SharkFest '17 Europe

My TCP ain't your TCP

Stack behavior back then and today

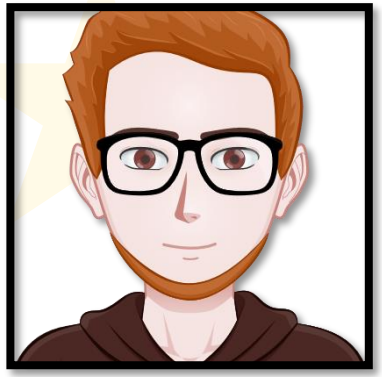
Simon Lindermann

Miele & Cie KG

9th November 2017

About me?

- Working and learning in IT since 2006
  - Employer: Miele Germany
  - Network Engineer
  - Part time freelancer
- As a balance to computer stuff:



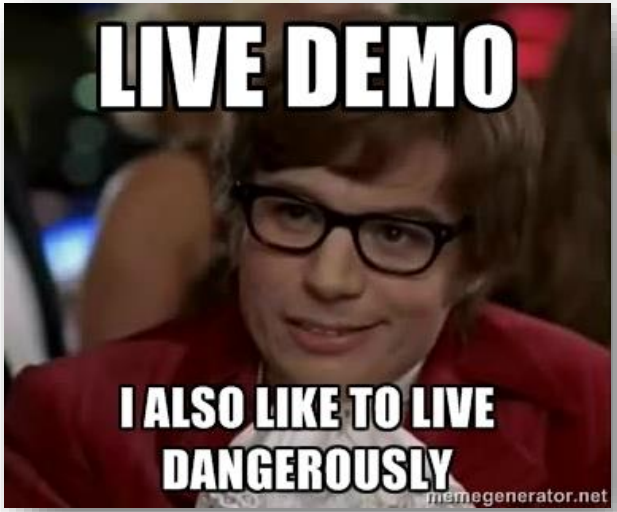
Retrospective

- Traces shown during Demo
  - https://cloud.local-area.network/index.php/s/tzNNwmbzwIkDSqz
    PW: sf17eu

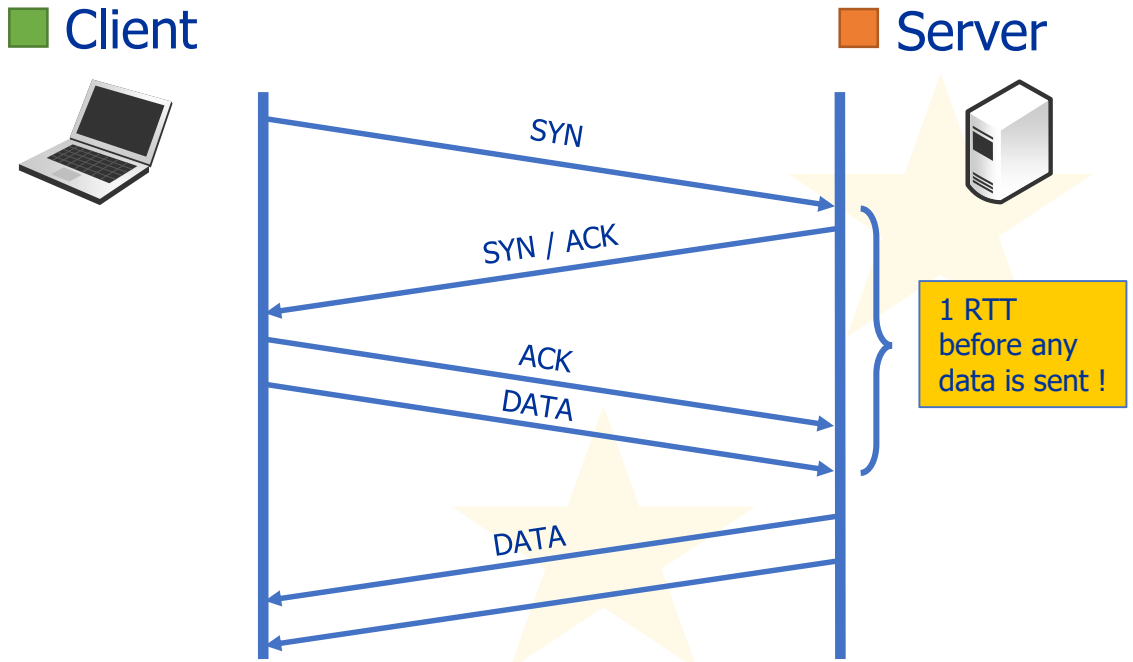
- Tools and commands
  - https://gallery.technet.microsoft.com/NTttcp-Version-528-Now-f8b12769
  - https://technet.microsoft.com/de-de/library/hh826152.aspx
  - https://technet.microsoft.com/de-de/library/hh826120.aspx
  - https://technet.microsoft.com/de-de/library/hh826132.aspx

Demo time

- Demo #1



Traditional 3-Way Handshake



Hello, would you like to hear a TCP joke?

