Hello!

Connection Established!

TCP - Architecture, Enhancements & Tuning

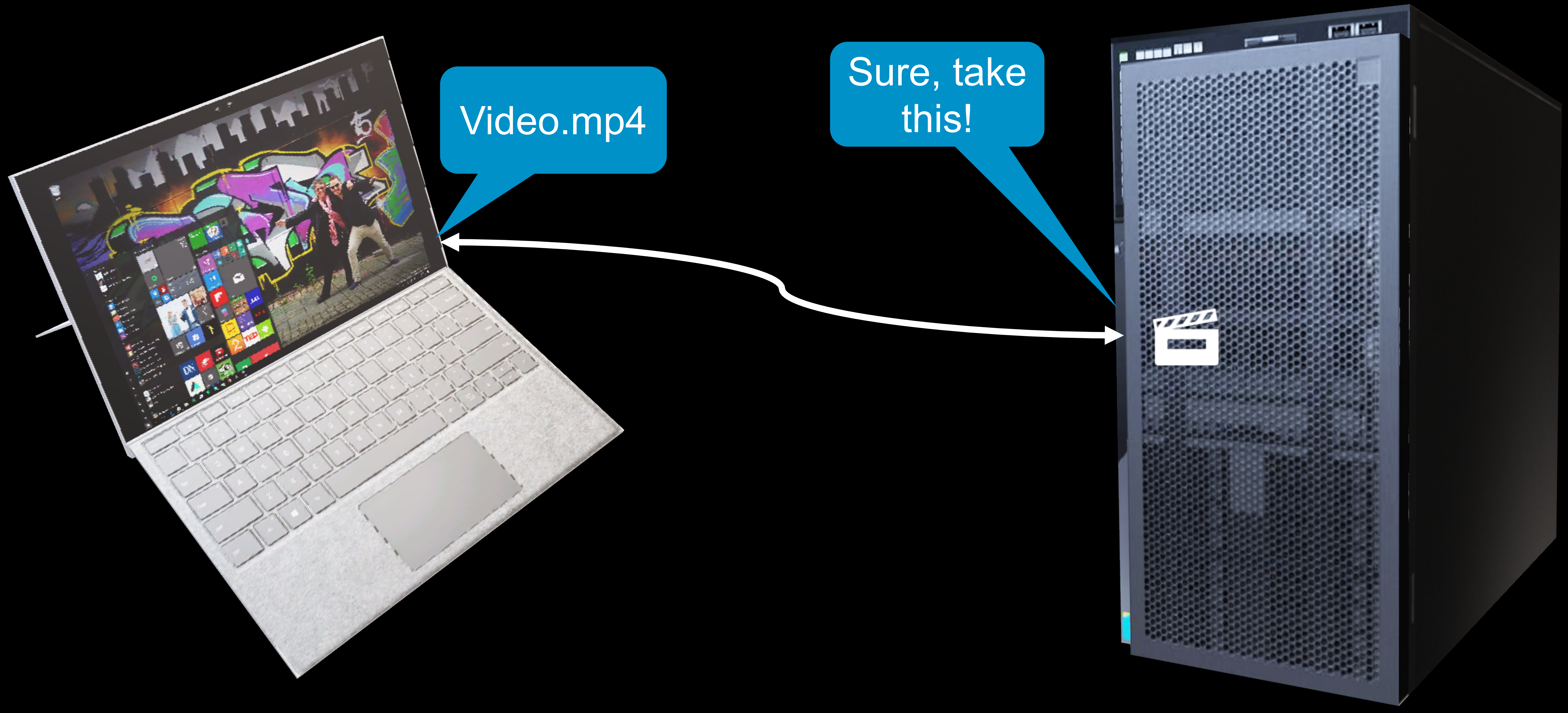
Dinesh Dhakal

Site Reliability Engineer

Today’s agenda

11:00	Introductions
11:05	Core Functionality
11:15	Enhancements and Extensions
11:30	Tuning of TCP Parameters on Linux
11:40	The March Ahead
11:45	Q&A

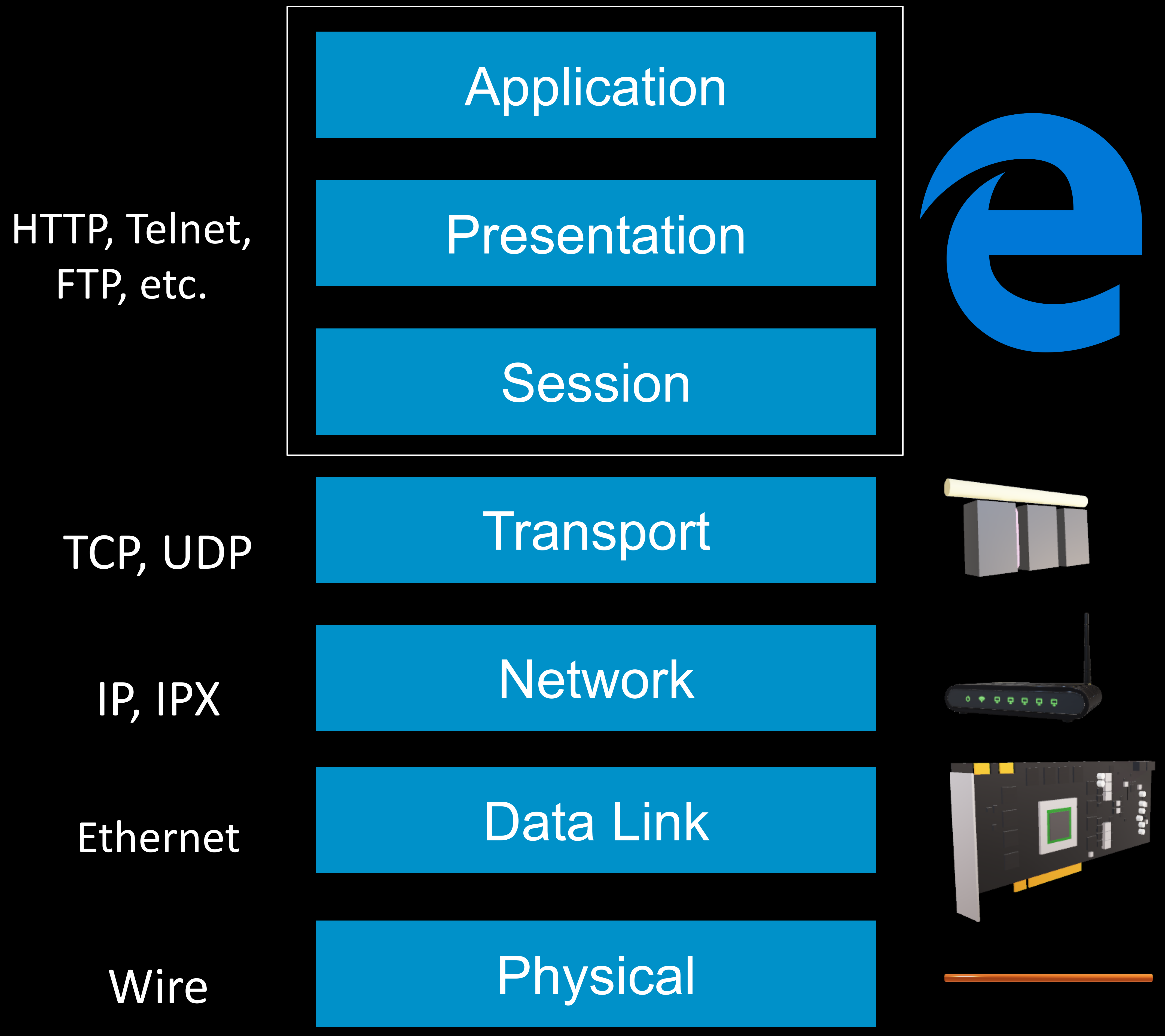
Let’s Talk



Video.mp4

Sure, take this!

The Network Stack



HTTP, Telnet, FTP, etc.

Application

Presentation

Session

TCP, UDP

Transport

IP, IPX

Network

Ethernet

Data Link

Wire

Physical

What should the Transport layer do?

