Multipath TCP: Protocol Overview and Research Areas

Nigel Williams

njwilliams@swin.edu.au

Centre for Advanced Internet Architectures (CAIA)
Swinburne University of Technology

Talk Summary

- MPTCP overview
- Implementing it in FreeBSD
- Research Goals

MPTCP

- Proposed Extensions to TCP
  - A. Ford, C. Raiciu, M. Handley, O. Bonaventure, S. Barre
  - … and others!
- Allow a single TCP connection to use multiple paths*

*the specification is for devices with multiple *addresses*

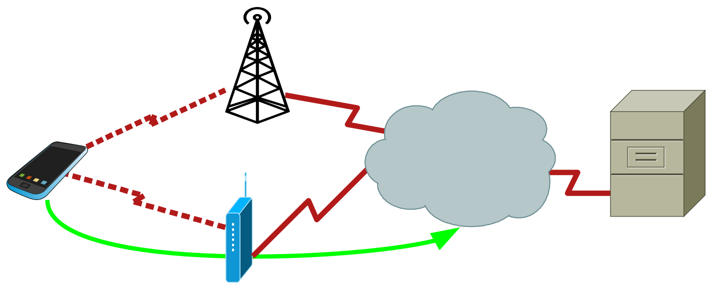
Why MPTCP?

- Devices have more paths
  - Wifi + 3G mobile devices.
  - Wifi + 3G + Ethernet netbooks.
  - Multi-homed servers.
  - Data centres (various topologies).
- Many applications use TCP
  - But TCP doesn't take advantage of these extra paths.

Example Scenario

- Mobile TCP Session

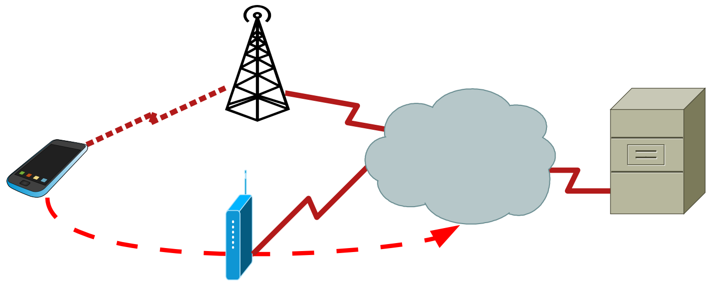
  Uses only one of the available paths.



Example Scenario

- Mobile TCP Session

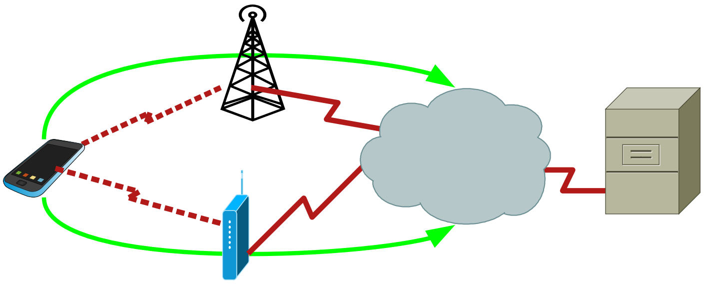
  The connection drops when wifi is no longer available.



Example Scenario

- Mobile MPTCP Session

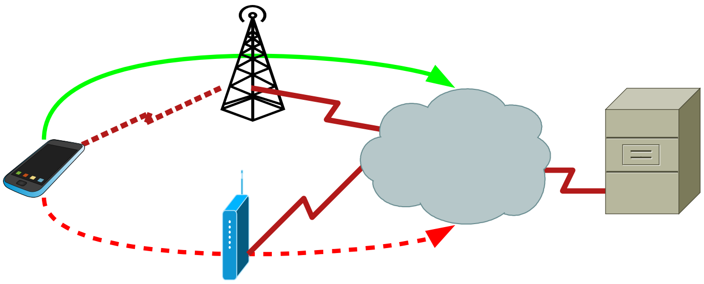
  The connection now has multiple paths associated with it.



Example Scenario

- Mobile MPTCP Session

  The connection is maintained by moving traffic to 3G when wifi fails.



In a nutshell

- Adds Redundancy
  - Maintains a connection when links fail.
- Reduces Congestion
  - Steers traffic away from congested links.
- Increases efficiency
  - Makes use of additional paths.

