



SharkFest'19 US



My TCP ain't your TCP - ain't no TCP?

How new implementations speed up the internet
... and make engineers drink

Simon Lindermann

Miele & Cie KG / Freelancer



Demo Traces



<https://cloud.local-area.network/index.php/s/7Ojmw9hhnDf9yBb>

<https://bit.ly/2Iae7A7> (Shortlink)



About me?



- Working and learning in IT since 2006
 - Employer: Miele Germany
 - Network Architect
 - Part time freelancer

Contact



@SimonLindermann



sl@local-area.network

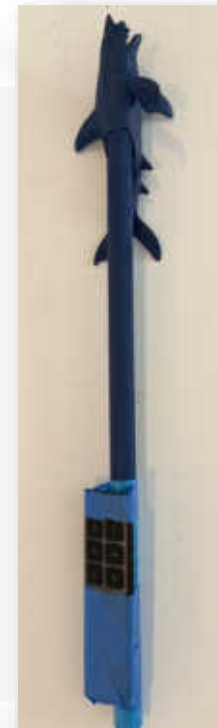


Shark Pointer





LaserShark Pointer



What would an engineer do?



No revolution, but evolution



Algorithms

- TCP Tahoe
- TCP Reno
- TCP New Reno
- (TCP SACK)

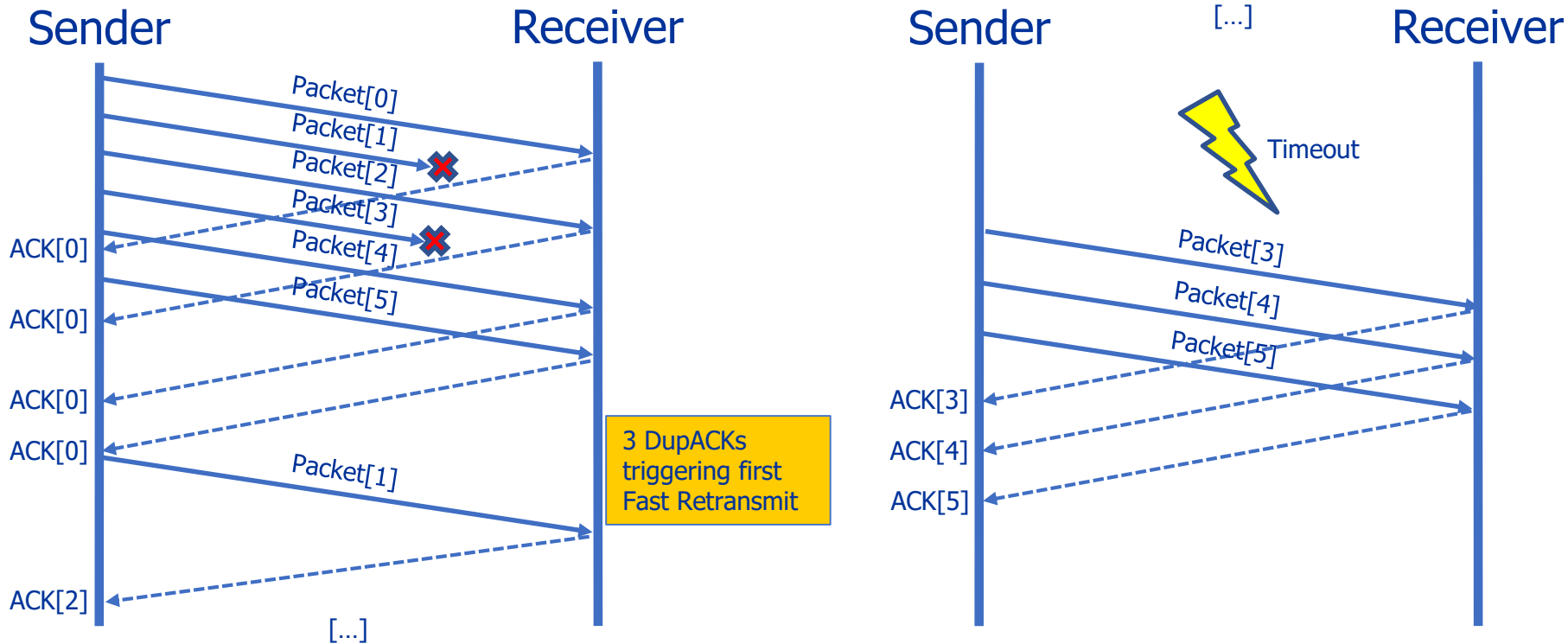
Components

- Slow Start
 - Additive Increase Multiplicative Decrease (AIMD)
 - Fast Retransmit
 - Fast Recovery
 - Partial ACK
- Check out my talk from 2018 for this stuff!*

- Selective ACK

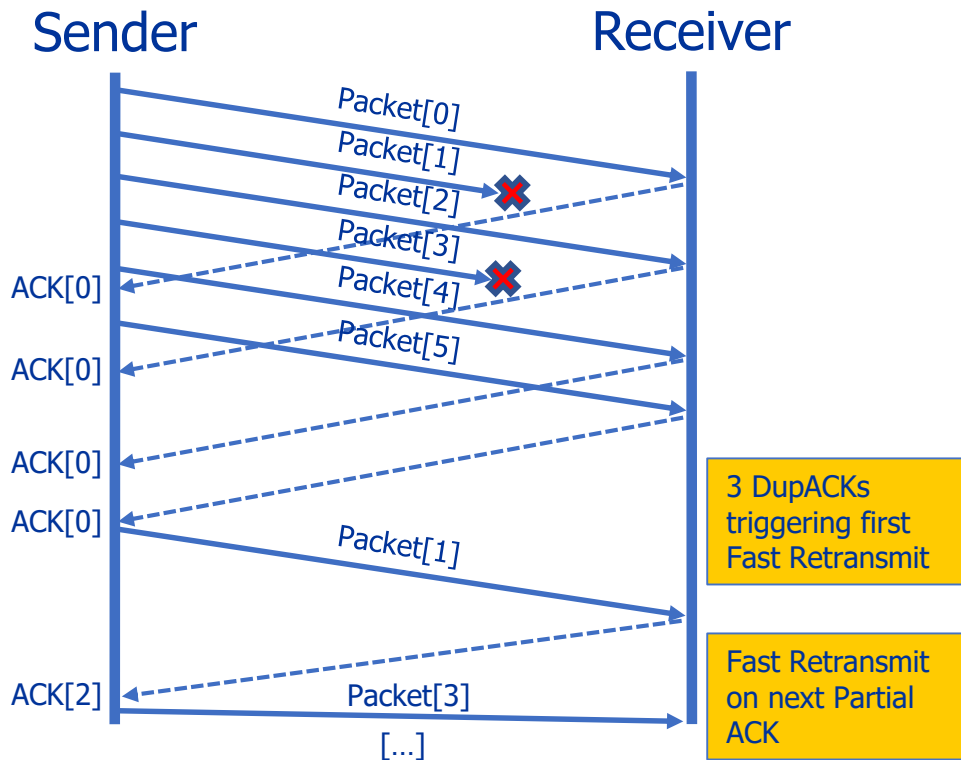


Signaling loss in the old days...





TCP New Reno: Partial ACK



- Partial ACKs trigger Fast Retransmits of multiple lost segments
- Second lost segment gets retransmitted immediately after the **first** Partial ACK



TCP SACK



RFC 2018

“TCP may experience poor performance when multiple packets are lost from one window of data.”

