Demystifying and Mitigating TCP Stalls at the Server Side

Jianer Zhou • Qinghua Wu • Zhenyu Li • Steve Uhlig
Peter Steenkiste • Jian Chen • Gaogang Xie

Proc. of ACM IMC '11, pp. 155-170, 2011.

Presented by Buting Ma and Ryan Marcotte

Motivation

Why should we care about diagnosing problems with TCP in the wild?

- Importance of perceived performance
    - Throughput
    - Latency
    - …
- TCP optimization has received study
    - Congestion control (FastTCP, Compound TCP, TCP Cubic, Remy CC, …)
    - Optimizing retransmissions (tail loss probe, limited transmit, adaptive reordering threshold, …)
    - …
    - Timeout retransmissions?
- Server-side mitigation more easily implemented and deployed

What are TCP Stalls?

- An event where the duration between consecutive packets received/sent by the sender is large

$$\min(\overset{\text{Free parameter (=2)}}{\tau} \cdot \underbrace{\mathrm{SRTT}}_{\text{Smoothed Round Trip Time}}, \underbrace{\mathrm{RTO}}_{\text{Retransmission Timeout}})$$

- Intuition: a TCP sender should be able to receive or send at least one packet during two RTTs

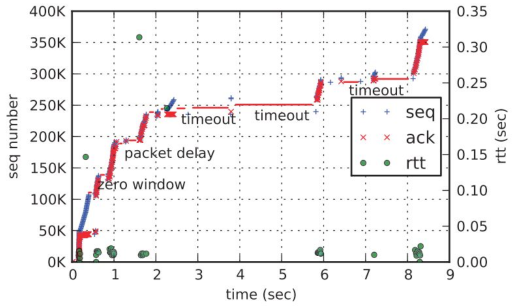
Contributions

- A tool (TAPO) that classifies causes of TCP stalls
    - Testing and analysis on packet-level traces from a popular Chinese service provider
    - Focus on timeout retransmission stalls
- A technique (S-RTO) to mitigate timeout retransmission stalls
    - Deployment of S-RTO in production network
    - Analysis of performance in this deployment

Dataset Description

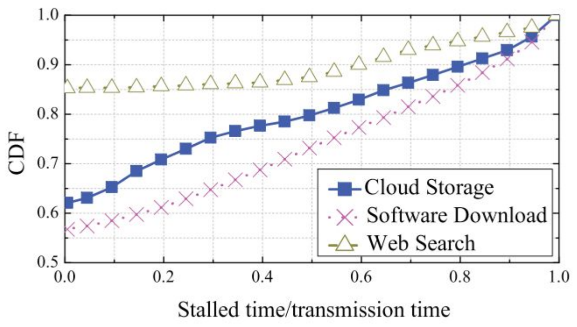
- Packet-level traces from front-end servers of Qihoo 360
  - Provider of web search, mobile assistant, cloud storage, etc.
  - More than 600 million users
- The datasets
  - Web search
  - Cloud storage download
  - Security software download
- The traces
  - Daily, 12/22/14 to 12/28/14
  - One hour each during peak evening traffic
  - 3.35 billion packets, 6.4 million flows

TCP Stalls in the Dataset



Spends more than half of the time stalled!

TCP Stalls in the Dataset (cont.)

