# Multipathed, Multiplexed, Multilateral Transport Protocols

## Decoupling transport protocols from what's below

Catherine (Kate) Pearce
Sr Security Consultant, Cisco Security Services
Feb 22, 2016

# Who?

# Who?

- Catherine (Kate) Pearce

# Who?

- Catherine (Kate) Pearce
  - @secvalve

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders
  - Research what's ahead

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders
  - Research what's ahead

- Distinguishing Features:

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders
  - Research what's ahead

- Distinguishing Features:
  - Loud, Yellow

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders
  - Research what's ahead

- Distinguishing Features:
  - Loud, Yellow
  - Or is that "Loud Yellow"?

# Who?

- Catherine (Kate) Pearce
  - @secvalve

- Sr. Security Consultant (Customer Focused)
  - Break & Report
  - Coach the builders
  - Research what's ahead

- Distinguishing Features:
  - Loud, Yellow
  - Or is that "Loud Yellow"?

ALL OPINIONS ENTIRELY MY OWN.
NO official Cisco representations of any kind

MPTCP changes

***fundamental assumptions***

about

***how TCP works****

MPTCP changes

***fundamental assumptions***

about

***how TCP works****

*Use it to break things today*

MPTCP changes

***fundamental assumptions***

about

***how TCP works****

*Use it to break things today*

*Adapt to it for tomorrow*

QUIC also changes

***fundamental assumptions***

about

***how HTTP works****

QUIC also changes

***fundamental assumptions***

about

***how HTTP works****

*Use it to break things today*

QUIC also changes

***fundamental assumptions***

about

***how HTTP works****

*Use it to break things today*

*Adapt to it for tomorrow*

# 2 Simple Examples: #1

192.168.88.165 - PuTTY

```
root@deb7min2:~# curl 192.168.88.164
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been
</body></html>
root@deb7min2:~#
```

# 2 Simple Examples: #1

192.168.88.165 - PuTTY

```
root@deb7min2:~# curl 192.168.88.164
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has been
</body></html>
root@deb7min2:~#
```

# 2 Simple Examples: #1

Snorby - Dashboard - Chro... | [Terminal - pst@pst-virtual-... | Terminal - pst@pst-virtual-... | Capturing from Pseudo-dev...

Capturing from Pseudo-device that captures on all interfaces [Wireshark 1.6.7 ]

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: tcp.port == 80 Expression... Clear Apply

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 80 | 12.278794 | 192.168.88.165 | 192.168.88.164 | TCP | 88 | 34668 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=1249 |
| 81 | 12.278839 | 192.168.88.164 | 192.168.88.165 | TCP | 88 | http > 34668 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1460 SACK_PERM=1 |
| 82 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] [ACK] Seq=1 Ack=1 Win=58432 Len=0 TSval=12490113 TSecr=1247 |
| 83 | 12.279003 | 192.168.88.165 | 192.168.88.164 | HTTP | 166 | GET / HTTP/1.1 |
| 84 | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] | [illegible] [AC 0 TSval=12474351 TSecr=124 |
| 85 | 12.280095 | 192.168.88.165 | 192.168.88.164 | TCP | 88 | 39757 > http [SY 460 SACK_PERM=1 TSval=1249 |
| 86 | 12.280111 | 192.168.88.164 | 192.168.88.165 | TCP | 92 | http > 39757 [SY Len=0 MSS=1460 SACK_PERM=1 |
| 87 | 12.280222 | 192.168.88.165 | 192.168.88.164 | TCP | 92 | 39757 > http [AC TSval=12490113 TSecr=1247 |
| 88 | 12.280338 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | [TCP Window Upda ck=1 Win=85696 Len=0 TSval |
| 89 | 12.283984 | 192.168.88.164 | 192.168.88.165 | HTTP | 548 | HTTP/1.1 200 OK |
| 90 | 12.284014 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 39757 > http [AC =0 TSval=12490114 TSecr=12 |
| 91 | 12.284045 | 192.168.88.165 | 192.168.88.164 | TCP | 88 | [TCP Dup ACK 90# k=461 Win=88576 Len=0 TSva |
| 92 | 12.284064 | 192.168.88.164 | 192.168.88.165 | TCP | 88 | [TCP Dup ACK 89# Ack=1 Win=85696 Len=0 TSva |
| 93 | 12.284073 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 39757 > ht 6 Len=0 TSval=12490114 TSe |
| 94 | 12.284077 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [F Len=0 TSval=12490114 TSec |
| 95 | 12.284086 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | [TCP Dup ACK 93# k=461 Win=88576 Len=0 TSva |
| 96 | 12.284091 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 39757 [F 6 Len=0 TSval=12474352 TSe |
| 97 | 12.284099 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 39757 > http [AC =0 TSval=12490114 TSecr=12 |
| 98 | 12.284103 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [F Len=0 TSval=12474352 TSec |
| 99 | 12.284126 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [AC 0 TSval=12490114 TSecr=124 |

Mark Packet (toggle)
Ignore Packet (toggle)
Set Time Reference (toggle)
Manually Resolve Address
Apply as Filter
Prepare a Filter
Conversation Filter
Colorize Conversation
Follow TCP Stream
Follow SSL Stream
Copy
Decode As...
Print...
Show Packet in New Window

# 2 Simple Examples: #1

Snorby - Dashboard - Chro... | [Terminal - pst@pst-virtual-... | Terminal - pst@pst-virtual-... | Capturing from Pseudo-dev... | Follow TCP Stream

Capturing from Pseudo-device that captures on all interfaces [Wireshark 1.6.

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: tcp.stream eq 2 Expression... Clear Apply

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 80 | 12.278794 | 192.168.88.165 | 192.168.88.164 | TCP | 88 | 34668 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK |
| 81 | 12.278839 | 192.168.88.164 | 192.168.88.165 | TCP | 88 | http > 34668 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MS |
| 82 | 12.278924 | 192.168.88.165 | 192.168.88.164 | TCP | 96 | 34668 > http [ACK] Seq=1 Ack=1 Win=58432 Len=0 TSval=1 |
| 83 | 12.279003 | 192.168.88.165 | 192.168.88.164 | HTTP | 166 | GET / HTTP/1.1 |
| 84 | 12.279256 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [ACK] Seq=1 Ack=79 Win=57152 Len=0 TSval= |
| 94 | 12.284077 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T |
| 98 | 12.284103 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T |
| 99 | 12.284126 | 192.168.88.165 | | | | |

Follow TCP Stream

Stream Content

```
GET / HTTP/1.1
User-Agent: curl/7.26.0
Host: 192.168.88.164
Accept: */*
```

# 2 Simple Examples: #1

Snorby - Dashboard - Chro... | [Terminal - pst@pst-virtual-... | Terminal - pst@pst-virtual-... | Capturing from Pseudo-dev... | Follow TCP Stream

Capturing from Pseudo-device that captures on all interfaces [Wireshark 1.6.

File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

Filter: tcp.stream eq 2 Expression... Clear Apply

| No. | Time | Source | Destination | Protocol | Length | Info |
|---|---|---|---|---|---|---|
| 80 | 12.278794 | 192.168.88.165 | 192.168.88.164 | TCP | 88 | 34668 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK |
| 81 | 12.278839 | 192.168.88.164 | 192.168.88.165 | TCP | 88 | http > 34668 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MS |
| 82 | 12.278924 | 192.168.88.165 | 192.168.88.164 | TCP | 96 | 34668 > http [ACK] Seq=1 Ack=1 Win=58432 Len=0 TSval=1 |
| 83 | 12.279003 | 192.168.88.165 | 192.168.88.164 | HTTP | 166 | GET / HTTP/1.1 |
| 84 | 12.279256 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [ACK] Seq=1 Ack=79 Win=57152 Len=0 TSval= |
| 94 | 12.284077 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T |
| 98 | 12.284103 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T |
| 99 | 12.284126 | 192.168.88.165 | | | | |

Follow TCP Stream

Stream Content

```
GET / HTTP/1.1
User-Agent: curl/7.26.0
Host: 192.168.88.164
Accept: */*
```

**Wait, What!?!!**

# 2 Simple Examples: #1

Snorby - Dashboard - Chro...

File Edit View Go Capture

Filter: tcp.stream eq 2

| No. | Time | Source | | | | |
|---|---|---|---|---|---|---|
| 80 | 12.278794 | 192.168.88 | | | | |
| 81 | 12.278839 | 192.168 | | | | |
| 82 | 12.278924 | 192.168.88 | | | | |
| 83 | 12.279003 | 192.168.88 | | | | |
| 84 | 12.279256 | 192.168.88 | | | | |
| 94 | 12.284077 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T |
| 98 | 12.284103 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T |
| 99 | 12.284126 | 192.168.88.165 | | | | |

192.168.88.165 - PuTTY

```
root@deb7min2:~# curl 192.168.88.164
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has b
</body></html>
root@deb7min2:~#
```

Follow TCP Stream

Stream Content

```
GET / HTTP/1.1
User-Agent: curl/7.26.0
Host: 192.168.88.164
Accept: */*
```

# 2 Simple Examples: #1

Snorby - Dashboard - Chro...

File Edit View Go Capture

Filter: tcp.stream eq 2

| No. | Time | Source | | | | |
|---|---|---|---|---|---|---|
| 80 | 12.278794 | 192.168.88 | | | | |
| 81 | 12.278839 | 192.168 | | | | |
| 82 | 12.278924 | 192.168.88 | | | | |
| 83 | 12.279003 | 192.168.88 | | | | |
| 84 | 12.279256 | 192.168.88 | | | | |
| 94 | 12.284077 | 192.168.88.165 | 192.168.88.164 | TCP | 76 | 34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T |
| 98 | 12.284103 | 192.168.88.164 | 192.168.88.165 | TCP | 76 | http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T |
| 99 | 12.284126 | 192.168.88.165 | | | | |

192.168.88.165 - PuTTY

```
root@deb7min2:~# curl 192.168.88.164
<html><body><h1>It works!</h1>
<p>This is the default web page for this server.</p>
<p>The web server software is running but no content has b
</body></html>
root@deb7min2:~#
```

Follow TCP Stream

Stream Content

```
GET / HTTP/1.1
User-Agent: curl/7.26.0
Host: 192.168.88.164
Accept: */*
```

**Wait, What!?!!**

# 2 Simple Examples: #2

```
# nc 192.168.1.25 3000
```

# 2 Simple Examples: #2

# 2 Simple Examples: #2

```
root@deb7min-LEFT:/home/username# netstat --tcp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      1 10.2.2.111:43272       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.3.3.111:33145       192.168.22.145:3000     SYN_SENT
tcp        0      1 10.1.1.111:43769       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.1.1.111:52605       192.168.22.145:3000     SYN_SENT
tcp        0      0 192.168.1.34:50818     192.168.1.25:3000       ESTABLISHED
tcp        0      1 192.168.1.34:34095     192.168.22.145:3000     SYN_SENT
tcp        0      1 10.3.3.111:36916       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.2.2.111:40284       192.168.22.145:3000     SYN_SENT
tcp6       0      0 2601:6:1700:168:2:40378 2601:6:1700:168:20:3000 ESTABLISHED
tcp6       0      0 2601:6:1700:168:2:41699 2601:6:1700:168:20:3000 ESTABLISHED
```

# 2 Simple Examples: #2

```
root@deb7min-LEFT:/home/username# netstat --tcp
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      1 10.2.2.111:43272       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.3.3.111:33145       192.168.22.145:3000     SYN_SENT
tcp        0      1 10.1.1.111:43769       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.1.1.111:52605       192.168.22.145:3000     SYN_SENT
tcp        0      0 192.168.1.34:50818     192.168.1.25:3000       ESTABLISHED
tcp        0      1 192.168.1.34:34095     192.168.22.145:3000     SYN_SENT
tcp        0      1 10.3.3.111:36916       192.168.1.25:3000       SYN_SENT
tcp        0      1 10.2.2.111:40284       192.168.22.145:3000     SYN_SENT
tcp6       0      0 2601:6:1700:168:2:40378 2601:6:1700:168:20:3000 ESTABLISHED
tcp6       0      0 2601:6:1700:168:2:41699 2601:6:1700:168:20:3000 ESTABLISHED
```

**Err?**

# *What's going on here?*

# *What's going on here?*

→ *Let's talk about upcoming transport protocols*

Introduction ✔

# Background

(TCP, Why Change it?)

MPTCP
Background Redux
QUIC
Implications
Conclusion and Takeaways

(Why NOT to change TCP)

# Multiplexing

| ~ Layer | Technique | Endpoint | Endpoint Definition |
|---|---|---|---|
| 1 | Uniplex | Wire | Implicit |
| 1/2 | Circuit Switching | Physical Wire Address | Local Network Address |
| 3 | Packet Switching | Routed Logical Address | Routable Network Address (Mac/IP) |
| 4 | Packet Switched Transport | Software Logical Port (linked to a single logical network address) | Routable Network Address<br>+ Transport Protocol<br>+ Port |
| 4+ | Multipath Networking | Process with a Logical Connection Identifier. (Linked to n logical network addresses + transports) | + Multipath Protocol Identifier<br>+ Connection Identifier<br><br>(Transport & Network Agnostic – as long as one given) |
| 5 / 6 / 7 | N/A | URI/URL | URI/URL (Transport-agnostic) |

# Internet Scale Explosion

# Internet Scale Explosion

- Host count

# Internet Scale Explosion

- Host count
- Interfaces / host

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host
- Connections / application

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host
- Connections / application

More Addresses!
(Carrier NAT if that doesn't work)

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host
- Connections / application

More Addresses!
(Carrier NAT if that doesn't work)

More Compute

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host
- Connections / application

More Addresses!
(Carrier NAT if that doesn't work)

More Compute

Workarounds & New Application protocols

# Internet Scale Explosion

- Host count
- Interfaces / host
- Applications / host
- Connections / application

More Addresses!
(Carrier NAT if that doesn't work)

??????

More Compute

Workarounds & New Application protocols

# Why do you care?

# Why do you care?

- Familiar Problems
  - Address Space Exhaustion
  - Route Table Explosion

# Why do you care?

- Familiar Problems
  - Address Space Exhaustion
  - Route Table Explosion
- “New” Problems
  - NAT Table Explosion
  - Client-Controlled routing - Route Arbitrage and Swarming

# TO BE CLEAR:

# TO BE CLEAR:

These technologies are more culture shock than direct vulnerability / concern

# TO BE CLEAR:

These technologies are more culture shock than direct vulnerability / concern

Personally, I like them, and want them to succeed

# TO BE CLEAR:

These technologies are more culture shock than direct vulnerability / concern

Personally, I like them, and want them to succeed

Network tools and operators, need to be ready

# Current TCP is rather limited

# Current TCP is rather limited

Doesn’t support use cases for:

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking

Makes a lot of round trips

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking

Makes a lot of round trips

Blocks stream on retransmits

# Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking


Makes a lot of round trips

Blocks stream on retransmits

# Current TCP is rather limited

Doesn't support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking


Makes a lot of round trips

Blocks stream on retransmits

# Why is this happening?

# Why is this happening?

Networks need multipath, but there's more than one way to do it

# Why is this happening?

Networks need multipath, but there's more than one way to do it

1. MPTCP Extends TCP to Multiplex

# Why is this happening?

Networks need multipath, but there's more than one way to do it

1. MPTCP Extends TCP to Multiplex
2. QUIC IGNORES TCP to handle it itself

# Why is this happening?

Networks need multipath, but there's more than one way to do it

1. MPTCP Extends TCP to Multiplex
2. QUIC IGNORES TCP to handle it itself

But, these technologies change the way the internet behaves

Introduction ✔
Background (Why Change TCP) ✔

# MPTCP

Background Redux (Why NOT to change TCP)
QUIC
Implications
Conclusion and Takeaways

# Earlier:

## Current TCP is rather limited

# Earlier:

## Current TCP is rather limited

Doesn’t support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking

# Multipath TCP

# Multipath TCP

Multipath TCP is an extension to TCP that adds the above functionality

# Multipath TCP

Multipath TCP is an extension to TCP that adds the above functionality

**AND**: it works over existing infrastructure

- (it ***IS*** TCP… just more so)

# Multipath TCP

Multipath TCP is an extension to TCP that adds the above functionality

**AND**: it works over existing infrastructure

- (it ***IS*** TCP… just more so)

# Multipath TCP

Multipath TCP is an extension to TCP that adds the above functionality

**AND**: it works over existing infrastructure

- (it ***IS*** TCP… just more so)

**BUT**: nothing much else understands it

# Motivations and Advantages

# Motivations and Advantages

- TCP implements connections between

# Motivations and Advantages

- TCP implements connections between IP:PORT & IP:PORT, "*without regard to path*"

# Motivations and Advantages

- TCP implements connections between IP:PORT & IP:PORT, "*without regard to path*"

- NOT between endpoint A and endpoint B

# Motivations and Advantages

- TCP implements connections between IP:PORT & IP:PORT, "*without regard to path*"
- NOT between endpoint A and endpoint B
- In the past this was a distinction without a difference, but not any more

# Riding atop of TCP

# Riding atop of TCP

- **An MPTCP Connection** is defined by a connection ID

# Riding atop of TCP

- **An MPTCP Connection** is defined by a connection ID

- It is composed of multiple *streams, where each stream is a regular* TCP connection (with an option strapped on)

# MPTCP Characteristics

# MPTCP Characteristics

- Backwards compatibility

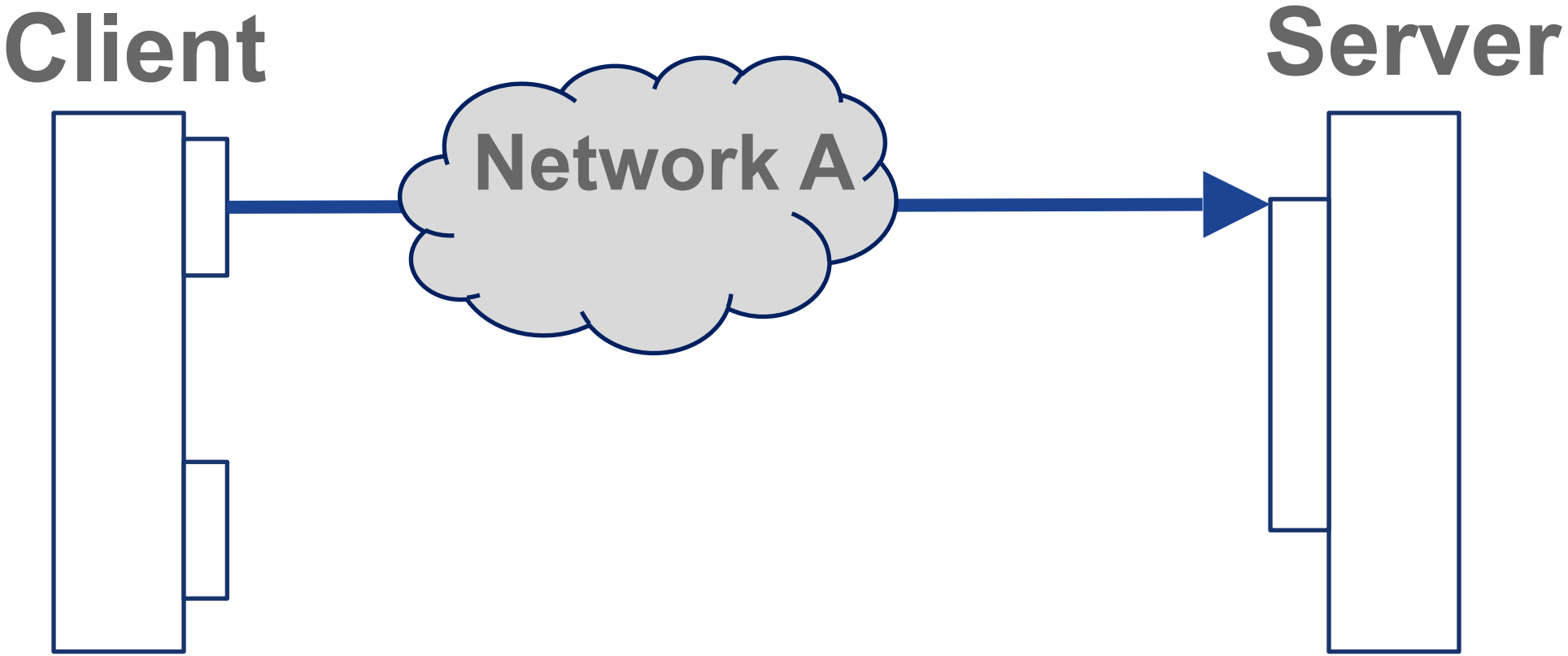
# MPTCP Characteristics

- Backwards compatibility
- Performance >= now

# MPTCP Characteristics

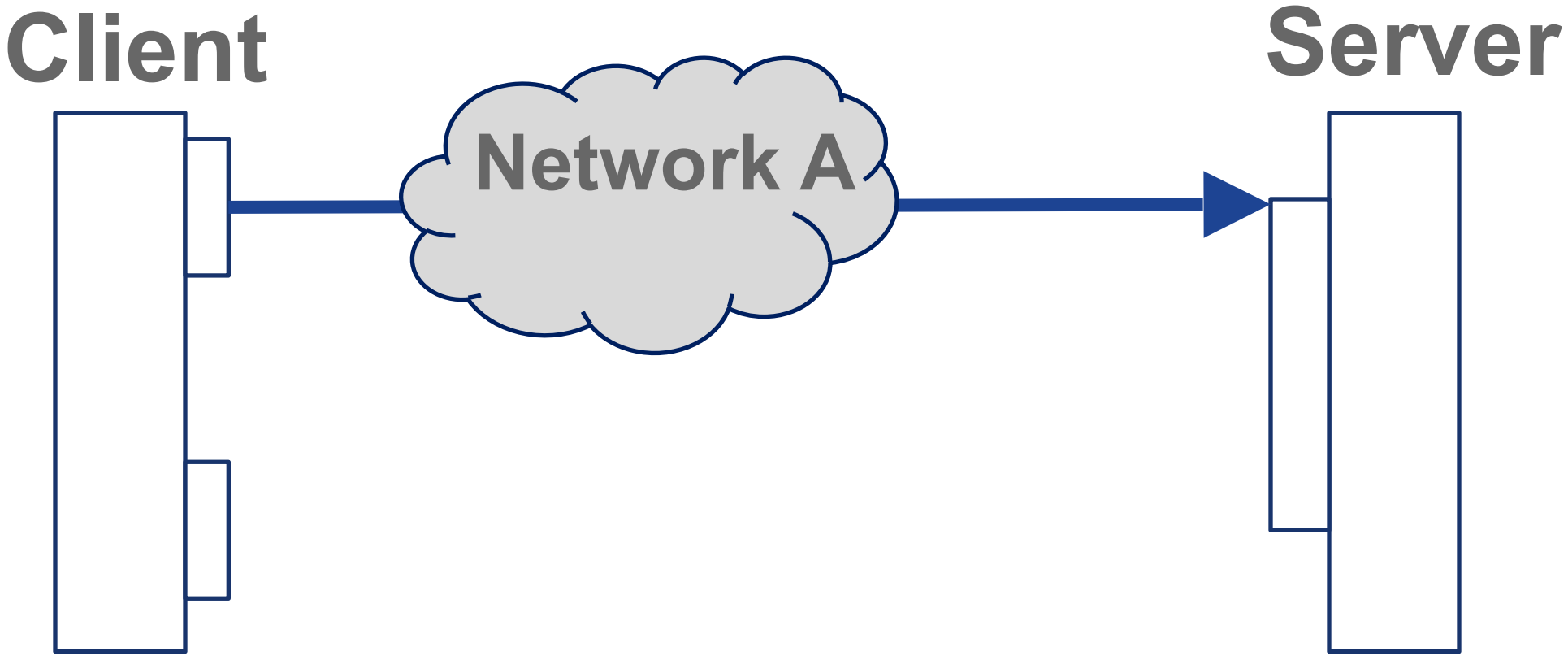
- Backwards compatibility
- Performance >= now
- Security >= now

