druschel P, *"TCP: Improving Startup Dynamics by Adaptive Timers and Congestion Control"*, Rice University Technical Report, June 1998

[11] Allman M, Hayes C and Ostermann S, *"An Evaluation of TCP's with Larger Initial Windows"*, Computer Communication Review, July 1998

[12] Fei A, Pei G, Liu R and Lixia Zhang, *"Measurements on Delay and Hop-Count of the Internet"*, Globecom 98, November 98, Australia

[13] Paxson V, Allman M, Dawson S, Fenner W, Griner J, Heavens I, Lahey K, Semke J, Volz B, *"Known TCP Implementation Problems"*, Internet RFC 2525, March 1999

[14] Allman M and Paxson V, "On Estimating End-to-End Network Path Properties", Proceedings of SIGCOMM 99, August 1999, Cambridge, MA

[15] Paxson, V and Allman M, *"Computing TCP's Retransmission Timer"*, Internet RFC 2988, November 2000

[16] Ludwig R and Slower K, *"The Eifel Retransmission Timer"* Computer Communications Review, Volume 30, #3, July 2000

[17] Allman M, Griner J and Richard A, *"TCP Behaviour in Networks with Dynamic Propagation Delay"*, Proceedings of Globecom 2000, ??? 2000

[18] ftp://moat.nlanr.net/pub/MOAT/Traces, National Laboratory for Applied Network Research

[19] Seddigh N, *"Performance Analysis of TCP's Retransmission Timeout Mechanism"*, Technical Report, Carleton University, December 2000