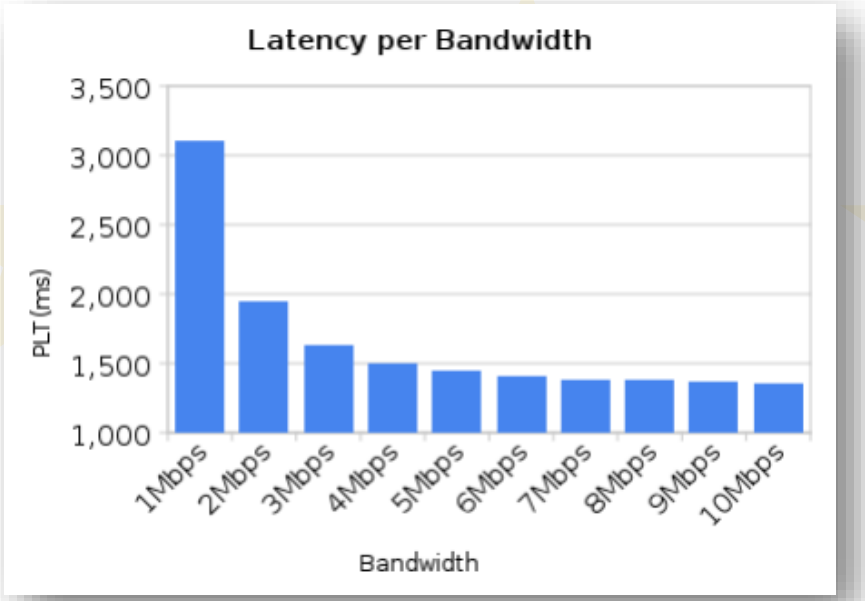
Yes, I'd like to hear a TCP joke.

OK, I'll tell you a TCP joke.

[…]

Latency matters! (1/2)

- Page Load Time (PLT) in ms
- No significant benefit beyond 5 Mbps bandwidth

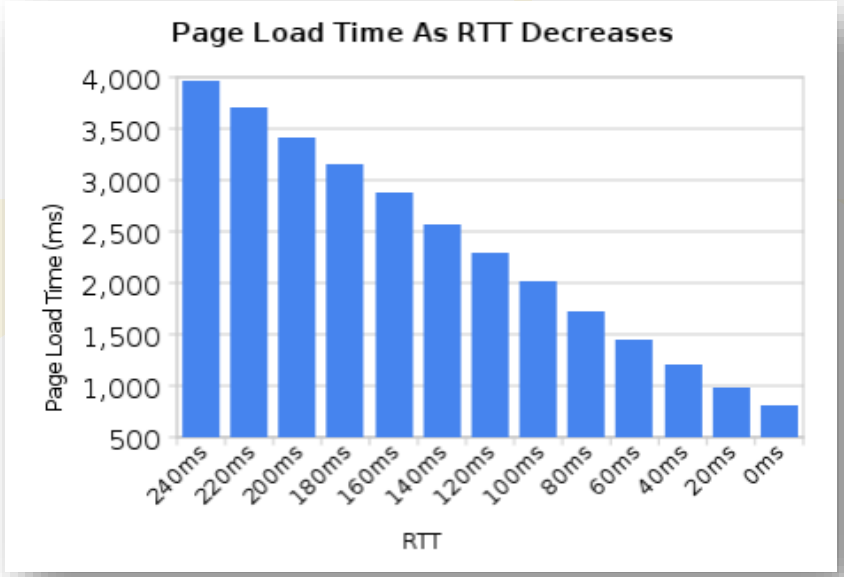


[*] „More Bandwidth Doesn't Matter" - Mike Belshe – 2010 – (https://goo.gl/EQunxX)

Latency matters! (2/2)

- Linear correlation between PLT and latency
- Usual cellular networks with varying latency > 100ms

What can be done to improve this?

