Using MPTCP to Reduce Latency for Cloud Based Mobile Applications

Karl-Johan Grinnemo
Associate Senior Lecturer, Karlstad University

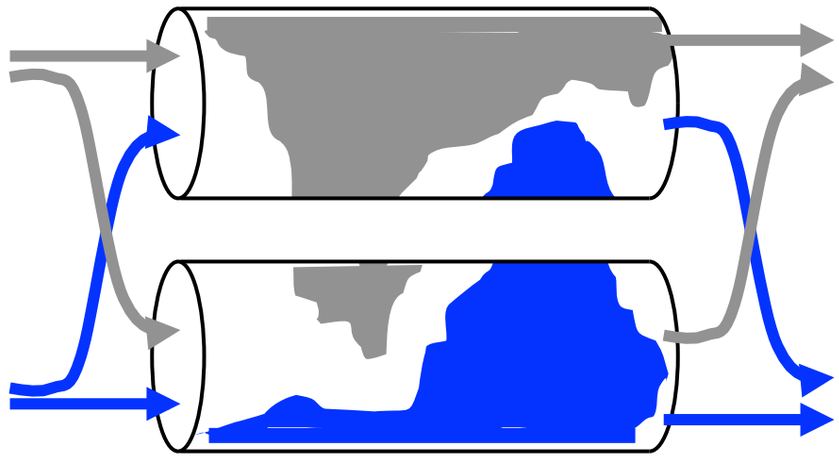
The 2nd International NorNet Users Workshop (NNUW-2), Simula, Oslo, August 28-29, 2014

