TCP, Congestion Control, Glue Protocols

15 January 2025
Lecture 10

Some Slides Credits: Steve Zdancewic (UPenn)

Topics for Today

- TCP
  - Handshake and Basics
  - Congestion Control
- Glue Protocols
  - ICMP

- Sources in PD:
  - TCP: 5.2
  - TCP Congestion Control: 6.3
  - ICMP: 3.2.8

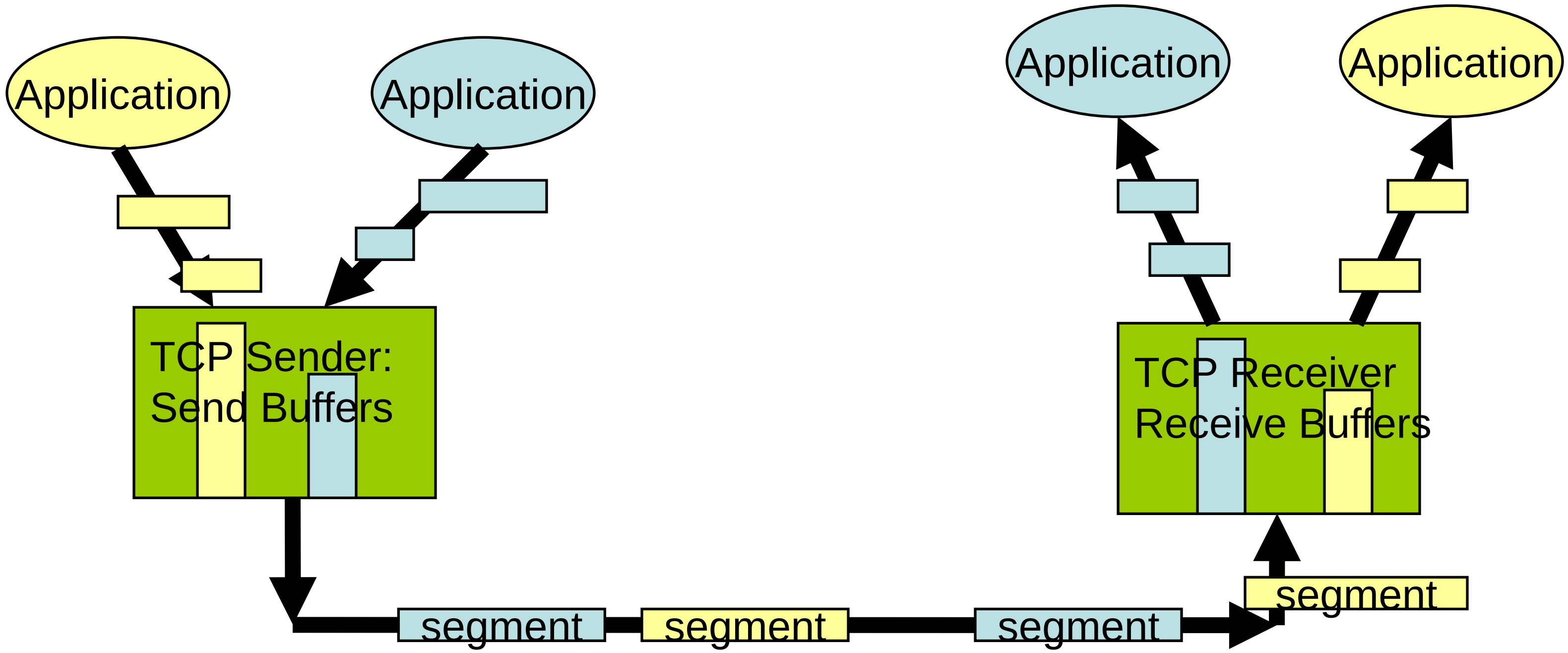
Transmission Control Protocol (TCP)

- Most widely used protocol for reliable byte streams
  - Reliable, in-order delivery of a stream of bytes
  - Full duplex: pair of streams, one in each direction
  - Flow and congestion control mechanisms
  - Like UDP, supports ports

- Built on top of IP (hence TCP/IP)