Problems

- Applications send byte streams
- Underlying IP network is stateless
- Devices are of varied capabilities
- Multiple processes need reliable communication
- Cannot control all the variables

Requirements

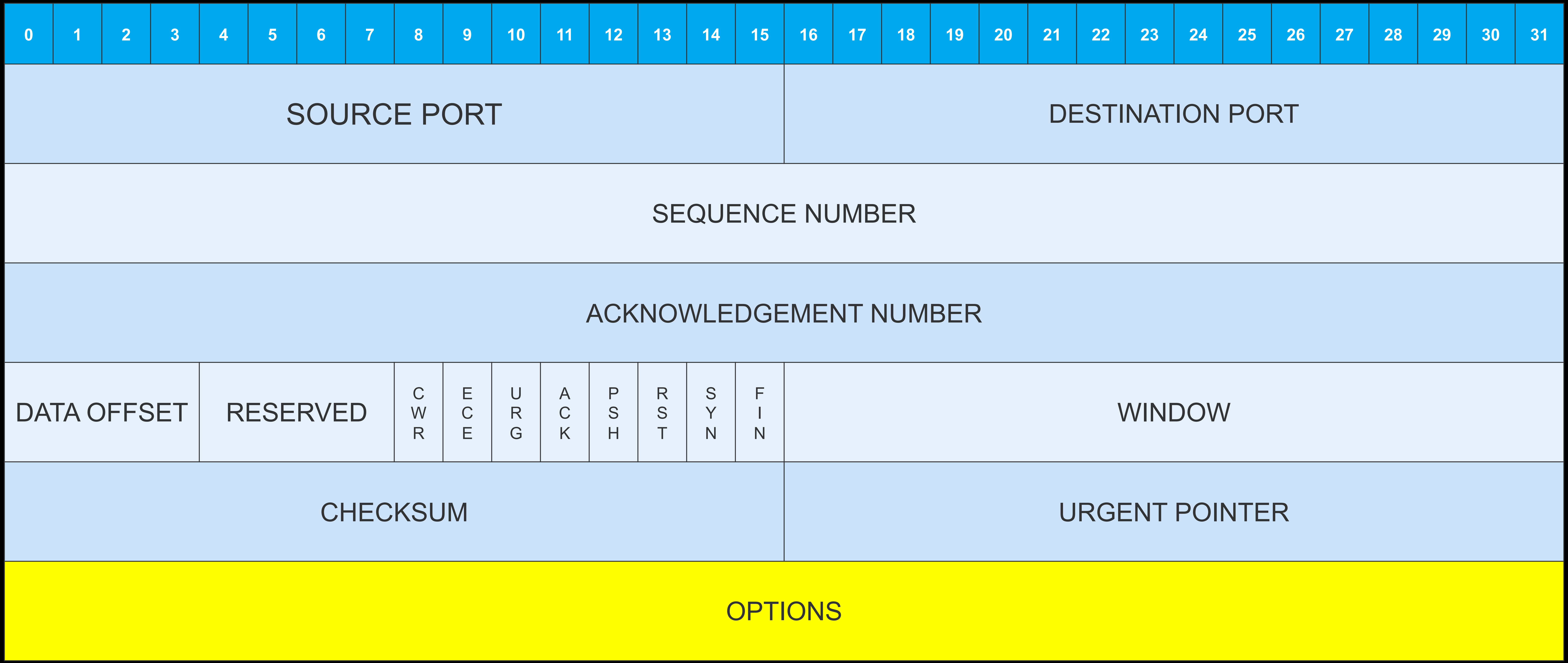
- Ordered Segmentation
- Stateful Communication
- Flow Control
- Multiplexing
- Reliability and Congestion Control

TCP – Architecture

TCP Core concepts

Requirements	How TCP addresses it
Ordered Segmentation	Sequence Numbers
Stateful Communication	Connections
Flow Control	TCP Window Size
Multiplexing	Port Numbers
Reliability and Congestion Control	Acknowledgements and Retransmissions

TCP Header



<table>
<tr><td>0</td><td>1</td><td>2</td><td>3</td><td>4</td><td>5</td><td>6</td><td>7</td><td>8</td><td>9</td><td>10</td><td>11</td><td>12</td><td>13</td><td>14</td><td>15</td><td>16</td><td>17</td><td>18</td><td>19</td><td>20</td><td>21</td><td>22</td><td>23</td><td>24</td><td>25</td><td>26</td><td>27</td><td>28</td><td>29</td><td>30</td><td>31</td></tr>
<tr><td colspan="16">SOURCE PORT</td><td colspan="16">DESTINATION PORT</td></tr>
<tr><td colspan="32">SEQUENCE NUMBER</td></tr>
<tr><td colspan="32">ACKNOWLEDGEMENT NUMBER</td></tr>
<tr><td colspan="4">DATA OFFSET</td><td colspan="4">RESERVED</td><td>CWR</td><td>ECE</td><td>URG</td><td>ACK</td><td>PSH</td><td>RST</td><td>SYN</td><td>FIN</td><td colspan="16">WINDOW</td></tr>
<tr><td colspan="16">CHECKSUM</td><td colspan="16">URGENT POINTER</td></tr>
<tr><td colspan="32">OPTIONS</td></tr>
</table>

TCP Segments

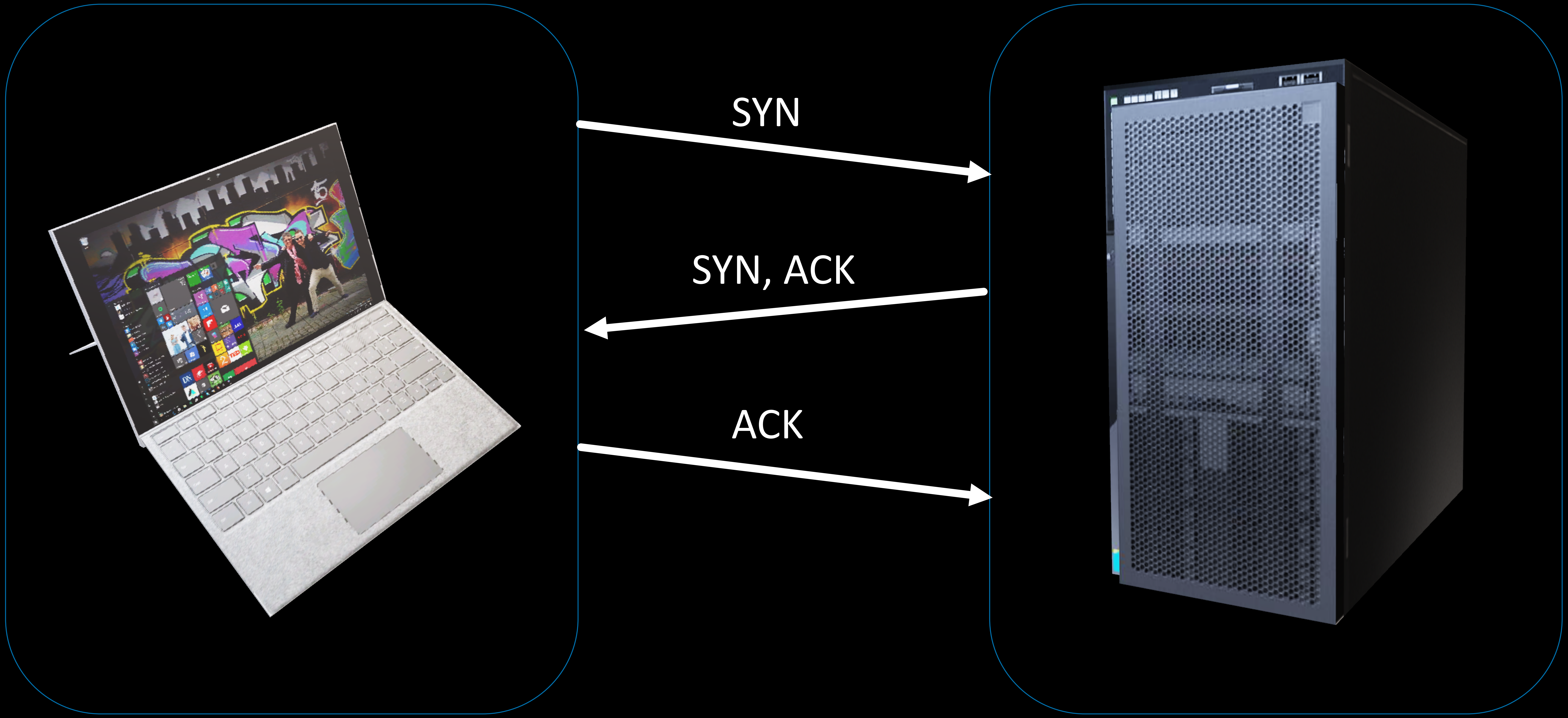
1011101010111101000010101010100010100100010001000100101010010101010101…

