

Multipath TCP

Breaking today's networks
with tomorrow's protocol

Speakers - Who are we?

- Catherine (Kate) Pearce
 - Security Consultant / Pentester
 - Loves her wine the way she likes her RFCs (Dry)
 - New Zealand transplant

- Patrick Thomas
 - Senior Security Consultant / Pentester
 - Application Security focus





MPTCP changes
fundamental assumptions
about
*how TCP works**

*Use it to break
things today*

*Adapt to it for
tomorrow*

 *Well... kinda

Not Layer 4?
Totally the same.

Layer 4?
Buckle Up.

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]

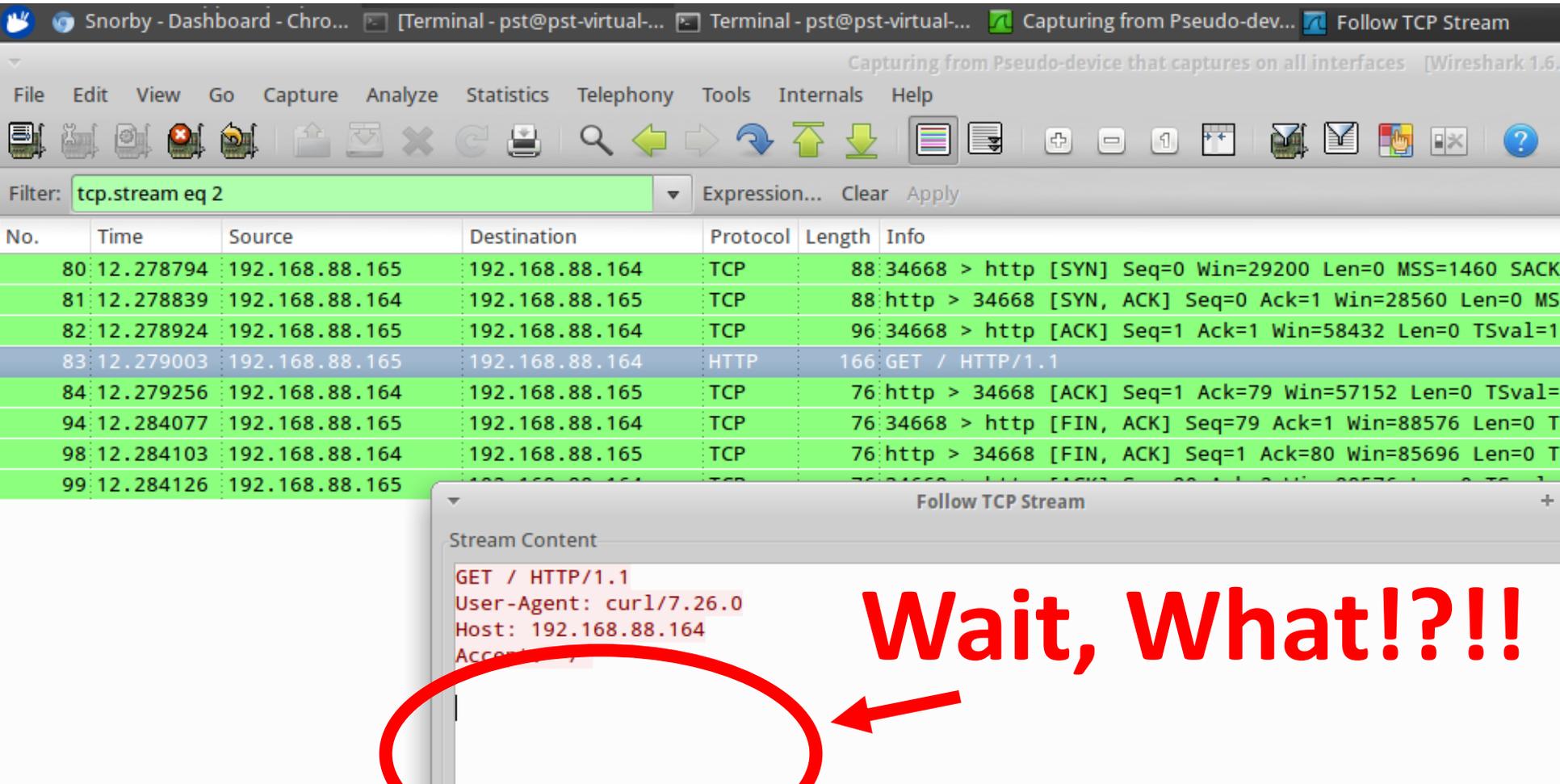
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help

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=12490113 TSecr=12474351
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 TSval=12490113 TSecr=12474351
82	12.278924	192.168.88.165	192.168.88.164	TCP	96	34668 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490113 TSecr=12474351
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=1 Win=0 Len=0 TSval=12474351 TSecr=12490113
85	12.280095	192.168.88.165	192.168.88.164	TCP	88	39757 > http [SYN] Seq=0 Win=29200 Len=0 MSS=1460 SACK_PERM=1 TSval=12490113 TSecr=12474351
86	12.280111	192.168.88.164	192.168.88.165	TCP	92	http > 39757 [SYN, ACK] Seq=0 Ack=1 Win=28560 Len=0 MSS=1460 SACK_PERM=1 TSval=12490113 TSecr=12474351
87	12.280222	192.168.88.165	192.168.88.164	TCP	92	39757 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490113 TSecr=12474351
88	12.280338	192.168.88.164	192.168.88.165	TCP	76	[TCP Window Update] Seq=1 Win=0 Len=0 TSval=12490113 TSecr=12474351
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 [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490113 TSecr=12474351
91	12.284045	192.168.88.165	192.168.88.164	TCP	88	[TCP Dup ACK 90#] Seq=1 Ack=1 Win=0 Len=0 TSval=12490113 TSecr=12474351
92	12.284064	192.168.88.164	192.168.88.165	TCP	88	[TCP Dup ACK 89#] Seq=0 Ack=1 Win=85696 Len=0 TSval=12490114 TSecr=12474351
93	12.284073	192.168.88.165	192.168.88.164	TCP	76	39757 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490114 TSecr=12474351
94	12.284077	192.168.88.165	192.168.88.164	TCP	76	34668 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490114 TSecr=12474351
95	12.284086	192.168.88.165	192.168.88.164	TCP	76	[TCP Dup ACK 93#] Seq=1 Ack=1 Win=0 Len=0 TSval=12490114 TSecr=12474351
96	12.284091	192.168.88.164	192.168.88.165	TCP	76	http > 39757 [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12474352 TSecr=12490113
97	12.284099	192.168.88.165	192.168.88.164	TCP	76	39757 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490114 TSecr=12474351
98	12.284103	192.168.88.164	192.168.88.165	TCP	76	http > 34668 [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12474352 TSecr=12490113
99	12.284126	192.168.88.165	192.168.88.164	TCP	76	34668 > http [ACK] Seq=1 Ack=1 Win=0 Len=0 TSval=12490114 TSecr=12474351

Frame 83: 166 bytes on wire (1328 bits), 166 bytes captured (1328 bits)
Linux cooked capture
Internet Protocol Version 4, Src: 192.168.88.165 (192.168.88.165), Dst: 192.168.88.164 (192.168.88.164)
Transmission Control Protocol, Src Port: 34668 (34668), Dst Port: http (80), Seq: 1, Ack: 1, Len: 78
Hypertext Transfer Protocol

2 Simple Examples: #1



Filter: tcp.stream eq 2

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	192.168.88.164	TCP	76	34668 > http [FIN, ACK] Seq=80 Ack=1 Win=85696 Len=0 T

Follow TCP Stream

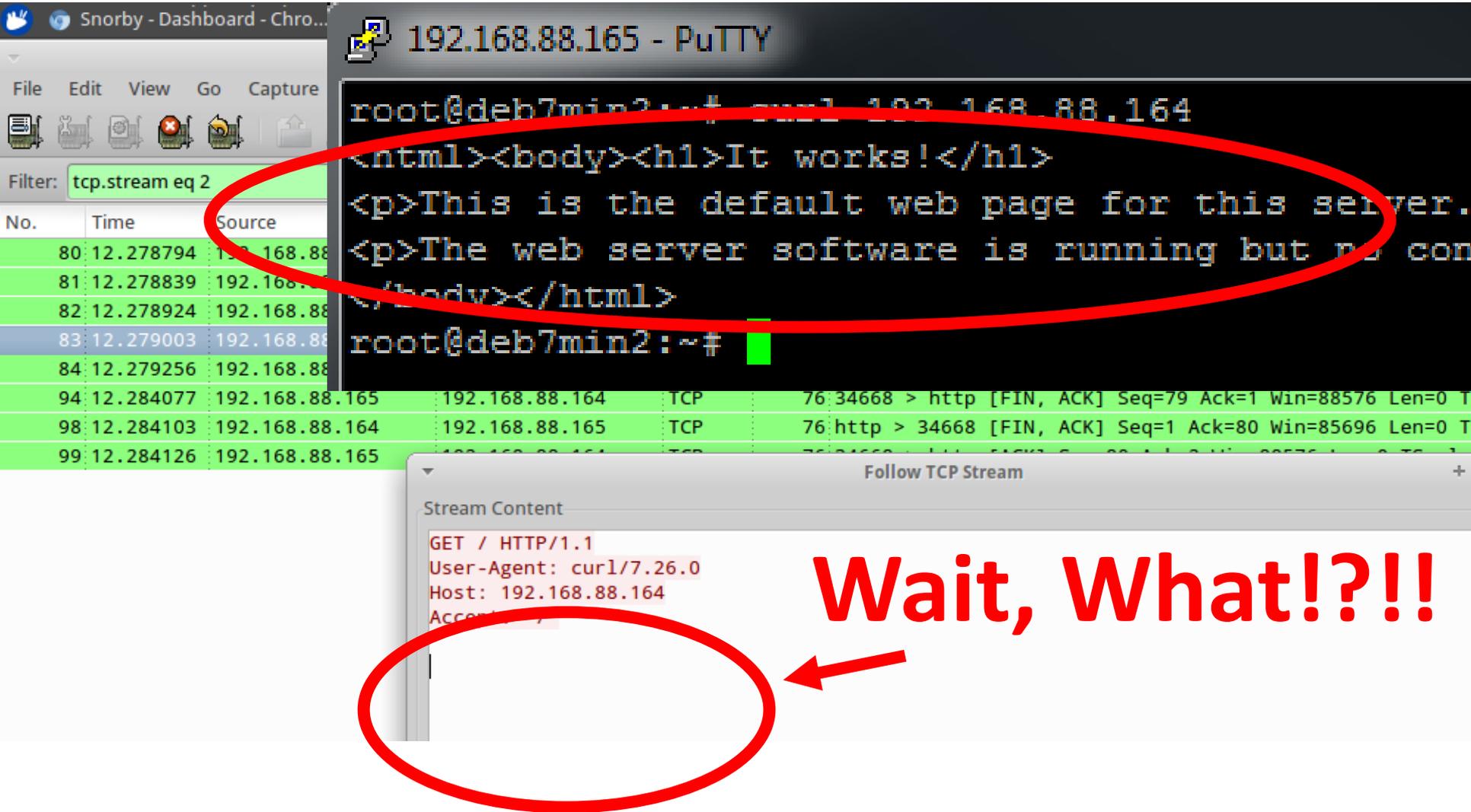
Stream Content

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

```

Wait, What!?!?

2 Simple Examples: #1



The image shows a Snorby dashboard on the left and a PuTTY terminal window on the right. The terminal window shows a root user at a deb7min2 machine running a curl command to fetch a page from 192.168.88.164. The output of the curl command is HTML code that says "It works!". A red circle highlights the HTML output in the terminal. Below the terminal, a "Follow TCP Stream" window shows the raw HTTP request: "GET / HTTP/1.1", "User-Agent: curl/7.26.0", "Host: 192.168.88.164", and "Accept: /*.*/". A red circle highlights the "Accept" header in the stream content, with a red arrow pointing to it from the text "Wait, What!?!".

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>The web server software is running but no con
</body></html>
root@deb7min2:~#
```

Filter: tcp.stream eq 2

No.	Time	Source	Destination	Protocol	Length	Info
80	12.278794	192.168.88.164	192.168.88.165	TCP	76	34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T...
81	12.278839	192.168.88.165	192.168.88.164	TCP	76	http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T...
82	12.278924	192.168.88.164	192.168.88.165	TCP	76	34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T...
83	12.279003	192.168.88.165	192.168.88.164	TCP	76	http > 34668 [FIN, ACK] Seq=1 Ack=80 Win=85696 Len=0 T...
84	12.279256	192.168.88.164	192.168.88.165	TCP	76	34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T...
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	192.168.88.164	TCP	76	34668 > http [FIN, ACK] Seq=79 Ack=1 Win=88576 Len=0 T...

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

```
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 2601:6:1700:168:2:41599 2601:6:1700:168:20:3000 ESTABLISHED
```

Err? 



Sense
This makes none



Why did we see that?

→ Let's talk about MPTCP

...but first, why change TCP?

Current TCP is rather limited

Doesn't support use cases for:

- High Availability
- Link Aggregation
- Multihoming
- Mesh networking

Multipath TCP

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

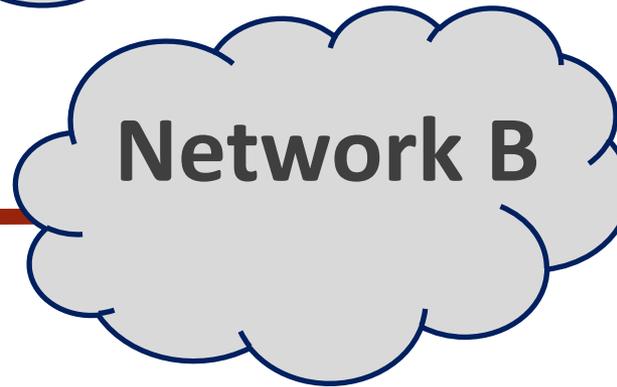
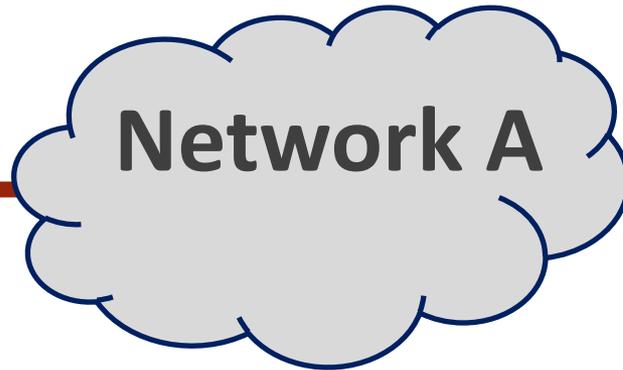
AND: it works over existing infrastructure

- (it *IS* TCP... just more so)

BUT: nothing much else understands it
– including security tools

MPTCP – Basic Use Cases

Client



Server



For seamless roaming

For high availability

TO BE CLEAR:

MPTCP is more culture shock than security vulnerability

We like MPTCP

We want MPTCP to succeed

Network security isn't ready



Background

Technical Introduction

Key Security Effects

Perimeter Security

Network Management

MPTCP Future

What got me thinking about this?

■ I saw this...

 **Hacker News** [new](#) | [comments](#) | [ask](#) | [jobs](#) | [submit](#) [login](#)

👤 Apple seems to also believe in Multipath TCP ([uclouvain.be](#))
10 points by [rhapsodyv](#) 278 days ago | [discuss](#) | [save to pocket](#)

What got me thinking about this?

- Which led to this...