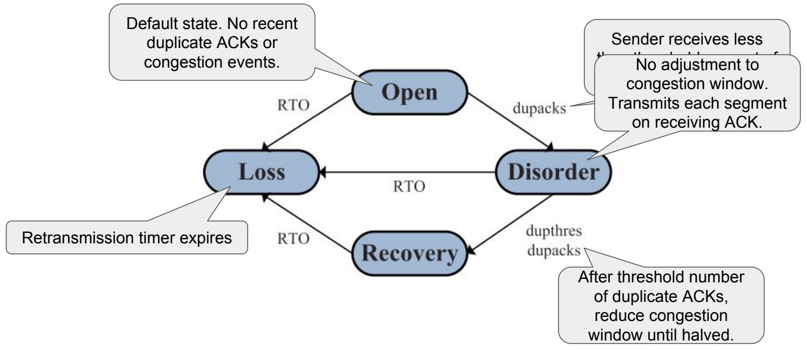


Challenges in TCP Stall Diagnosis

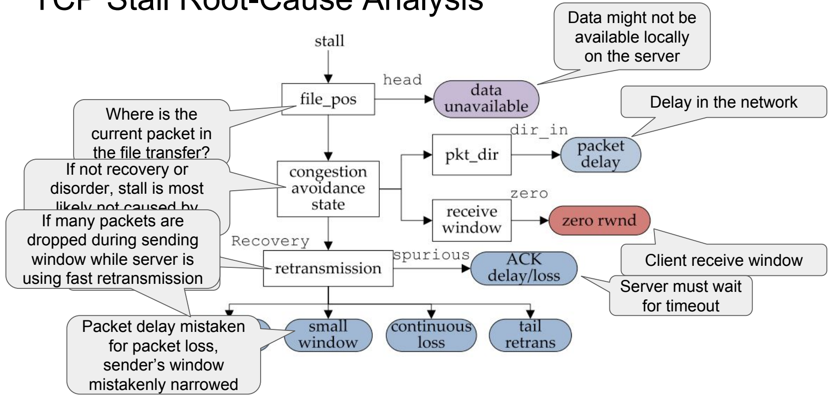
- Statistics of TCP performance parameters must be obtained through measurement
  - Some parameters may not be available from kernel
  - Dumping kernel logs of front-end servers could be problematic
- Client-side events are not visible to server-side tools

Based on this, there is a need for server-side diagnosis tools.

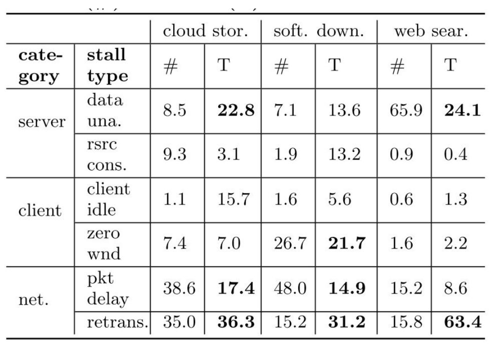
Review: TCP Congestion Avoidance


Default state. No recent duplicate ACKs or congestion events.
Sender receives less
No adjustment to congestion window. Transmits each segment on receiving ACK.
Open
RTO
dupacks
Loss
RTO
Disorder
RTO
Recovery
dupthres dupacks
Retransmission timer expires
After threshold number of duplicate ACKs, reduce congestion window until halved.

TCP Stall Root-Cause Analysis


stall
file_pos
head
data unavailable
Data might not be available locally on the server
Where is the current packet in the file transfer?
Delay in the network
dir_in
pkt_dir
packet delay
congestion avoidance state
If not recovery or disorder, stall is most likely not caused by
zero
receive window
zero rwnd
If many packets are dropped during sending window while server is using fast retransmission
Recovery
retransmission
spurious
ACK delay/loss
Client receive window
Server must wait for timeout
Packet delay mistaken for packet loss, sender's window mistakenly narrowed
small window
continuous loss
tail retrans

Overall Statistics of TCP Stalls

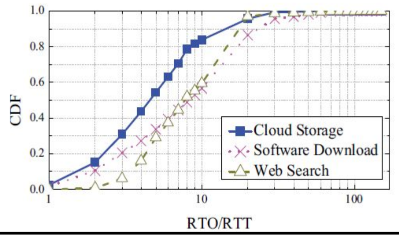


		cloud stor.		soft. down.		web sear.	
cate-gory	**stall type**	#	T	#	T	#	T
server	data una.	8.5	**22.8**	7.1	13.6	65.9	**24.1**
	rsrc cons.	9.3	3.1	1.9	13.2	0.9	0.4
client	client idle	1.1	15.7	1.6	5.6	0.6	1.3
	zero wnd	7.4	7.0	26.7	**21.7**	1.6	2.2
net.	pkt delay	38.6	**17.4**	48.0	**14.9**	15.2	8.6
	retrans.	35.0	**36.3**	15.2	**31.2**	15.8	**63.4**

Timeout Retransmission Stalls

Timeout retransmission stalls degrade TCP performance significantly because the **TCP sender cannot transmit packets for a period equal to the RTO**, which can be tens of RTTs (see Figure 1). It also forces the TCP sender to **ramp up its congestion window from 1 MSS** (max segment size) after the stall.

