Distributed Computer Systems Engineering

CIS 508: Lecture 9
Networking Protocols I

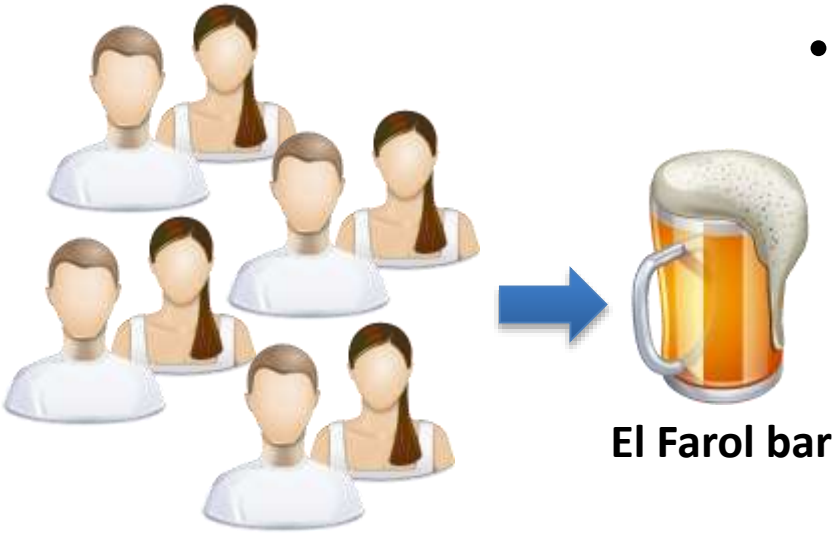
Lecturer: Sid C.K. Chau
Email: ckchau@masdar.ac.ae

Masdar INSTITUTE CIS

Distributed Resource Allocation

- Distributed systems often share resource
  - E.g. network capacity, computation, storage
- Not all processes are aware of the state of resource sharing
  - Communication is often needed between resource providers and resource customers
  - There may be delay between usage congestion (or idleness) and the awareness of resource customers
- Key example: sharing bandwidth over the Internet
- End-users need to adapt to the dynamic resource availability by changing data rate
- *TCP* is a congestion control algorithm that allows distributed resource allocation over network capacity

Toy Problem: El Farol Bar

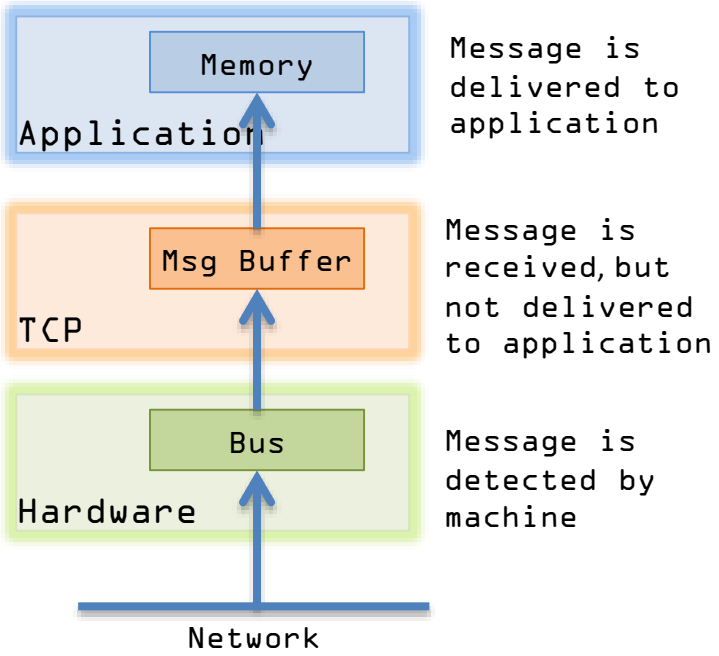


- At the weekend, people like to go to El Farol Bar to chill out
- El Farol Bar is finite, and can only accommodate limited number of people
  - Too many people: overcrowded
  - Too few people: not fun
  - Should you go to El Farol Bar or not?
- Every person makes his/her decision in a distributed manner
- A distributed congestion control problem

TCP (Transport Control Protocol)

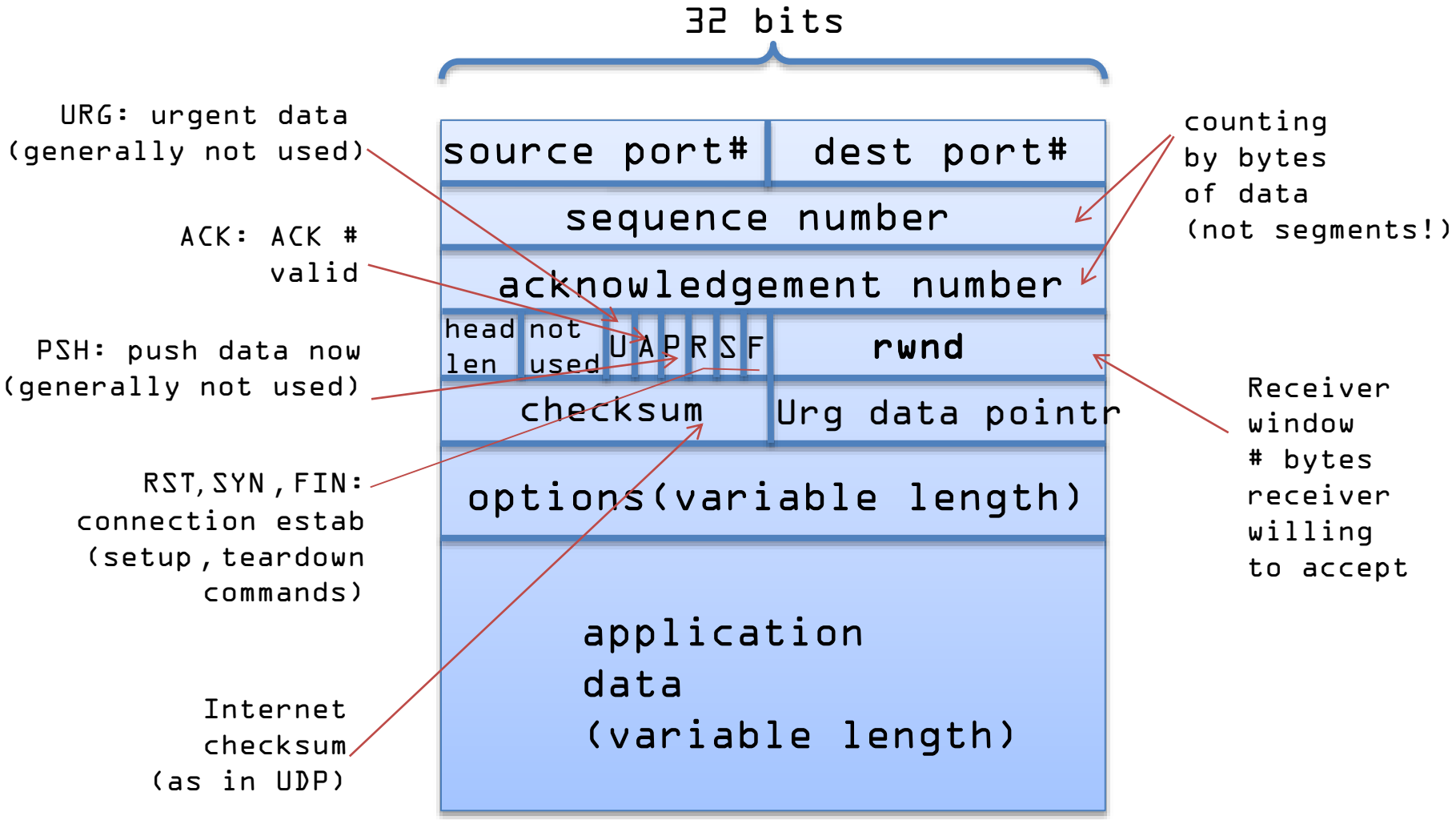
- *Point-to-point*
  - One sender, one receiver
- *Reliable end-to-end transmission*
  - Overcome unreliable link-to-link
- *Full duplex data*
  - Bi-directional data flow in same connection
  - MSS: maximum segment size
- *Connection-oriented*
  - Handshaking (exchange of control messages) initiate the session states of sender & receiver before data exchange
- *Flow controlled*
  - Sender will not overwhelm receiver

