Decoupling TCP from IP with Multipath TCP

Olivier Bonaventure
http://inl.info.ucl.ac.be
http://perso.uclouvain.be/olivier.bonaventure

Thanks to Sébastien Barré, Christoph Paasch, Grégory Detal, Mark Handley, Costin Raiciu, Alan Ford, Micchio Honda, Fabien Duchene and many others

April 2016
Agenda

The motivations for Multipath TCP

- The changing Internet
- The Multipath TCP Protocol
- Multipath TCP use cases
The origins of TCP

The Unix pipe model

```
Last login: Tue Nov 13 10:07:47 on ttys006
You have new mail.
mbpobo:~ obo$ echo "1234 abbsbbbs" | wc -c
  14
mbpobo:~ obo$  
```
The TCP bytestream model

Client

ABCDEF...111232

0988989 ... XYZZ

Server

IP: 1.2.3.4

IP: 4.5.6.7
Endhosts have evolved

Mobile devices have multiple wireless interfaces
User expectations
What technology provides 3G celltower

IP 1.2.3.4
What technology provides

3G celltower

IP 1.2.3.4

IP 5.6.7.8
What technology provides

When IP addresses change TCP connections have to be re-established!
Equal Cost Multipath

Packet arrival:
\[ \text{Hash}(\text{IP}_{\text{src}}, \text{IP}_{\text{dst}}, \text{Prot}, \text{Port}_{\text{src}}, \text{Port}_{\text{dst}}) \mod \#oif \]

Packets from one TCP connection follow same path

Different connections follow different paths

How prevalent is ECMP?

- Analysis of ISP network topologies

G. Detal, Ch. Paasch, S. van der Linden, P. Mérindol, G. Avoine, O. Bonaventure, Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, to appear in Computer Networks
ECMP in datacenters

G. Detal, Ch. Paasch, S. van der Linden, P. Mérindol, G. Avoine, O. Bonaventure, Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, Computer Networks, April 2013
Agenda

• The motivations for Multipath TCP

The changing Internet

• The Multipath TCP Protocol

• Multipath TCP use cases
The Internet architecture that we explain to our students

A typical "academic" network
The end-to-end principle
In reality

– almost as many middleboxes as routers
– various types of middleboxes are deployed

A middlebox zoo

Web Security Appliance

VPN Concentrator

SSL Terminator

ACE XML Gateway

PIX Firewall Right and Left

Cisco IOS Firewall

IP Telephony Router

Streamer

Voice Gateway

Content Engine

NAT

http://www.cisco.com/web/about/ac50/ac47/2.html
How to model those middleboxes?

• In the official architecture, they do not exist
• In reality...
TCP segments processed by a router

<table>
<thead>
<tr>
<th>Field</th>
<th>IP</th>
<th>TCP</th>
</tr>
</thead>
<tbody>
<tr>
<td>Source port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Flags</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Fragment Offset</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Protocol</td>
<td></td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Total length</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Identification</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Ver | IHL | ToS | Total length
---|-----|-----|----------------|
Identification | Flags | Frag. Offset
TTL | Protocol | Checksum
Source IP address
Destination IP address
Source port | Destination port
Sequence number
ACKnowledgment number
THL | Reserved | Flags | Window
Checksum | Urgent pointer
Options
Payload

Ver | IHL | ToS | Total length
---|-----|-----|----------------|
Identification | Flags | Frag. Offset
TTL | Protocol | Checksum
Source IP address
Destination IP address
Source port | Destination port
Sequence number
ACKnowledgment number
THL | Reserved | Flags | Window
Checksum | Urgent pointer
Options
Payload
TCP segments processed by a NAT

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>ToS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Frag. Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>ToS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td></td>
<td>Flags Frag. Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Options</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
TCP segments processed by a NAT (2)

• active mode ftp behind a NAT

220 ProFTPD 1.3.3d Server (BELNET FTPD Server) [193.190.67.15]
ftp_login: user `null>' pass `null>' host `ftp.belnet.be'
Name (ftp.belnet.be:obo): anonymous
 ---> USER anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
 ---> PASS XXXX
 ---> PORT 192,168,0,7,195,120
200 PORT command successful
 ---> LIST
150 Opening ASCII mode data connection for file list
lrw-r--r-- 1 ftp ftp 6 Jun 1 2011 pub -> mirror
226 Transfer complete
TCP segments processed by an ALG running on a NAT

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>ToS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Frag. Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Options

Payload

---

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>ToS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Frag. Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Options

Payload
How transparent is the Internet?

- 25th September 2010 to 30th April 2011
- 142 access networks
- 24 countries
- Sent specific TCP segments from client to a server in Japan

How to extend TCP?

RFC1323

• Large window extension
  – supports >64KBytes windows by shifting window field in TCP segment header

Snd.Wind.Scale=10
Rcv.Wind.Scale= 4
SYN+ACK WScale=4
SYN+WScale=10
SYN+WScale=10
SYN+WScale=4
SYN+WScale=10
Snd.Wind.Scale=4
Rcv.Wind.Scale=10
How to extend TCP? RFC1323

• What happens with middleboxes?