 **Hacker News** | [new](#) | [comments](#) | [ask](#) | [jobs](#) | [submit](#) | [login](#)

Olivier Bonaventure

Homepage and blog

[Home](#) | [About](#) | [CV](#) | [Publications](#) | [Teaching](#) | [People](#)

« [Is your network ready for iOS7 and Multipath TCP ?](#)
Quickly producing time-sequence diagrams »

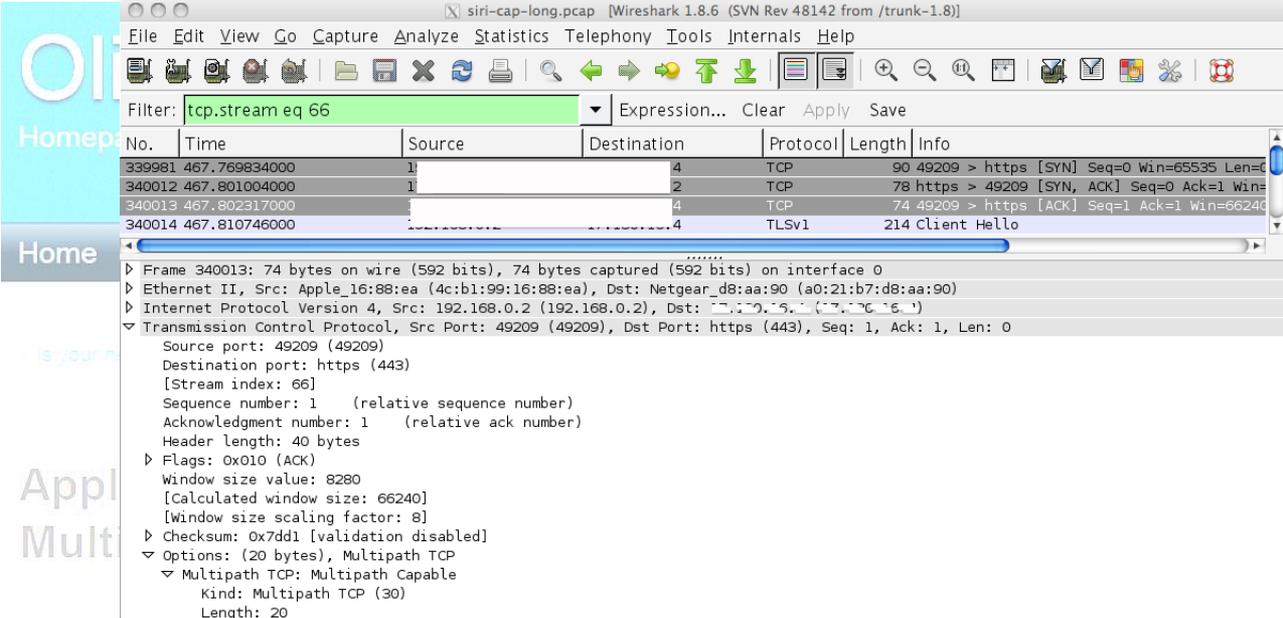
September 18, 2013

Apple seems to also believe in Multipath TCP

What got me thinking about this?

■ Which contained this...

[Y Hacker News](#) [new](#) | [comments](#) | [ask](#) | [jobs](#) | [submit](#) [login](#)



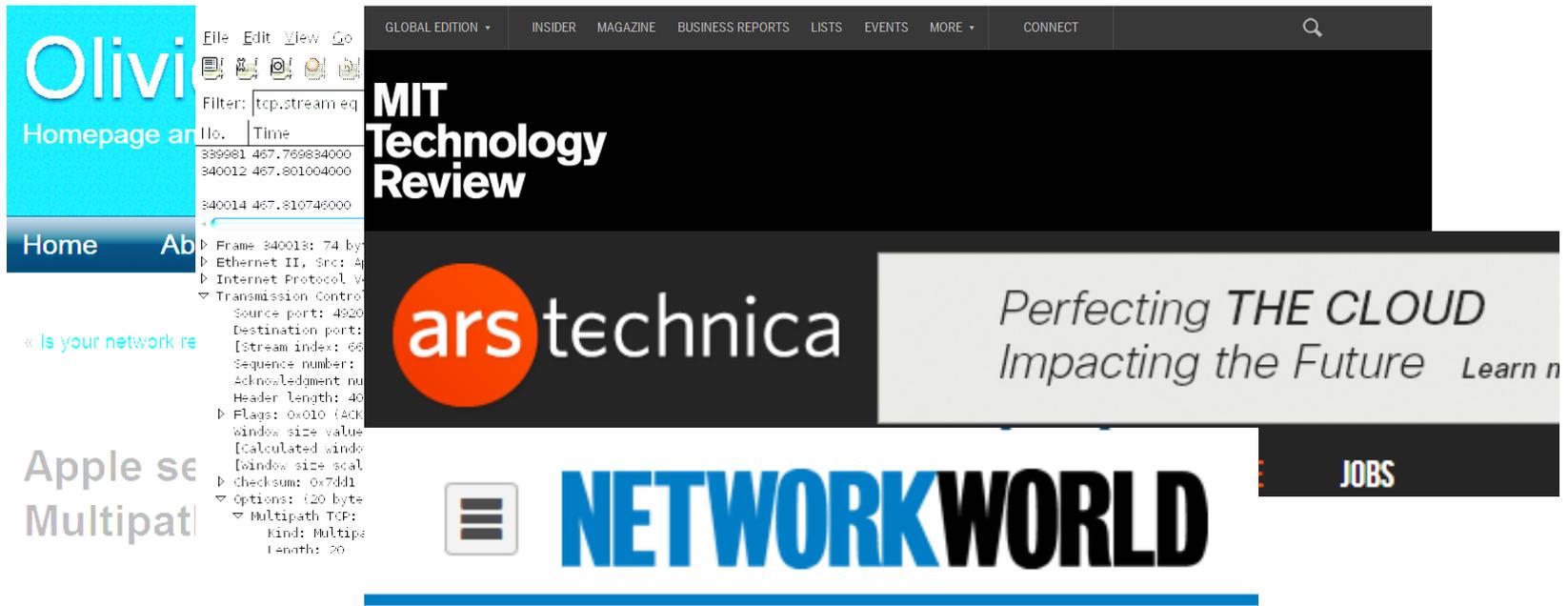
The image shows a Wireshark 1.8.6 interface with a capture file named 'siri-cap-long.pcap'. The filter is set to 'tcp.stream eq 66'. The packet list shows several packets, with packet 340013 selected. The packet details pane shows the following information:

- Frame 340013: 74 bytes on wire (592 bits), 74 bytes captured (592 bits) on interface 0
- Ethernet II, Src: Apple_16:88:ea (4c:b1:99:16:88:ea), Dst: Netgear_d8:aa:90 (a0:21:b7:d8:aa:90)
- Internet Protocol Version 4, Src: 192.168.0.2 (192.168.0.2), Dst: 192.168.0.4 (192.168.0.4)
- Transmission Control Protocol, Src Port: 49209 (49209), Dst Port: https (443), Seq: 1, Ack: 1, Len: 0
 - Source port: 49209 (49209)
 - Destination port: https (443)
 - [Stream index: 66]
 - Sequence number: 1 (relative sequence number)
 - Acknowledgment number: 1 (relative ack number)
 - Header length: 40 bytes
 - Flags: 0x010 (ACK)
 - Window size value: 8280
 - [Calculated window size: 66240]
 - [Window size scaling factor: 8]
 - Checksum: 0x7dd1 [validation disabled]
 - Options: (20 bytes), Multipath TCP
 - Multipath TCP: Multipath Capable
 - Kind: Multipath TCP (30)
 - Length: 20

What got me thinking about this?

- Then other media outlets started covering it...

[Y](#) [Hacker News](#) [new](#) | [comments](#) | [ask](#) | [jobs](#) | [submit](#) [login](#)



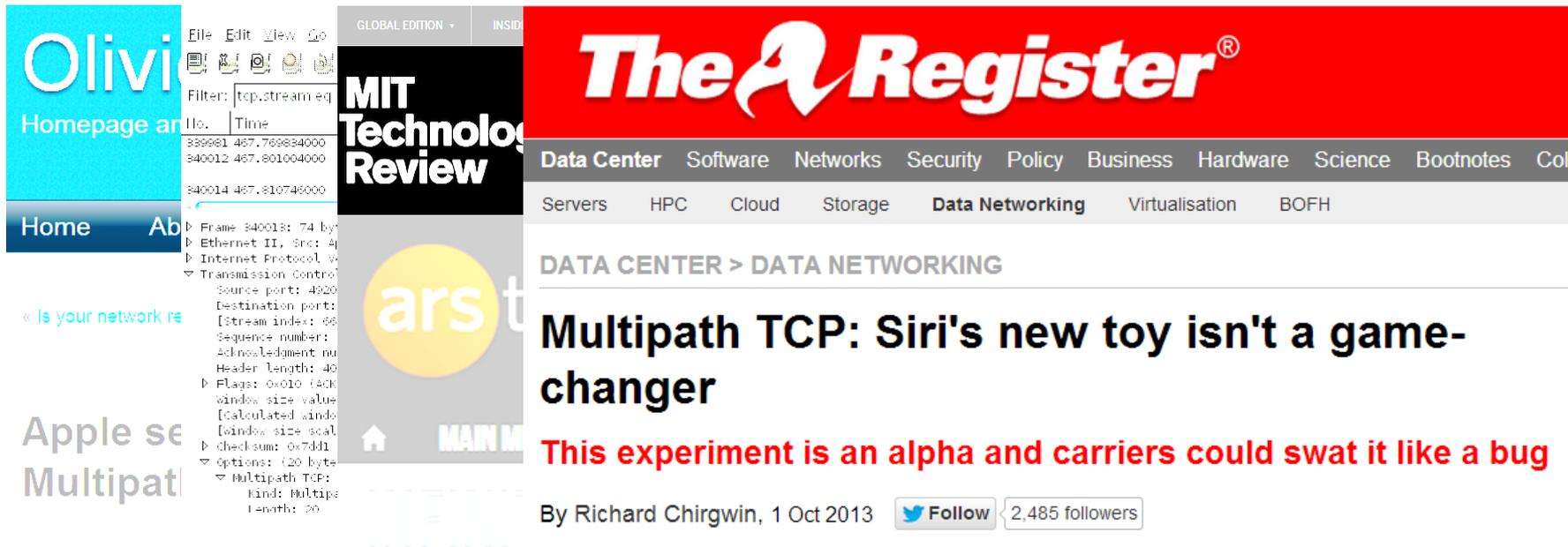
The collage features several key elements:

- Wireshark Packet Capture:** A network traffic analysis window showing a filter for 'tcp.stream eq' and a list of packets. The selected packet (No. 340014) is expanded to show its structure: Frame 340013: 74 bytes on wire (Ethernet II, Src: Apple, Dst: Apple, Protocol: TCP, Length: 74), Transmission Control Protocol, Source port: 4920, Destination port: [Stream index: 66], Sequence number: [Acknowledgment nu], Header Length: 40, Flags: 0x010 (ACK), window size value [Calculated window size scal], [window size scal], checksum: 0x7dd1, Options: (20 bytes), Multipath TCP: Kind: Multipath, Length: 20.
- MIT Technology Review:** A prominent logo in the center.
- Ars Technica:** A logo with the text 'ars technica' below it.
- Perfecting THE CLOUD:** A banner with the text 'Perfecting THE CLOUD Impacting the Future Learn n'.
- NetworkWorld:** A large logo at the bottom with a 'JOBS' button to its right.

What got me thinking about this?

- Then other media outlets started covering it...not always positively

[Y](#) [Hacker News](#) [new](#) | [comments](#) | [ask](#) | [jobs](#) | [submit](#) [login](#)



The Register

Data Center Software Networks Security Policy Business Hardware Science Bootnotes Col

Servers HPC Cloud Storage **Data Networking** Virtualisation BOFH

DATA CENTER > DATA NETWORKING

Multipath TCP: Siri's new toy isn't a game-changer

This experiment is an alpha and carriers could swat it like a bug

By Richard Chirgwin, 1 Oct 2013 [Follow](#) 2,485 followers

What got me thinking about this?

- And then...

SILENCE

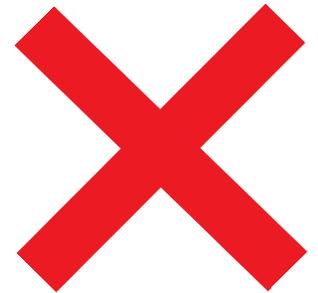
- BUT, the rate of progress was unprecedented for a major change to TCP

Was anyone thinking about security?

- The security of MPTCP itself



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



... not so much

That's what this session is about

- What does multipath TCP mean for security *today*?
- What could it (or similar tech) mean to network security *a decade from now*?
- With a couple of attacks and tools...



Background

Technical Introduction

Key Security Effects

Perimeter Security

Network Management

MPTCP Future

Motivations and Advantages

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

MPTCP Characteristics

- Backwards compatibility
- Performance \geq now
- Security \geq now

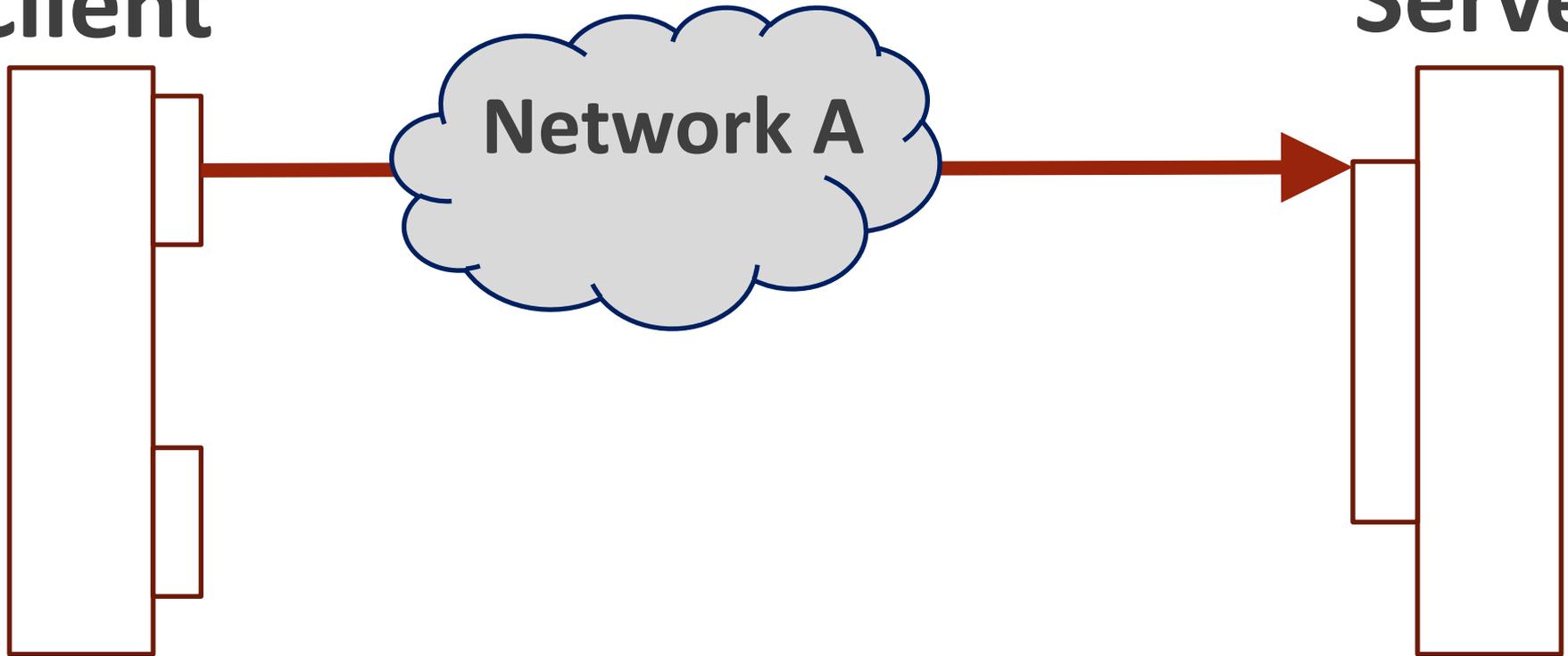
Riding on top of TCP

- **An MPTCP Connection** is defined by a connection ID
- It is comprised of multiple *streams*, where each stream is a regular TCP connection (with an option strapped on)

