Low-Latency Scheduling for MPTCP

Per Hurtig

September 12, 2018

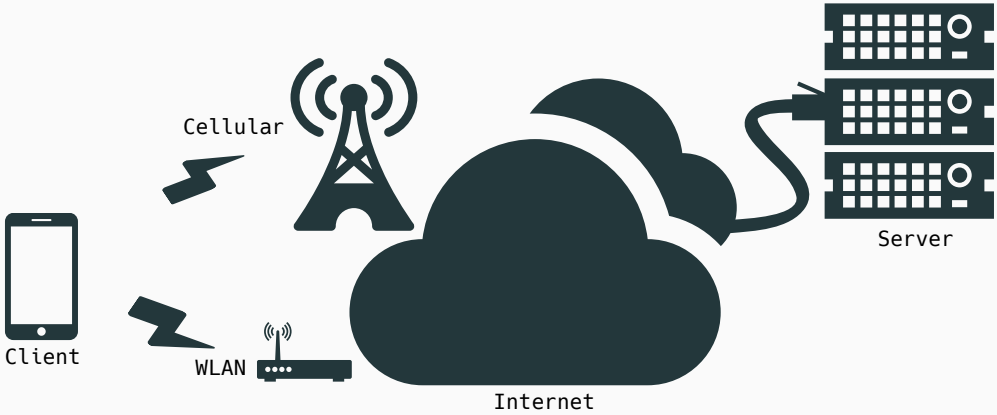
Low latency as an enabler for future Internet services

Agenda

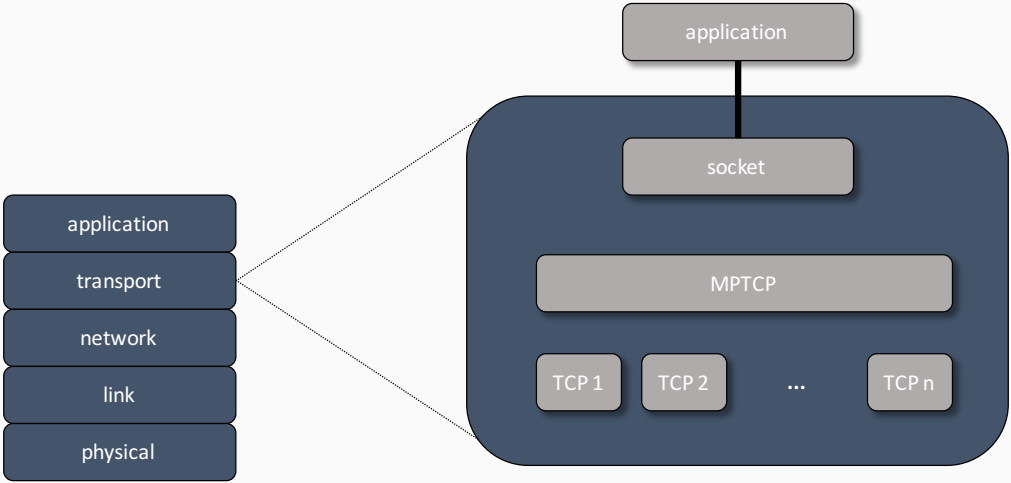
1. Multi-path TCP (MPTCP)
2. Problems, problems, problems
3. Low-Latency Scheduling for MPTCP
4. Summary

Multi-path TCP (MPTCP)