SYN+WScale=10

Snd.Wind.Scale= 0
Rcv.Wind.Scale= 0

SYN+ACK

SYN+ACK

SYN

SYN

Router 1

Snd.Wind.Scale= 0
Rcv.Wind.Scale= 0
How to extend TCP?
RFC1323

- What happens with middleboxes?

```
SYN+WScale = 10
SYN+ACK
WScale = 4
Rcv.Wind.Scale = 10
Snd.Wind.Scale = 0
Rcv.Wind.Scale = 0
```

```
SYN+WScale = 10
SYN+ACK
WScale = 4
Rcv.Wind.Scale = 10
Snd.Wind.Scale = 4
Rcv.Wind.Scale = 10
```
End-to-end transparency today

Middleboxes don't change the Protocol field, but many discard packets with an unknown Protocol field.
Agenda

• The motivations for Multipath TCP

• The changing Internet

The Multipath TCP Protocol

• Multipath TCP use cases
Design objectives

• Multipath TCP is an *evolution* of TCP

• Design objectives
  – Support unmodified applications
  – Work over today’s networks (IPv4 and IPv6)
  – Works in all networks where regular TCP works
TCP Connection establishment

• Three-way handshake

SYN, seq=1234, Options

SYN+ACK, ack=1235, seq=5678, Options

ACK, seq=1235, ack=5679
Data transfer

seq=1234,"abcd"

ACK, ack=1238, win=4

seq=1238,"efgh"

ACK, ack=1242, win=0
Connection release

seq=1234,"abcd"

RST
Connection release

seq=1234,"abcd"
FIN, seq=1238
ACK, ack=1239
seq=345,"ijkl"
FIN, seq=349
FIN, ack=350
## Identification of a TCP connection

All TCP segments contain the four tuple:

- $IP_{source}$
- $IP_{dest}$
- $Port_{source}$
- $Port_{dest}$

### Four tuple

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ver</td>
<td>Version</td>
</tr>
<tr>
<td>IHL</td>
<td>Internet Header Length</td>
</tr>
<tr>
<td>ToS</td>
<td>Type of Service</td>
</tr>
<tr>
<td>Total length</td>
<td>Total length of the header</td>
</tr>
<tr>
<td>Identification</td>
<td>Identification of the packet</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>Frag. Offset</td>
<td>Fragment offset</td>
</tr>
<tr>
<td>TTL</td>
<td>Time to Live</td>
</tr>
<tr>
<td>Protocol</td>
<td>Protocol</td>
</tr>
<tr>
<td>Checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>Source IP address</td>
<td>Source IP address</td>
</tr>
<tr>
<td>Destination IP address</td>
<td>Destination IP address</td>
</tr>
<tr>
<td>Source port</td>
<td>Source port</td>
</tr>
<tr>
<td>Destination port</td>
<td>Destination port</td>
</tr>
<tr>
<td>Sequence number</td>
<td>Sequence number</td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td>Acknowledgment number</td>
</tr>
<tr>
<td>THL</td>
<td>Header Type Length</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved</td>
</tr>
<tr>
<td>Flags</td>
<td>Flags</td>
</tr>
<tr>
<td>Window</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Checksum</td>
</tr>
<tr>
<td>Urgent pointer</td>
<td>Urgent pointer</td>
</tr>
</tbody>
</table>

**Options**

- Payload
The *new* bytestream model

Client

ABCDEF...111232

0988989 ... XYZZ

Server

IP:1.2.3.4

IP:2.3.4.5

IP:4.5.6.7

IP:6.7.8.9

D C B A
The Multipath TCP protocol

Control plane

- How to manage a Multipath TCP connection that uses several paths?

- Data plane
  - How to transport data?

- Congestion control
  - How to control congestion over multiple paths?
A naïve Multipath TCP

seq=123, "abc"

seq=126, "def"
A naïve Multipath TCP
In today's Internet?

SYN+Option

SYN+ACK+Option

ACK

seq=123, "abc"

There is no corresponding TCP connection

seq=126, "def"
Design decision

– A Multipath TCP connection is composed of one or more regular TCP subflows that are combined

• Each host maintains state that glues the TCP subflows that compose a Multipath TCP connection together

• Each TCP subflow is sent over a single path and appears like a regular TCP connection along this path
Multipath TCP and the architecture

A *regular* TCP connection

- What is a *regular* TCP connection?
  - It starts with a three-way handshake
    - SYN segments may contain special options
  - All data segments are sent in sequence
    - There is no gap in the sequence numbers
  - It is terminated by using FIN or RST
Multipath TCP

- SYN+Option
- SYN+ACK+Option
- ACK
- SYN+OtherOption
- SYN+ACK+OtherOption
- ACK
How to combine two TCP subflows?

- SYN+Option
- SYN+ACK+Option
- ACK
- SYN+OtherOption
- SYN+ACK+OtherOption
- ACK

How to link with blue subflow?
How to link TCP subflows?

SYN, Port$_{src}=1234$, Port$_{dst}=80$ + Option

SYN+ACK[...]

ACK

A NAT could change addresses and port numbers

SYN, Port$_{src}=1235$, Port$_{dst}=80$ + Option[link Port$_{src}=1234$, Port$_{dst}=80$]
How to link TCP subflows?

1. SYN, Port_{src} = 1234, Port_{dst} = 80
   + Option [Token = 5678]
   SYN+ACK+Option [Token = 6543]
   ACK

2. MyToken = 5678
   YourToken = 6543

3. SYN, Port_{src} = 1235, Port_{dst} = 80
   + Option [Token = 6543]

MyToken = 6543
YourToken = 5678
TCP subflows

• Which subflows can be associated to a Multipath TCP connection?