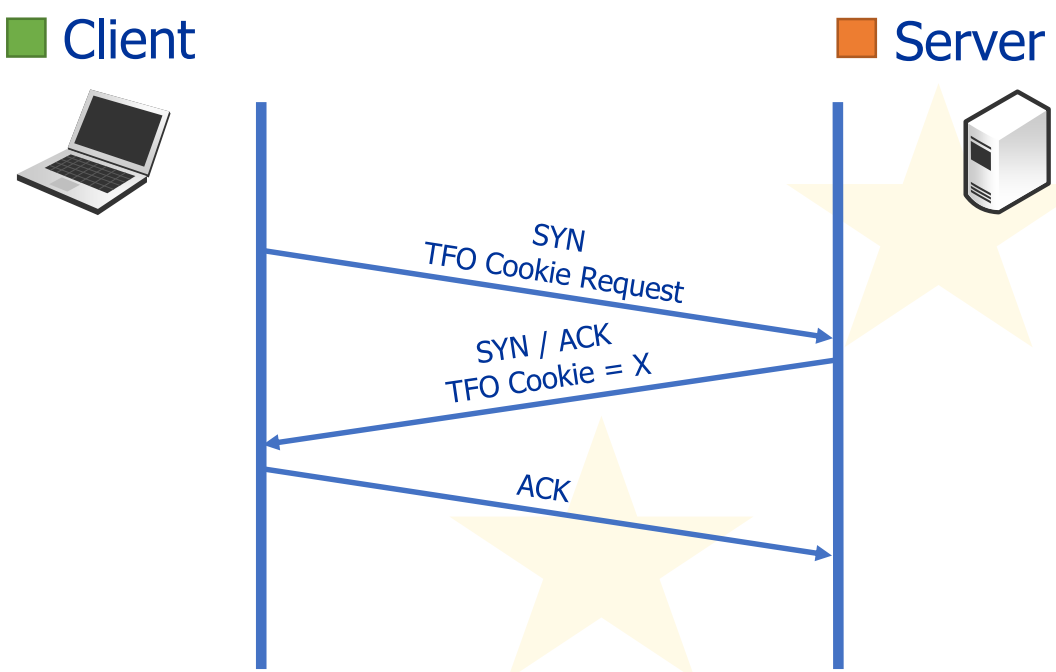


[*] „More Bandwidth Doesn't Matter" - Mike Belshe – 2010 – (https://goo.gl/EQunxX)

TCP Fast Open (TFO)

- Introduced in 2012 (RFC 7413)
  - https://tools.ietf.org/html/rfc7413
- Reduces the time before data can be sent
- Enables servers to reply immediately to requests

TFO 1st TCP connection



Cookie Exchange

How does the server generate a cookie?

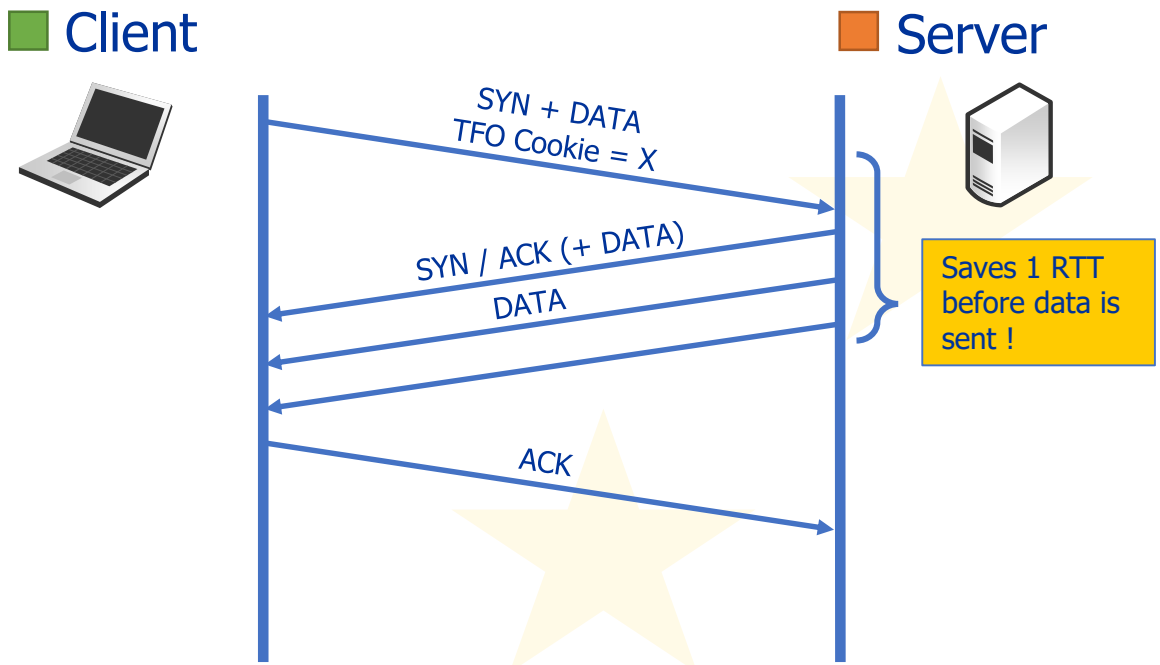
X = Encode(Client-IP+secret)