Why extend TCP?

- Could use SCTP
  - Doesn't always work well on today's Internet (e.g. with middleboxes).
  - Applications need to add support for sctp.
- Or add more TCP connections
  - Need to modify applications.
  - Don't get congestion control benefits, potentially increase congestion. (unless you implement coupled cc at application).
- MPTCP
  - Designed to work in the Internet as it is today.
  - Works with existing applications.
  - Still compatible with single-path TCP.

MPTCP Design

- Basic architectural overview
  - See 'Links and related info' for more depth.
  - More details on why design decisions were made in the related docs.
- Basically a single TCP connection that has multiple subflows
  - Application 'sees' a single TCP connection.
  - Each of the subflows acts a bit like a standard TCP session (with some important modifications).
  - Control using TCP Options field – there is a new MPTCP option, which has its own subtypes.
  - Use congestion control to help decide which path to send traffic on.

MPTCP Design

- MPTCP Control messages passed in TCP options field

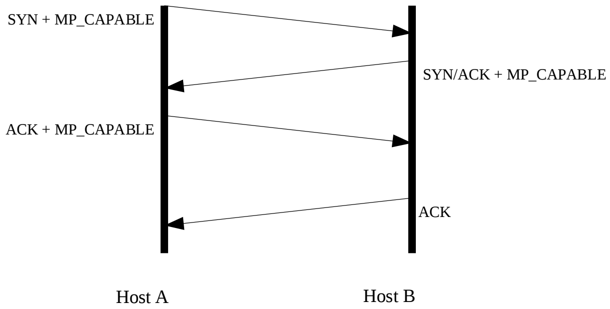
20 Bytes	40 Bytes	
TCP Header	TCP Options	Data

- MPTCP subtypes

Symbol	Name	Value
MP_CAPABLE	Multipath Capable	0x0
MP_JOIN	Join Connection	0x1
DSS	Data Sequence Signal (Data ACK and Data Sequence Mapping)	0x2
ADD_ADDR	Add Address	0x3
REMOVE_ADDR	Remove Address	0x4
MP_PRIO	Change Subflow Priority	0x5
MP_FAIL	Fallback	0x6
MP_FASTCLOSE	Fast Close	0x7

MPTCP Connection Set up

- Piggyback on 3-way handshake
  - The MP_CAPABLE option is included in the handshake phase
  - Though it's actually a 4-way handshake before multipath is enabled
  - MP_JOIN adds new subflows once a connection is established

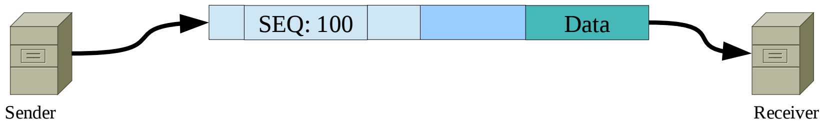


Data Sequence Space

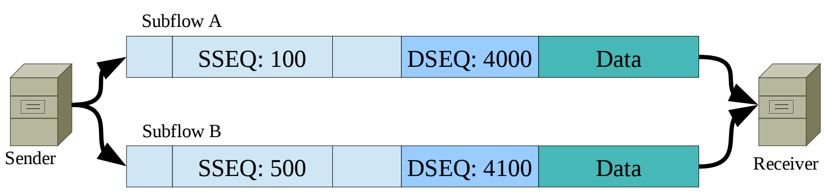
- How does regular TCP work?
  - Segment data and use sequence numbers to track the segments
  - Allows retransmits, reassembly etc
- MPTCP needs to aggregate segments from multiple subflows
  - Paths may have different BW, RTT, so re-ordering is likely
  - Each subflow should retain its own sequence space (e.g. to prevent trouble with middleboxes)
- Connection-level sequence space
  - Use data-sequence numbers to aggregate segments from multiple subflows. Pass data sequence numbers as TCP options.
  - Subflows share send and receive buffers (but still perform reassembly and retransmit at the subflow level).