MPTCP Use Case



MPTCP Architecture

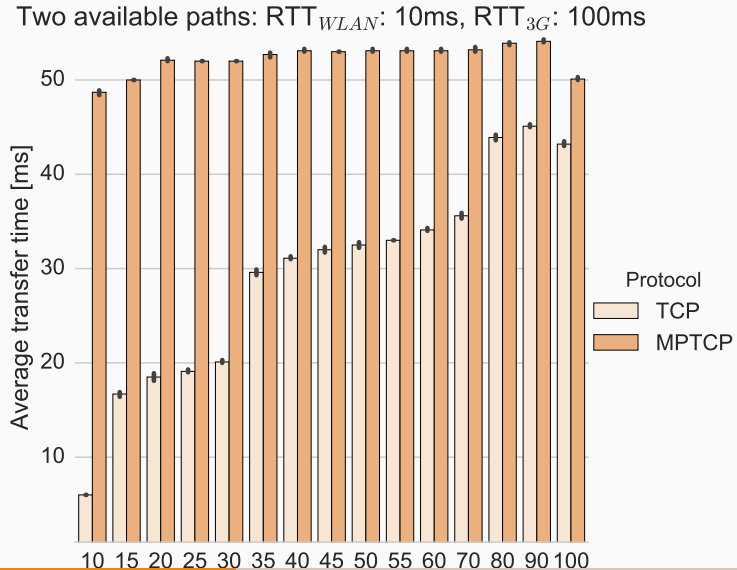


MPTCP

- MPTCP
  - defined in RFC 6824
  - set of extensions to TCP
- Allows multiple subflows per connection
  - one per network interface
- MPTCP makes use of a coupled congestion control
  - defined in RFC 6356
  - for network fairness

Problems, problems, problems

MPTCP


Two available paths: RTT$_{WLAN}$: 10ms, RTT$_{3G}$: 100ms
Average transfer time [ms]
10
20
30
40
50
10 15 20 25 30 35 40 45 50 55 60 70 80 90 100
Protocol
TCP
MPTCP

Why?

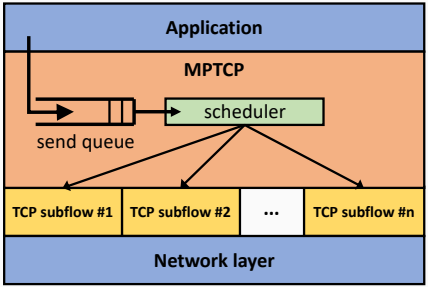
Shouldn't a multi-path protocol perform well over multiple paths?

- MPTCP does not handle asymmetric paths very well
  - The **scheduling** of data on different paths becomes a problem
  - Throughput and, especially, latency are negatively affected
- Blocking effects occur when too much data arrives out-of-order
- Also, poor interaction effects between MPTCP and TCP
  - RTT-reset problem
  - TCP Small Queues (TSQ)
  - ...

MPTCP Scheduling

Linux MPTCP provides a modular scheduling framework

- Default scheduler uses path RTTs
  - uses path with lowest RTT
  - only *available* paths
- Limitations
  - **path asymmetry not considered…**
  - path RTT too simplistic metric
  - unavailable paths are sometimes a better choice
  - interaction effects
    - TCP/MPTCP



Proposed Scheduling Enhancements

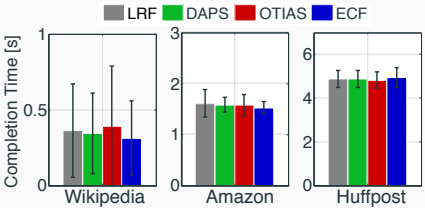
A number of schedulers have been proposed

LRF The default scheduler, schedules on the path with lowest RTT

DAPS Tries to maximize probability of in-order arrival

OTIAS Schedules data to minimize transmission time

ECF Waits for fast, unavailable, paths to become available



Low-Latency Scheduling for MPTCP

Improved Scheduling

- Most schedulers does not consider full or busy paths
- Why not, if the resulting transmission becomes faster?
- Our algorithm is based on the shortest transfer time (STTF)
  - calculates the transmission time for each segment over each path
  - ignores if a path is not available or already filled
  - the implementation also address TCP/MPTCP interaction problems (RTT-reset, TSQ, ...)