TCP End-to-End Model

- Buffering corrects errors but may introduce delays



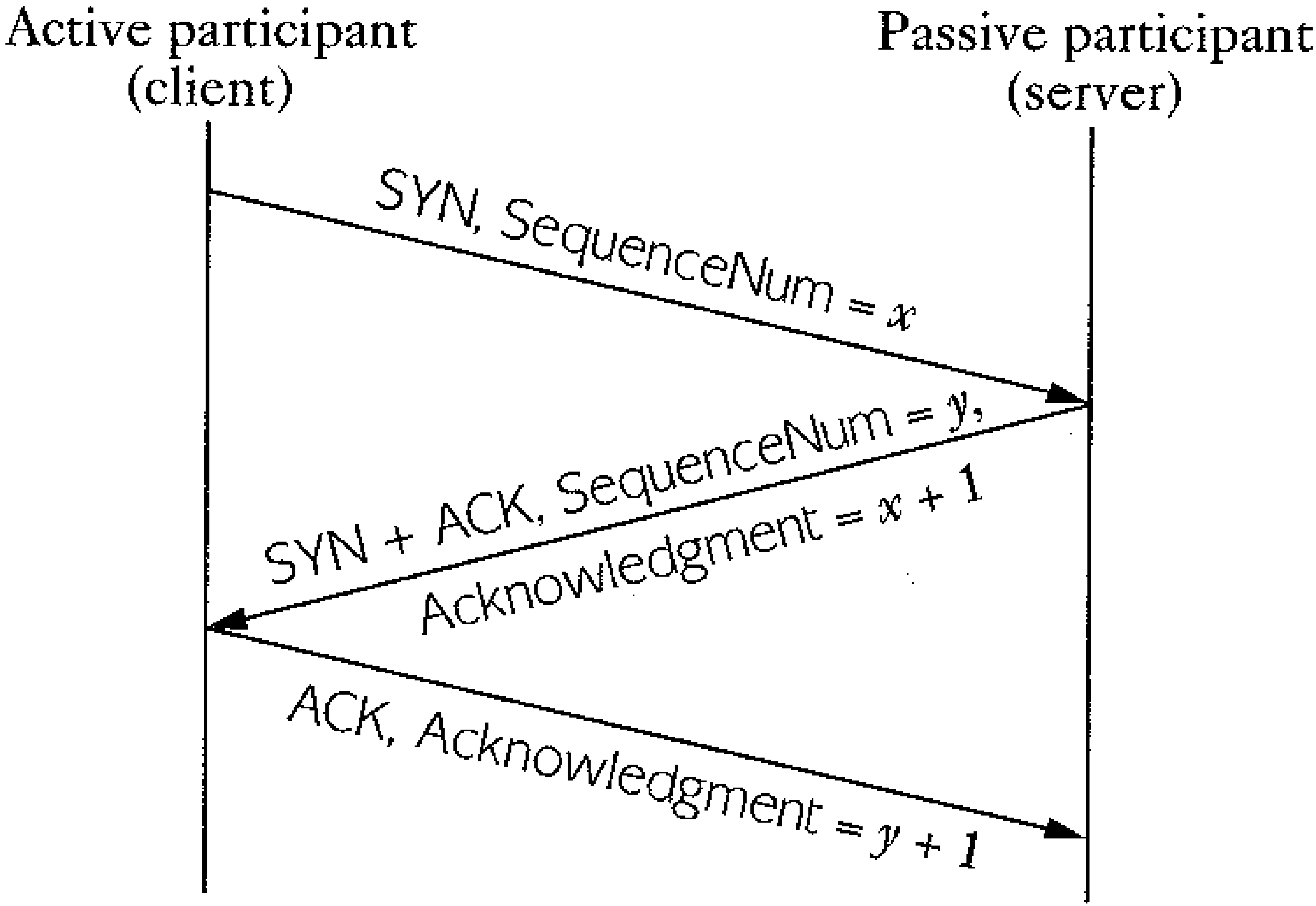
Packet Format

- Flags
  - SYN
  - FIN
  - RESET
  - PUSH
  - URG
  - ACK

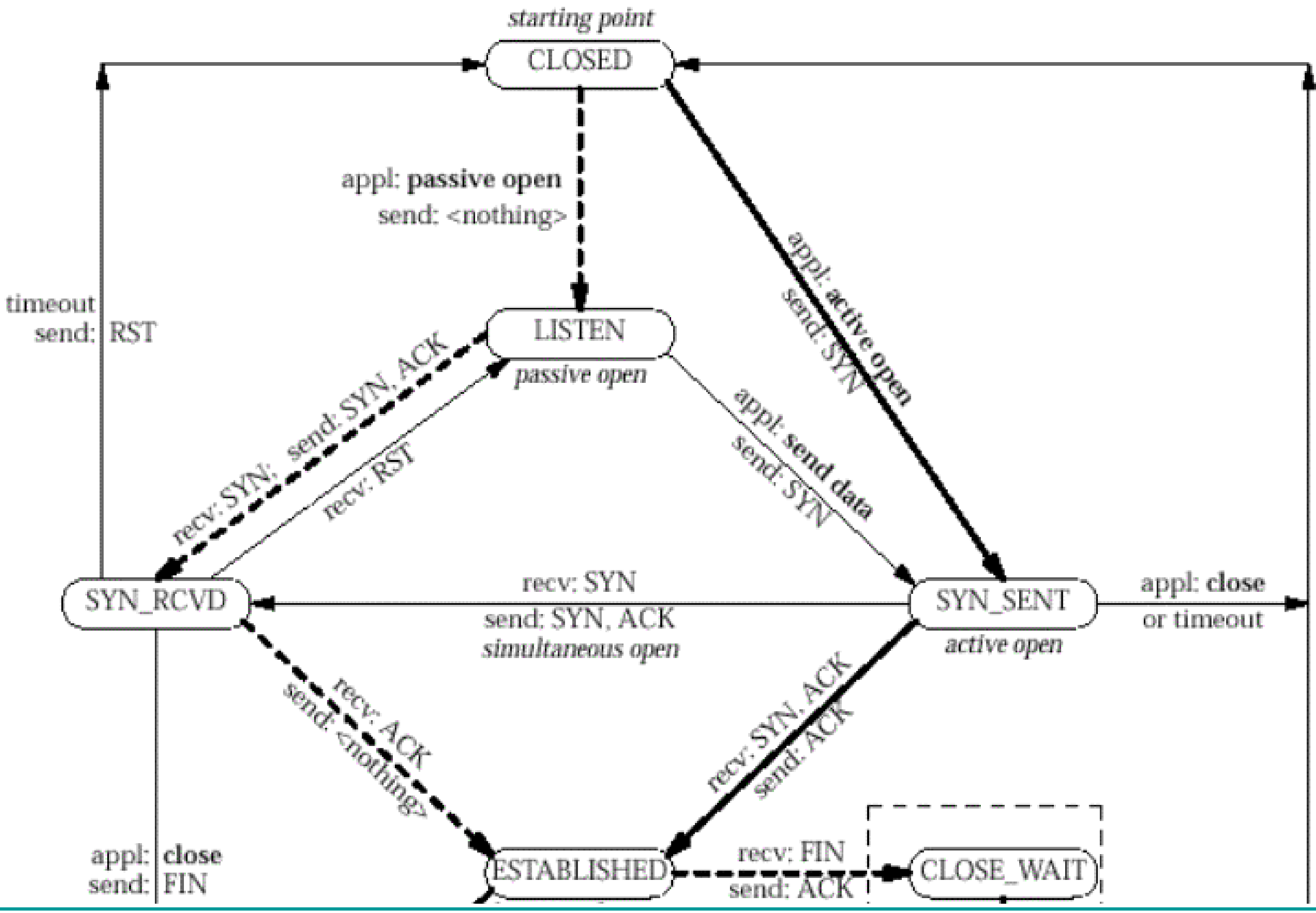
- Fields

<table>
<tr><th colspan="3">0</th><th>15   31</th></tr>
<tr><td colspan="3">Source Port</td><td>Destination Port</td></tr>
<tr><td colspan="4">Sequence Number</td></tr>
<tr><td colspan="4">Acknowledgement</td></tr>
<tr><td>HL</td><td>0</td><td>Flags</td><td>Advertised Window</td></tr>
<tr><td colspan="3">Checksum</td><td>Urgent Pointer</td></tr>
<tr><td colspan="4">Options (variable)</td></tr>
<tr><td colspan="4">Data</td></tr>
</table>