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

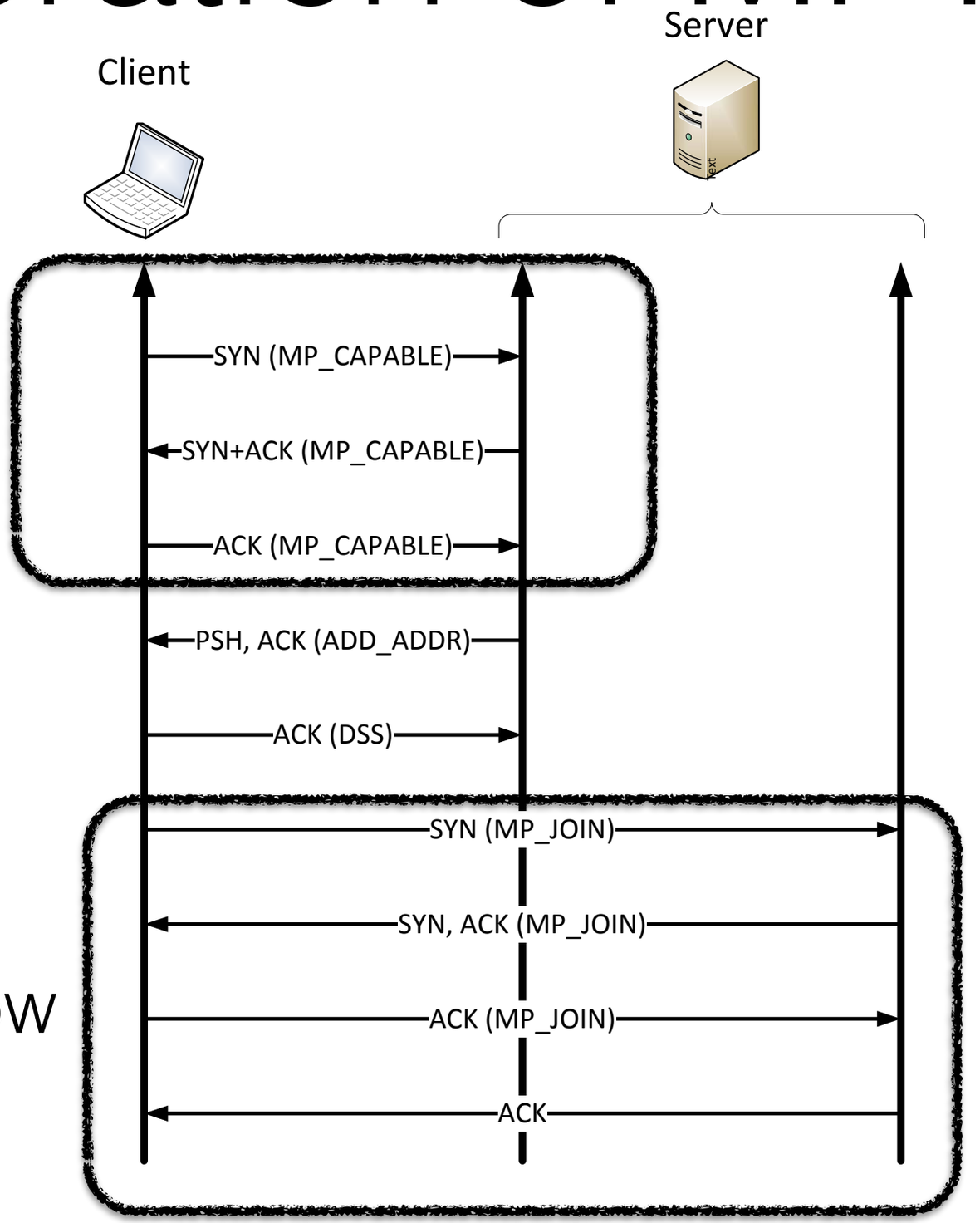
    2. Do not harm

    3. Balance congestion

Operation of MPTCP

Connection Establishment

Additional Subflow



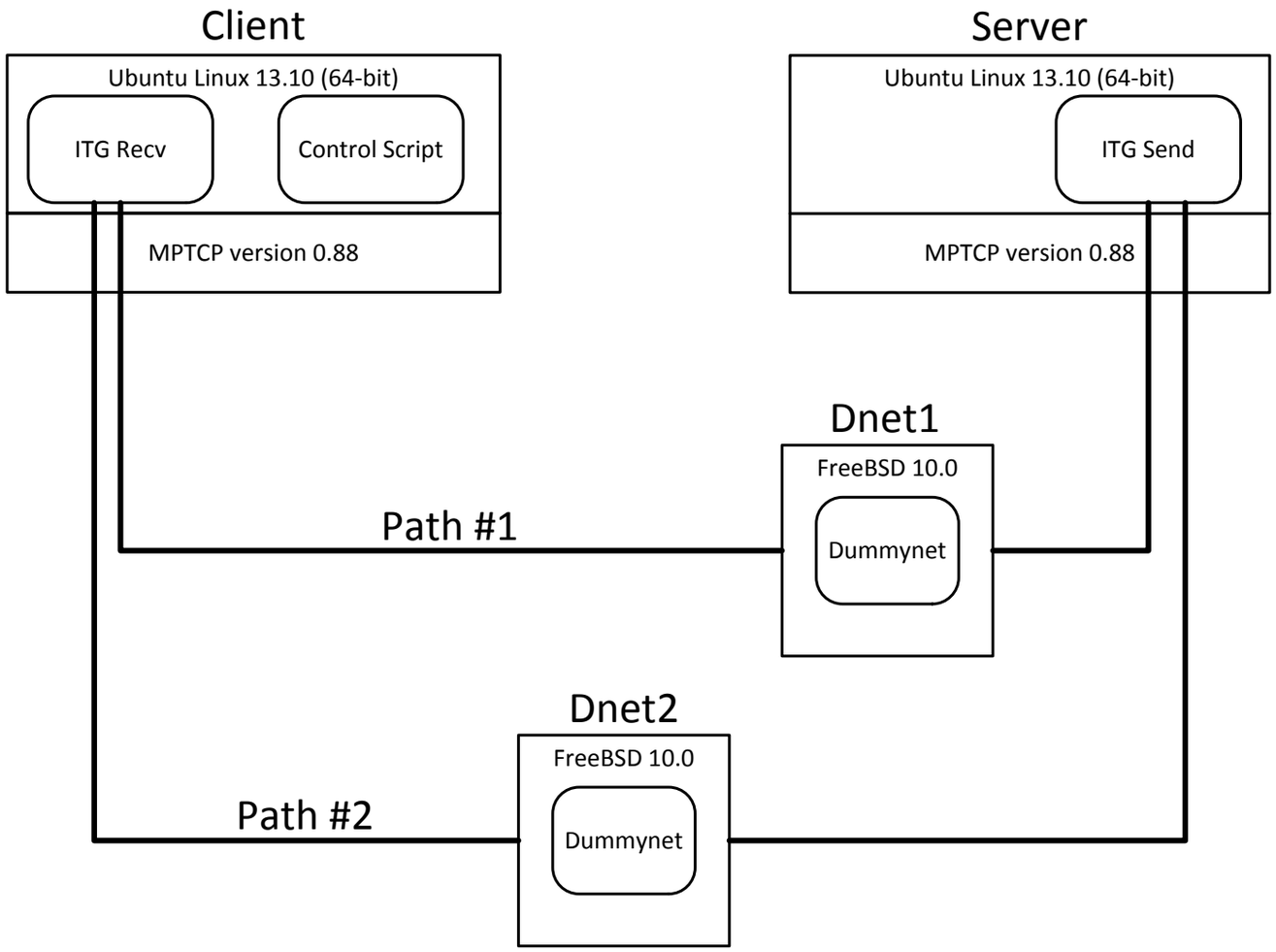
MPTCP Schedulers