Example

Encrypt IPv4 address with AES128 + padding and truncate to 64 bits:

"7ba53e3e6e360838"

TFO 2nd TCP connection



- A second connection to the same destination server requires the cookie to be sent by the client within the SYN.
- Unlike in the original TCP RFC (793) data in the SYN is now allowed to be forwarded to the application.
- DoS protection provided by the TFO cookie

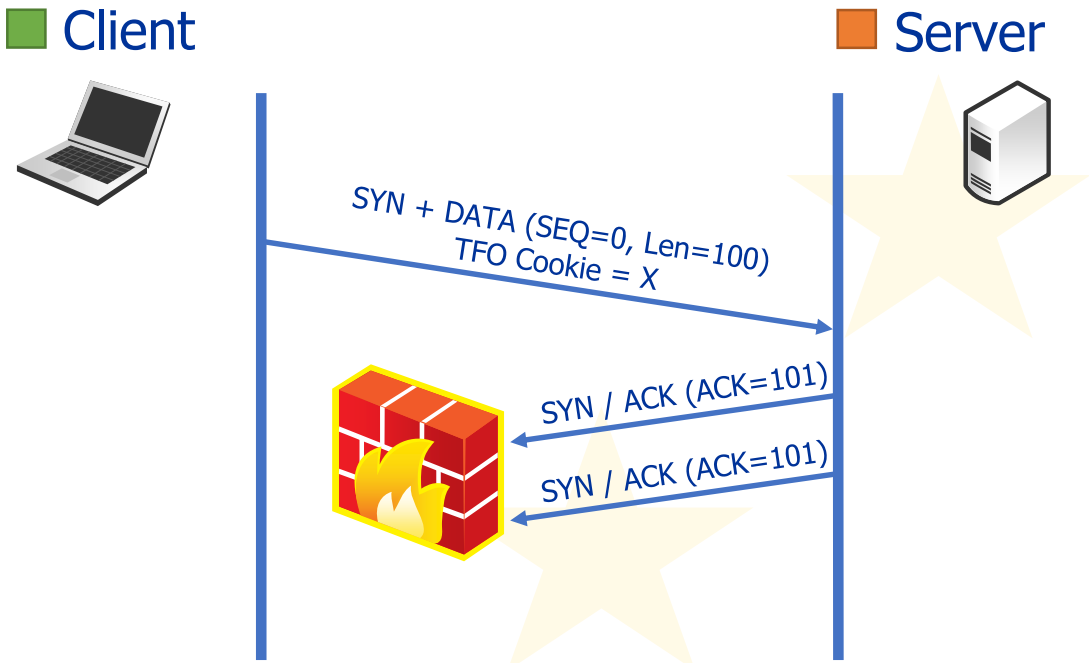
Demo time

- Demo #1 (revisited)

TFO drawbacks and challenges

- "Best" worst-case: Connections getting refused
  - Clients can react w/o having to wait for RTO (~1 sec)
- Data in SYN packets is getting stripped
  - TCP option not recognized by all devices out there
- Packets are dropped
  - Unexpected high ACK value in servers SYN/ACK
    - Details on the next slide…

TFO blocked SYN/ACKs



Middle boxes such as firewalls expect the SYN/ACK ACK-numbers to be 0+1

References

- „Network Support for TCP Fast Open“
  - Christoph Paasch - https://youtu.be/Qo9rFpiLMWI
- „More Bandwidth Doesn‘t Matter (much)“
  - Mike Belshe - https://goo.gl/EQunxX
- “TCP Fast Open”
  - RFC 7413 - https://tools.ietf.org/html/rfc7413