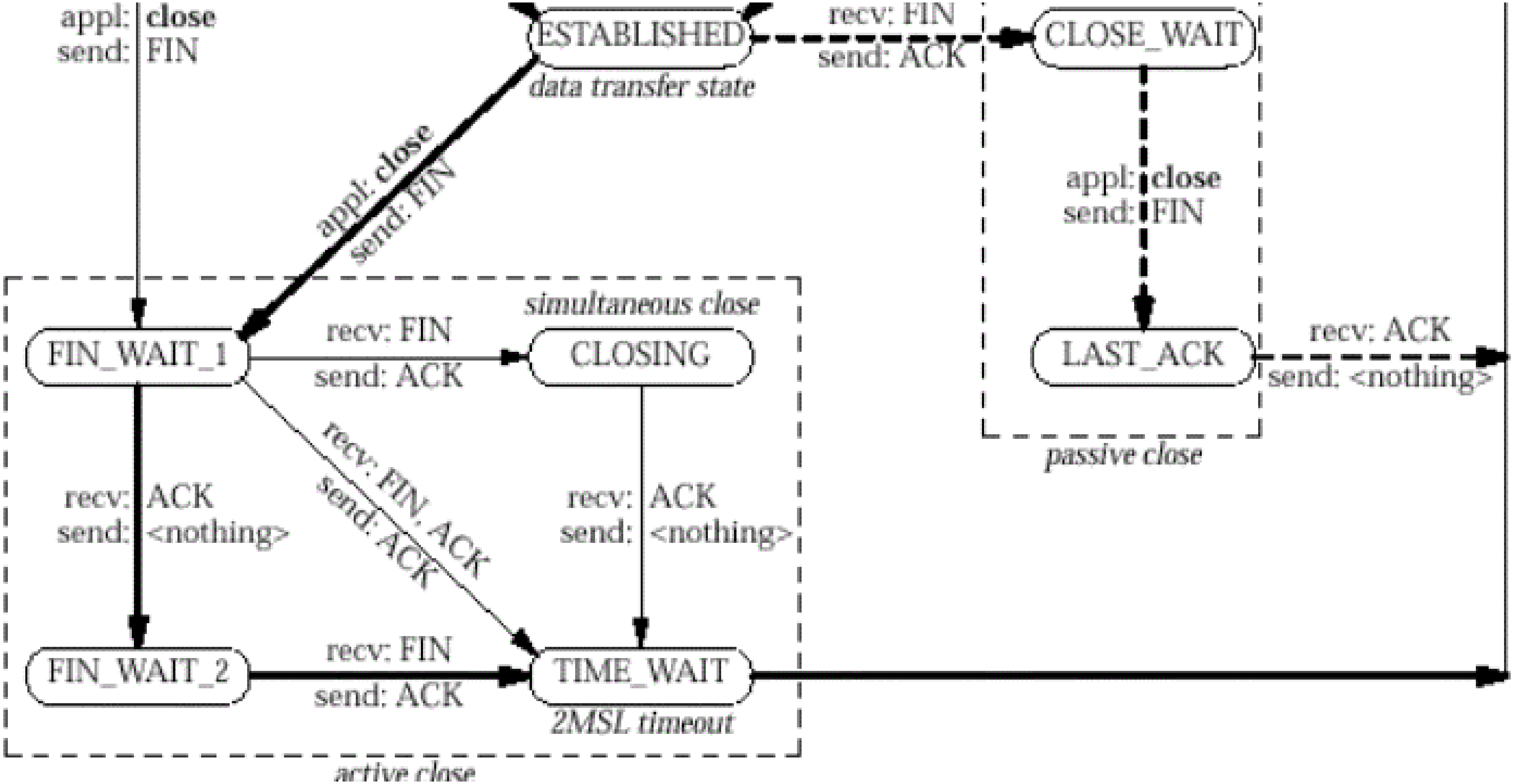
Three-Way Handshake



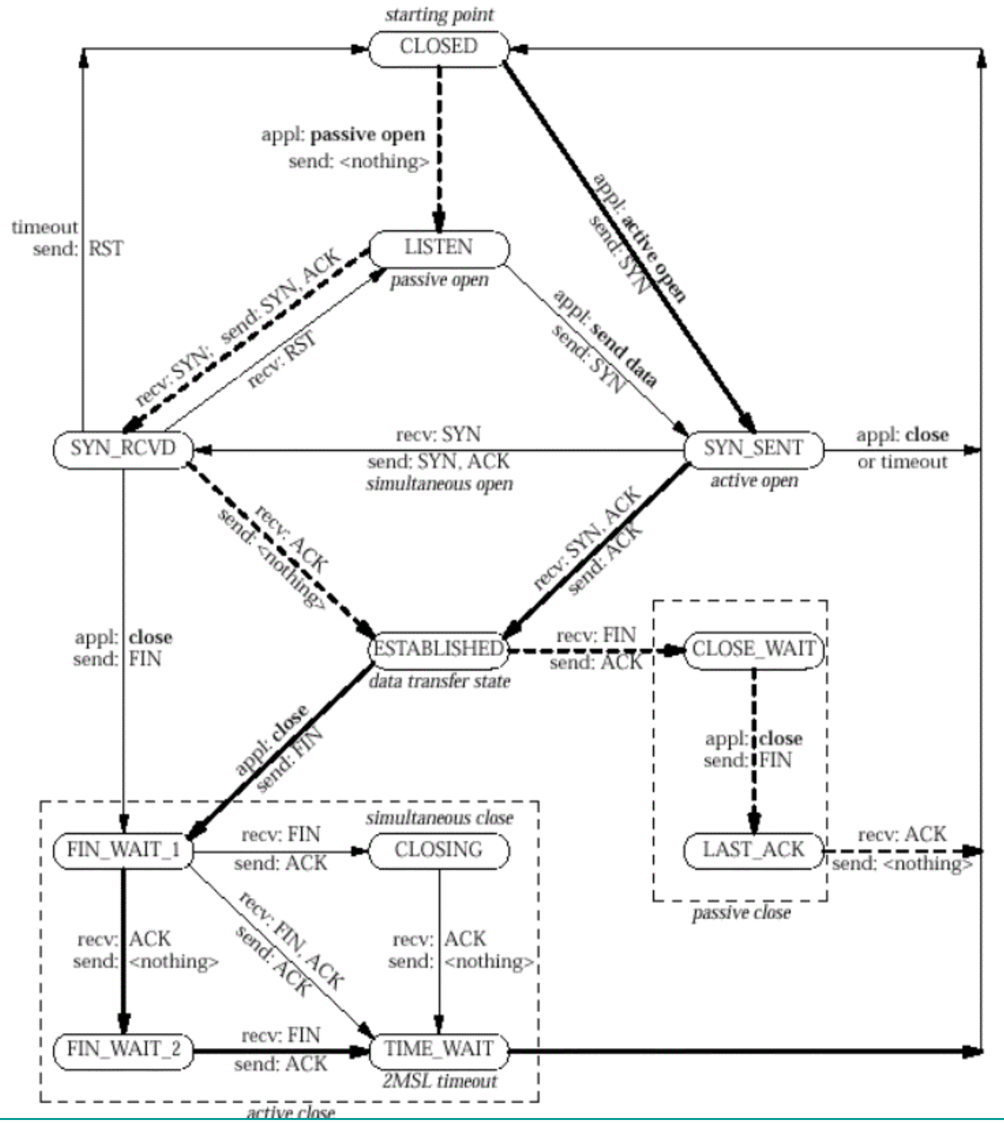
TCP State Transitions (1/2)



TCP State Transitions (2/2)



TCP State Transitions



Two Flags

URG

- Sender has urgent data

SYN	FIN	RST	PSH	URG	ACK
-	-	-	-	1	-

Urgent Pointer =100

Byte #	0	100	101	500
	Urgent Data		Other Data	

- Recipient's TCP forwards the data to the app with priority

PUSH

- Small critical piece of data to send
- Sender sets PUSH bit
- Result:
  - Sender's TCP sends the data immediately (don't buffer until it's close to MTU)
  - Receiver's TCP sends the data to the app immediately, not buffering

TCP Sender and Receiver

TCP Receiver

- Maintains an input buffer for the application
  - Application sees only in-order, correct bytes
- Advertises $\mathrm{aw} = (bsize - filled)$ as advertised window for sliding window
- Keeps track of bytes that arrived ok ($ackBytes$)
- Responds with $ackBytes$ and aw on each send
  - If $aw = 0$, no more space
  - Typically sent with a scaling factor

TCP Sender

- Maintains a sending buffer
- Sending application is blocked until room in the buffer for its write
- Sliding window stores data until acknowledged by receiver
- Sliding window expands and contracts dynamically
  - aw affects size