[...]

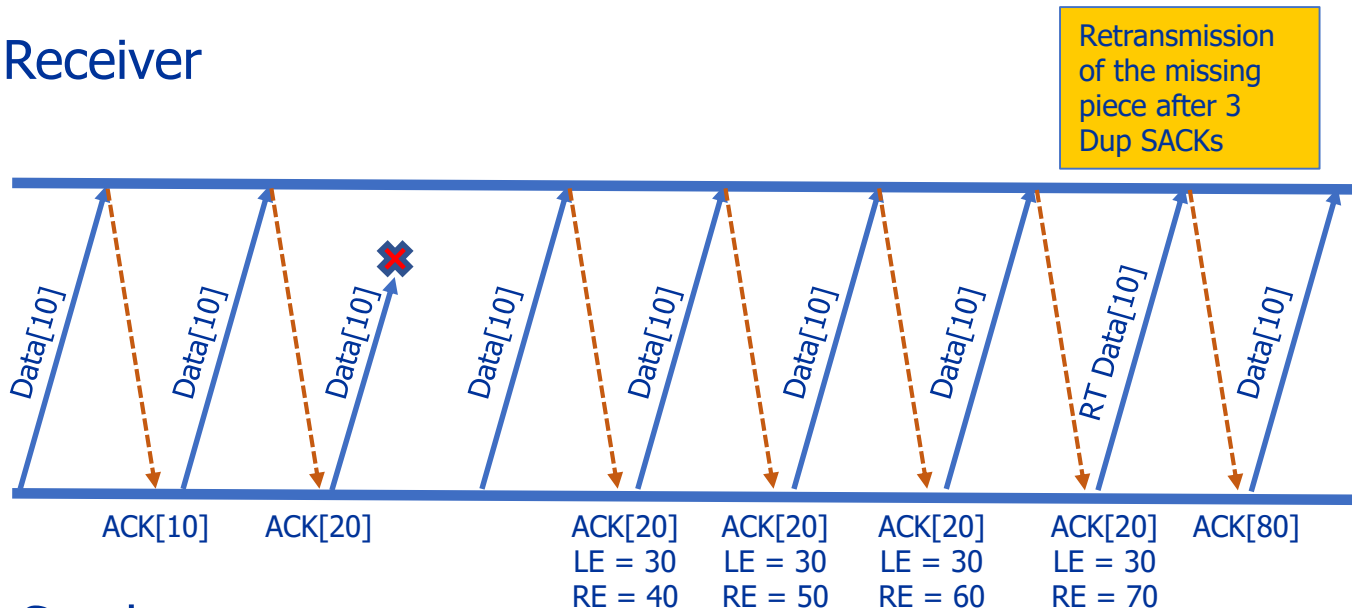
“The receiving TCP sends back SACK packets to the sender informing the sender of data that has been received. The sender can then retransmit only the missing data segments.”



TCP SACK



Receiver



Sender



Demo



- Open Trace in your Wireshark:

`"sack_and_dsack.pcap"`



OK, that's the old stuff!



Lets think outside the box!





But now - Revolution!



Multipath TCP (MPTCP)

- RFC 6182 / 6824
- Latest implementations (suggested)
<https://tools.ietf.org/html/draft-ietf-mptcp-rfc6824bis-17>



Motivation

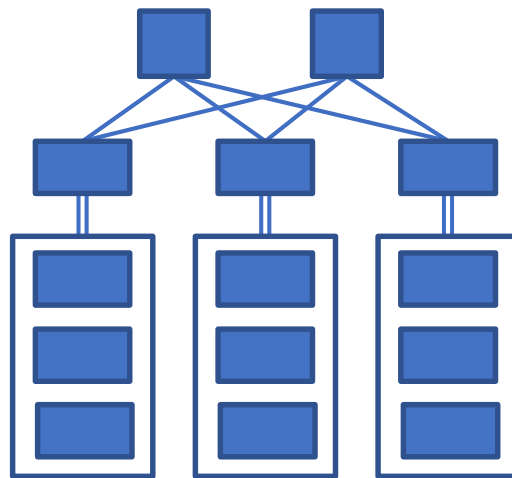


Networks becoming multipath

- Link aggregation
- High availability

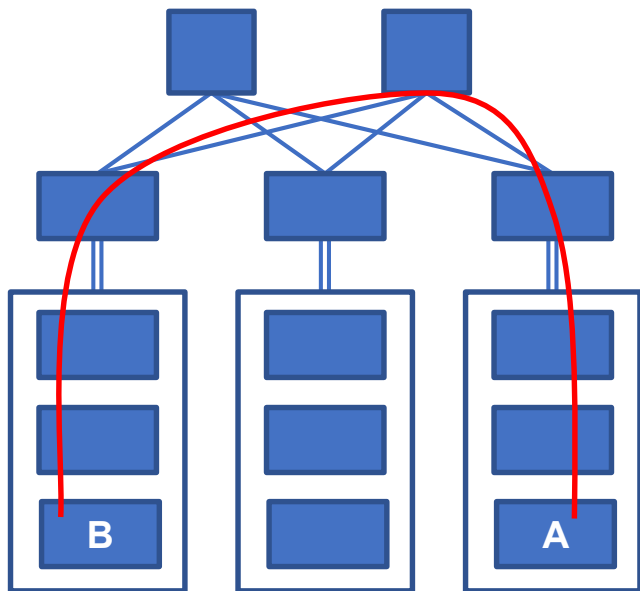
Redundant networks

- Mobile phones utilizing Wifi & 4G
- Multi-homed Servers





Flow distribution (1/2)

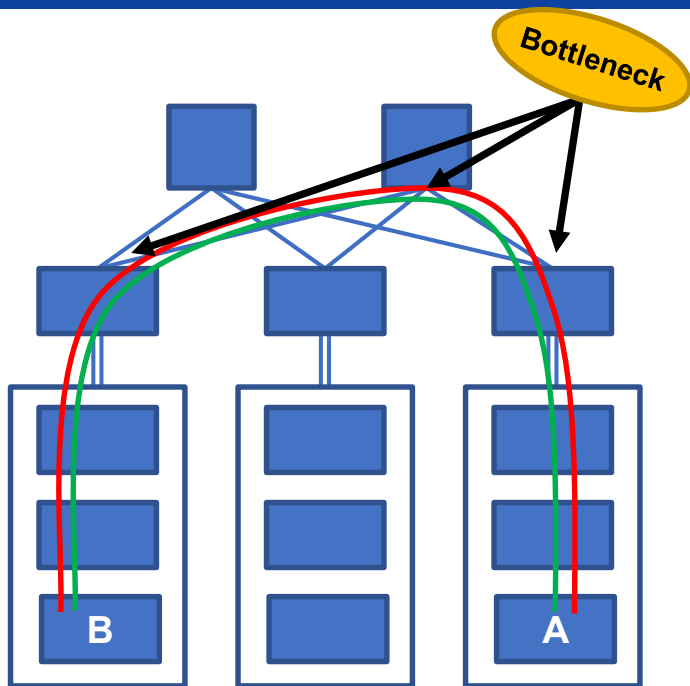


Server A to Server B

- Load balancing across links usually based on
 - MAC addresses
 - IP addresses
- Single TCP flow



Flow distribution (2/2)



Server A to Server B

- Load balancing across links usually based on
 - MAC addresses or
 - IP addresses
- 2nd TCP flow, same SRC/DST



TCP Statements



- TCP only uses a single path regardless of the network topology
- It's always tied to a single SRC and DST address of client and server
 - If one or the other changes → Connection break!