Criteria limiting data flow

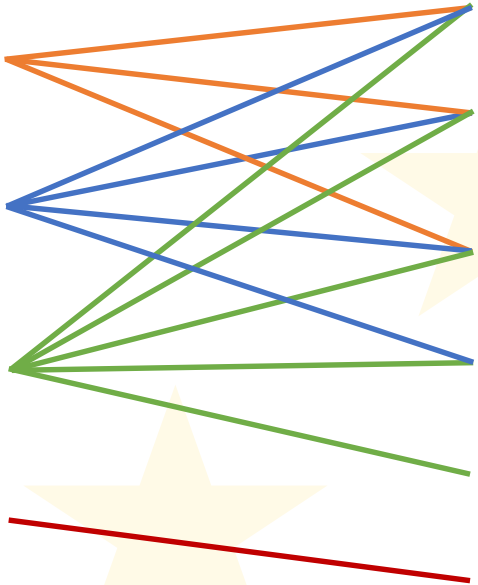
- Performance of
  - Sender
  - Receiver
  - Network (LAN, WAN)
- Quality of
  - Network
  - Congestion control mechanisms

TCP Congestion Control

No revolution, but evolution

Algorithms

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



Components

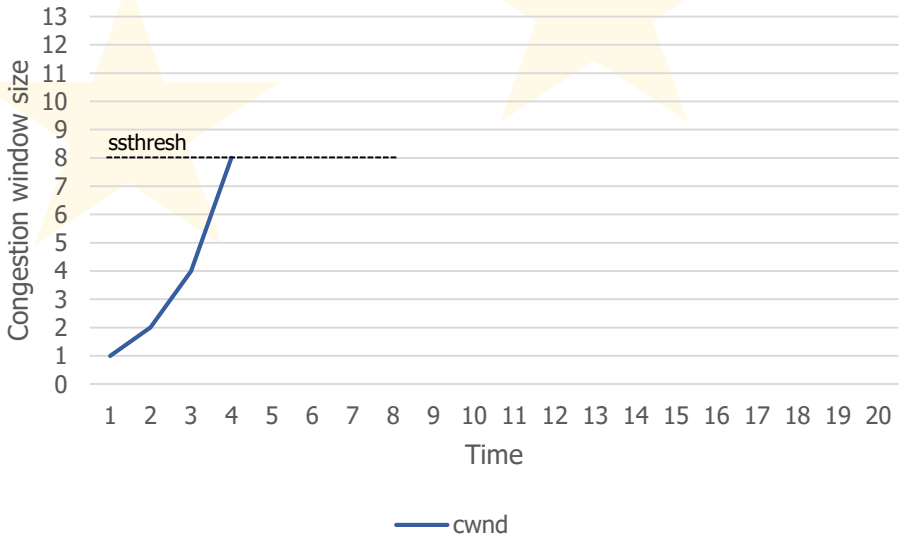
- Slow Start
- Additive Increase Multiplicative Decrease (AIMD)
- Fast Retransmit
- Fast Recovery
- Partial ACK
- Selective ACK

Slow Start

- Congestion window (cwnd)
  - cwnd = x*MSS (max. segment size)
  - Initial size varies:
    x = 1, 2, 4 or 10
  - Increasing exponentially with each ACK

→ Doubling the size on every round-trip (RTT)*

Until: cwnd >= ssthresh
(Slow Start Threshold)