Simple advertised window interaction

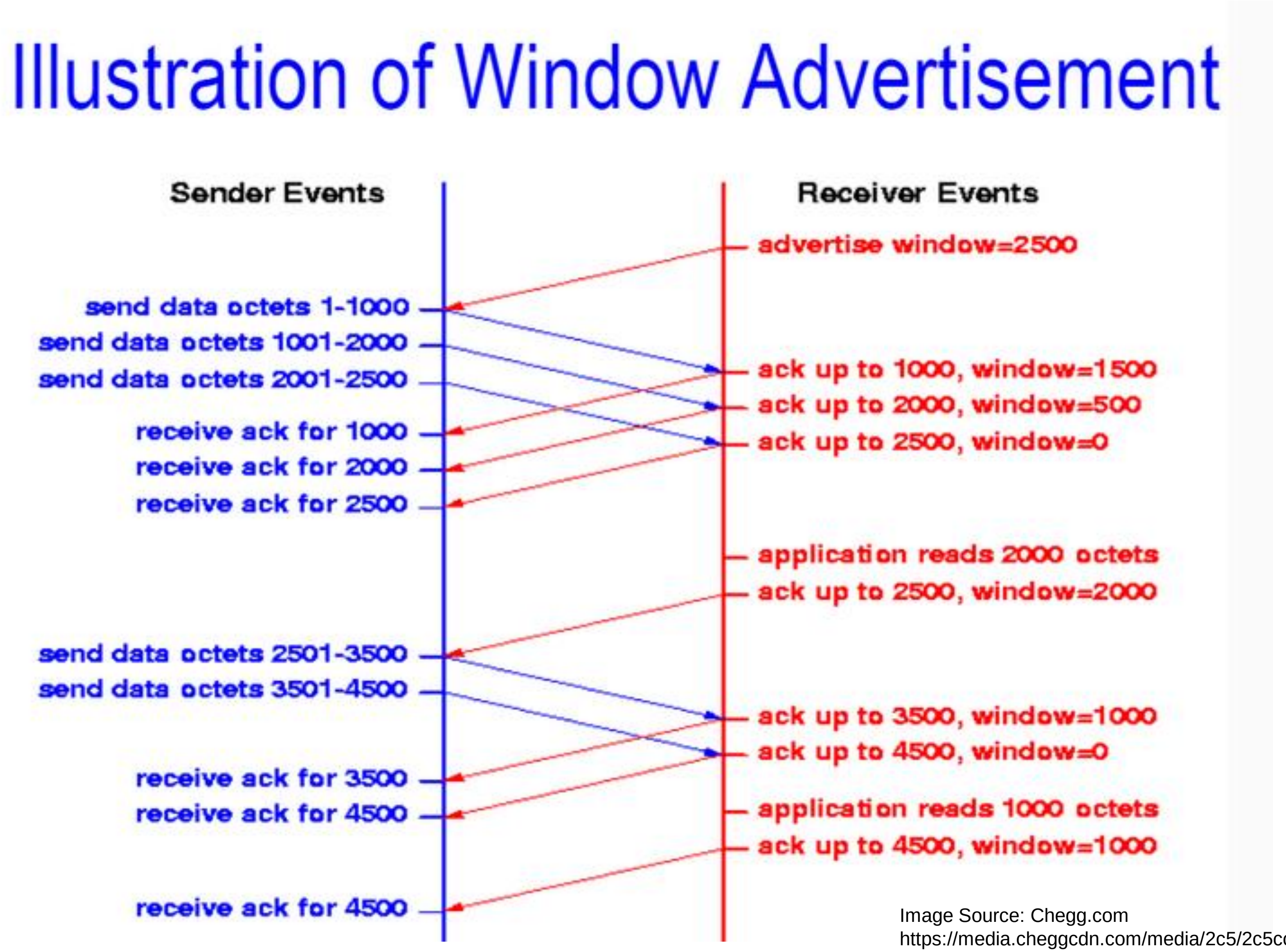


Image Source: Chegg.com
https://media.cheggcdn.com/media/2c5/2c5cc97c-eb97-4376-b57d-0257161bf650/php3ADfPM.png

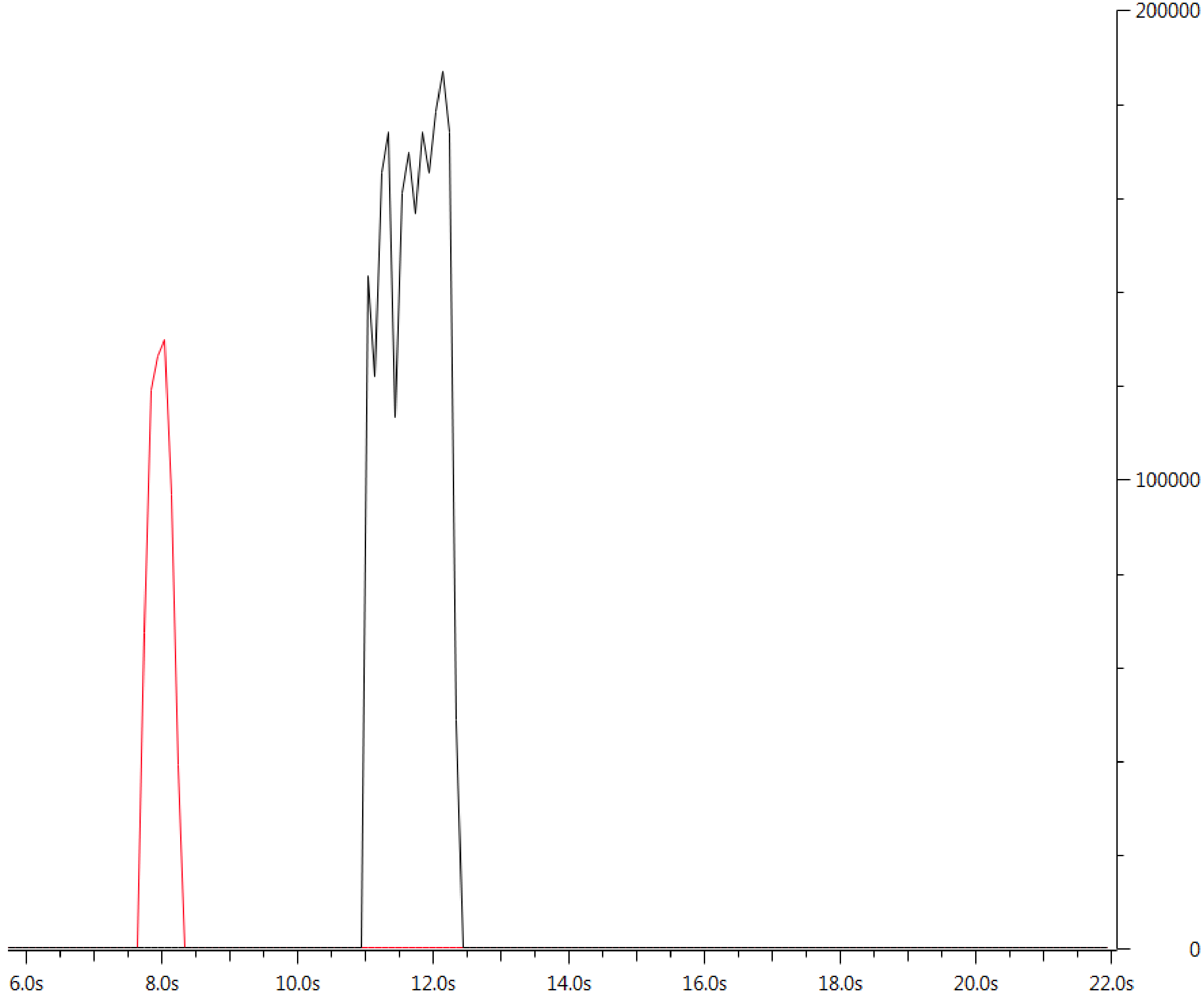
So Far

- TCP
  - Handshake and Basics
  - Congestion Control
- Glue Protocols

TCP Flow & Congestion Control

- Flow vs. Congestion Control
  - Flow control protects the recipient from being overwhelmed (aw)
  - Congestion control protects the network from being overwhelmed.
- TCP Congestion Control
  - Additive Increase / Multiplicative Decrease
  - Slow Start
  - Fast Retransmit and Fast Recovery