TCP Flow Control

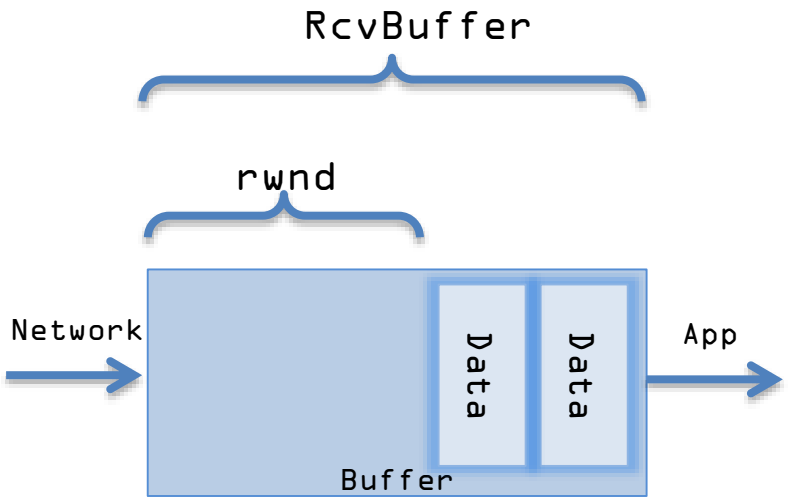


- *Flow Control*
  - Receiver controls sender
  - Sender will not overflow receiver's buffer by transmitting too much and too fast

TCP Packet Structure



TCP Flow Control

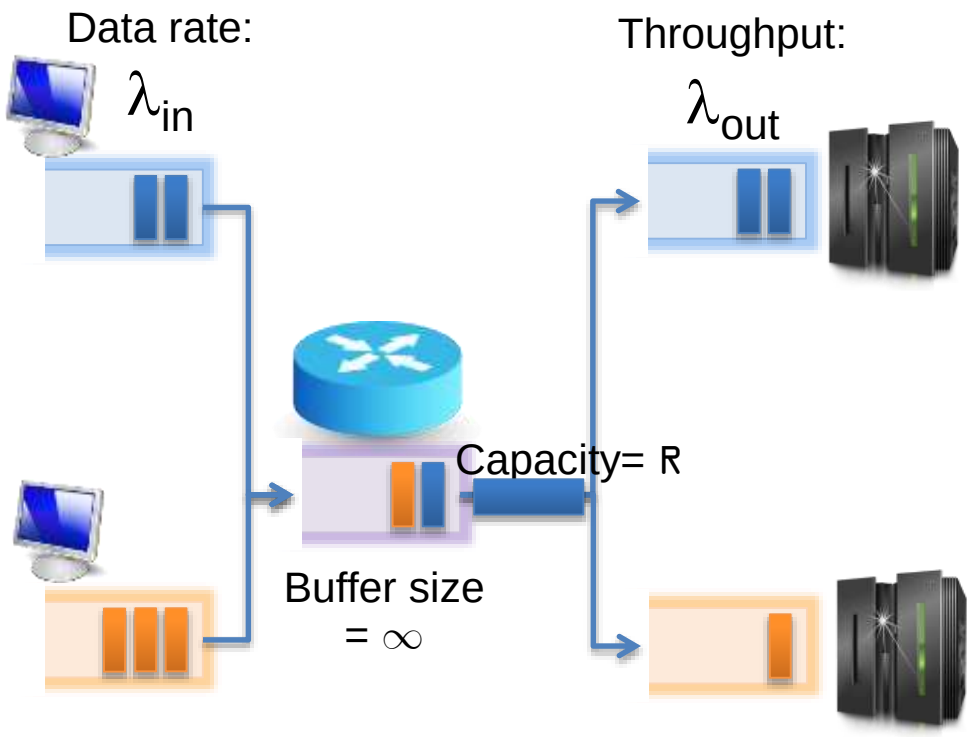


- Receiver advertises free buffer space by including `rwnd` value in TCP header of receiver-to-sender segments
  - `RcvBuffer` size set via socket options (default is 4096 bytes)
  - many operating systems autoadjust `RcvBuffer`
- Sender limits amount of unacknowledged ("in-flight") data to receiver's `rwnd` value
- Guarantee receiver buffer will not overflow

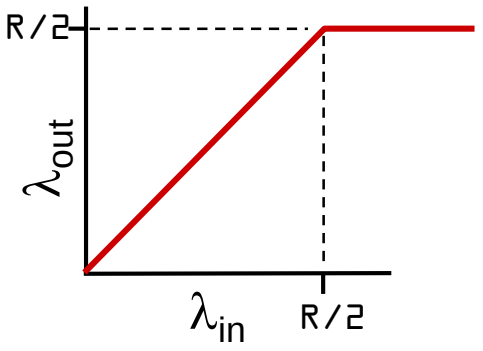
Principles of Congestion Control

- *Congestion*
  - Basic idea
    - "*Too many sources sending too much data too fast for the network to handle*"
  - Different from flow control
    - "*One sender sending too much data too fast for the receiver to handle*"
  - Problems
    - Lost packets (buffer overflow at routers)
    - Long delays (queueing in router buffers)

Congestion: Example 1

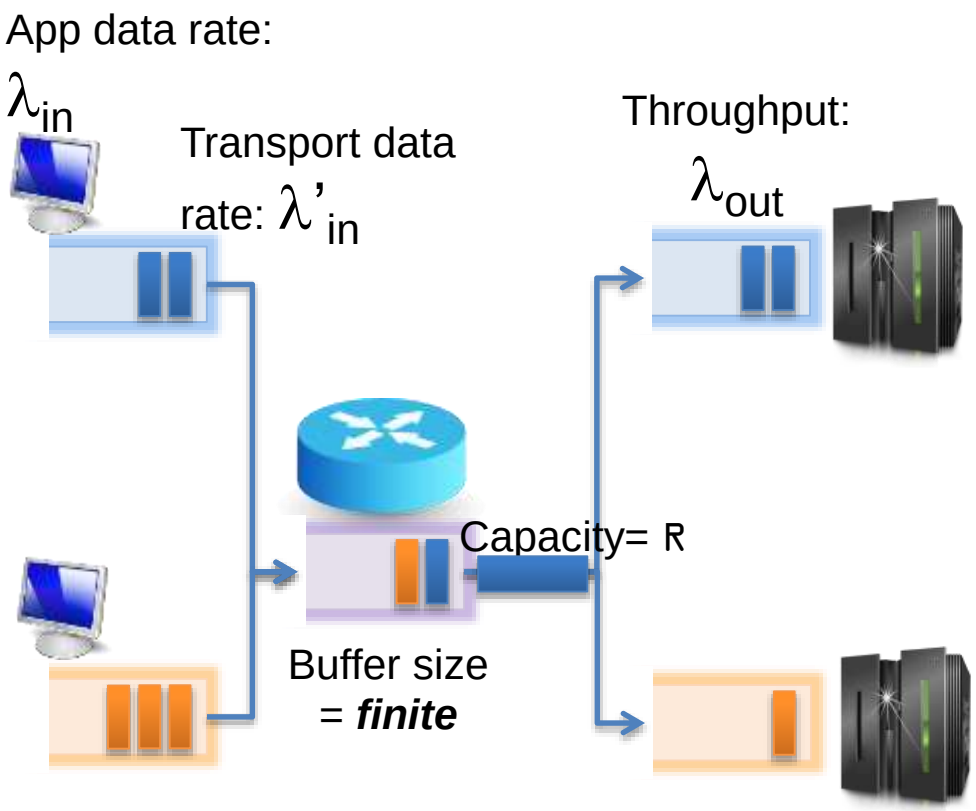


- Two senders, two receivers
- One router, infinite buffers
- Output link capacity R
- No retransmission