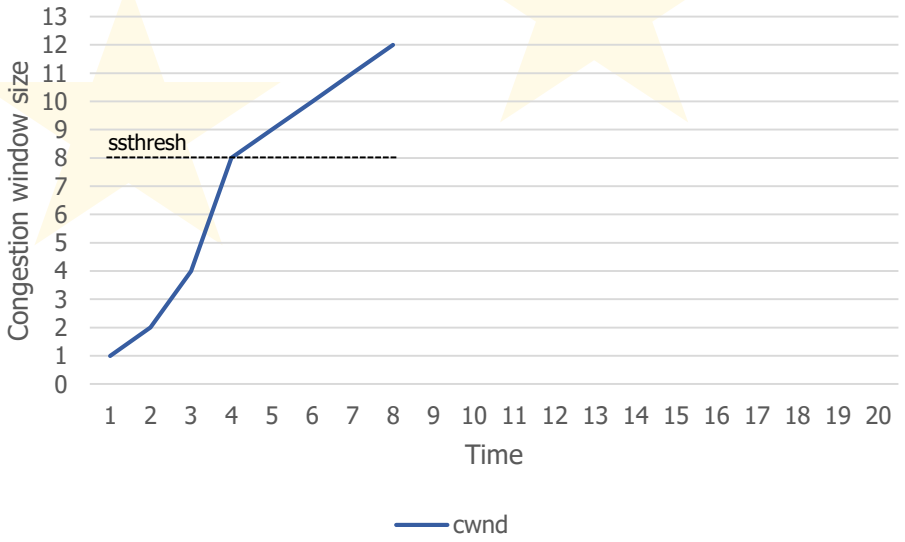


[*] provided there is an ACK every round-trip

Congestion Avoidance

- Starting when:
  - cwnd >= ssthresh
- On each ACK:
  - cwnd = cwnd + 1 MSS

→ Linear growth of cwnd



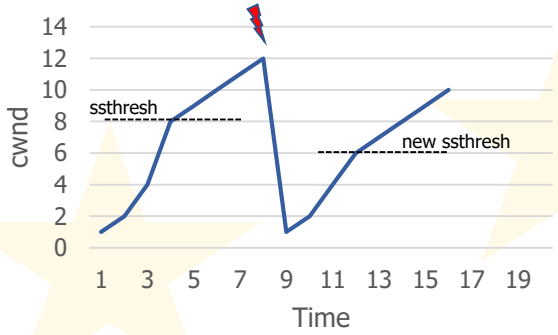
Key take away message

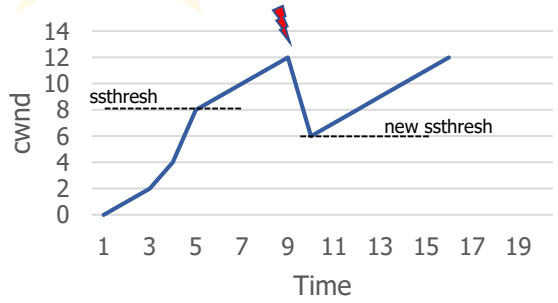
TCP Reno: Packet Loss

- Loss detection on timeout RTO
  - sshtresh = last cwnd / 2
  - cwnd = 1
  - Slow start (again)
  - (equivalent to TCP Tahoe)

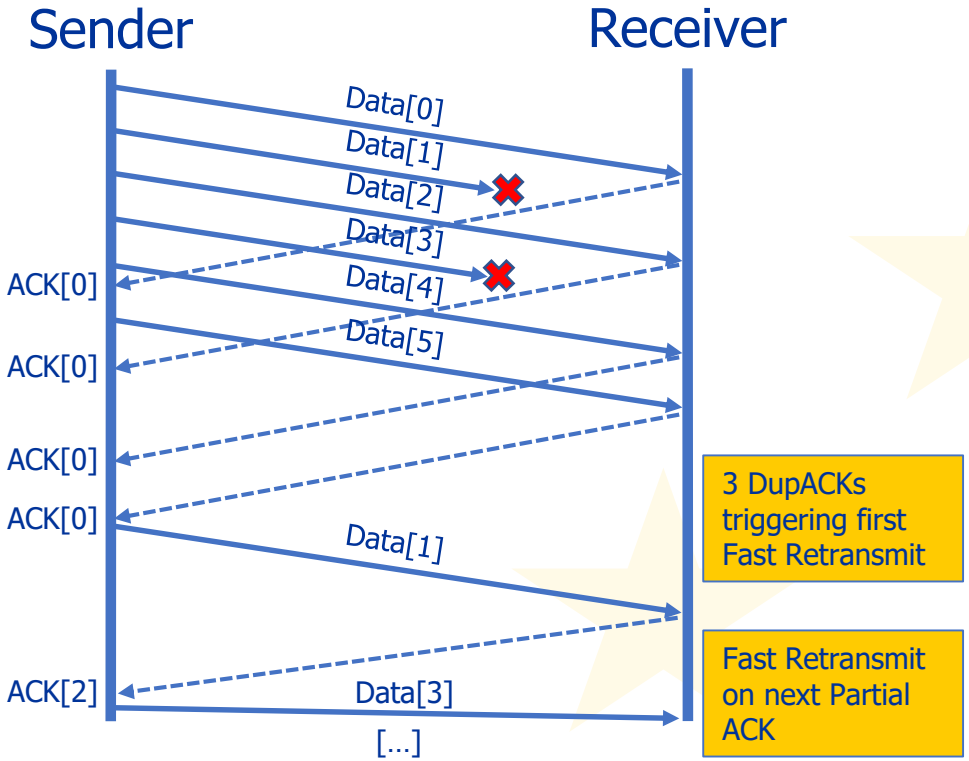
- Loss detection with Dup ACKs
  - sshtresh = last cwnd / 2
  - cwnd = ssthresh
  - Congestion Avoidance, no Slow Start
  - "Fast" Recovery

- OK, cool! But what about multiple packet loss?