TCP versus UDP Sending



Legend:
---- UDP Sending
---- TCP Sending

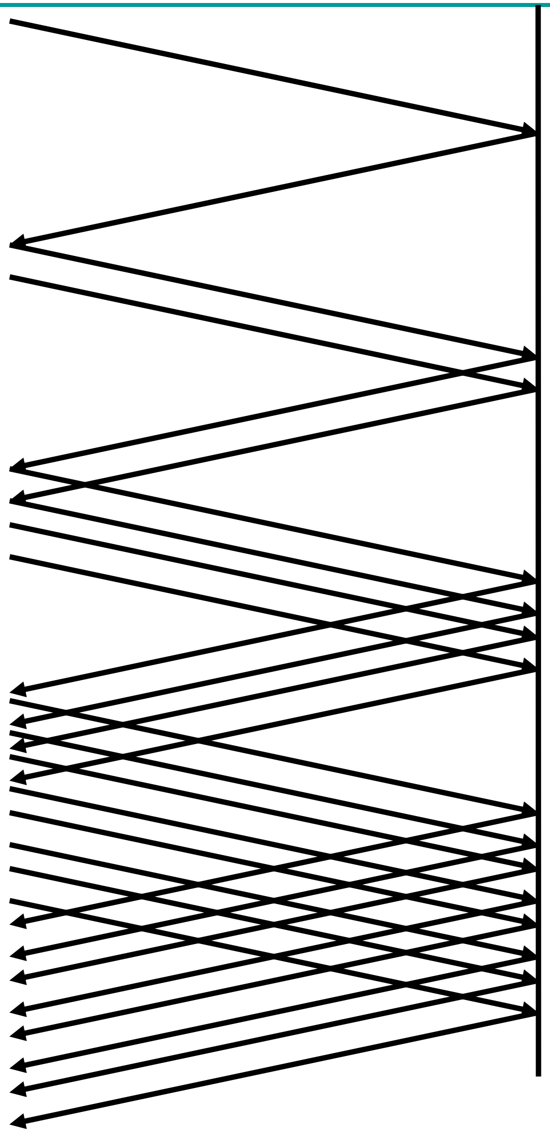
TCP Slow Start

- If we initialize cw to the sender's desired window size and start sending the entire window at once, it could cause immediate network congestion

- Instead, start "slowly" by setting $cw = 2\ packets$ (minimum 576B)

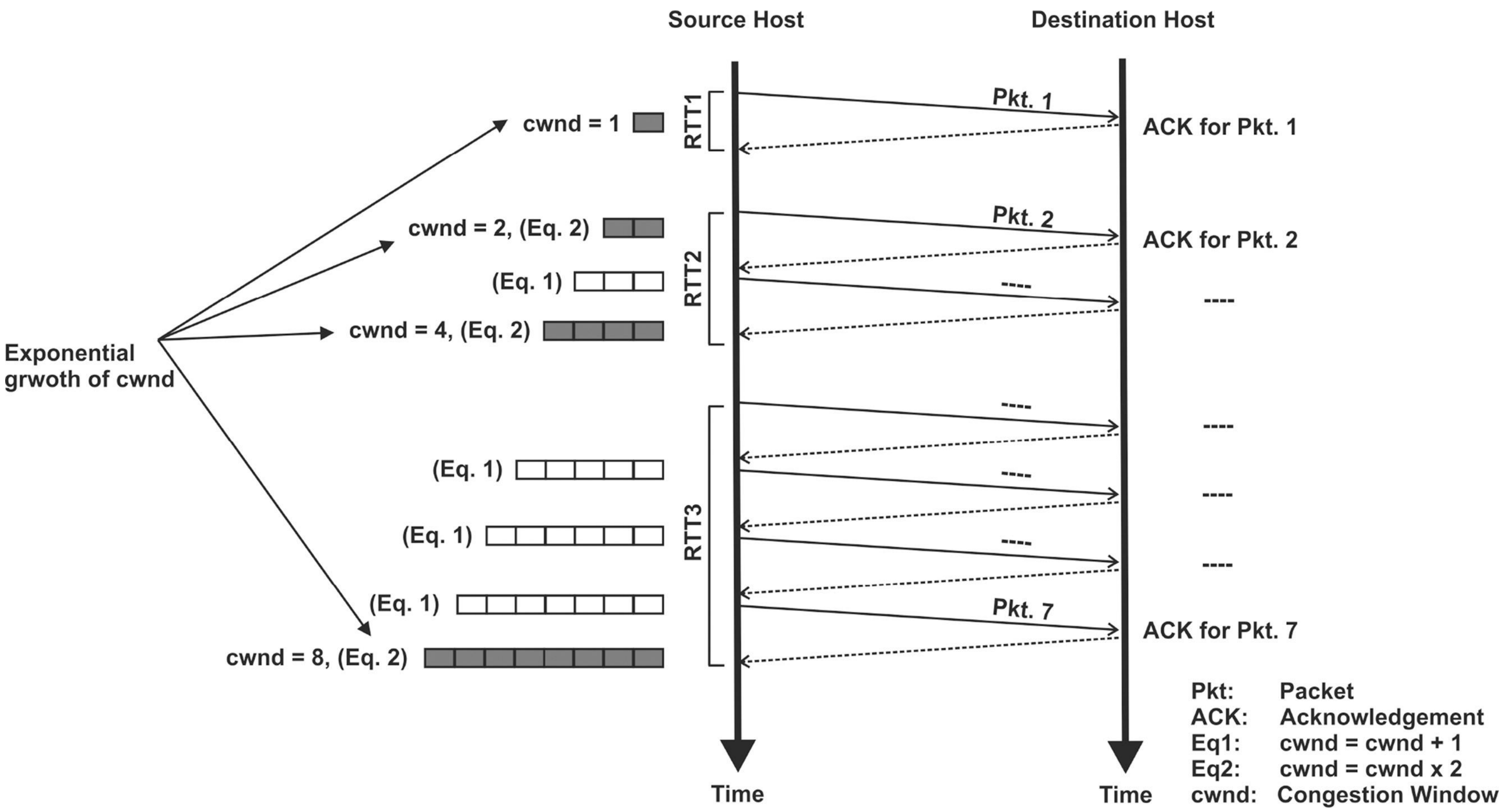
- When an ACK arrives, increase by the number of packets acknowledged (effectively $cw\ {*}{=}\ 2$ each RTT)

- Continue until ACKs do not arrive or flow control dominates.
  - $SWS = \min(cw, aw)$



Slow Start Illustrated

Fig. 2. Exponential growth of congestion window during slow start phase (Wang et al., 2014).



TCP Congestion Control Varieties

- Tahoe (older)

- Reno (Coming Up)
- New Reno (bit different)
  - How to deal with multiple drops and reordering
- Vegas: very different, per packet timers

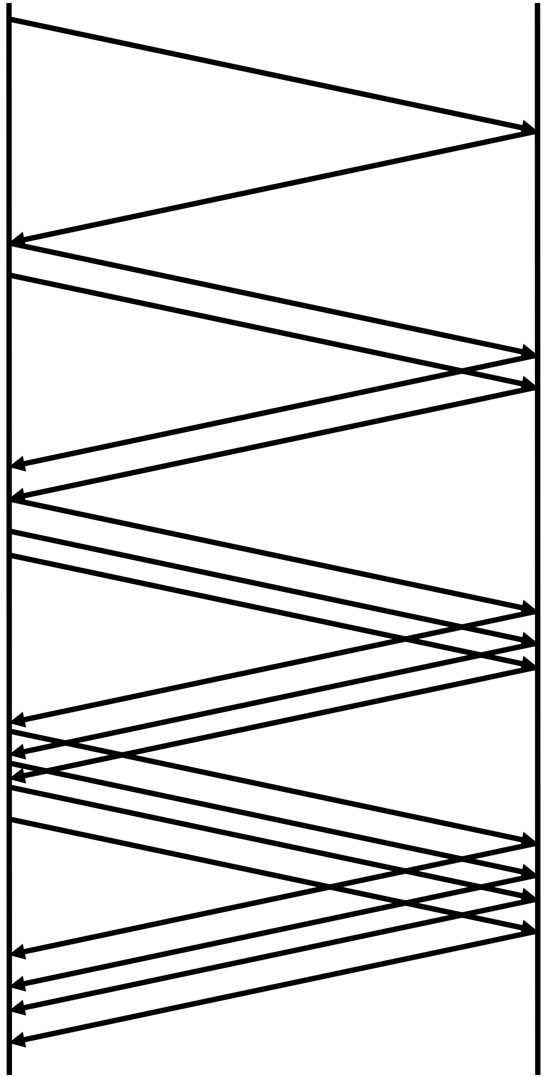
- SACK: Very different, uses selective acknowledgments
  - What arrived – ranges of noncumulative ACKs