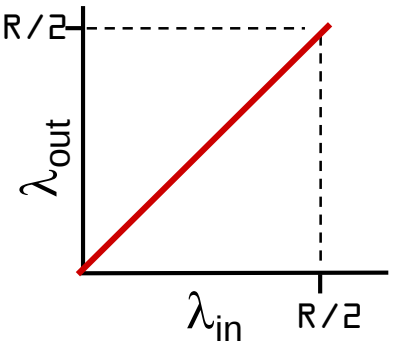


- maximum per-connection throughput R/2

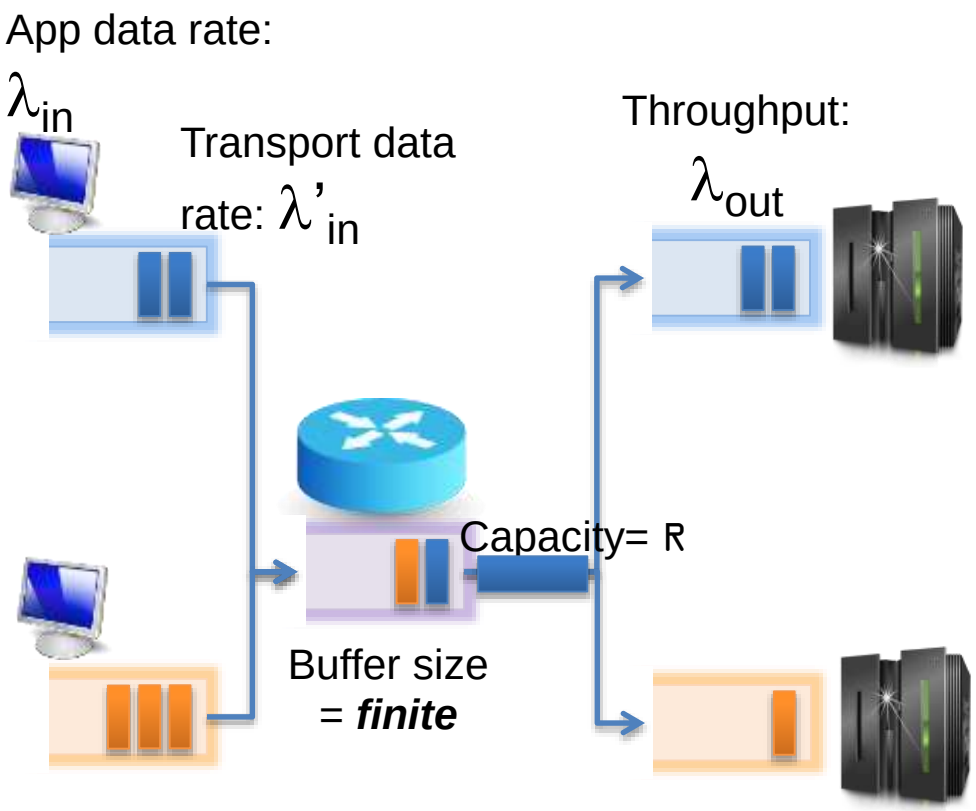
Congestion: Example 2



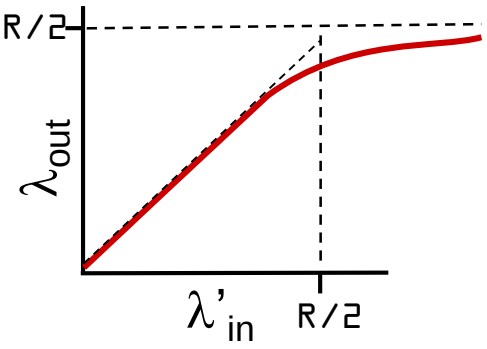
- One router, finite buffers
- Packets may be dropped
- Sender retransmits packets
- Transport data rate (including retransmission) is larger than application data rate
  - $\lambda'_{in} \geq \lambda_{in}$



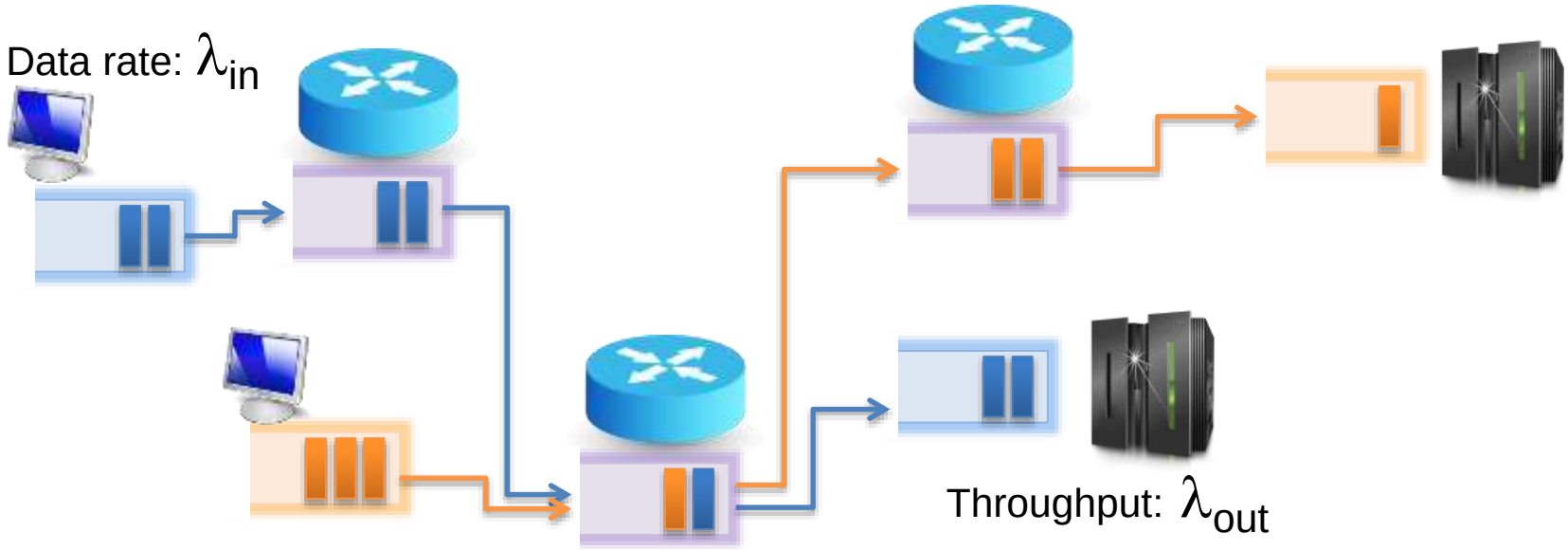
Congestion: Example 2



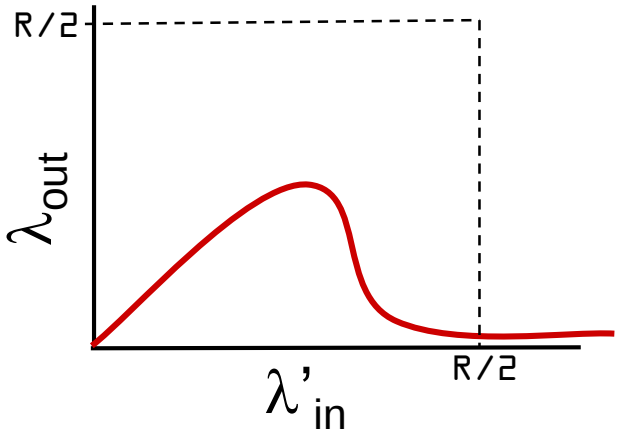
- $\lambda'_{in} \geq \lambda_{in}$
- Ideally, sender only resends if packet known to be lost
- When sending at R/2, some packets are retransmissions but asymptotic throughput is still R/2



Congestion: Example 3

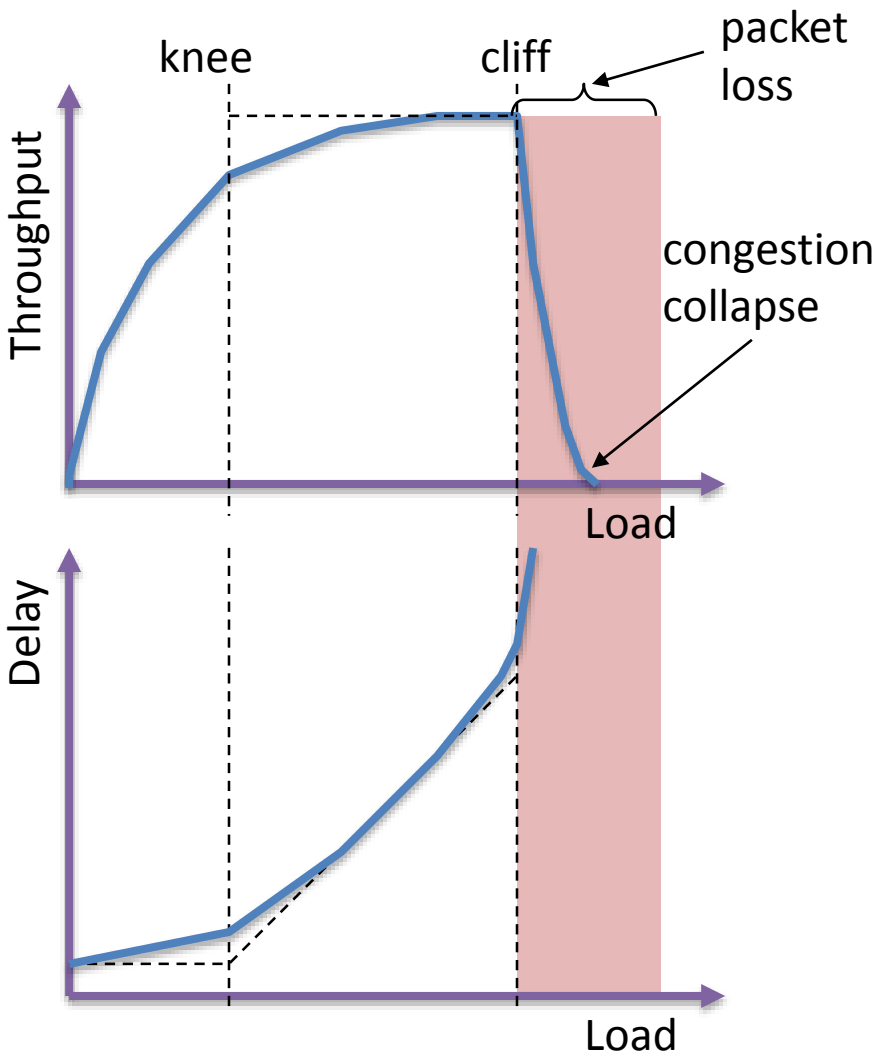


- Some intermediate links are shared
- When packet dropped, upstream transmission capacity used for that packet was wasted



