Using MPTCP to Reduce Latency for Cloud Based Mobile Applications

Karl-Johan Grinnemo
Associate Senior Lecturer, Karlstad University
Purpose

• Study and evaluate the use of MPTCP to reduce latency for cloud-based mobile applications

• Considered factors:
  • Traffic type
  • Round-trip time (RTT)
  • Packet loss
Overview of MPTCP

• MPTCP is a set of extensions to standard TCP that enables a single data flow to be separated and carried across multiple connections

• Realize resource pooling

• Three main goals:
  1. Improve throughput
  2. Do not harm
  3. Balance congestion
Operation of MPTCP

Connection Establishment

Additional Subflow
MPTCP Schedulers

• Two tasks: select subflow, select segment

• *Shortest-RTT-First*: send segments on the subflow with shortest RTT, then on the subflow with the next shortest RTT etc. Select segments in FIFO order.

• *Round-Robin*: send segments on each available subflow in round-robin order. Select segments in FIFO order.
Related Work

• Barré et al. studied impact of RTT asymmetry and packet loss on throughput performance

• Chen, Nguyen, Paasch, Raiciu et al. used MPTCP for multihomed 3G/4G/WiFi connections

• Cao, Walid et al. have considered congestion control schemes that provide a better tradeoff between throughput and throughput responsiveness

• Raiciu et al have proposed ”Retransmission and Penalization” for mitigating the effects of HoLB

• Ferlin-Oliviera, Paasch, Dreibholz et al. have studied ways to manage bufferbloat, e.g., ”Multipath Transport Bufferbloat Mitigation” (MPT-BM)
Experiment Setup

**Client**
- Ubuntu Linux 13.10 (64-bit)
- ITG Recv
- Control Script
- MPTCP version 0.88

**Server**
- Ubuntu Linux 13.10 (64-bit)
- ITG Send
- MPTCP version 0.88

**Dnet1**
- FreeBSD 10.0
- Dummynet

**Dnet2**
- FreeBSD 10.0
- Dummynet

**Path #1**

**Path #2**

### General Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Values</th>
</tr>
</thead>
<tbody>
<tr>
<td>Traffic Type</td>
<td>Netflix, Google Maps, Google Docs</td>
</tr>
<tr>
<td>Bandwidth</td>
<td>100 Mbps</td>
</tr>
<tr>
<td>RTT</td>
<td>10 ms, 40 ms, 100 ms, 200 ms, 400 ms</td>
</tr>
<tr>
<td>Packet-loss rate</td>
<td>0%, 2%, 4%</td>
</tr>
</tbody>
</table>

### Linux Kernel Settings

<table>
<thead>
<tr>
<th>Kernel Parameter</th>
<th>TCP</th>
<th>MPTCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>net.core.wmem_max</td>
<td>104857600</td>
<td></td>
</tr>
<tr>
<td>net.core.rmem_max</td>
<td>104857600</td>
<td></td>
</tr>
<tr>
<td>net.ipv4.tcp_congestion_control</td>
<td>cubic</td>
<td>coupled</td>
</tr>
<tr>
<td>net.ipv4.tcp_wmem</td>
<td>4096 87380</td>
<td>104857600</td>
</tr>
<tr>
<td>net.ipv4.tcp_rmem</td>
<td>4096 87380</td>
<td>104857600</td>
</tr>
<tr>
<td>net.mptcp.mptcp_enabled</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>net.mptcp.mptcp_debug</td>
<td>N/A</td>
<td>0</td>
</tr>
<tr>
<td>net.mptcp.mptcp_checksum</td>
<td>N/A</td>
<td>1</td>
</tr>
<tr>
<td>net.mptcp.mptcp_path_manager</td>
<td>N/A</td>
<td>fullmesh</td>
</tr>
<tr>
<td>net.mptcp.mptcp_syn_retries</td>
<td>N/A</td>
<td>3</td>
</tr>
</tbody>
</table>
Google Maps
Loss Free Paths

Tests with same RTT on both paths

Tests with different RTT on both paths
Lossy Paths, Equal RTT

(a) $RTT_1 = 10$ ms, $RTT_2 = 10$ ms
(b) $RTT_1 = 40$ ms, $RTT_2 = 40$ ms
(c) $RTT_1 = 100$ ms, $RTT_2 = 100$ ms

Netflix

(a) $RTT_1 = 10$ ms, $RTT_2 = 10$ ms
(b) $RTT_1 = 40$ ms, $RTT_2 = 40$ ms
(c) $RTT_1 = 100$ ms, $RTT_2 = 100$ ms

Google Docs
Lossy Paths, Unequal RTT

(a) $RTT_1 = 10$ ms, $RTT_2 = 40$ ms

(b) $RTT_1 = 10$ ms, $RTT_2 = 100$ ms

Netlix

(a) $RTT_1 = 10$ ms, $RTT_2 = 40$ ms

(b) $RTT_1 = 10$ ms, $RTT_2 = 100$ ms

Google Docs
Conclusions

• Could provide gains for intense traffic such as Netflix and Google Maps

• Latency reductions possible despite limited differences in RTT and packet loss

• Larger differences in RTT between paths significantly increase latency