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

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Name</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>MP_CAPABLE</td>
<td>Multipath Capable</td>
<td>0x0</td>
</tr>
<tr>
<td>MP_JOIN</td>
<td>Join Connection</td>
<td>0x1</td>
</tr>
<tr>
<td>DSS</td>
<td>Data Sequence Signal (Data Ack and Data Sequence Mapping)</td>
<td>0x2</td>
</tr>
<tr>
<td>ADD_ADDR</td>
<td>Add Address</td>
<td>0x3</td>
</tr>
<tr>
<td>REMOVE_ADDR</td>
<td>Remove Address</td>
<td>0x4</td>
</tr>
<tr>
<td>MP_PRIO</td>
<td>Change Subflow Priority</td>
<td>0x5</td>
</tr>
<tr>
<td>MP_FAIL</td>
<td>Fallback</td>
<td>0x6</td>
</tr>
<tr>
<td>MP_FASTCLOSE</td>
<td>Fast Close</td>
<td>0x7</td>
</tr>
</tbody>
</table>
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

```
Host A          Host B
SYN + MP_CAPABLE  SYN/ACK + MP_CAPABLE
ACK + MP_CAPABLE  ACK
```

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

Sender

Un-ACK'd list

Sender

Un-ACK'd list

Receiver

Reassembly List

Receive Buffer

Sender

Un-ACK'd list

Receiver

Reassembly List

Receive Buffer
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. Baric, M. Handly, D. Wischik “Coupled Congestion Control for Multipath Transport Protocols, RFC 6356”]

- 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.

*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

MPTCP Session Control

Packet Scheduler

Congestion Controller

Path Manager

Table of paths available to each MPTCP connection

Decide which subflow the next segment is sent on

Calculate congestion windows for subflows
### Simplified Sender Path

- **Input Path**
  - Send buffer len > 0?
  - Update CC
- **Output Path**
  - Get segment length
  - Copy segment from send buffer
  - Send segment

- Congestion Controller updated
- Packet Scheduler decides if this subflow should send the next segment
- All subflows copy segments from a shared send buffer

### Simplified Receiver Path

- **Input Path**
  - ACK timer/ACKNOW
  - If in-order at subflow level, append to connection level queues
- **Output Path**
  - Calculate window advertisement
  - Any Data-level ACKs?
  - Send ACK

- Out of order (at data-level) segments are placed into a reassembly queue. If in order then can place into receive buffer
- All subflows advertise the same receive window (at the data-level)
- ACK data-level segments if needed
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/](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**
alpha \times \text{bytes\_acked} \times \text{MSS\_i} \quad \text{bytes\_acked} \times \text{MSS\_i} \\
\text{For ACK, increase cwnd\_i by Min} \left(\frac{\text{cwnd\_total}}{\text{cwnd\_i}}, \frac{\text{cwnd\_total}}{\text{cwnd\_i}}\right) \\
\text{Max}\left(\frac{\text{cwnd\_i}}{\text{rtt\_i}^2}\right) \\
\alpha = \left(\frac{\text{sum(cwnd\_i/rtt\_i)}}{\text{sum(cwnd\_i/rtt\_i)}}\right)^2 \\
\text{cwnd\_i} = \text{cwnd of subflow i} \\
\text{cwnd\_total} = \text{total cwnd of all subflows in connection} \\
\text{MSS\_i} = \text{MSS of subflow i} \\
\text{rtt\_i} = \text{smoothed RTT estimate for subflow i} \\
\alpha = \text{“aggressiveness” of MPTCP connection}