MPTCP – Simple Case

Client

Server

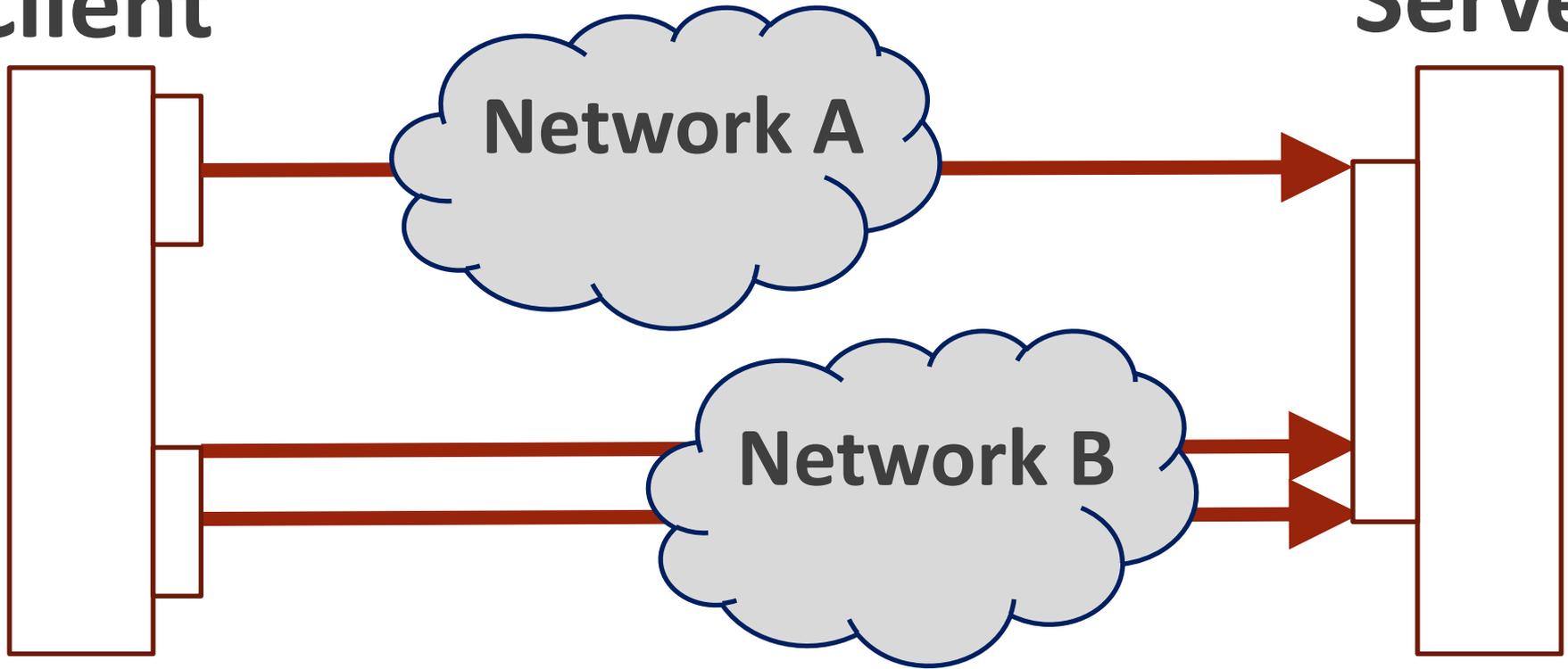


MPTCP connection looks like TCP so far...

MPTCP – Simple Case

Client

Server

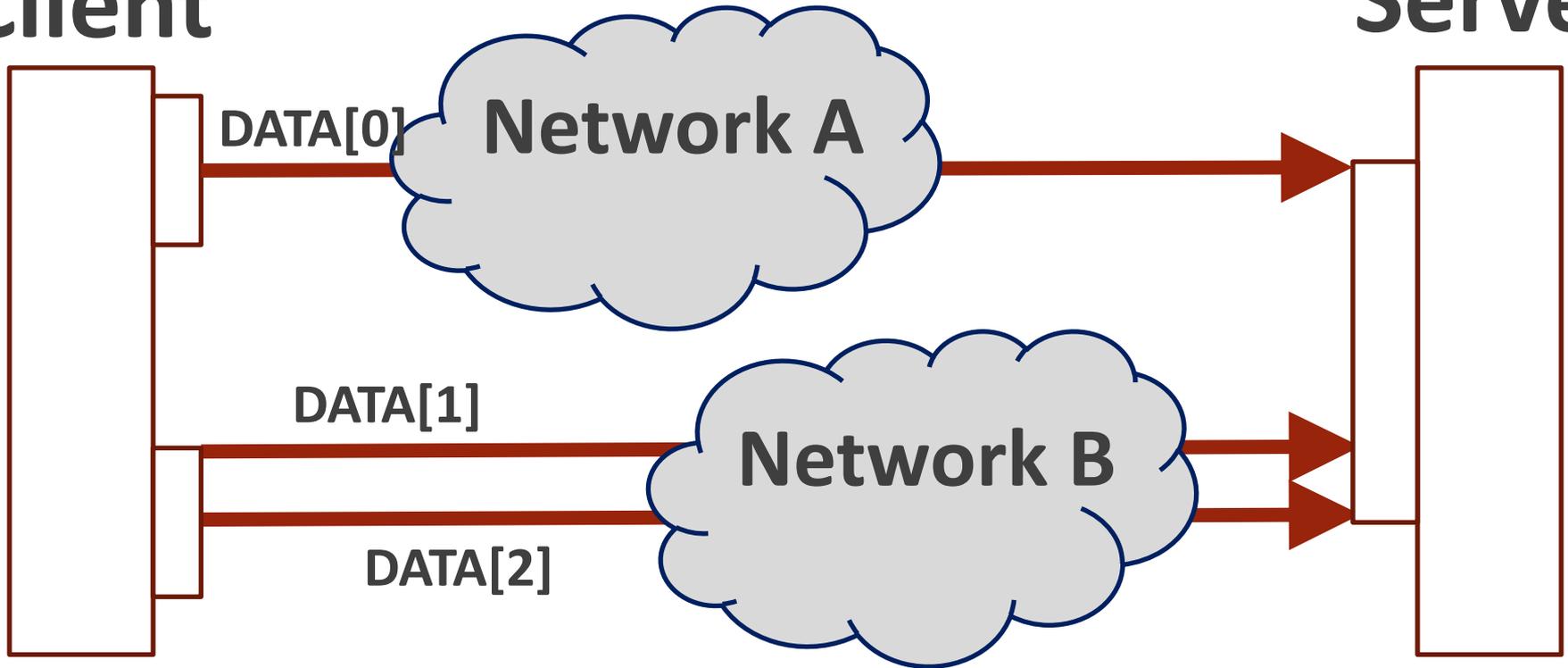


N different TCP connections, contributing to ***ONE*** logical data flow

MPTCP – Simple Case

Client

Server

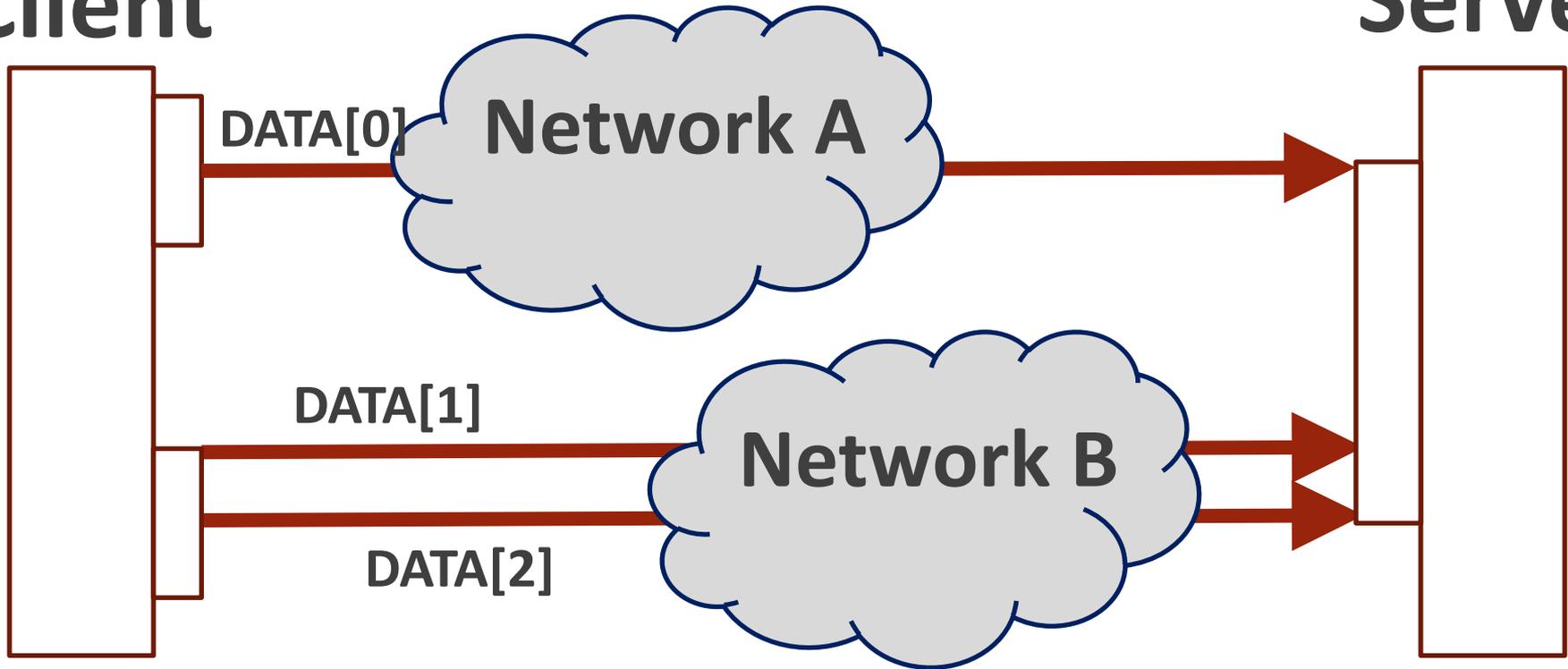


N different TCP connections, contributing to **ONE** logical data flow... data flows through any/all

MPTCP – Simple Case

Client

Server

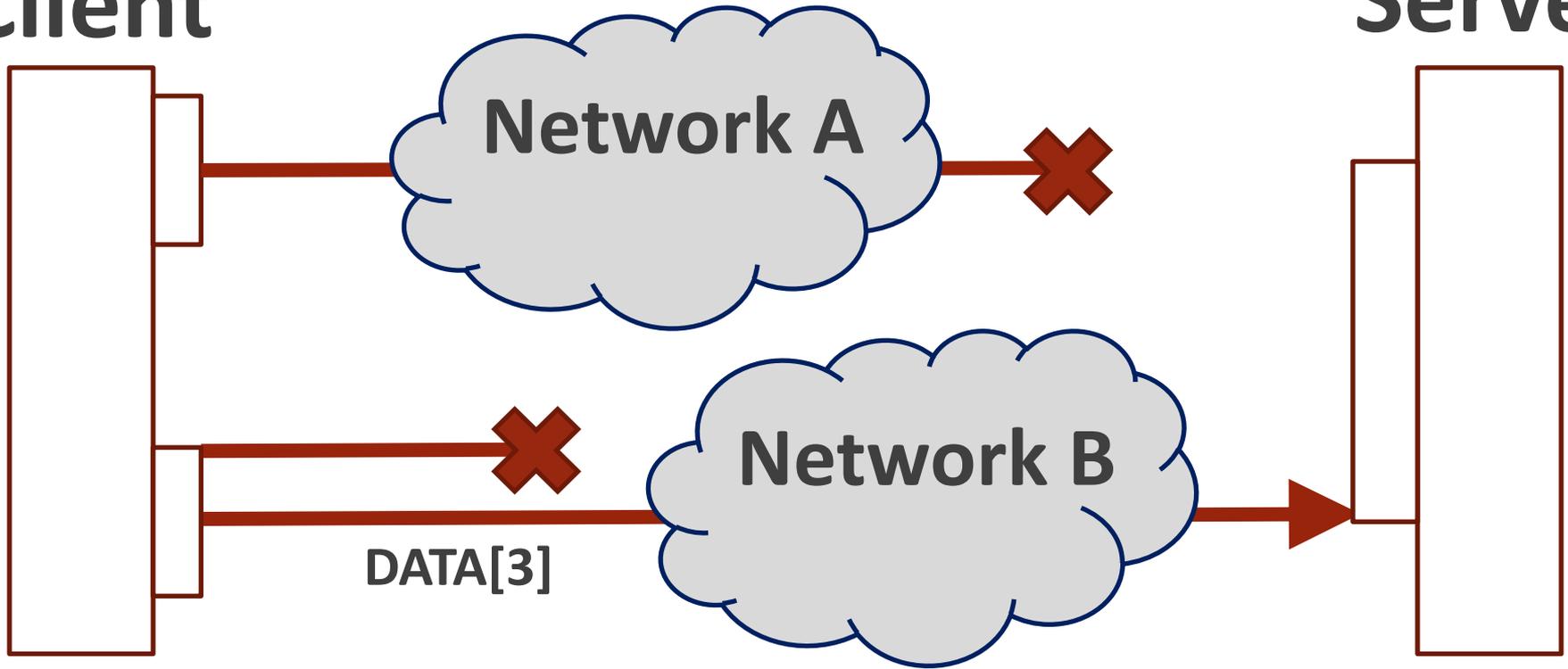


Sender of a packet can choose to use
any flow *(this will be important)*

MPTCP – Simple Case

Client

Server

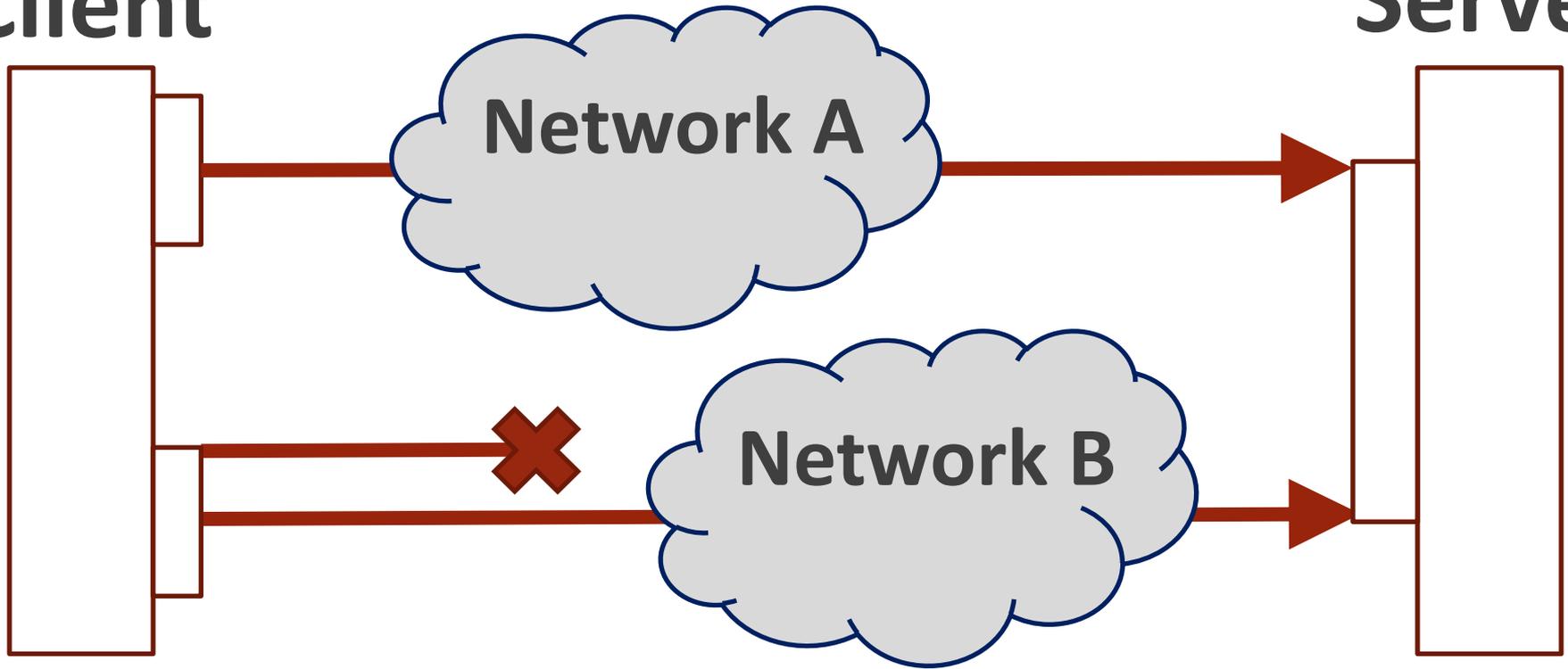


Any subset of connections can drop,
overall flow continues.