Image credits: https://www.redbubble.com, ebay.com, http://gclipart.com/clipart-potato_8434/

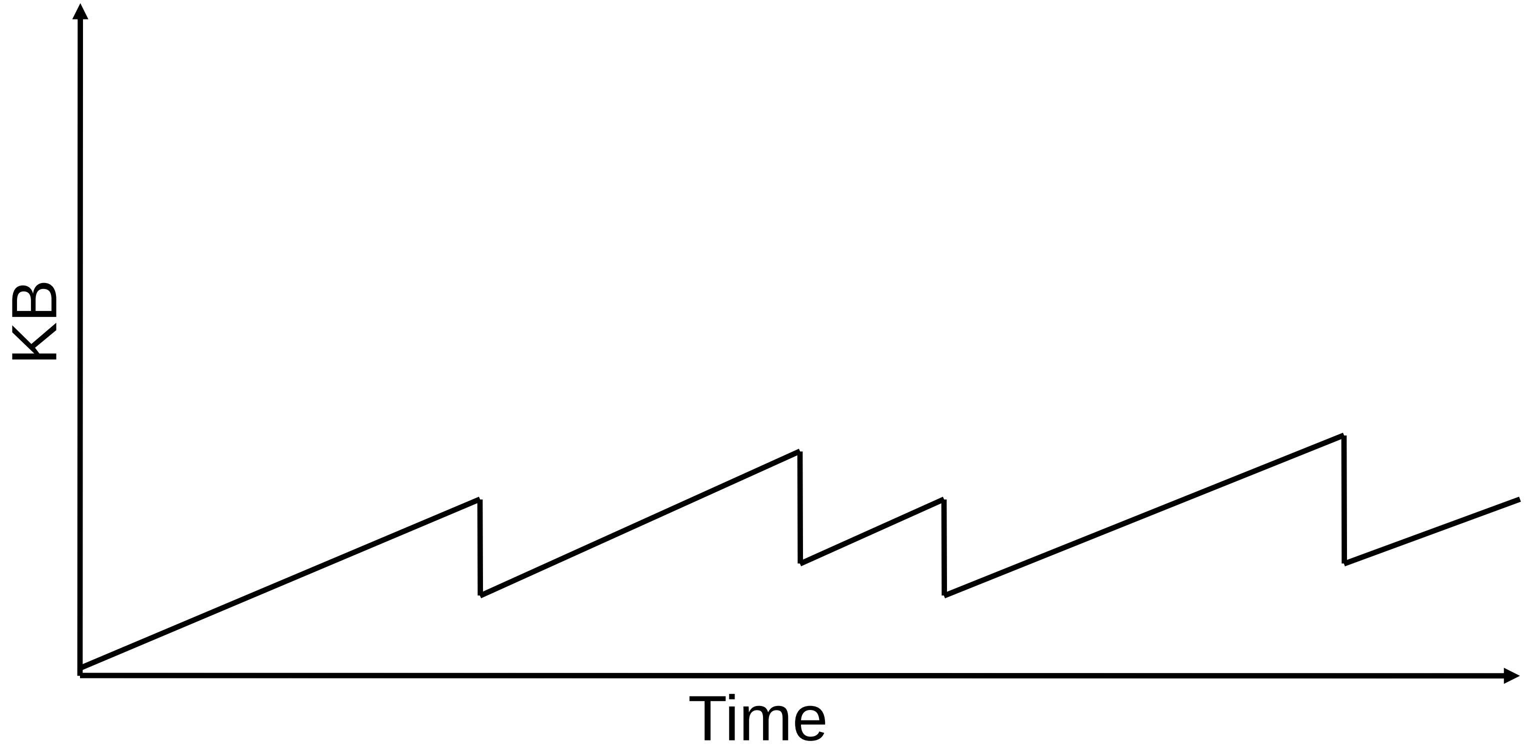
Image source: cafepress.com

Reno AIMD

- Variable *CongestionWindow* (cw) is used to control the number of unacknowledged transmissions (in addition to aw)
- cw is increased linearly each RTT until timeouts for ACKs are missed.
- When an ACK is missed, cw is decreased by half ($cw \ *= 0.5$) to reduce the pressure on the network quickly.
- Called "additive increase / multiplicative decrease".



TCP Sawtooth Pattern



TCP CUBIC

- Meant for Long Latency High Bandwidth
  - Think cloud

$$W(t) = C(t-K)^3 + W_{max} \tag{1}$$

where C is a CUBIC parameter, t is the elapsed time from the last window reduction, and K is the time period that the above function takes to increase W to W_{max} when there is no further loss event and is calculated by using the following equation:

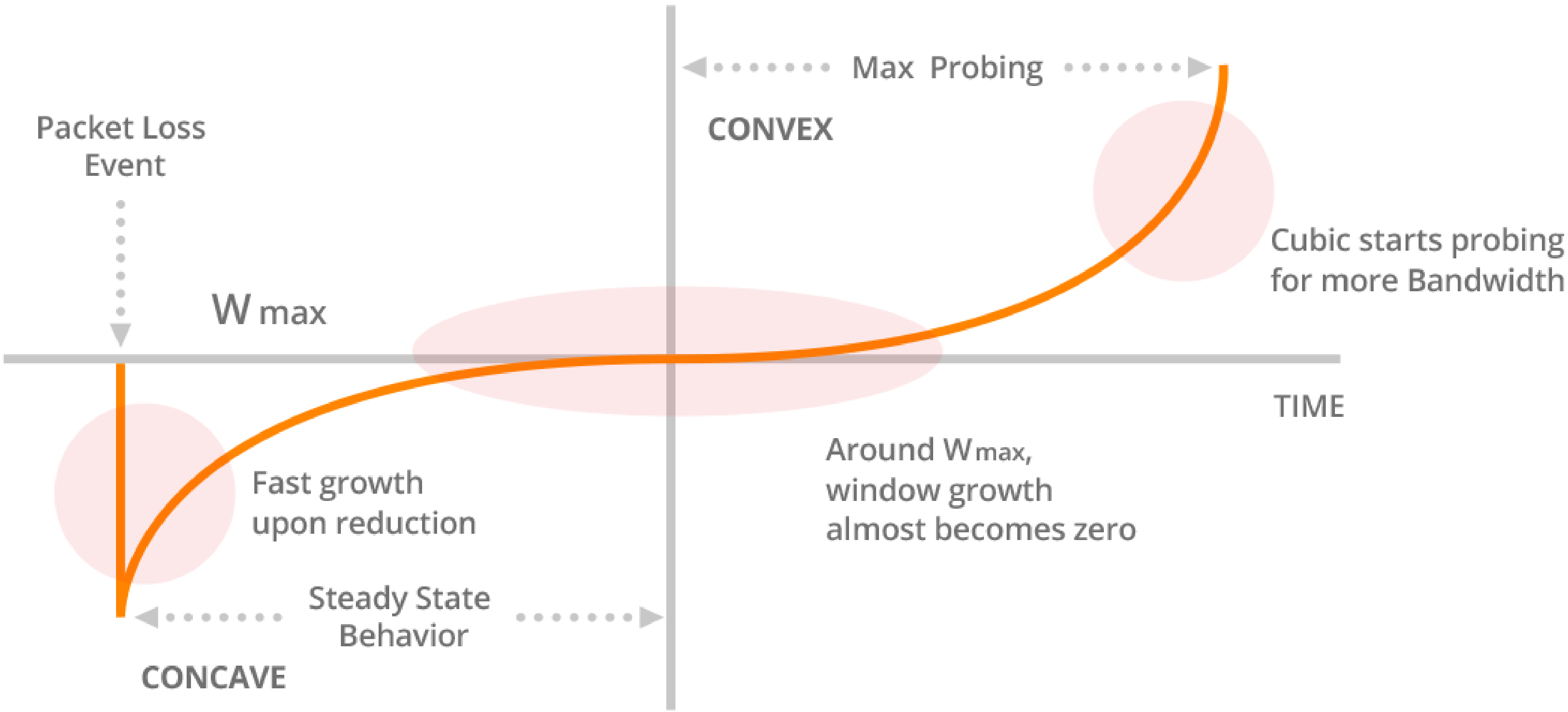
$$K = \sqrt[3]{\frac{W_{max}\beta}{C}} \tag{2}$$