# MPTCP – Simple Case

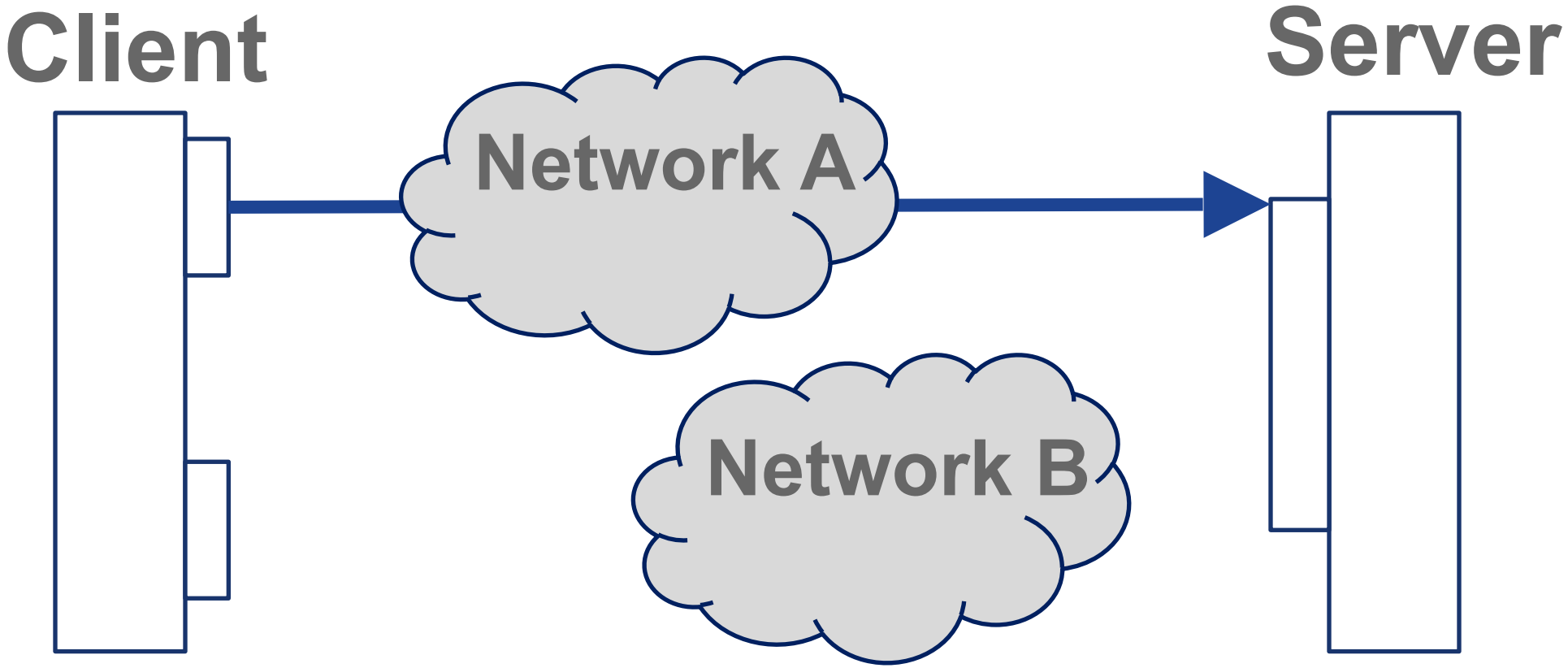



MPTCP connection looks like TCP so far…

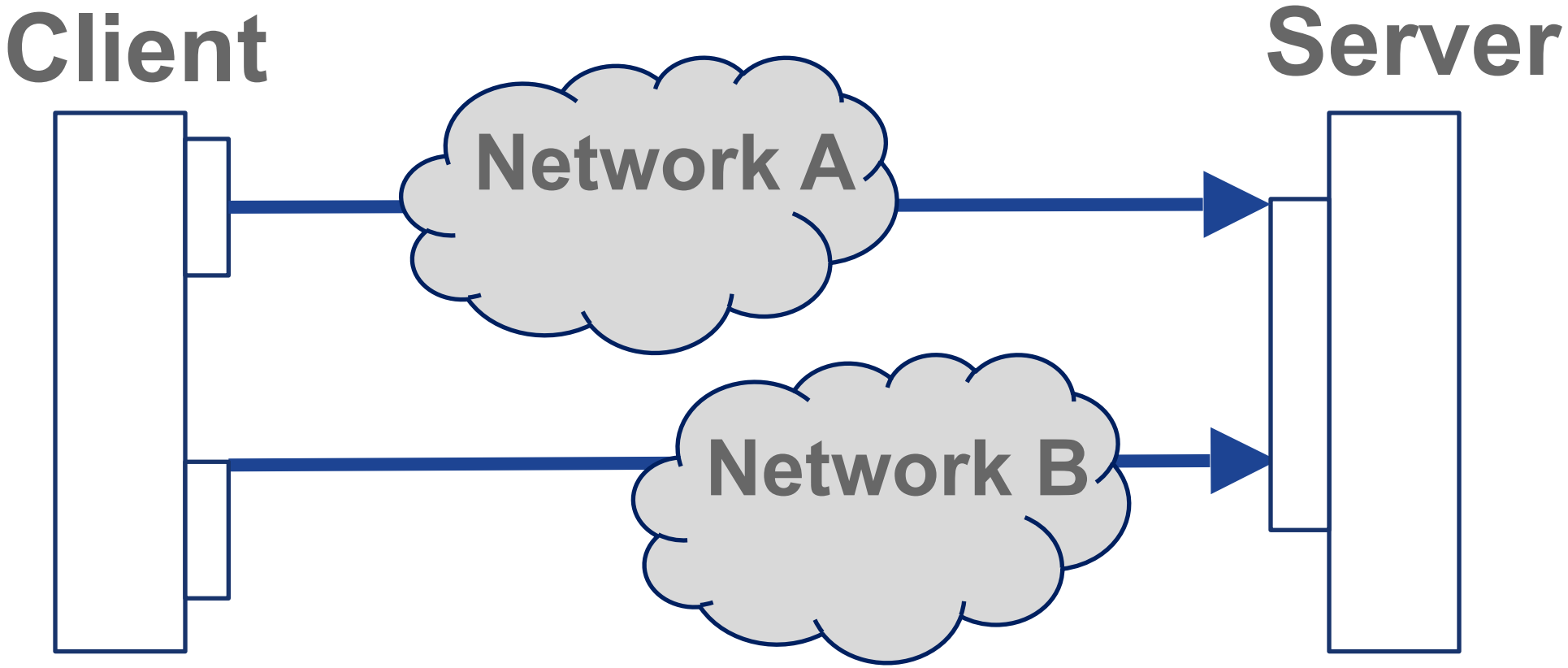
# MPTCP – Simple Case



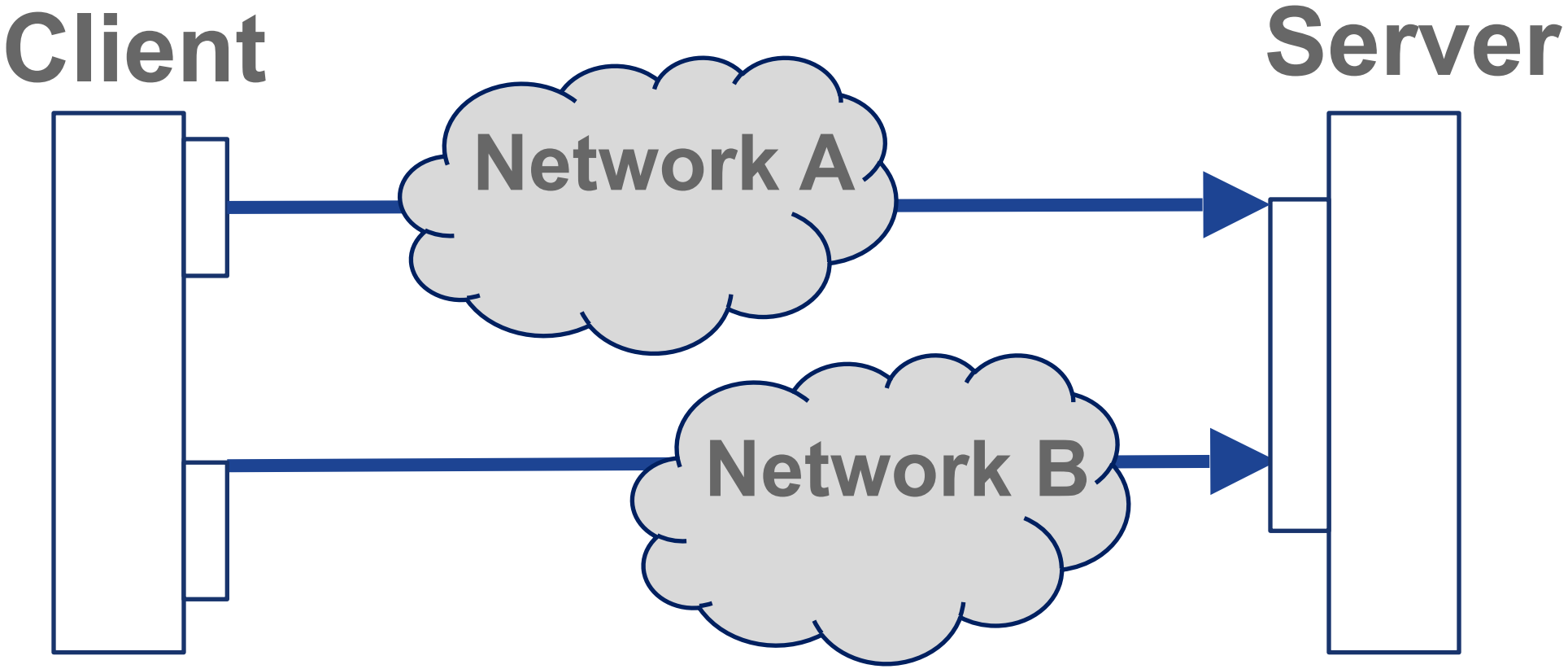
# MPTCP – Simple Case



# MPTCP – Simple Case

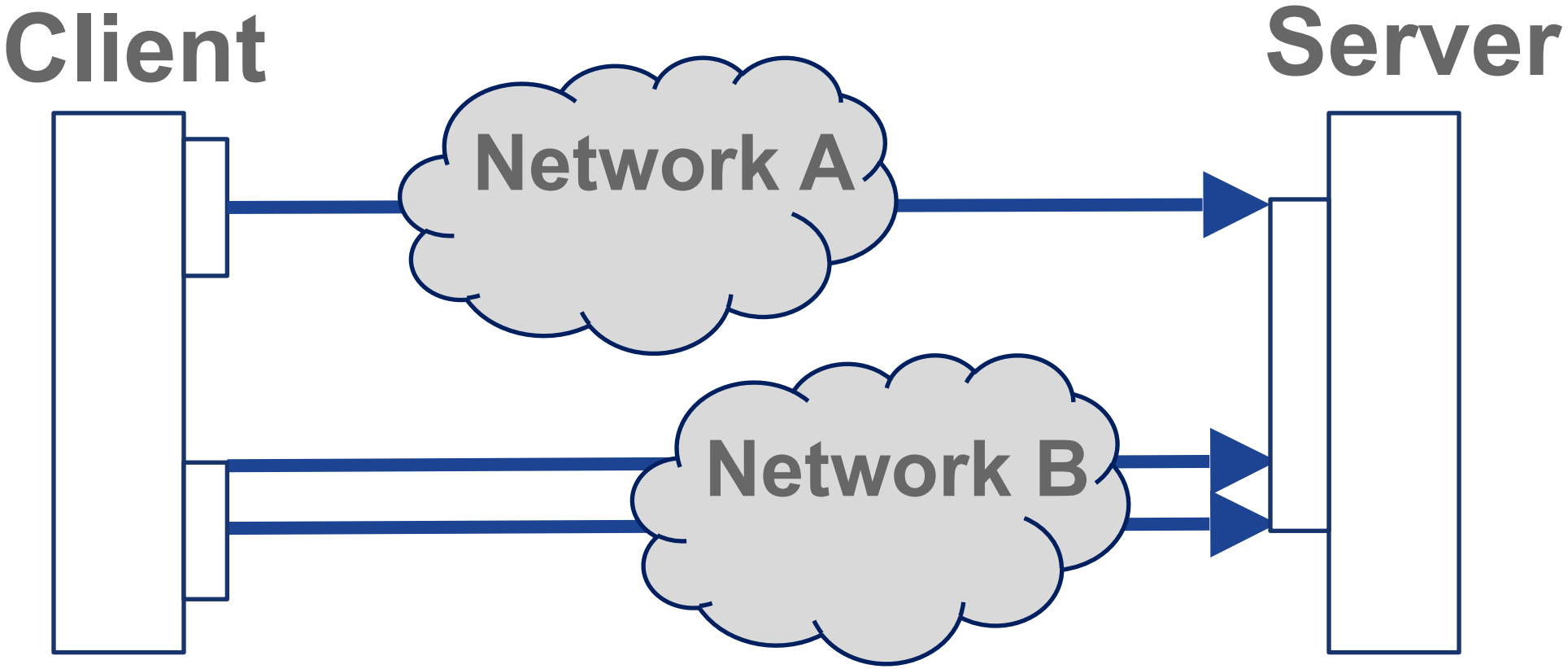


# MPTCP – Simple Case



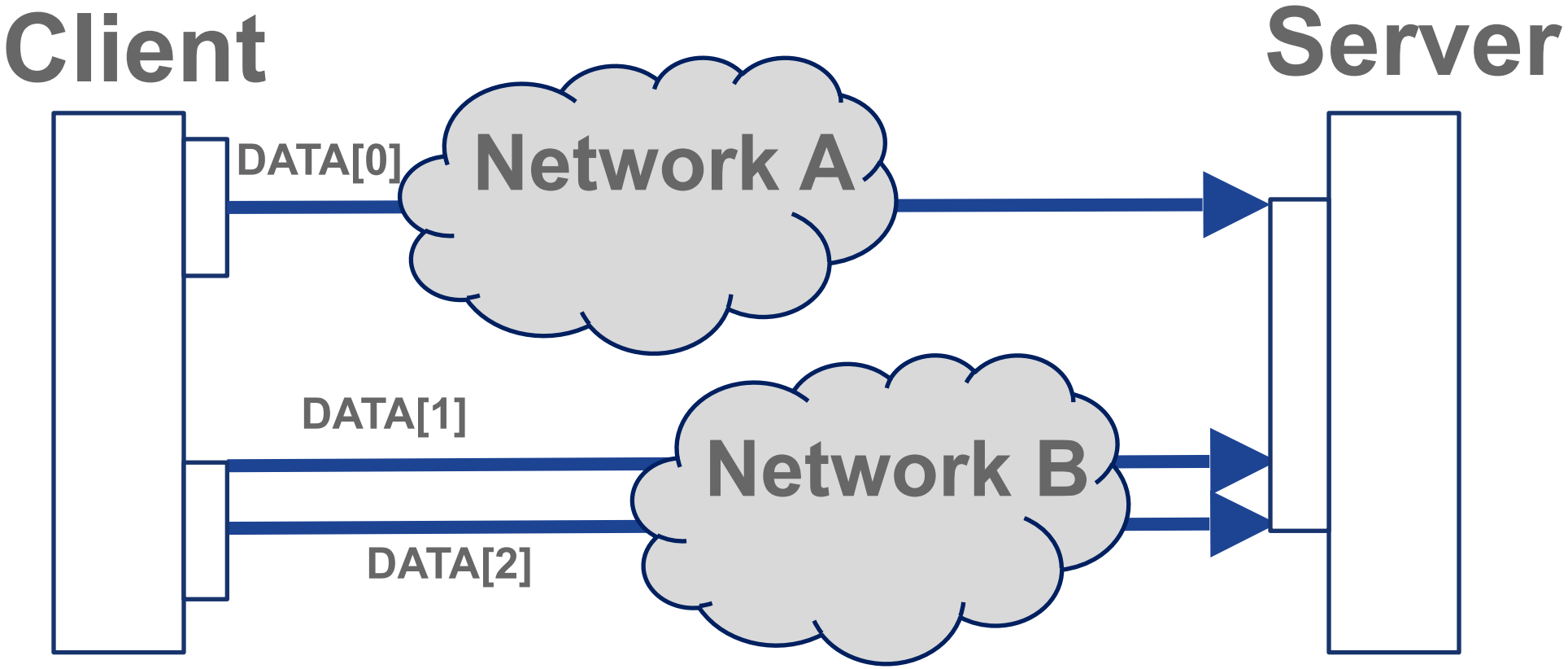

N different TCP connections, contributing to ***ONE*** logical data flow

# MPTCP – Simple Case



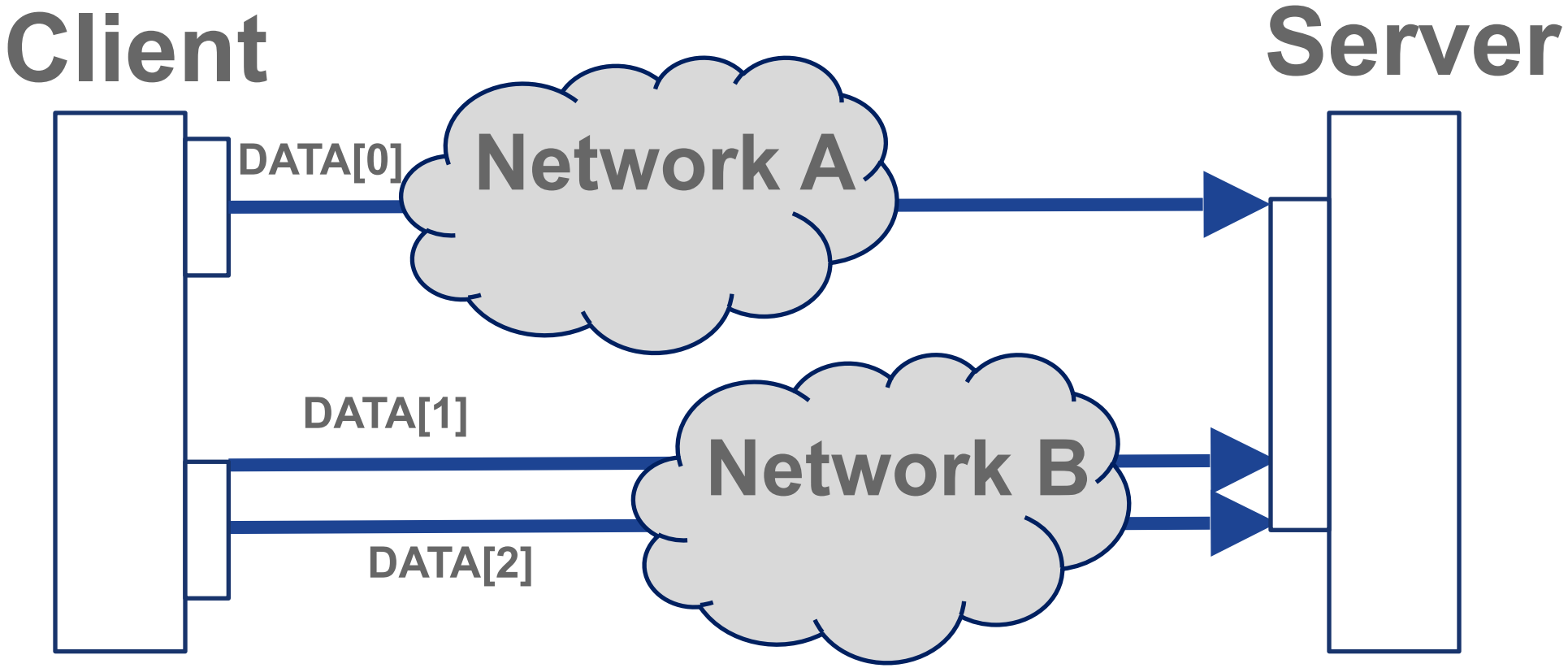

N different TCP connections, contributing to ***ONE*** logical data flow

# MPTCP – Simple Case



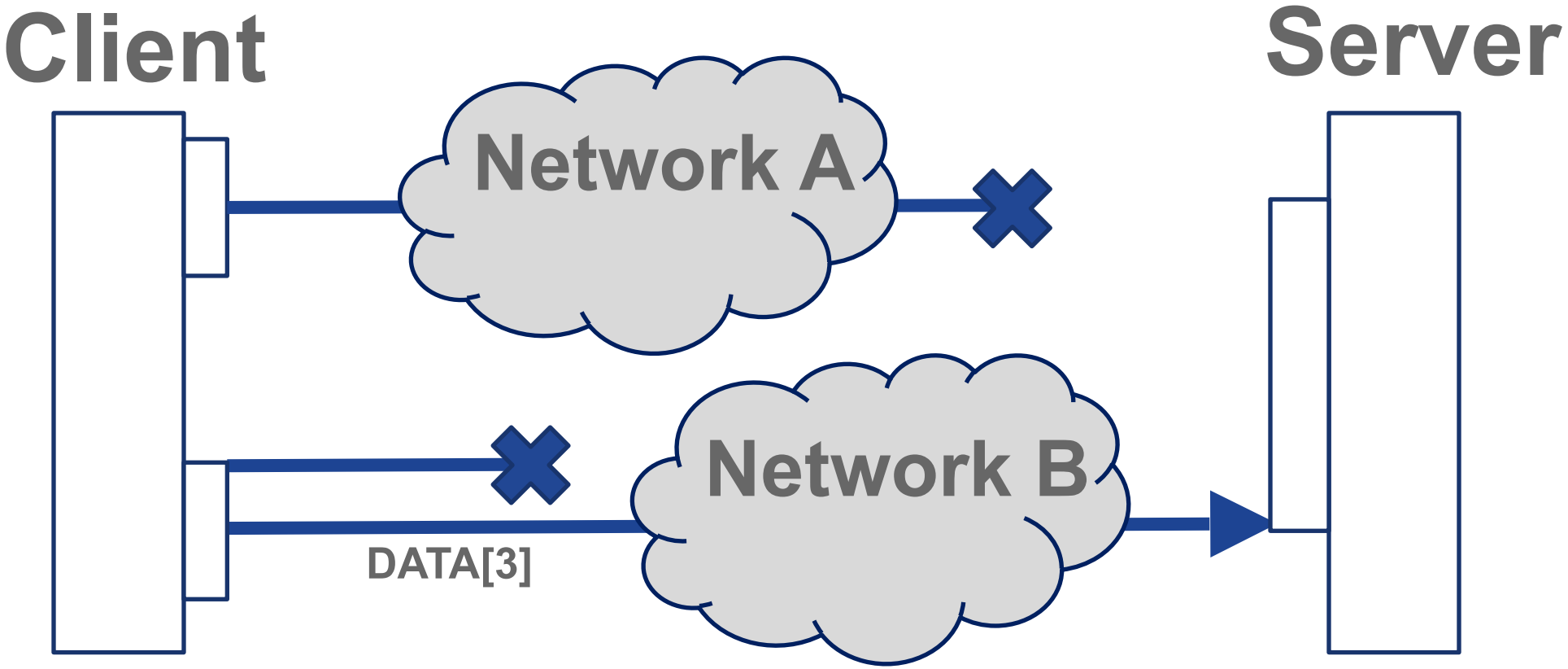

N different TCP connections, contributing to ***ONE*** logical data flow… data flows through any/all

# MPTCP – Simple Case



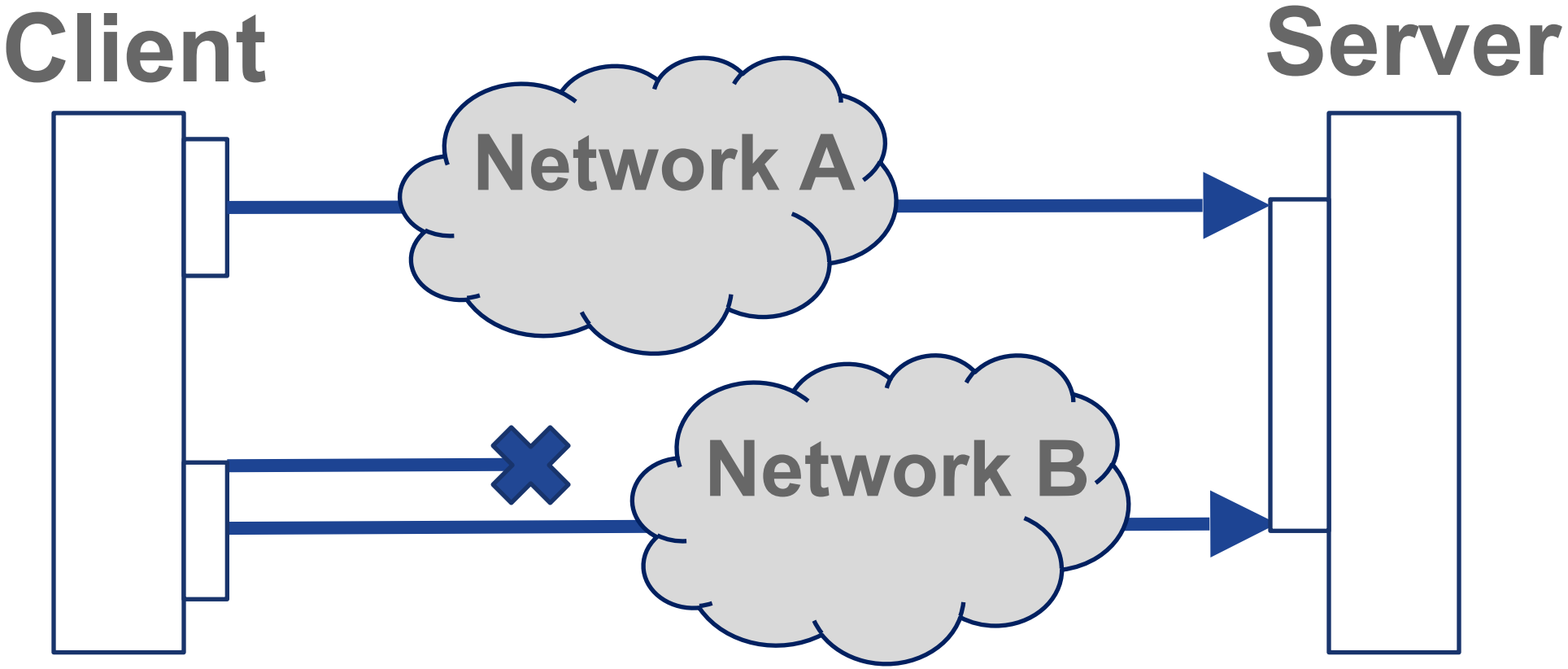

**Sender** of a packet can choose to use any flow *(this will be important)*

# MPTCP – Simple Case



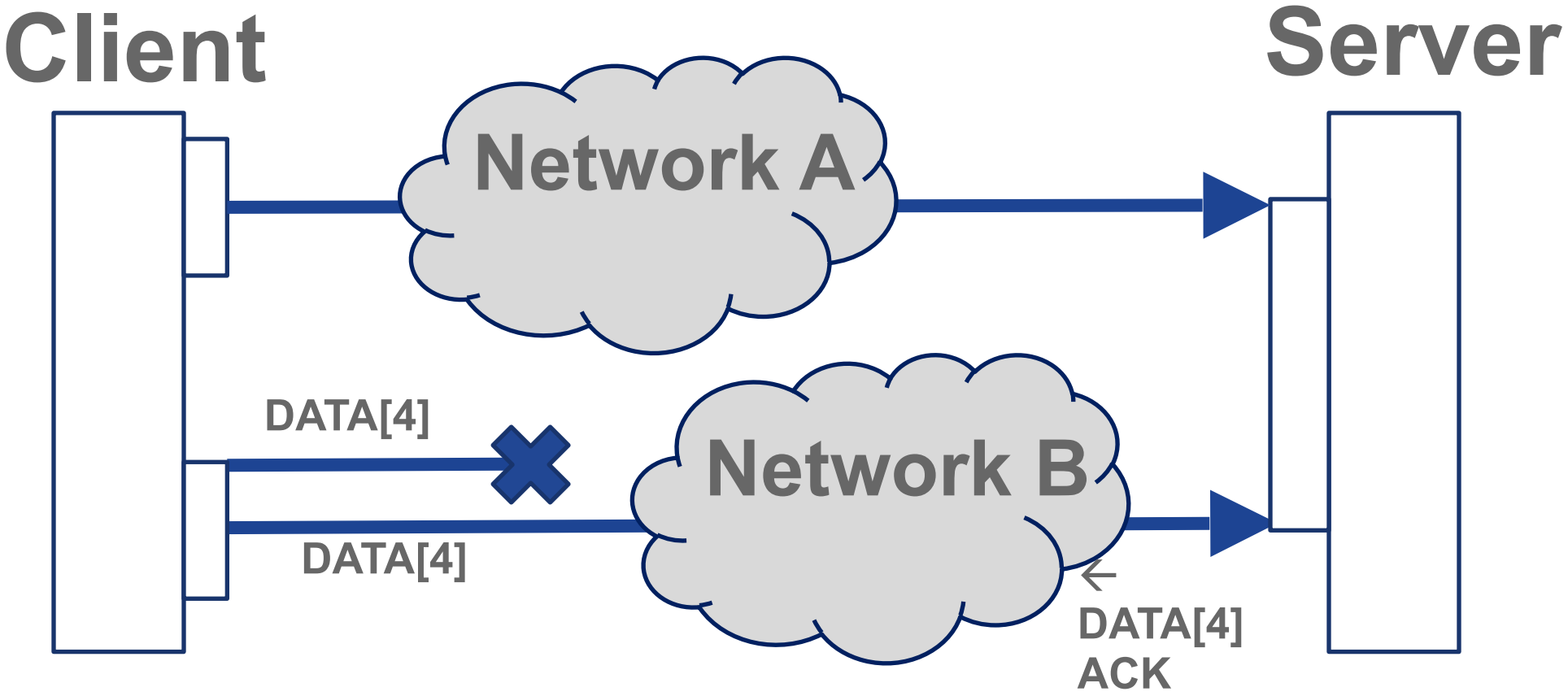

Any subset of connections can drop, overall flow continues.