- 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



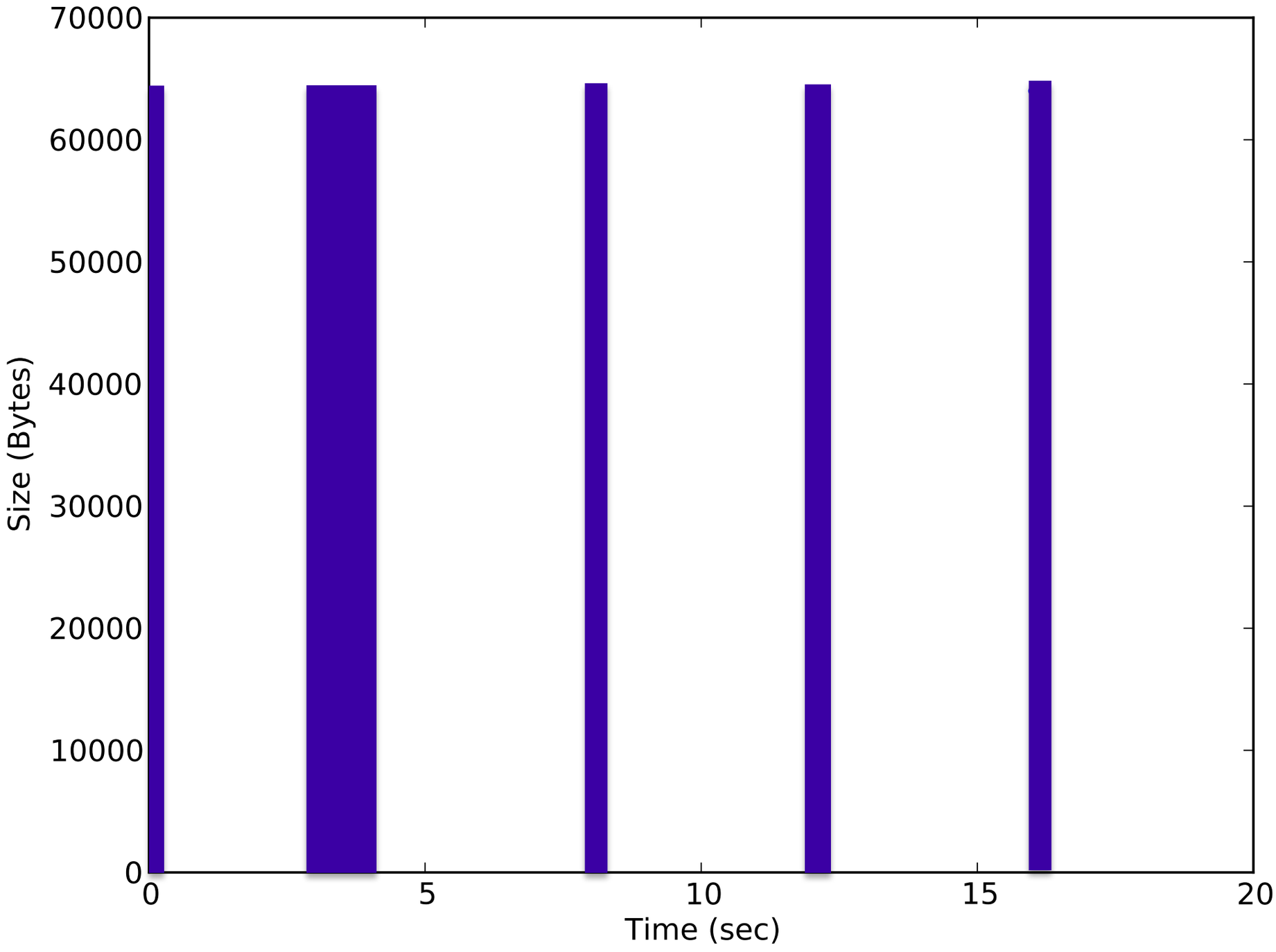
GENERAL SETTINGS

Parameter	Values
`Traffic Type`	Netflix, Google Maps, Google Docs
`Bandwidth`	100 Mbps
`RTT`	10 ms, 40 ms, 100 ms, 200 ms, 400 ms
`Packet-loss rate`	0%, 2%, 4%

