How Hard Can It Be?
Designing and Implementing a Deployable Multipath TCP

Costin Raiciu
University Politehnica of Bucharest

Joint work with: Christoph Paasch, Sebastien Barre, Alan Ford, Fabien Duchene, Michio Honda, Olivier Bonaventure, Mark Handley

Thanks to trilogy, CHANGE, Google, Cisco
Networks are becoming multipath

Mobile devices have multiple wireless connections
Networks are becoming multipath

Datacenters have redundant topologies
Networks are becoming multipath

Datacenters have redundant topologies
Networks are becoming multipath

Servers are multi-homed
How do we use these networks?

TCP.

Used by most applications, offers byte-oriented reliable delivery, adjusts load to network conditions.
TCP is single path

A TCP connection

Uses a single-path in the network regardless of network topology

Is tied to the source and destination addresses of the endpoints
Mismatch between network and transport creates problems
Collisions in datacenters

[Fares et al - A Scalable, Commodity Data Center Network Architecture - Sigcomm 2008]
How hard can it be?
Designing and
Implementing a
Deployable Multipath TCP
Deployable Multipath TCP
How hard can it be?
Designing
Implementing
Deployable Multipath TCP
How hard can it be?
Designing
Implementing
**Goal:** A Deployable Multipath TCP

*We want to evolve TCP to be able to use multiple paths in the network.*

Multipath TCP must meet the following goals:

**GOAL 1:** Support *unmodified applications*

**GOAL 2:** Work over *today’s networks*

**GOAL 3:** Work whenever *TCP would work*
Our Linux kernel Multipath TCP implementation supports legacy apps and works well over:

deployed 3G and Wifi networks, existing datacenters and the Internet at large.
Deployable Multipath TCP

How hard can it be?

Designing
Implementing
It can be pretty hard.
It can be pretty hard.

Mark Handley *suggested* we start working on designing MPTCP in spring 2007.
It can be pretty hard.

Mark Handley *suggested* we start working on designing MPTCP in spring 2007

Five years later, here we are – we finally nailed this!
It can be pretty hard.

Mark Handley suggested we start working on designing MPTCP in spring 2007

Five years later, here we are – we finally nailed this!

Why was it this difficult?

Internet Architecture is a living thing.
Protocol Layering

- Link layers (eg Ethernet) are local to a particular link
- Routers look at IP headers to decide how to route a packet.
- TCP provides reliability via retransmission, flow control, etc.
- Application using OS’s TCP API to do its job.
Middleboxes
### TCP Header Structure

<table>
<thead>
<tr>
<th>Field</th>
<th>Bit 0</th>
<th>Bit 15</th>
<th>Bit 16</th>
<th>Bit 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Version</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IHL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TOS</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ECN</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination Port</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Reserved</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Code bits</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Receive Window</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urgent Pointer</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Bit 0</td>
<td>Bit 15</td>
<td>Bit 16</td>
<td>Bit 31</td>
<td></td>
</tr>
<tr>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td>-------</td>
<td></td>
</tr>
<tr>
<td>Version</td>
<td>IHL</td>
<td>TOS</td>
<td>ECN</td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Fragment Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Header Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length</td>
<td>Reserved</td>
<td>Code bits</td>
<td>Receive Window</td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent Pointer</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Deployable Multipath TCP
How hard can it be?

Designing
Implementing
MPTCP Connection Management
MPTCP Connection Management

MPTCP Connection Management
MPTCP Connection Management

Enable MPTCP if SYN has MP_CAPABLE
MPTCP Connection Management

Enable MPTCP if SYN has MP_CAPABLE

ENABLED
MPTCP Connection Management

Enable MPTCP if SYN/ACK has MP_CAPABLE

Enable MPTCP if SYN has MP_CAPABLE

SYN/ACK MP_CAPABLE

ENABLED
Enable MPTCP if SYN/ACK has MP_CAPABLE

ENABLED

Enable MPTCP if SYN has MP_CAPABLE

ENABLED
MPTCP Connection Management
MPTCP Connection Management

Subflow 1

SYN
JOIN

Subflow 1
MPTCP Connection Management

Subflow 1

Syn/Ack

Join
MPTCP Connection Management
MPTCP Connection Management

Subflow 1

Subflow 2

Subflow 1

Subflow 2
That was easy!
Almost too easy...
Enable MPTCP if SYN/ACK has MP_CAPABLE