# MPTCP – Simple Case



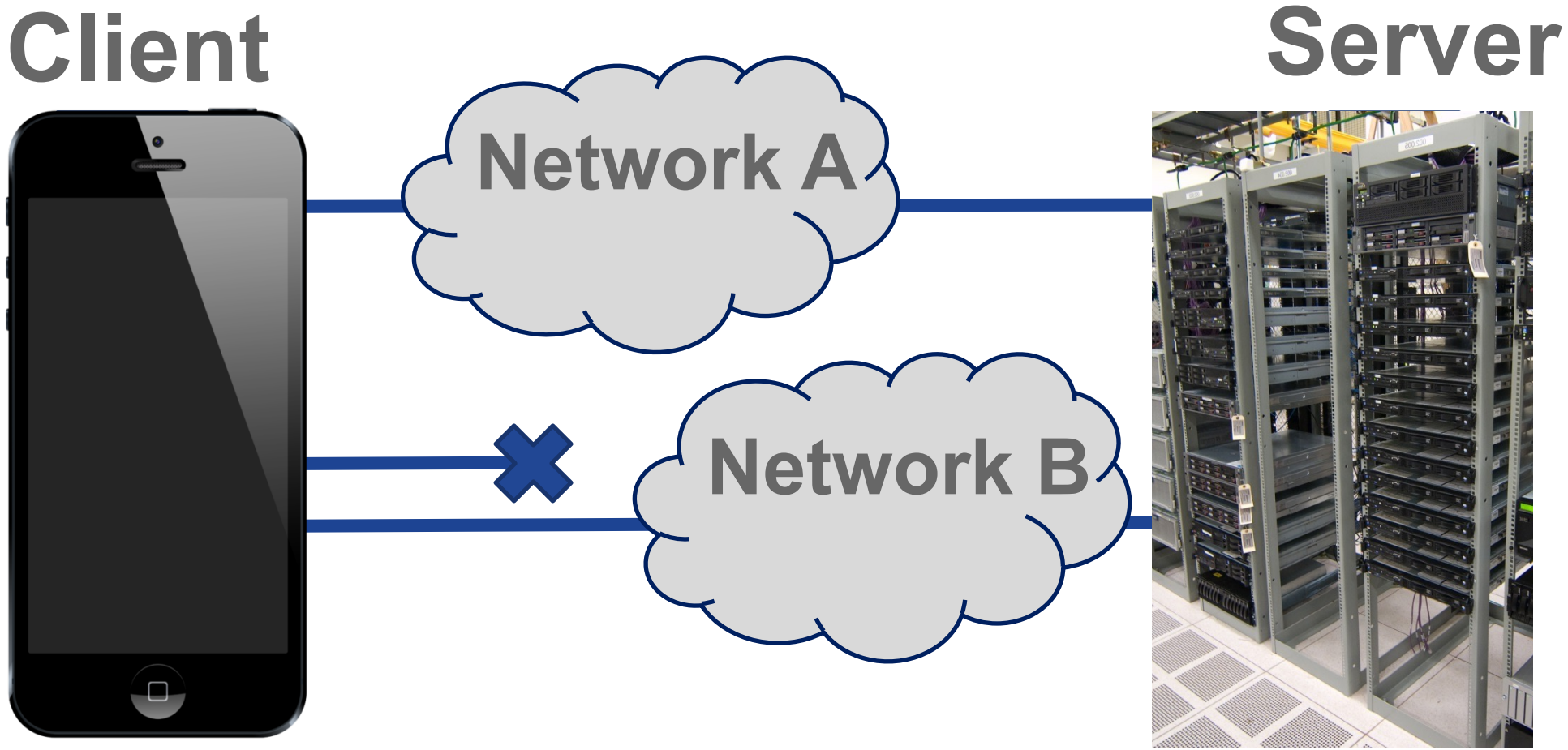

Connections can be re-added at any time

# MPTCP – Simple Case




Un-ACK'd data can be quickly resent over a different flow… first ACK is good enough!

# MPTCP – Basic Use Cases




For seamless roaming

For high availability

# How is MPTCP implemented? – TCP Option

<table>
<tr><td>Bits 0 - 7</td><td>Bits 8 - 15</td><td>Bits 16 - 23</td><td colspan="2">Bits 24 - 31</td></tr>
<tr><td colspan="2">Source Port</td><td colspan="3">Destination port</td></tr>
<tr><td colspan="5">TCP Sequence Number</td></tr>
<tr><td colspan="5">TCP Acknowledgement Number (if Ack Set)</td></tr>
<tr><td>Data Offset</td><td>Reserved</td><td>TCP Flags (Ack, Syn etc)</td><td colspan="2">Window Size</td></tr>
<tr><td colspan="2">Checksum</td><td colspan="3">Urgent Pointer (if URG Set)</td></tr>
<tr><td>0x1e (MPTCP Option Type)</td><td>Length</td><td>Subtype</td><td>MPTCP Ver</td><td>MPTCP Flags</td></tr>
<tr><td colspan="5">Remaining MPTCP Subtype Data</td></tr>
<tr><td colspan="5">Packet DATA</td></tr>
</table>

# What does it look like?

- Packet Breakdown - WireShark

Filter: tcp | Expression... | Clear | Apply | Save

| No. | Time | Source | Destination | Protocol | Length | Info | MPTCP |
|---|---|---|---|---|---|---|---|
| 7 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 86 | 41974 > qip-msgd [SYN] Seq=0 Win=5120 | Multipath Capable |
| 8 | 26.23 | 192.168.110.30 | 192.168.110.10 | TCP | 86 | qip-msgd > 41974 [SYN, ACK] Seq=0 Ack= | Multipath Capable |
| 9 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 94 | 41974 > qip-msgd [ACK] Seq=1 Ack=1 Win= | Multipath Capable,Data Sequer |

```
▷ Checksum: 0x6c1b [validation disabled]
▽ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
  ▷ Maximum segment size: 256 bytes
  ▷ TCP SACK Permitted Option: True
  ▷ Timestamps: TSval 39509940, TSecr 0
  ▷ No-Operation (NOP)
  ▷ Window scale: 7 (multiply by 128)
  ▽ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 12
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ▽ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 376768013894042967
```

```
0000  00 0c 29 3d 01 86 00 0c  29 4d e0 21 08 00 45 00   ..)=.... )M.!..E.
```

# What does it look like?

- Packet Breakdown - WireShark

Filter: tcp | Expression... | Clear | Apply | Save

| No. | Time | Source | Destination | Protocol | Length | Info | MPTCP |
|---|---|---|---|---|---|---|---|
| 7 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 86 | 41974 > qip-msgd [SYN] Seq=0 Win=5120 | Multipath Capable |
| 8 | 26.23 | 192.168.110.30 | 192.168.110.10 | TCP | 86 | qip-msgd > 41974 [SYN, ACK] Seq=0 Ack= | Multipath Capable |
| 9 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 94 | 41974 > qip-msgd [ACK] Seq=1 Ack=1 Win= | Multipath Capable,Data Sequer |

```
▷ Checksum: 0x6c1b [validation disabled]
▽ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
  ▷ Maximum segment size: 256 bytes
  ▷ TCP SACK Permitted Option: True
  ▷ Timestamps: TSval 39509940, TSecr 0
  ▷ No-Operation (NOP)
  ▷ Window scale: 7 (multiply by 128)
  ▽ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 12
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ▽ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 376768013894042967
```

```
0000  00 0c 29 3d 01 86 00 0c  29 4d e0 21 08 00 45 00   ..)=.... )M.!..E.
```

# What does it look like?

- Packet Breakdown - WireShark

Filter: tcp | Expression... | Clear | Apply | Save

| No. | Time | Source | Destination | Protocol | Length | Info | MPTCP |
|---|---|---|---|---|---|---|---|
| 7 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 86 | 41974 > qip-msgd [SYN] Seq=0 Win=5120 | Multipath Capable |
| 8 | 26.23 | 192.168.110.30 | 192.168.110.10 | TCP | 86 | qip-msgd > 41974 [SYN, ACK] Seq=0 Ack= | Multipath Capable |
| 9 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 94 | 41974 > qip-msgd [ACK] Seq=1 Ack=1 Win= | Multipath Capable,Data Sequer |

```
▷ Checksum: 0x6c1b [validation disabled]
▽ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
  ▷ Maximum segment size: 256 bytes
  ▷ TCP SACK Permitted Option: True
  ▷ Timestamps: TSval 39509940, TSecr 0
  ▷ No-Operation (NOP)
  ▷ Window scale: 7 (multiply by 128)
  ▽ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 12
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ▽ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 376768013894042967
```

**TCP Options field**

```
0000  00 0c 29 3d 01 86 00 0c  29 4d e0 21 08 00 45 00   ..)=.... )M.!..E.
```

# What does it look like?

- Packet Breakdown - WireShark

Filter: tcp | Expression... | Clear | Apply | Save

| No. | Time | Source | Destination | Protocol | Length | Info | MPTCP |
|---|---|---|---|---|---|---|---|
| 7 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 86 | 41974 > qip-msgd [SYN] Seq=0 Win=5120 | Multipath Capable |
| 8 | 26.23 | 192.168.110.30 | 192.168.110.10 | TCP | 86 | qip-msgd > 41974 [SYN, ACK] Seq=0 Ack= | Multipath Capable |
| 9 | 26.23 | 192.168.110.10 | 192.168.110.30 | TCP | 94 | 41974 > qip-msgd [ACK] Seq=1 Ack=1 Win= | Multipath Capable,Data Sequer |

```
▷ Checksum: 0x6c1b [validation disabled]
▽ Options: (32 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale, Multipath TCP
  ▷ Maximum segment size: 256 bytes
  ▷ TCP SACK Permitted Option: True
  ▷ Timestamps: TSval 39509940, TSecr 0
  ▷ No-Operation (NOP)
  ▷ Window scale: 7 (multiply by 128)
  ▽ Multipath TCP: Multipath Capable
      Kind: Multipath TCP (30)
      Length: 12
      0000 .... = Multipath TCP subtype: Multipath Capable (0)
      .... 0000 = Multipath TCP version: 0
    ▽ Multipath TCP flags: 0x81
        1... .... = Checksum required: 1
        .... ...1 = Use HMAC-SHA1: 1
      Multipath TCP Sender's Key: 376768013894042967
```

**Option 30 (0x1E)**

**TCP Options field**

```
0000  00 0c 29 3d 01 86 00 0c  29 4d e0 21 08 00 45 00   ..)=.... )M.!..E.
```

# Path Management - Linux

# Path Management - Linux

- The Linux Path Manager has two primary path managers at present
  - Fullmesh – n:n (all to all)
  - Ndiffports – 1-1 interfaces, n-1 ports

# Path Management - Linux

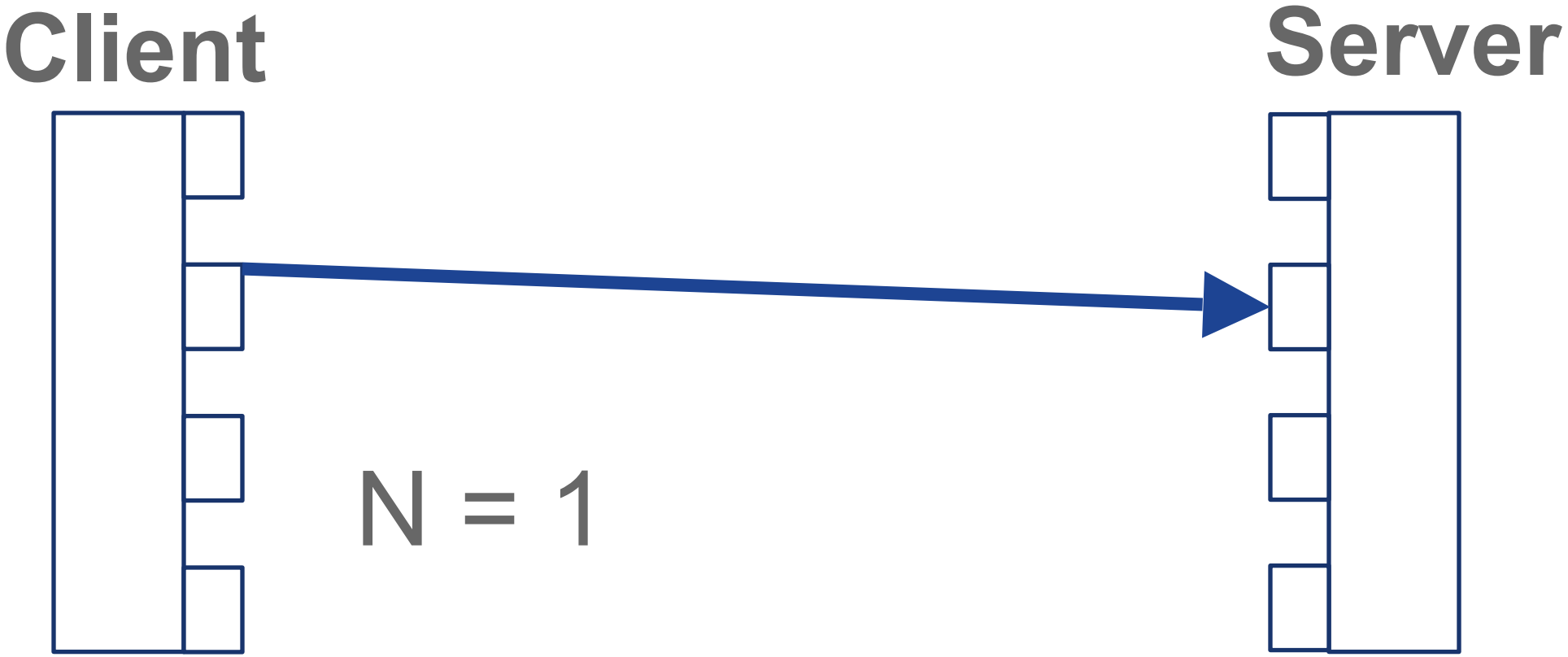
- The Linux Path Manager has two primary path managers at present
  - Fullmesh – n:n (all to all)
  - Ndiffports – 1-1 interfaces, n-1 ports

- This is in the **TCP stack**… application layers get MPTCP for free (mostly)