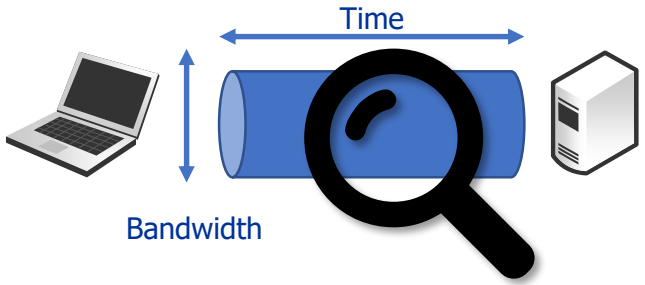
TCP New Reno: Partial ACK



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

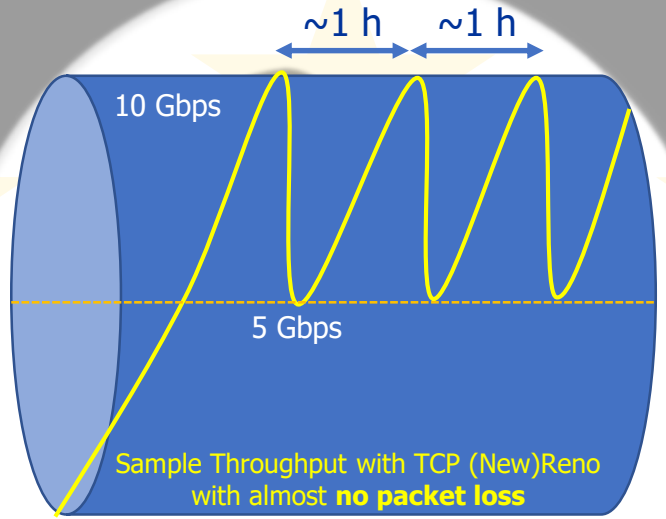
Advisements

- Does more and more bandwidth solve the congestion problem?
- Does (New) Reno address large BDP networks?



Congestion Control in LFN

- Large Fat Networks (LFN)
  - Long RTT, Large Pipe
- To fill a 10 Gbps, 100ms pipe
  - ~120 MBytes Window size
  - Both (cwnd and rwin)
- Major drawbacks
  - Time to ramp up the cwnd with TCP (New)Reno
  - Time to react on congestion events



Glue stuff together: CTCP

- Compound TCP introduced by Microsoft with Windows Vista and Server 2008
  - Patches for Windows XP and Server 2003 available
- Sliding window is a sum of two variants
  - Loss based window with AIMD
  - Delay based window estimating queueing delays

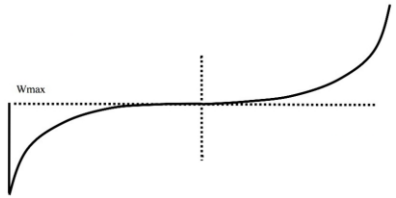
Principle of CTCP

- Efficiency
  - Improve network throughput on high speed network links
- RTT Fairness
  - Good intra-protocol fairness with different RTTs on competing flows
- TCP Fairness
  - Make better use of free bandwidth, but do not steal bandwidth from other flows

Along with Windows 10

- New: CUBIC TCP
  - Improved Congestion Control for high bandwidth/latency networks >100ms
  - Aggressive window growth independent of RTT
  - More fairness between short and long RTT connections*

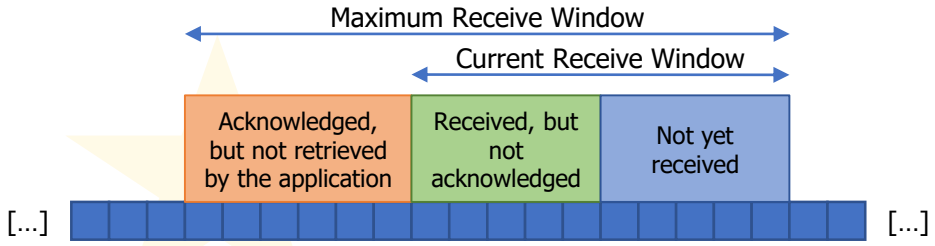
- New: Increased Initial Congestion Window (IW)
  - IW10 instead of IW4 with TCP Reno
  - Improved Slow Start speed



[*] Good fairness between CUBIC flows – not the case for CUBIC vs. Reno

Receive Window

- Former TCP stacks (e.g. Windows XP) used static receive window sizes
  - Defaults are based on link speeds, configured values or SO_RCVBUF Windows socket



[*] „The Cable Guy TCP Receive Window Auto-Tuning" – Joseph Davies– (https://goo.gl/5FYSRq)