MPTCP – Simple Case

Client

Server

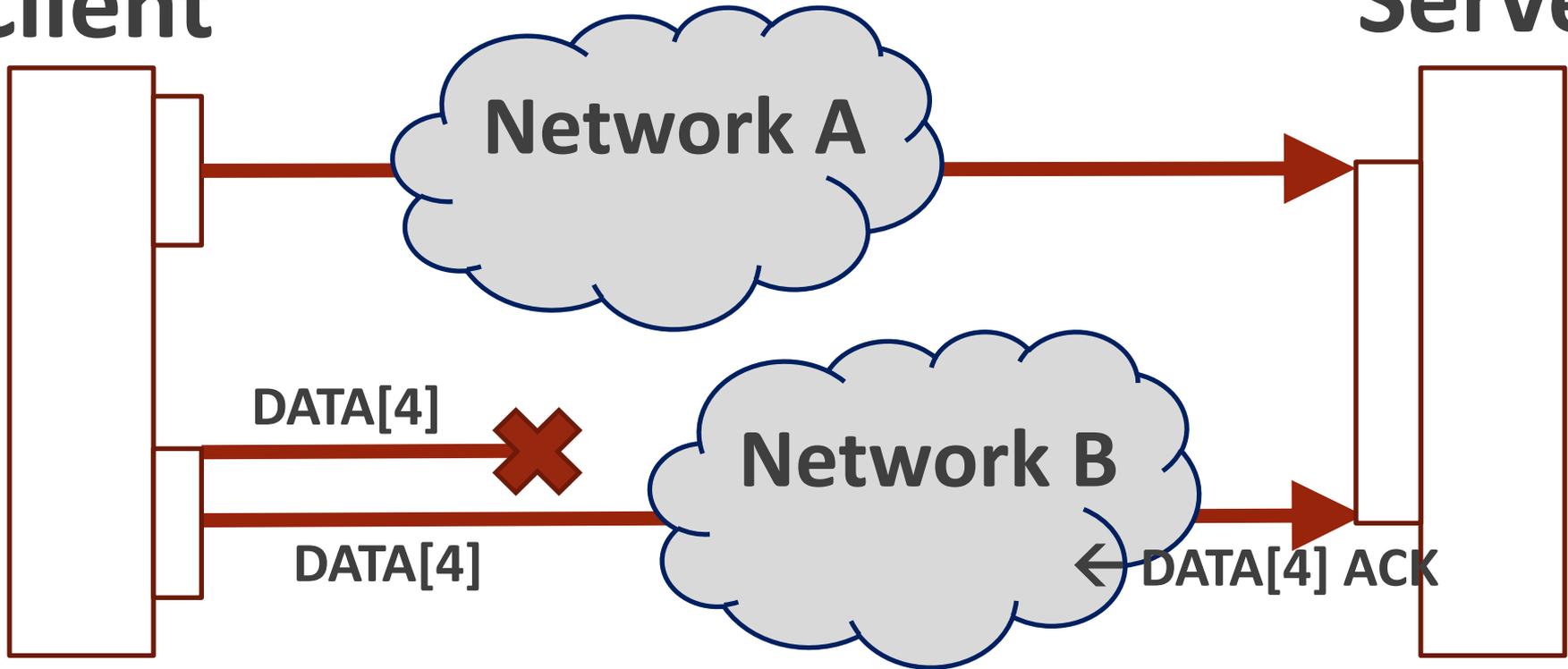


Connections can be re-added at any time

MPTCP – Simple Case

Client

Server



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

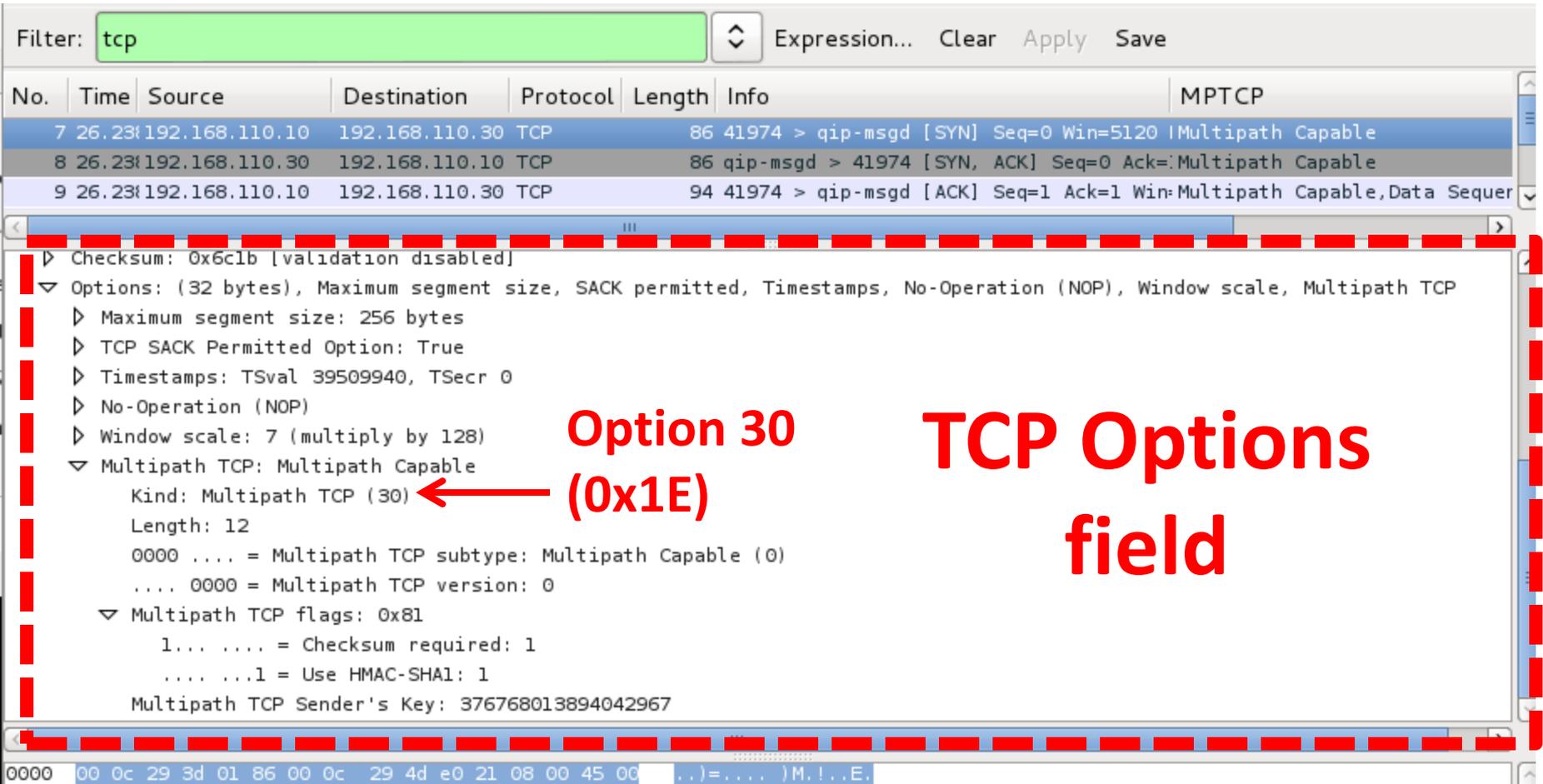


How is MPTCP implemented? – TCP Option

Bits 0 - 7		Bits 8 - 15		Bits 16 - 23		Bits 24 - 31	
Source Port				Destination port			
TCP Sequence Number							
TCP Acknowledgement Number (if Ack Set)							
Data Offset	Reserved	TCP Flags (Ack, Syn etc)		Window Size			
Checksum				Urgent Pointer (if URG Set)			
0x1e (MPTCP Option Type)		Length		Subtype	MPTCP Ver	MPTCP Flags	
Remaining MPTCP Subtype Data							
Packet DATA							

What does it look like?

■ Packet Breakdown - Wireshark



Filter: tcp

No.	Time	Source	Destination	Protocol	Length	Info	MPTCP
7	26.231	192.168.110.10	192.168.110.30	TCP	86	41974 > qip-msgd [SYN] Seq=0 Win=5120 Multipath Capable	
8	26.231	192.168.110.30	192.168.110.10	TCP	86	qip-msgd > 41974 [SYN, ACK] Seq=0 Ack= Multipath Capable	
9	26.231	192.168.110.10	192.168.110.30	TCP	94	41974 > qip-msgd [ACK] Seq=1 Ack=1 Win= Multipath Capable, Data Sequen	

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) ← **Option 30 (0x1E)**
 - 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.



How is MPTCP implemented? – MPTCP Subtypes

- 8 currently defined (ones relevant in **bold**)
- **MP_CAPABLE** - Signals MPTCP support
- **MP_JOIN** - Add incoming subflow to the connection
- **DSS** - How to map this stream's data against the overall data flow
- **ADD_ADDR** - This address is also a way to reach me
- **REMOVE_ADDR** - Please stop using [address] to reach me
- MP_PRIO
- MP_FAIL
- MP_FASTCLOSE

Path Management - Linux

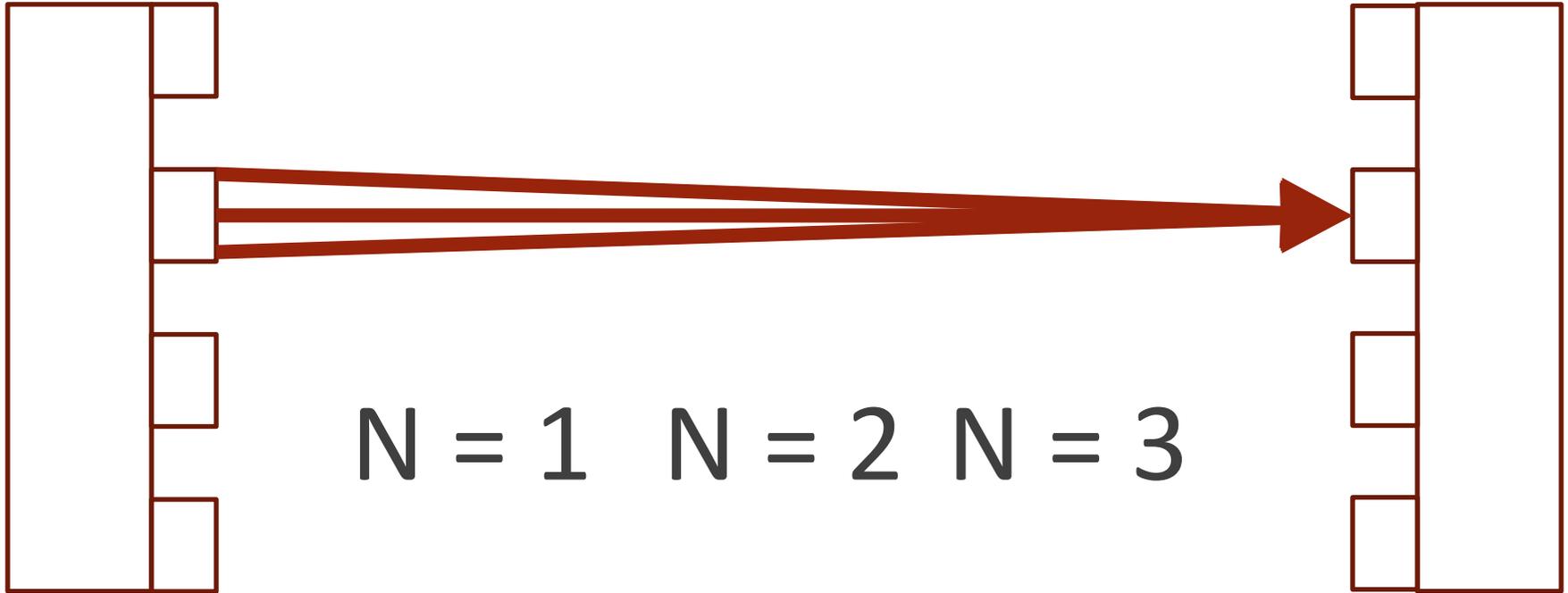
- The Linux Path Manager has two primary path managers at present
 - Fullmesh – n:n (all to all)
 - Ndiffports – 1-1 interfaces, n-1 ports

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

Path Management - ndiffports

Client

Server

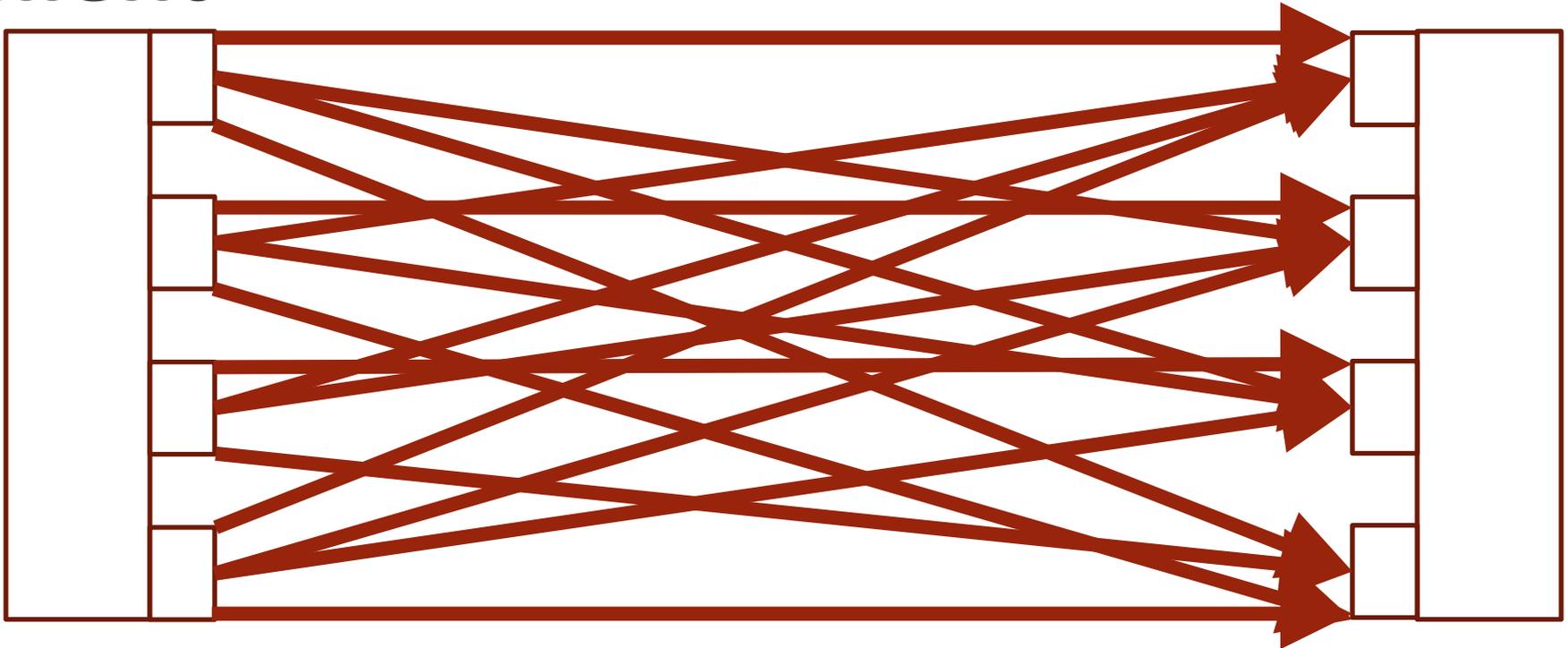


N different source ports,
1 destination port

Path Management - fullmesh

Client

Server



All possible paths used

Deeper technical details

- TCP Handshake with additional details
 - Data sequence numbering
 - Truncation of SHA1 of host key
 - Authentication
 - MP_JOIN - Challenge-response HMAC of other host's key, Nonce, AddressID
 - MP_FASTCLOSE – Other party's key in plaintext
 - Routing
 - Packet *sender* decides which data goes down which path
- More on this later...**