# Path Management - ndiffports

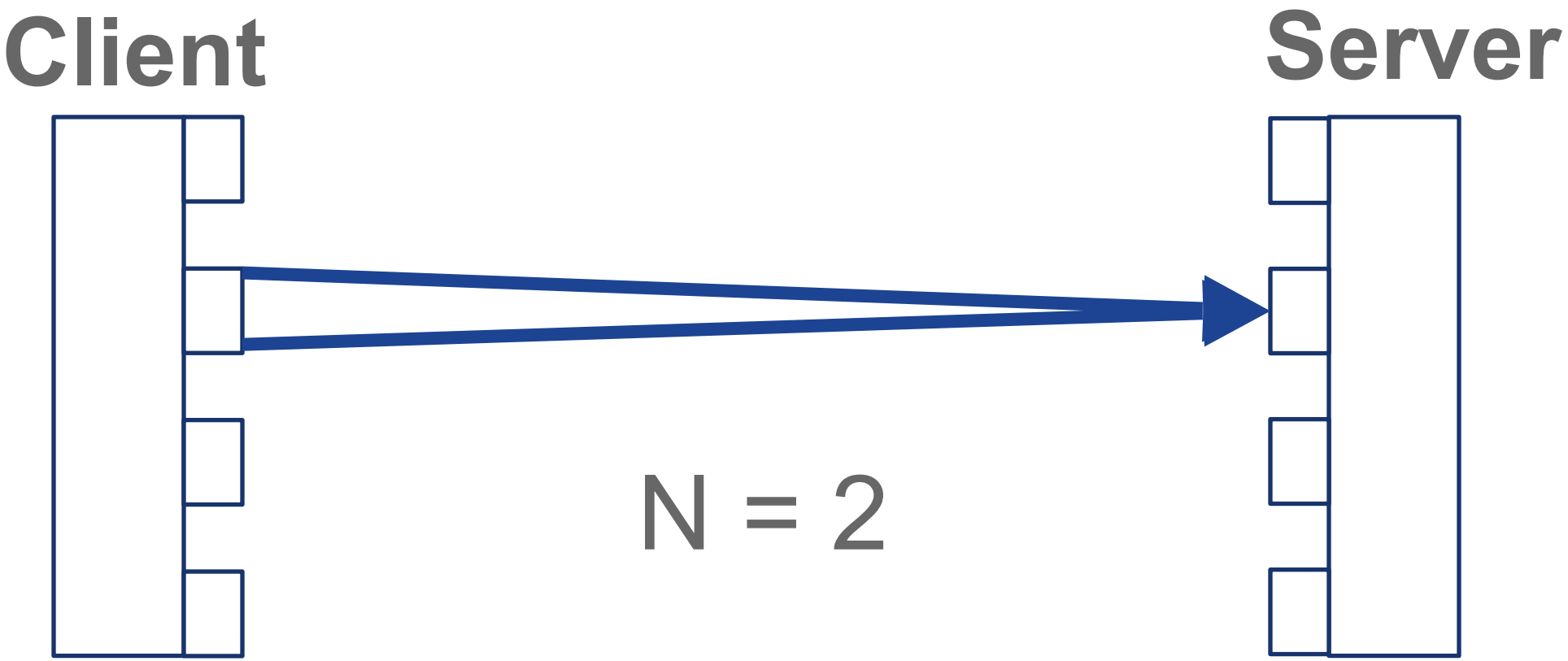

N different source ports,
1 destination port

# Path Management - ndiffports



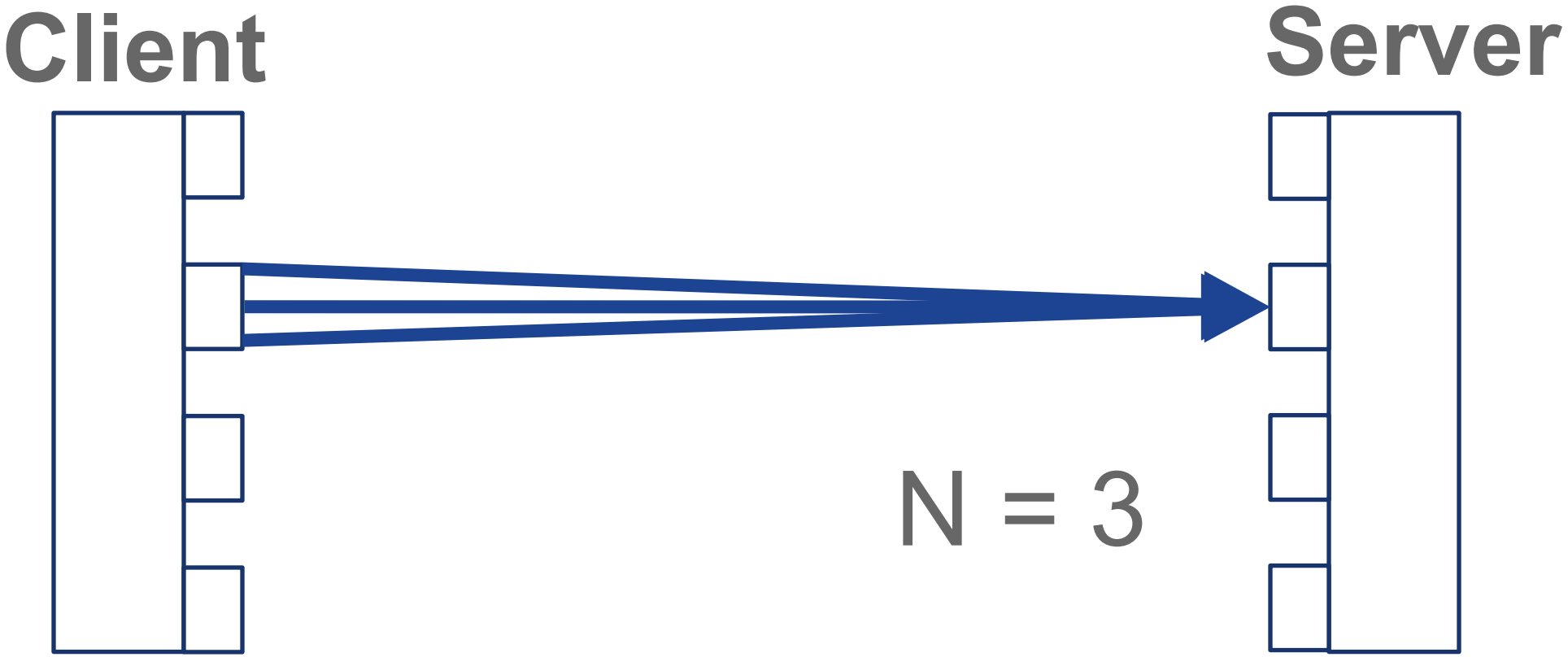

N different source ports,
1 destination port

# Path Management - ndiffports




N different source ports,
1 destination port

# Path Management - ndiffports

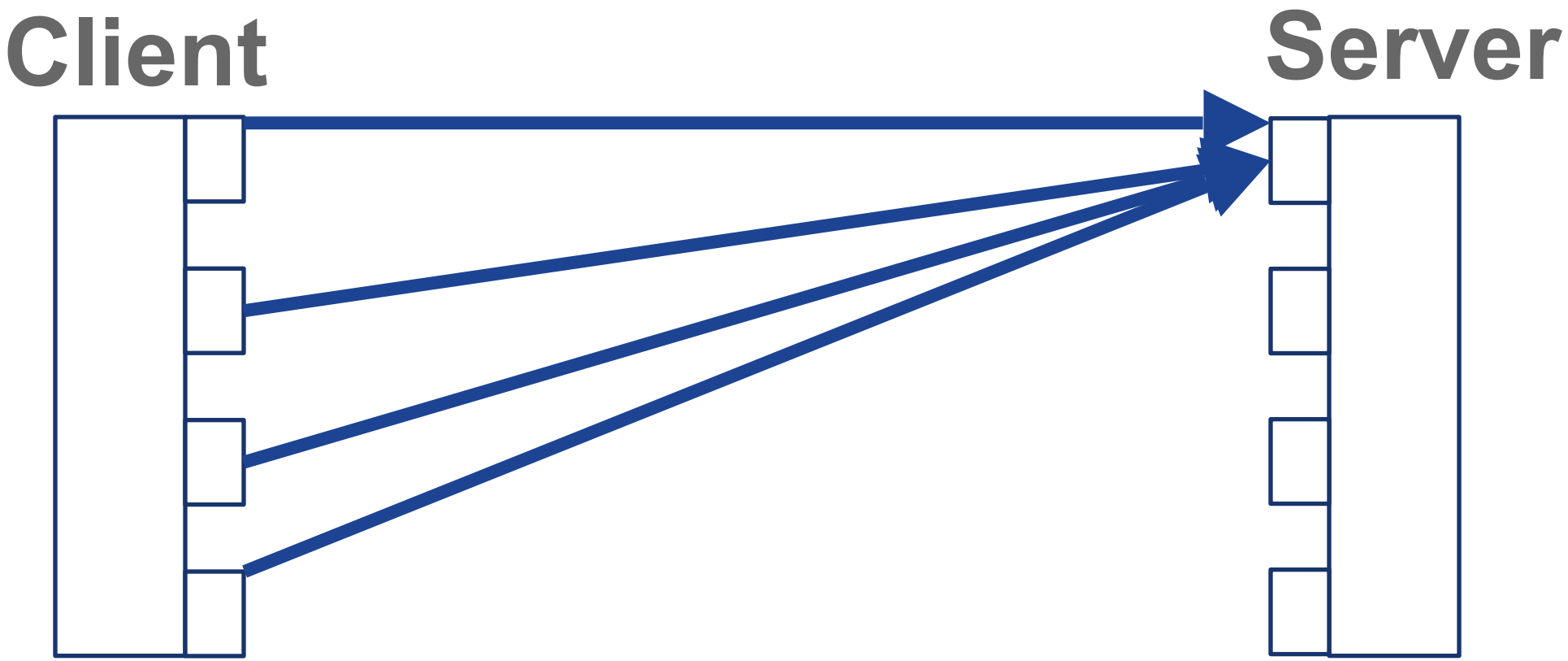



N different source ports,
1 destination port

# Path Management - fullmesh

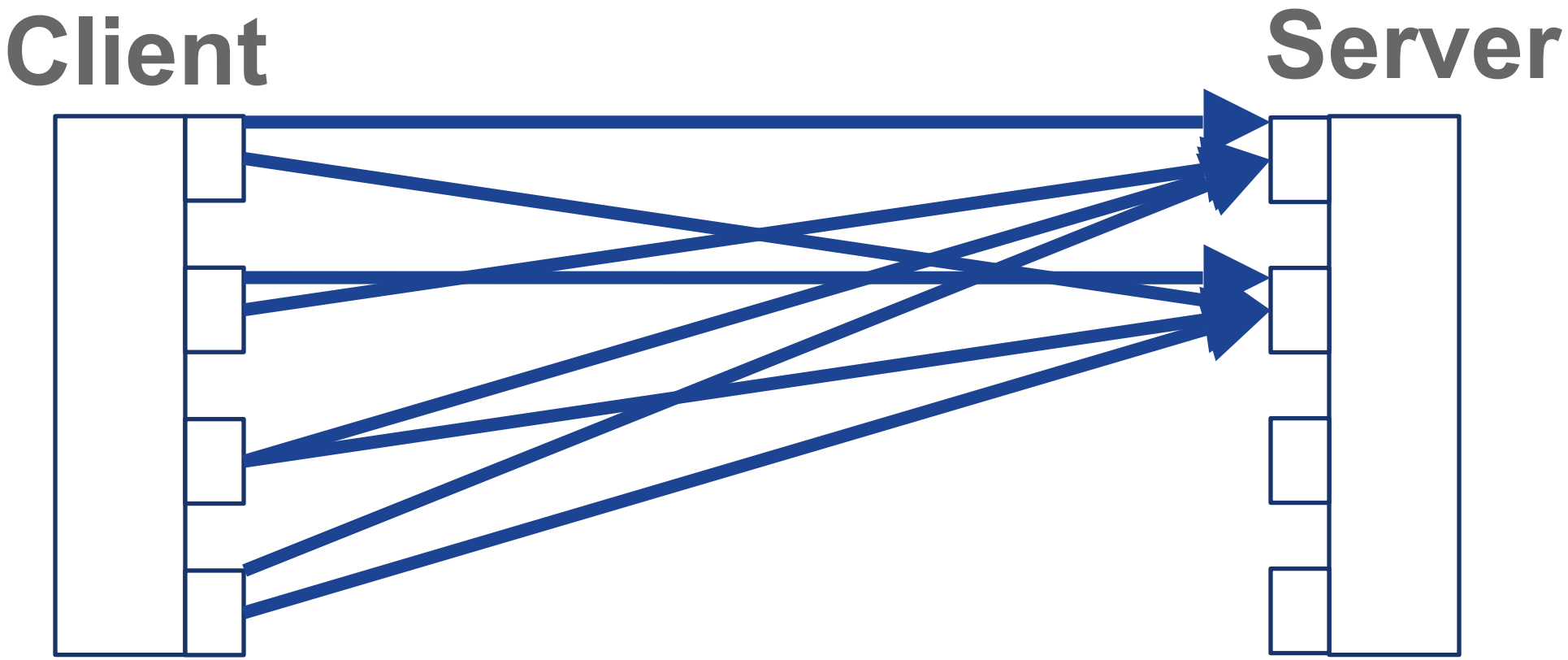

All possible paths used

# Path Management - fullmesh



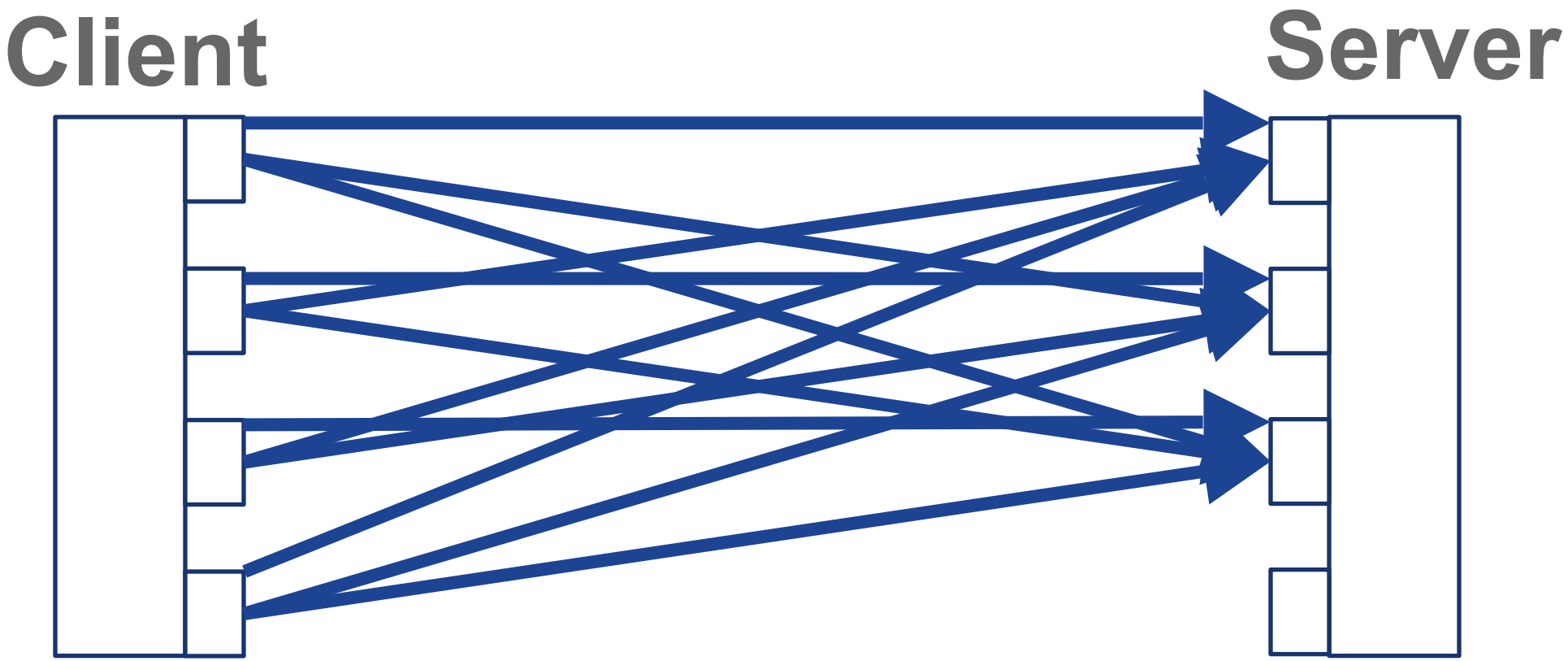

All possible paths used

# Path Management - fullmesh



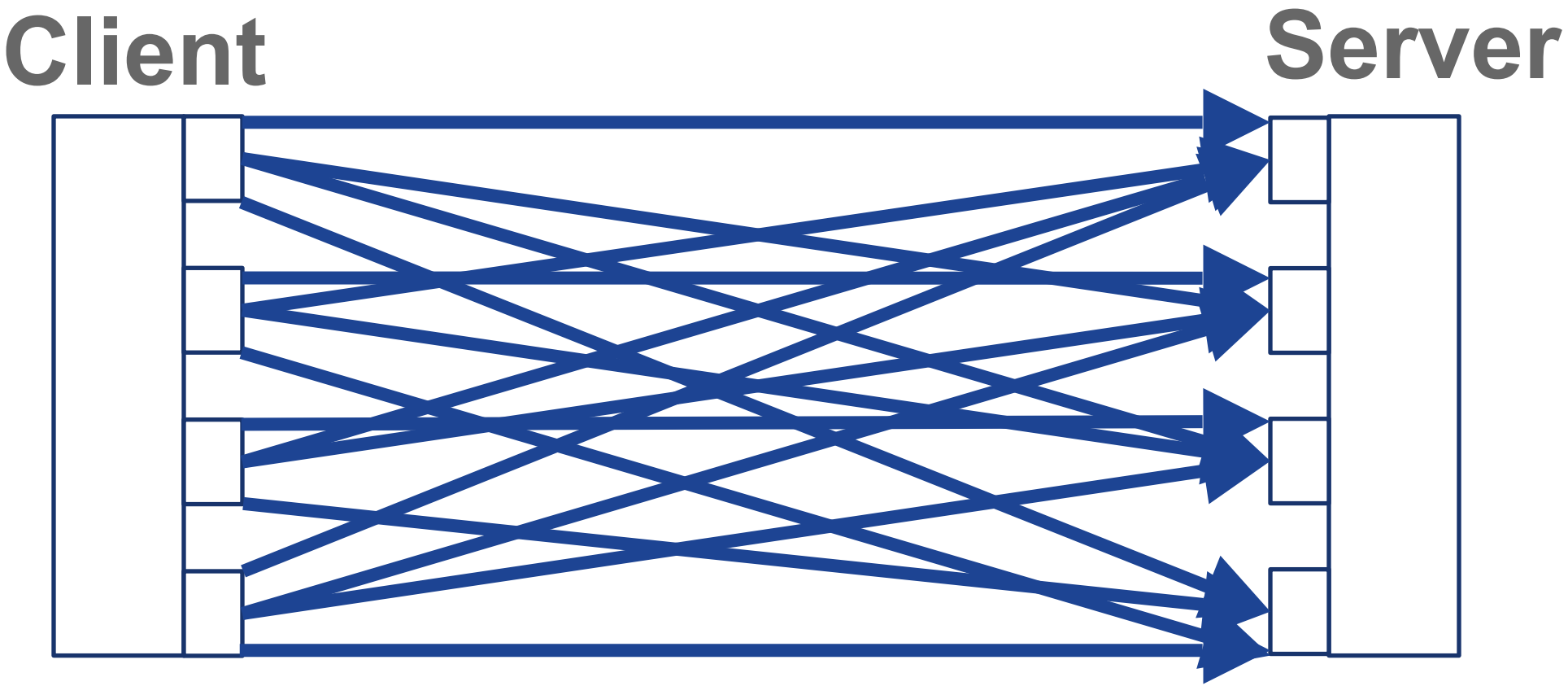

All possible paths used

# Path Management - fullmesh




All possible paths used

# Path Management - fullmesh




All possible paths used

Introduction ✔
Background (Why Change TCP) ✔
MPTCP ✔

# Background Redux (Why NOT TO change TCP)

QUIC
Implications
Conclusion and Takeaways

# Why NOT Change TCP?

# Why NOT Change TCP?

## Lessons from MPTCP

# Why NOT Change TCP?

## Lessons from MPTCP

- Slow moving

# Why NOT Change TCP?

## Lessons from MPTCP

- Slow moving
- Meddleboxes limit protocol deployability

# Why NOT Change TCP?

## Lessons from MPTCP

- Slow moving
- Meddleboxes limit protocol deployability
- Chicken and egg deployment

# Why NOT Change TCP?

# Why NOT Change TCP?

TCP Characteristics:

# Why NOT Change TCP?

TCP Characteristics:

- Handshake design

# Why NOT Change TCP?

TCP Characteristics:

- Handshake design
- Outside user-space

# Why NOT Change TCP?

TCP Characteristics:

- Handshake design
- Outside user-space
- End-of-line Blocking

# Why NOT Change TCP?

# Why NOT Change TCP?

If you can’t change TCP what’s left?

# Why NOT Change TCP?

If you can’t change TCP what’s left?

- SCTP?
  - Same problems, but amplified

# Why NOT Change TCP?

If you can’t change TCP what’s left?

- SCTP?
  - Same problems, but amplified

# Why NOT Change TCP?

If you can’t change TCP what’s left?

- SCTP?
  - Same problems, but amplified
- UDP?
  - But it doesn’t do ANYTHING fancy?
  - Exactly.

Introduction ✔
Background (Why Change TCP) ✔
MPTCP ✔
Background Redux (Why NOT TO change TCP) ✔

# QUIC

Implications
Conclusion and Takeaways

# QUIC
# (Quick UDP Internet Connections)

# QUIC
## (Quick UDP Internet Connections)

- You thought MPTCP developed fast?

# QUIC
## (Quick UDP Internet Connections)

- You thought MPTCP developed fast?

- QUIC was even QUIC-ker
  - Already in use on many Google properties
  - Youtube, Google search, and more
  - Likely several percent of your traffic

# Current TCP is rather limited

# Current TCP is rather limited

Makes a lot of round trips

# Current TCP is rather limited

Makes a lot of round trips

Blocks stream on retransmits

# Current TCP is rather limited

Makes a lot of round trips

Blocks stream on retransmits

# QUIC
# (Quick UDP Internet Connections)

# QUIC
## (Quick UDP Internet Connections)

UDP transport protocol

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*
- *Always Encrypted*

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*
- *Always Encrypted*

**Open Source**

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*
- *Always Encrypted*

**Open Source**

***User Space***

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*
- *Always Encrypted*

**Open Source**

***User Space***

- *No OS requirements*

# QUIC
## (Quick UDP Internet Connections)

**UDP transport protocol**

- *Google championed successor to SPDY*
- *Latency Optimized*
- *Reliable, Multiplexed*
- *Always Encrypted*

**Open Source**

***User Space***

- *No OS requirements*
- *Fast Evolving*

# QUIC connection – Latency Reduction

# QUIC connection – Latency Reduction

- 0-RTT Connection establishment (1 sometimes)

# QUIC connection – Latency Reduction

- 0-RTT Connection establishment (1 sometimes)
- FEC-based packet-loss recovery

# QUIC connection – Latency Reduction

- 0-RTT Connection establishment (1 sometimes)
- FEC-based packet-loss recovery
- Flow Control at both connection and stream level

# QUIC connection – Latency Reduction

- 0-RTT Connection establishment (1 sometimes)
- FEC-based packet-loss recovery
- Flow Control at both connection and stream level
- Certificate and header compression

# QUIC connection – Benefits

- Google claims:
  - 30% reduction in video rebuffers
  - 10% reduction in page load times

https://peering.google.com/#/learn-more/quic

# QUIC connection - Overview

# QUIC connection - Overview

- UDP Port 80 and 443

# QUIC connection - Overview

- UDP Port 80 and 443

- Upgrade headers in HTTP:
  - Alternate-Protocol: [port]:quic
  - Alternate-Protocol: 443:quic

# QUIC Infodump 0% [Starting…]

# QUIC Infodump 0% [Starting…]

- Quic isn’t simple.
  - Read the RFC, docs, and source

# QUIC Infodump 0% [Starting…]

- Quic isn’t simple.
  - Read the RFC, docs, and source

- QUIC is way more intricate than MPTCP

# QUIC Infodump 0% [Starting…]

- Quic isn’t simple.
  - Read the RFC, docs, and source

- QUIC is way more intricate than MPTCP

# QUIC Infodump 0% [Starting…]

- Quic isn't simple.
  - Read the RFC, docs, and source
- QUIC is way more intricate than MPTCP
- I'm going to RACE through some details to show the complexity of QUIC

# QUIC Infodump 5% [x----------] Packet Types

# QUIC Infodump 5% [x----------] Packet Types

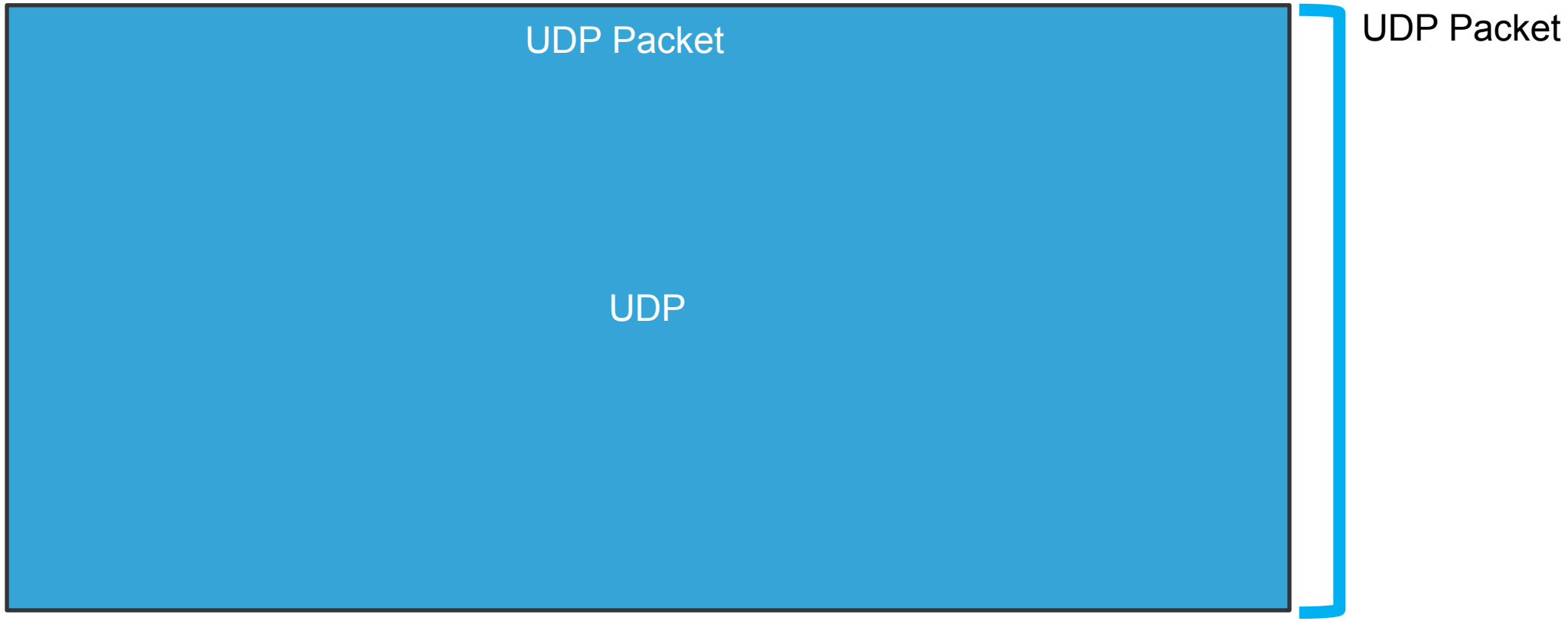
- Regular Packets
  - Frame Packets

# QUIC Infodump 5% [x----------] Packet Types

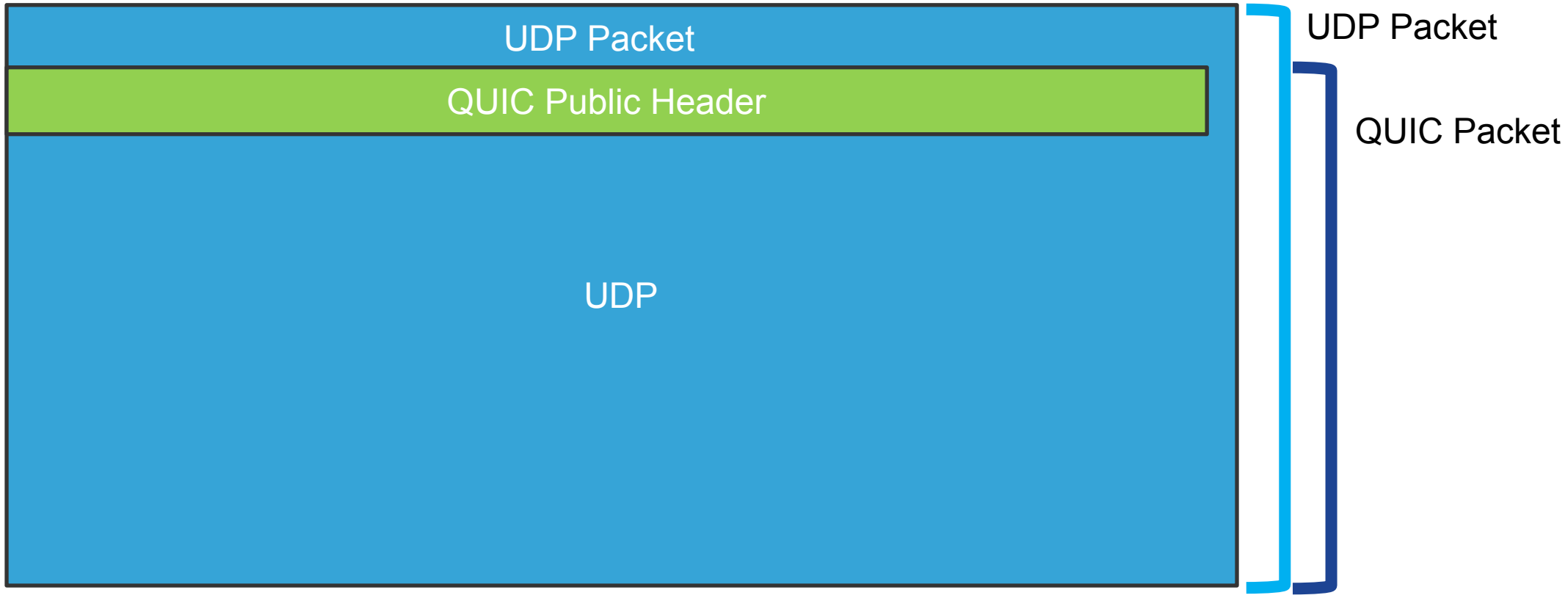
- Regular Packets
  - Frame Packets
- Special packet types
  - Version negotiation,
  - Public reset

# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview

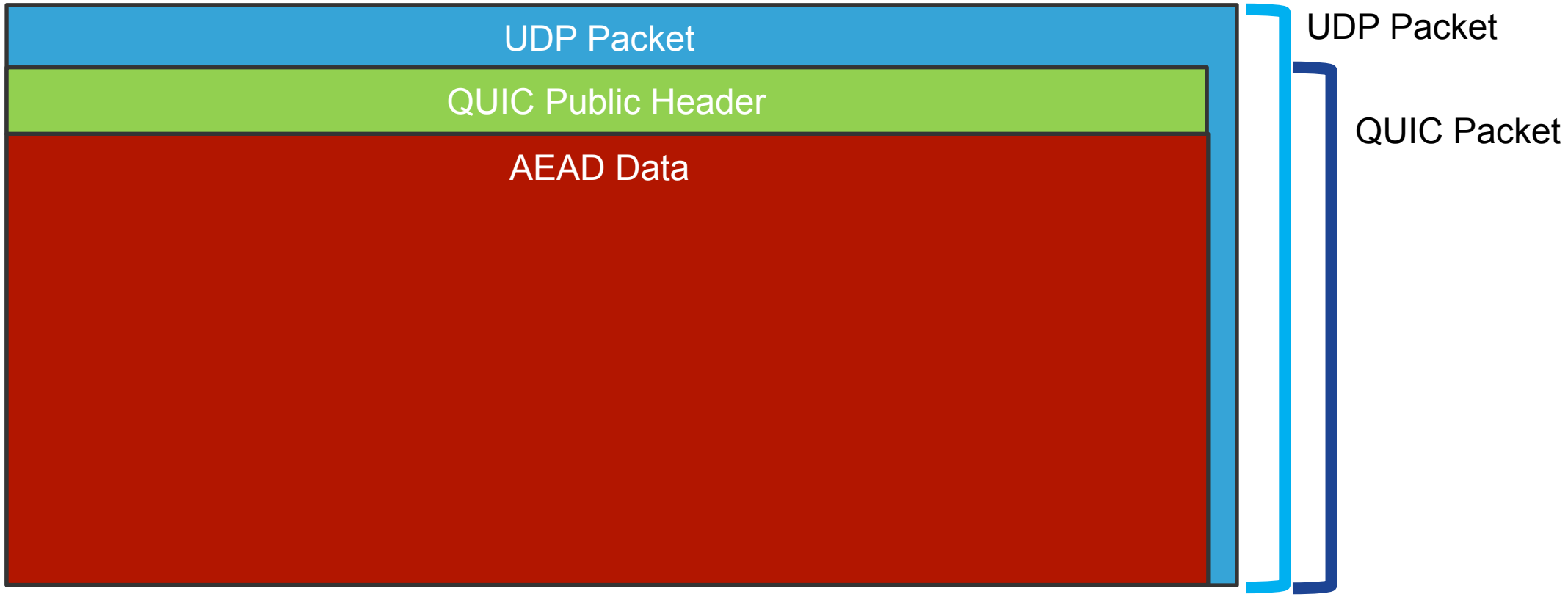
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