– At least one of the elements of the four-tuple needs to differ between two subflows
  • Local IP address
  • Remote IP address
  • Local port
  • Remote port
Subflow agility

• Multipath TCP supports
  – addition of subflows
  – removal of subflows
The Multipath TCP protocol

• Control plane
  – How to manage a Multipath TCP connection that uses several paths?

Data plane
  – How to transport data?

• Congestion control
  – How to control congestion over multiple paths?
How to transfer data?

seq=123,"a"
ack=124

seq=125,"c"
ack=126

seq=124,"b"
ack=125

seq=126,"d"
ack=127
How to transfer data in today's Internet?

seq=123,"a"
ack=124

seq=125,"c"
ack=126

seq=124,"b"
ack=125

Gap in sequence numbering space
Some DPI will not allow this!
Multipath TCP Data transfer

- Two levels of sequence numbers
Multipath TCP
Data transfer

Dseq=0, seq=123, "a"
DAck=1, ack=124
DSeq=2, seq=124, "c"
DAck=3, ack=125
DSeq=1, seq=456, "b"
DAck=2, ack=457
Multipath TCP
How to deal with losses?

• Data losses over one TCP subflow
  – Fast retransmit and timeout as in regular TCP

  \[
  \text{Dseq}=0, \text{seq}=123, "a"
  \]

  \[
  \text{DAck}=1, \text{ack}=124
  \]

  \[
  \text{Dseq}=0, \text{seq}=123, "a"
  \]

  \[
  \text{DAck}=1, \text{ack}=124
  \]
Multipath TCP

- What happens when a TCP subflow fails?

Dseq=0, seq=123, "a"

DSeq=1, seq=456, "b"

DAck=0, ack=457

Dseq=0, seq=457, "a"

DAck=2, ack=458
Retransmission heuristics

- Heuristics used by current Linux implementation
  - Fast retransmit is performed on the same subflow as the original transmission
  - Upon timeout expiration, reevaluate whether the segment could be retransmitted over another subflow
  - Upon loss of a subflow, all the unacknowledged data are retransmitted on other subflows
Flow control

• How should the window-based flow control be performed?
  – Independant windows on each TCP subflow
  – A single window that is shared among all TCP subflows
Independant windows

\[ Dseq = 0, seq = 123, "a" \]

\[ DAck = 1, ack = 124, win = 0 \]

\[ DSeq = 1, seq = 456, "b" \]

\[ DAck = 2, ack = 457, win = 100 \]

\[ Dseq = 2, seq = 457, "c" \]

\[ DAck = 3, ack = 458, win = 100 \]
Independent windows possible problem

- Dseq=0, seq=123, "a"
- DSeq=1, seq=456, "b"
- DAck=2, ack=457, win=0

• Impossible to retransmit, window is already full on green subflow
A single window shared by all subflows

Dseq=0, seq=123, "a"
DSeq=1, seq=456, "b"
Dseq=2, seq=457, "c"
DAck=1, ack=124, win=10
DAck=2, ack=457, win=10
DAck=3, ack=458, win=10
A single window shared by all subflows

Impact of middleboxes

Dseq=0, seq=123, "a"

DAck=1, ack=124, win=100

DSeq=1, seq=456, "b"

DAck=2, ack=457, win=100

DAck=2, ack=457, **win=5**
Multipath TCP Windows

- Multipath TCP maintains one window per Multipath TCP connection
  - Window is relative to the last acked data (Data Ack)
  - Window is shared among all subflows
    - It's up to the implementation to decide how the window is shared
  - Window is transmitted inside the window field of the regular TCP header
  - If middleboxes change window field,
    - use largest window received at MPTCP-level
    - use received window over each subflow to cope with the flow control imposed by the middlebox
Multipath TCP buffers

Scheduler

Transmit queues, process only regular TCP header

send(…)

socket

Multipath TCP

recv(…)

TCP1

TCP2

Reorder queue, processes only TCP header

MPTCP-level, resequencing possible
Sending Multipath TCP information

• How to exchange the Multipath TCP specific information between two hosts?

• Option 1
  – Use TLVs to encode data and control information inside payload of subflows

• Option 2
  – Use TCP options to encode all Multipath TCP information

Multipath TCP with only options

- **Advantages**
  - Normal way of extending TCP
  - Should be able to go through middleboxes or fallback

- **Drawbacks**
  - Limited size of the TCP options, notably inside SYN
  - What happens when middleboxes drop TCP options in data segments
Multipath TCP using TLV

• Advantages
  – Multipath TCP could start as regular TCP and move to Multipath only when needed
  – Could be implemented as a library in userspace
  – TLVs can be easily extended

• Drawbacks
  – TCP segments contain TLVs including the data and not only the data
    • problem for middleboxes, DPI, ..
  – Middleboxes become more difficult

Is it safe to use TCP options?

- Known option (TS) in Data segments

Is it safe to use TCP options?