8080	22003
SEQ : 1101	
ACK : 2201	
Data : 101110101…	

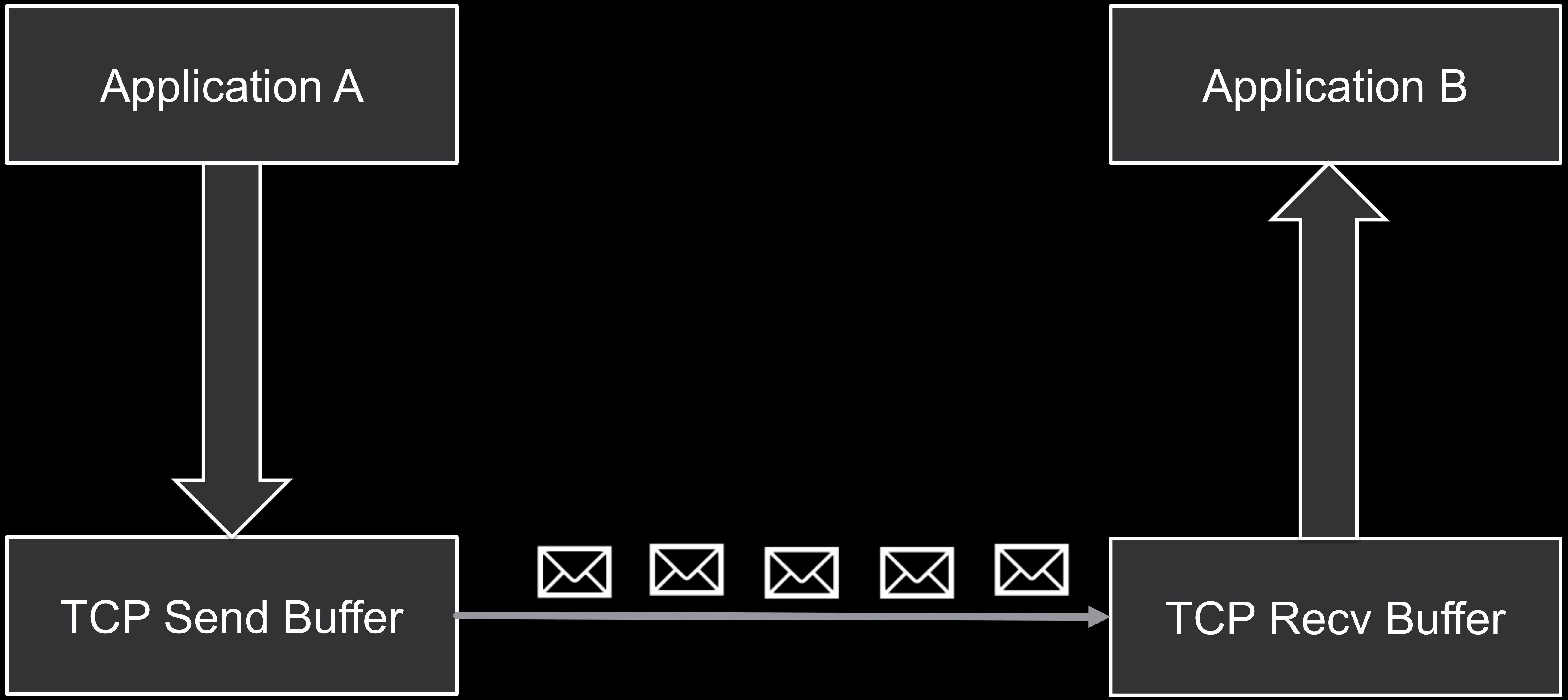
8080	22003
SEQ : 1102	
ACK : 2202	
Data : 101110101…	

8080	22003
SEQ : 1103	
ACK : 2203	
Data : 101110101...	