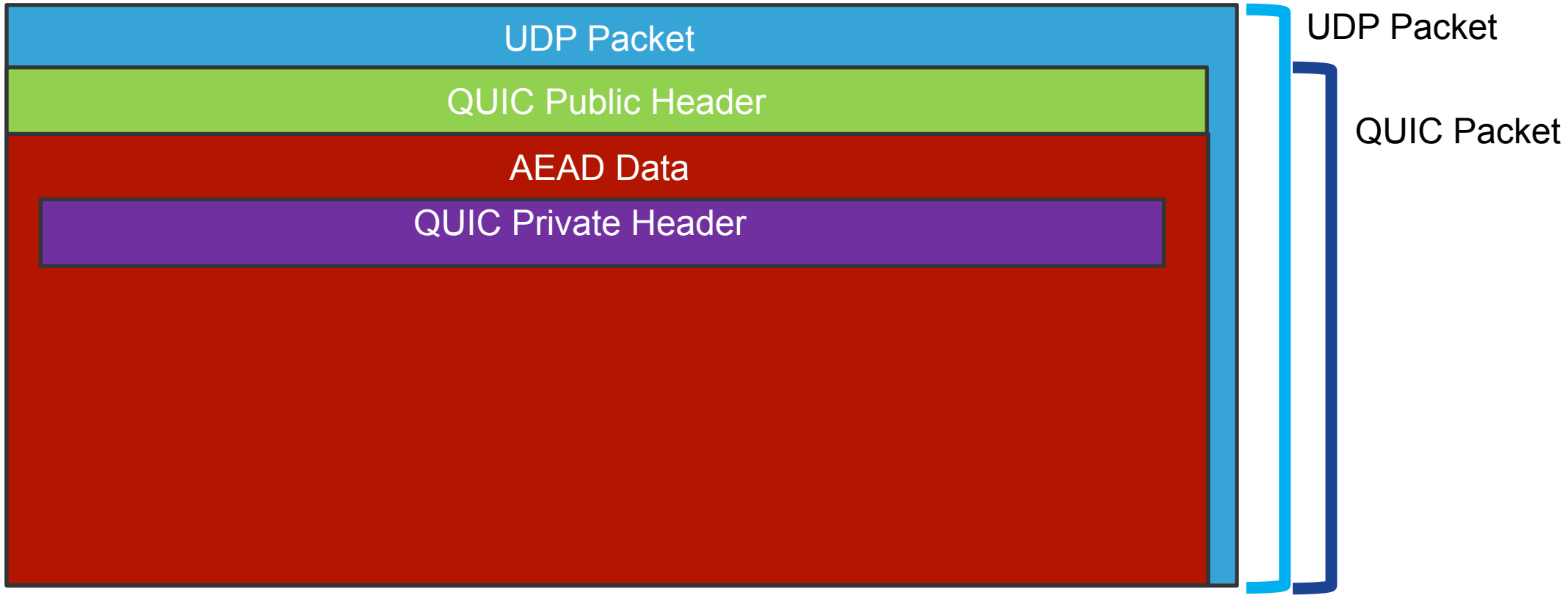
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



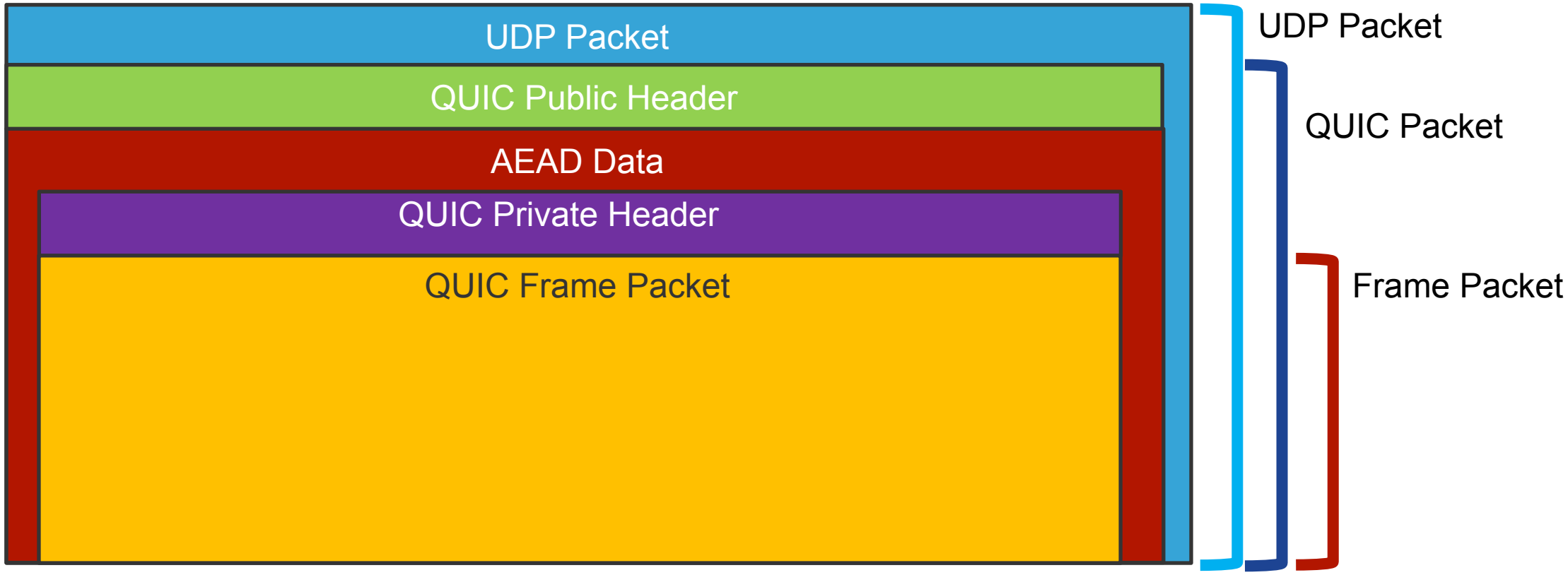
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



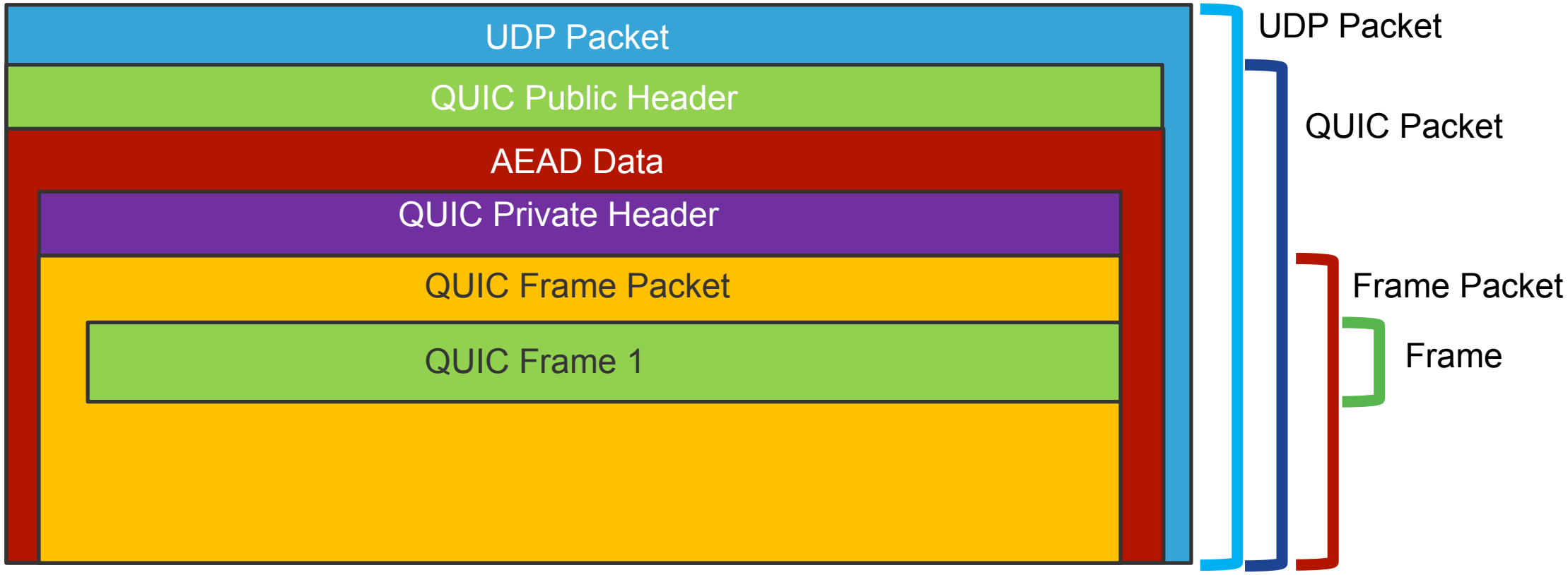
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



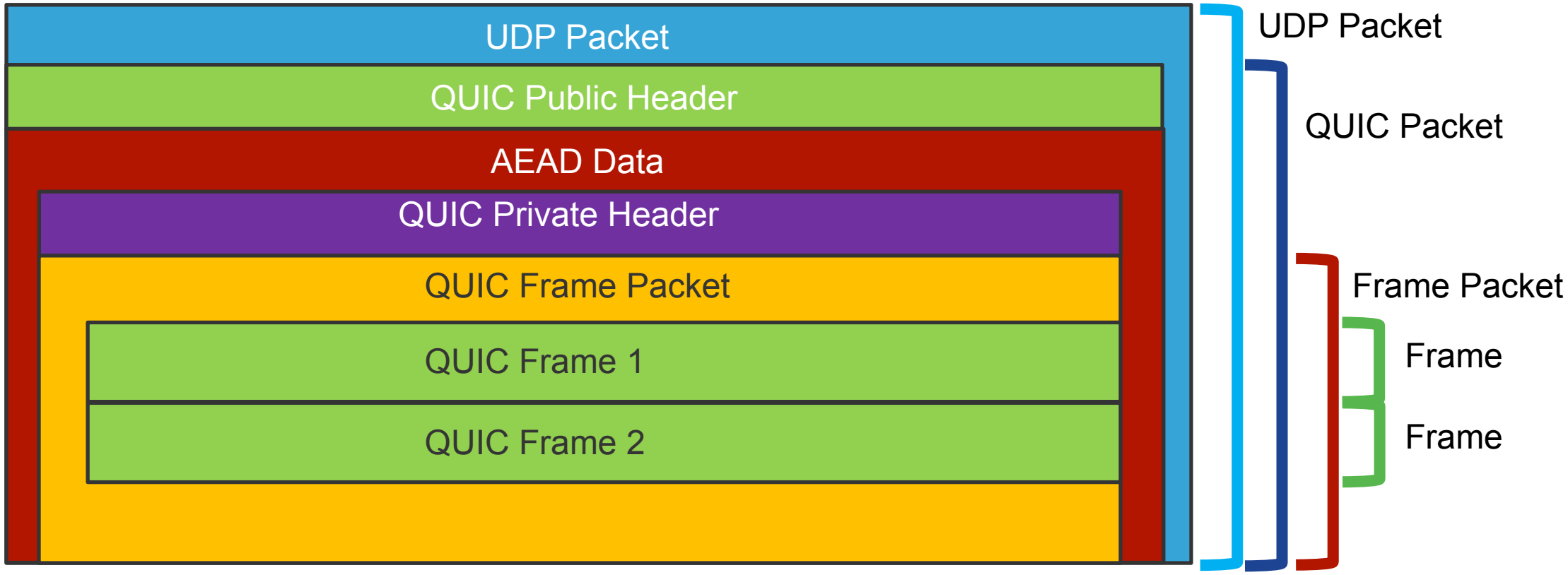
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



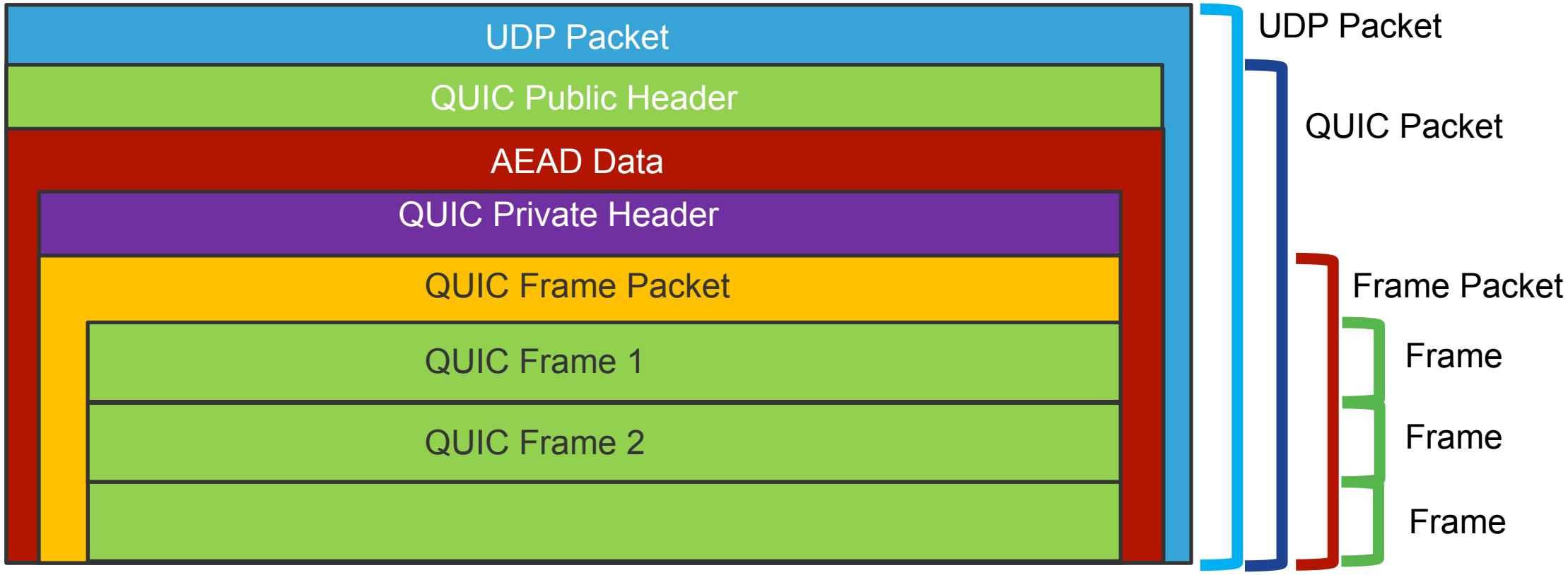
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



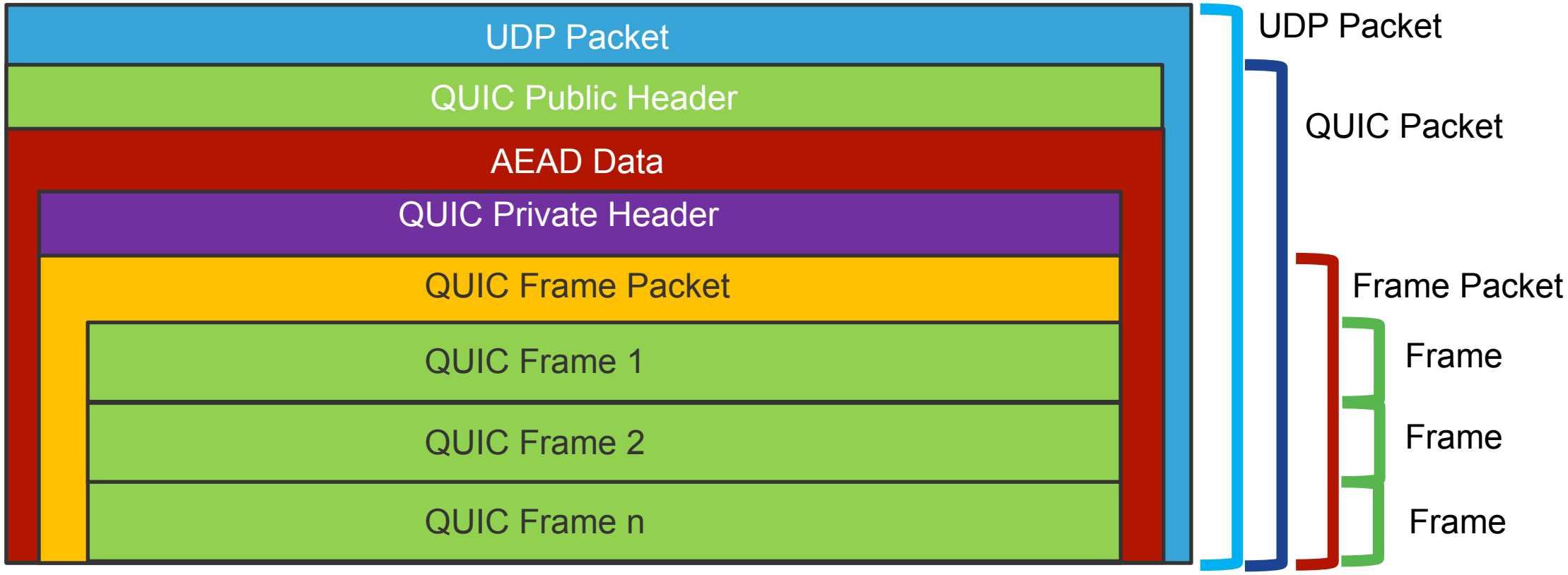
# QUIC Infodump 10%[--x----------]
# Packet Structure- Overview



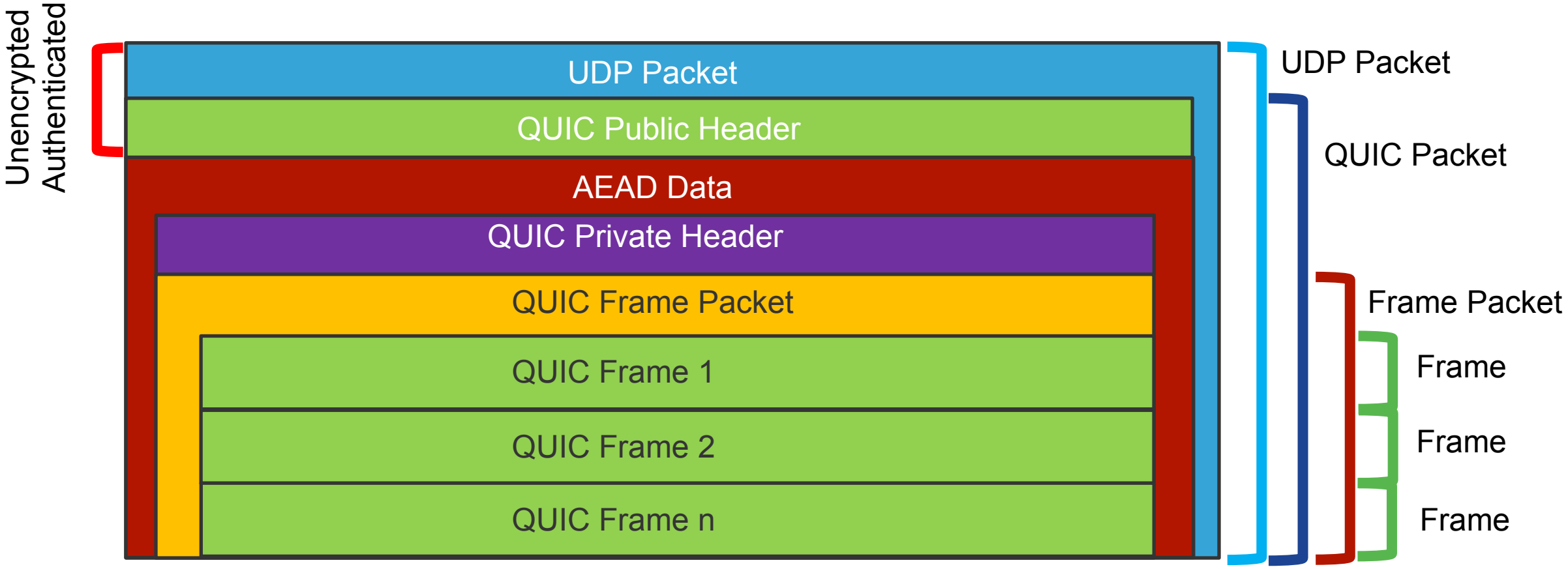
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



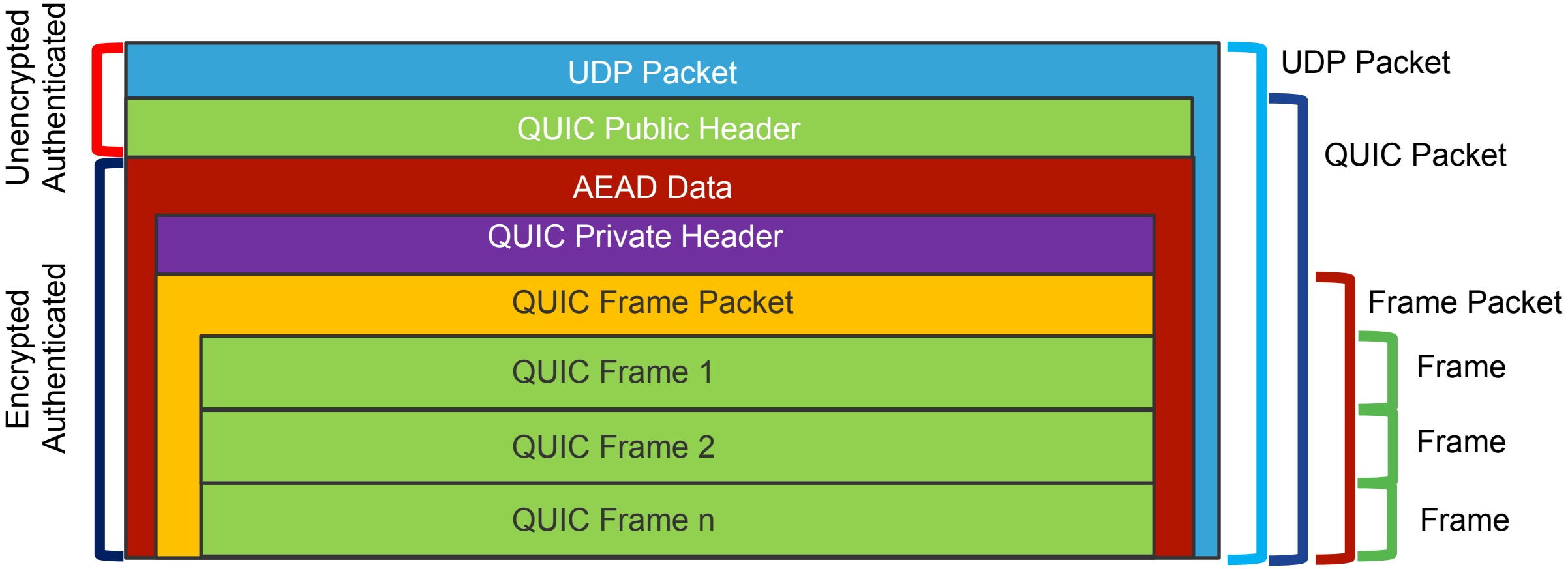
# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



# QUIC Infodump 10% [--x----------]
# Packet Structure- Overview



# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection
- ONE QUIC Packet

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection
- ONE QUIC Packet

*Contains*

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection
- ONE QUIC Packet

*Contains*

- 0-1 Frame Packets

# QUIC Infodump 14%[--x----------]
# Packet Structure- Overview

- ONE QUIC Connection

- ONE QUIC Packet

*Contains*

- 0-1 Frame Packets

*Each containing*

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection

*Contains*

- N Streams

- ONE QUIC Packet

*Contains*

- 0-1 Frame Packets

*Each containing*

- N frames

# QUIC Infodump 14% [--x----------]
# Packet Structure- Overview

- ONE QUIC Connection

*Contains*

- N Streams

- ONE QUIC Packet

*Contains*

- 0-1 Frame Packets

*Each containing*

- N frames

# QUIC Infodump 17% [---x----------]
# Packet Structure- Overview

# QUIC Infodump 17% [---x----------]
# Packet Structure- Overview

- Two primary parts to a [regular] QUIC packet

# QUIC Infodump 17% [---x----------]
# Packet Structure- Overview

- Two primary parts to a [regular] QUIC packet
  - Public Header (Authenticated, NOT Encrypted)

# QUIC Infodump 17% [---x----------]
# Packet Structure- Overview

- Two primary parts to a [regular] QUIC packet
  - Public Header (Authenticated, NOT Encrypted)
  - Private (Authenticated, Encrypted)

# QUIC Infodump 17% [---x----------]
# Packet Structure- Overview

- Two primary parts to a [regular] QUIC packet
  - Public Header (Authenticated, NOT Encrypted)
  - Private (Authenticated, Encrypted)

- AEAD (authenticated encryption and associated data) data directly after Public header, used to interpret Private data

# QUIC Infodump 25%[-----x----------] Headers

# QUIC Infodump 25% [-----x----------] Headers

- Public Header (Authenticated, NOT Encrypted)
  - Flags
  - Connection ID (Variable length, optional)
  - Packet Number (Variable length)
  - Version Header (If flag set)

