CSCI-1680
Transport Layer III
Congestion Control Strikes Back

Chen Avin

Based partly on lecture notes by David Mazières, Phil Levis, John Jannotti, Peterson & Davie, Rodrigo Fonseca

and “Computer Networking: A Top Down Approach” - 6th edition

Last Time

- Flow Control
- Congestion Control

Today

- **More TCP Fun!**
- **TCP Throughput**
- **TCP fairness**
- **TCP on Lossy Links**
- **Congestion Control versus Avoidance**
  - Getting help from the network
- **Cheating TCP**

TCP Throughput

- **Assume a TCP congestion of window W (segments), round-trip time of RTT, segment size MSS**
  - Sending Rate $S = W \times MSS / RTT$ (1)
- **Drop: W = W/2**
  - grows by MSS for $W/2$ $RTTs$, until another drop at $W \approx W$
- **Average window then 0.75xS**
  - From (1), $S = 0.75\ W\ MSS / RTT$ (2)
- **Loss rate is 1 in number of packets between losses:**
  - $Loss = 1 / (W/2 + W/2+1 + W/2 + 2 + \ldots + W)$
    $= 1 / (3/8\ W^2)$ (3)

TCP Throughput (cont)

– Loss = $8/(3W^2)$ $\Rightarrow W = \sqrt{\dfrac{8}{3 \cdot Loss}}$ (4)

– Substituting (4) in (2), *S* = 0.75 *W MSS* / *RTT* ,

Throughput ≈ $1.22 \times \dfrac{MSS}{RTT \cdot \sqrt{Loss}}$

TCP Futures: TCP over "long, fat pipes"

- **example: 1500 byte segments, 100ms RTT, want 10 Gbps throughput**
- **requires W = 83,333 in-flight segments**
- **throughput in terms of segment loss probability, L [Mathis 1997]:**

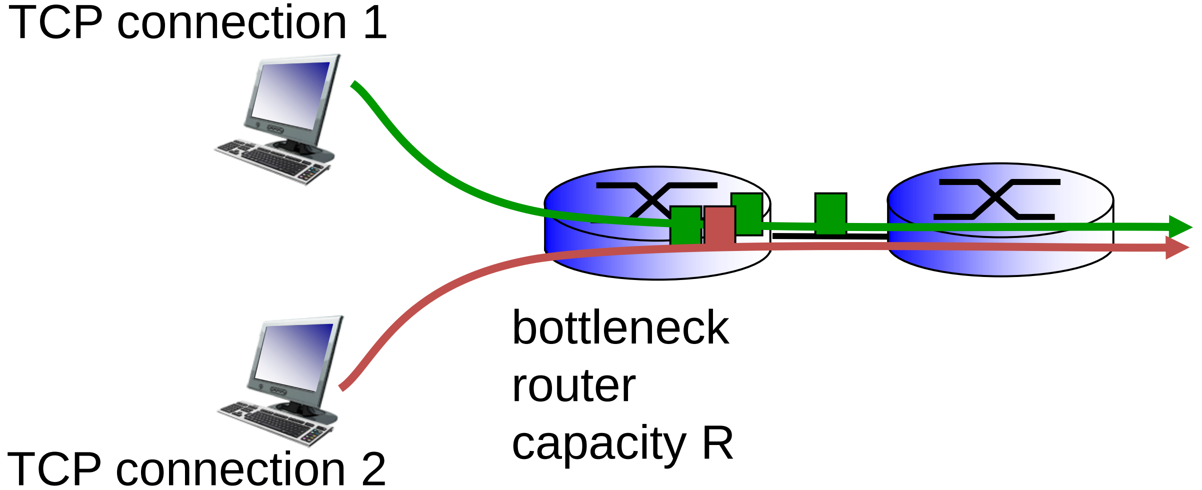
$$\text{TCP throughput} = \frac{1.22 \cdot \text{MSS}}{\text{RTT}\sqrt{L}}$$