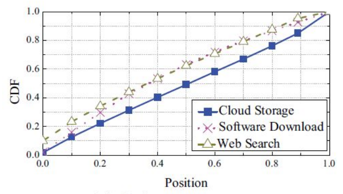
- Double retransmission stalls
  - f-double stalls & t-double stalls
- Tail retransmission stalls
- Small in-fight retransmission stall
- Continuous loss stall
- ACK delay or loss stall



Double Retransmission Stalls

A double retransmission refers to a case where **the retransmitted packet itself**, either recovered by fast retransmit or a timeout retransmission, **is dropped or delayed**, causing a new timeout retransmission.

expensive: cloud storage 45%, software download 61%, web search services 42%

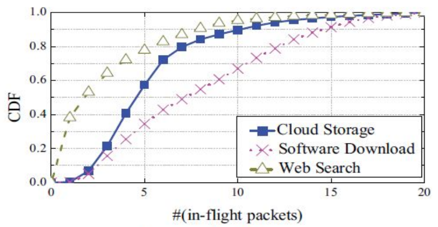


(a) Relative position

position = index of the retransmitted packet / the number of data packets in the flow

almost uniform distribution: stalls are caused by random packet drops.

Web search 10% at 0 position: Some web search flows contain only one packet.



(b) *in_flight* size

The in flight size is the number of packet that have been sent out but have not yet been acknowledged by either an ACK or SACKs

Web search tends to have a smaller in flight size because web search flows are shorter and the cwnd often never ramps up to a high value.

After a double retransmission stall, the sender have to **1. retransmit these in-flight packets, 2. set the cwnd to one.** This is a significant drop in cwnd for both the cloud storage and software download services, further degrading performance.

F-double stall and T-double stall

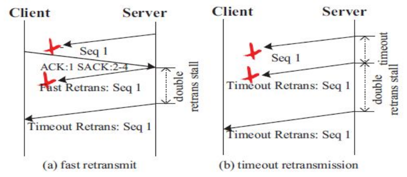


Figure 8: Two different scenarios of double retransmission stall.

F: fast, T:timeout

The most significant difference between the two types is that a t-double stall delays the data segment (and the entire flow) by two or more timeouts.

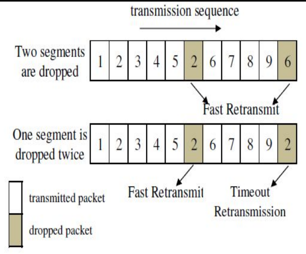


Figure 9: The transmission sequences in fast retransmit and double retransmission.

· Top scenario: two different segments are dropped (2 & 6). Both (2 & 6) can be recovered via fast retransmit.
· Bottom scenario: same segment (2) is dropped twice.The first loss is recovered by fast retransmit, but **the second loss of can only be recovered through a timeout retransmission** (a double retransmission stall) , as the TCP sender has **already marked segment 2 as retransmitted.**

Both scenarios transmit the same amount of data.
f-double stalls could be eliminated through a slightly more aggressive and efficient retransmission strategies that avoid timeouts without adding further congestion to the network.

Table 6: Percentage of each type of double retransmission stalls in terms of stalled time.

	cloud s.	software d.	web search
f-double stall	62.3%	52.7%	55.6%
t-double stall	37.7%	47.3%	44.4%

F-double stall more than 50% in all 3 types of traffic.

mitigating f-double stalls can significantly reduce the impact of double retransmissions and improve performance.

Recently proposed retransmission mechanisms like Early Retransmit , Tail Loss Probe do not help here because they either only reduce the threshold of fast retransmit or require the TCP sender to be in the Open state.

Tail retransmission stalls

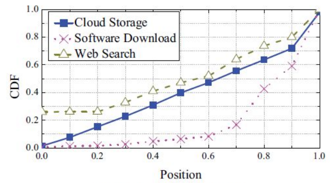
A tail retransmission stall happens as a result of a retransmission **at the tail of a flow**. At the end of a flow,the receiver **cannot generate the three dupacks needed for fast retransmit** so a timeout is needed for recovery.

Table 5: Percentage of retransmission stalls (%) in terms of volume (#) and time (T) for different causes.