- Instead of AIMD drop by 50% → Drop by 20%
- Window increase based on:
  - $\beta = 0.2, C = 0.4$

Sangtae Ha, Injong Rhee, and Lisong Xu. 2008. CUBIC: a new TCP-friendly high-speed TCP variant. SIGOPS Oper. Syst. Rev. 42, 5 (July 2008), 64–74. https://doi.org/10.1145/1400097.1400105

TCP CUBIC's Graph

https://www.noction.com/blog/tcp-transmission-control-protocol-congestion-control



Congestion Control Algorithms

Operating System/System	Default TCP Congestion Control Algorithm
MacOS	TCP CUBIC
Microsoft Windows	TCP Compound
Linux	TCP CUBIC
Sun Solaris	TCP Fusion
YouTube (Google)	TCP BBR
Android	TCP CUBIC
iOS	TCP CUBIC
Amazon CloudFront	TCP BBR
Facebook	COPA (?) over QUIC

AI for Congestion Control?

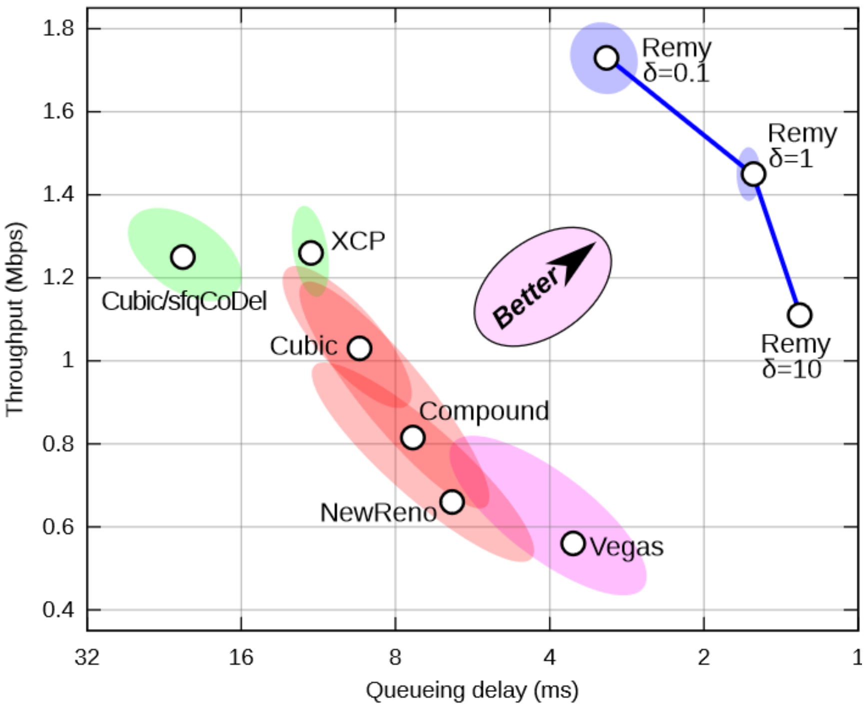
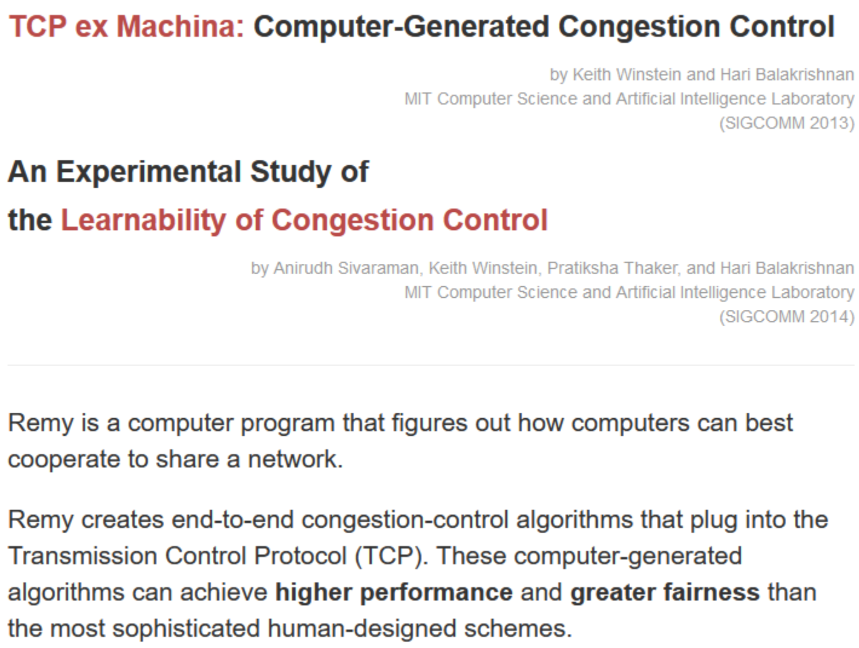