➜ to achieve 10 Gbps throughput, need a loss rate of $L = 2 \cdot 10^{-10}$ *– a very small loss rate!*

- **new versions of TCP for high-speed**

TCP Fairness

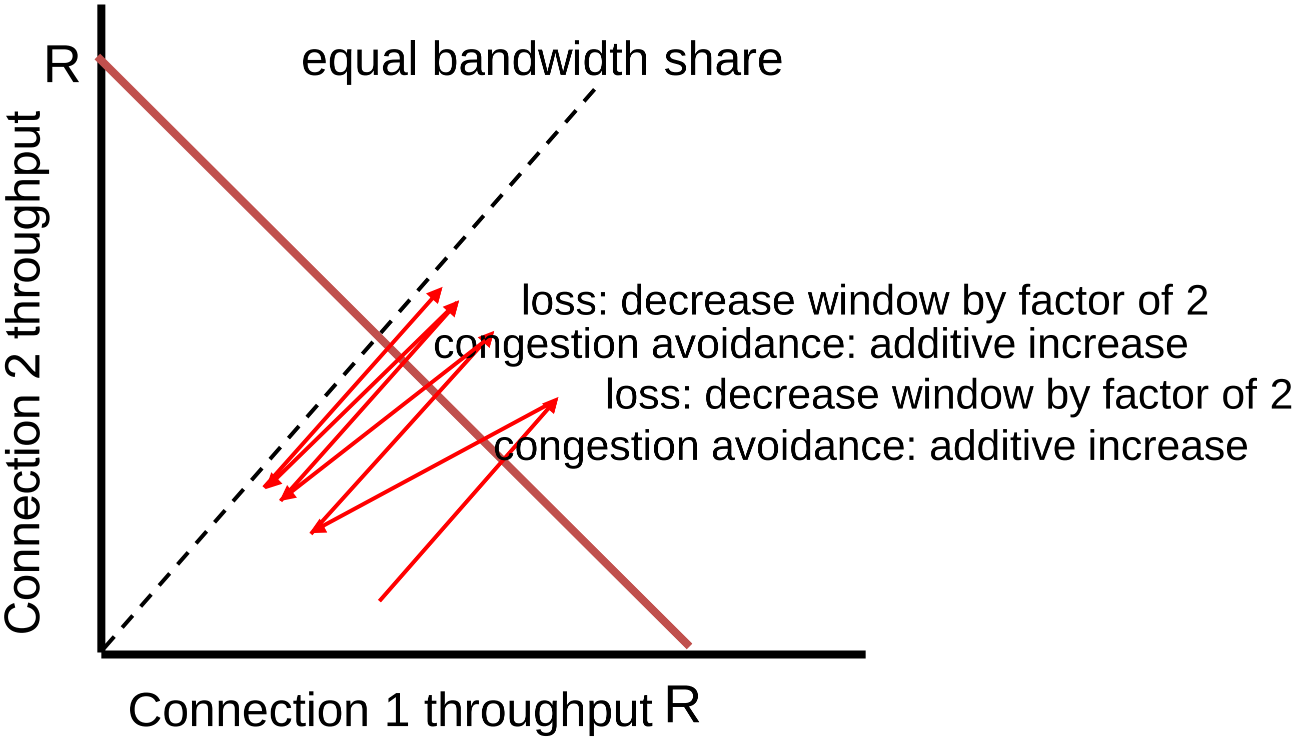
fairness goal: **if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K**



Why is TCP fair?

two competing sessions:

- **additive increase gives slope of 1, as throughout increases**
- **multiplicative decrease decreases throughput proportionally**



Fairness (more)

Fairness and UDP

- **multimedia apps often do not use TCP**
  - do not want rate throttled by congestion control
- **instead use UDP:**
  - send audio/video at constant rate, tolerate packet loss

Fairness, parallel TCP connections

- **application can open multiple parallel connections between two hosts**
- **web browsers do this**
- **e.g., link of rate R with 9 existing connections:**
  - new app asks for 1 TCP, gets rate R/10
  - new app asks for 11 TCPs, gets R/2