Deployment goals of recent MPTCP implementation



- TCP needs to evolve and utilize multiple paths for the same data transport
- It **MUST** meet the following criteria:
 - 1) Support unmodified applications
 - 2) Work on today's networks
 - 3) Work whenever TCP works, or fallback to TCP



Easy peasy!



... or not?



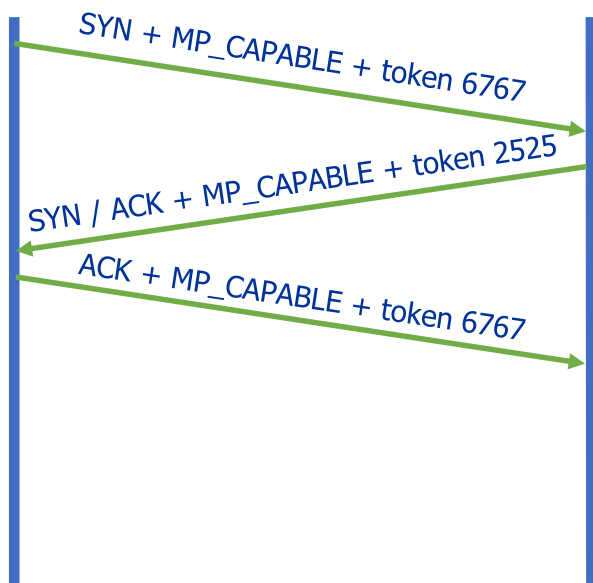
MPTCP Session Setup



Client



Server

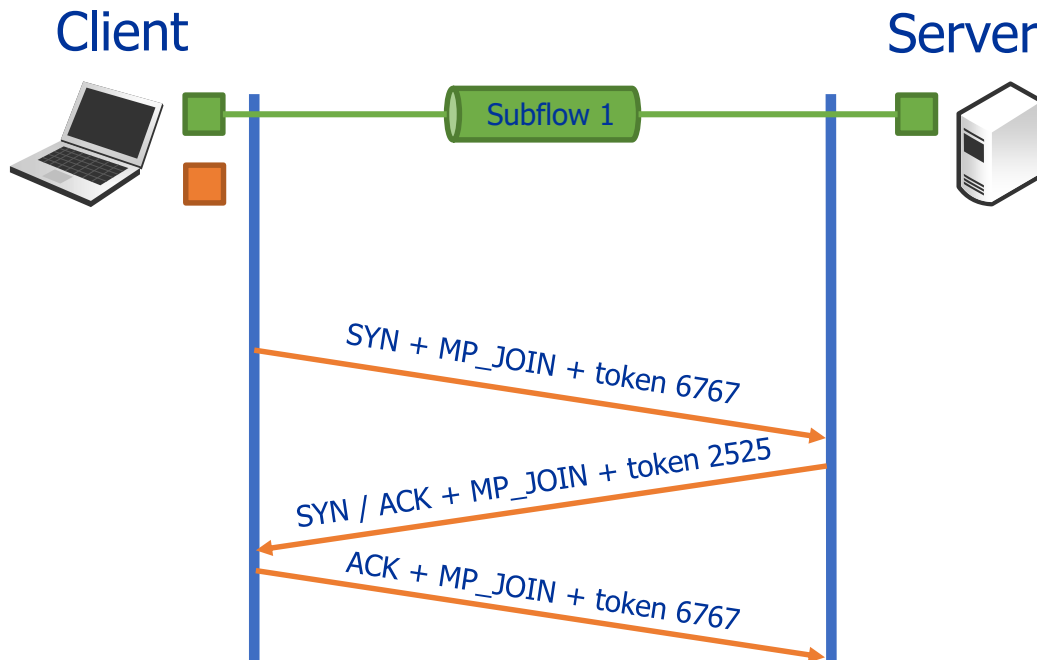


Connection identifier

- analog to ephemeral TCP ports
- Used to associate multiple subflows



MPTCP Session Setup

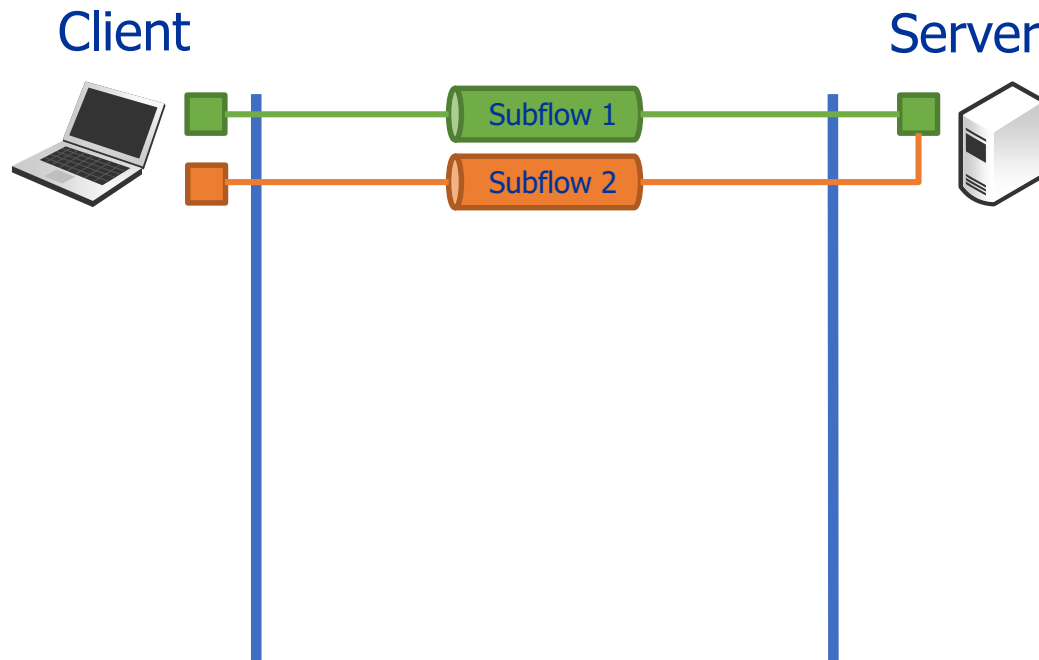


Connection identifier

- similar to ephemeral TCP ports
- Used to associate multiple subflows



MPTCP Session Setup



Connection identifier

- similar to ephemeral TCP ports
- Used to associate multiple subflows



The packet



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Byte 0				Byte 1				Byte 3				Byte 4																			
Version		Header len		TOS				Total length																							
Identification				Flags		Fragment Offset																									
TTL				Protocol				Header Checksum																							
Source IP																															
Destination IP																															
Source Port								Destination Port																							
SEQ Number																															
ACK Number																															
Header len		Reserved		Code bits				Window size																							
Checksum								Urgent pointer																							
Options																															
Data																															



The packet



0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Byte 0				Byte 1				Byte 3				Byte 4																			
Version		Header len		TOS				Total length																							
Identification								Flags		Fragment Offset																					
TTL				Protocol				Header Checksum																							
Source IP																															
Destination IP																															
Source Port								Destination Port																							
SEQ Number																															
ACK Number																															
Header len		Reserved		Code bits				Window size																							
Checksum								Urgent pointer																							
Options																															
Data																															