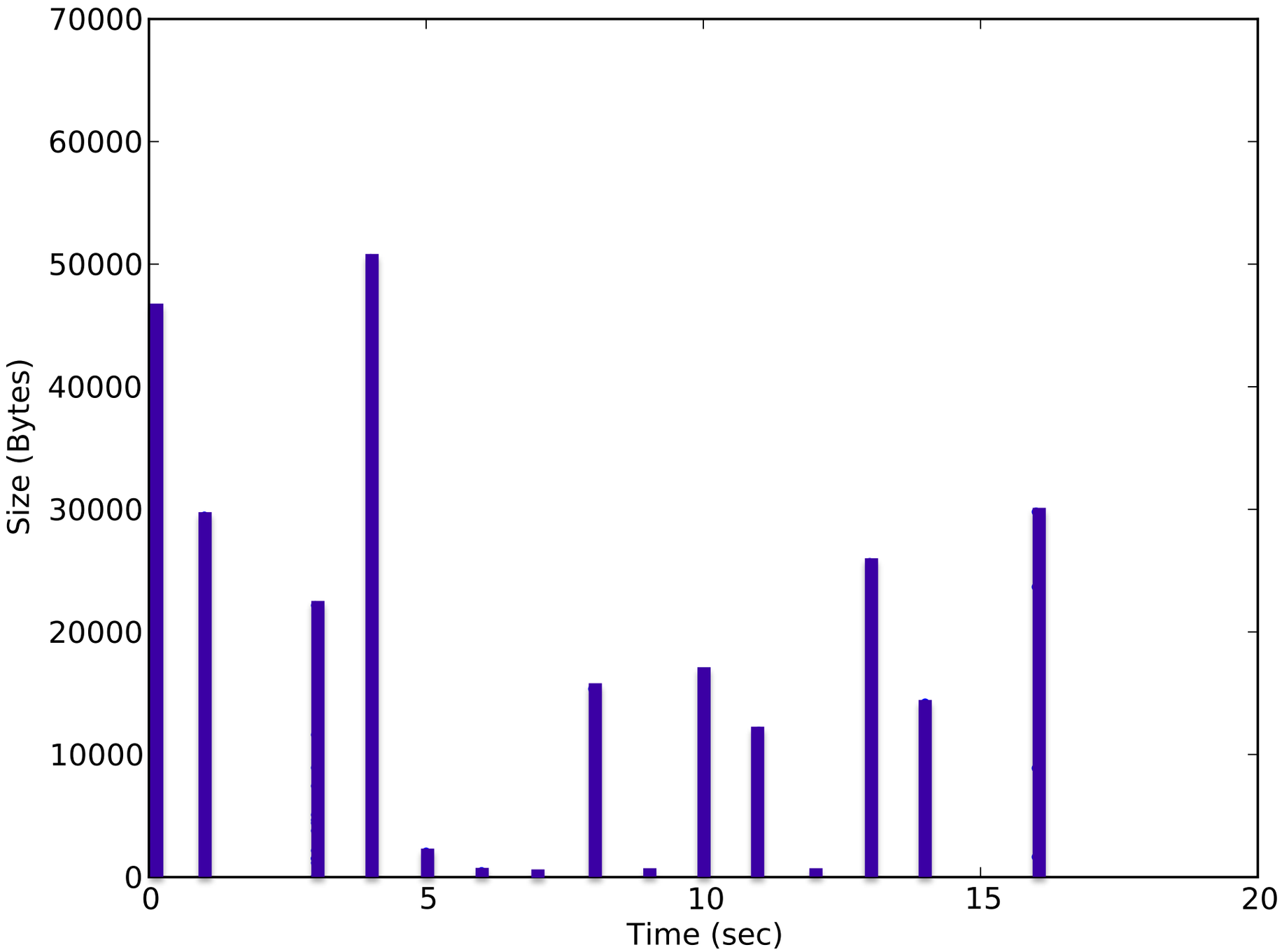
LINUX KERNEL SETTINGS

Kernel Parameter	TCP	MPTCP
`net.core.wmem_max`	104857600	
`net.core.rmem_max`	104857600	
`net.ipv4.tcp_congestion_control`	cubic	coupled
`net.ipv4.tcp_wmem`	4096 87380 104857600	
`net.ipv4.tcp_rmem`	4096 87380 104857600	
`net.mptcp.mptcp_enabled`	0	1
`net.mptcp.mptcp_debug`	N/A	0
`net.mptcp.mptcp_checksum`	N/A	1
`net.mptcp.mptcp_path_manager`	N/A	fullmesh
`net.mptcp.mptcp_syn_retries`	N/A	3