TCP Friendliness

- **Can other protocols co-exist with TCP?**
  - E.g., if you want to write a video streaming app using UDP, how to do congestion control?
- **Equation-based Congestion Control**
  - Instead of implementing TCP's CC, estimate the rate at which TCP would send. Function of what?
  - RTT, MSS, Loss
- **Measure RTT, Loss, send at that rate!**

Approaches towards congestion control

Two broad approaches towards congestion control:

End-end congestion control:

- no explicit feedback from network
- congestion inferred from end-system observed loss, delay
- approach taken by TCP

Network-assisted congestion control:

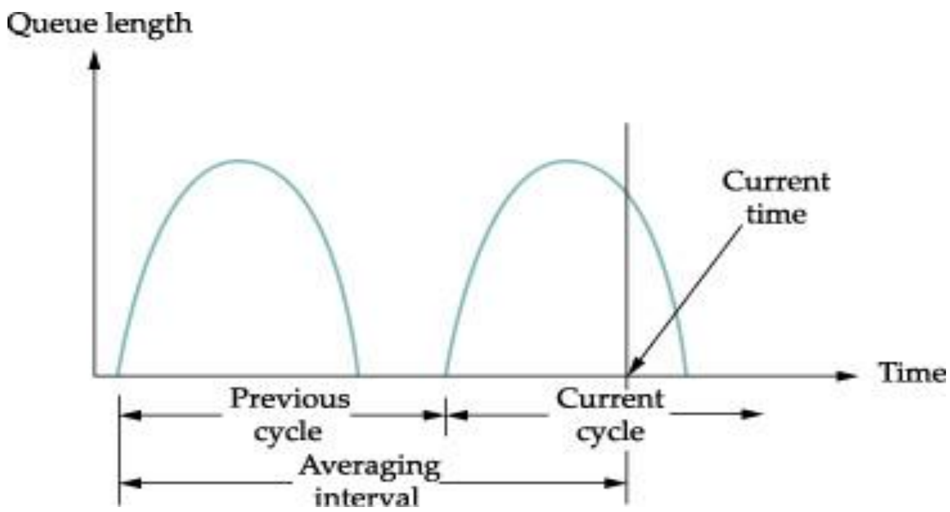
- routers provide feedback to end systems
- single bit indicating congestion (SNA, DECbit, TCP/IP ECN, ATM)
- explicit rate sender should send at

Congestion Avoidance

- **TCP creates congestion to then back off**
  – Queues at bottleneck link are often full: increased delay
  – Sawtooth pattern: jitter
- **Network-assisted congestion control:**
  – Predict when congestion is about to happen
  – Reduce rate before packets start being discarded
  – Call this congestion avoidance instead of congestion control
- **Two approaches**
  – router-centric: e.g., DECbit and RED gateways
  – host-centric: e.g., TCP vegas

DECbit

- **Add binary congestion bit to each packet header**
- **Router:**
  - monitors average queue length over last busy_idle cycle



  - set congestion bit if average queue length > 1
  - attempts to balance throughput vs. delay