Congestion Collapse

- Knee – point after which
  - Throughput increases very slow
  - Delay increases fast
- Cliff – point after which
  - Throughput starts to decrease very fast to zero (congestion collapse)
  - Delay approaches infinity
- In queueing theory
  - Delay = 1/(1 – utilization)



Congestion Control Approaches

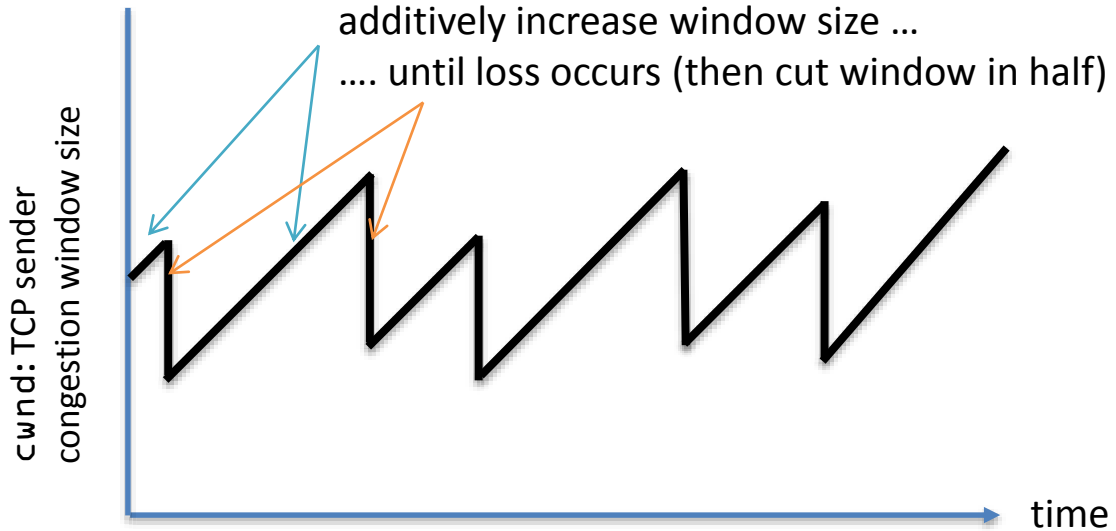
Two approaches:

1. End-to-end congestion control
   - No explicit feedback from network
   - Congestion inferred from end-system observed loss, delay
   - Approach taken by TCP
2. Network-assisted congestion control
   - Routers provide feedback to end systems
   - Single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
   - Explicit rate for sender to send

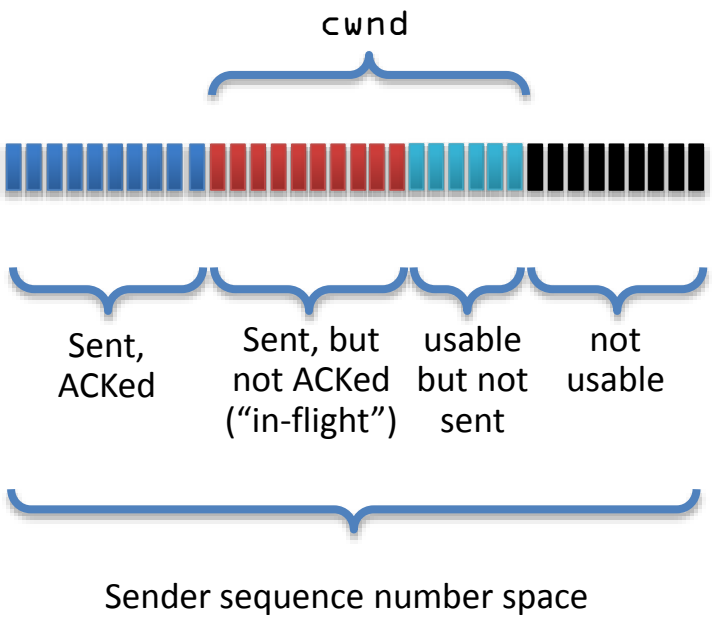
TCP Congestion Control

- Sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
  - *Additive increase:* increase `cwnd` by 1 every RTT until loss detected
  - *Multiplicative decrease*: cut `cwnd` in half after loss

AIMD saw tooth behavior: probing for bandwidth

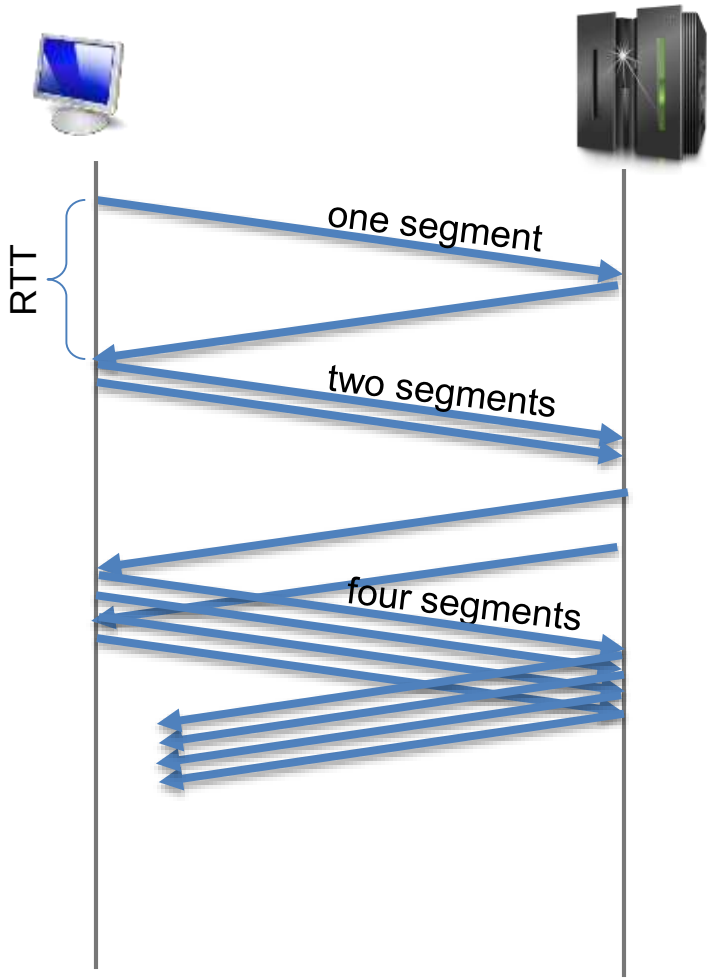


TCP Congestion Control



- Sender limits transmission
- `LastByteSent - LastByteAcked < cwnd`
- `cwnd` is dynamic, function of perceived network congestion
- *TCP sending rate:* send cwnd bytes, wait RTT for ACKS, then send more bytes (`cwnd`/RTT bytes/sec)

TCP Slow Start

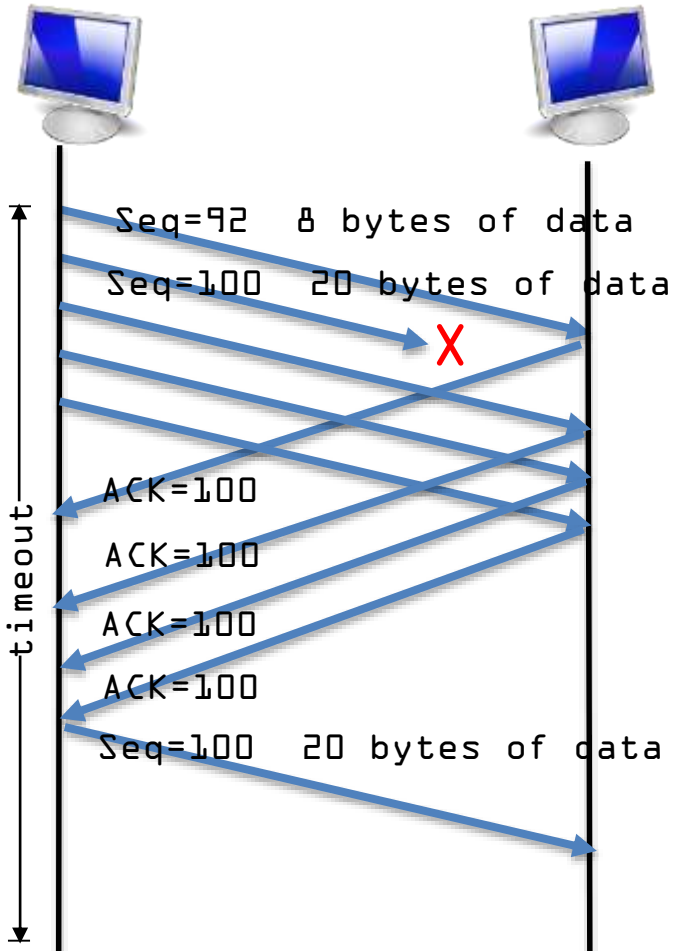


- When connection begins, increase rate exponentially until first loss event
  - Initially `cwnd` = 1
  - Double `cwnd` every RTT
  - Done by incrementing `cwnd` for every ACK received
- Basic idea: initial rate is slow but ramps up exponentially fast