The packet

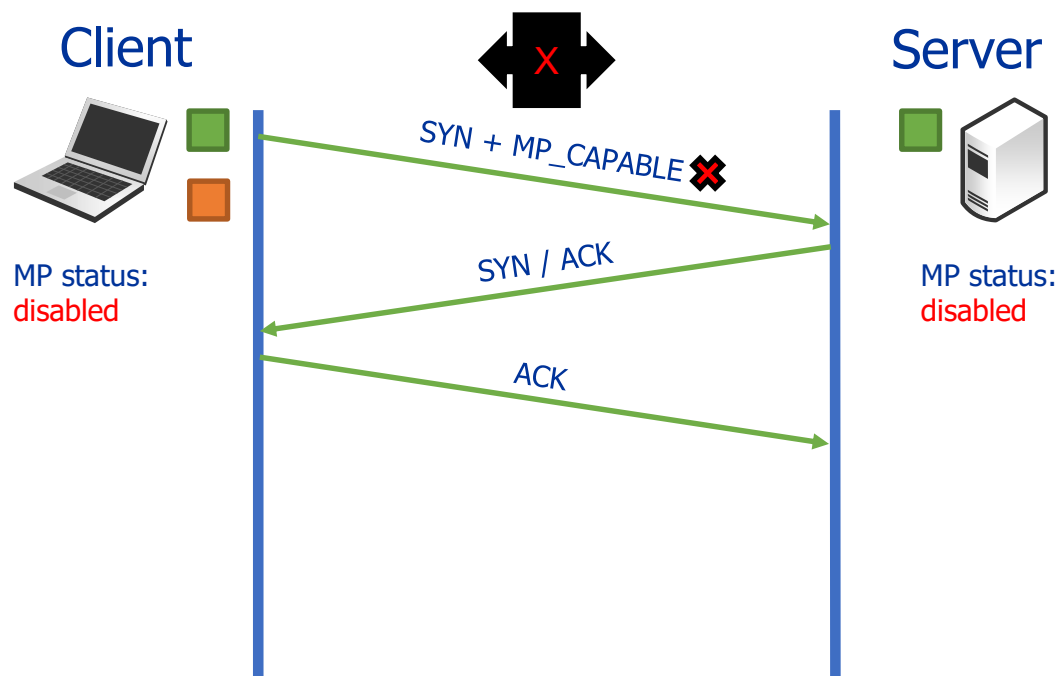


0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31
Byte 0				Byte 1				Byte 3				Byte 4																			
Version		Header len		TOS				Total length																							
Identification						Flags		Fragment Offset																							
TTL				Protocol				Header Checksum																							
Source IP																															
Destination IP																															
Source Port								Destination Port																							
SEQ Number																															
ACK Number																															
Header len		Reserved		Code bits				Window size																							
Checksum								Urgent pointer																							
Options																															
Data																															





Nasty middle boxes

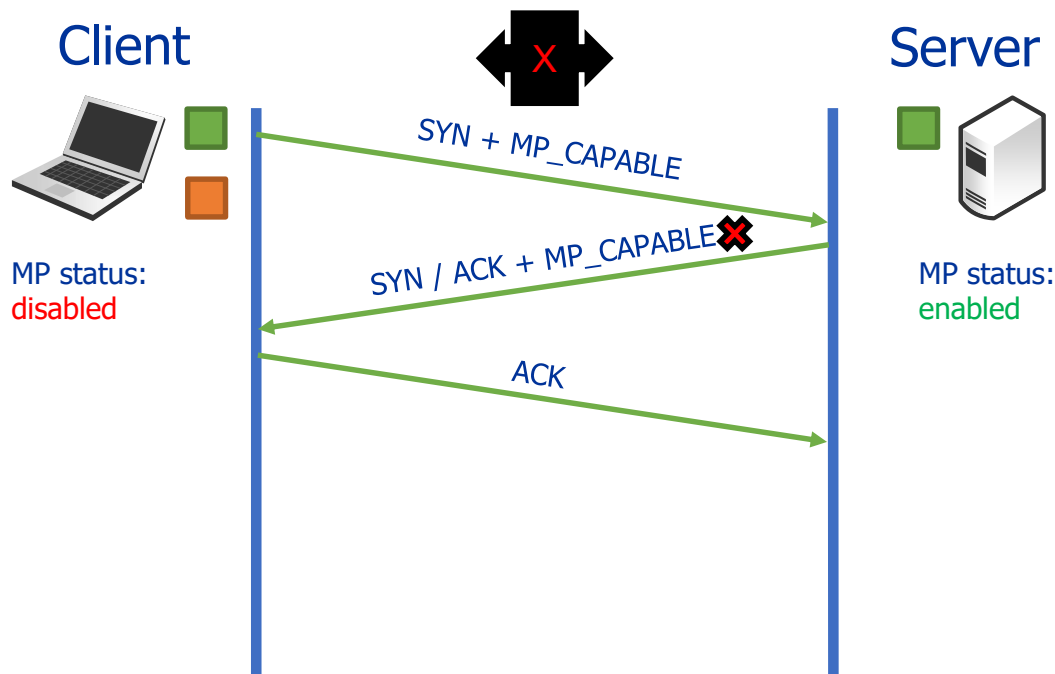


Middle boxes can drop TCP options:

a) SYN MP_CAPABLE



Nasty middle boxes

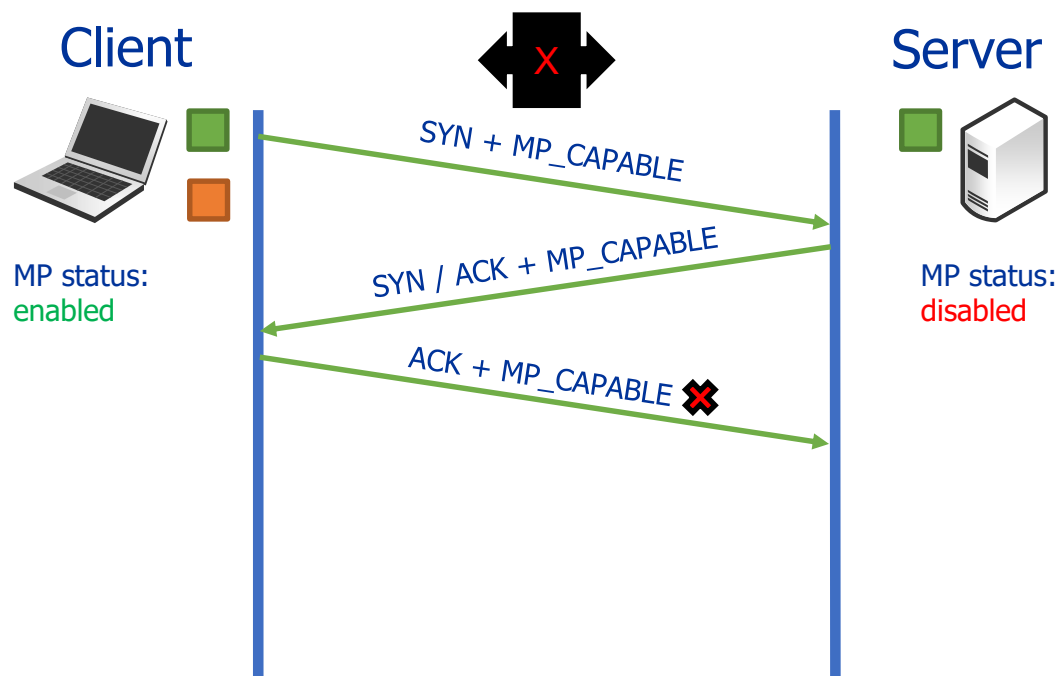


Middle boxes can drop TCP options:

b) SYN / ACK MP_CAPABLE



Nasty middle boxes

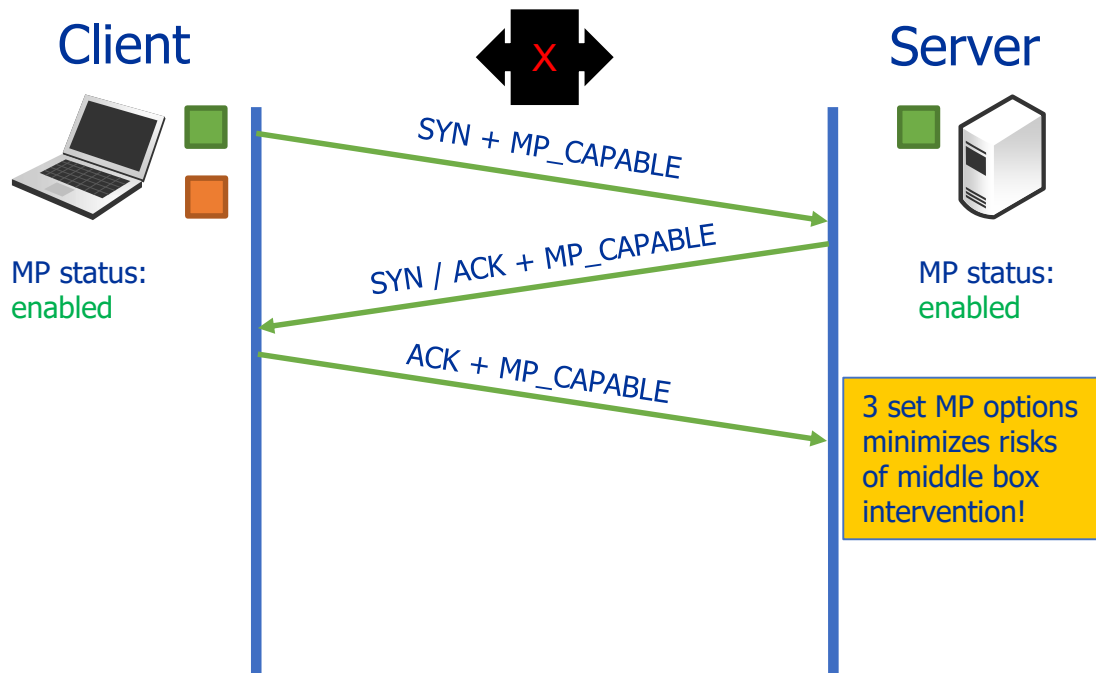


Middle boxes can drop TCP options:

c) ACK MP_CAPABLE



Nasty middle boxes



Summary

Don't expect the path between two endpoints is always the same!

Have a fallback option available!



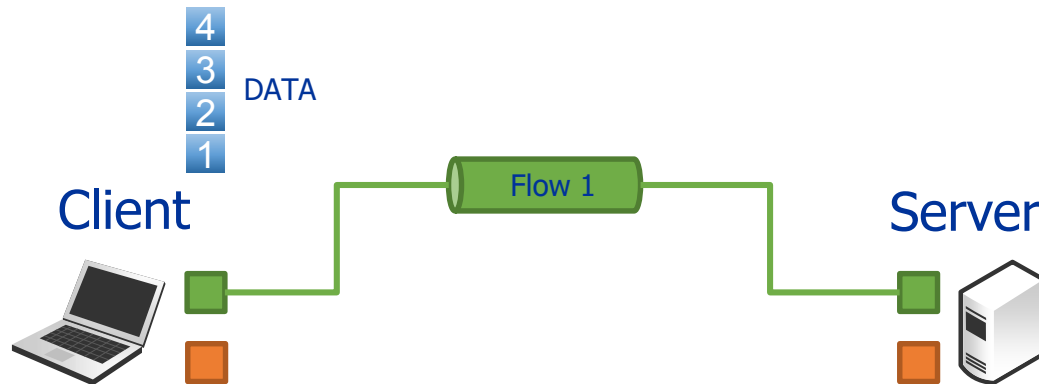
Demo



- Live Demo



How to send Data?

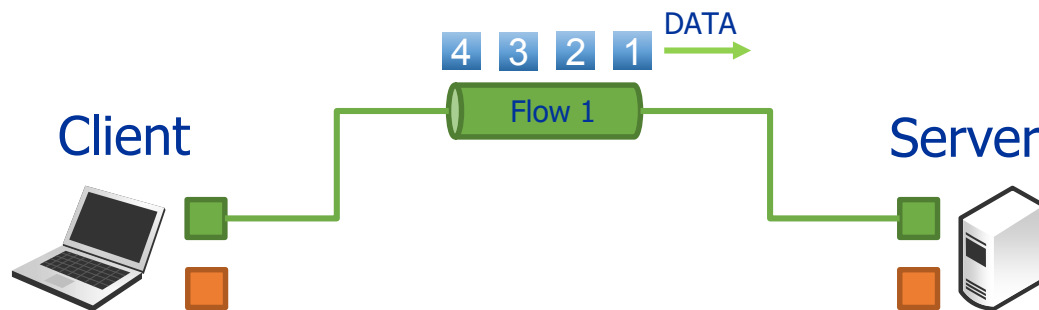


With regular TCP

Application passes Data to MPTCP



How to send Data?

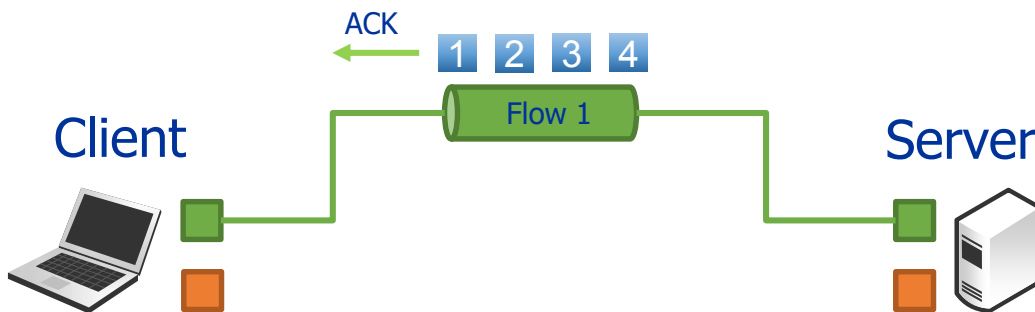


With regular TCP

Data will be send over a single connection in the order of appearance



How to send Data?