End Hosts

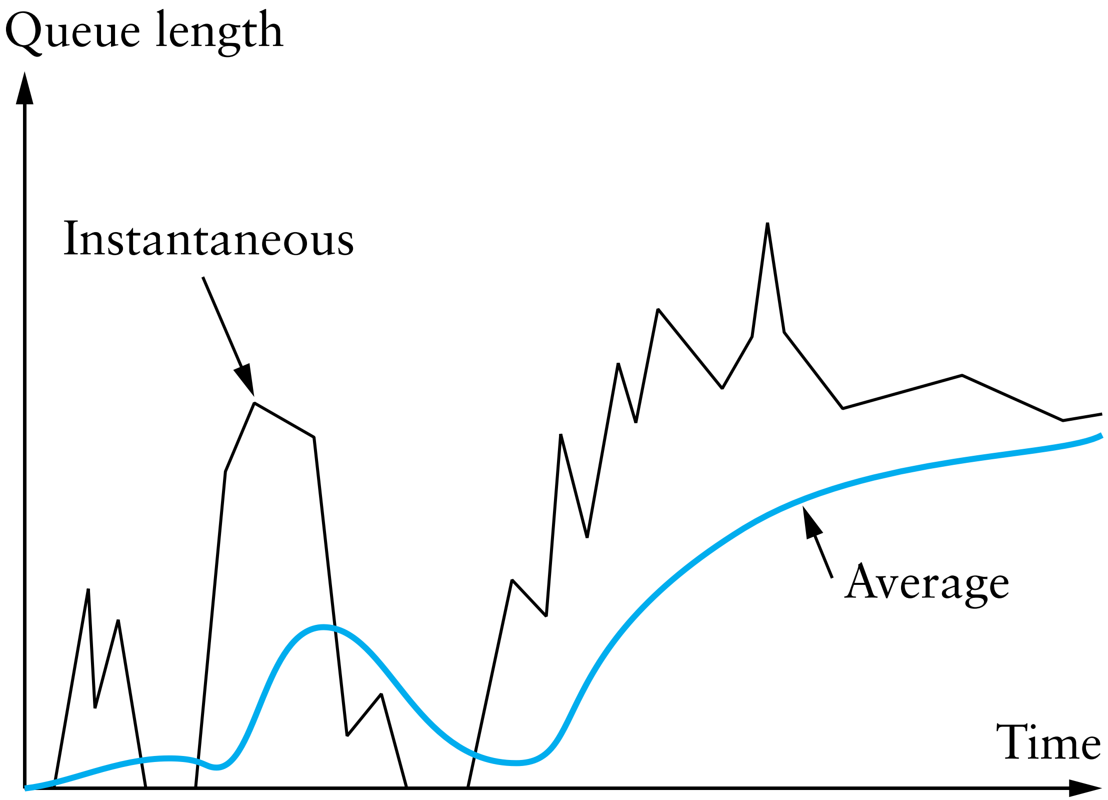
- **Destination echoes bit back to source**
- **Source records how many packets results in set bit.**
- **If less than 50% of last window's worth had bit set**
  - increase `CongWin` by 1 packet
- **If more than 50% of last window's worth had bit set**
  - decrease `CongWin` by 0.875 times

Random Early Detection (RED)

- **Notification is implicit**
  - Just drop the packet (TCP will timeout or dup ACKs)
  - Could make explicit by marking the packet (ECN)
- **Early random drop**
  - Rather than wait for queue to become full, drop each arriving packet with some drop probability whenever the queue length exceeds some drop level.

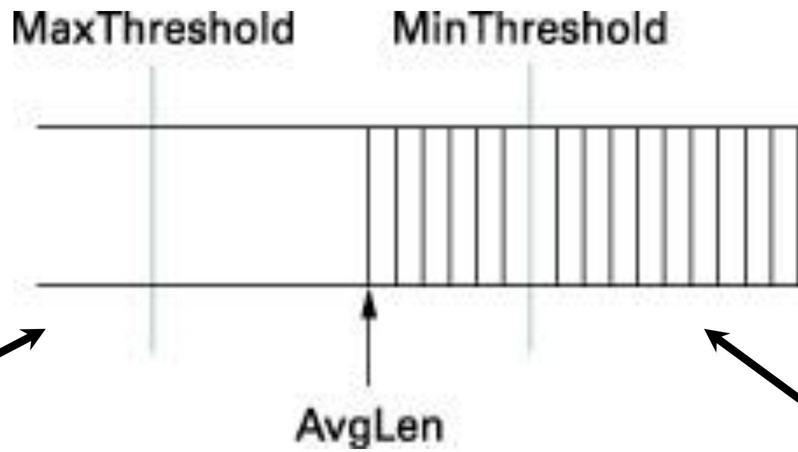
RED Details

- **Compute average queue length (EWMA)**
  - Don't want to react to very quick fluctuations



RED details (cont)