Reaction to Loss

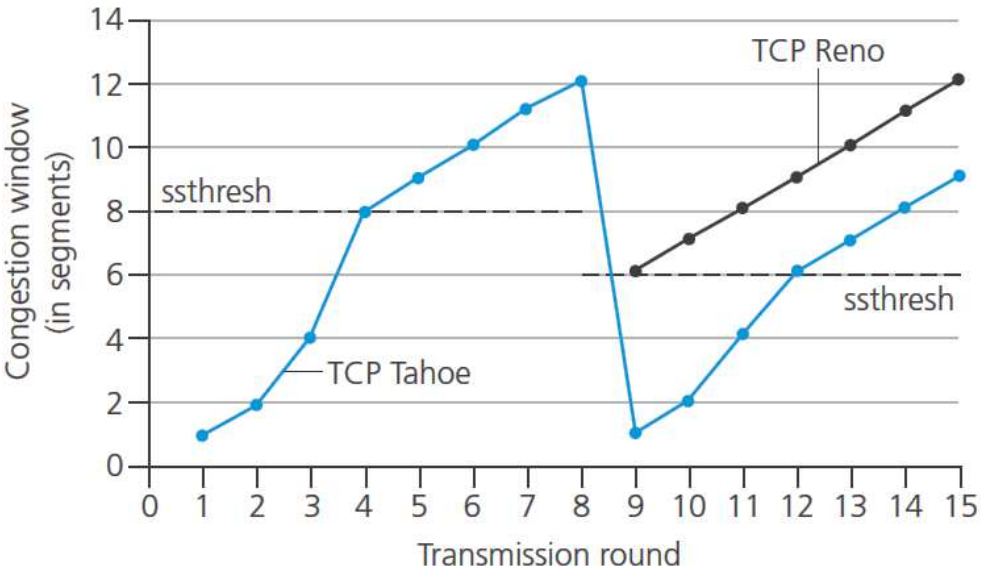
- Loss indicated by timeout:
  - `cwnd` set to 1
  - window then grows exponentially (as in slow start) to threshold, then grows linearly
- Loss indicated by 3 duplicate ACKs (TCP RENO)
  - Duplicate ACKs indicate network capable of delivering some segments
  - `cwnd` is cut in half window then grows linearly
- TCP Tahoe always sets `cwnd` to 1 (timeout or 3 duplicate acks)

TCP: Fast Retransmit



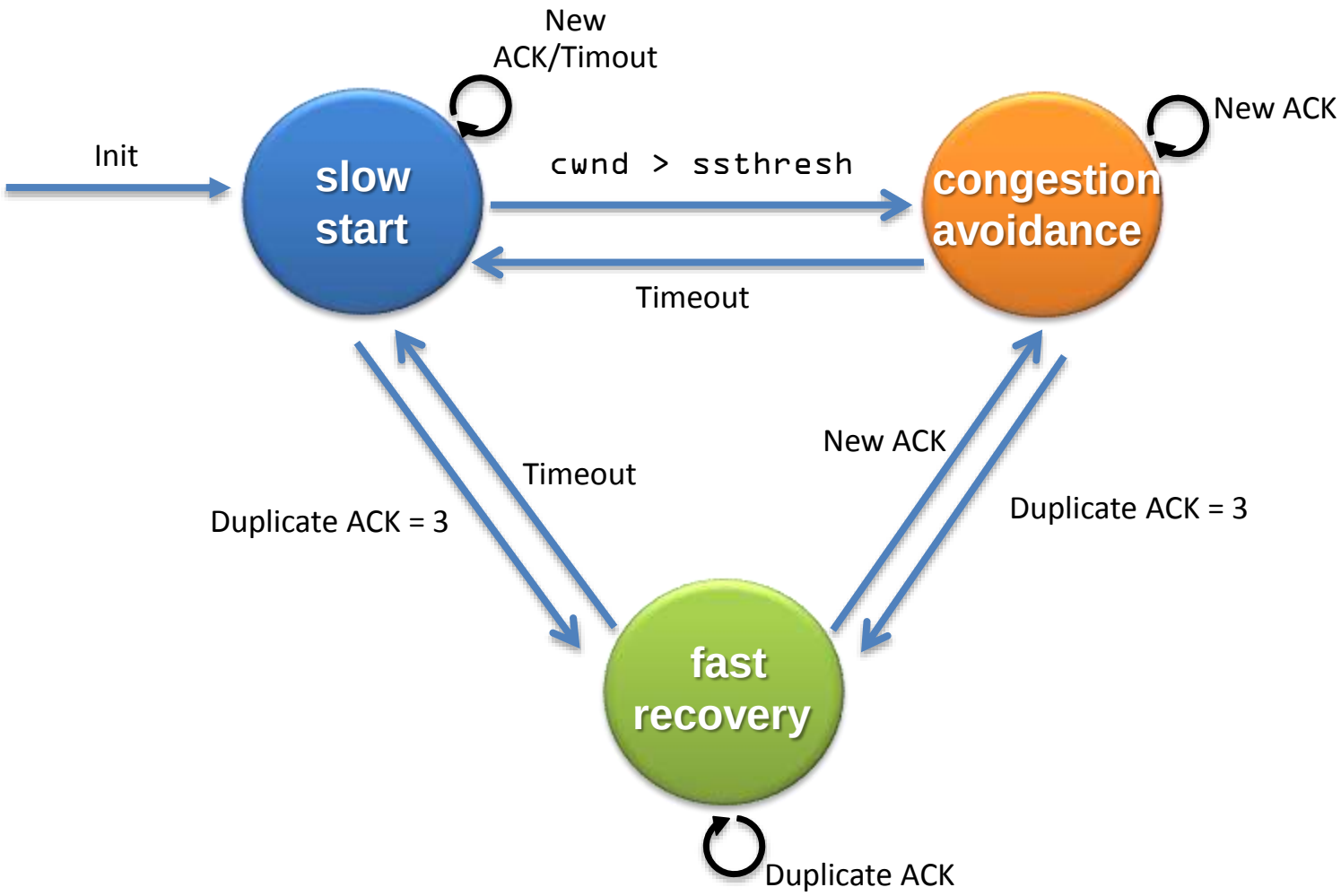
fast retransmit after sender receipt of triple duplicate ACK

Slow Start to Congestion Avoidance



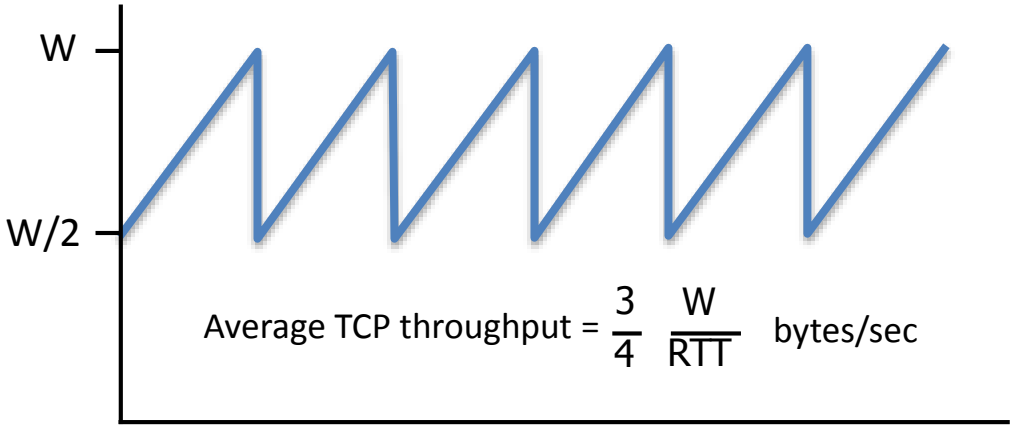
- Exponential increase switch to linear increase
  - when `cwnd` gets to 1/2 of its value before timeout
- Variable `ssthresh`
- On loss event, `ssthresh` is set to 1/2 of `cwnd` just before loss event