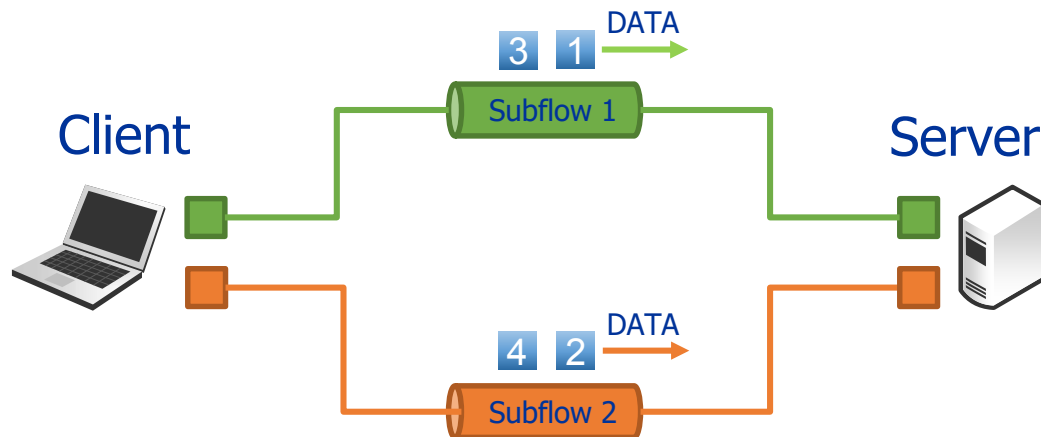


With regular TCP

ACKs are going back in the order of received data



How to send Data?



MPTCP Strawman design

Distribute Data evenly over all connections

No problem so far...



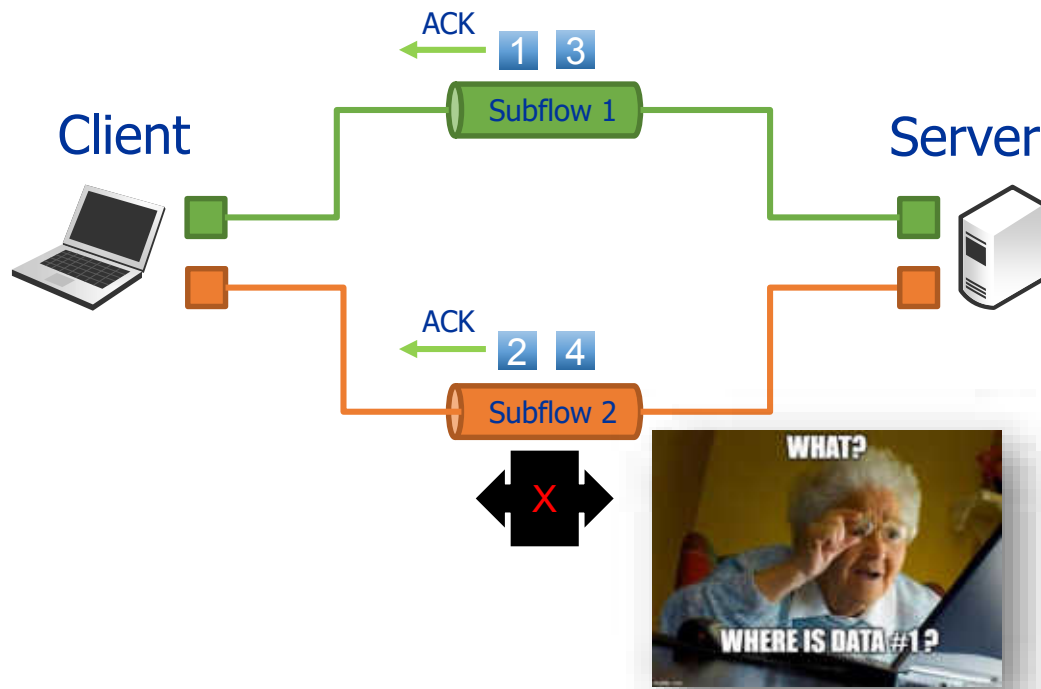
How to send Data?



MPTCP Strawman design

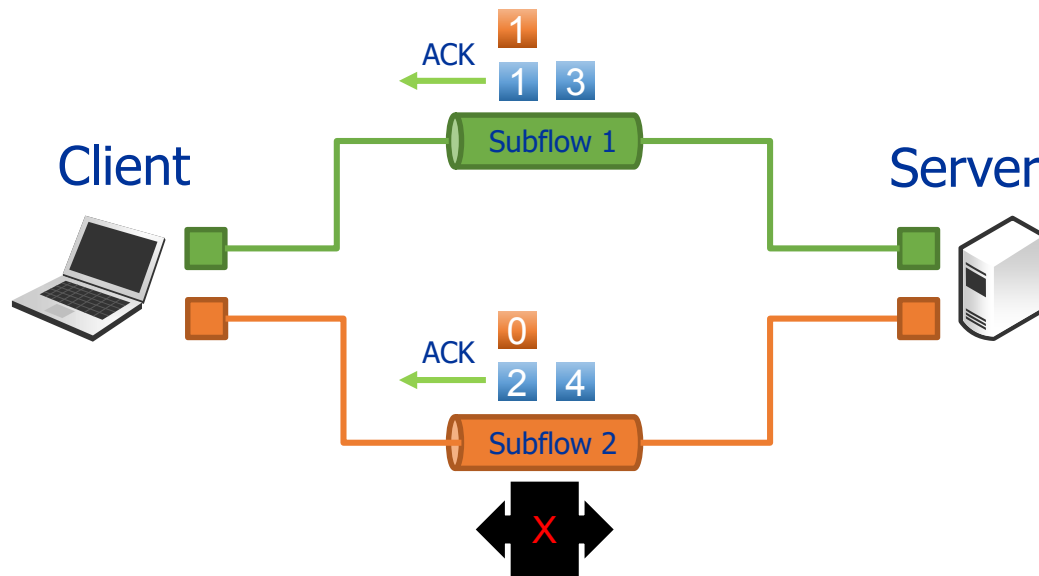
Oh no, a middle box!

- Path 2 did not see Data 1
- ACK 2 cumulatively acknowledges Data 1 & 2





How to send Data?



MPTCP Strawman design

Chances are middle box will:

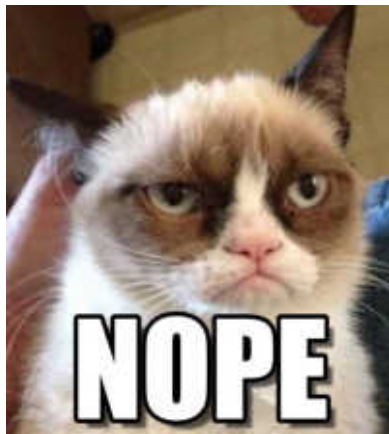
- a) Drop the ACK
- b) Correct the ACK
- c) RST the connection



The NOPES



- Don't use TCP Sequence numbers across multiple flows
- Don't expect the network to be as smart as your clients/servers