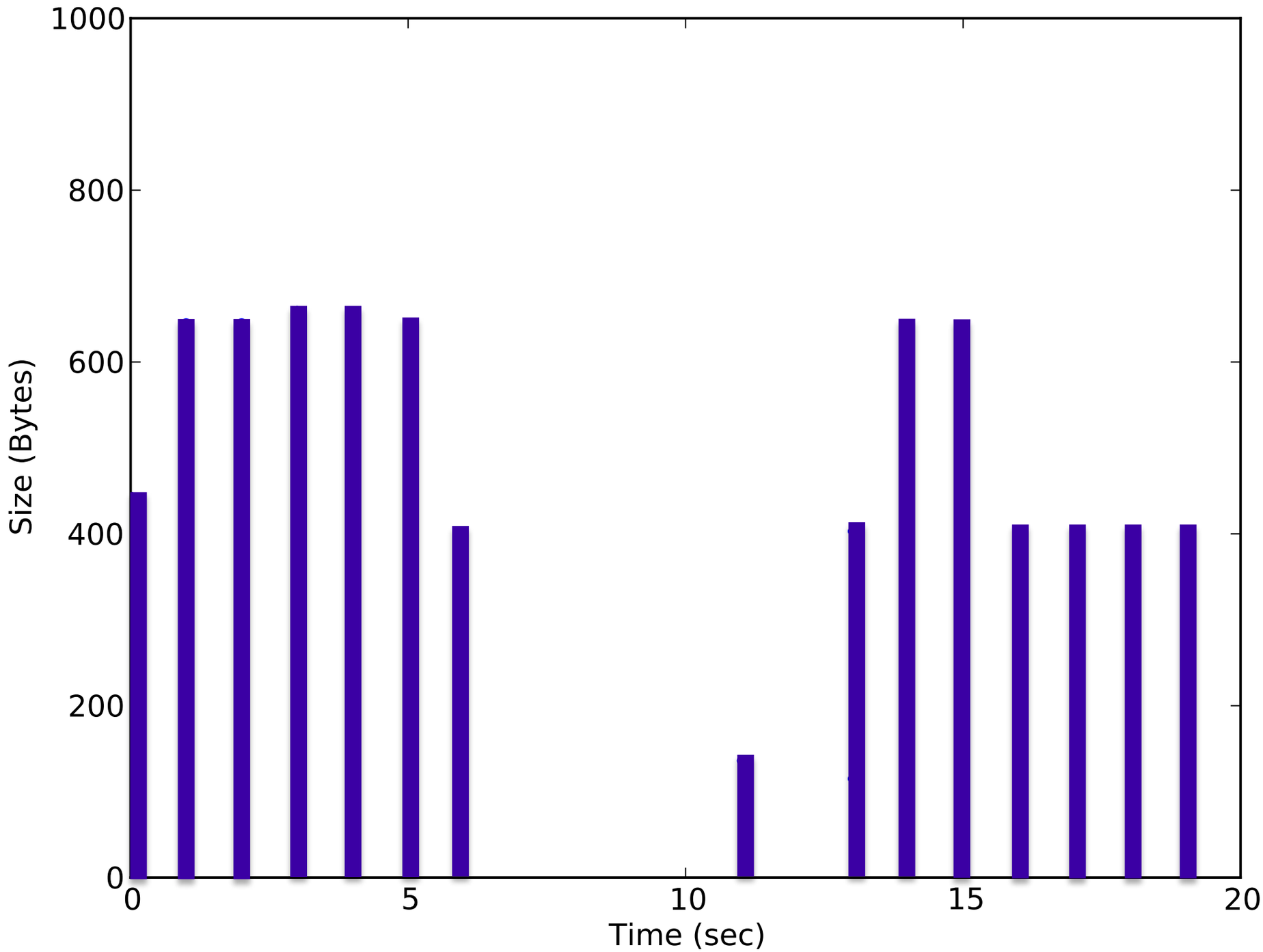
Netflix


70000
60000
50000
40000
30000
20000
10000
0
Size (Bytes)
0
5
10
15
20
Time (sec)

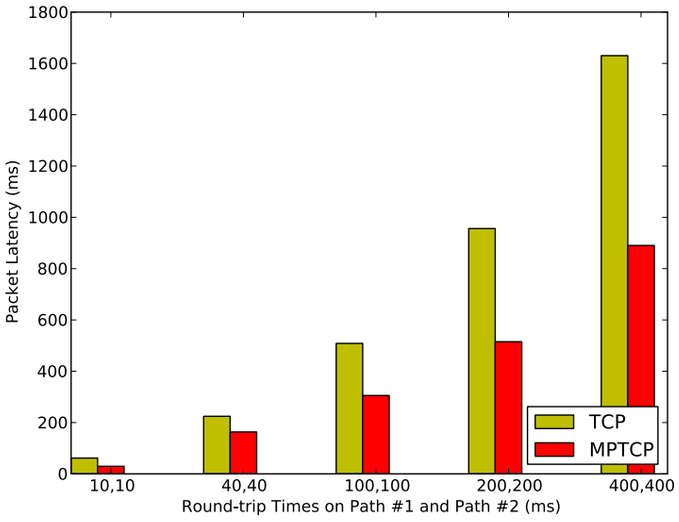
Google Maps


70000
60000
50000
40000
30000
20000
10000
0
Size (Bytes)
0
5
10
15
20
Time (sec)