Release Item #1 - Cheatsheet

MPTCP Cheatsheet

MPTCP Header:

Bits 0 - 7		Bits 8 - 15		Bits 16 - 23		Bits 24 - 31	
Source Port				Destination port			
TCP Sequence Number							
TCP Acknowledgement Number (if Ack Set)							
Data Offset	Reserved	TCP Flags (Ack, Syn etc)		Window Size			
Checksum				Urgent Pointer (if URG Set)			
MP_Capable	Length	Subtype	MPTCP-Ver	MPTCP Flags			
Remaining MPTCP Subtype Data							
Packet DATA							

MPTCP Subtype	HEX	Flags?	Other Likely fields of interest
MP_CAPABLE	0x0		
MP_JOINS	0x1		
DSS	0x2		
ADD_ADDR	0x3		
REMOVE_ADDR	0x4		
MP_PRIO	0x5		
MP_FAIL	0x6		
MP_FASTCLOSE	0x7		

Getting the MPTCP Sequence Numbers:

Key	64 Bit number supplied by host	
Initial DSN (ISDN):	SHA1(key)[-64:]	Binary mode hash, network byte order

Initial DSS		
Subflow DSS	mapping likely starts at ISDN[0:32] + TCP ISN + 1	TCP Seq is 32 bits, + 1 for the SYN
MP_JOIN		

MP_JOIN Authentication (RFC 6824 Fig 8)

A	B
TCP_SYN, MP_JOIN (TokenB, NonceA) ->	
	<- TCP_SYN_ACK, MP_JOIN(HMAC(Key=KB+KA, Msg = NonceA + NonceB), NonceB)
TCP_ACK, MP_JOIN(HMAC(Key=KA+KB,Msg=NonceB+NonceA) ->	
	<- TCP_ACK
<i>Token = ConnectionID = SHA1(Key)[0:32] of Other Party's key. (Capture from either steps 2 or 3 in the first handshake)</i>	

Detecting MPTCP things

Passive:	Usage	Inbound Connection Attempts	Detect inbound connection attempts - Look for the SYN packets with MPTCP Header	TCP(SYN) TCP Option= 30 ** 00 ...
		Successful Handshake	(Pre-viability) Look for Ack Packets with MPTCP Option header	TCP(ACK) TCP Option = 30 ** 00 ...
		Valid Handshake	MPTCP Option header Look for Ack Packets with the MPTCP Option Header	TCP(ACK) TCP Option = 30 ** 00 ...
		MPTCP Joins	TCP SYN Packets with MPTCP TCP Option and an MP_JOIN subtype	TCP(SYN) TCP Option = 30 ** 01 ...
	Attacks	MPTCP Simple	Non look for non sequential last	

So who's using it?

- Nearly no one is using it large scale (yet), with a few exceptions
 - Apple (Siri)
 - Some other experimental stuff?
- Given that, there's a surprising number of implementations
 - Implementations available for several OS's (including Linux, BSD, Android), and baked in some way into commercial kit (Citrix, Cisco, Apple, Oracle, F5)
 - **NOT Windows**

Availability – Getting it working

- Linux
 - Linux reference implementation via apt-get (multipath-tcp.org) -- best way right now
 - Can work in Kali, but ... challenges
- Nicolas Maître made *a ridiculously useful*, near complete, SCAPY implementation
 - We're based some tools on this code, and fixed some bugs along the way
 - <https://github.com/nimai/mptcp-scapy>



Background

Technical Introduction

Key Security Effects

Perimeter Security

Network Management

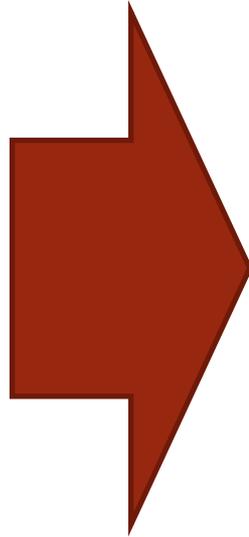
MPTCP Future

MPTCP's Key Security 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

Because of these...

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



... if your approach to security requires *any* of these...

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

...then something is probably going to break

How practical are these attacks?

- Today? Extremely.
 - But only if both endpoints speak MPTCP
 - Of which... there aren't many. Yet.

- In an MPTCP world, a bit less
 - But we have to change the way we do things in network security

MPTCP's Key Security Effects

- All of those things can be *partially* mitigated with MPTCP aware infrastructure and security tools.
- But overall, there remain some interesting shifts in how network flows work – especially if we go in with “well meaning” intent

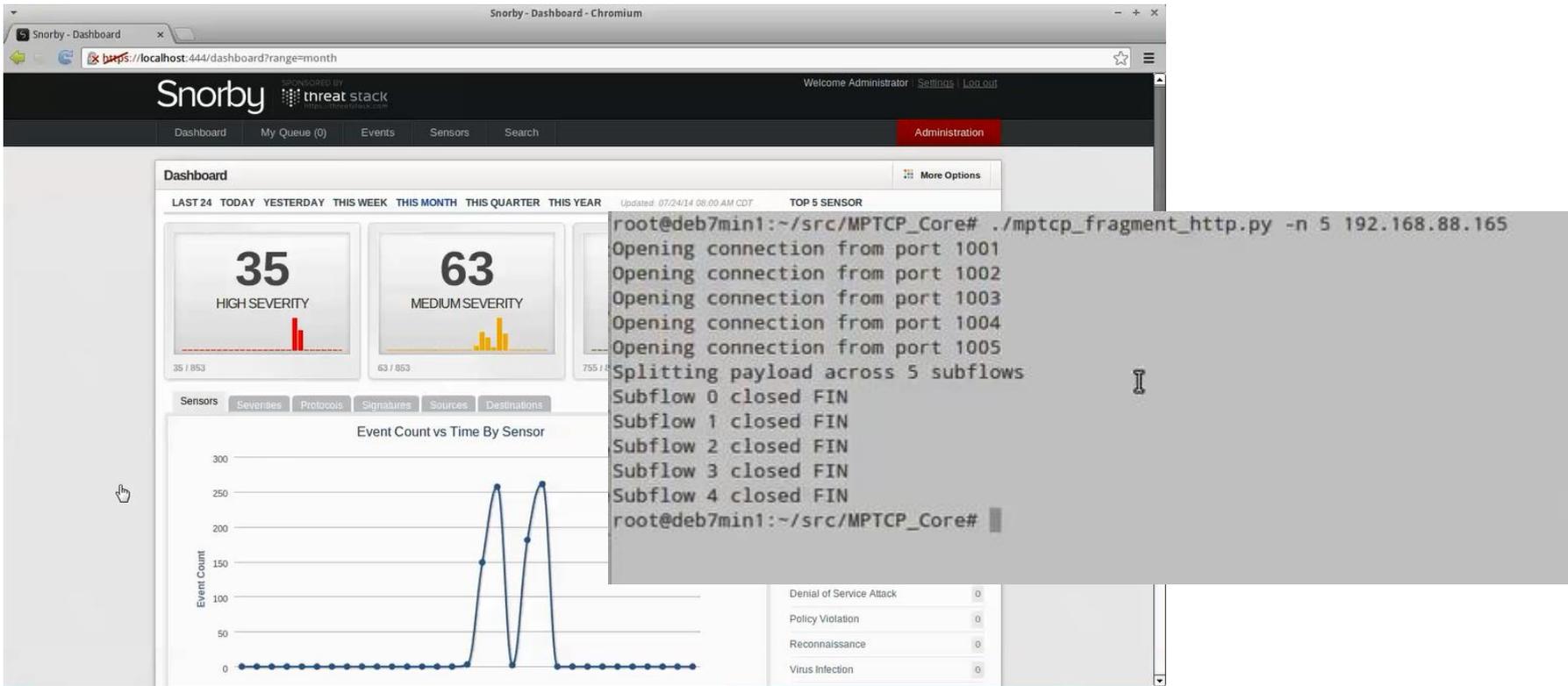
MPTCP's Key Security Effects

A few slides back...

- The **packet sender decides** which data goes down which path.
- Normal/benign clients won't choose pathological fragmentation schemes
 - But there's nothing stopping us...

Release Item #2: PoC tool for MPTCP IDS Evasion

■ Demo!



The screenshot displays the Snorby dashboard interface. The top navigation bar includes 'Dashboard', 'My Queue (0)', 'Events', 'Sensors', 'Search', and 'Administration'. The main dashboard area shows event counts for 'HIGH SEVERITY' (35) and 'MEDIUM SEVERITY' (63). A line graph titled 'Event Count vs Time By Sensor' shows two distinct peaks. A terminal window is overlaid on the right, showing the execution of the MPTCP evasion tool. The terminal output includes the command `./mptcp_fragment_http.py -n 5 192.168.88.165`, followed by five 'Opening connection from port' messages (1001-1005), a 'Splitting payload across 5 subflows' message, and five 'Subflow X closed FIN' messages.

```
root@deb7min1:~/src/MPTCP_Core# ./mptcp_fragment_http.py -n 5 192.168.88.165
Opening connection from port 1001
Opening connection from port 1002
Opening connection from port 1003
Opening connection from port 1004
Opening connection from port 1005
Splitting payload across 5 subflows
Subflow 0 closed FIN
Subflow 1 closed FIN
Subflow 2 closed FIN
Subflow 3 closed FIN
Subflow 4 closed FIN
root@deb7min1:~/src/MPTCP_Core#
```



Background

Technical Introduction

Key Security Effects

Perimeter Security

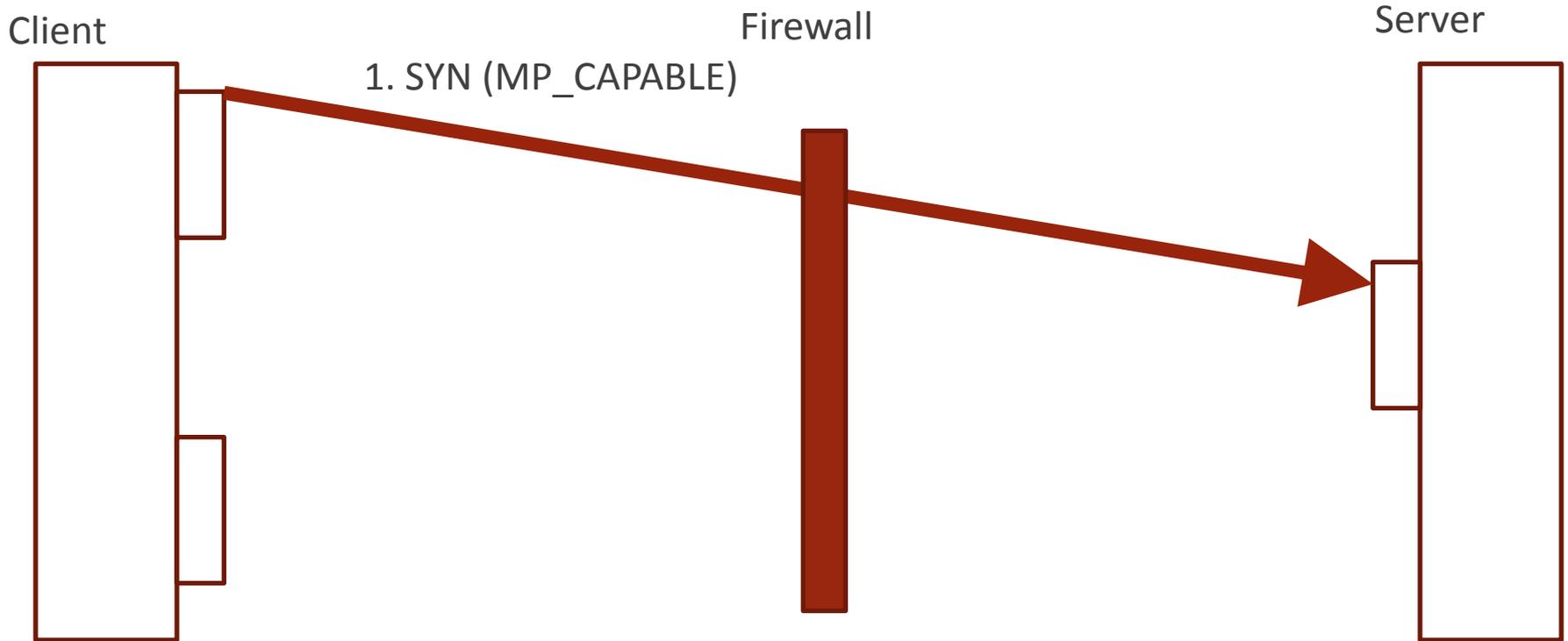
Network Management

MPTCP Future

MPTCP and ... Firewalls

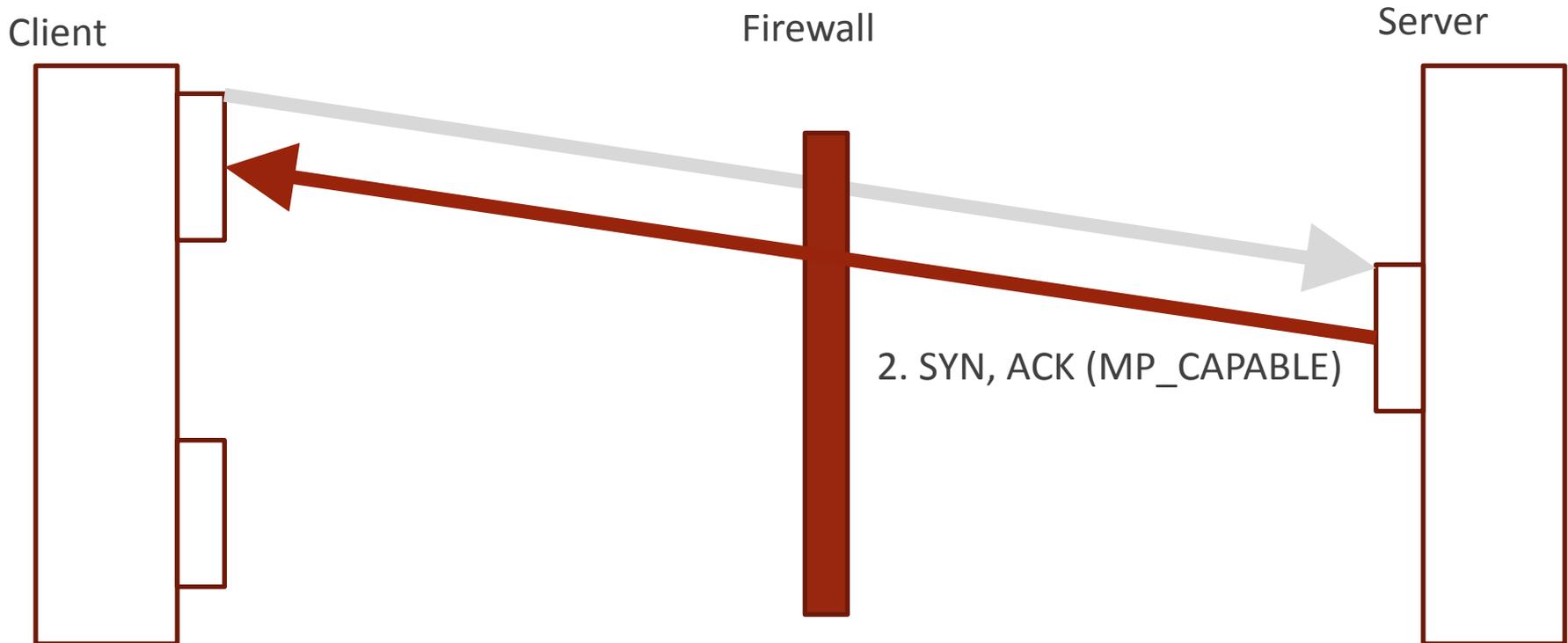
- MPTCP changes things for perimeters
- How'd you like an outbound incoming connection?