TCP Congestion Control: Simplified

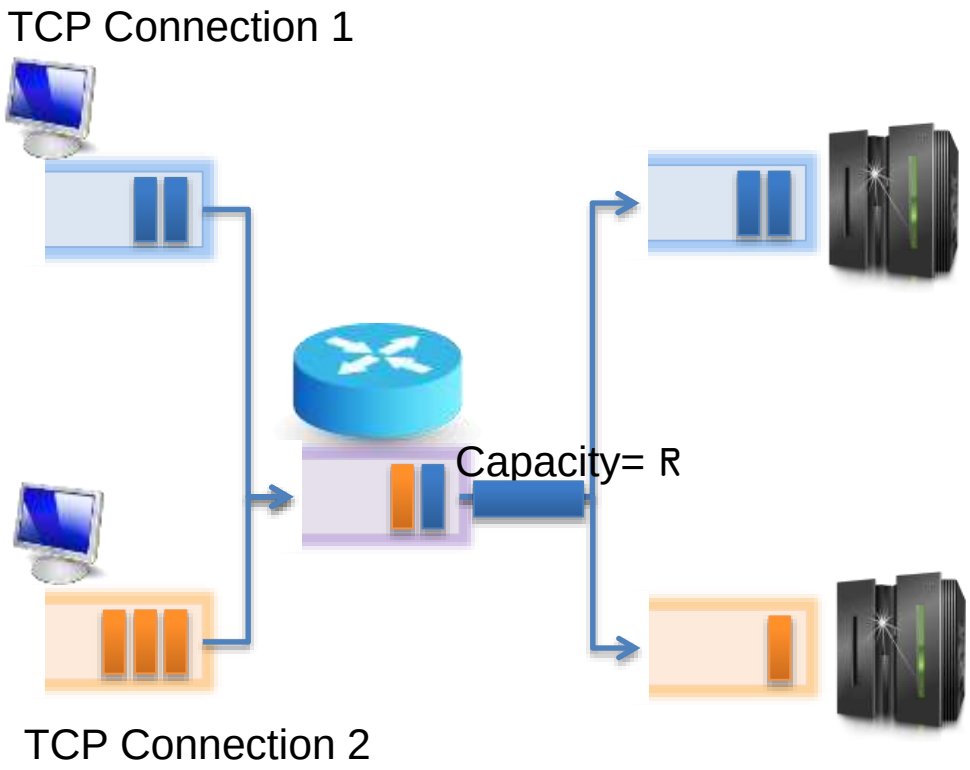


TCP Throughput

- Average TCP throughput as function of window size RTT
  - Ignore slow start, assume always data to send
- Window size W (measured in bytes) where loss occurs
  - Average window size (# in-flight bytes) is ¾ W
  - Average throughput is ¾ W per RTT

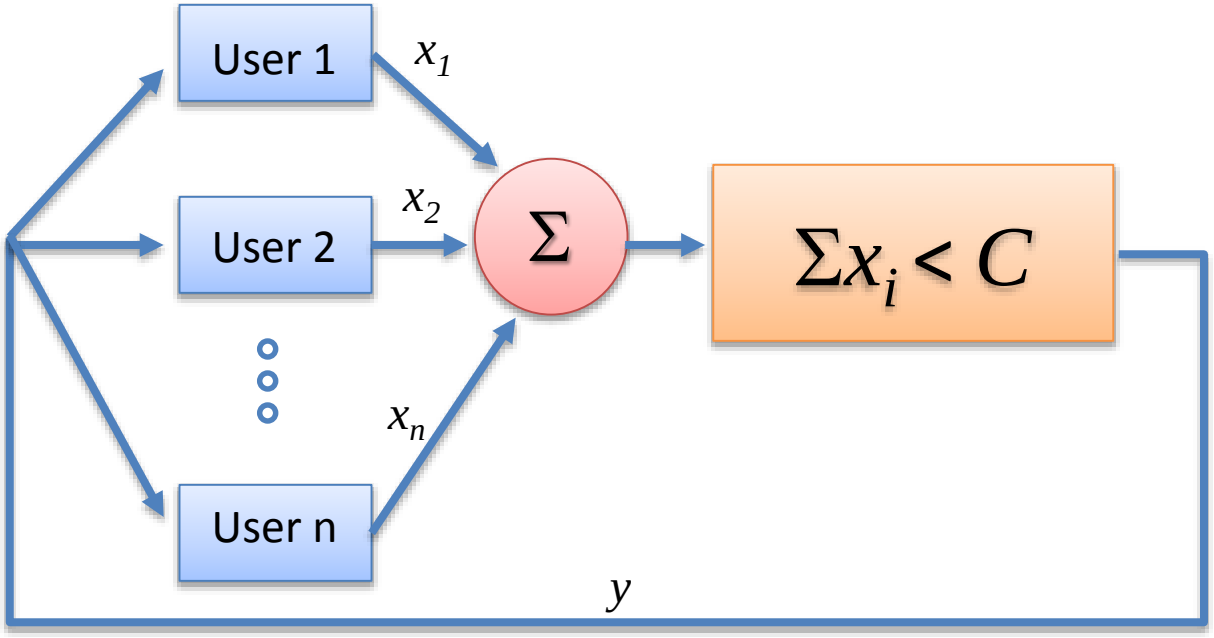


TCP Fairness



- *Fairness goal:*
  - If *K* TCP sessions share same bottleneck link of bandwidth R
  - Each should have average rate of R/*K*

System Control Model



- Simple, yet powerful model
- Explicit binary signal of congestion

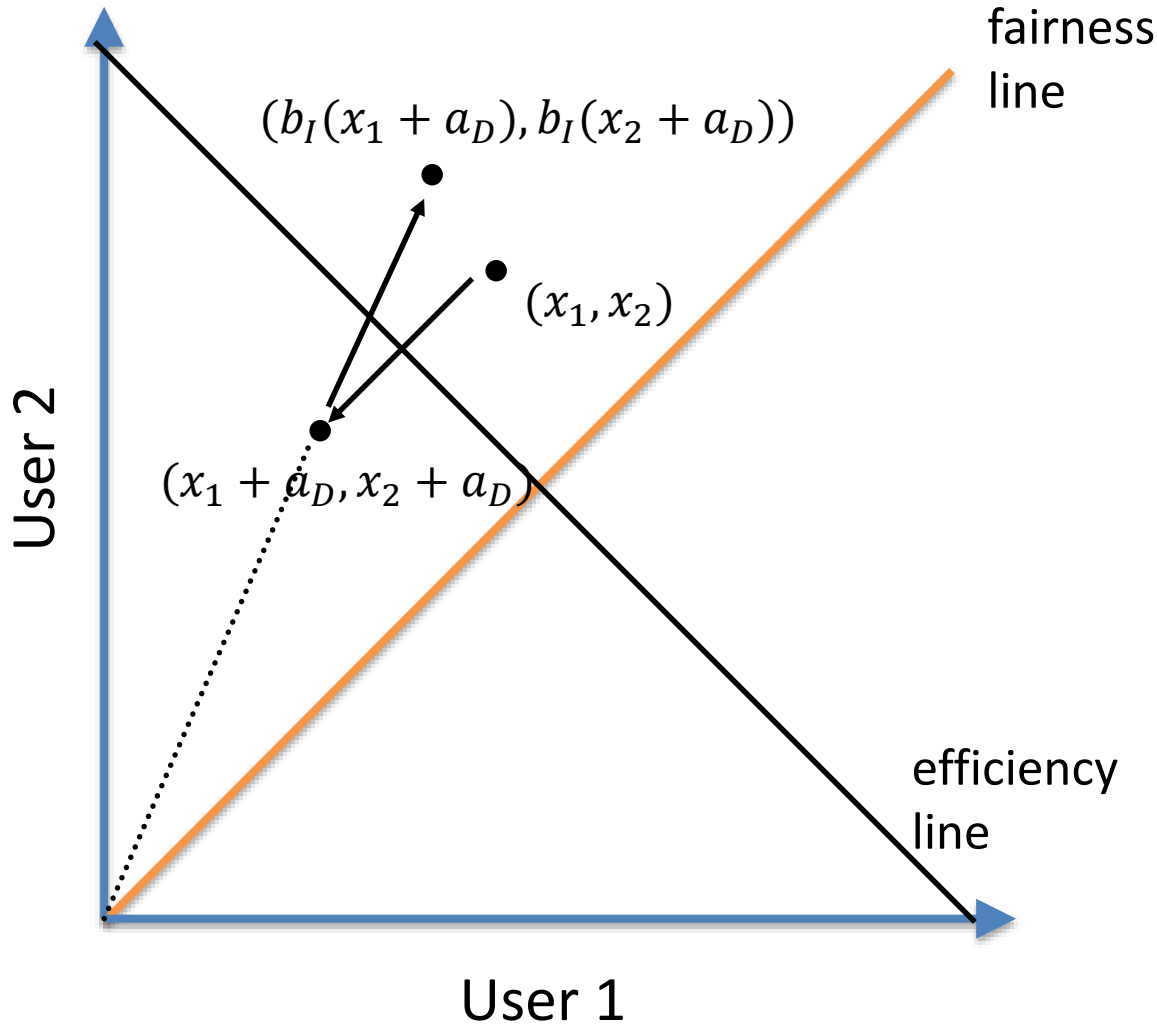
Possible Strategies

$$x_i(t+1) = \begin{cases} a_I + b_I x_i(t) & \text{Increase} \\ a_D + b_D x_i(t) & \text{Decrease} \end{cases}$$

- Multiplicative increase, additive decrease
  - $a_I = 0\,, b_I > 1\,, a_D < 0\,, b_D = 1$
- Additive increase, additive decrease
  - $a_I > 0\,, b_I = 1\,, a_D < 0\,, b_D = 1$
- Multiplicative increase, multiplicative decrease
  - $a_I = 0\,, b_I > 1\,, a_D = 0\,, b_D < 1$
- Additive increase, multiplicative decrease
  - $a_I > 0\,, b_I = 1\,, a_D = 0\,, b_D < 1$
- Which one?

