Demystifying and Mitigating TCP Stalls at the Server Side

Jianer Zhou  
Qinghua Wu  
Zhenyu Li  
Gaogang Xie

Steve Uhlig  
Peter Steenkiste  
Jian Chen

ICT-CAS  
QMUL  
CMU  
Qihoo 360

http://fi.ict.ac.cn
Server’s Role on Internet

Key to connect client and data center

Client

High loss + delay

Front-end server

Data Center

Stable

the TCP connections we focus
TCP Stalls in Server

Example of TCP stalls within a flow

set duration > 2*RTT as a stall

a 10 seconds TCP flow spends over 5 seconds to stall
Our Motivation and Goal

Stalls seriously affect the performance of TCP flow. Mitigating stalls can improve TCP performance.

- A TCP stall root cause diagnosis tool
- Analysis of the stalls in 3 types of services
- Mitigation of TCP stalls
Tapo: Tree-based Diagnosis Tool

TCP analysis tool which analyze side stalls.

https://github.com/tcp-tool/tapo
Dataset

- Collect TCP flows of Qihoo360’s cloud storage, software download and web search services from front-end server.
- Over 7 days, one peak hour each day, totally 3.35 billion packets, 6.4 million flows.

<table>
<thead>
<tr>
<th>Service</th>
<th>#flows</th>
<th>avg. speed (B/s)</th>
<th>avg. flow size</th>
<th>pkt loss</th>
<th>avg. RTT</th>
<th>avg. RTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cloud storage</td>
<td>2.2M</td>
<td>540K</td>
<td>1.7MB</td>
<td>3.9%</td>
<td>143ms</td>
<td>1.2s</td>
</tr>
<tr>
<td>Software down.</td>
<td>0.9M</td>
<td>413K</td>
<td>129KB</td>
<td>4.1%</td>
<td>147ms</td>
<td>1.6s</td>
</tr>
<tr>
<td>Web search</td>
<td>3.3M</td>
<td>644K</td>
<td>14KB</td>
<td>2.1%</td>
<td>106ms</td>
<td>0.9s</td>
</tr>
</tbody>
</table>
RTO vs. RTT

RTO is much larger than RTT, even over 1-10 seconds.
Basic Stall Information

Distribution of ratio of stalled time to transmission time

over 20% of the flows are stalled for half of their lifetime
An Overview of Stalls

Timeout stall dominate stall in all three services.

A quest to look into timeout stalls
Timeout Stall

- Fast retransmit
- Timeout stall

Small cwnd/rwnd → Cont. loss → ACK loss → Double retr. → Tail retr.

<table>
<thead>
<tr>
<th>type</th>
<th>Cloud storage (Time(%))</th>
<th>Software download (Time(%))</th>
<th>Web search (Time(%))</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double retr.</td>
<td>45.4</td>
<td>60.8</td>
<td>41.9</td>
</tr>
<tr>
<td>Tail retr.</td>
<td>5.0</td>
<td>0.4</td>
<td>36.0</td>
</tr>
</tbody>
</table>

Most serious affect web search
Double Retransmission

Two situations trigger double retransmission:

- Timeout retr. packets loss, causes timeout double stall
- Fast retransmit packets loss, causes fast double stall
## Double Retransmission

<table>
<thead>
<tr>
<th></th>
<th>Cloud storage</th>
<th>Software down.</th>
<th>Web search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Timeout double stall</td>
<td>37.7</td>
<td>47.3</td>
<td>44.4</td>
</tr>
<tr>
<td><strong>Fast double stall</strong></td>
<td><strong>62.3</strong></td>
<td><strong>52.7</strong></td>
<td><strong>55.6</strong></td>
</tr>
</tbody>
</table>

In **Recovery state**, the network may not be congested seriously. The lost packet may be recovered by retransmitting it again.
Tail Retransmission

- The lost packet is the last packet, not enough dupacks to trigger fast retransmit.

<table>
<thead>
<tr>
<th></th>
<th>Cloud storage</th>
<th>Software down.</th>
<th>Web search</th>
</tr>
</thead>
<tbody>
<tr>
<td>Open state</td>
<td>60.1</td>
<td>41.3</td>
<td>10</td>
</tr>
<tr>
<td>Recovery state</td>
<td>39.9</td>
<td>58.7</td>
<td>90</td>
</tr>
</tbody>
</table>

- TLP (Tail Loss Probe, sigcomm13) recovers tail retr. just in Open state.
- We try to mitigate those timeout stalls in Open and Recovery state.
Smart RTO

set timer

1. If `packets_out < T1` and `cur_pkt not retrans` is true, set `SRTO_timer(2 * RTT)`.

2. If `SRTO_timer` triggers, retransmit the packet.

3. If `cwnd > T2` and `state != recovery` is true, adjust `cwnd` as `cwnd/2`. Otherwise, adjust `cwnd` as `cwnd`.

lost packets recovery

cwnd adjust
S-RTO Position in TCP

- Recover lost packet
- Fast retransmit
- S-RTO
- Traditional RTO

Slightly aggressive
Reduce the RTO affection

Conservative
Affect TCP performance
S-RTO implementation

- Linux kernel 2.6.32.
- Deploy S-RTO on front-end servers hosting web search and cloud storage.
- For comparison, also deploy TLP in the same kernel.
Evaluation results

S-RTO improves the web search native latency by 11.3%,

<table>
<thead>
<tr>
<th></th>
<th>Web search</th>
<th>Cloud s. (short flows)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>TLP</td>
<td>S-RTO</td>
</tr>
<tr>
<td>Quantile</td>
<td>-1.2%</td>
<td>-1.2%</td>
</tr>
<tr>
<td>50</td>
<td>-4.7%</td>
<td>-2.9%</td>
</tr>
<tr>
<td>95</td>
<td>-5.1%</td>
<td>-11.3%</td>
</tr>
<tr>
<td>mean</td>
<td>-5.1%</td>
<td>-11.3%</td>
</tr>
<tr>
<td>#(flows)</td>
<td>880K</td>
<td>844K</td>
</tr>
</tbody>
</table>

Comparison of latency reduction between TLP and S-RTO
Trade-off

- As an aggressive way to recover lost packets, we indeed see an increased retransmission rate.

<table>
<thead>
<tr>
<th></th>
<th>Linux</th>
<th>TLP</th>
<th>S-RTO</th>
</tr>
</thead>
<tbody>
<tr>
<td>Web search</td>
<td>2.2%</td>
<td>2.3%</td>
<td>3.0%</td>
</tr>
<tr>
<td>Cloud storage</td>
<td>2.7%</td>
<td>2.9%</td>
<td>3.9%</td>
</tr>
</tbody>
</table>

Retransmission packet ratio

- Reducing unnecessary retransmit packets is our future work.
Conclusion

We aim to shed light on the massive TCP stalls and try to mitigate them.

- A TCP analysis tool
- Analysis of the stalls in three services, especially the timeout stalls.
- SRTO: mitigating timeout stall

future
Reduce unnecessary retransmit
Thank you

http://fi.ict.ac.cn