- Unknown option in Data segments

Multipath TCP options

• TCP option format

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Option-specific data</th>
</tr>
</thead>
</table>

• Initial design
  – One option kind for each purpose (e.g. Data Sequence number)

• Final design
  – A single variable-length Multipath TCP option
Multipath TCP option

• A single option type
  – to minimise the risk of having one option accepted by middleboxes in SYN segments and rejected in segments carrying data

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Subtype</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>Subtype specific data (variable length)</td>
</tr>
</tbody>
</table>

Data sequence numbers and TCP segments

• How to transport Data sequence numbers?
  – Same solution as for TCP
    • Data sequence number in TCP option is the Data sequence number of the first byte of the segment

<table>
<thead>
<tr>
<th>Source port</th>
<th>Destination port</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sequence number</strong></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
</tr>
<tr>
<td>THL</td>
<td>Reserved</td>
</tr>
<tr>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td><strong>Datasequence number</strong></td>
<td></td>
</tr>
<tr>
<td>Payload</td>
<td></td>
</tr>
</tbody>
</table>
Multipath TCP
Data transfer

Dseq=0, seq=123, "a"

Dseq=1, seq=456, "b"

Dseq=2, seq=124, "c"

DAck=1, ack=124

DAck=3, ack=125

DAck=2, ack=457
Which middleboxes change TCP sequence numbers?

- Some firewalls change TCP sequence numbers in SYN segments to ensure randomness
  – fix for old windows 95 bug

- Transparent proxies terminate TCP connections
Middlebox interference

• Data segments

Data, seq=12, "ab"

Data, seq=14, "cd"

Data, seq=12, "abcd"

Such a middlebox could also be the network adapter of the server that uses LRO to improve performance.
Segment coalescing

Data sequence numbers and middleboxes

- seq=123, Dseq=0, "a"
- seq=124, DSeq=2, "c"
- seq=456, DSeq=1, "b"
- seq=123, DSeq=2, "ac"
- seq=123, DSeq=0, "ac"

Copies one option in coalesced segment.
Data sequence numbers and middleboxes

seq=123, Dseq=0, "ab"

DSeq=0, seq=123, "a"
DSeq=0, seq=124, "b"

Middlebox only understands regular TCP
A "middlebox" that both splits and coalesces TCP segments
Data sequence numbers and middleboxes

• How to avoid desynchronisation between the bytestream and data sequence numbers?

• Solution
  – Multipath TCP option carries mapping between Data sequence numbers and (difference between initial and current) subflow sequence numbers
    • mapping covers a part of the bytestream (length)
Multipath TCP
Data transfer

seq=123, DSS[0->123, len=1], "a"

DAck=1, ack=124

seq=124, DSS[2->124, len=1], "c"

DAck=3, ack=125

seq=456, DSS[1->456, len=1], "b"

DAck=2, ack=457
Data sequence numbers and middleboxes

seq=123, DSS[0->123, len=1], "a"

seq=124, DSS[2->124, len=1], "c"

DAck=2, ack=125

seq=125, DSS[2->125, len=1], "c"

seq=456, DSS[1->456, len=1], "b"

DSeq=0, ack=457
Data sequence numbers and middleboxes

seq=123, DSS[0->123, len=1], "a"

seq=124, DSS[2->124, len=1], "c"

DAck=0, ack=125

seq=125, DSS[0->125, len=1], "a"

DAck=3, ack=126

seq=456, DSS[1->456, len=1], "b"
Multipath TCP and middleboxes

• With the DSS mapping, Multipath TCP can cope with middleboxes that
  – combine segments
  – split segments

• Are they the most annoying middleboxes for Multipath TCP?
  – Unfortunately not
TCP sequence number and middleboxes

The worst middlebox

seq=123, DSS[1->123, len=2], "ab"

DAck=3, ack=125

seq=125, DSS[3->125, len=2], "cd"

seq=128, DSS[3->125, len=2], "cd"

seq=123, DSS[1->123, len=2], "aXXXb"

DAck=3, ack=128

• Is this an academic exercise or reality?
The worst middlebox

• Is unfortunately very old...
  – Any ALG for a NAT

220 ProFTPD 1.3.3d Server (BELNET FTPD Server) [193.190.67.15]
ftp_login: user `null>' pass `null>' host `ftp.belnet.be'
Name (ftp.belnet.be:obo): anonymous
--> USER anonymous
331 Anonymous login ok, send your complete email address as your password
Password:
--> PASS XXXX
--> PORT 192,168,0,7,195,120
200 PORT command successful
--> LIST
150 Opening ASCII mode data connection for file list
lrw-r--r-- 1 ftp ftp 6 Jun 1 2011 pub -> mirror
226 Transfer complete
Coping with the worst middlebox

• What should Multipath TCP do in the presence of such a worst middlebox?
  – Do nothing and ignore the middlebox
    • but then the bytestream and the application would be broken and this problem will be difficult to debug by network administrators
  – Detect the presence of the middlebox
    • and fallback to regular TCP (i.e. use a single path and nothing fancy)

Multipath TCP **MUST** work in all networks where regular TCP works.
Detecting the worst middlebox?

- How can Multipath TCP detect a middlebox that modifies the bytestream and inserts/removes bytes?

  - Various solutions were explored

  - In the end, Multipath TCP chose to include its own checksum to detect insertion/deletion of bytes
The worst middlebox

seq=123, DSS[1->123, len=2, V], "ab"

seq=123, DSS[1->123, len=2, Inv], "aXXXb"

RST, last DSeq=0

RST, last DSeq=0

seq=456, DSS[1->456, len=2, V], "ab"

DAck=3, ack=458
Multipath TCP
Data sequence numbers

• What should be the length of the data sequence numbers?