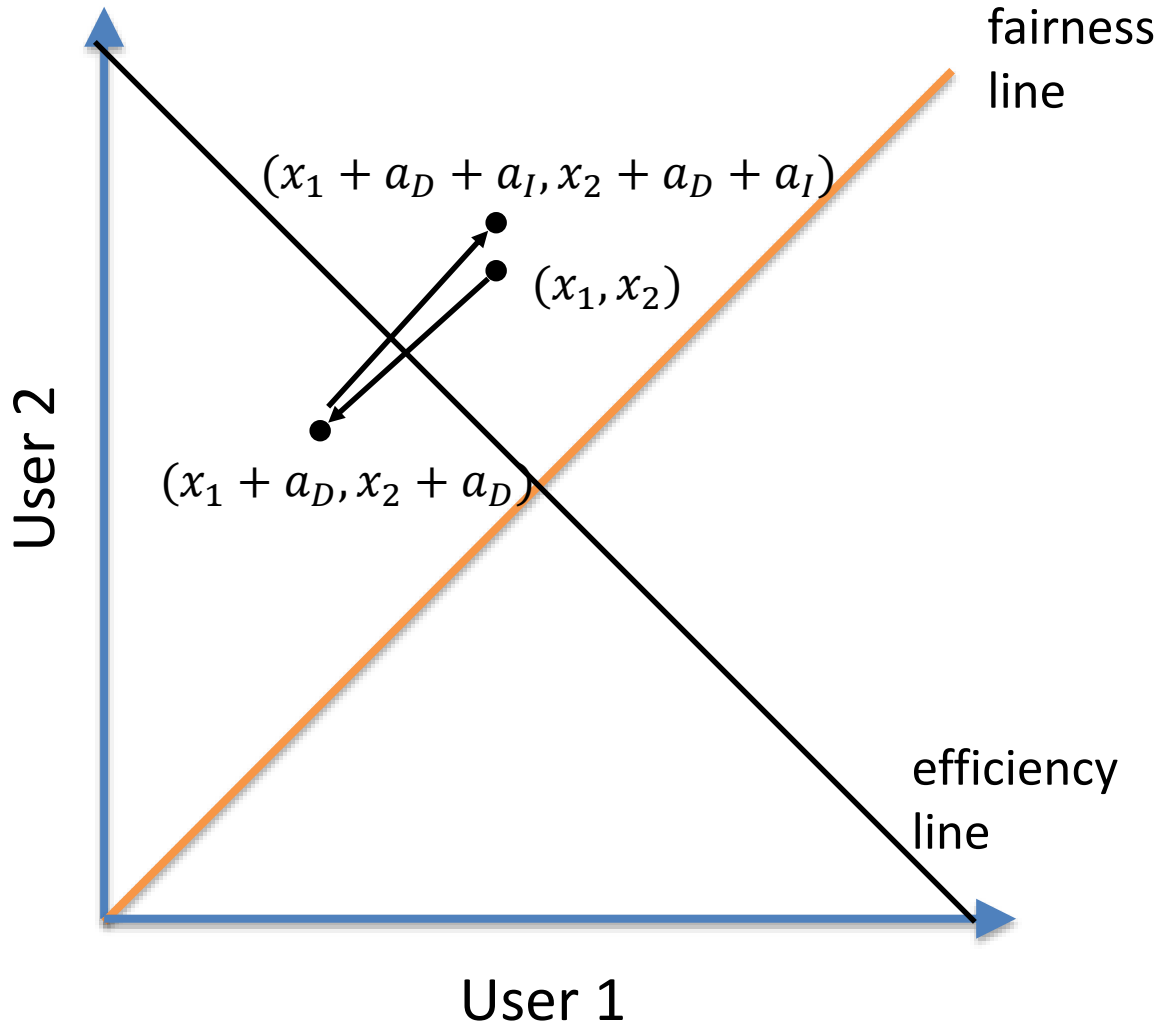
Multiplicative Increase, Additive Decrease

- Fixed point at $x_1 = x_2 = \frac{b_I a_D}{1 - b_I}$
- But fixed point is unstable