Shortest Transfer Time First (STTF) Scheduling

The idea behind STTF is very simple; for each segment to schedule, calculate its transmission time considering:

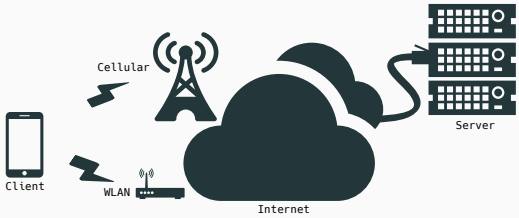
- data already in flight/queued
- current congestion state (slow-start/cong-avoid)

Algorithm 1 STTF Scheduling

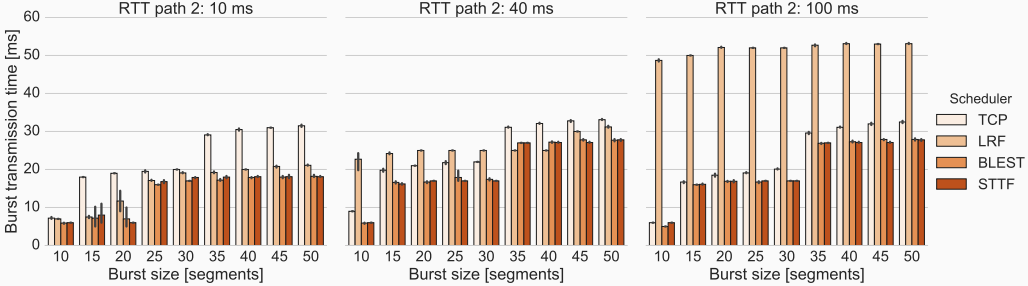
```
1: mptcp_sttf_reschedule()
2: for each unsent segment p do
3:     for each subflow s do
4:         T_s^p = transfer_time(s,p)
5:         if T_s^p < T_min then
6:             T_min = T_s^p
7:             selected_subflow = s
```

Experiments

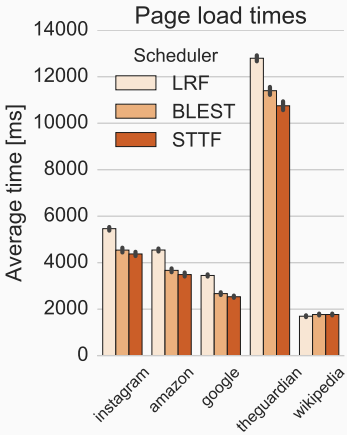
- STTF is implemented in Linux MPTCP
- STTF has been thorougly evaluated:
  - static burst experiments
  - synthetic traffic
    - Web
    - Google Maps
    - Netflix
  - live experiments (MONROE testbed)

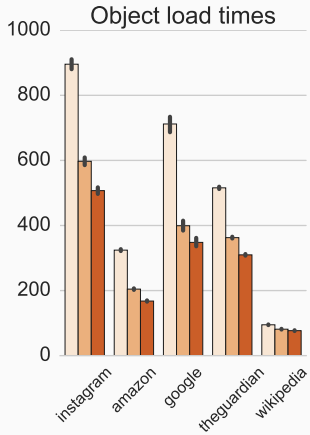


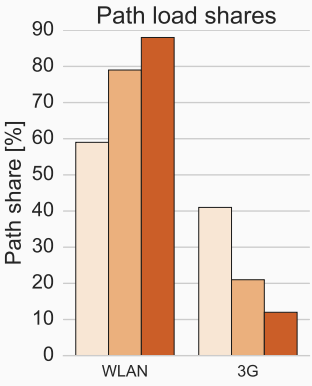
Static Burst Results



Web Experiments







Summary

- MPTCP does not handle asymmetric paths well
- Especially, the data scheduler cannot provide low-latency transmission
- By explicitly considering latency, more effective scheduling strategies (e.g. STTF) can be implemented
- Poor interaction effects between TCP and MPTCP
  - not previously known effects
  - probably many undiscovered side-effects

Questions/Comments?