MPTCP and ... Firewalls



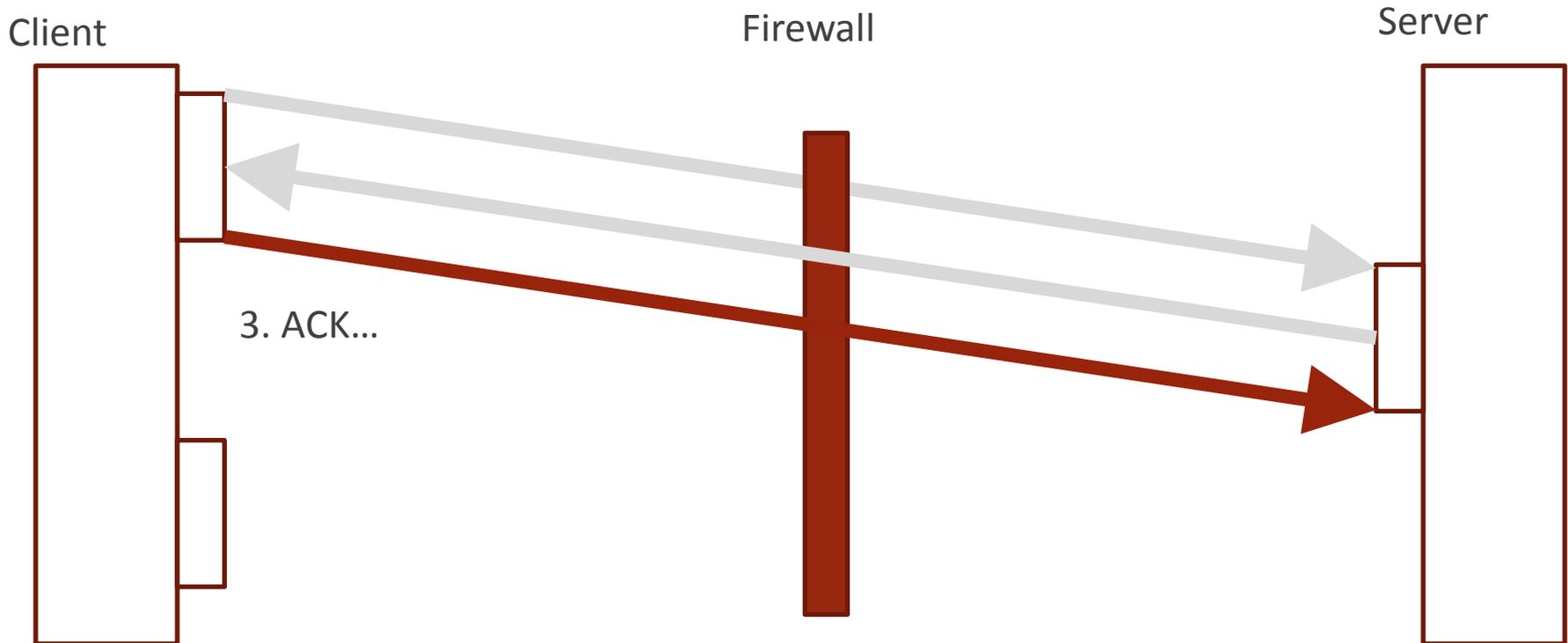
- MPTCP connection looks like TCP so far...

MPTCP and ... Firewalls



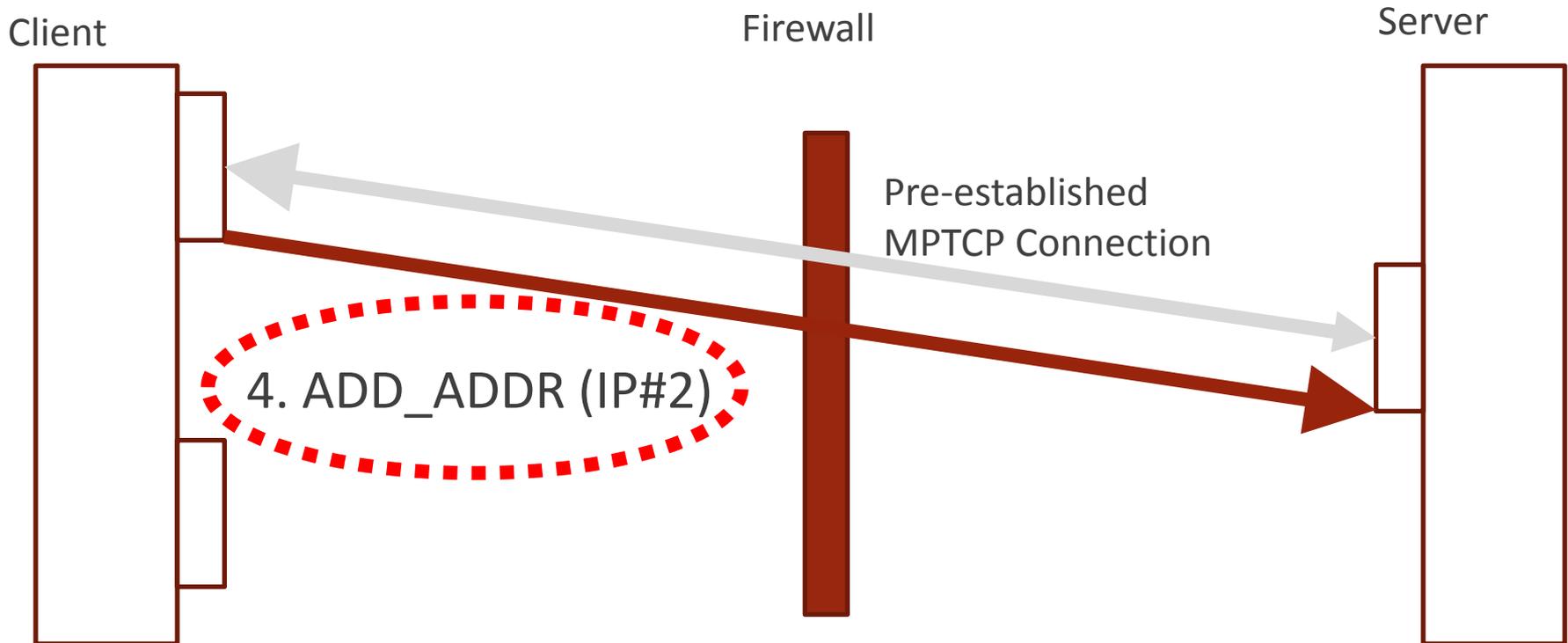
- Still seems pretty standard, albeit with extra TCP OPTIONS

MPTCP and ... Firewalls



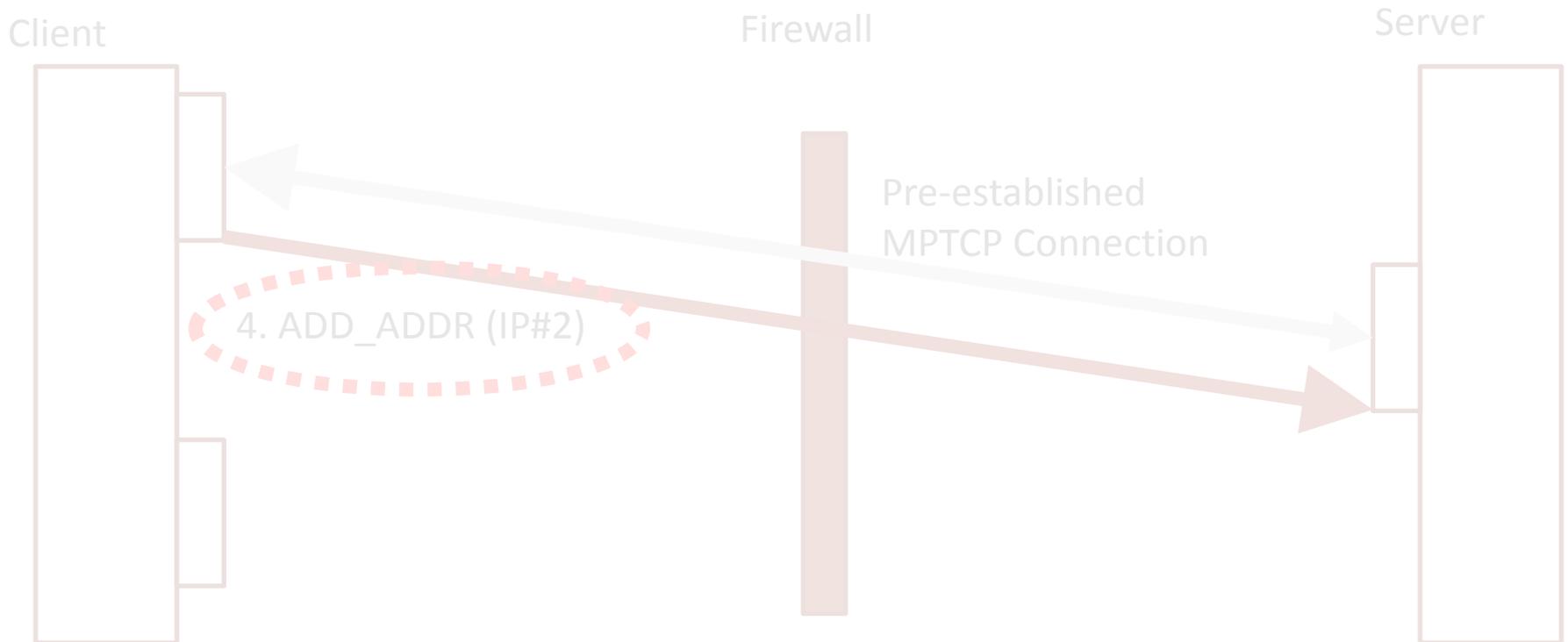
- OK, so it's a TCP connection with an additional options... so what?

MPTCP and ... Firewalls



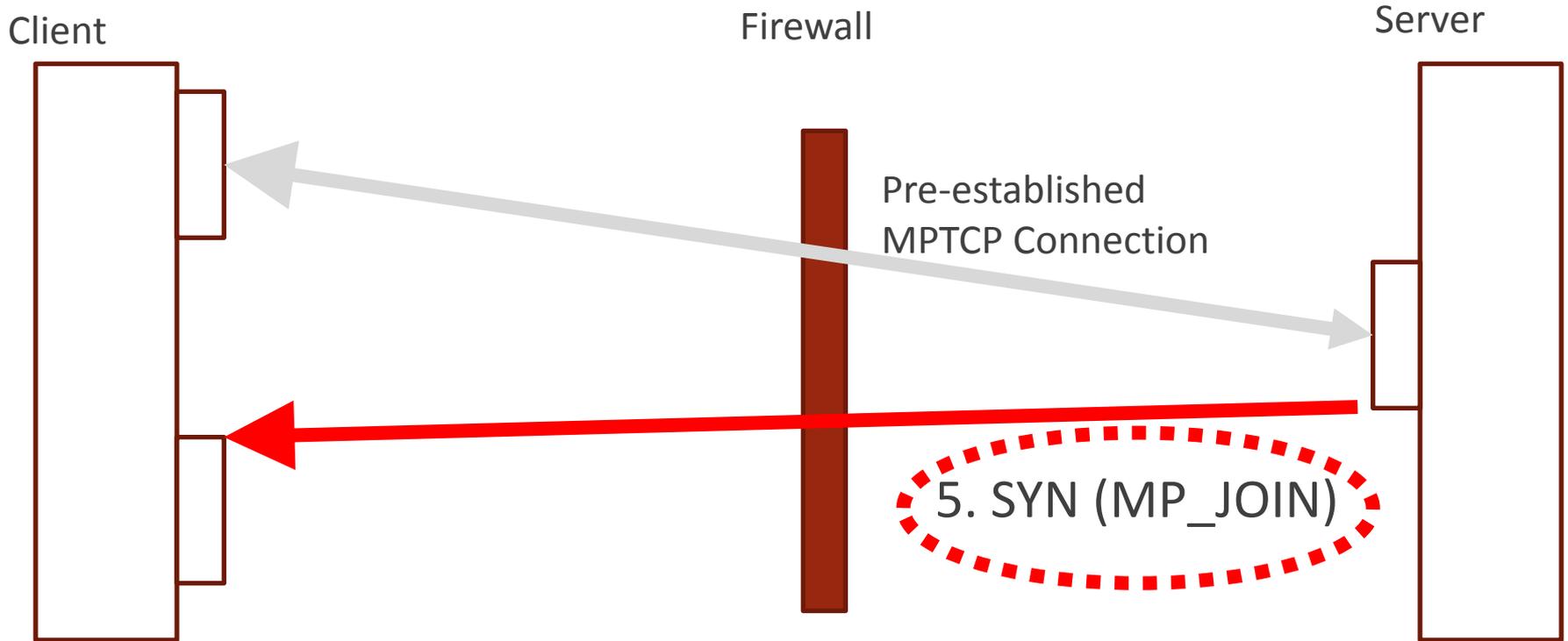
- Well, what if the client tells the server about a new address?

MPTCP and ... Firewalls



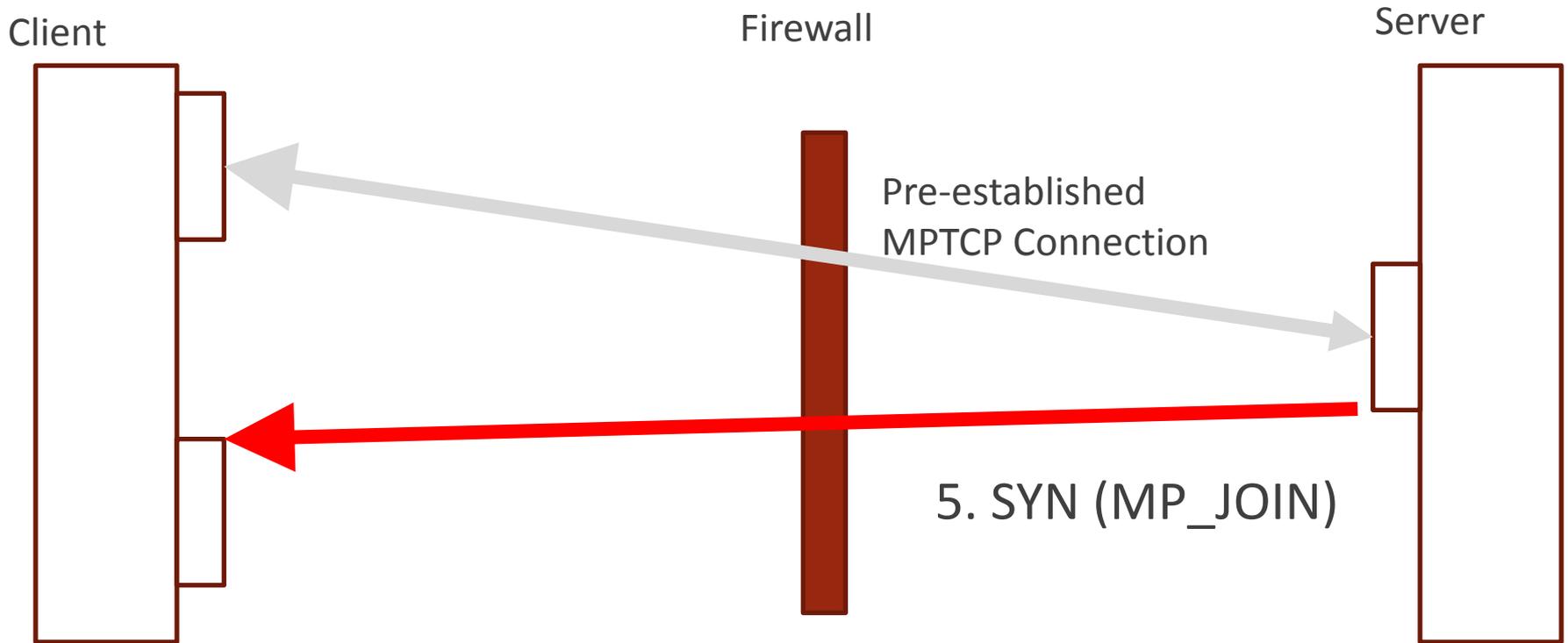
- Now, the “Internal” host may set up a connection to the advertised address

MPTCP and ... Firewalls



- Is this new connection incoming or outgoing?