Connection Establishment – 3 way handshake

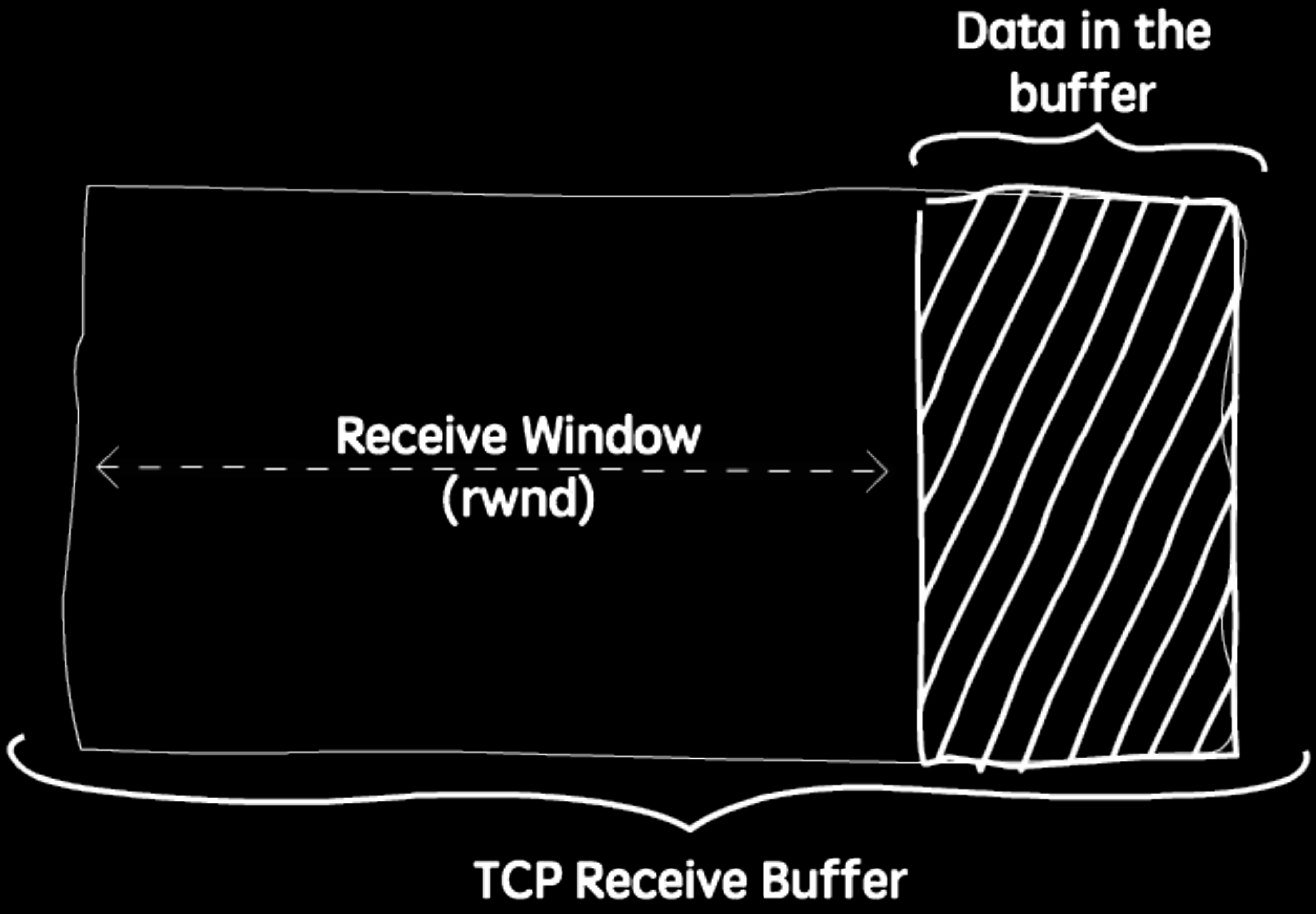


TCP Sockets



Application A
Application B
TCP Send Buffer
TCP Recv Buffer

Flow Control – Sliding Window



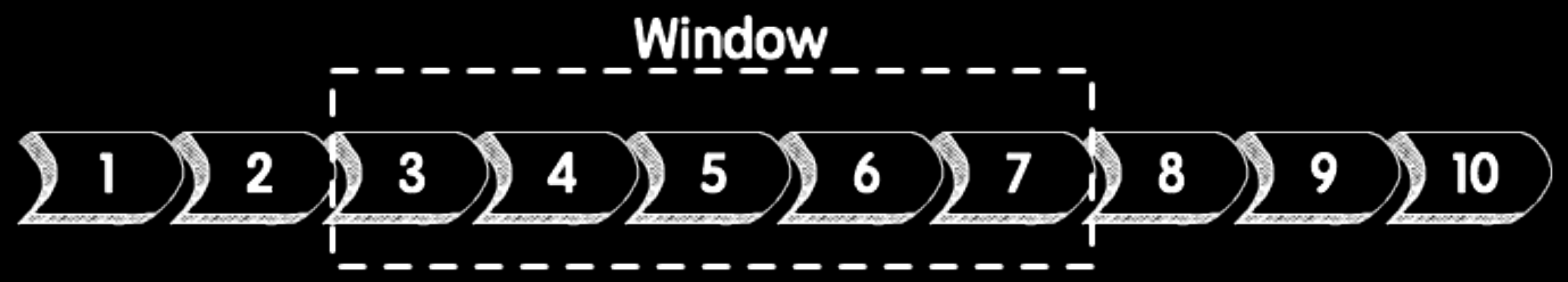
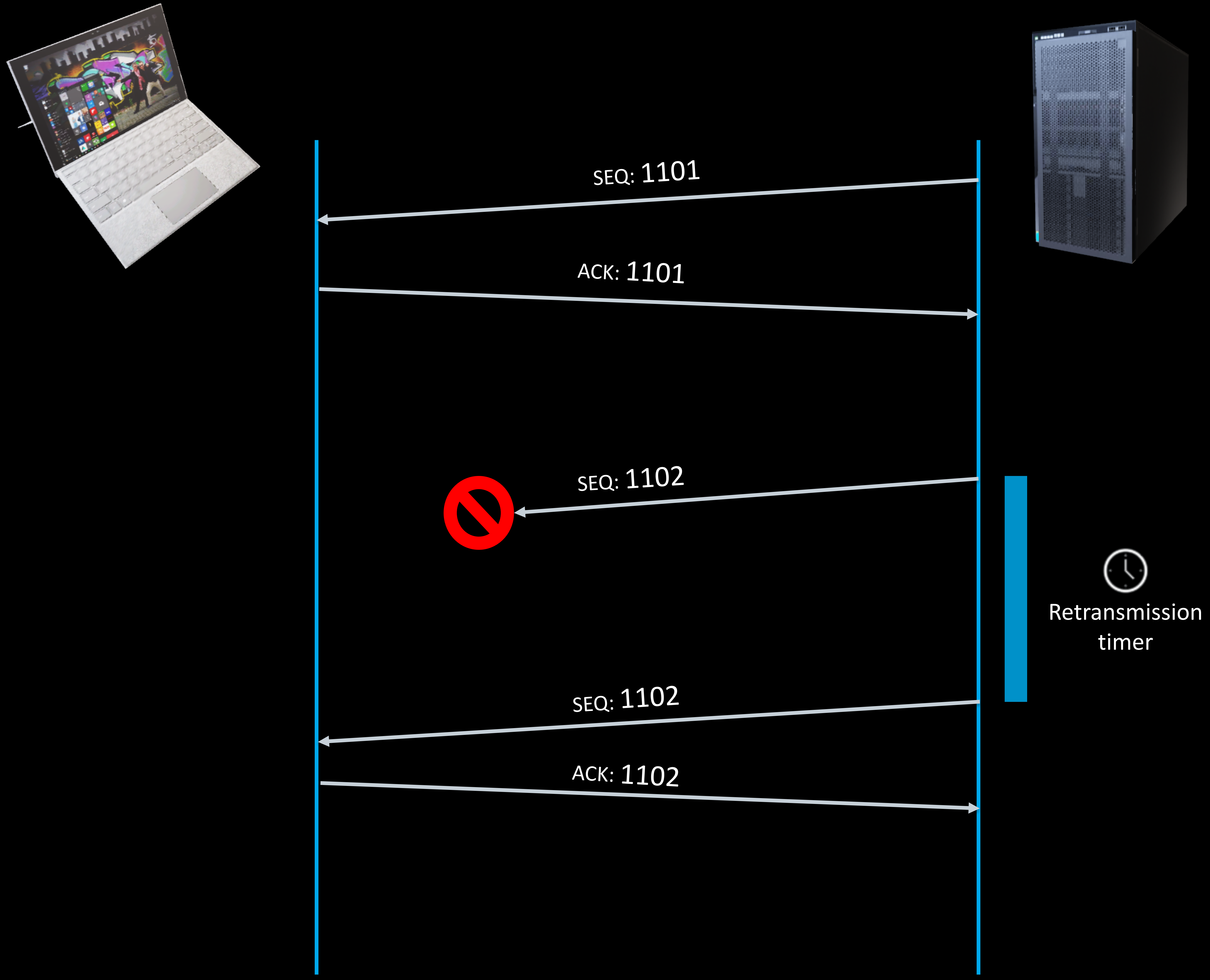


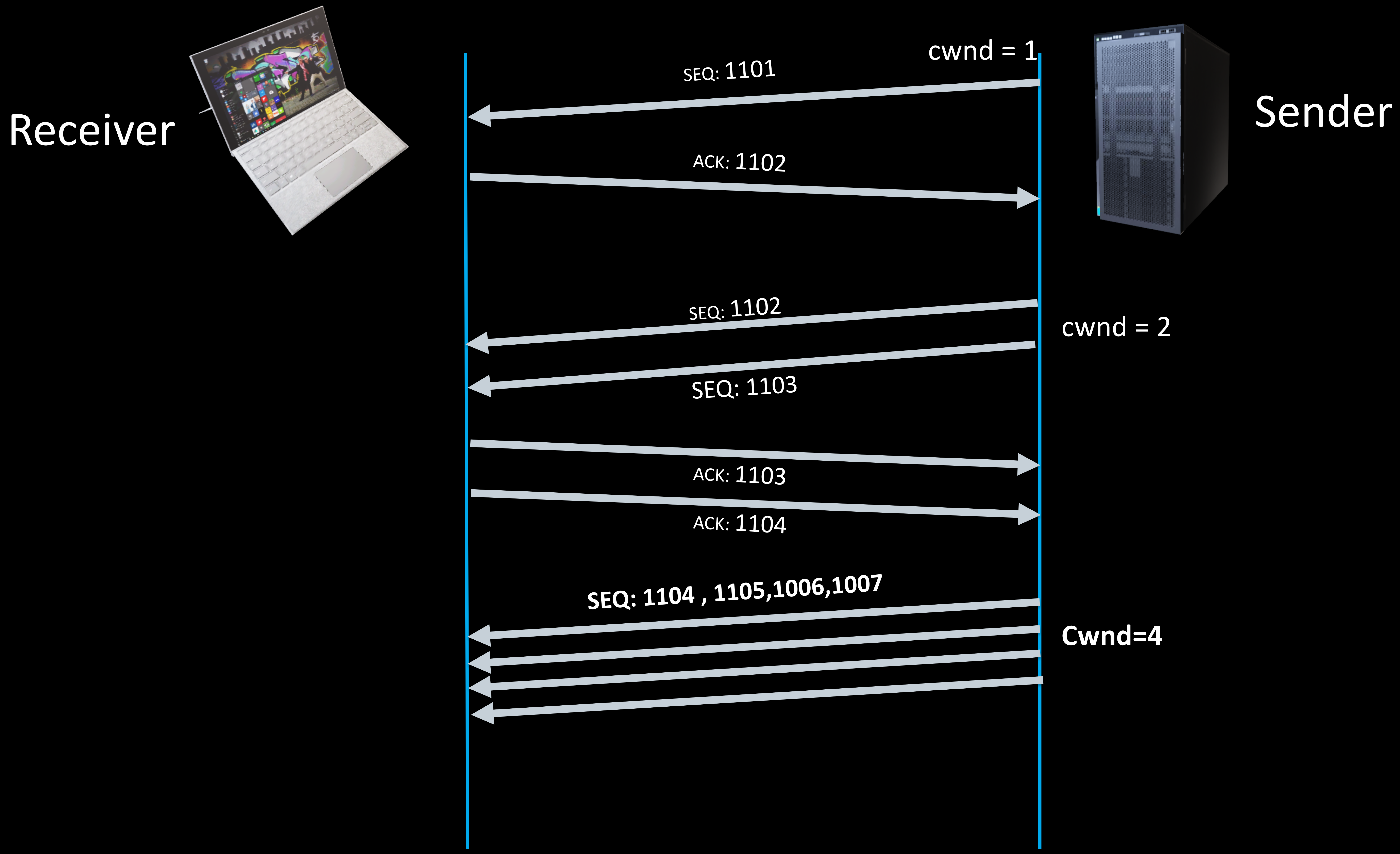
Image source : https://www.brianstorti.com/tcp-flow-control/