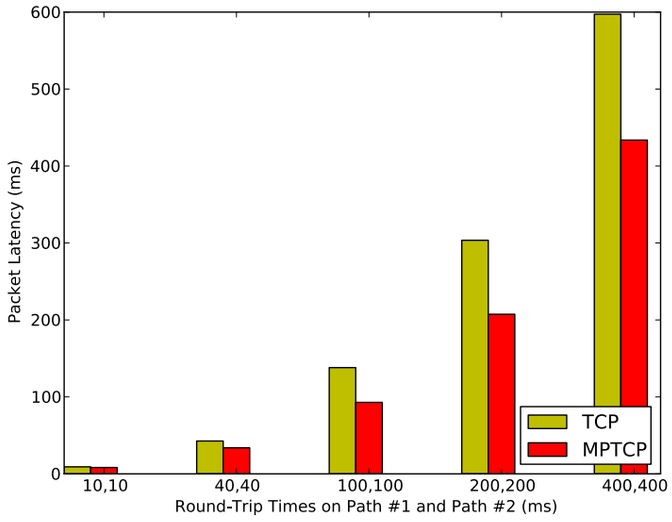
Google Docs



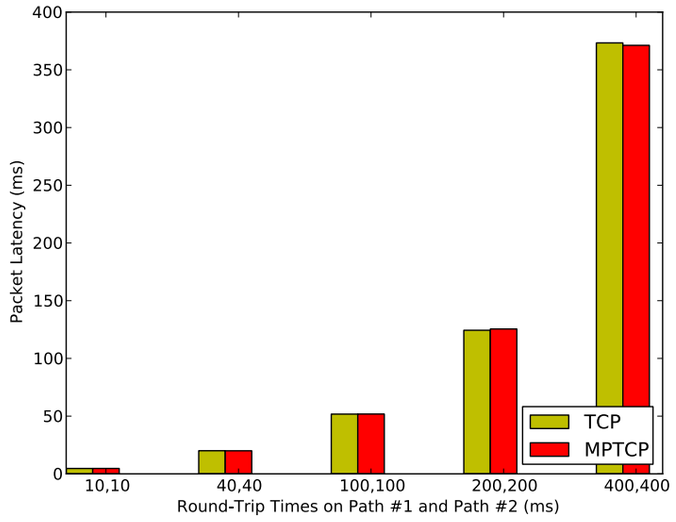
Loss Free Paths



(a) Netflix

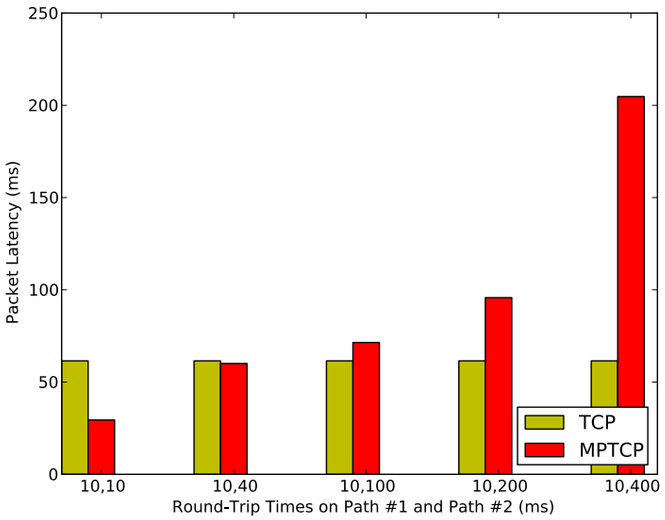


(b) Google Maps

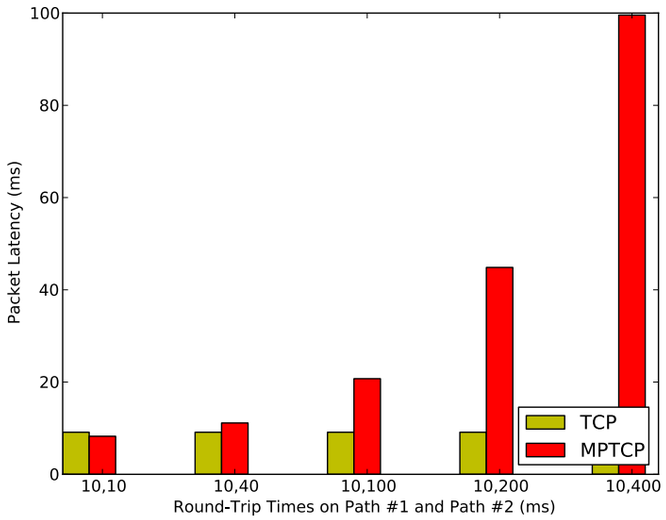


(c) Google Docs

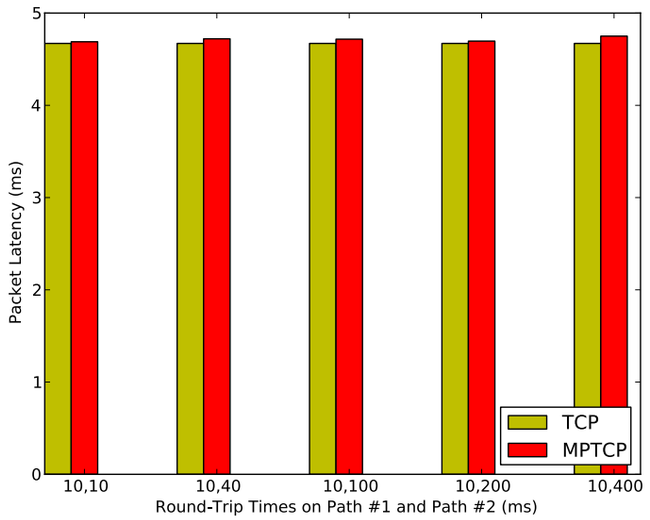
Tests with same RTT on both paths



(a) Netflix



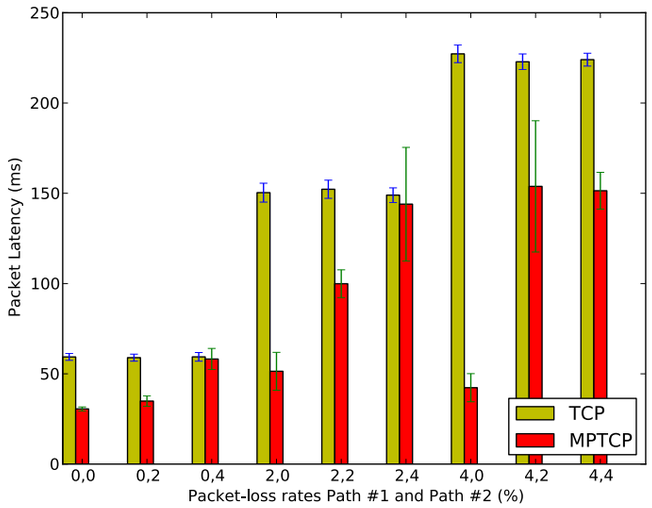
(b) Google Maps



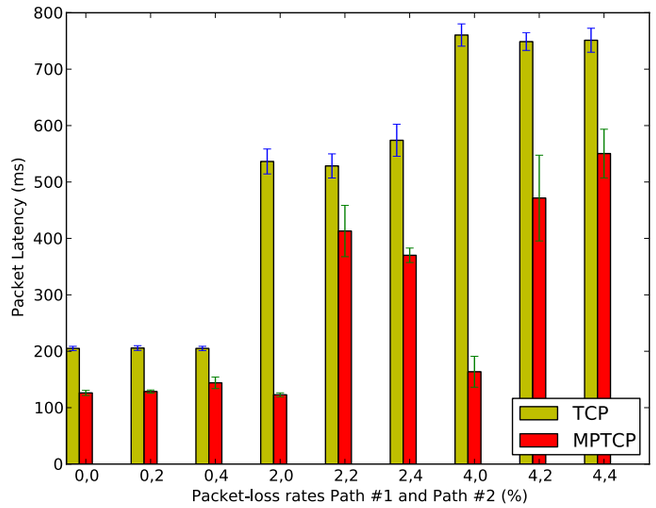