# QUIC Infodump 25% [-----x----------] Headers

- Public Header (Authenticated, NOT Encrypted)
  - Flags
  - Connection ID (Variable length, optional)
  - Packet Number (Variable length)
  - Version Header (If flag set)

- Private (Authenticated, Encrypted)
  - Flags
  - FEC (Optional)
  - Frame or FEC Payload

# QUIC Infodump 40%[--------x----------] Frames

# QUIC Infodump 40% [--------x----------] Frames

- Frame Packet (contains frames)
  - Type
    - Special
    - Regular
  - N Frames

# QUIC Infodump 40% [--------x----------] Frames

- Frame Packet (contains frames)
  - Type
    - Special
    - Regular
  - N Frames
- FEC Packet
  - NULL-padded XOR of the data packets in the payload group
  - Payload group details in the Private header

# QUIC Infodump 47%[----------x---------] Frame Types

# QUIC Infodump 47%[----------x---------] Frame Types

- Control Frames
  - Padding
  - RST Stream
  - Connection Close
  - Goaway
  - Window Update
  - Blocked
  - Stop_waiting
  - Ping

# QUIC Infodump 47%[----------x---------] Frame Types

- Control Frames
  - Padding
  - RST Stream
  - Connection Close
  - Goaway
  - Window Update
  - Blocked
  - Stop_waiting
  - Ping

- Special Frames
  - Stream
  - Ack
  - Congestion feedback

- Most relevant:
  - Stream frame
  - ACK Frame

# QUIC Infodump 52% [----------x--------] Stream Frame

# QUIC Infodump 52% [----------x--------] Stream Frame

- Contained inside a frame packet

# QUIC Infodump 52% [----------x--------] Stream Frame

- Contained inside a frame packet
- Fields
  - Type (Frame header: flags about lengths of other items below)
  - Stream ID
  - Offset
  - Length (0+, optional, omission means full packet)

# QUIC Infodump 57%[----------x-------]
# ACK Frame

# QUIC Infodump 57%[----------x-------] ACK Frame

- Contained inside a frame packet

# QUIC Infodump 57%[----------x-------]
# ACK Frame

- Contained inside a frame packet
- Fields
  - Type (Frame header: flags about forms of other items below)
  - Largest Observed packet #
  - ACK Delay time
  - Timestamp Section (Used as congestion indicator)
  - Missing Packet Section (~NACK)
  - Revived Packet Section (Which did FEC revive)

# QUIC Infodump 65% [----------x------] Addressing

# QUIC Infodump 65% [----------x------] Addressing

- Connection ID
  - 64 bit, client chosen
  - Independent of Network Address
  - Source Address token (from previous connection)

# QUIC Infodump 65% [----------x------] Addressing

- Connection ID
  - 64 bit, client chosen
  - Independent of Network Address
  - Source Address token (from previous connection)
- Stream ID
  - Data flow WITHIN a connection

# QUIC Infodump 77% [----------x----]
# Cryptographic

# QUIC Infodump 77% [----------x----] Cryptographic

- Replaces, and Comparable to, TLS
  - Client reuses encryption parameters from previous connection

# QUIC Infodump 77% [----------x----] Cryptographic

- Replaces, and Comparable to, TLS
  - Client reuses encryption parameters from previous connection

- IP Spoofing protection
  - Requires an IP-validated Source Address Token, or negotiates a new one (0/1 RTT)
  - No trivial reflection attacks as in NTP/DNS

# QUIC Infodump 84%[----------x---]
# Forward Error Correction

# QUIC Infodump 84%[----------x---]
# Forward Error Correction

- XOR of all packets in FEC block

# QUIC Infodump 84%[----------x---]
# Forward Error Correction

- XOR of all packets in FEC block
- FEC block size is variable

# QUIC Infodump 84%[----------x---]
# Forward Error Correction

- XOR of all packets in FEC block
- FEC block size is variable
- If “revived by FEC” then indicate this ID as NACK and REVIVED

# QUIC Infodump 93%[----------x]
# Connection Management

# QUIC Infodump 93%[----------x]
# Connection Management

- All connections assumed to be left in open state until user leaves page

# QUIC Infodump 93%[----------x]
# Connection Management

- All connections assumed to be left in open state until user leaves page
- Address mobility supported

# QUIC Infodump 93%[----------x]
# Connection Management

- All connections assumed to be left in open state until user leaves page
- Address mobility supported
- Multipath planned, but not yet there

# QUIC Infodump 99% [----------x]
# HTTP/2 Integration

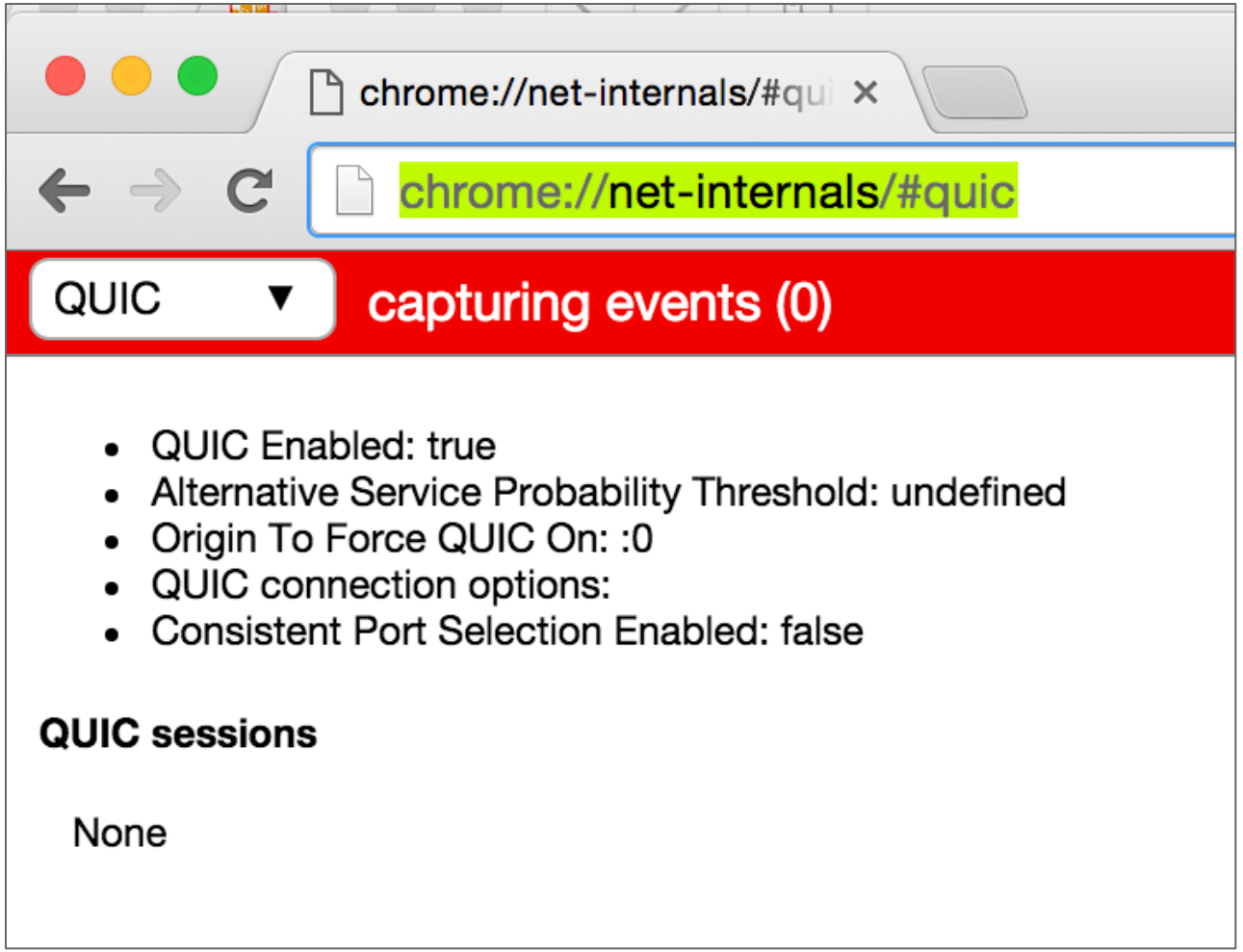
- QUIC intends to handle most HTTP/2 stream management

# QUIC Infodump 100% [ Finishing… ]
# Network Requirements

- UDP Unicast unimpeded
  - No filters
  - No Rate Limiters
- UDP NAT good practice
  - E.g. Timeouts set reasonably

# Debugging UIC

- Chrome:

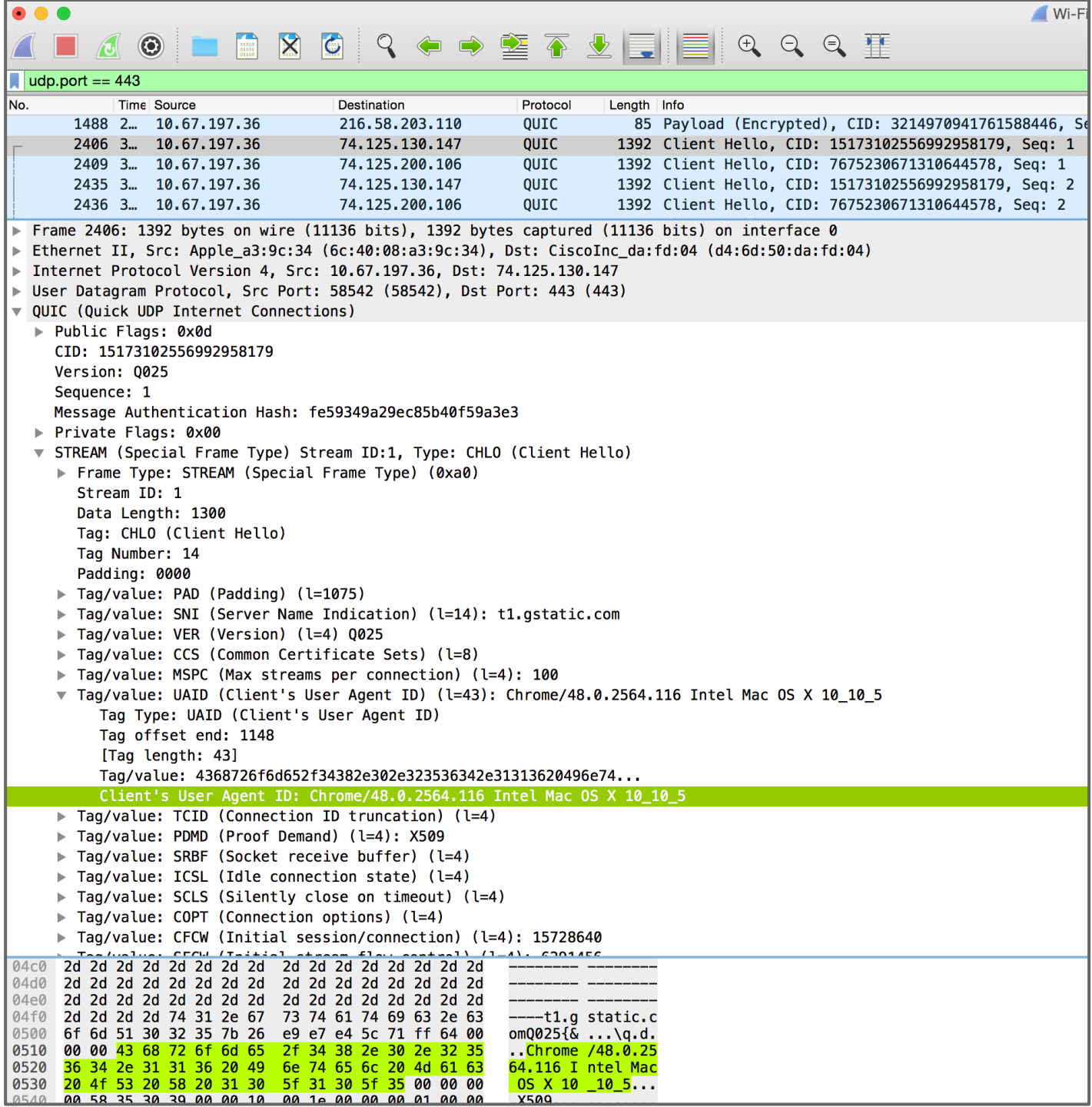



- chrome://net-internals/#quic

# Debugging QUIC

- Wireshark:

QUIC dissector =>



Introduction ✔
Background (Why Change TCP) ✔
MPTCP ✔
Background Redux (Why NOT TO change TCP) ✔
QUIC ✔

# Implications

Conclusion and Takeaways

# Was anyone thinking about the network environment?

# Was anyone thinking about the network environment?

- The security and stability of MPTCP itself

# Was anyone thinking about the network environment?

- The security and stability of MPTCP itself

# Was anyone thinking about the network environment?

- The security and stability of MPTCP itself

- What changes like this *could mean* for network security

# Was anyone thinking about the network environment?

- The security and stability of MPTCP itself

- What changes like this *could mean* for network security

… not so much

# Was anyone thinking about the network environment?

- T
- W
  ne

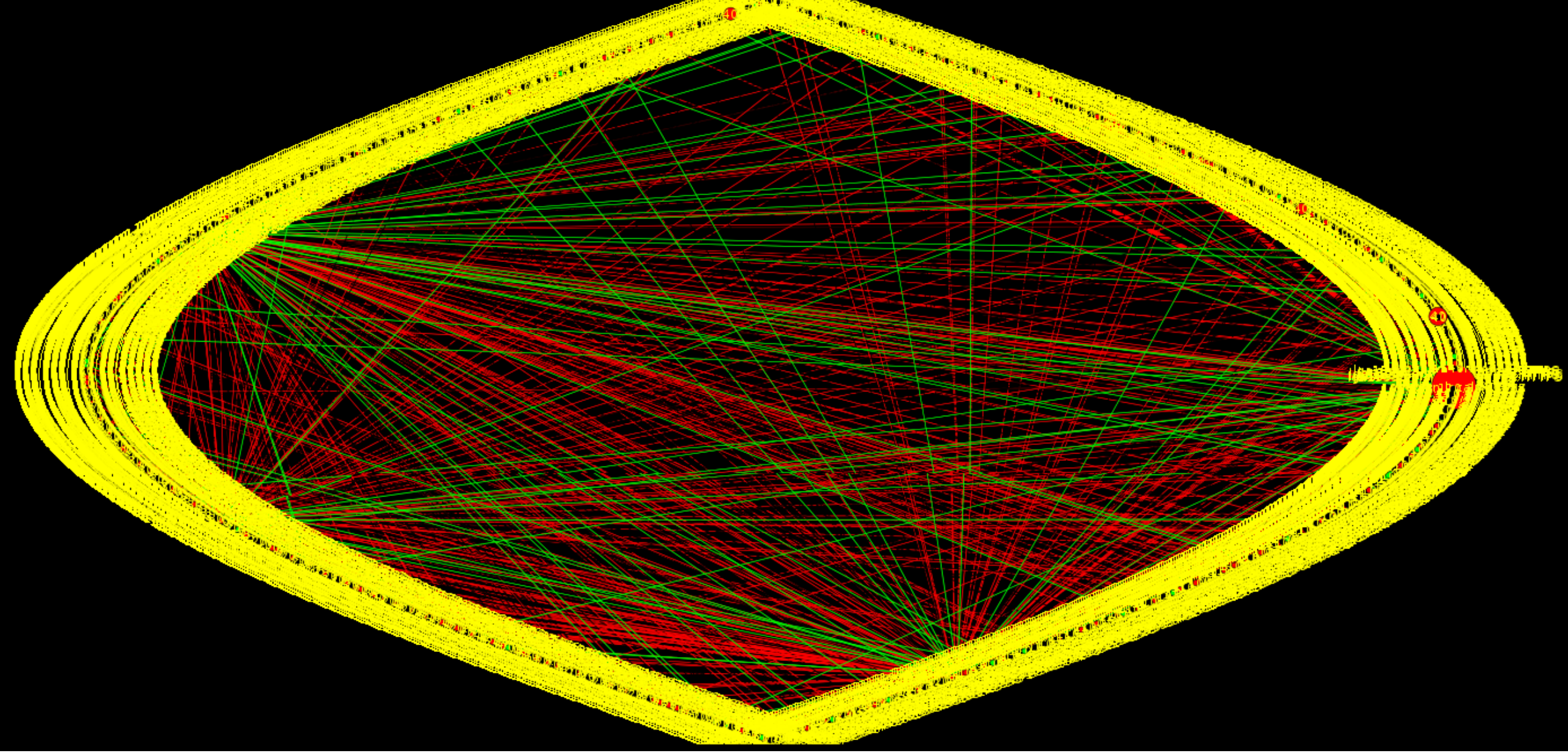
QUIC put more work into this

- Google also operates the network
- But they also put in "Protection" against network operator interference

ot so much

# What does it look like?

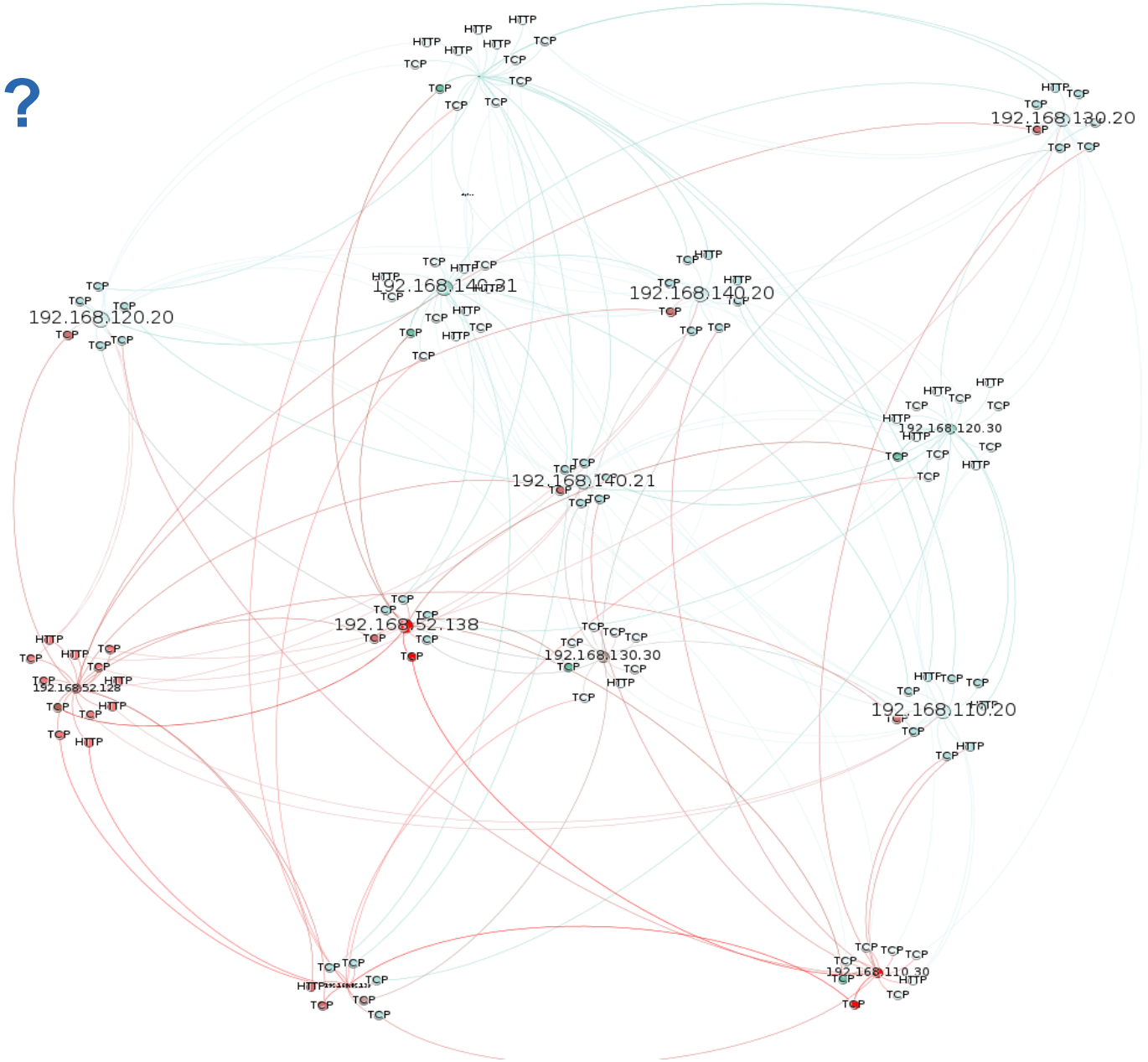
- On the network: If you don’t understand
- …



Each yellow blob is actually part of an address label

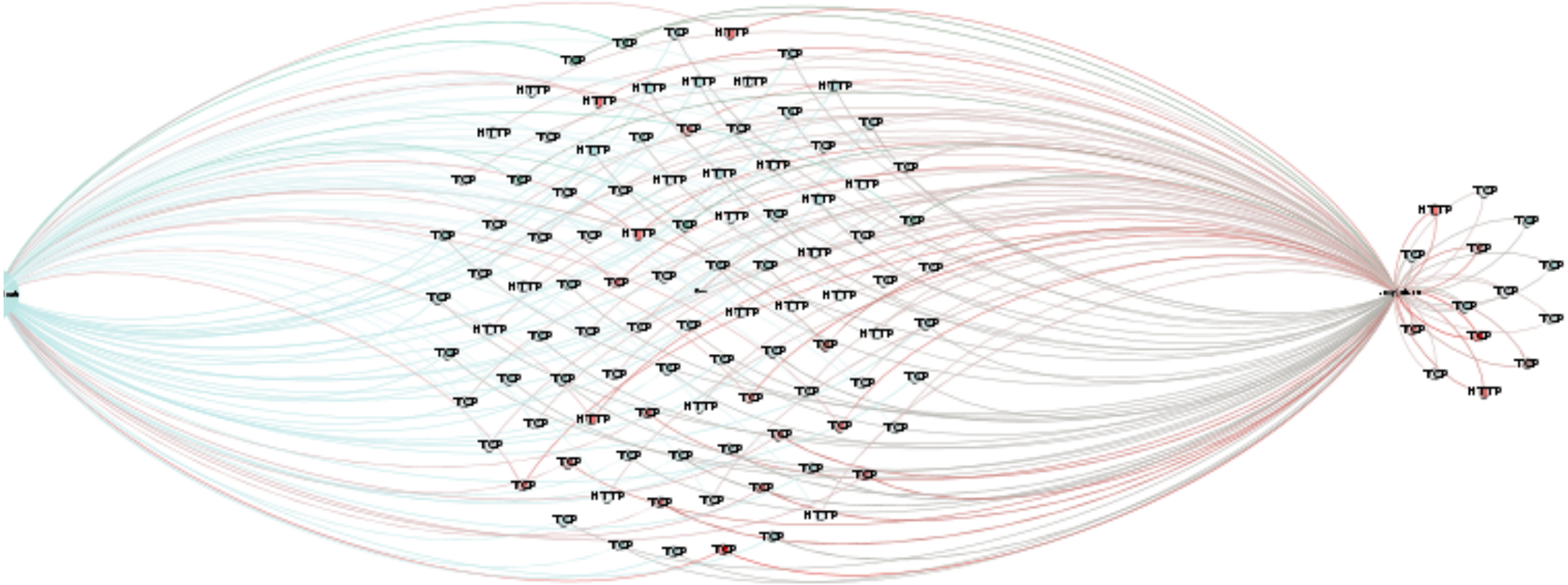
# What does it look like?