Retransmission



Enhancements

Slow Start Phase



Receiver

Sender

cwnd = 1

SEQ: 1101

ACK: 1102

SEQ: 1102

cwnd = 2

SEQ: 1103

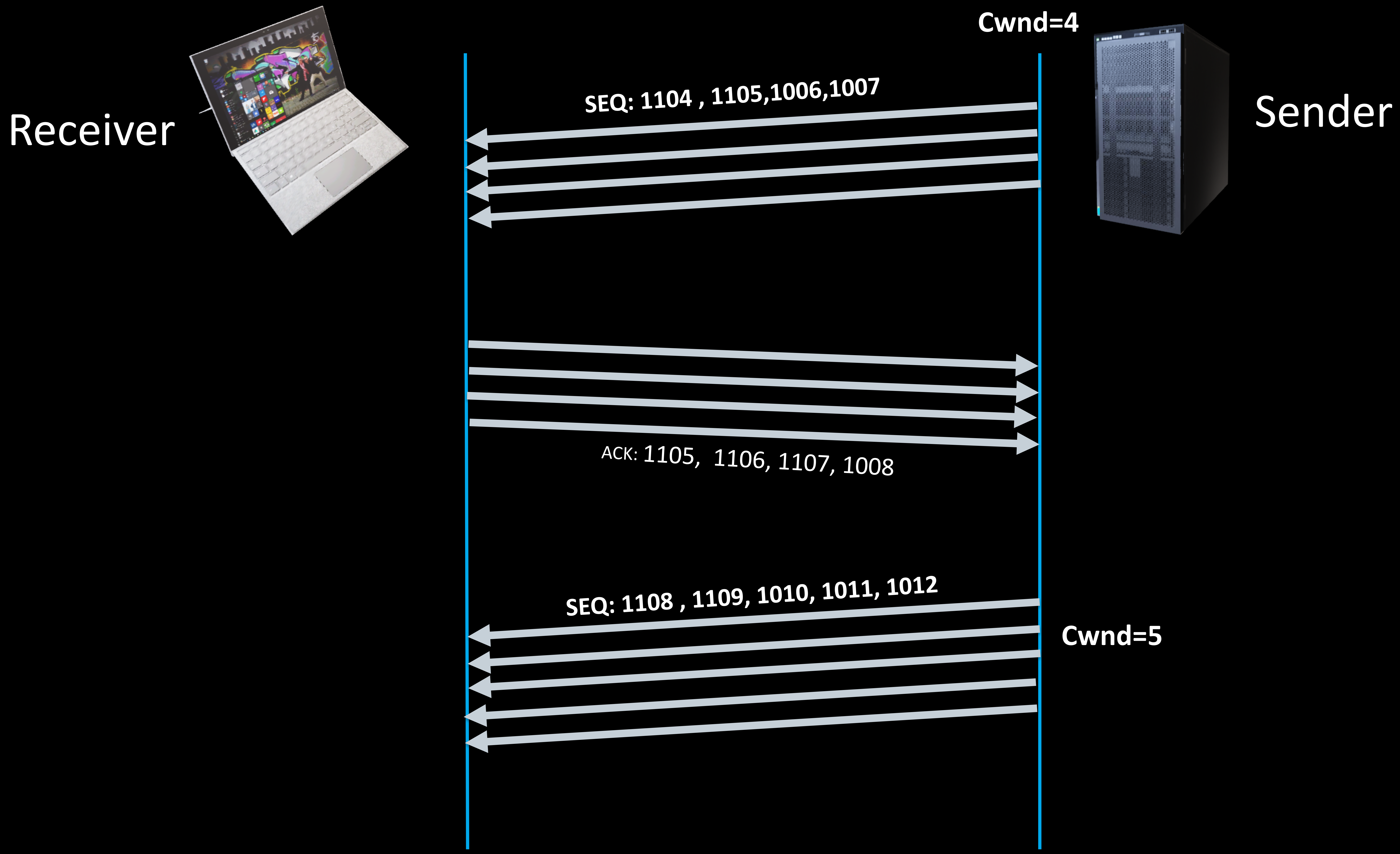
ACK: 1103

ACK: 1104

SEQ: 1104 , 1105,1006,1007

Cwnd=4

Congestion Avoidance



Receiver

Sender

Cwnd=4

SEQ: 1104 , 1105,1006,1007

ACK: 1105, 1106, 1107, 1008

SEQ: 1108 , 1109, 1010, 1011, 1012

Cwnd=5

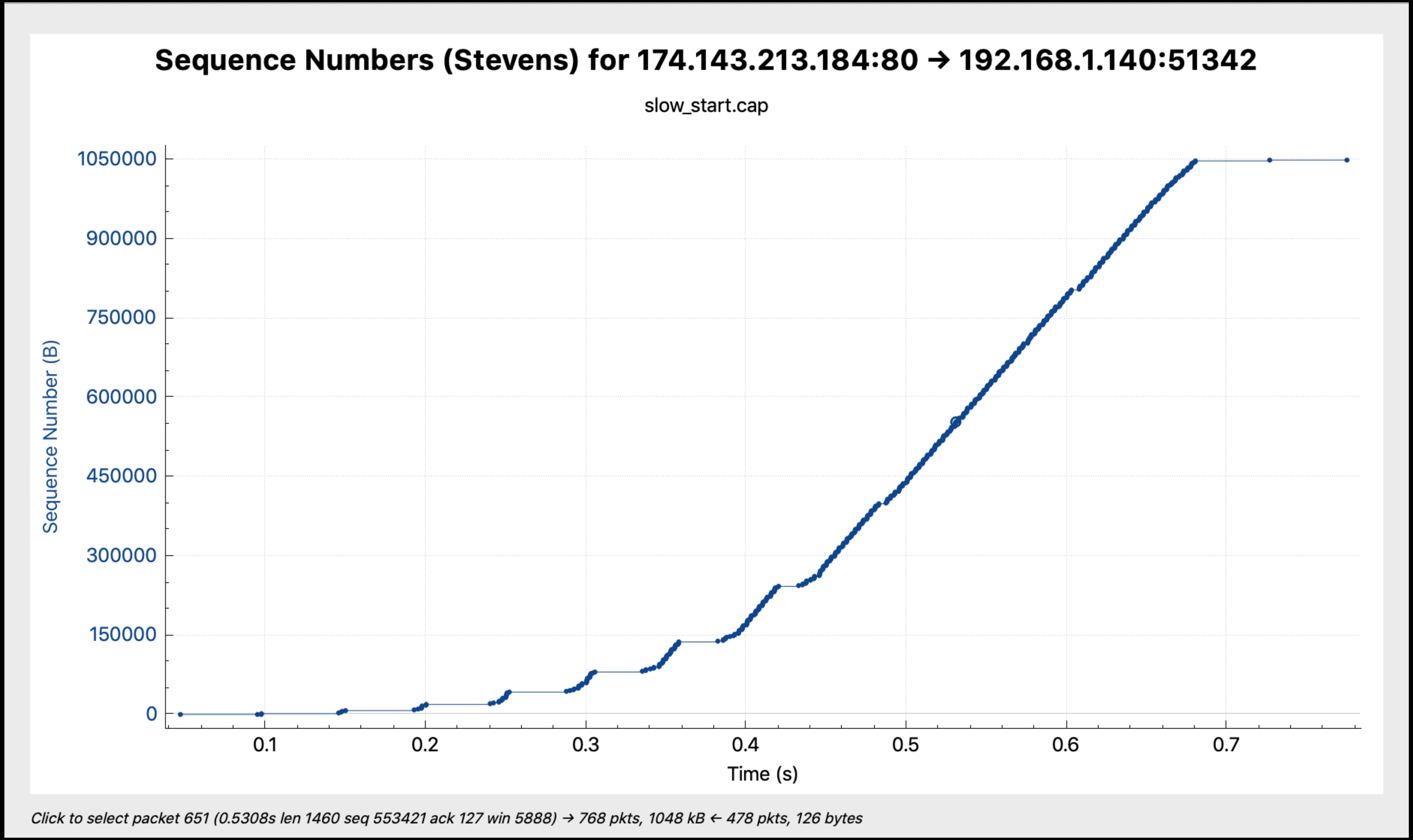
Congestion Control Enhancements

- Slow Start
  - Slow start when Congestion window (cwnd) < slow start threshold (ssthresh)
    - Typically, ssthresh starts at 65535 bytes.
    - cwnd += min (N, SMSS)

SMSS – Sender Max Segment Size

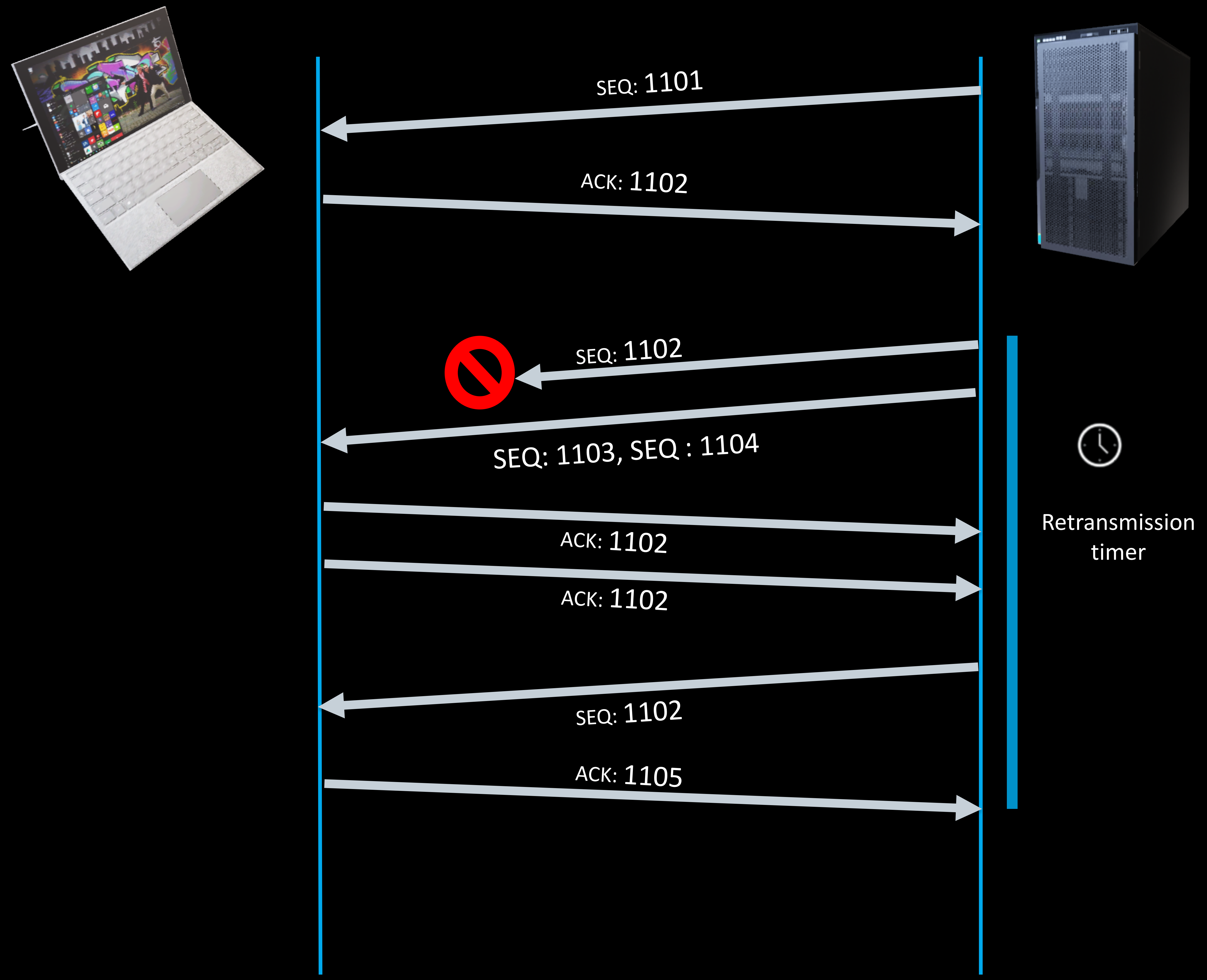
- Congestion Avoidance
  - Congestion avoidance when cwnd > ssthresh
    - On ACK: cwnd += SMSS*SMSS/cwnd
  - ssthresh = min(cwnd,rwnd) / 2 when congestion

TCP Slow Start



Click to select packet 651 (0.5308s len 1460 seq 553421 ack 127 win 5888) → 768 pkts, 1048 kB ← 478 pkts, 126 bytes

Fast Retransmission



Fast Recovery

- Receiver sends duplicate ack → Segments have left the network
- Artificially inflates the cwnd as segments sent are *assumed* to have left network
  - cwnd = ssthresh + 3 * SMSS
  - Every Additional ack : cwnd = cwnd + SMSS
- When a new segment is acknowledged –
  - Cwnd = ssthresh

Loss Recovery Enhancements

- TCP Selective Acknowledgment Options
  - Informs the sender about OOR segments received
  - Uses the TCP options fields to acknowledge the received segments
- Partial acks
  - Aims to reduce the number of duplicate acks needed for retransmit
  - Specifically useful for cases of continuous packet loss
  - Every partial ack in the gap triggers retransmit of next unacked segment

TCP Tuning

Bandwidth Delay Product

- The amount of data that can be in transit in the network
- Product of Bandwidth and Delay (RTT)
  - 1 Mbps X 70ms = 0.88 MByte
- Buffer sizes can be appropriately tuned to gain max utilization of bandwidth