- **Two queue length thresholds**



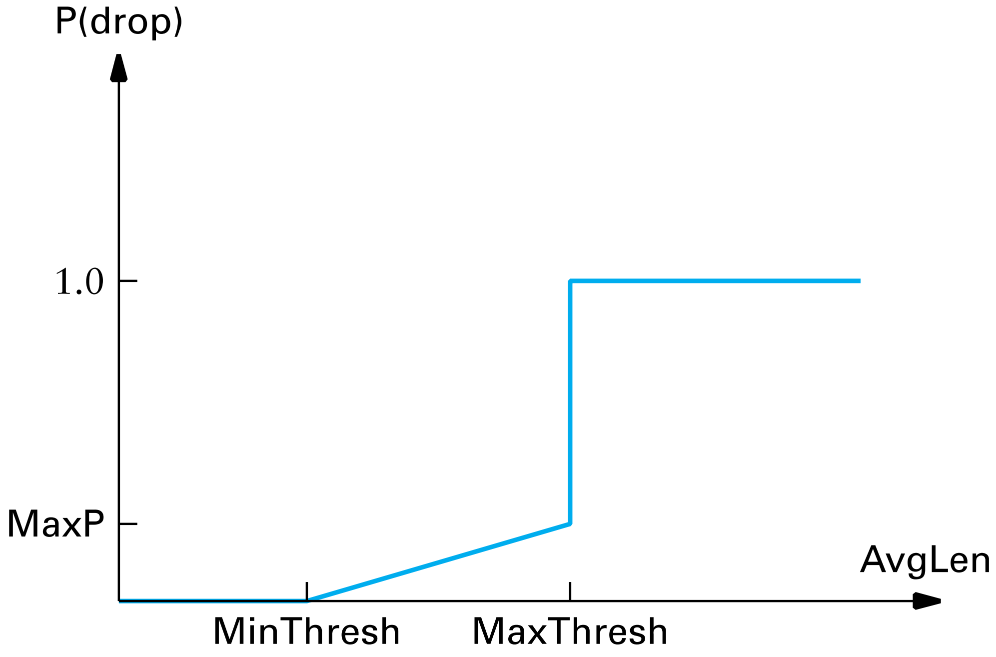
Drop packet

Keep packet

Calculate probability p and drop packet
with probability p

RED Drop Probability

- **Define two thresholds: MinThresh, MaxThresh**
- **Drop probability:**



- **Improvements to spread drops (see book)**

TCP Vegas: Host based CA

- **Idea: Source watches for some sign that router's queue is building up and congestion happen too, for example:**
  - RTT grows
  - Sending rate flatten
- **"Fast TCP"**
  - base RTT (on "empty" network, minimum measured)
  - observed RTT
  - Difference is used to estimate queues lengths