  – 32 bits
    • compact and compatible with TCP
    • wrap around problem at highspeed requires PAWS

  – 64 bits
    • wrap around is not an issue for most transfers today
    • takes more space inside each segment
Multipath TCP
Data sequence numbers

• Data sequence numbers and Data acknowledgements

  – Maintained inside implementation as 64 bits field

  – Implementations can, as an optimisation, only transmit the lower 32 bits of the data sequence and acknowledgements
**Data Sequence Signal option**

<table>
<thead>
<tr>
<th>Field</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Kind</td>
<td>Type of data packet</td>
</tr>
<tr>
<td>Length</td>
<td>Length of the payload, in octets</td>
</tr>
<tr>
<td>Subtype</td>
<td>Subtype of the data packet</td>
</tr>
<tr>
<td>Reserved</td>
<td>Reserved bits for future use</td>
</tr>
<tr>
<td>F</td>
<td>Flag bit, indicates presence of Data ACK</td>
</tr>
<tr>
<td>m</td>
<td>Mapping bit, indicates presence of DSN</td>
</tr>
<tr>
<td>M</td>
<td>Mapping bit, indicates presence of mapping</td>
</tr>
<tr>
<td>a</td>
<td>Flag bit, indicates Data ACK is 8 octets</td>
</tr>
<tr>
<td>A</td>
<td>Flag bit, indicates Data Ack present</td>
</tr>
<tr>
<td>Data ACK</td>
<td>Data ACK field, 4 or 8 octets, depending on flags</td>
</tr>
<tr>
<td>Data Sequence Number</td>
<td>Data Sequence Number field, 4 or 8 octets, depending on flags</td>
</tr>
<tr>
<td>Subflow Sequence Number</td>
<td>Subflow Sequence Number field, 4 octets</td>
</tr>
<tr>
<td>Data-level Length</td>
<td>Data-level Length field, 2 octets</td>
</tr>
<tr>
<td>Checksum</td>
<td>Checksum field, 2 octets</td>
</tr>
</tbody>
</table>

- **Cumulative Data ack**: Indicates if Data ACK is present.
- **Length of mapping**: Can extend beyond this segment.
- **A = Data ACK present**
- **a = Data ACK is 8 octets**
- **M = mapping present**
- **m = DSN is 8**
- **Computed over data covered by entire mapping + pseudo header**: Computes the checksum over the entire mapping.
Cost of the DSN checksum

The Multipath TCP protocol

• Control plane
  – How to manage a Multipath TCP connection that uses several paths?

• Data plane
  – How to transport data?

Congestion control
  – How to control congestion over multiple paths?
TCP congestion control

• A linear rate adaptation algorithm

  • \( rate(t + 1) = \alpha_C + \beta_C rate(t) \) when the network is congested
  • \( rate(t + 1) = \alpha_N + \beta_N rate(t) \) when the network is not congested

To be fair and efficient, a linear algorithm must use additive increase and multiplicative decrease (AIMD)

```python
# Additive Increase Multiplicative Decrease
if congestion :
    rate=rate*betaC      # multiplicative decrease, betaC<1
else
    rate=rate+alphaN    # additive increase, v0>0
```
AIMD in TCP

- Congestion control mechanism
  - Each host maintains a *congestion window (cwnd)*
  - No congestion
    - Congestion avoidance (*additive increase*)
      - increase *cwnd* by one segment every round-trip-time
  - Congestion
    - TCP detects congestion by detecting losses
    - Mild congestion (fast retransmit – *multiplicative decrease*)
      - *cwnd*=*cwnd*/2 and restart congestion avoidance
    - Severe congestion (timeout)
      - *cwnd*=1, set slow-start-threshold and restart slow-start
Evolution of the congestion window

- **Slow-start**: exponential increase of cwnd
- **Congestion avoidance**: linear increase of cwnd
- **Fast retransmit**
- **Threshold**
Congestion control for Multipath TCP

• Simple approach
  – independant congestion windows
Independant congestion windows

- Problem
Coupled congestion control

• Congestion windows are coupled
  – congestion window growth cannot be faster than TCP with a single flow
  – Coupled congestion control aims at *moving traffic away from congested path*
Coupling the congestion windows

- **Principle**
  - The TCP subflows are not independent and their congestion windows must be coupled

- **EWTCP**
  - For each ACK on path r, $cwin_r = cwin_r + a/cwin_r$ (in segments)
  - For each loss on path r, $cwin_r = cwin_r / 2$
  - Each subflow gets window size proportional to $a^2$
  - Same throughput as TCP if $a = \frac{1}{\sqrt{n}}$

Can we split traffic equally among all subflows?

In this scenario, EWTCP would get 3.5 Mbps on the two hops path and 5 Mbps on the one hop path, less than the optimum of 12 Mbps for each Multipath TCP connection.

Linked increases congestion control

• Algorithm
  – For each loss on path r, \( c\text{win}_r = \frac{c\text{win}_r}{2} \)
  – Additive increase

\[
c\text{win}_r = c\text{win}_r + \min \left( \frac{\text{max} \left( \frac{c\text{wnd}_i}{(rtt_i)^2} \right)}{\left( \sum_i \frac{c\text{wnd}_i}{rtt_i} \right)^2}, \frac{1}{c\text{wnd}_r} \right)
\]

Other Multipath-aware congestion control schemes


The Multipath TCP protocol

Control plane

- How to manage a Multipath TCP connection that uses several paths?

