TCP modifications to reduce thin-stream latency

Andreas Petlund
Simula Research Laboratory AS
30.07.2010
Average RTT allows for a satisfactory user experience (in theory).
Maximum RTTs stretch the limit for a satisfactory experience.
When loss occurs and retransmissions must be made to recover, application-layer latencies reach critical levels.
Result: degraded user experience.

Highest observed application-layer latency: 67 seconds!

Interactive thin streams over TCP

<table>
<thead>
<tr>
<th>application (platform)</th>
<th>payload size (Bytes)</th>
<th>packet interarrival time (ms)</th>
<th>avg. bandwidth requirement (pps)</th>
<th>(bps)</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>average</td>
<td>min</td>
<td>max</td>
<td>average</td>
</tr>
<tr>
<td>World of Warcraft</td>
<td>26</td>
<td>6</td>
<td>1228</td>
<td>314</td>
</tr>
<tr>
<td>Anarchy Online</td>
<td>98</td>
<td>8</td>
<td>1333</td>
<td>632</td>
</tr>
<tr>
<td>Age of Conan</td>
<td>80</td>
<td>5</td>
<td>1460</td>
<td>86</td>
</tr>
<tr>
<td>BZFlag‡</td>
<td>30</td>
<td>4</td>
<td>1448</td>
<td>24</td>
</tr>
<tr>
<td>Casa (sensor network)</td>
<td>175</td>
<td>93</td>
<td>572</td>
<td>7287</td>
</tr>
<tr>
<td>Windows remote desktop</td>
<td>111</td>
<td>8</td>
<td>1417</td>
<td>318</td>
</tr>
<tr>
<td>Skype (2 users)‡</td>
<td>236</td>
<td>14</td>
<td>1267</td>
<td>34</td>
</tr>
<tr>
<td>SSH text session</td>
<td>48</td>
<td>16</td>
<td>752</td>
<td>323</td>
</tr>
</tbody>
</table>

‡ Application using TCP fallback due to UDP being blocked by a firewall.

Time-dependent applications
High retransmission latencies
Analysis of TCP for thin streams

Linux TCP flavours (2.6.16) analysed:

- New Reno
- SACK
- DSACK
- FACK
- DSACK+FACK
- Westwood
- BIC
- Vegas

Poor overall performance for interactive thin streams with all tested flavours.

New Reno best “on average” for thin-stream latency.

Thin streams need help with latency!

- Greedy streams (throughput) the driving force in TCP development.
- Mechanisms have been suggested that (partially) address the issue (e.g. Early Retransmit - RFC5827)
- Thin streams need more help to deal with latency issues.

Interactive, thin-stream applications that benefit from the thin-stream mechanisms include stock exchange applications, remote control of PCs (like RDP, VNC and SSH), voice over IP and networked games.
Timeouts and exp. backoff

Retransmission time-out (RTO) will double for each consecutive loss.

Use linear timeouts (LT) for thin streams

TCP and SCTP standard $RTO_{\text{min}}$: 1000ms
TCP in Linux uses a 200ms $RTO_{\text{min}}$
Fast retransmissions

- Thin streams often have $< 1$ packet per RTT.
- Timeout happens before a fast retransmission can be triggered.
- For thin streams: fast retransmit on first received dupACK (mFR)
- Following scheme from Early Retransmit (but consequently retransmit on first dupACK)
Redundant data bundling

- Preempting the experience of loss.
- Will not increase number of sent packets.
- Introduces inherent redundancy.
Thin-stream detection

Retransmission mechanisms:
- packets in flight (PIF) <= 4

Bundling:
- size_unacked(p1) + size(p2) < MSS

- Modifications triggered **only** when these conditions are met.
- All modifications are sender-side only. Tested to work with Windows (XP, Vista, 7), BSD, OSX and Linux as receivers.
Test results and analysis example

(unmodified) TCP New Reno: Exponential increase in latency with each subsequent retransmission.

Available from 2.6.34 Linux Kernel

RTT: 100
IAT: 200
PS: 100
Loss: 5%

Thin-stream modifications: Keep latency low, also when loss occurs
Fairness
Packet-based drop strategy, small buffer

Unmodified TCP: The thin streams are suppressed by the greedy.

- Greedy stream goodput shown
- 1Mbps bottleneck
- 120 Bytes packets
- RTT 100ms

The basic bundling mechanism is too aggressive in very high congestion scenarios.
Greedy stream goodput shown

1Mbps bottleneck

120 Bytes packets

RTT 100ms

Behaviour depends on drop strategy and queue length.
Questions / Discussion

Thin stream  vs  Thick stream