(c) Google Docs

Tests with different RTT on both paths

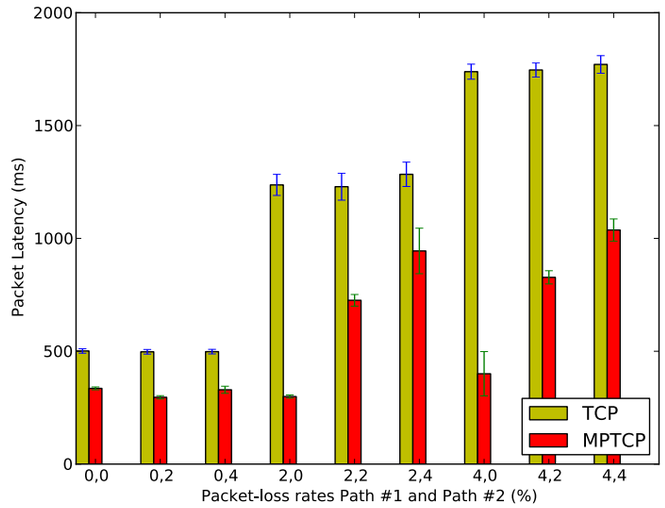
Lossy Paths, Equal RTT



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

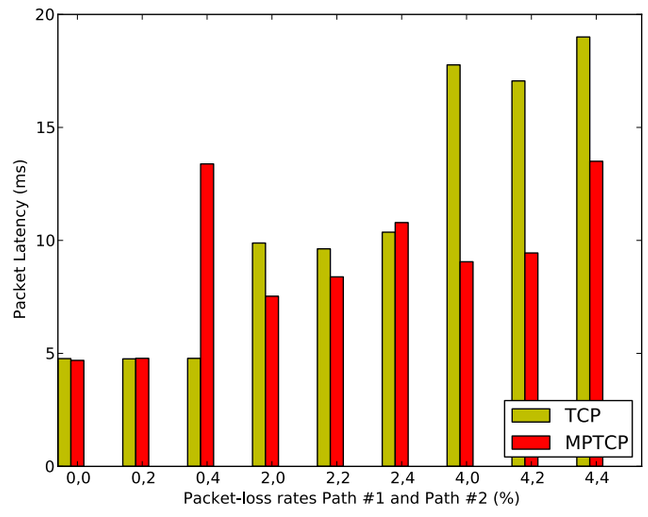


(b) RTT_1 = 40 ms, RTT_2 = 40 ms

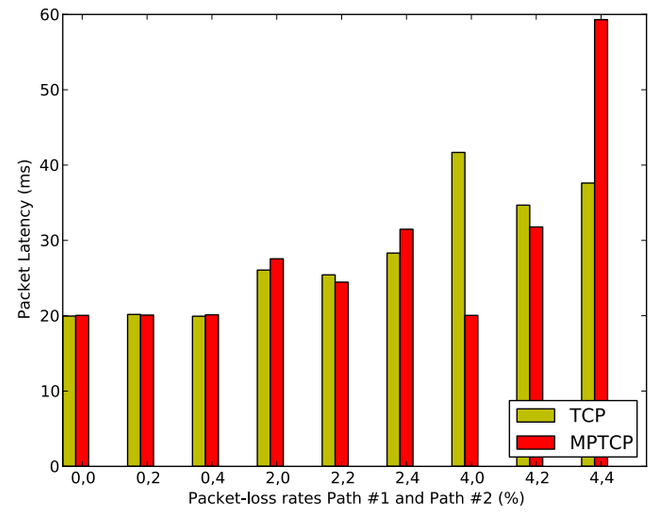


(c) RTT_1 = 100 ms, RTT_2 = 100 ms

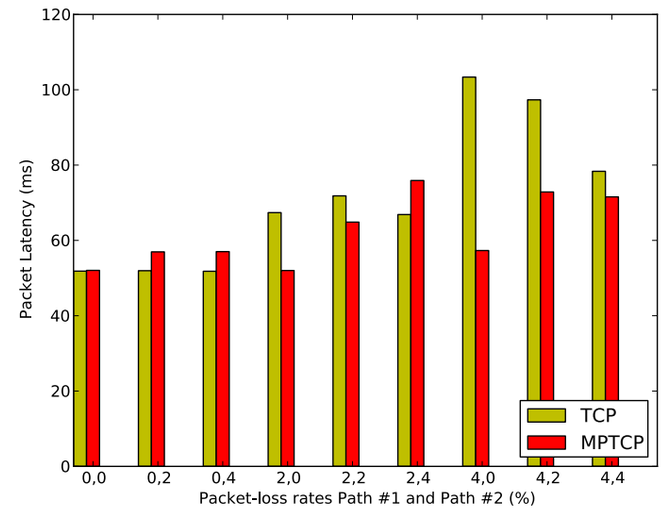
Netflix



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