*RIP Grumpy Cat



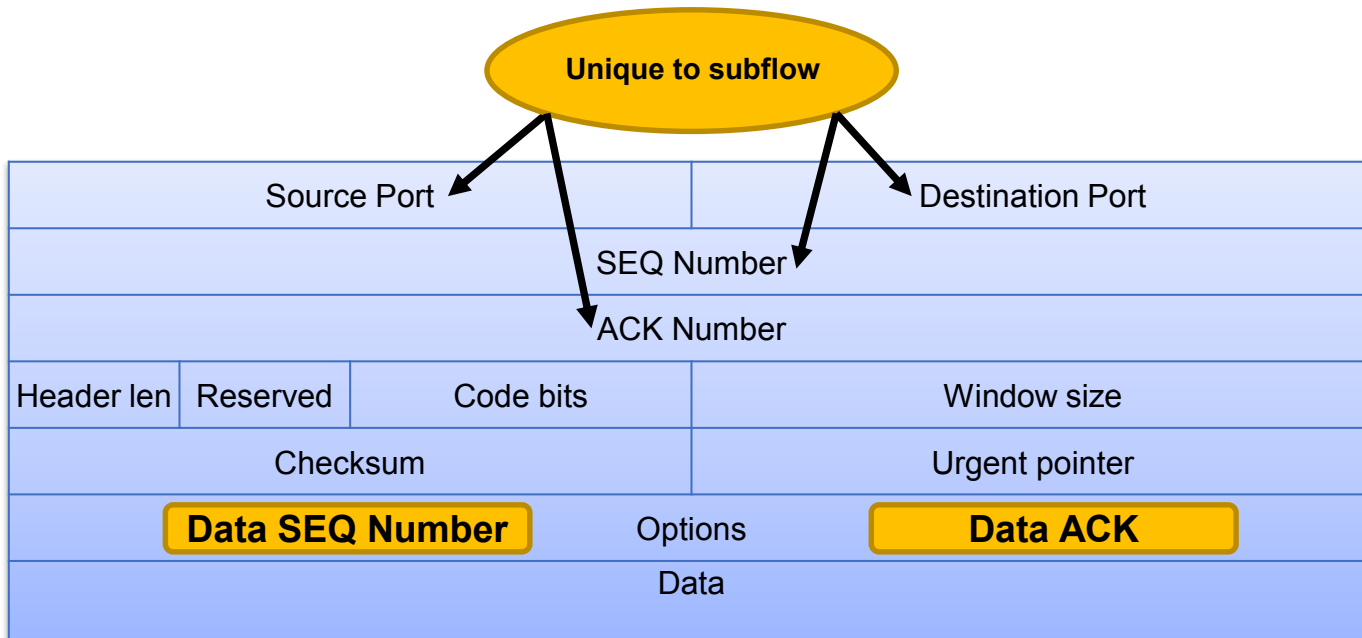
Instead



- Own sequence numbers for each subflow
 - Make MPTCP look like TCP with no gaps
- Additional data sequence numbers for MPTCP
 - Because reordering can still happen
- Additional data ACK for flow control

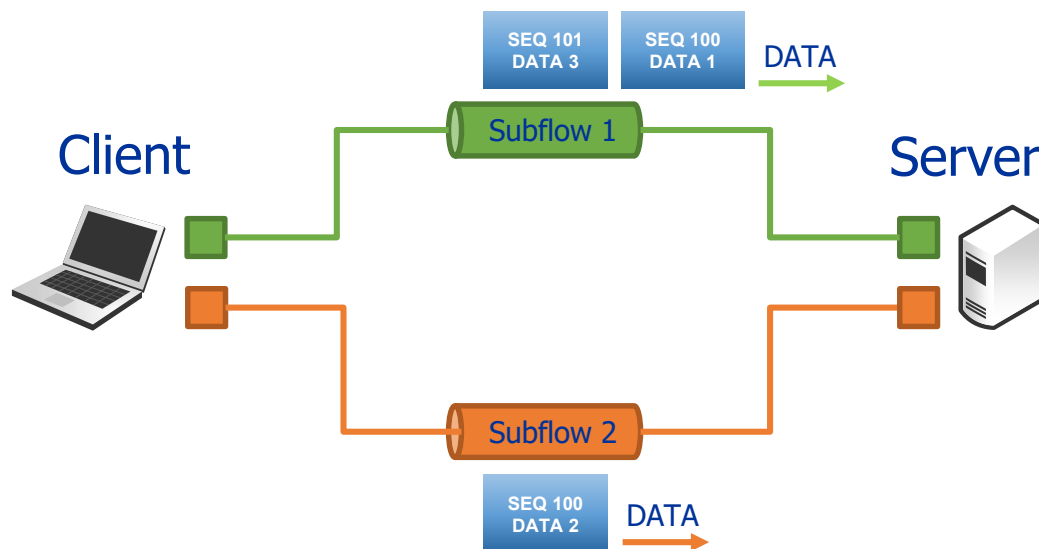


MPTCP Header





Sending data across subflows



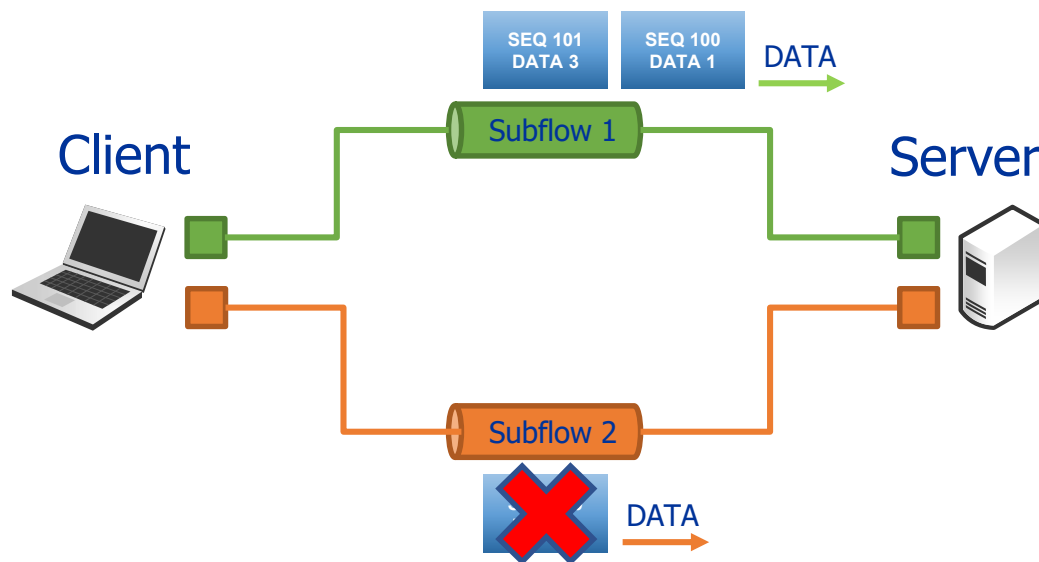
MPTCP data transmission

Sequence Numbers are unique per subflow

Data Sequence Numbers are shared across all subflows



How to react on packet loss?