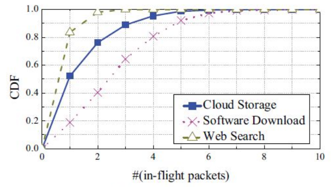
	cloud stor.		soft. down.		web search	
stall type	#	T	#	T	#	T
Double retr.	26.7	**45.4**	41.2	**60.8**	25.6	**41.9**
Tail retr.	4.8	5.0	0.4	0.4	44.4	**36.0**
Small cwnd	35.2	**27.3**	16.9	7.2	15.2	11.6
Small rwnd	0.4	0.3	10.6	3.7	0.87	0.3
Cont. loss	19.0	**10.1**	5.6	1.6	0.6	0.6
ACK delay/loss	6.3	6.5	14.9	**22.2**	2.1	1.8
Undeter.	7.4	6.1	10.3	4.4	11.1	7.8

In web search, account for 36% of all stalls.

Most of the web search flows are small, so packet loss is more likely to happen in the tail of the flow.



(a) Relative position



(b) *in_flight* size

Figure 10: Context for tail retransmission stalls.

a uniform distribution for both the cloud storage and web search services. Cloud storage: many files; web search: many flows in one session. But **for software dl, only one file.**

The flow of web search is short, most of the flows contain only one in-flight packet when the tail retransmission happens. For the other two services, the in-flight size is often no more than 3.

Table 7: Percentage of each type of tail retransmission stalls in terms of stalled time.

	cloud s.	software d.	web search
Open state	60.1%	41.3%	10.0%
Recovery state	39.9%	58.7%	90.0%

The web search service has the smallest fraction of tail stalls in Open state, indicating a higher likelihood that **at least one retransmitted packet has not been ACKed when a tail stalls happen**.

The Open state corresponds to a better network condition than the Recovery state, and thus a tail retransmissions happening in the **Open state can possibly be mitigated** via carefully retransmitting unacknowledged packets, which is the basis for **Tail Loss Probe (TLP)**.

Small in-flight retransmission stall

in-flight size is small if in-flight < 4MSS.
When a packet loss occurs while the in-flight size is small, **fast retransmit cannot be triggered due to too few duplicate ACKs**. The sender therefore has to wait until the retransmission timer expires.

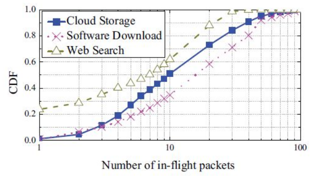


Figure 11: Distribution of in-flight size computed on each ACK.

20% of in-flight values observed in the cloud storage and software download services are below 4. In other words, once an in-flight packet is dropped, the server has to wait for the retransmission timeout in 20% of the cases.

Small rwnd or small cwnd

The in-flight size reflects the number of packets limited by either cwnd or rwnd.

Further distinguishing which variable limits the in-flight size, **rwnd or cwnd**,

Small rwnd retransmission stalls contribute very little to the total stalls for the cloud storage and web search services, thanks to large initial rwnd.

Small cwnd retransmission stalls are significant in all three services, but can **be solved using Early Retransmit**. However, early retransmit will not be triggered if there are two or more lost packets.

Continuous loss stall

Continuous loss means that **all outstanding packets (# >=4) in the current window are lost**.

When all outstanding packets are lost, the server has to wait for a timeout, mark all the outstanding packets as lost, and **retransmit all unacknowledged segments**. May lead to congestion.

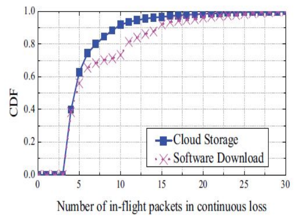


Figure 12: Distribution of in-flight size when continuous loss stalls happen.

Continuous loss stalls happen at any position in flows with similar likelihood.

ACK delay or loss stall

An ACK delay or loss stall happens when a sender does (not?) receive ACKs before the retransmission timer expires, while the segments are identified as not lost through DSACK.

ACK delay can be caused by the delayed-ACK mechanisms.
Trigger a timeout retransmission when delay > RTO.