RWIN on steroids

- TCP Window Scaling increases the receive window size
  - Allows up to 1 GB Window sizes instead of 64 Kbytes
  - Scaling value is static after initial 3-Way-Handshake
  - Helps a lot in LFNs but does not adjust to varying RTT

```
Checksum: 0x4a54 [unverified]
[Checksum Status: Unverified]
Urgent pointer: 0
Options: (12 bytes), Maximum segment size, No-Operation (NOP), Window scale, No-Operation (NOP), No-Opera
  TCP Option - Maximum segment size: 1460 bytes
  TCP Option - No-Operation (NOP)
  TCP Option - Window scale: 8 (multiply by 256)
      Kind: Window Scale (3)
      Length: 3
      Shift count: 8
      [Multiplier: 256]
  TCP Option - No-Operation (NOP)
  TCP Option - No-Operation (NOP)
  TCP Option - SACK permitted
```

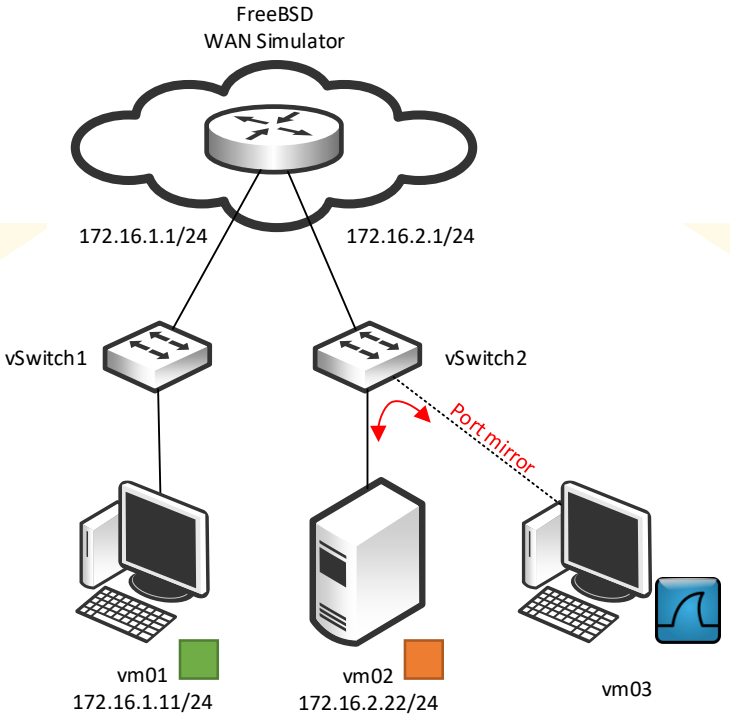
Shift count	...	8	7	6	5	4	3	2	1	0
Multiplier	...	256	128	64	32	16	8	4	2	1

Improved RWIN Auto-Tuning

- Enables Window Scaling by default
- Measures BDP and application retrieve rate
- Determines optimal RWIN per connection

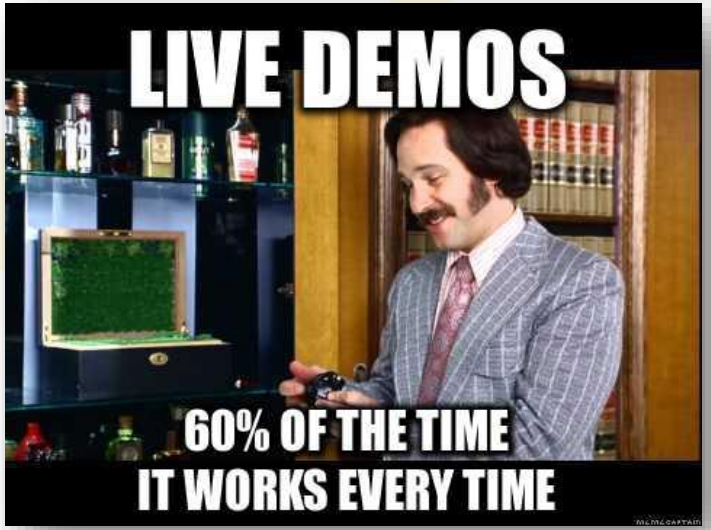
Lab Topology Windows Server

- Hyper-V based lab setup
- Linux Router
  - eth1 172.16.1.1/24
  - eth2 172.16.2.1/24
- vm01 Windows Server 2016
  - eth0 172.16.1.11/24
- vm02 Windows Server 2016
  - eth0 172.16.2.22/24



Demo time

- Demo #2



Thank you

Questions? Compliments? Wisdoms?

Please use the Guidebook App to provide feedback.

Contact

@SimonLindermann

sl@local-area.network