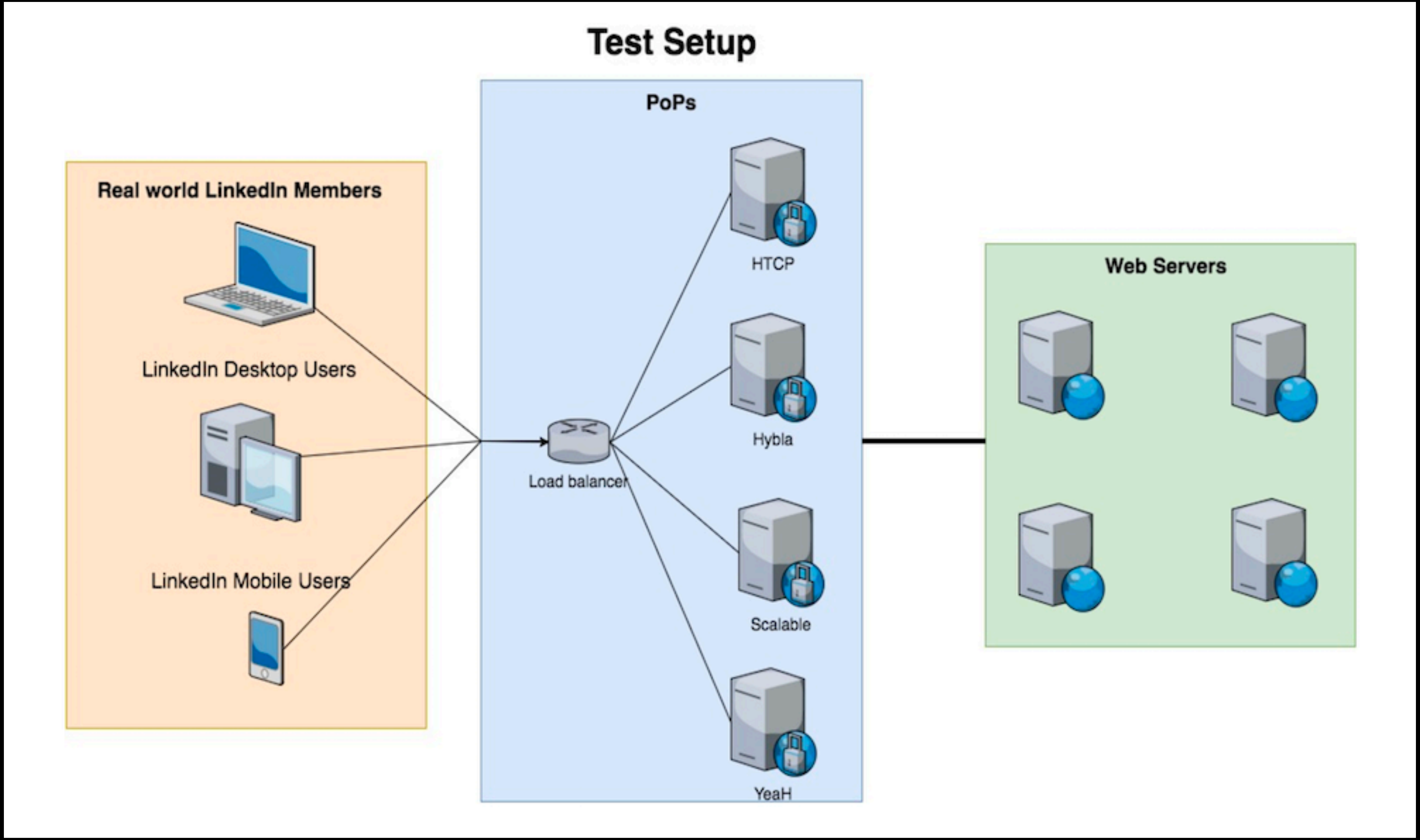
Buffers

- TCP Buffer sizes can be tuned for optimal use of Bandwidth
- net.core.rmem_max = 268435456
- net.core.wmem_max = 268435456
- net.ipv4.tcp_rmem = 4096 87380 134217728
  net.ipv4.tcp_wmem = 4096 65536 134217728

Some more Parameters…

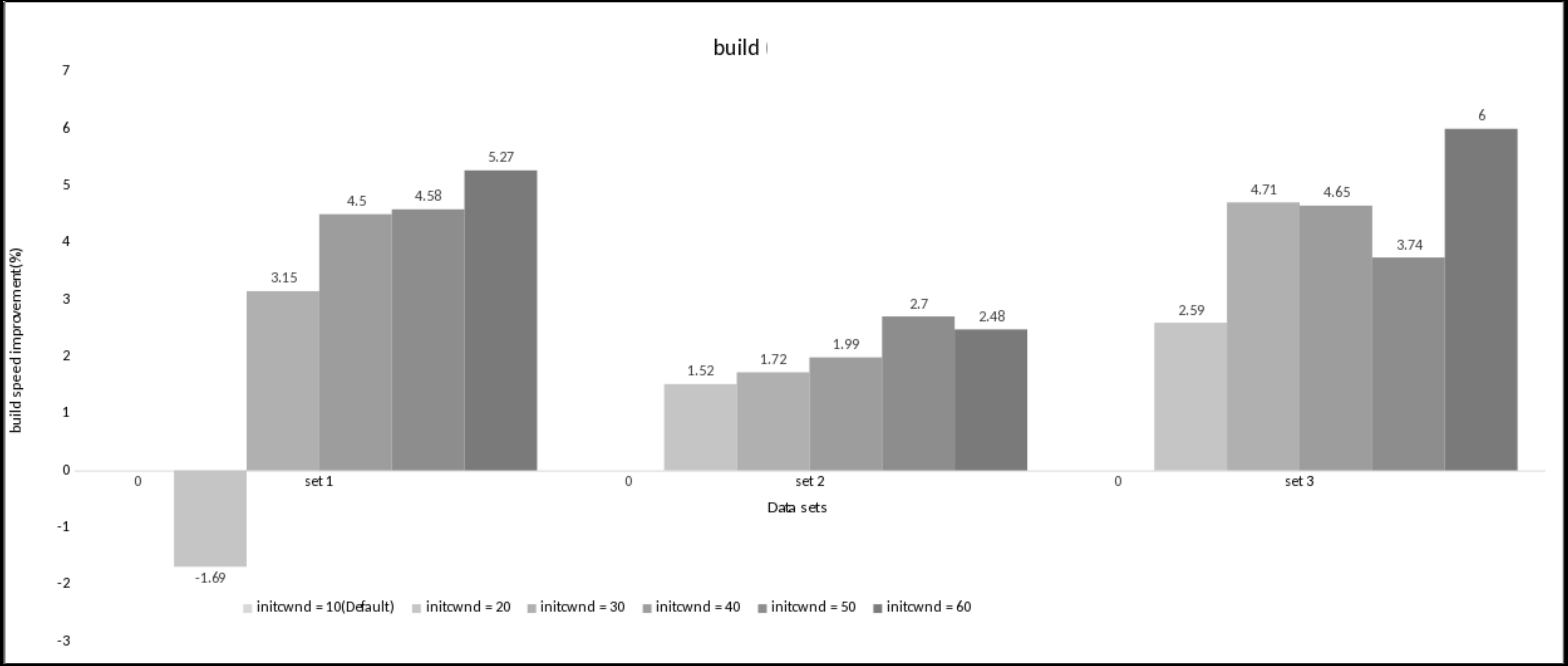
- Enable Selective Ack
  net.ipv4.tcp_sack = 1
- Enable Window Scaling -
  net.ipv4.tcp_window_scaling = 1
- MTU probing
  net.ipv4.tcp_mtu_probing = 0

Test Setup


Test Setup
PoPs
Real world LinkedIn Members
LinkedIn Desktop Users
LinkedIn Mobile Users
Load balancer
HTCP
Hybla
Scalable
YeaH
Web Servers