- On the network: If you don't understand it, but you cluster IPs



# What does it look like?

- On the network: If you do understand



- But you can only do this when you can see & correlate **all** related flows…

# MPTCP and … Network Management

# MPTCP and … Network Management

- These protocols are agnostic of IPv4 and IPv6, happily use both

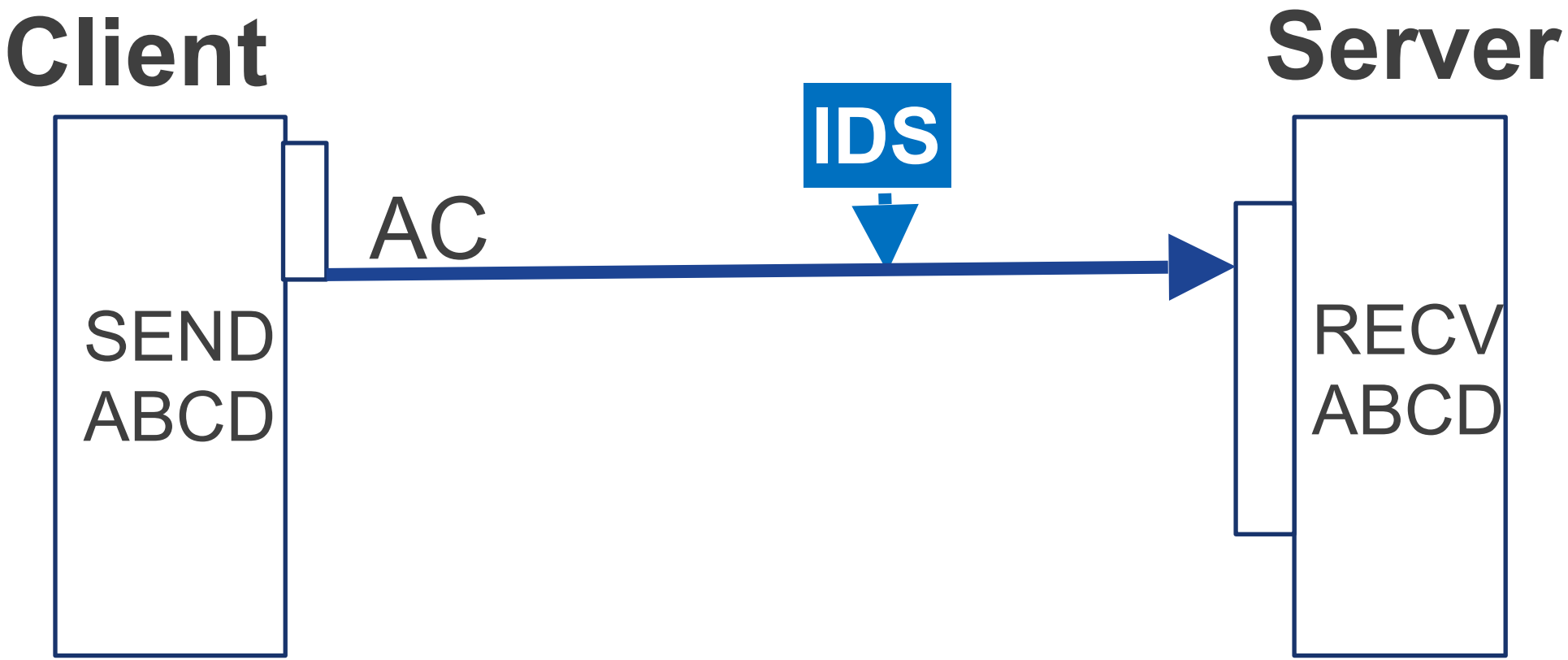
# MPTCP and … Network Management

- These protocols are agnostic of IPv4 and IPv6, happily use both
- If tool doesn't understand MPTCP, flows look like unrelated TCP streams

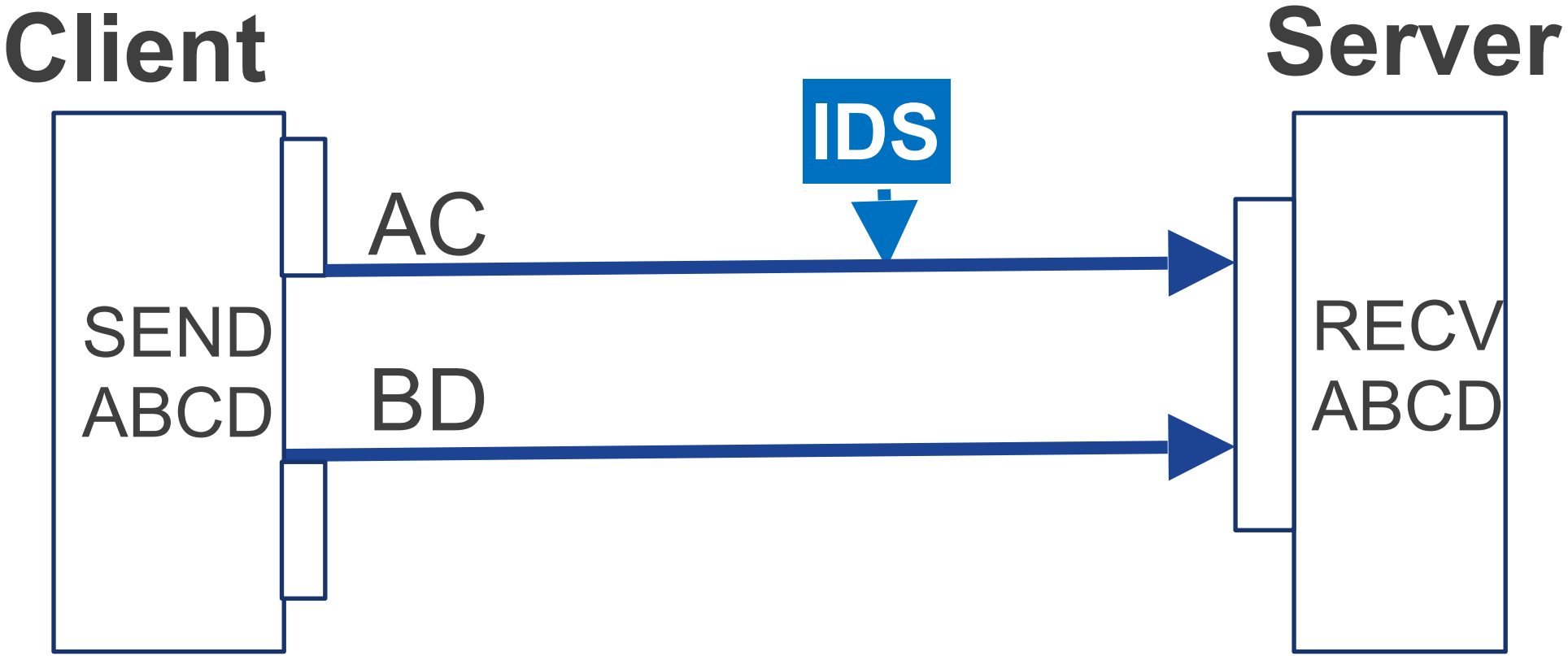
# MPTCP and … Network Management

- These protocols are agnostic of IPv4 and IPv6, happily use both
- If tool doesn't understand MPTCP, flows look like unrelated TCP streams
- If tool doesn't understand QUIC, flows look like unrelated UDP flows

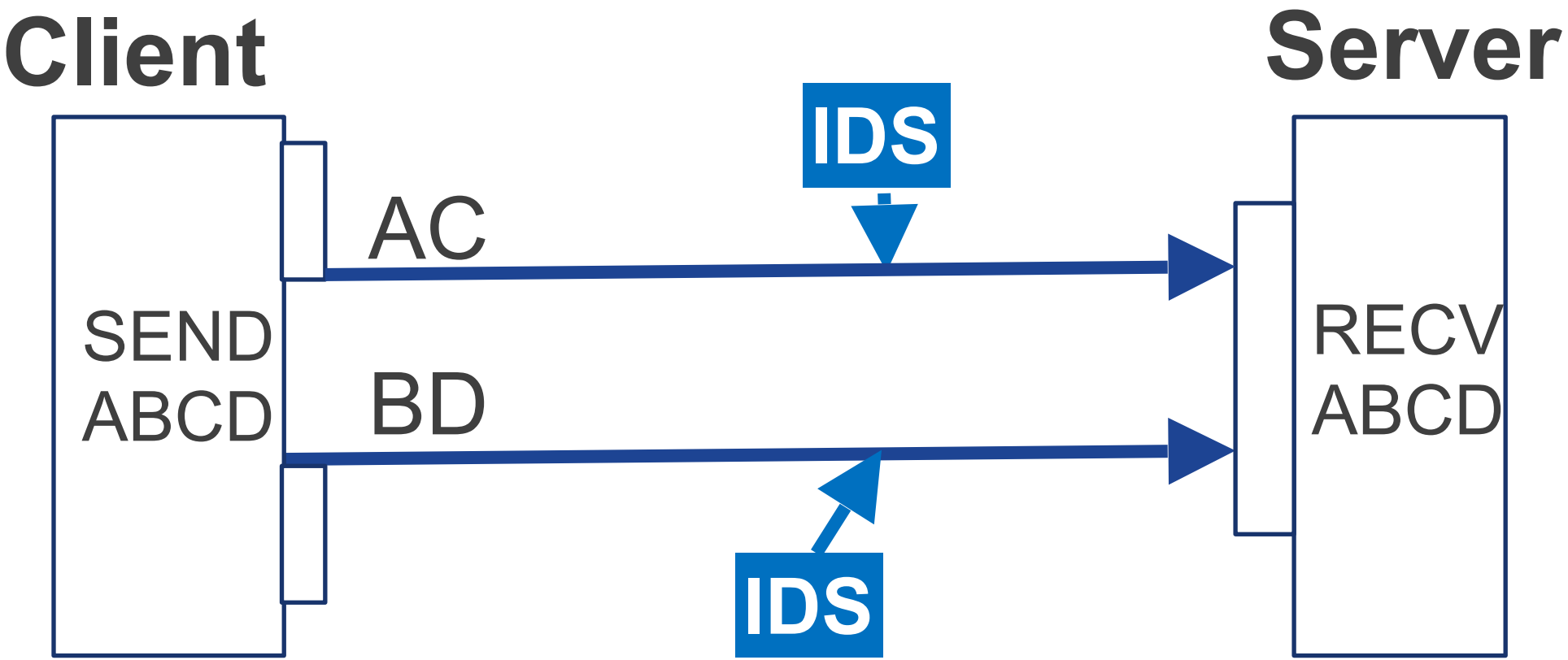
# Monitoring – Cross-path fragmentation



# Monitoring – Cross-path fragmentation



# Monitoring – Cross-path fragmentation



# Multipath and … Network monitoring

# Multipath and … Network monitoring

- How would your CGNAT handle an exponential increase in connections as well as clients?

# Multipath and … Network monitoring

- How would your CGNAT handle an exponential increase in connections as well as clients?

- Two devices:
  - One interface each: 1 flow
  - Four interfaces each: 16 flows

# Multipath and … Network monitoring

- How would your CGNAT handle an exponential increase in connections as well as clients?

- Two devices:
  - One interface each: 1 flow
  - Four interfaces each: 16 flows

- What about virtual interfaces, VPNs, proxies, load balancers..

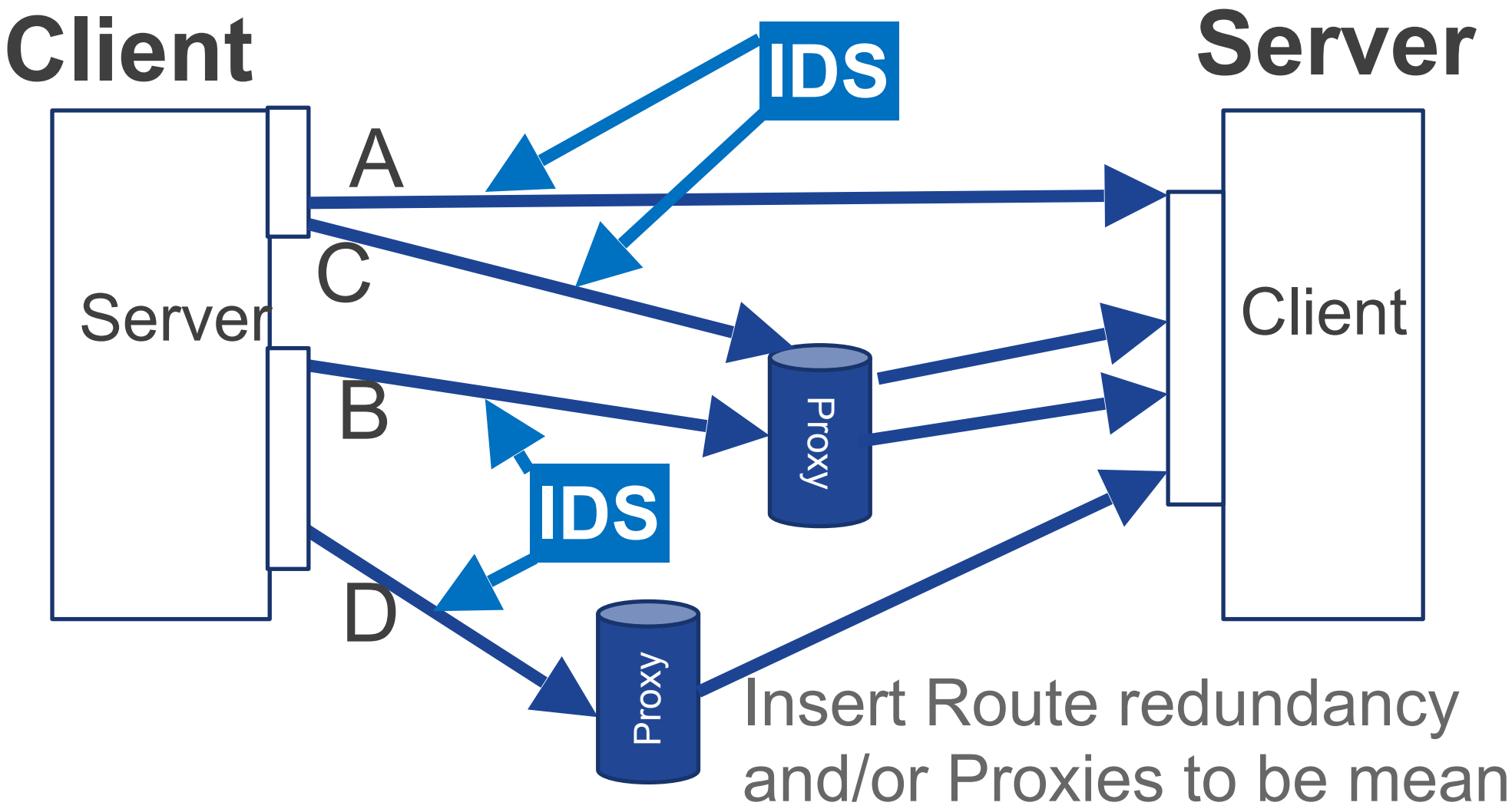
## Monitoring – Combined Cross-path and Cross-Stream fragmentation


Insert Route redundancy and/or Proxies to be mean

## Monitoring – Combined Cross-path and Cross-Stream fragmentation


Insert Route redundancy and/or Proxies to be mean

## Monitoring – Combined Cross-path and Cross-Stream fragmentation




Insert Route redundancy and/or Proxies to be mean

# Multipath and … Network monitoring

- How would your CGNAT handle an [illegible] increase in connections as [illegible]
- [illegible]
- What about virtu[illegible] proxies, load balancers..

*In trying to solve address exhaustion with NAT do we risk state processing exhaustion?*

# Because of these…

# Because of these…

- Cross-path

# Because of these…

- Cross-path
- Moving target

# Because of these…

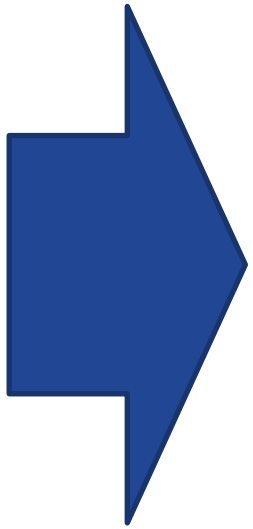
- Cross-path
- Moving target
- Connection resilience

# Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections

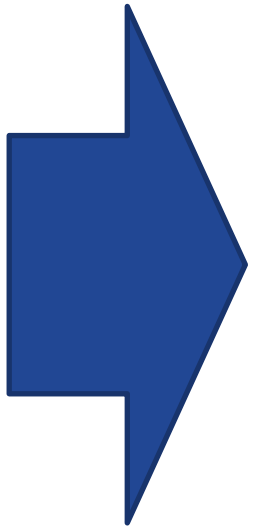
# Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections



# Because of these…

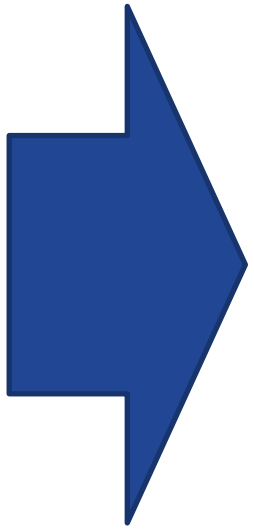
- Cross-path
- Moving target
- Connection resilience
- Reverse connections



## … if your approach to network management requires *any* of these…

# Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections

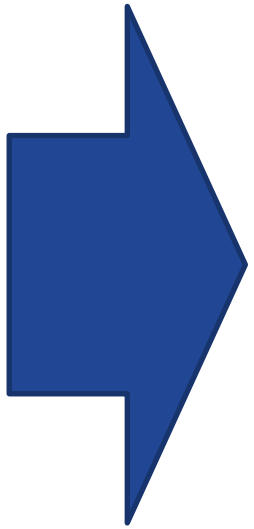


## … if your approach to network management requires *any* of these…

- See all app layer data in a TCP stream

# Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections

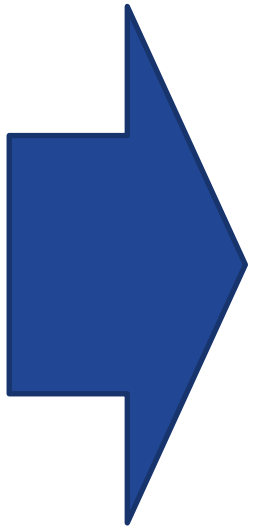


**… if your approach to network management requires *any* of these…**

- See all app layer data in a TCP stream
- Associate logical sessions to IP addresses

# Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections



## … if your approach to network management requires *any* of these…

- See all app layer data in a TCP stream
- Associate logical sessions to IP addresses
- Tamper with or close "bad" connections mid-stream

# Because of these…

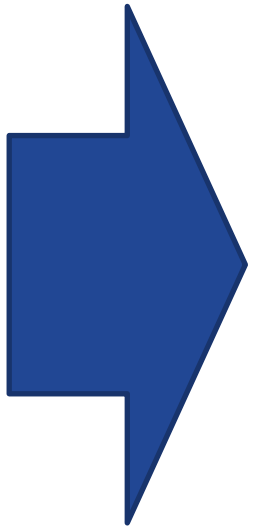
- Cross-path
- Moving target
- Connection resilience
- Reverse connections

## … if your approach to network management requires *any* of these…

- See all app layer data in a TCP stream
- Associate logical sessions to IP addresses
- Tamper with or close "bad" connections mid-stream
- Differentiate clients from servers based on connection direction

## Because of these…

- Cross-path
- Moving target
- Connection resilience
- Reverse connections



## … if your approach to network management requires *any* of these…

- See all app layer data in a TCP stream
- Associate logical sessions to IP addresses
- Tamper with or close "bad" connections mid-stream
- Differentiate clients from servers based on connection direction

## …then something is probably going to break

# Key Effects

# Key Effects

- Cross-path traffic fragmentation
  - That's the whole point!

# Key Effects

- Cross-path traffic fragmentation
  - That's the whole point!

- Moving target
  - Ability to change source and destination addresses in the middle of a connection

# Key Effects

- Cross-path traffic fragmentation
  - That's the whole point!

- Moving target
  - Ability to change source and destination addresses in the middle of a connection

# Key Effects

- Cross-path traffic fragmentation
  - That's the whole point!
- Moving target
  - Ability to change source and destination addresses in the middle of a connection
- Connection resilience
  - Has additional checksums that require capture of the initial packet to reliably fake
  - Until every subflow is dead the overall connection keeps going

# Key Effects

- Cross-path traffic fragmentation
  - That's the whole point!
- Moving target
  - Ability to change source and destination addresses in the middle of a connection
- Connection resilience
  - Has additional checksums that require capture of the initial packet to reliably fake
  - Until every subflow is dead the overall connection keeps going
- "Reverse" connections

# Key Effects

# Key Effects

A ~~few~~ lot of slides back…

- The **packet *sender* decides** which data goes down which path.

*“The Net interprets censorship as damage and routes around it.”*

John Gilmore, 1993

*“Multipath interprets modification as damage and routes around it.”*

Kate Pearce, 2014

# Key Effects

# Key Effects

A ~~few~~ lot of slides back…

- The **packet *sender* decides** which data goes down which path.

- Normal/benign clients won't choose pathological fragmentation schemes

# Key Effects

A ~~few~~ lot of slides back…

- The **packet *sender* decides** which data goes down which path.

- Normal/benign clients won't choose pathological fragmentation schemes

# MPTCP and … Firewalls

# MPTCP and … Firewalls

- MPTCP has address advertisement
  - This changes things for perimeters

# MPTCP and … Firewalls

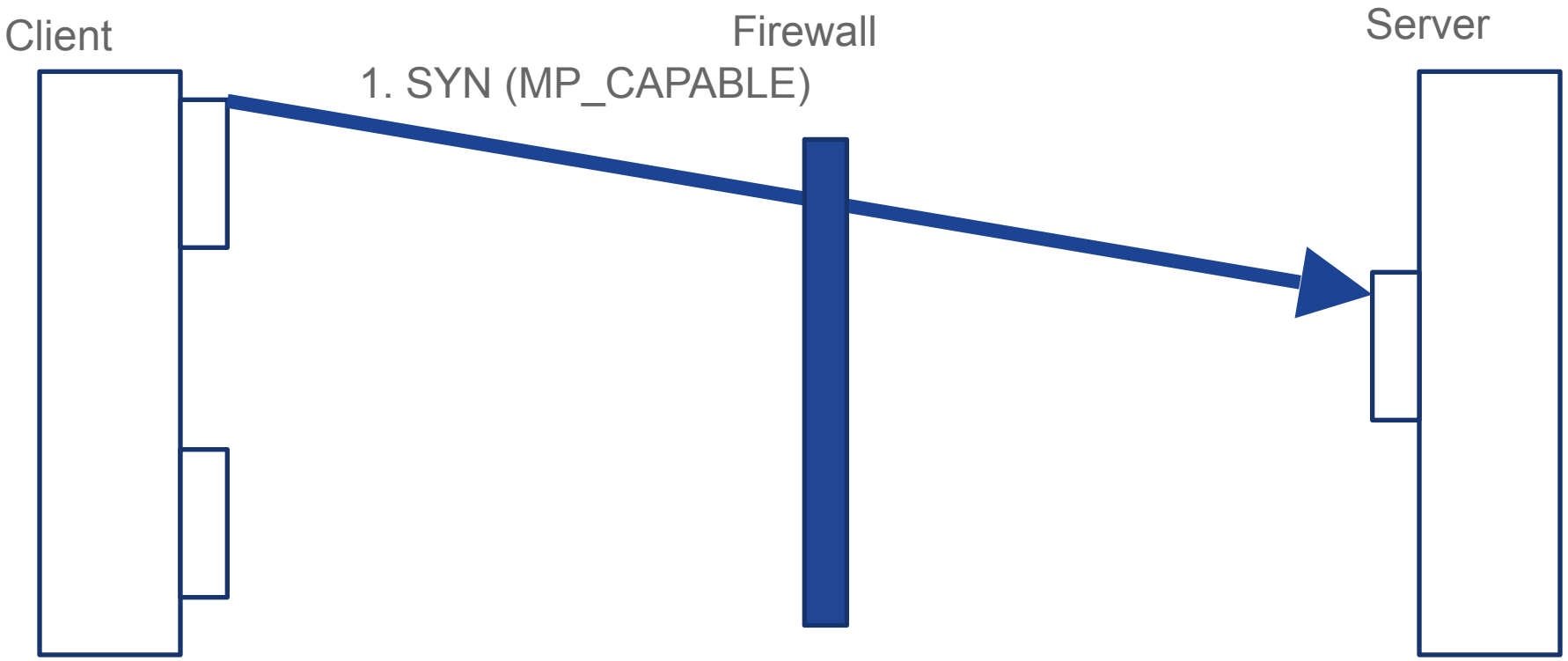
- MPTCP has address advertisement
  - This changes things for perimeters

# MPTCP and … Firewalls

- MPTCP has address advertisement
  - This changes things for perimeters

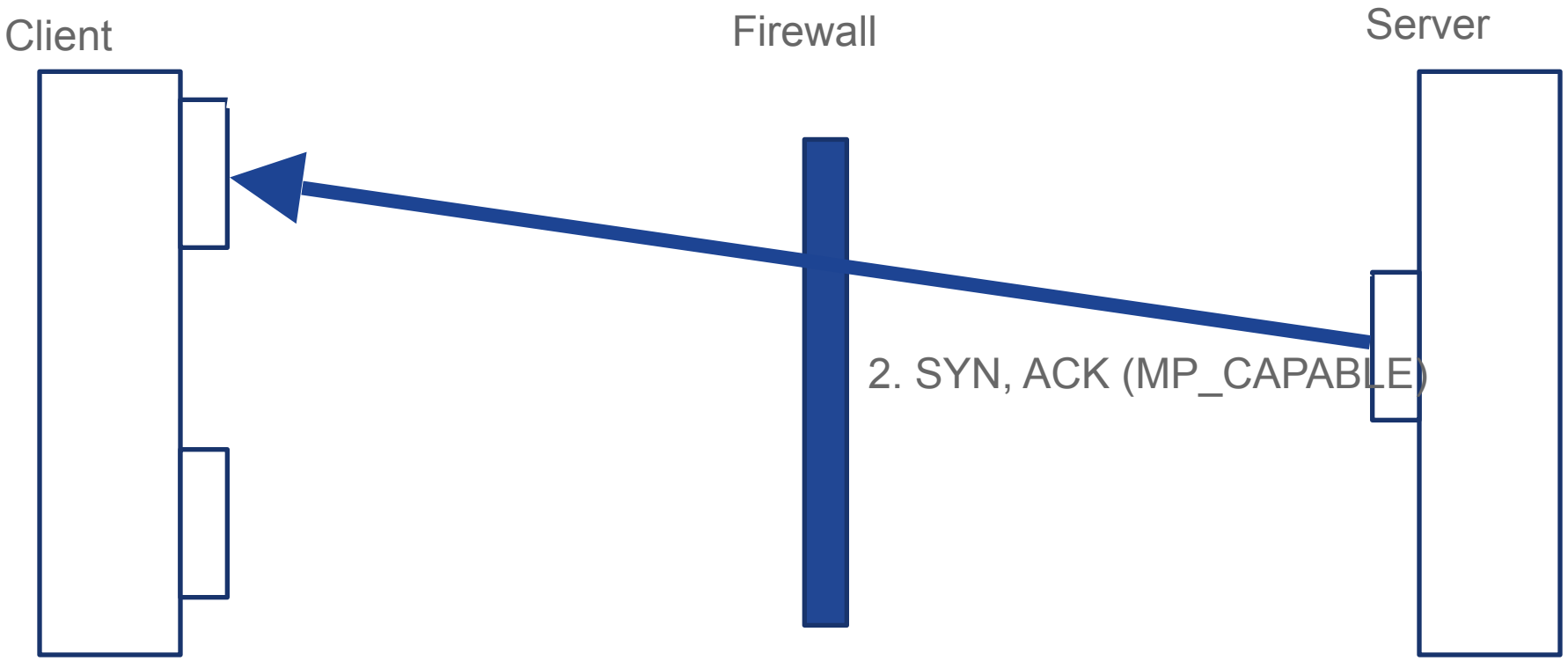
- How’d you like an outbound incoming connection?