Data plane

- How to transport data?

Congestion control

- How to control congestion over multiple paths?
The Multipath TCP control plane

- Connection establishment
- Closing a Multipath TCP connection
- Address dynamics
The Multipath TCP control plane

• Connection establishment
  – Beware of middleboxes that remove TCP options
  – Limited space inside TCP option in SYN

• Closing a Multipath TCP connection
  – Decouple closing the Multipath TCP connection from closing the subflows

• Address dynamics
Security threats

• Three main security threats were considered
  – flooding attack
  – man-in-the middle attack
  – hijacking attach

Security goal:
Multipath TCP should not be worse than regular TCP

Hijacking attack
Multipath TCP
Connection establishment

• Principle

SYN, MP_CAPABLE
SYN+ACK, MP_CAPABLE
ACK, MP_CAPABLE
seq=123, DSeq=1, "abc"
Roles of the initial TCP handshake

• Check willingness to open TCP connection
  – Propose initial sequence number
  – Negotiate Maximum Segment Size

• TCP options
  – negotiate Timestamps, SACK, Window scale

• Multipath TCP
  – check that server supports Multipath TCP
  – propose Token in each direction
  – propose initial Data sequence number in each direction
  – Exchange keys to authenticate subflows
How to extend TCP?

Theory

- TCP options were invented for this purpose
  - Exemple SACK

![Diagram showing SYN+SACK, SYN+ACK, and SACK enabled]
How to extend TCP?

- What happens when there are middleboxes on the path?
TCP options

• In SYN segments


© O. Bonaventure, 2011
How to extend TCP?
The worst case

- What happens when there are middleboxes on the path?

Client and server do not agree on TCP extension!!!
Multipath TCP handshake

- SYN, MP_CAPABLE[...]
- SYN+ACK, MP_CAPABLE[...]
- ACK+MPTCPOption

Why an option in third ACK?
Multipath TCP option in third ACK

No option, disable Multipath TCP
Multipath TCP handshake
Token exchange

SYN, MP_CAPABLE[ClientToken=1234]

SYN+ACK, MP_CAPABLE[ServerToken=5678]

ACK, MP_CAPABLE[ClientToken=1234]

Useful if server wants to send SYN +ACK without keeping any state
Initial Data Sequence number

• Why do we need an initial Data Sequence number?
  – Setting IDSN to a random value improves security
  – Hosts must know IDSN to prevent data injection attacks
Initial Data Sequence number

SYN, MP_CAPABLE

SYN+ACK, MP_CAPABLE

ACK

seq=123, DSS[456->123, len=2], "ab"

First Data or not?

seq=789, DSS[458->789, len=2], "cd"
Initial Data Sequence number

• How to negotiate the IDSN?

SYN, MP_CAPABLE[DSeq=23456]

SYN+ACK, MP_CAPABLE[DSeq=67890]

ACK
How to secure Multipath TCP

• Main goal
  – Authenticate the establishment of subflows

• Principles
  – Each host announces a key during initial handshake
    • keys are exchanged in clear
  – When establishing a subflow, use HMAC + key to authenticate subflow
Putting everything inside the SYN

• How can we place inside SYN segment?

  – Initial Data Sequence Number (64 bits)

  – Token (32 bits)

  – Authentication Key (the longer the better)
Constraint on TCP options

- Total length of TCP header: max 64 bytes
  - max 40 bytes for TCP options
  - *Options* length must be multiple of 4 bytes

<table>
<thead>
<tr>
<th>Ver</th>
<th>IHL</th>
<th>ToS</th>
<th>Total length</th>
</tr>
</thead>
<tbody>
<tr>
<td>Identification</td>
<td>Flags</td>
<td>Frag. Offset</td>
<td></td>
</tr>
<tr>
<td>TTL</td>
<td>Protocol</td>
<td>Checksum</td>
<td></td>
</tr>
<tr>
<td>Source IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Destination IP address</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Source port</td>
<td>Destination port</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sequence number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Acknowledgment number</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td><strong>THL</strong></td>
<td>Reserved</td>
<td>Flags</td>
<td>Window</td>
</tr>
<tr>
<td>Checksum</td>
<td>Urgent pointer</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*Options*

Payload
Key exchange

SYN, [MyKey="keyABC"]

SYN+ACK, [MyKey="keyDEF"]

ACK[MyKey="keyABC", YourKey="keyDEF"]

MyKey="keyABC"  
YourKey="keyDEF"

MyKey="keyDEF"  
YourKey="keyABC"

SYN,[NonceA=123]

SYN+ACK[NonceB=456,  
HMAC(123 || 456,"keyDEF || keyABC")]

ACK,[HMAC(456 || 123,"keyABC || keyDEF")]

MyKey="keyABC"  
YourKey="keyDEF"