Data Sequence Space

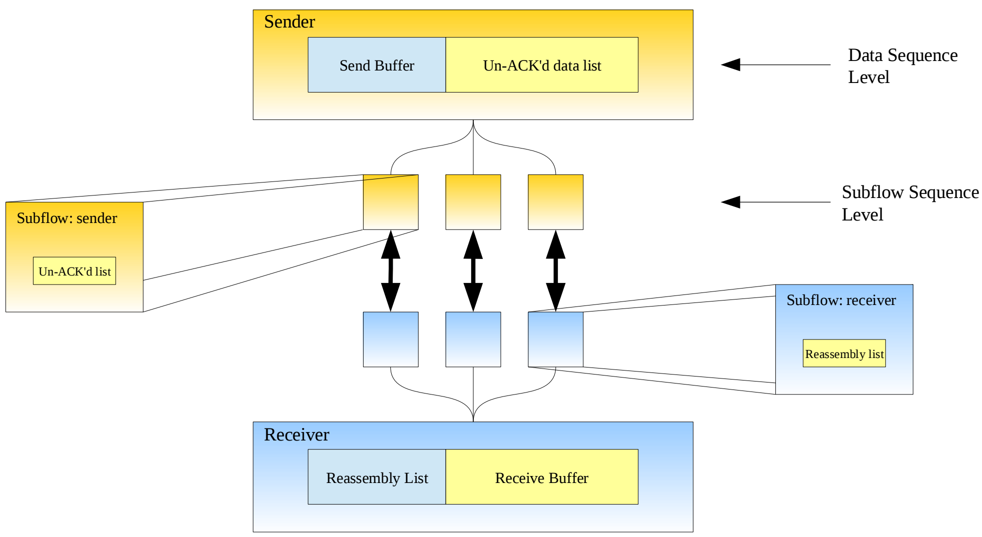
Standard TCP sequence numbering



MTCP sequence numbering – with subflow sequence and data sequence numbers

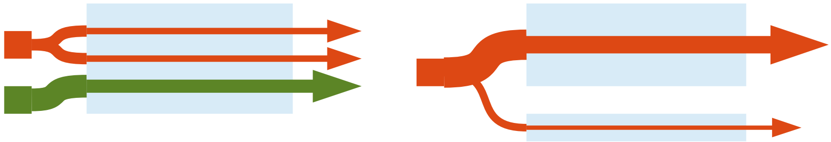


Sequence Spaces



Congestion Control

- Protocol designers' stated CC goals:
  - 1. “Perform at least as well as a single path flow would on the best of the paths available to it” (improve throughput)
  - 2. “Should not take up more capacity from any of the resources shared [paths] ... than if it were a single flow using one of these paths” (do not harm)
  - 3. “Should move as much traffic as possible off its most congested paths, subject to meeting the first two goals” (balance congestion)



Fair at bottlenecks

Use the most efficient path

[C. Raiciu, C. Paasch “Multipath TCP” Google Tech Talk, 2012, http://youtu.be/02nBaaIoFWU]
[C. Raicu, M. Handly, D. Wischik “Coupled Congestion Control for Multipath Transport Protocols, RFC 6356]

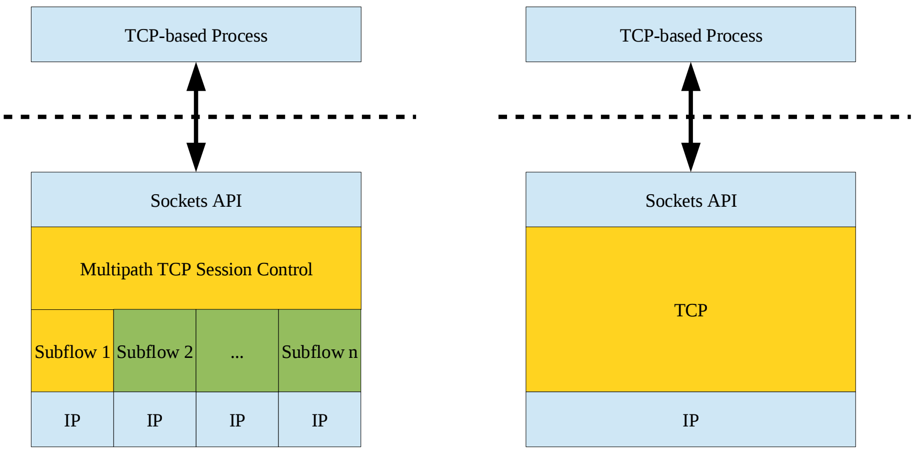
Congestion Control

- Designers recommend a “resource pooling” MPTCP congestion control algorithm
  - cwnd for each subflow, and the total cwnd of all the subflows. The CC algorithm uses this (and RTT) to adjust the window.
    - Moves segments away from congested links.
    - Favours higher capacity links.
    - Treats multiple links as a single pool of capacity.
    - Tune aggressiveness to match regular TCP
- Not part of the MPTCP specification – other algorithms need to be investigated