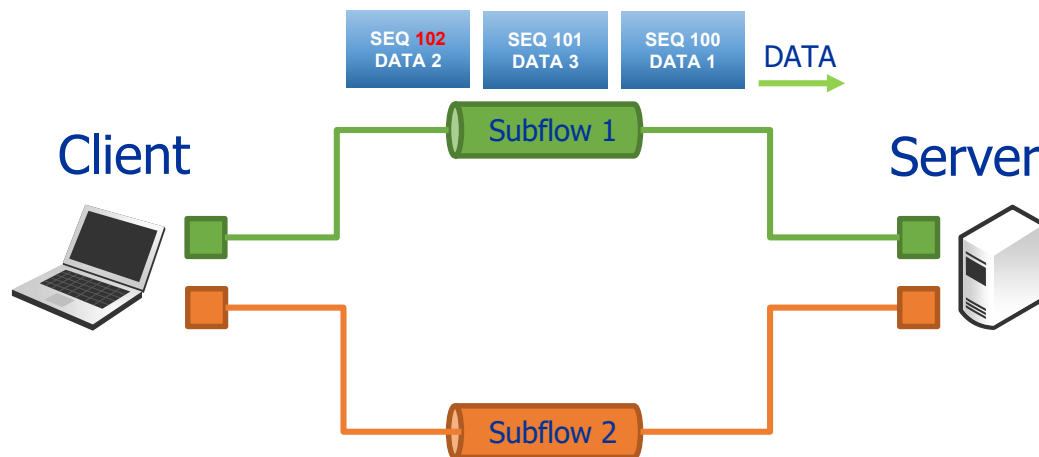


MPTCP data transmission

Subflow 2 experiences packet loss



How to react on packet loss?



MPTCP data transmission

Data will be retransmitted on subflow 1



Demo



- Open Trace in your Wireshark:

“mptcp_client01.pcap”



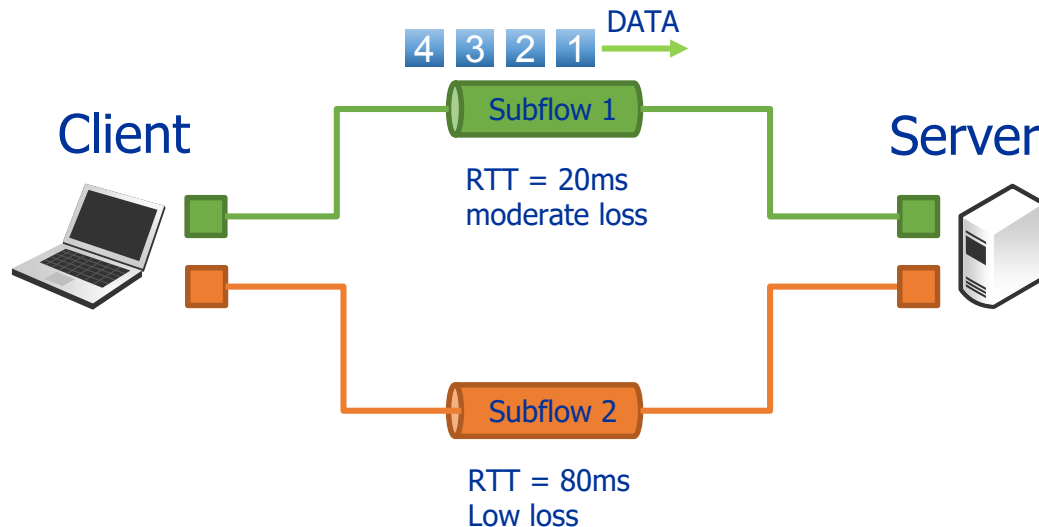
MPTCP Options



Value	Symbol	Name
0x0	MP_CAPABLE	Multipath Capable
0x1	MP_JOIN	Join Connection
0x2	DSS	Data Sequence Signal (Data ACK and data sequence mapping)
0x3	ADD_ADDR	Add Address
0x4	REMOVE_ADDR	Remove Address
0x5	MP_PRIO	Change Subflow Priority
0x6	MP_FAIL	Fallback
0x7	MP_FASTCLOSE	Fast Close
0xf	(PRIVATE)	Private Use within controlled testbeds



MPTCP Scheduler



Which path to use when?

Use the least congested path with the lowest RTT

However:

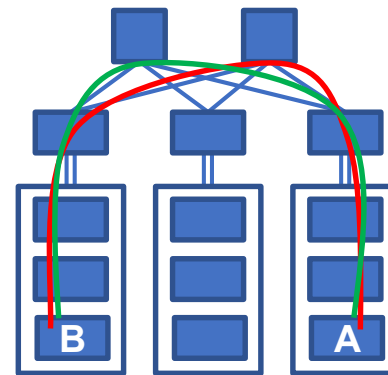
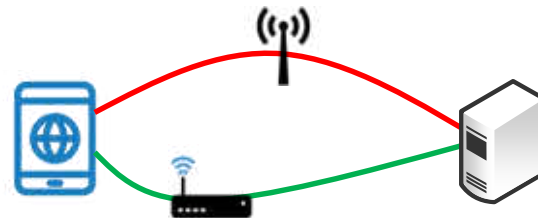
MPTCP throughput should never be less than best possible TCP throughput on a single path



Main MPTCP Use Cases



- Mobile phone
 - Improved stability
- Datacenters
 - Improved throughput





Who is using it today?



- Linux kernel (MultipathTCP-Linux)
- Apple iOS and macOS
- Citrix load balancers
- FreeBSD (FreeBSD-MPTCP)
- Oracle Solaris
- ...



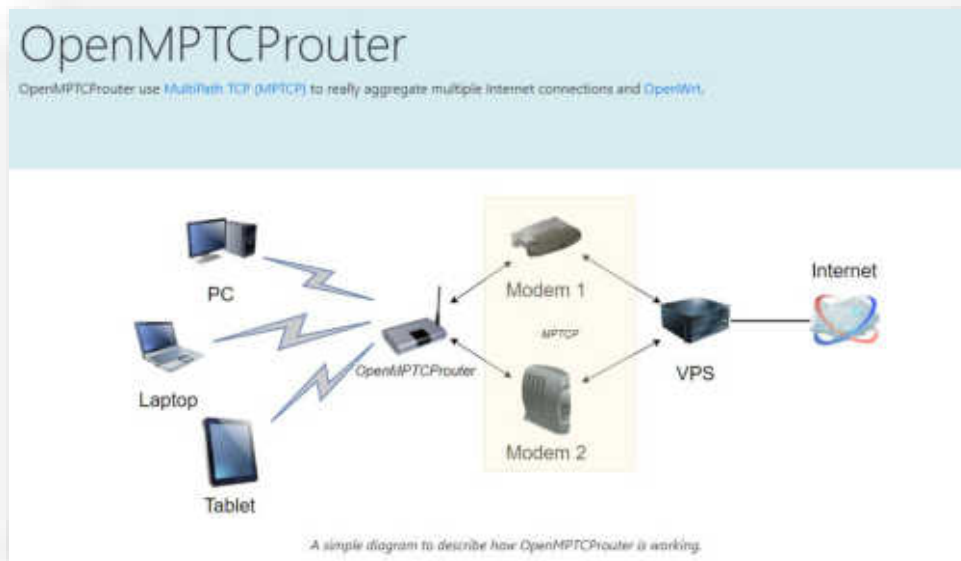
You might try!



- MPTCP doesn't require the application to be adjusted
- You do not need to change your network



Want to play at home?



<https://www.openmptcprouter.com/>



References



- <https://tools.ietf.org/html/rfc6182>
- <https://tools.ietf.org/html/draft-ietf-mptcp-rfc6824bis-17>
- <https://tools.ietf.org/html/rfc8041>

- <https://multipath-tcp.org/>
- <https://youtu.be/k-5pGlibiB3U>
- <https://youtu.be/bwh5pr2uxgQ>
- <https://arxiv.org/abs/1601.06043>

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



Thank you



Questions? Compliments? Wisdoms?

Please use the Guidebook App to provide feedback.

Contact



@SimonLindermann



sl@local-area.network