Enable MPTCP if SYN has MP_CAPABLE

ENABLED
MPTCP Connection Management

6% of access networks remove unknown options (14% on port 80)

[Honda et al. – Is It Still Possible to Extend TCP? – IMC 2011]
MPTCP Connection Management

Enable MPTCP if SYN/ACK has MP_CAPABLE
DISABLED

Enable MPTCP if SYN has MP_CAPABLE
ENABLED
MPTCP Connection Management

Enable MPTCP if SYN/ACK has MP_CAPABLE
DISABLED

Enable MPTCP if SYN has MP_CAPABLE and ACK has DATA_ACK
MPTCP Connection Management

Enable MPTCP if SYN/ACK has MP\_CAPABLE
DISABLED

Enable MPTCP if SYN has MP\_CAPABLE and ACK has DATA\_ACK
MPTCP Connection Management

Enable MPTCP if SYN/ACK has MP_CAPABLE
DISABLED

Enable MPTCP if SYN has MP_CAPABLE and ACK has DATA_ACK
DISABLED

To achieve GOAL 3:
When MPTCP operation is not possible, fallback to TCP.
Lesson

Negotiation used to be between two endpoints

In today’s Internet, negotiation is:

between two endpoints

and an unknown number of intermediaries

New protocol negotiation has to take this into account or it will fail
Sending Data
TCP Operation

1
TCP Operation
TCP Operation
TCP Operation
TCP Operation

4 ➔ 3 ➔ 2 ➔ ACK 1
TCP Operation
TCP Operation
TCP Operation
Strawman Design
Strawman Design
Strawman Design
Strawman Design
Strawman Design

[Diagram showing network traffic and ACKs]
Strawman Design

[Diagram with labeled boxes: 3, ACK 1, ACK 2, 4]
A third of access networks will “correct” or drop ACKs of unseen data
Strawman Design
Strawman Design
Ok, so what does work?

• We need a sequence space for each subflow
  – This will drive loss detection and retransmissions

• We need a data sequence number
  – This will put segments in order at the receiver

• We need a data ACK for flow control
  – Receive window is relative to Data ACK
MPTCP Data Transmission

SUBFLOW: 100
DATA:1
MPTCP Data Transmission

SUBFLOW: 100
DATA: 1

SUBFLOW: 200
DATA: 2
MPTCP Data Transmission

SUBFLOW: 101
DATA: 3

SUBFLOW: 100
DATA: 1

SUBFLOW: 200
DATA: 2
MPTCP Data Transmission

SUBFLOW: 101
DATA: 3

SUBFLOW: 100
DATA: 1

SUBFLOW: 200
DATA: 2
MPTCP Data Transmission

- **SUBFLOW: 101**
  - DATA: 3

- **SUBFLOW: 102**
  - DATA: 2

- **SUBFLOW: 100**
  - DATA: 1

- **SUBFLOW: 200**
  - DATA: 2
## TCP Packet Header

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 15</th>
<th>Bit 16</th>
<th>Bit 31</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source Port</td>
<td>Destination Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence Number</td>
<td>Acknowledgment Number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length</td>
<td>Reserved</td>
<td>Code bits</td>
<td>Receive Window</td>
</tr>
<tr>
<td></td>
<td>Checksum</td>
<td></td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- **Header**
  - **Length**
  - **Reserved**
  - **Code bits**
  - **Checksum**
  - **Urgent Pointer**
  - **Options**
  - **Data**

- **TCP Header Fields**
  - **Source Port**
  - **Destination Port**
  - **Sequence Number**
  - **Acknowledgment Number**
  - **Receive Window**

- **Header Length:** 20 Bytes
  - 0 - 40 Bytes
## MPTCP Packet Header

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 15</th>
<th>Bit 16</th>
<th>Bit 31</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subflow</strong> Source Port</td>
<td><strong>Subflow</strong> Destination Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subflow</strong> Sequence Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subflow</strong> Acknowledgment Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length</td>
<td>Reserved</td>
<td>Code bits</td>
<td>Receive Window</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
<td>Urgent Pointer</td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- Header Length: 20 Bytes
- Data: 0 - 40 Bytes
MPTCP Packet Header