Implementation

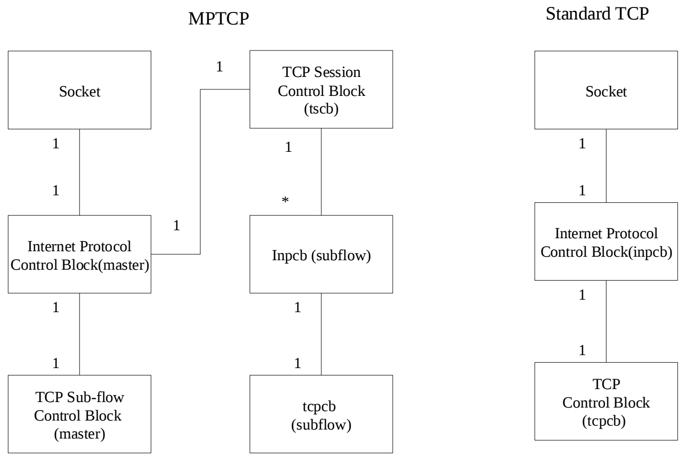
- Designed with research in mind
  - Should meet specification, but have hooks to change behaviour (adjust cc, retransmit strategies, access to subflows).
- Implemented as a shim in the Kernel
  - Need to change the TCP stack directly.
  - TCP Processes will still call the same socket API.

Implementation



*An MPTCP Connection with a single subflow acts like standard TCP

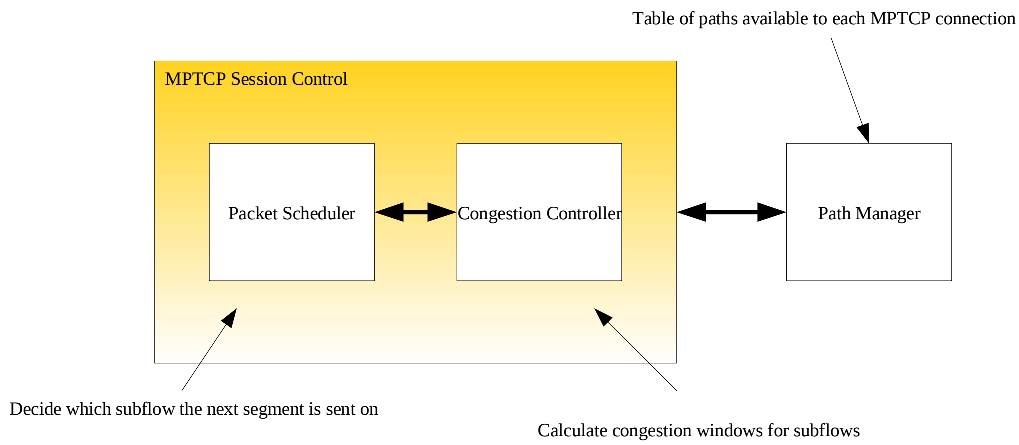
Protocol block changes



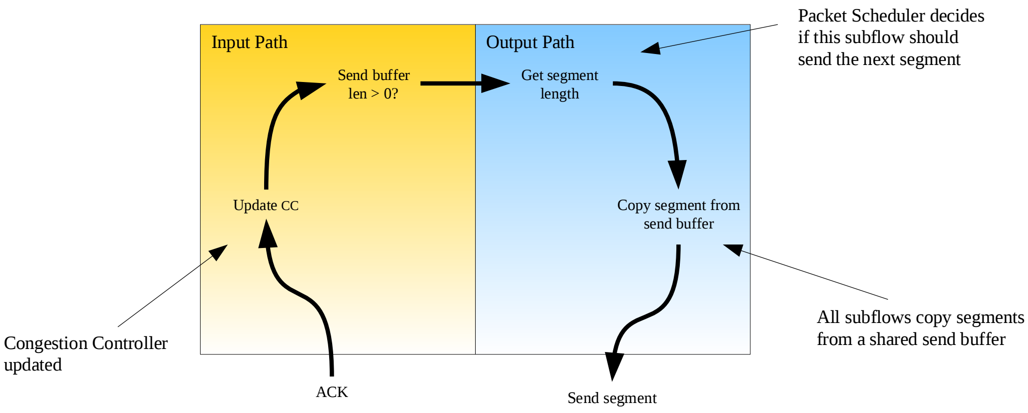
- (inpcb, tcpcb) for each subflow
- Always one tscb and at least one inpcb and tcpcb