MPTCP and ... Firewalls



- Is this new connection **incoming** or **outgoing**?

MPTCP and ... Firewalls



- Is this new connection **incoming** or **outgoing**?



Background

Technical Introduction

Key Security Effects

Perimeter Security

Network Management

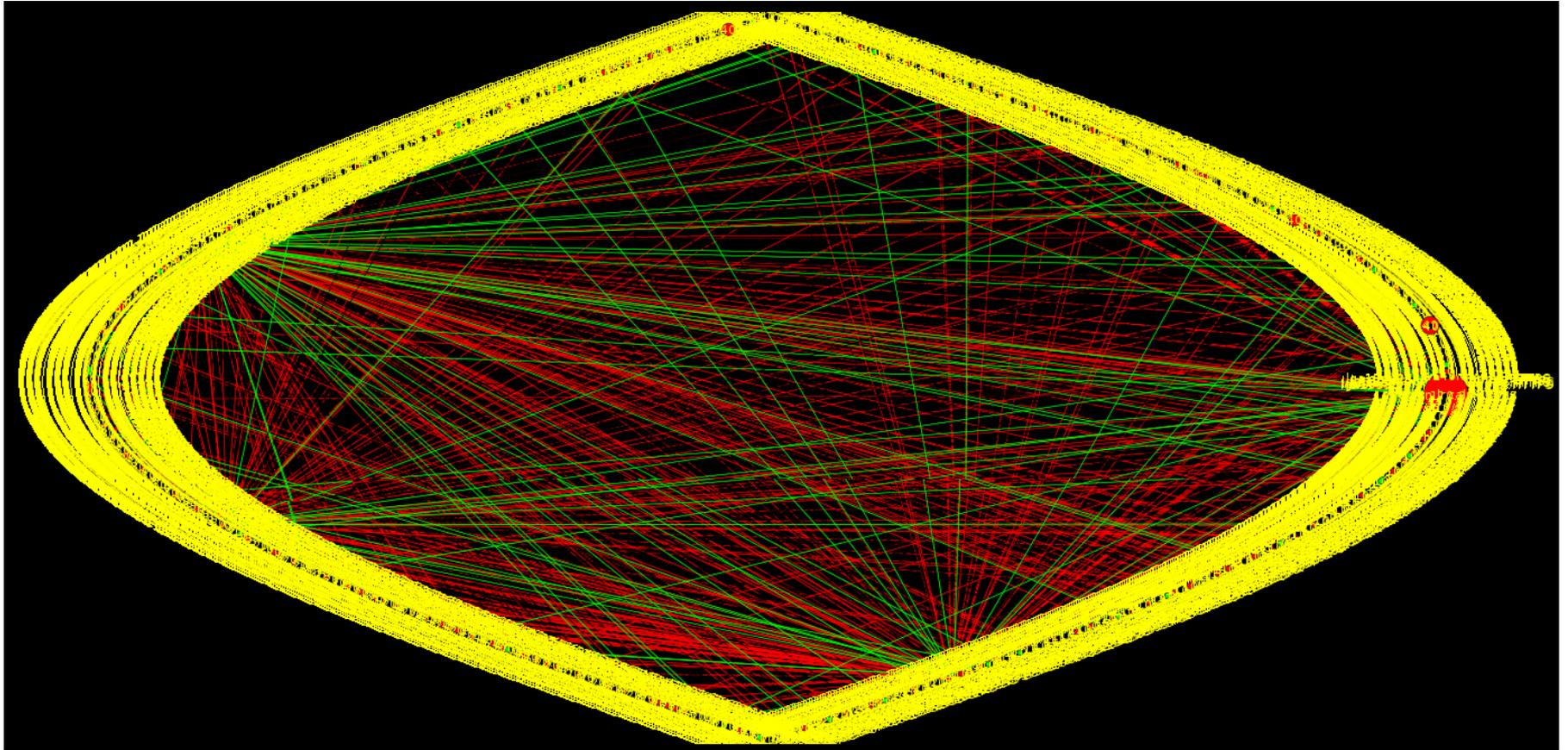
MPTCP Future

MPTCP and ... Network monitoring

- If tool doesn't understand MPTCP, flows look like unrelated TCP streams

What does it look like?

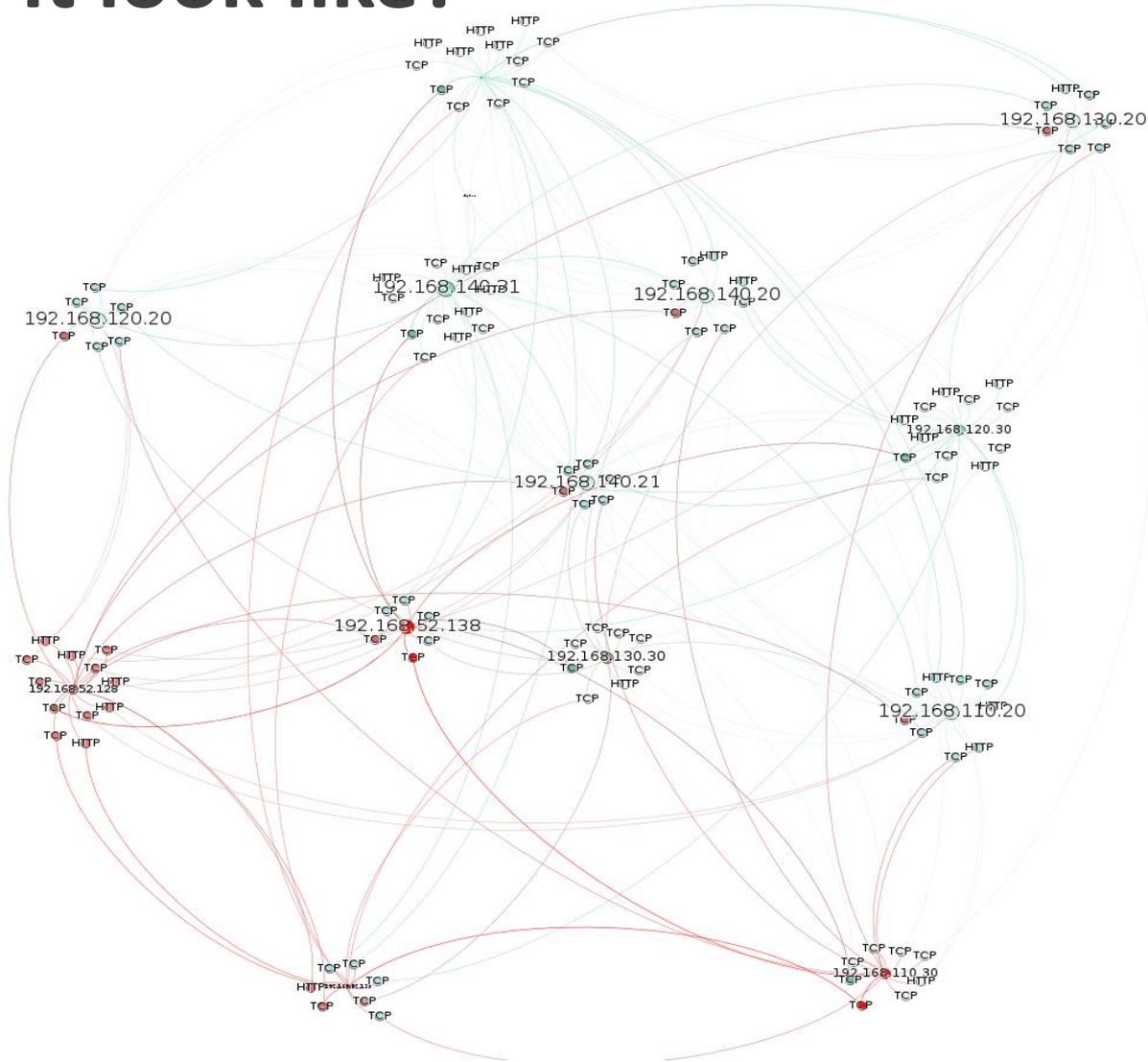
- On the network: If you don't understand



Each yellow blob is actually part of an address label

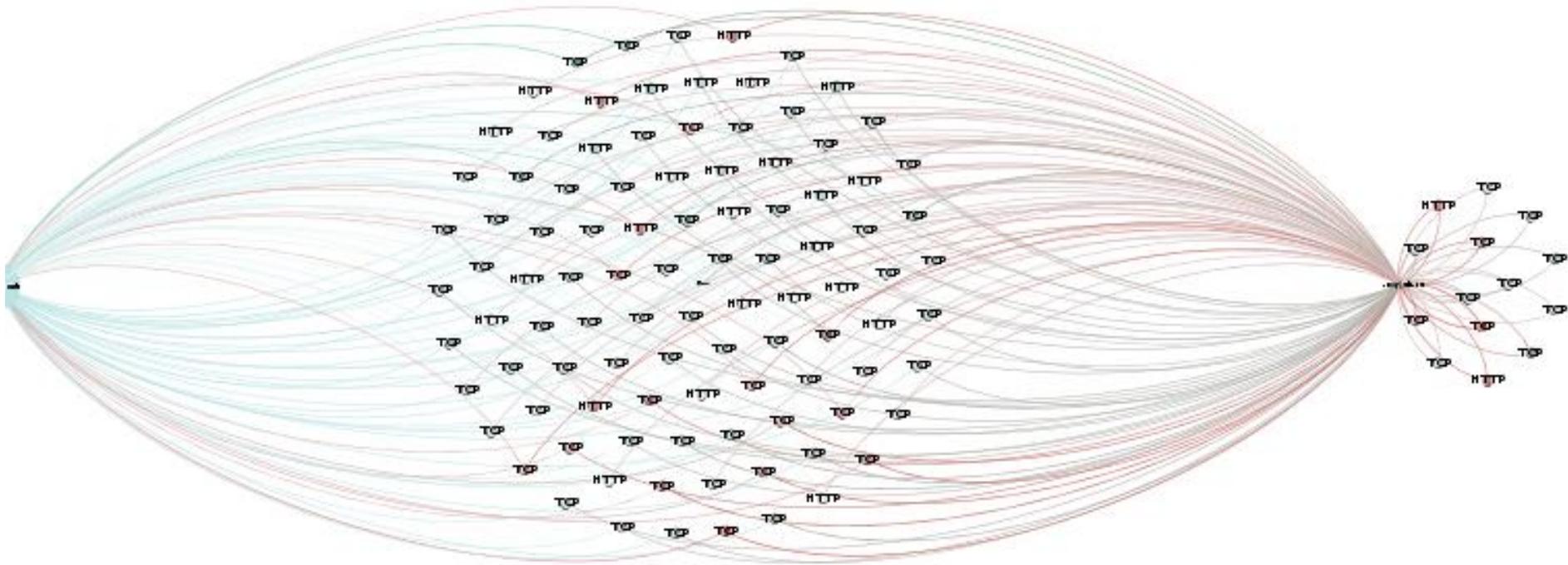
What does it look like?

- 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 Defense - Awareness

- People
- Technology
 - Check support
 - Look for use
- Architecture
 - Terminate it where you terminate SSL

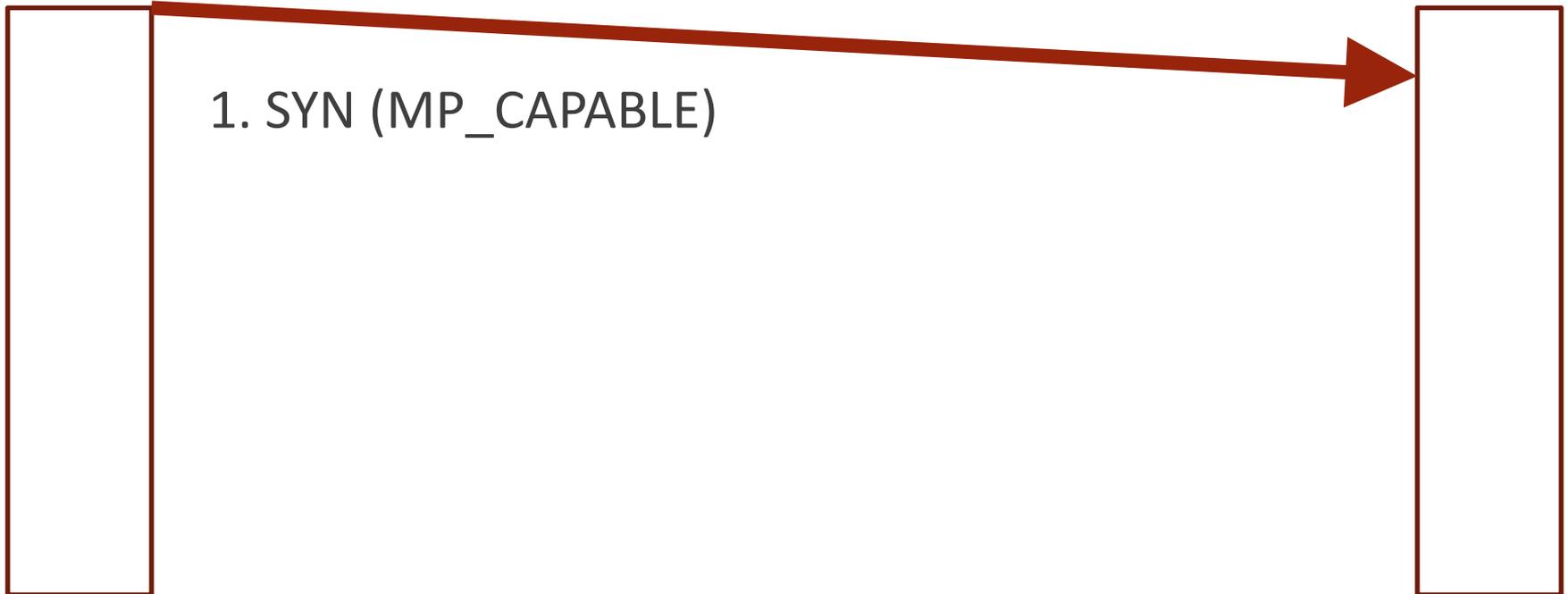
Release Item #3 - Scanner

- Accomplishes three things
 - Test device for *apparent* support
 - Test for *actual* support (as opposed to repeating the option blindly)
 - Test network path allows it to get there

