A Passive State-Machine Based Approach for Reliable Estimation of TCP Losses

Sushant Rewaskar Jasleen Kaur F. Donelson Smith
University of North Carolina at Chapel Hill
{rewaskar, jasleen, smithfd}@cs.unc.edu

Introduction: While it is well-known that TCP performance degrades significantly on experiencing packet losses, not much is known about the way in which TCP losses occur within the current Internet and how effectively does TCP's loss detection-recovery mechanisms deal with them. As a first crucial step in filling this gap, we address the issue of: *how to reliably obtain the loss process that real-world TCP connections experience.* We developed a passive analysis methodology for reliably inferring the loss processes from real-world TCP connection traces. We instantiate our methodology in analysis tools that emulate detailed sender-side state machines for several prominent TCP stacks (Windows, Linux, BSD (MacOS), and Solaris) and augment these with extra logic to correctly track TCP sender state as well as actual segment losses. Using these tools we analyze traces of more than 25 million TCP connections, collected from 4 different networks: *Abilene, UNC, MAWI working group (Japan), and Univ of Leipzig*.

Basic Approach: Our basic approach for passive inference of TCP losses is to: (i) emulate the state-machine of a TCP sender using the ACK stream to track the triggering of loss detection/recovery mechanisms, and (ii) augment the state machine with extra state and logic about the transmission order and timing of all previously-transmitted packets in order to classify retransmissions as necessary or not. Using this basic approach, we can classify segment retransmissions as triggered by: (i) RTOs, (ii) Triple Duplicate Acks (TDAs), (iii) Partial Acks (PAs), (iv) SACK, and (v) implicit (e.g. in response to ACK after a RTO retransmission). For any retransmission, our analysis also attempts to find out if the retransmission was unnecessary (i.e., the original transmission had successfully reached the receiver).

Practical Challenges: Three concerns complicate implementation of the above.

Diverse and Non-documented TCP Stacks: TCP implementations written by different operating system vendors may differ (sometimes significantly) in either their interpretations or their conformance to TCP specification/standards. We extract sufficient details about the implementation of loss detection/recovery in several prominent stacks by using an approach similar to the t-bit approach described in [1]. Details of the extracted characteristics can be found in [2].

Delays and Losses between a Monitor and the Sender: The data and ACK stream observed at a trace-collecting monitor may differ from the one seen at the TCP sender due to delays, losses or reordering between the sender and the monitor. We deal with this by using loss indications in the ACK stream to trigger only tentative state changes; these are confirmed only by subsequent retransmission behavior by the sender. We infer network reordering by detecting (i) reordering in the packet's IP-id field, and (ii) whether an out-of-sequence segment appears within a fraction of the connection's minimum RTT relative to the next higher sequence number segment.

Non-availability of SACK Options: A large number of traces do not capture SACK blocks present in the TCP option field. In absence of this information we classify a segment retransmission as SACK-triggered using the following heuristics: (i) the connection is in fast retransmit/recovery (FR/R), (ii) the retransmission is not explained by

Trace	# Conn.	# Packets	# OOS	% No Cause	% Network Reorder	Retransmissions						
						Total	% RTO	% TDA	% PA	% SACK	% Implicit	% Unexp
Abilene-OC48-2002	7.1M	160.1M	410K	9.6	18	296K	32.5	11.5	2.0	4.5	18.3	3.5
Liepzig-1Gbps-2003	2.4M	17.3M	51K	1.7	0.47	50K	46.3	5.9	1.9	4.9	27.7	11.1
Japan-155Mbps-2004	0.3M	3.7M	52K	0.4	2.9	50K	47.5	11.9	2.6	1.7	31.4	1.6
UNC-1Gbps-2005	14.5M	151M	698K	6.69	29	446K	34.2	6.1	2.8	1.5	13.2	6.0
Ibiblio-1Gbps-2005	0.9M	158.9M	504K	0.7	0.2	500K	33.1	14.0	4.8	0.0	44.1	3.0

Table 1. Classification of OOS

either RTO or a PA, and (iii) the sequence number of the retransmitted segment is less than the highest sequence number that was in flight when the connection entered FR/R.

Validations: We validate our analysis tools against TCP connections for which the "ground truth" about each Out of Order Segment (OOS) is known. We modify *tbit* to simulate different packet loss scenarios that would trigger sender responses by withholding ACKs, sending duplicate ACKs, and providing SACK blocks reflecting gaps in received sequence numbers. We also implemented the ability to generate desired variability in RTT by delaying the ACKs. The procedures we used have two parts (1) to verify that each TCP implementation responds in real operation as we expected, and (2) to verify that the state machine analysis tools correctly emulate each implementation's responses. For the first part we processed the tcpdump traces with tcptrace and other tools to verify the implementations' responses. For the second part, we used tcpdump traces as input to the state machine analysis programs and recorded their outputs. Comparing the results from the state machine analysis with the known implementation responses, we could validate state machines inferences about sender states.

Analysis of TCP Connections: Table 1 lists the traces used in our analysis. These traces are collected from links with transmission capacity ranging from 155 Mbps to OC-48. For our analysis, we use only those connections that transmit at least 10 segments and had at least one OOS. Table 1 also shows the classification for OOS events in the five traces. We find that

- Our tools were able to infer the cause for more than 90% of the OOS events.
- In 4 out of the 5 traces more than 50% of the retransmissions were due to timeouts. Only 10-15% of retransmissions are triggered by TDA.
- 20-45% of retransmissions are implicit and are caused by TCP's Go-Back-N style retransmissions after RTO. 24-42% of these are unnecessary.

The fraction of unnecessary retransmissions can range from 18-38%. The results are quite consistent with those reported by Allman for three of the traces but are higher for two. This may be because Allman does not track RTO or TDA triggered unnecessary retransmissions. We find that while a majority of connections (80%) send less than 2 unnecessary retransmissions, a non-negligible fraction (7%) send more than 5.

Implications of Our Analysis: Our analysis of real-world TCP connections suggests important implications for TCP performance (especially for development of analytic models of TCP throughput as a function of loss rates). Understanding prevalence of RTO and duplicate-ACK triggered retransmissions are important for modeling TCP throughput. Finally, the stateful analysis capability of the tool facilitates identifying not only the TCP mechanism causing the performance limitation (e.g. TDA triggered retransmission) but also the underlying cause for this (for e.g. network reordering vs. actual loss).

References:

1. J. Padhye and S. Floyd. On inferring tcp behavior. In *Proceedings of ACM SIGCOMM*, 2001.
2. S. Rewaskar, J. Kaur, and D. Smith. Passive inference of TCP losses. *Tech Report, Department of Computer Science, University of North Carolina at Chapel Hill*, Oct 2005.