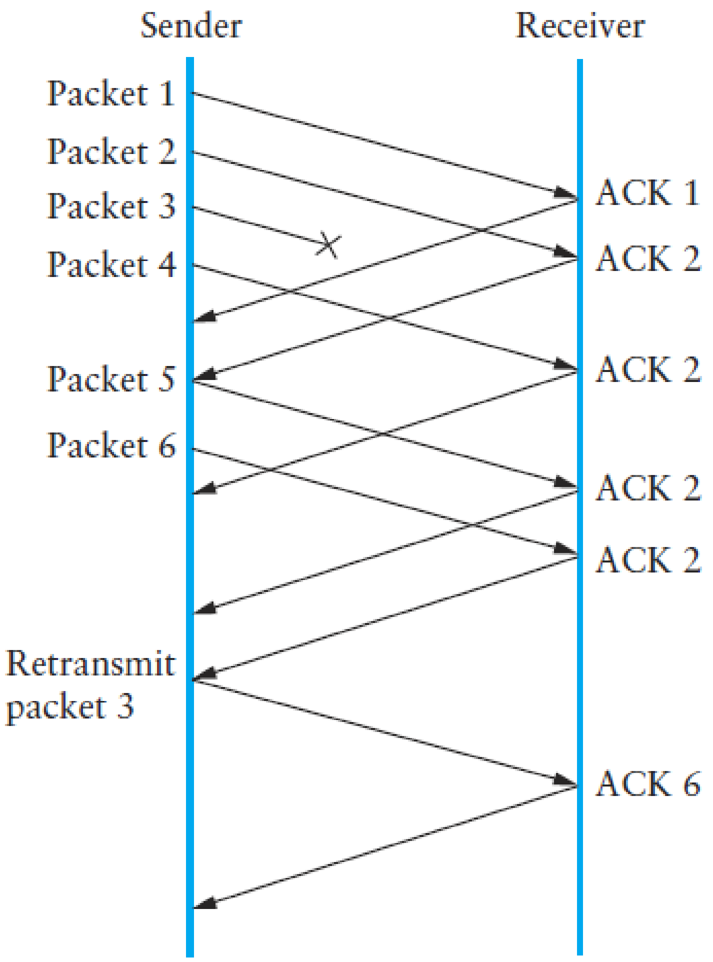
Figure 4: Results for each of the schemes over a 15 Mbps dumbbell topology with $n = 8$ senders, each alternating between flows of exponentially-distributed byte length (mean 100 kilobytes) and exponentially-distributed off time (mean 0.5 s). Medians and 1-σ ellipses are shown. The blue line represents the efficient frontier, which here is defined entirely by the RemyCCs.

https://web.mit.edu/remy/TCPexMachina.pdf



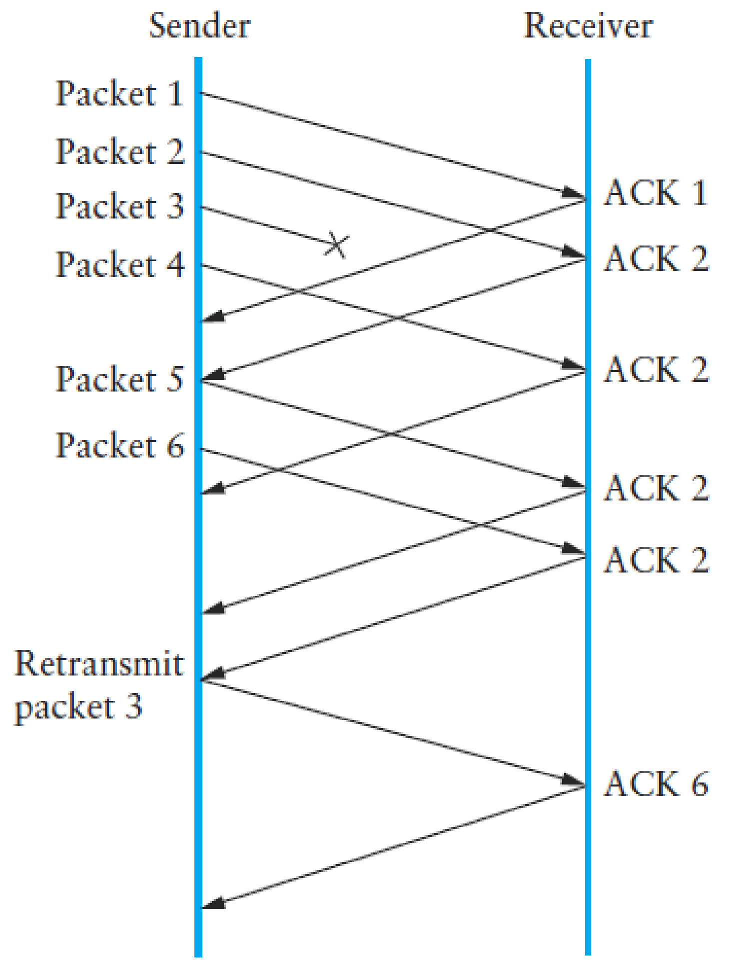
TCP: Fast Retransmit

- Lone packet losses can slow down transmission
- The timeout must be reached and it's likely that the connection will go dead if the SWS or Congestion Window is reached



TCP: Fast Retransmit

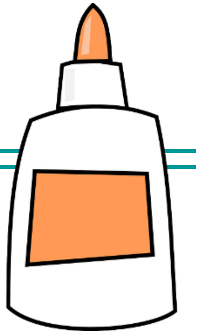
- Alternative: Fast Retransmit
  - Send cumulative ACKs as we saw in Sliding Window
  - If 3 duplicate ACKs arrive, retransmit the next one missing
    - Duplicate ACK means packets with no data, ACKing something already ACK'd and with same aw
  - Can increase throughput by 20%
  - Doesn't solve all the problems (if SWS is small or during Slow Start)



So Far

- TCP
  - Handshake and Basics
  - Congestion Control
- Glue Protocols

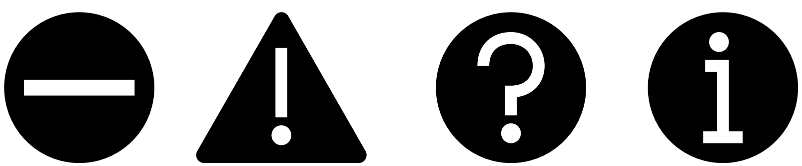
"Glue" Protocols



- Need some way to handle error conditions
- Need some way to map addresses at one level to addresses at another level.
  - Example: Machine addresses (Ethernet) to IP addresses
- Need someway to provide IP addresses
  - What about computers entering/leaving?
- Need some way to provide human-readable addresses
  - So you can write www.kinneret.ac.il instead of 212.150.112.29

ICMP: Internet Control Message Protocol

- Collection of error & control messages
- Sent back to the source when Router or Host cannot process packet correctly
- Error Examples:
  - Destination host unreachable
  - Reassembly process failed
  - TTL reached 0
  - IP Header Checksum failed
- Control Example:
  - Redirect – tells source about a better route



ICMP Sample Trace

No.	Time	Source	Destination	Protocol	Length	Info
155	22.572657000	10.0.0.3	64.233.166.160	ICMP	106	Echo (ping) request id=0x0001, seq=44/11264, ttl=1 (no response found!)
156	22.573156000	10.0.0.138	10.0.0.3	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
157	22.627346000	10.0.0.3	64.233.166.160	ICMP	106	Echo (ping) request id=0x0001, seq=45/11520, ttl=1 (no response found!)
158	22.628191000	10.0.0.138	10.0.0.3	ICMP	134	Time-to-live exceeded (Time to live exceeded in transit)
159	22.628941000	10.0.0.3	64.233.166.160	ICMP	106	Echo (ping) request id=0x0001, seq=46/11776, ttl=1 (no response found!)

```
⊞ Frame 155: 106 bytes on wire (848 bits), 106 bytes captured (848 bits) on interface 0
⊞ Ethernet II, Src: Dell_e6:7f:66 (44:a8:42:e6:7f:66), Dst: D-Link_75:8a:ab (00:22:b0:75:8a:ab)
⊞ Internet Protocol Version 4, Src: 10.0.0.3 (10.0.0.3), Dst: 64.233.166.160 (64.233.166.160)
⊟ Internet Control Message Protocol
    Type: 8 (Echo (ping) request)
    Code: 0
    Checksum: 0xf7d2 [correct]
    Identifier (BE): 1 (0x0001)
    Identifier (LE): 256 (0x0100)
    Sequence number (BE): 44 (0x002c)
    Sequence number (LE): 11264 (0x2c00)
  ⊞ [No response seen]
  ⊞ Data (64 bytes)
```

Conclusion

- TCP
  - Handshake and Basics
  - Congestion Control
- Glue Protocols