MyKey="keyDEF"  
YourKey="keyABC"
TCP options in the wild

• **MSS** option [4 bytes]
  – Used only inside SYN segments

• **Timestamp** option [10 bytes]
  – Used in potentially all segments

• **Window scale** option [3 bytes]
  – Used only inside SYN segments

• **SACK permitted** option [2 bytes]
  – Used only inside SYN segments

• **Selective Acknowledgements** [N bytes]
  – Used in data segments

Only 20 bytes left inside SYN!
The **MP_CAPABLE** option

**A:** DSN Checksum required or not

**B:** Extension

---

<table>
<thead>
<tr>
<th>Kind</th>
<th>Length</th>
<th>Subtype</th>
<th>Version</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>F</th>
<th>G</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Option Sender’s Key (64 bits)

Option Receiver’s Key (64 bits)

(if option Length == 20)

---

**Used inside SYN**

**Only in third ACK**

**crypto**
Initial Data Sequence Numbers and Tokens

• Computation of initial Data Sequence Number

\[ \text{IDSN}_A = \text{Lower}_{64}(\text{SHA}-1(\text{Key}_A)) \]
\[ \text{IDSN}_B = \text{Lower}_{64}(\text{SHA}-1(\text{Key}_B)) \]

• Computation of token

\[ \text{Token}_A = \text{Upper}_{32}(\text{SHA}-1(\text{Key}_A)) \]
\[ \text{Token}_B = \text{Upper}_{32}(\text{SHA}-1(\text{Key}_B)) \]

There is a small risk of collision, different keys same token
Cost of the Multipath TCP handshake

The Multipath TCP control plane

• Connection establishment in details

• Closing a Multipath TCP connection

• Address dynamics
Closing a Multipath TCP connection

• How to close a Multipath TCP connection?
  – By closing all subflows?
Closing a Multipath TCP connection

seq=123, DSS[1->123 ...], "ab"

seq=125, DSS[5->125,DATA_FIN ...], "end"

DAck=9, ack=128

seq=128, FIN

ack=129

seq=456, DSS[3->456], "cd"

DAck=5, ack=458
Closing a Multipath TCP connection

- FAST Close

```
SYN, [...]  SYN+ACK, [...]  ACK[...]
SYN+FAST_CLOSE  ACK[...]
RST
SYN,...]  SYN+ACK[...]
ACK[..]  RST
```
The Multipath TCP control plane

- Connection establishment in details

- Closing a Multipath TCP connection

- Address dynamics
Multipath TCP
Address dynamics

• How to learn the addresses of a host?
  
  IP=2.3.4.5
  IP=3.4.5.6
  IP6=2a00:1450:400c:c05::69

• How to deal with address changes?
  
  IP=1.2.3.4
  IP=4.5.6.7
Address dynamics

- Basic solution: multihomed server

  SYN, [...]  
  SYN+ACK, [...]  
  ACK[...]

  ADD_ADDR[3.4.5.6]  
  ADD_ADDR[2a00:1450:400c:c05::69]  
  SYN,[...]  
  SYN+ACK[...]  
  ACK[..]
Address dynamics

- Basic solution: mobile client

  - SYN, [...]
  - SYN+ACK, [...]
  - ACK[...]
  - ADD_ADDR [4.5.6.7]
  - SYN, [...]
  - SYN+ACK[...]
  - ACK[...]
  - REMOVE_ADDR[1.2.3.4]
Address dynamics in today's Internet
Address dynamics with NATs

• Solution

– Each address has one identifier
  • Subflow is established between id=0 addresses
– Each host maintains a list of <address, id> pairs of the addresses associated to an MPTCP endpoint
– MPTCP options refer to the address identifier
  • ADD_ADDR contains <address, id>
  • REMOVE_ADDR contains <id>
Address dynamics

SYN, [...] → SYN+ACK, [...] → ACK[...]

ADD_ADDR [4.5.6.7, id=1] → SYN,[id=1...] → SYN+ACK[...]

ACK[...] → REMOVE_ADDR[id=0]

IP=2.3.4.5

IP=1.2.3.4

IP=4.5.6.7
Agenda

• The motivations for Multipath TCP

• The changing Internet

• The Multipath TCP Protocol
  • Multipath TCP use cases
    – Datacenters
    – Smartphones
    – Commercial deployments
TCP on servers

• How to increase server bandwidth?

• Load balancing techniques
  – packet per packet
  – per flow load balancing
    • each TCP connection is mapped onto one interface
Increasing server bandwidth with Multipath TCP

- Load balancing with Multipath TCP
  - Congestion control efficiently uses the two links for each MPTCP connection
  - Automatic failover in case of failures
How fast can Multipath TCP go?

How fast can Multipath TCP go?

http://www.multipath-tcp.org
Datacenters evolve

• Traditional Topologies are tree-based
  – Poor performance
  – Not fault tolerant

• Shift towards multipath topologies: FatTree, BCube, VL2, Cisco, EC2

Fat Tree Topology [Fares et al., 2008; Clos, 1953]

K=4

1Gbps

K Pods with K Switches each

Racks of servers

Aggregation Switches
Fat Tree Topology [Fares et al., 2008; Clos, 1953]

K=4

Aggregation Switches

K Pods with K Switches each

Racks of servers

Collisions
TCP in FAT tree networks

Cost of collisions

How to get rid of these collisions?

- Consider TCP performance as an optimisation problem
The Multipath TCP way

Two subflows differ by their source port

ECMP balances the subflows over different paths

MPTCP better utilizes the FatTree network

![Graph showing throughput vs. rank of flow for MPTCP and TCP]


See also G. Detal, et al., Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, Computer Networks, April 2013 for extensions to ECMP for MPTCP.
How many subflows does Multipath TCP need?