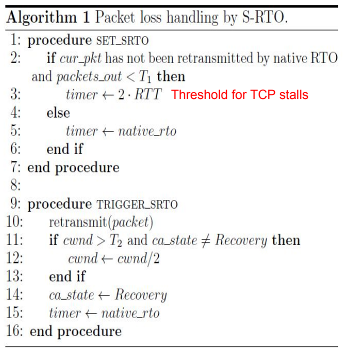
This also explains the observation that software download suffers from more ACK delay stalls, because software download flows are more likely to have a small in-flight size(due to small rwnd), less frequent ACK and more likely delay > RTO.

S-RTO: TOWARDS MITIGATING TCP STALLS

Some TCP stalls, like f-double and tail retransmission stalls in Open state, need to be mitigated by retransmitting unacknowledged packets slightly more aggressively, rather than waiting for an expensive timeout retransmission.

However, simply shortening RTO may trigger spurious retransmissions and subsequently take the sender several RTTs to increase the congestion window from 1 MSS to the original value, during which the sender fails to fully utilize the available bandwidth.

Algorithm



Algorithm 1 Packet loss handling by S-RTO.

```
 1: procedure SET_SRTO
 2:     if $cur\_pkt$ has not been retransmitted by native RTO
        and $packets\_out < T_1$ then
 3:         $timer \leftarrow 2 \cdot RTT$   Threshold for TCP stalls
 4:     else
 5:         $timer \leftarrow native\_rto$
 6:     end if
 7: end procedure
 8:
 9: procedure TRIGGER_SRTO
10:     retransmit($packet$)
11:     if $cwnd > T_2$ and $ca\_state \neq Recovery$ then
12:         $cwnd \leftarrow cwnd/2$
13:     end if
14:     $ca\_state \leftarrow Recovery$
15:     $timer \leftarrow native\_rto$
16: end procedure
```

S-RTO keeps a probe timer for each flow, but is only active when a timeout retransmission (as opposed to fast retransmit) is likely to happen.

packets_out < T1 means in-flight size is relatively small, indicating no enough dupacks for fast retransmission.

Carefully manage the congestion window to avoid that too small cwnd and following stalls.

S-RTO falls back to the native RTO mechanism for recovery if the packet retransmitted by S-RTO is dropped.

Performance evaluation of S-RTO

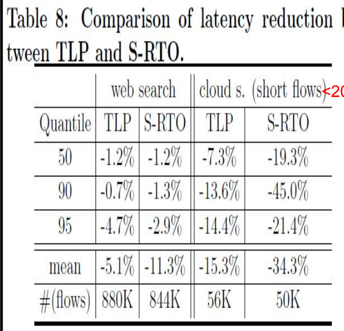


Table 8: Comparison of latency reduction between TLP and S-RTO.

	web search		cloud s. (short flows)<200KB	
Quantile	TLP	S-RTO	TLP	S-RTO
50	-1.2%	-1.2%	-7.3%	-19.3%
90	-0.7%	-1.3%	-13.6%	-45.0%
95	-4.7%	-2.9%	-14.4%	-21.4%
mean	-5.1%	-11.3%	-15.3%	-34.3%
#(flows)	880K	844K	56K	50K

The latency: the time between the client initiates a request and all response packets have been acknowledged.

- S-RTO achieves a mean improvement of 11% on web seatch, because a few (last 1%)flows suffers very long latency due to double retransmission stalls.
- Performance on large flows of cloud storage is not good, because 1. Large flows last longer so some stalls(tail) have less impact; 2.slightly more aggressive retransmissions can increase the congestion of network

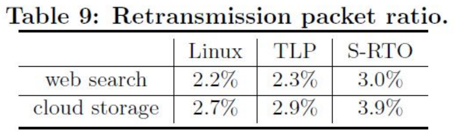


Table 9: Retransmission packet ratio.

	Linux	TLP	S-RTO
web search	2.2%	2.3%	3.0%
cloud storage	2.7%	2.9%	3.9%

Both TLP and S-RTO can trigger unnecessary retransmissions.

Caused by the retransmission of delayed (but not lost) packets.

May slightly increase congestion.

This increase in retransmissions is however reasonable, and it does not hurt TCP fairness as the congestion window still follows the AIMD (Additive-Increase/Multiplicative-Decrease) principle of TCP.

Discussion

How much congestion is introduced? How it could influence the congestion extension at burst?

We appreciate focus on classification “in the wild”, but what about ground-truth? How do we know that this classification is valid?