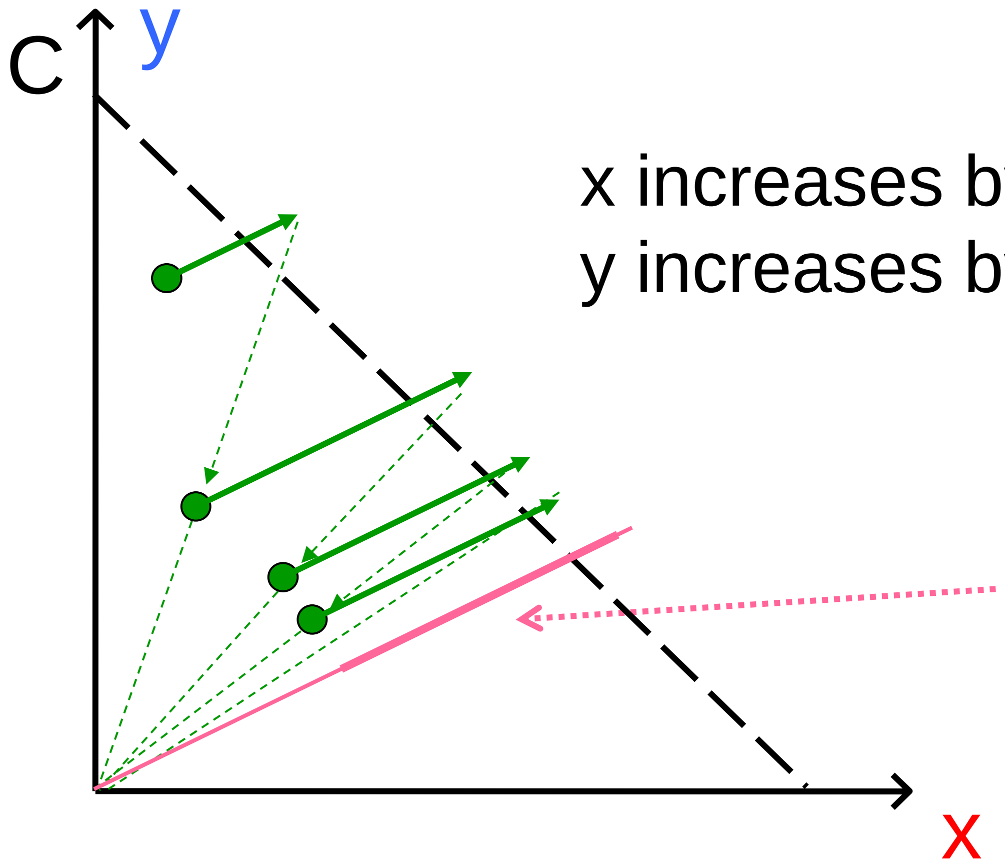
What happens if not everyone cooperates?

- **TCP works extremely well when its assumptions are valid**
  - All flows correctly implement congestion control
  - Losses are due to congestion

Cheating TCP

- **Three possible ways to cheat**
  - Increasing cwnd faster
  - Large initial cwnd
  - Opening many connections
  - Ack Division Attack