Can we improve Multipath TCP?

- Two subflows may follow similar paths
Improving ECMP

- ECMP's hash
  - good load balancing
  - impossible to predict result

- CFLB
  - replaces hash with block cipher
  - hosts can select paths for Multipath TCP subflows provided they know datacenter topology

G. Detal, Ch. Paasch, S. van der Linden, P. Mérandol, G. Avoine, O. Bonaventure, Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, to appear in Computer Networks
Multipath TCP with CFLB in Fat-Tree

G. Detal, Ch. Paasch, S. van der Linden, P. Méridol, G. Avoine, O. Bonaventure, *Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks*, to appear in Computer Networks
Multipath TCP on EC2

• Amazon EC2: infrastructure as a service
  – We can borrow virtual machines by the hour
  – These run in Amazon data centers worldwide
  – We can boot our own kernel

• A few availability zones have multipath topologies
  – 2-8 paths available between hosts not on the same machine or in the same rack
  – Available via ECMP
Amazon EC2 Experiment

• 40 medium CPU instances running MPTCP
• During 12 hours, we sequentially ran all-to-all `iperf` cycling through:
  – TCP
  – MPTCP (2 and 4 subflows)
MPTCP improves performance on EC2

Agenda

• The motivations for Multipath TCP

• The changing Internet

• The Multipath TCP Protocol

• Multipath TCP use cases
  – Datacenters
  – Smartphones
  – Commercial deployments
Motivation

• One device, many IP-enabled interfaces
ssh with Multipath TCP
MPTCP over WiFi/3G

8Mbps, 20ms

2Mbps, 150ms
MPTCP over WiFi/3G

MPTCP over WiFi/3G

Multipath TCP increases throughput.
MPTCP over WiFi/3G

What happened here?
Understanding the performance issue

Window full!
No new data can be sent on WiFi path

Reinject segment on fast path
Halve congestion window on slow subflow
Usage of 3G and WiFi

• How should Multipath TCP use 3G and WiFi?
  
  – Full mode
    • Both wireless networks are used at the same time
  
  – Backup mode
    • Prefer WiFi when available, open subflows on 3G and use them as backup
  
  – Single path mode
    • Only one path is used at a time, WiFi preferred over 3G
Live streaming
Multipath TCP: Full mode

SYN...

SYN+ACK...

ACK...

SYN...

SYN+ACK,...

ACK ...

WiFi

3G/LTE
Multipath TCP: Backup mode

Dynamically change backup status of flow
Multipath TCP: Backup mode

- What happens when link fails?

SYN... SYN+ACK...

ACK... SYN,MP_JOIN[Backup...]

SYN+ACK,...

ACK ...

REM_ADDR[id=0]
• Multipath TCP supports break before make
Evaluation scenario

WiFi: Belgacom ADSL2+
(~8 Mbps, ~30 ms)

3G: Mobistar
(~2 Mbps, ~80 ms)
Recovery after failure

Recovery after failure

Agenda

• The motivations for Multipath TCP
• The changing Internet
• The Multipath TCP Protocol

• Multipath TCP use cases
  – Datacenters
  – Smartphones

IPv4/IPv6 coexistence
IPv6 is coming ...

But IPv4 and IPv6 perf. may differ

Happy eyeballs
How to get best of IPv4 and IPv6?

IPv6:::cafe

IPv6:::beef

1.2.3.4

ADD_ADDR[5.6.7.8]

ACK

SYN+ACK MP_JOIN...

ACK...

SYN+MP_JOIN...
Agenda

• The motivations for Multipath TCP

• The changing Internet

• The Multipath TCP Protocol

• Multipath TCP use cases
  – Datacenters
  – Smartphones

  Commercial deployments
Multipath TCP use cases

The beast

VRT lanceert "The beast" voor videojournalisten
Multipath TCP use cases
Low latency for Siri

- Long-lived TLS connections

“Hey Siri, what song is this?”
Through the Shazam app, Siri can tell you what song is playing around you.
Multipath TCP use cases

High bandwidth on smartphones

- Koreans want 800+ Mbps on smartphones
Faster broadband networks?
Multipath TCP use cases

Hybrid Access Networks

- DSL
- 4G/LTE
- Multipath TCP
- Regular TCP
- Hybrid Access Gateway
Conclusion

• Multipath TCP is becoming a reality
  – Due to the middleboxes, the protocol is more complex than initially expected
  – RFC has been published
  – there is running code!
  – Multipath TCP works over today's Internet!

• What's next?
  – More use cases
  – Measurements and improvements to the protocol
    • Time to revisit 20+ years of heuristics added to TCP
Try it by yourself!

http://multipath-tcp.org
References

• The Multipath TCP protocol
  – http://www.multipath-tcp.org


Implementations

• Linux
  – http://www.multipath-tcp.org

• FreeBSD

• Simulators
  – http://nrg.cs.ucl.ac.uk/mptcp/implementation.html
Middleboxes


Multipath congestion control

– Background


– Coupled congestion control


Multipath congestion control

– More


Use cases

– Datacenter


G. Detal, Ch. Paasch, S. van der Linden, P. Mérindol, G. Avoine, O. Bonaventure, Revisiting Flow-Based Load Balancing: Stateless Path Selection in Data Center Networks, Computer Networks, April 2013

– Mobile