MPTCP Scanner

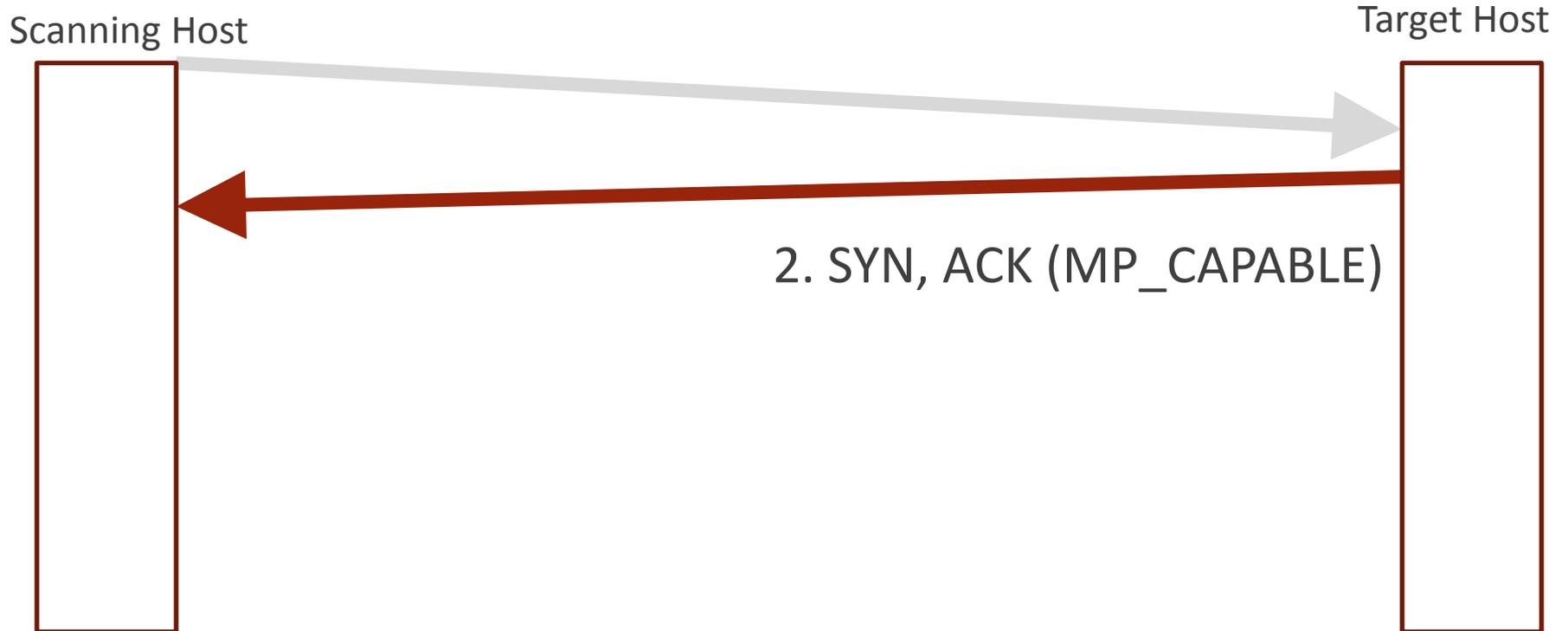
Scanning Host

Target Host



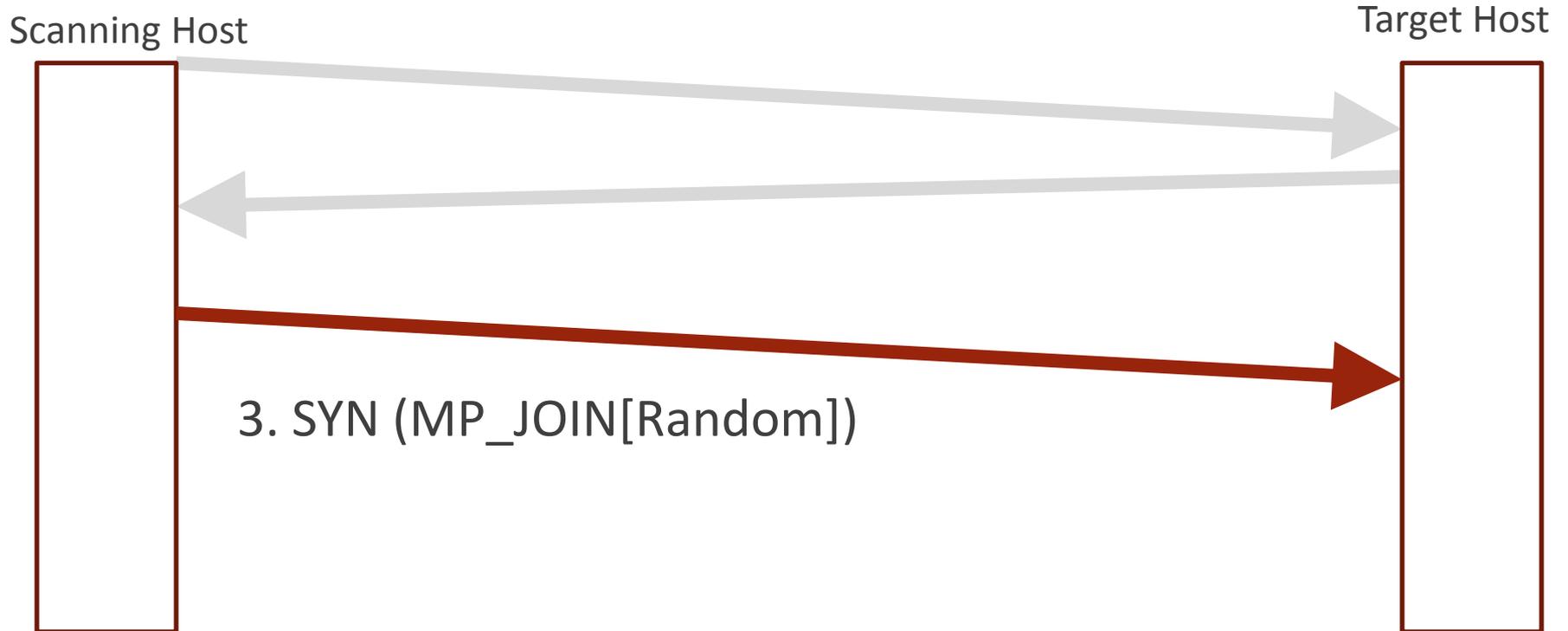
- Send an MP_CAPABLE syn

MPTCP Scanner



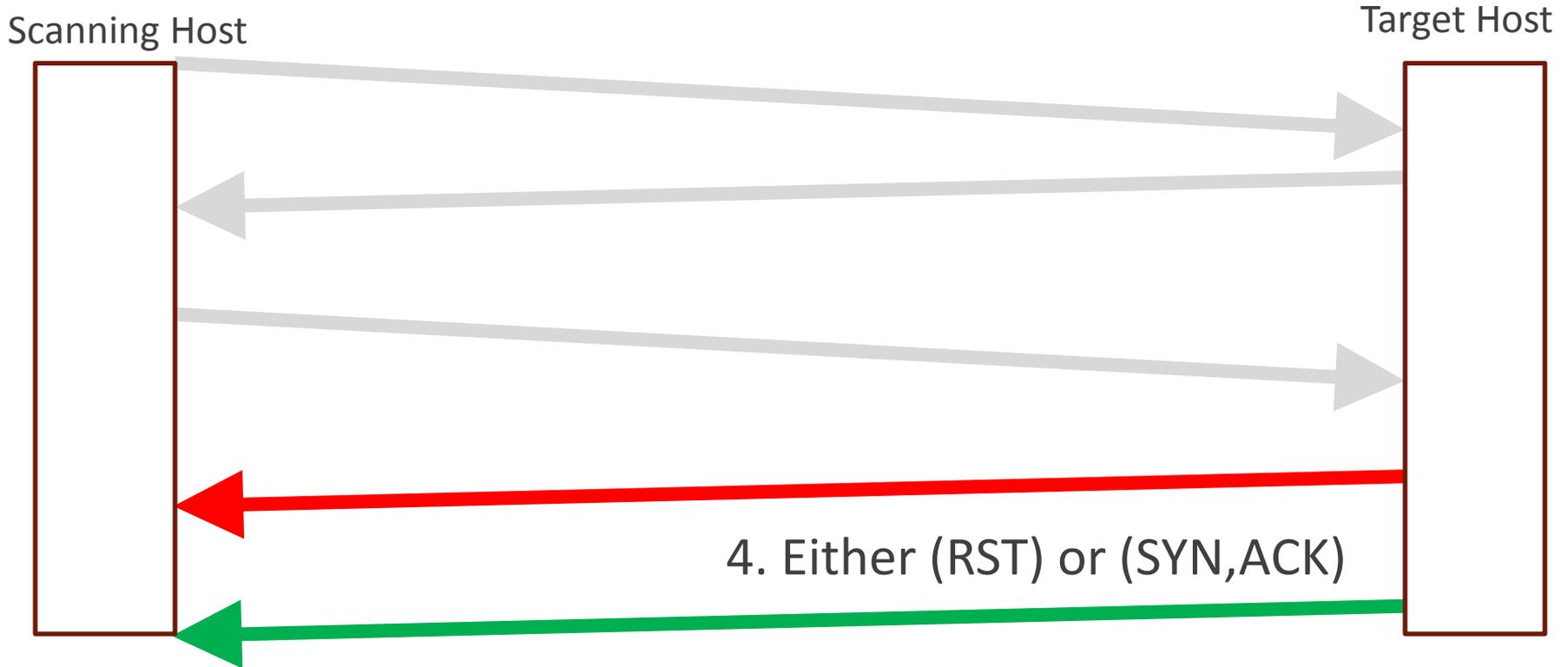
- We got an MP_CAPABLE response.. But is it genuine?

MPTCP Scanner



- Send a join to an invalid connection ID

MPTCP Scanner



- An MPTCP host will RST an invalid join,
- An ACK reply indicates TCP only

MPTCP Stripping

- Transparent proxy on primary path
 - Either no MPTCP support, or only on the one interface

- Firewall rules:

strip-options 30 - iptables,

tcp-options 30 30 clear - Cisco IOS

MPTCP and Active Network Security

- To track & modify MPTCP, you must
 1. Capture the initial handshake
 2. Perform non-trivial calculations to determine
 - Connection membership
 - Correct checksum or modified traffic



Background
Technical Introduction
Key Security Effects
Perimeter Security
Network Management
MPTCP Future

MPTCP and ... Privacy

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

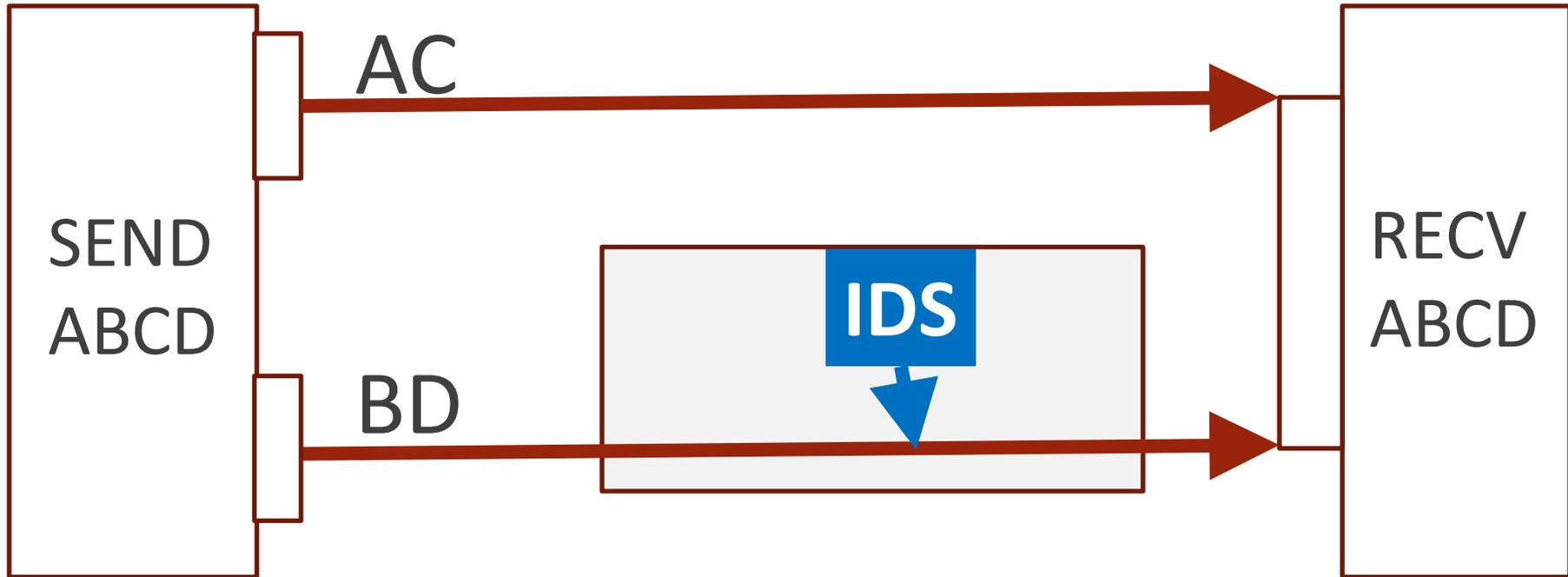
The MPTCP Future

- How do we gain these advantages securely when most things support MPTCP?
- Some changes still need to be made
- Some things will never be the same

Future – Collaborative/Probabilistic IDS

Client

Server



How can the IDS make sense of traffic if it can only ever see fragments?

The MPTCP Future

- What will change in a multipath future that simply cannot work with most existing security models?
 - Split trust crypto
 - Multipath agility
- Some research into privacy effects already underway

Other Ideas

- Making proxy tools to undertake these attacks
- Stream hopping (ala FHSS)
- Mixing it with TOR
- Building distributed networks on MPTCP-like packets
- Hijack connections on fastclose

Conclusions

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

Participation/Competition

PCAP info @

<http://bit.ly/UYluPp>

#BHMPTCP

Also see our workshop at DEFCON's
Wall of Sheep (Sunday)



Questions?

Catherine Pearce

@secvalve

cpearce@neohapsis.com

Patrick Thomas

@coffeetocode

pthomas@neohapsis.com

Downloads on Github:

<https://github.com/Neohapsis/mptcp-abuse>

More stuff will be released @

<http://labs.neohapsis.com>

References 1

- Implementations & vendors
 - Linux (UCLouvain, multipath-TCP.org) (<http://github.com/multipath-tcp/>)
 - MPTCP Scapy - <https://github.com/nimai/mptcp-scapy>
 - BSD - <http://caia.swin.edu.au/urp/newtcp/mptcp/>
 - Android - <http://multipath-tcp.org/pmwiki.php/Users/Android>
 - Apple - https://opensource.apple.com/source/xnu/xnu-2422.1.72/bsd/netinet/mptcp*
 - Cisco - <http://www.cisco.com/c/en/us/support/docs/ip/transmission-control-protocol-tcp/116519-technote-mptcp-00.html>
- MPTCP Security
 - IETF MPTCP workinggroup - Threat Analysis for TCP Extensions for Multipath Operation with Multiple Addresses - <http://tools.ietf.org/html/rfc6181>
 - Ford, A. (2010), 'Multipath TCP Security Issues : Current Solution Space Why we need security'.
 - Zhuang, R. 2013. Investigating the Application of Moving Target Defenses to Network Security. ISRCS, 2013 6th International Symposium on. Available at <http://people.cis.ksu.edu/~zhangs84/papers/ISRCS13.pdf>

References 2

■ Theroretical backgrounds

- Stability issues - Kelly, F. & Voice, T., 2005. Stability of end-to-end algorithms for joint routing and rate control. ACM SIGCOMM Computer Communication Review, 35(2), pp.5–12.
- Routing and congestion control - Key, P., Massoulié, L. & Towsley, D., 2006. Combining multipath routing and congestion control for robustness. In Information Sciences and Systems, 2006 40th Annual Conference on. IEEE, pp. 345–350.
- Honda, M., Nishida, Y. & Raiciu, C., 2011. Is it still possible to extend TCP? Proc. ..., p.181. Available at: <http://conferences.sigcomm.org/imc/2011/docs/p181.pdf> .

■ MPTCP Background & development:

- 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/>
- IANA TCP Options - <https://www.iana.org/assignments/tcp-parameters/tcp-parameters.xhtml>

References 3

■ Other

- Multi Network Manager - Evensen, K.R. MULTI Network Manager (MNM), 2013.
<http://github.com/kristrev/multi>

- See the Whitepaper for MANY more references