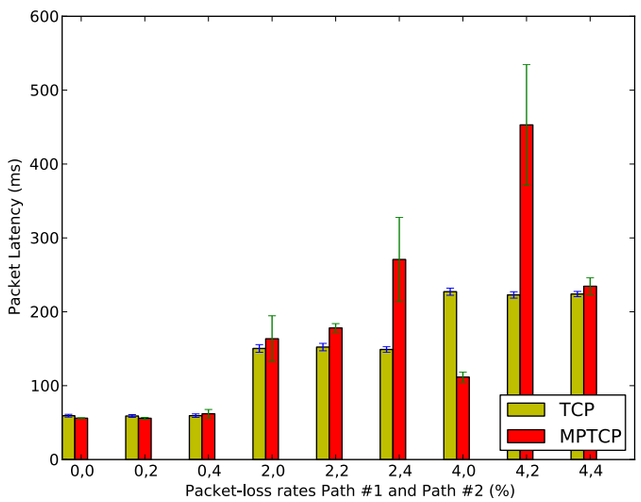
(b) RTT_1 = 40 ms, RTT_2 = 40 ms



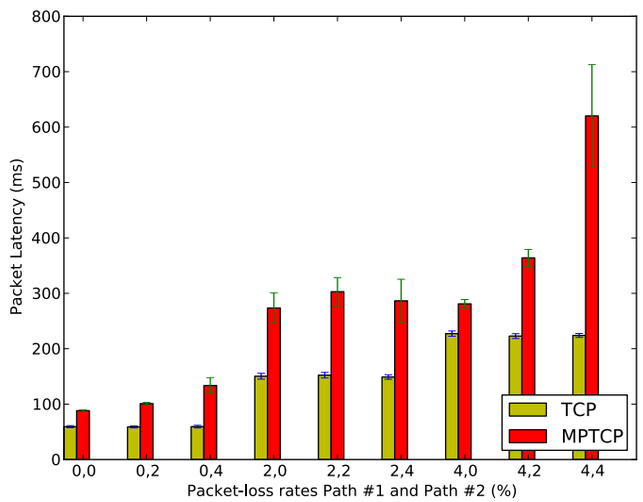
(c) RTT_1 = 100 ms, RTT_2 = 100 ms

Google Docs

Lossy Paths, Unequal RTT

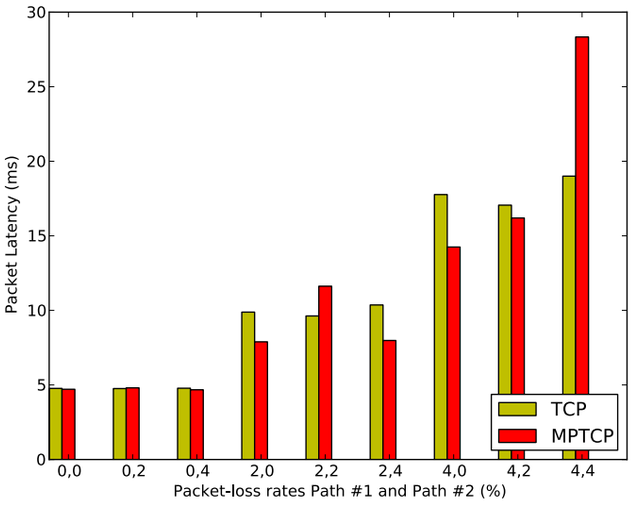


(a) RTT_1 = 10 ms, RTT_2 = 40 ms

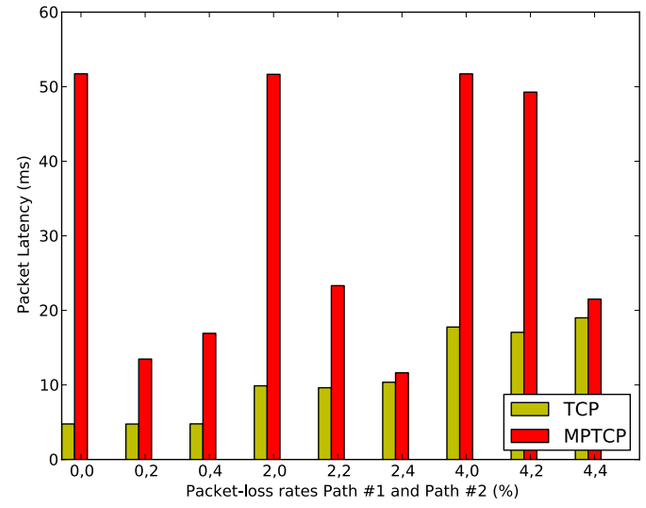


(b) RTT_1 = 10 ms, RTT_2 = 100 ms

Netflix



(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