Initial Congestion Window

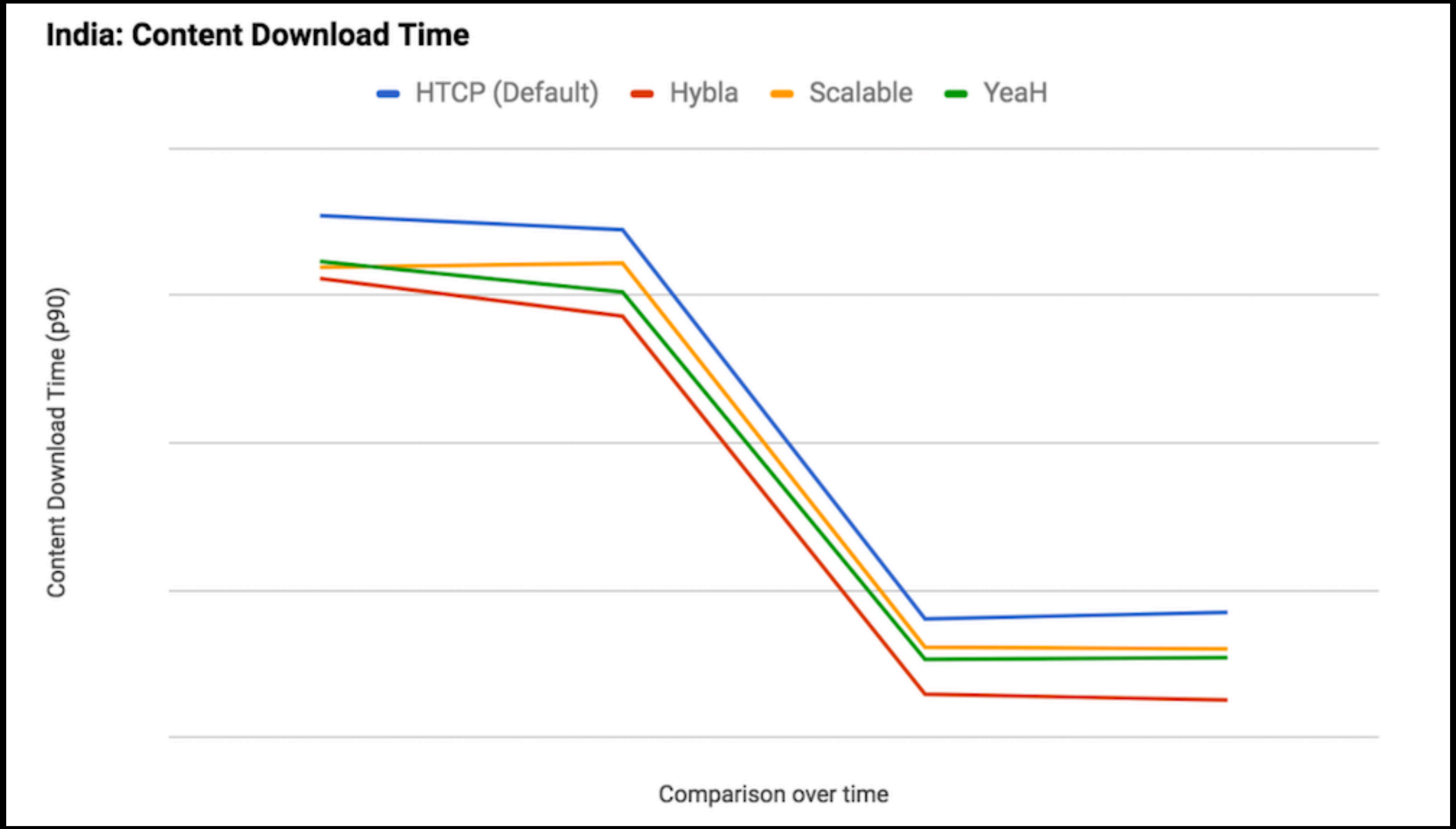
- Increasing initcwnd can reduce the number of Round Trips thus increasing performance

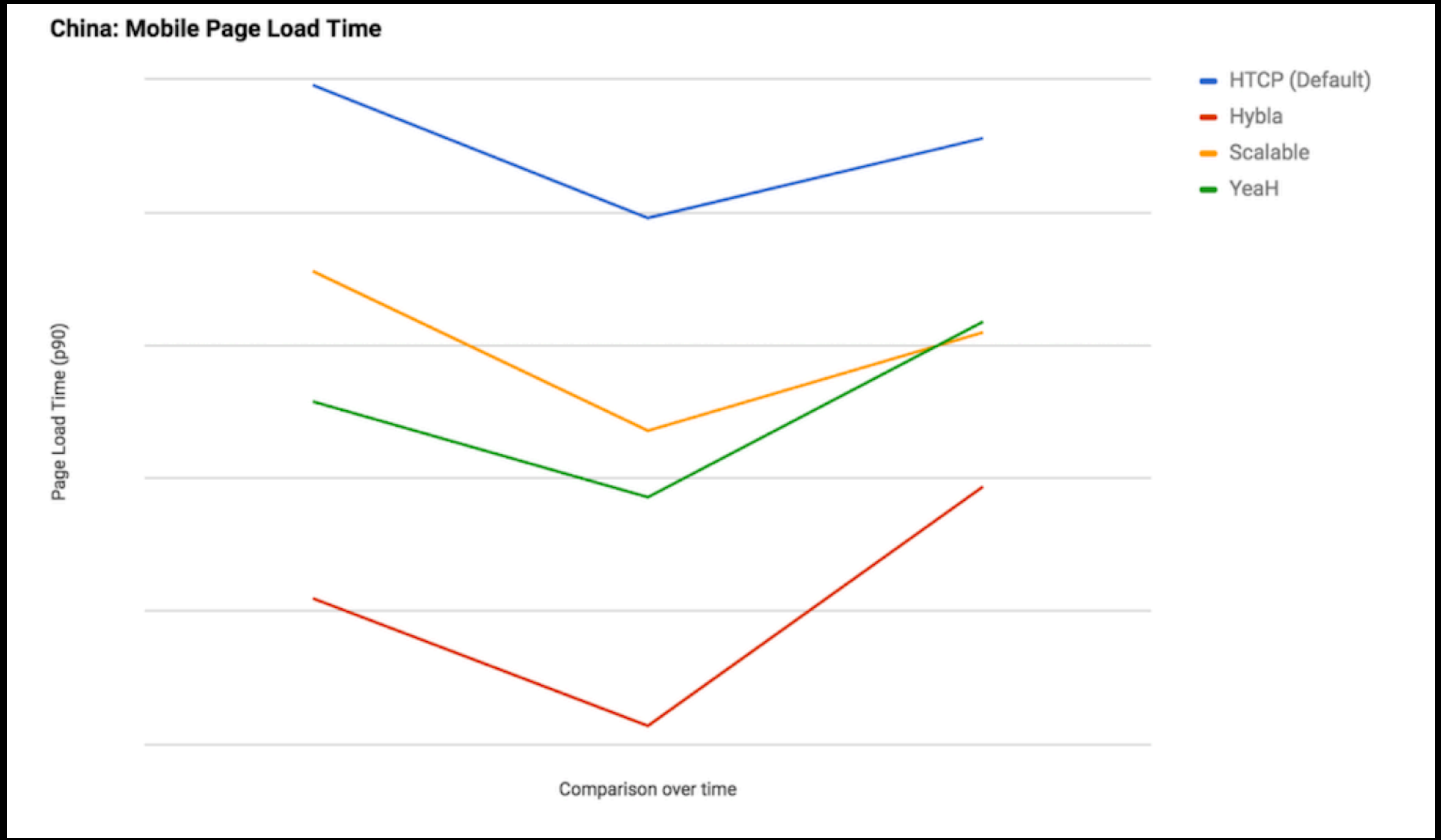


Congestion Control Algorithms

Algorithm	What it does best
TCP-Hybla	Built for networks with long round trip delays. Window update is based on a ratio of current RTT and a reference RTT0.
TCP-Scalable	Built for performance on high-speed, wide area networks. Window updates use fixed increase and decrease parameters.
TCP-YeaH	Built to be fair, efficient, and prevent Lossy-Link penalties. Switches between fast and slow modes, based on an estimate of queued packets.
HTCP	Built for long distance, high-speed transmission. Window updates are based on time since last loss event. This is the default algorithm on our Linux machines.

Site Speed Improvements





The March Ahead

QUIC

- Intended to eventually replace TCP and TLS on the web
- Provides security features like authentication and encryption, that are typically handled by a higher layer protocol
- Establishes multiple connections over UDP
- Avoids head of line blocking by using multiple HTTP streams mapped to multiple QUIC connections

SCTP

- TCP provides both reliable data transfer and strict transmission ordered delivery of data
- Head-of-line blocking in TCP causes delays
- SCTP is a message based reliable protocol
- Reliable transmission of both ordered and unordered data streams.
- Multihoming support and transparent fail over

Thank You