Implementation

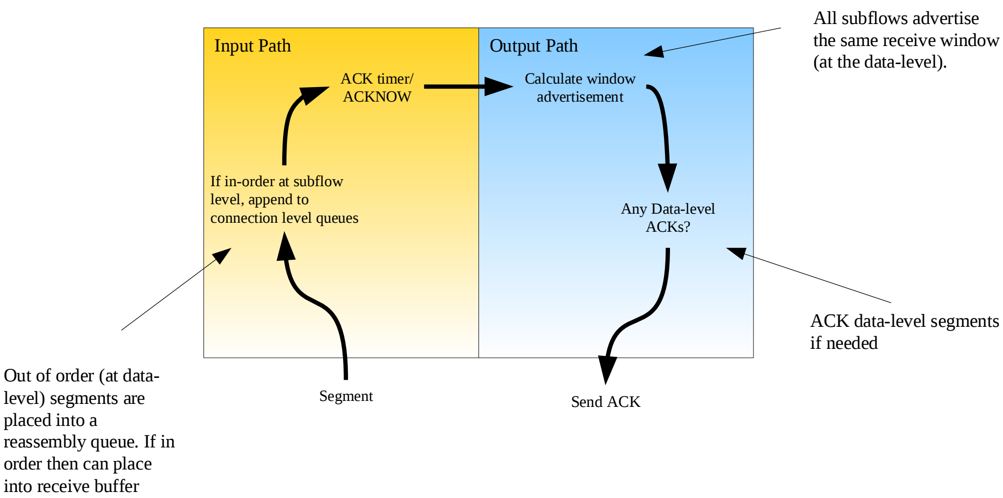
- Logical Components



Simplified Sender Path



Simplified Receiver Path



Research

- Congestion Control
  - Using a mixture of loss-based and delay-based cc algorithms across subflows on the same MPTCP Connection.
- Statistics used to make scheduling decisions
  - cwnd, RTT, other metrics.
- Policies for making scheduling decisions
  - Always evenly distribute, backup only, path costs. “Smart” segment choice.
- Opening new subflows
  - When should an additional subflow be initiated? (Perhaps not at all for short-lived flows, but when for long-lived flows?).

Links and Related Info

- MultiPath TCP – Linux Kernel implementation: http://mptcp.info.ucl.ac.be/
  - MPTCP Papers, presentations (that are better than mine) and source code
- Internet Drafts/RFCs
  - TCP Extensions for Multipath Operation with Multiple Addresses (draft)
  - MPTCP Application Interface Considerations (draft)
  - Coupled Congestion Control for Multipath Transport Protocols (RFC 6356)
- Papers
  - How hard can it be? Designing and Implementing a Deployable Multipath TCP: http://inl.info.ucl.ac.be/system/files/nsdi12-final125.pdf
  - Design, implementation and evaluation of congestion control for multipath TCP: http://www.cs.ucl.ac.uk/staff/c.raiciu/files/mptcp-nsdi.pdf

Congestion Control

```
                                  alpha * bytes_acked * MSS_i   bytes_acked * MSS_i
For ACK, increase cwnd_i by min (----------------------- , --------------------)
                                        cwnd_total                 cwnd_i
```

```
                  Max (cwnd_i/rtt_i^2)
alpha = (-----------------------)
         (sum(cwnd_i/rtt_i))^2
```

```
cwnd_i       = cwnd of subflow i
cwnd_total   = total cwnd of all subflows in connection
MSS_i        = MSS of subflow i
rtt_i        = smoothed RTT estimate for subflow i
alpha        = "agressiveness" of MPTCP connection
```