# MPTCP and … Firewalls



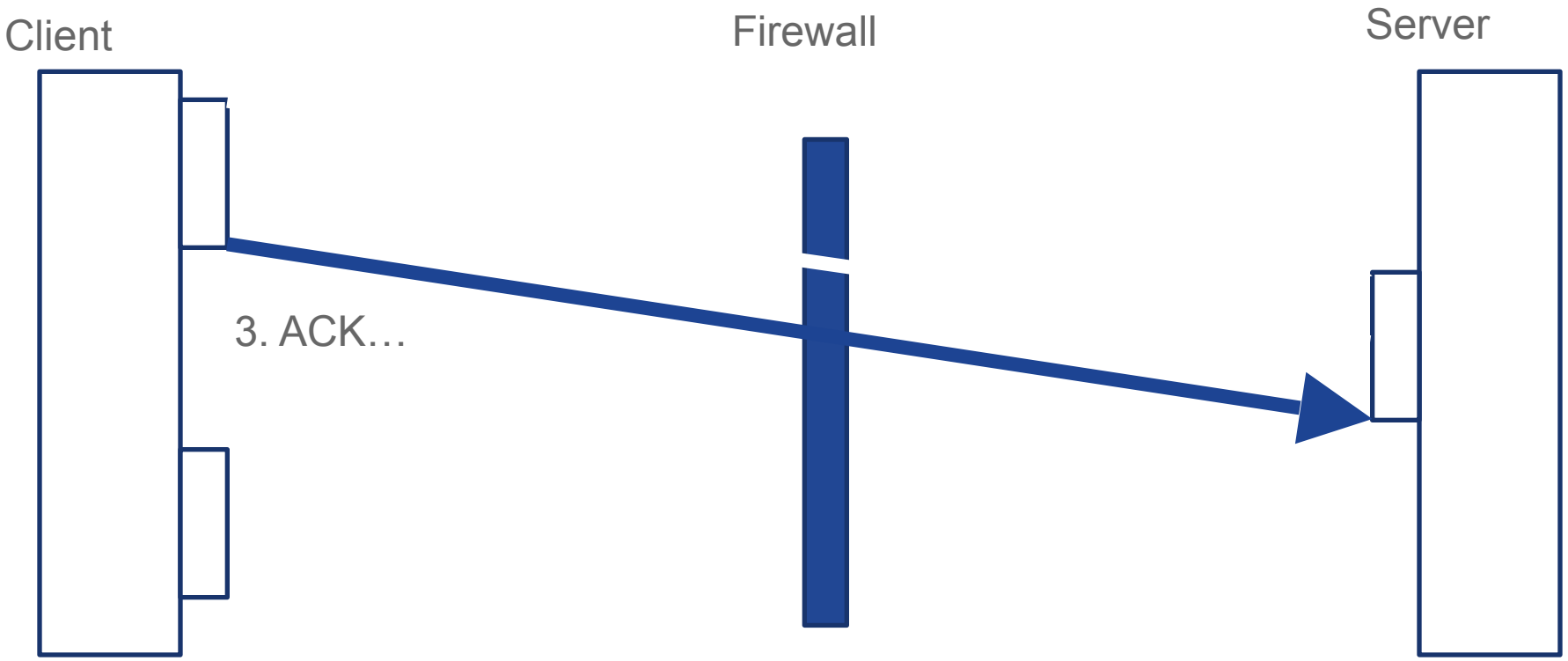

- MPTCP connection looks like TCP so far…

# MPTCP and … Firewalls



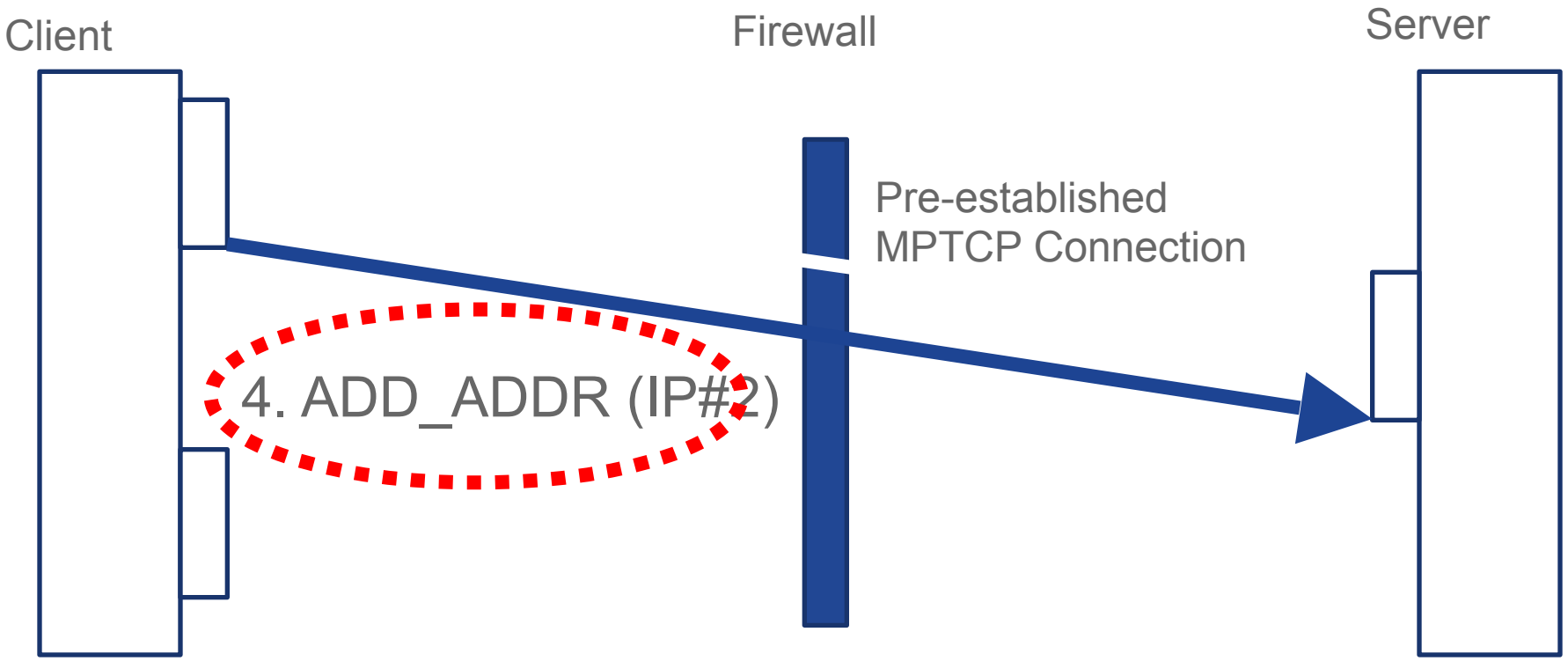

- Still seems pretty standard, albeit with extra TCP OPTIONS

# MPTCP and … Firewalls



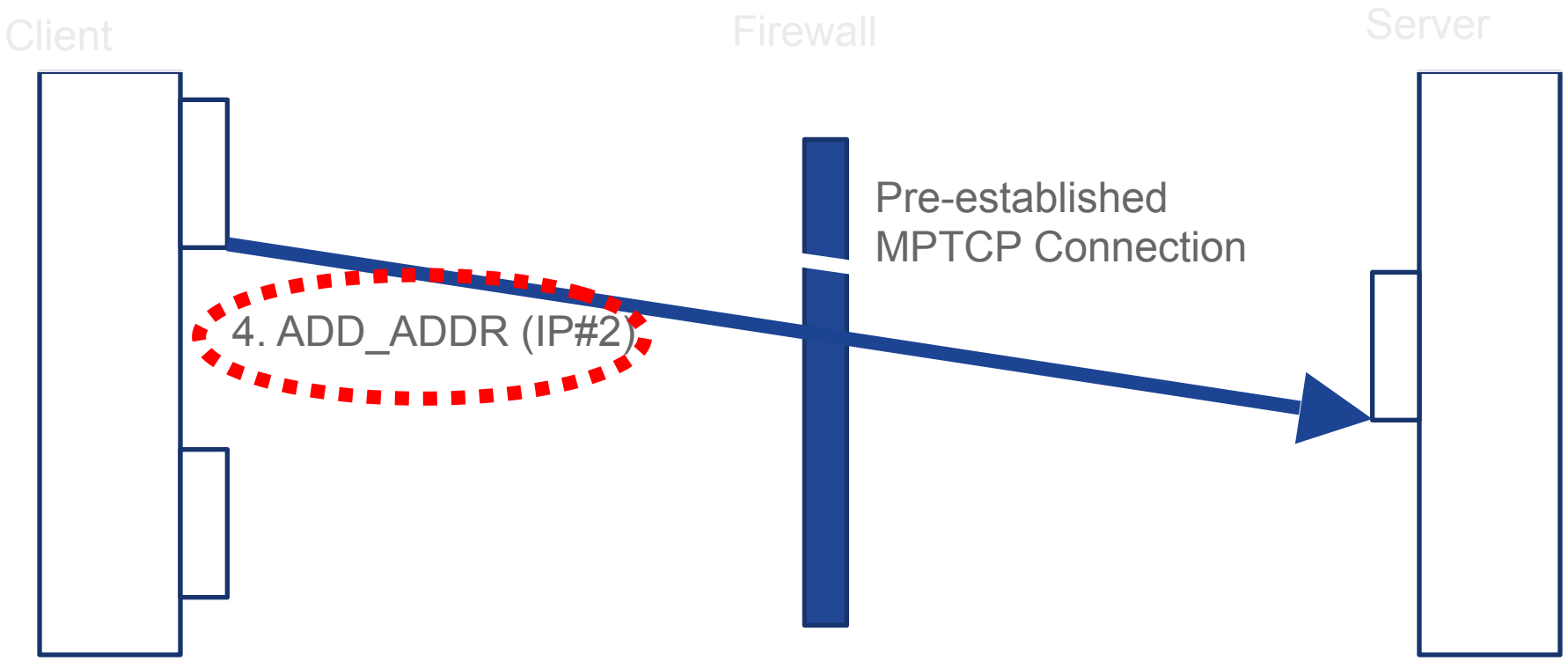

- OK, so it’s a TCP connection with an additional options… so what?

# MPTCP and … Firewalls



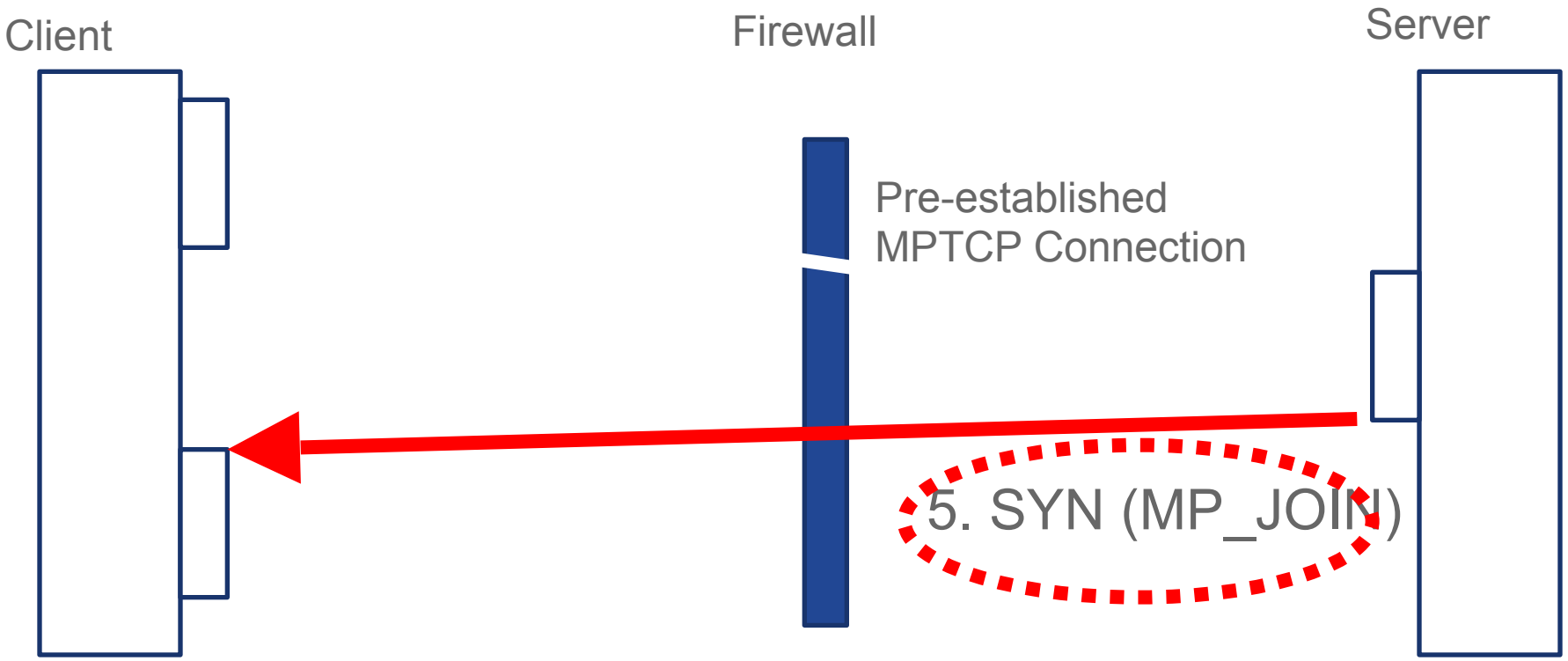

- Well, what if the client tells the server about a new address?

# MPTCP and … Firewalls

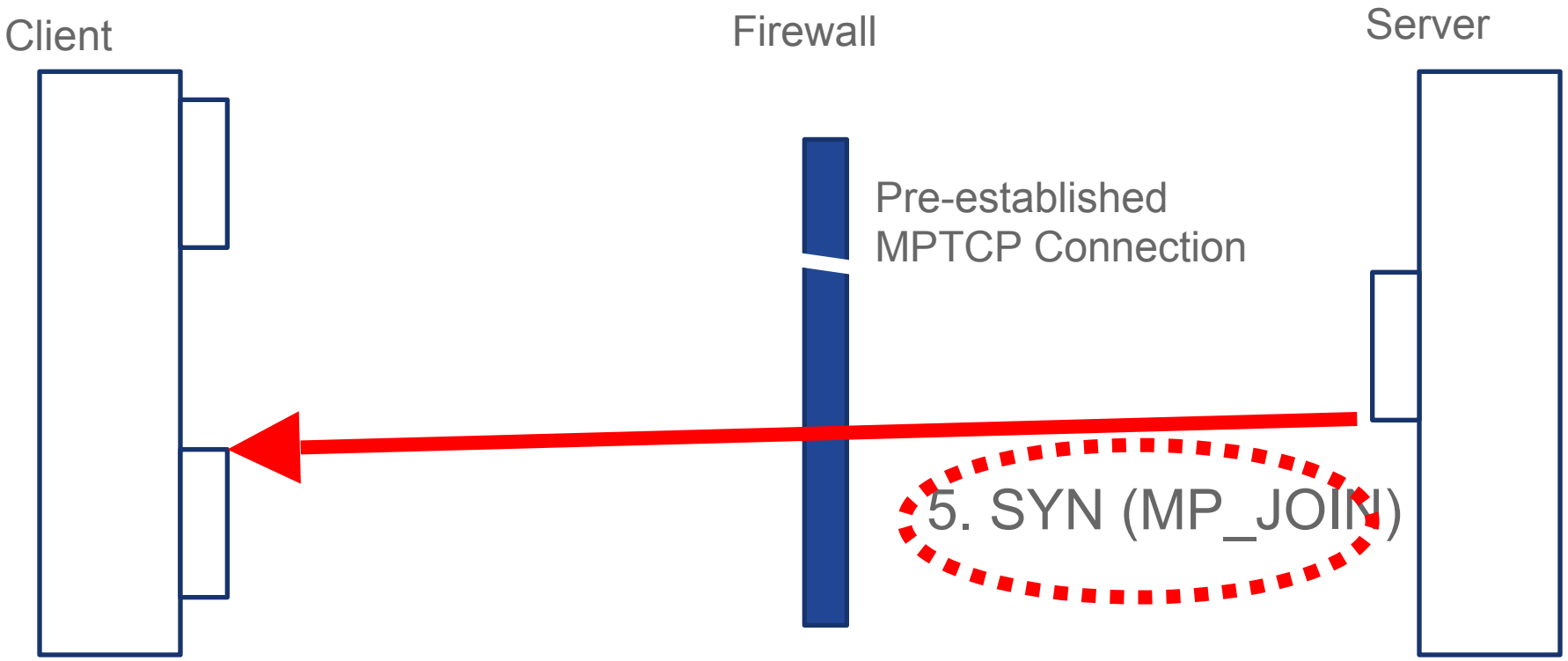



- Now, the “Internal” host <u>may</u> set up a connection to the advertised address

# MPTCP and … Firewalls



# MPTCP and … Firewalls




- Is this new connection incoming or outgoing?

# MPTCP and … Firewalls


- Is this new connection incoming or outgoing?

# Multipath and … Privacy

# Multipath and … Privacy

- Multipath shifts power towards endpoints, and away from infrastructure & ISP's

# Multipath and … Privacy

- Multipath shifts power towards endpoints, and away from infrastructure & ISP's
- I don't fully trust my ISP or Cellular company…

# Multipath and … Privacy

- Multipath shifts power towards endpoints, and away from infrastructure & ISP's
- I don't fully trust my ISP or Cellular company…
- But they probably don't trust each other either!

Introduction ✔
Background (Why Change TCP) ✔
MPTCP ✔
Background Redux (Why NOT TO change TCP) ✔
QUIC ✔
Implications ✔

# Conclusion and Takeaways

# So… things people took away from my MPTCP work

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)
- It’s a problem that only exists because of MPTCP

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)
- It’s a problem that only exists because of MPTCP
- NO

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)
- It’s a problem that only exists because of MPTCP
- NO
- NO

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)
- It’s a problem that only exists because of MPTCP
- NO
- NO
- NO

# So… things people took away from my MPTCP work

- It’s a problem of tools (NO)
- It’s a problem that only exists because of MPTCP
- NO
  - NO
  - NO
- NO
  - NO
  - NO
- NO
  - NO
  - NO
- NO
  - NO
  - NO

# Key Insights- Multipath

- The (IP) network's job is to transmit data from end to end. (?)

*“Within each network,* ***communication may be disrupted due to unrecoverable mutation of the data*** *or missing data.* ***End-to-end restoration procedures are desirable to allow complete recovery from these conditions****.”*

[CERF, VINTON G., and ROBERT E. KAHN. "A Protocol for Packet Network Intercommunication." (1974).]

# Key Insights – Content Inspection

# Key Insights – Content Inspection

Many network management approaches rely too much on

# Key Insights – Content Inspection

Many network management approaches rely too much on

- Metadata

# Key Insights – Content Inspection

Many network management approaches rely too much on

- Metadata
- Data inspection

# Key Insights – Content Inspection

Many network management approaches rely too much on

- Metadata
- Data inspection
- Context Inspection

# Key Insights – Content Inspection

Many network management approaches rely too much on

- Metadata
- Data inspection
- Context Inspection

The Internet wasn't designed to validate your control over network data flows.

# Technical Takeaways

# Technical Takeaways

- Address != connection endpoint

# Technical Takeaways

- Address != connection endpoint
- State exhaustion could become the new address exhaustion

# Technical Takeaways

- Address != connection endpoint
- State exhaustion could become the new address exhaustion
- Multipath tech breaks assumptions many didn’t even know we were making

# Conclusions

# Conclusions

- Multipath communications are awesome, and they're coming

# Conclusions

- Multipath communications are awesome, and they're coming
- Multipath communication confounds business & security models relying on stateful inspection or lookup

# Conclusions

- Multipath communications are awesome, and they're coming
- Multipath communication confounds business & security models relying on stateful inspection or lookup
- Now is the time for network security and network management to prepare

# References and Links

- MPTCP:

- Raiciu, C. et al., 2012. How hard can it be? designing and implementing a deployable multipath TCP. NSDI, (1). Available at: https://www.usenix.org/system/files/conference/nsdi12/nsdi12-final125.pdf.
- ACM Queue - Multipath TCP, Decoupled from IP, TCP is at last able to support multihomed hosts - Christoph Paasch and Olivier Bonaventure, UCL - http://queue.acm.org/detail.cfm?id=2591369
- IETF Working group - http://datatracker.ietf.org/wg/mptcp/

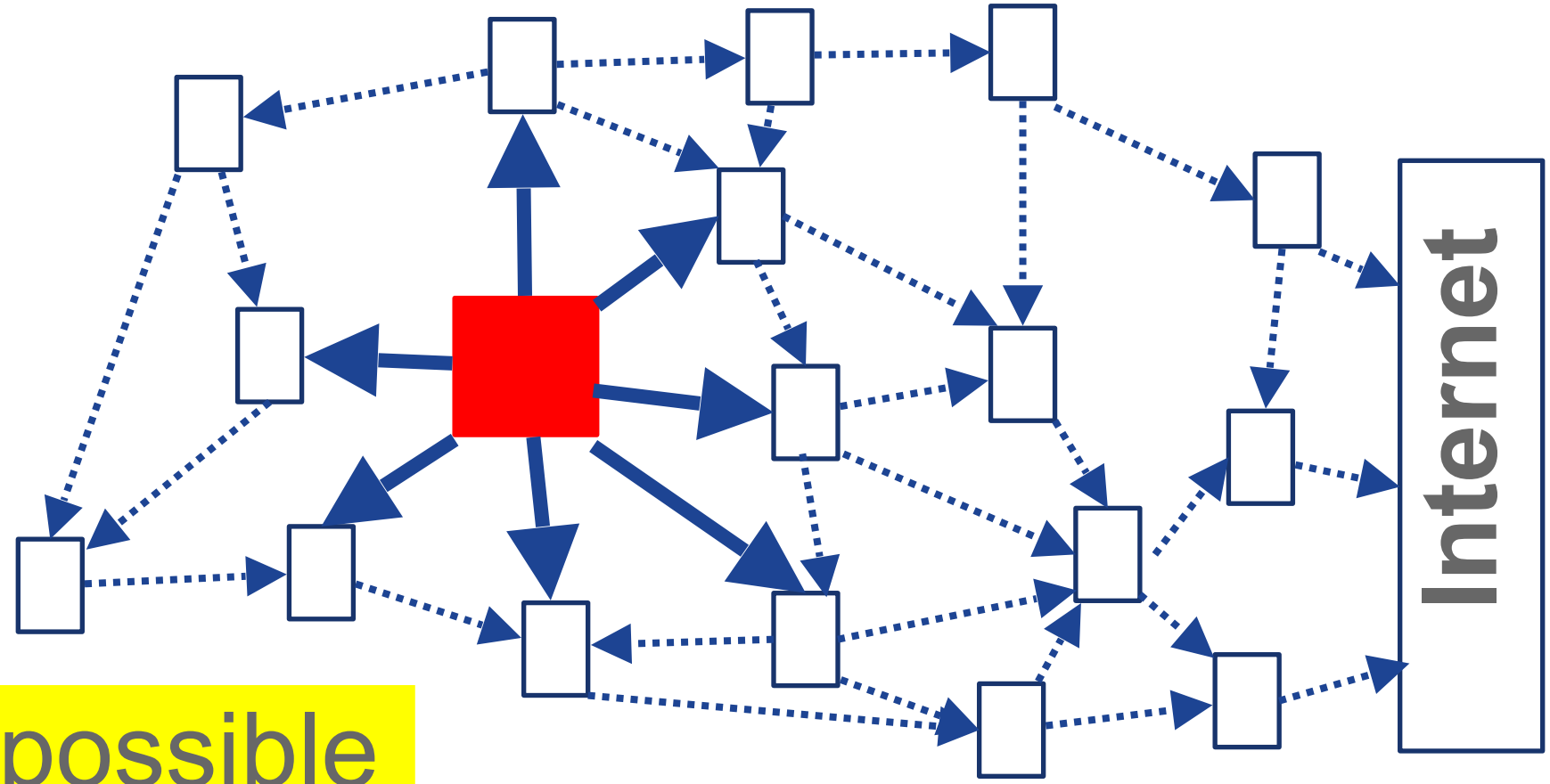
- QUIC

- https://peering.google.com/#/learn-more/quic
- https://www.chromium.org/quic
- https://tools.ietf.org/html/draft-tsvwg-quic-protocol-02
- https://tools.ietf.org/html/draft-tsvwg-quic-loss-recovery-01

# Questions?

Catherine (Kate) Pearce
Twitter: @secvalve
katpearc@cisco.com

# Path Management – mesh networks




All possible paths can be used ***at the same time***