- **Subflow** Source Port
- **Subflow** Destination Port
- **Subflow** Sequence Number
- **Subflow** Acknowledgment Number

**Header Length**
- Reserved
- Code bits

**Connection** Receive Window

**Checksum**

**Urgent Pointer**

**Options**

**Data**

- UMP
- Bit 0 - Bit 15
- Bit 16 - Bit 31

20 Bytes

0 - 40 Bytes

relative to Data ACK
## MPTCP Packet Header

<table>
<thead>
<tr>
<th>Bit 0</th>
<th>Bit 15</th>
<th>Bit 16</th>
<th>Bit 31</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Subflow</strong> Source Port</td>
<td><strong>Subflow</strong> Destination Port</td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subflow</strong> Sequence Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>Subflow</strong> Acknowledgment Number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Header Length</td>
<td>Reserved</td>
<td>Code bits</td>
<td>Connection</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data sequence number ?</td>
<td>Options</td>
<td>Data ACK ?</td>
<td></td>
</tr>
<tr>
<td>Data sequence number ?</td>
<td>Data</td>
<td>Data ACK ?</td>
<td></td>
</tr>
</tbody>
</table>

- **Checksum**: 20 Bytes
- **Data sequence number**: 0 - 40 Bytes
- **Data ACK**: 0 - 40 Bytes
Sending Data ACKs in the payload sucks
Sending Data ACKs in the payload leads to **deadlocks**
Client

Read Request 1
Sends Response 1

Read Request 2
Sends Response 2

Read Request 3
Sends Response 1

Server

Response 3

Response 2

Response 1

Read Request 1, Sends Response 1

Read Request 2, Sends Response 2

Read Request 2, Sends Response 2
Deadlock

Client

Client blocked by server receive window

Even though the client app has read the data, Data Ack still cannot be sent

Server

Read Request 1, Response 1

Data waiting in receive buffer (Request 2 till finished)

Server blocked on client receive window (receive window will only open when Data Ack received)

Request 1

Request 2

First part of request 3

First part of Response 1
Design space for feasible solutions is quite narrow

There are not too many things that could have been done differently

Read paper for:

- Flow control
- Dealing with content-changing middleboxes
- Dealing with TSO/LRO
- Connection teardown
- Fast receive code
- Middlebox tests
- Evaluation
Deployable Multipath TCP
How hard can it be?
Designing
Implementing
MPTCP over WiFi/3G

8Mbps, 20ms

2Mbps, 150ms
MPTCP over WiFi/3G

Average Goodput in Mbps vs. Rcv/Snd-Buffer size in MB.

- TCP over WiFi
- TCP over 3G
- Regular MPTCP
MPTCP over WiFi/3G

The graph shows the performance of MPTCP over WiFi/3G. The red line represents the regular MPTCP, while the blue lines represent TCP over WiFi and TCP over 3G, respectively. The graph indicates that MPTCP increases throughput compared to single-path TCP over WiFi and 3G.
What happened here?
MPTCP over WiFi/3G
MPTCP over WiFi/3G
MPTCP over WiFi/3G
MPTCP over WiFi/3G
MPTCP over WiFi/3G

Wifi path blocked by the Receive Window
MPTCP over WiFi/3G

REINJECT SEGMENT ON WIFI
MPTCP over WiFi/3G

REINJECT SEGMENT ON WiFi

HALVE CONGESTION WINDOW OF 3G SUBFLOW

Receive Window

1 3 2
MPTCP over WiFi/3G

Receive Window

1 4 3 2 1
MPTCP over WiFi/3G

Average Goodput in Mbps

Rcv/Snd-Buffer size in MB

- TCP over WiFi
- TCP over 3G
- regular MPTCP
- MPTCP optimized
Demo
Conclusions

• Designing a Multipath TCP isn’t difficult.
• Designing a deployable Multipath TCP is much harder.
  – Need to understand the evolving and undocumented Internet architecture.
  – Need defensive mechanisms to fall back to TCP behaviour when all else fails.
• Most extensions to TCP now face the same hurdles.
Conclusions (2)

• Designing a performant MPTCP needs care.
  – Especially need careful management of buffering to avoid unwanted interactions between subflows.
MPTCP allows standard applications to reap the benefits of multipath networks

- It is deployable today
- Try out the code – http://mptcp.info.ucl.ac.be/