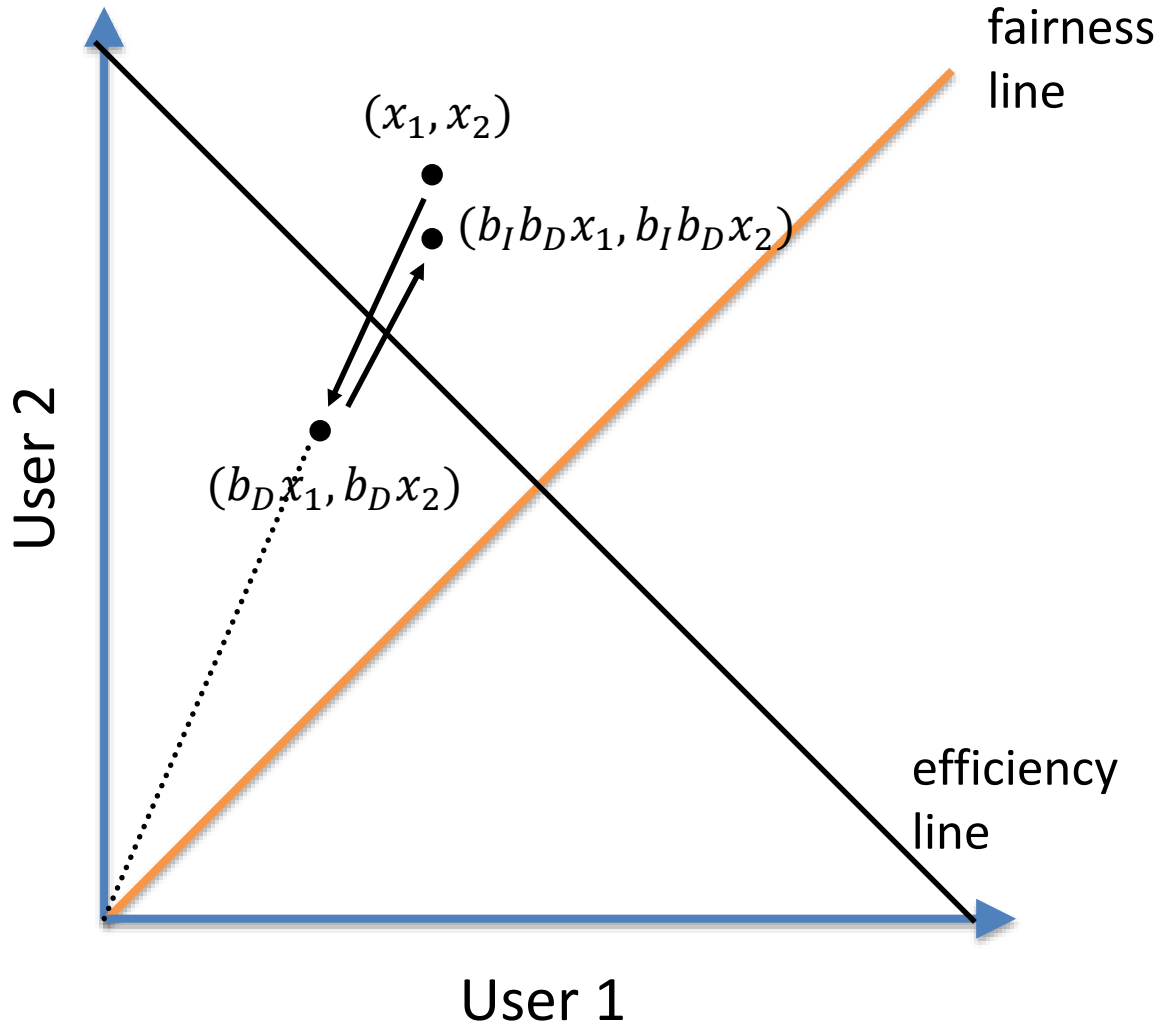
Additive Increase, Additive Decrease

- Reaches stable cycle, but does not converge to fairness



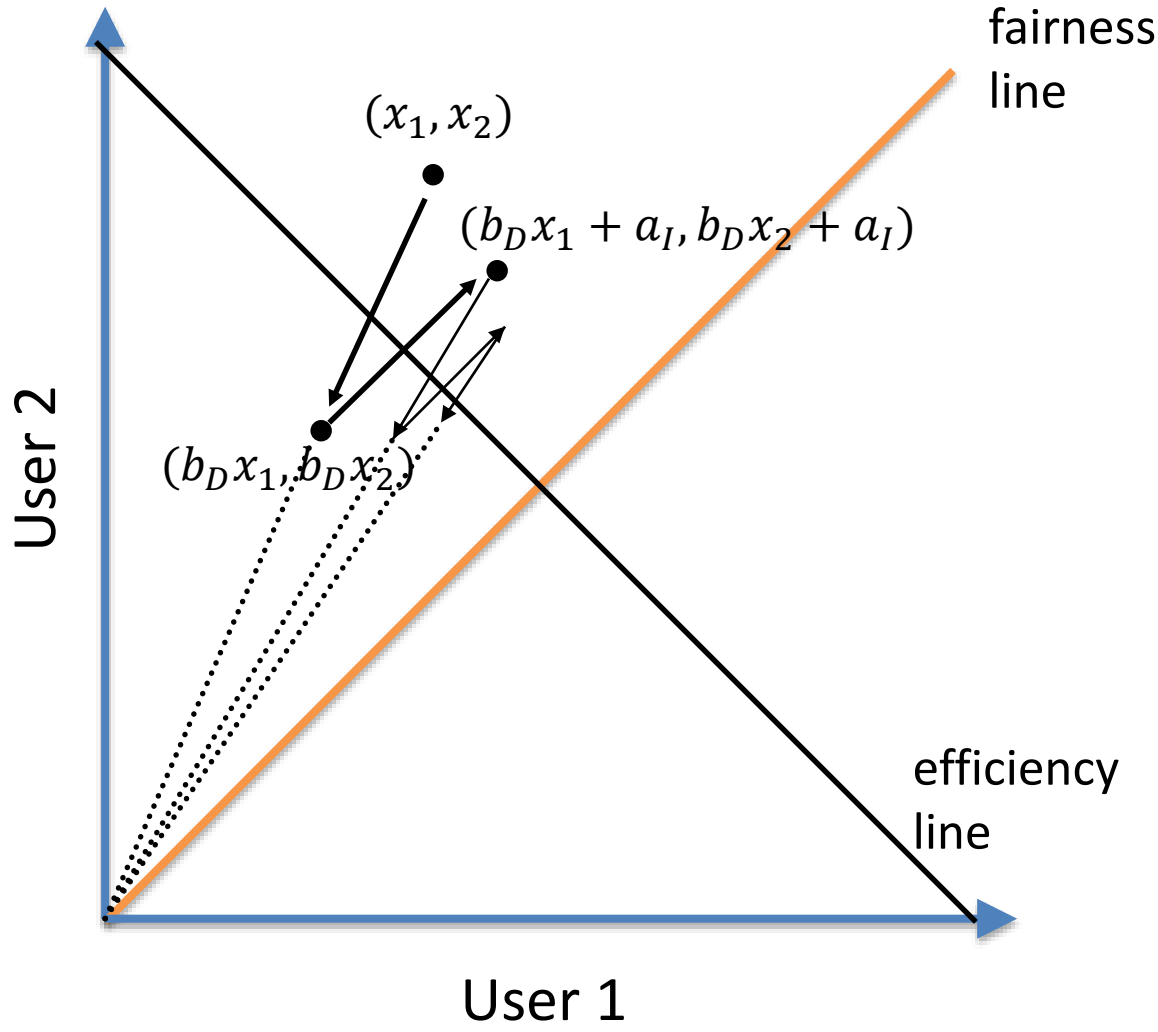
Multiplicative Increase, Multiplicative Decrease

- Converges to stable cycle, but is not fair



Additive Increase, Multiplicative Decrease

- Converges to stable and fair cycle



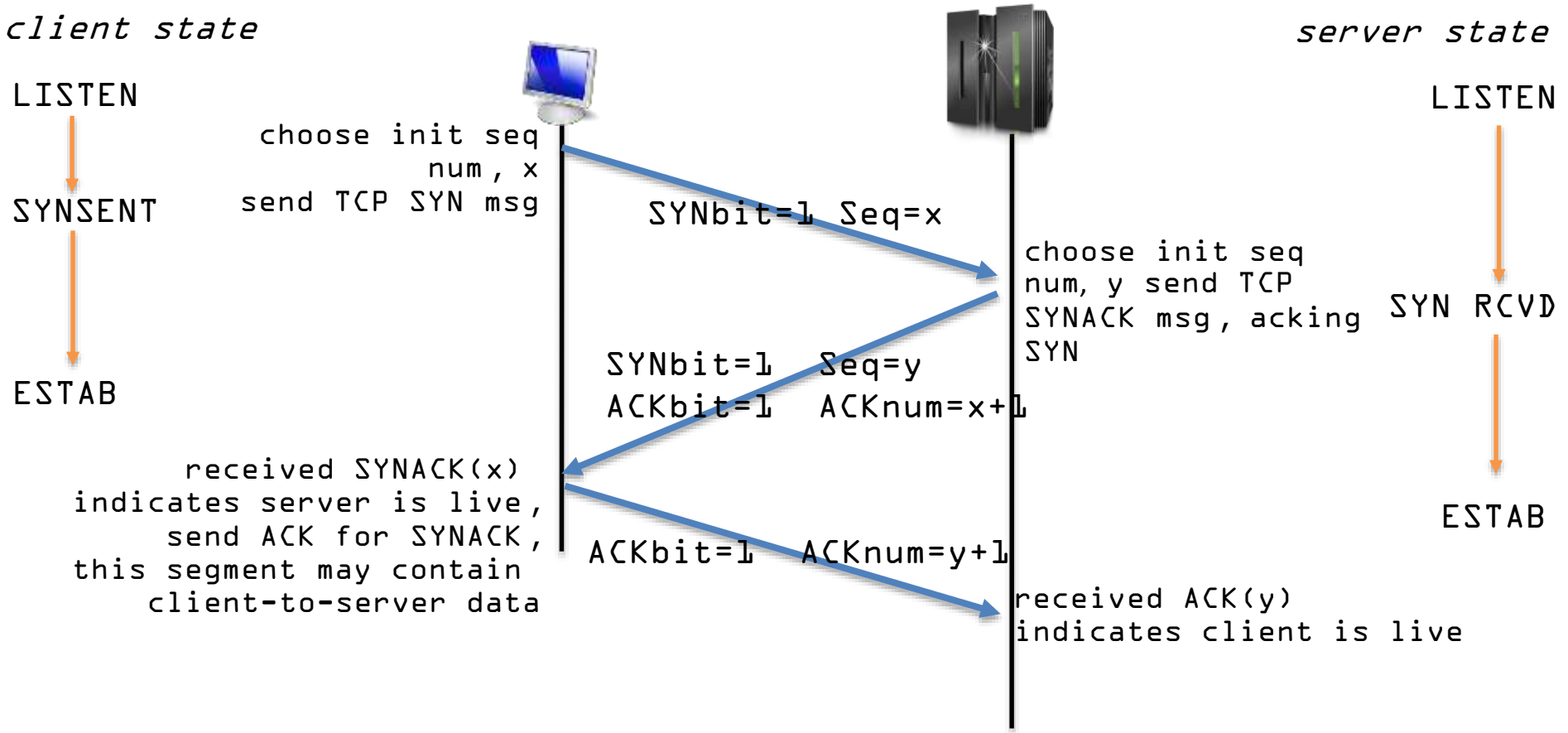
Extensions to TCP

- Selective acknowledgements: TCP SACK
- Explicit congestion notification: ECN
- Delay-based congestion avoidance: TCP Vegas
- Discriminating between congestion losses and other losses: cross-layer signaling and guesses
- Randomized drops (RED) and other router mechanisms
- Other Issues with TCP
  - Fairness: Throughput depends on RTT
  - High speeds: To reach 10gbps, packet losses occur every 90 minutes!
  - Selfish flows: Can get all the bandwidth they like

DDoS

- DDoS (distributed denial-of-service attack)
- When multiple systems flood the bandwidth or resources of a targeted system, usually one or more web servers
- These systems are compromised by attackers using a variety of methods
- Aim to reduce the availability of service
- Effects:
  - Loss of revenue
  - Server unable to preform timely operations (control system)
  - Prevent access to information (web server)

TCP: 3-way handshake



Example: SYN Flood

- When a client attempts to start a TCP connection to a server, the client and server exchange a series of messages which normally runs like this:
  - The client requests a connection by sending a SYN (synchronize) message to server
  - The server acknowledges sending SYN-ACK to client
  - The client responds with an ACK, and connection is established
- Attacker initiates a lot of SYN messages to a server
- Immediate abort (no closing connection message) after SYN
- Server left in time-out mode to detect aborted connections

References

- Textbook
  - Computer Networking (Chap. 3)
    James F. Kurose and Keith W. Ross; Pearson Addison-Wesley