Increasing cwnd Faster

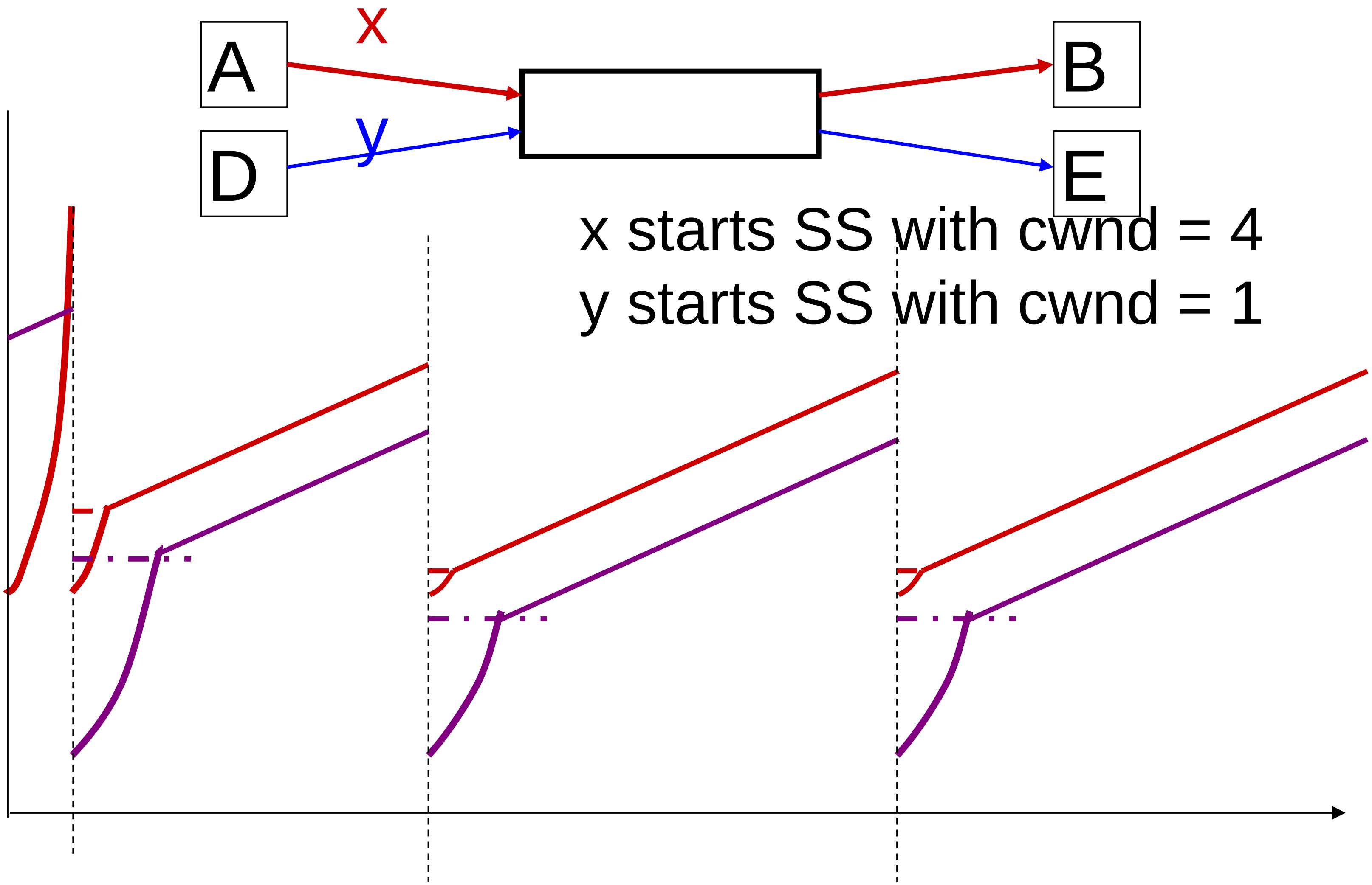


x increases by 2 per RTT
y increases by 1 per RTT

Limit rates:
x = 2y

Figure from Walrand, Berkeley EECS 122, 2003

Larger Initial Window



x starts SS with cwnd = 4

y starts SS with cwnd = 1

Figure from Walrand, Berkeley EECS 122, 2003

Open Many Connections

- **Web Browser: has to download k objects for a page**
  - Open many connections or download sequentially?

- **Assume:**
  - A opens 10 connections to B
  - B opens 1 connection to E
- **TCP is fair among connections**
  - A gets 10 times more bandwidth than B

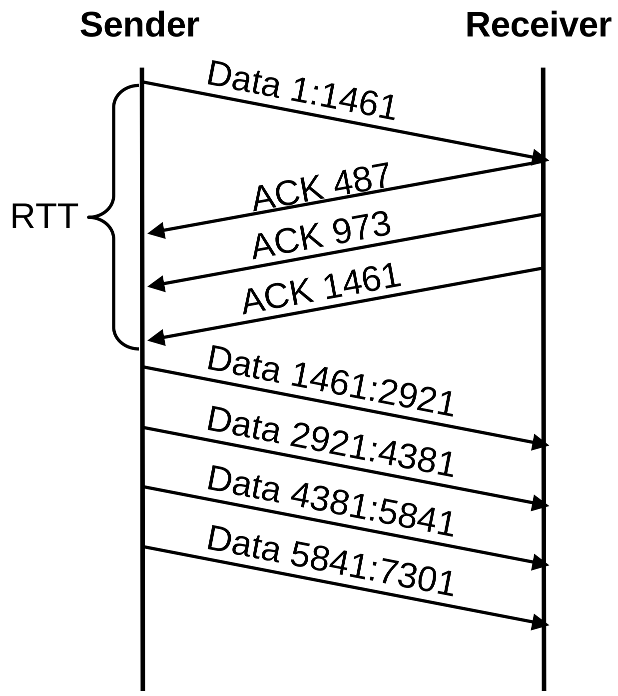
Figure from Walrand, Berkeley EECS 122, 2003

Exploiting Implicit Assumptions

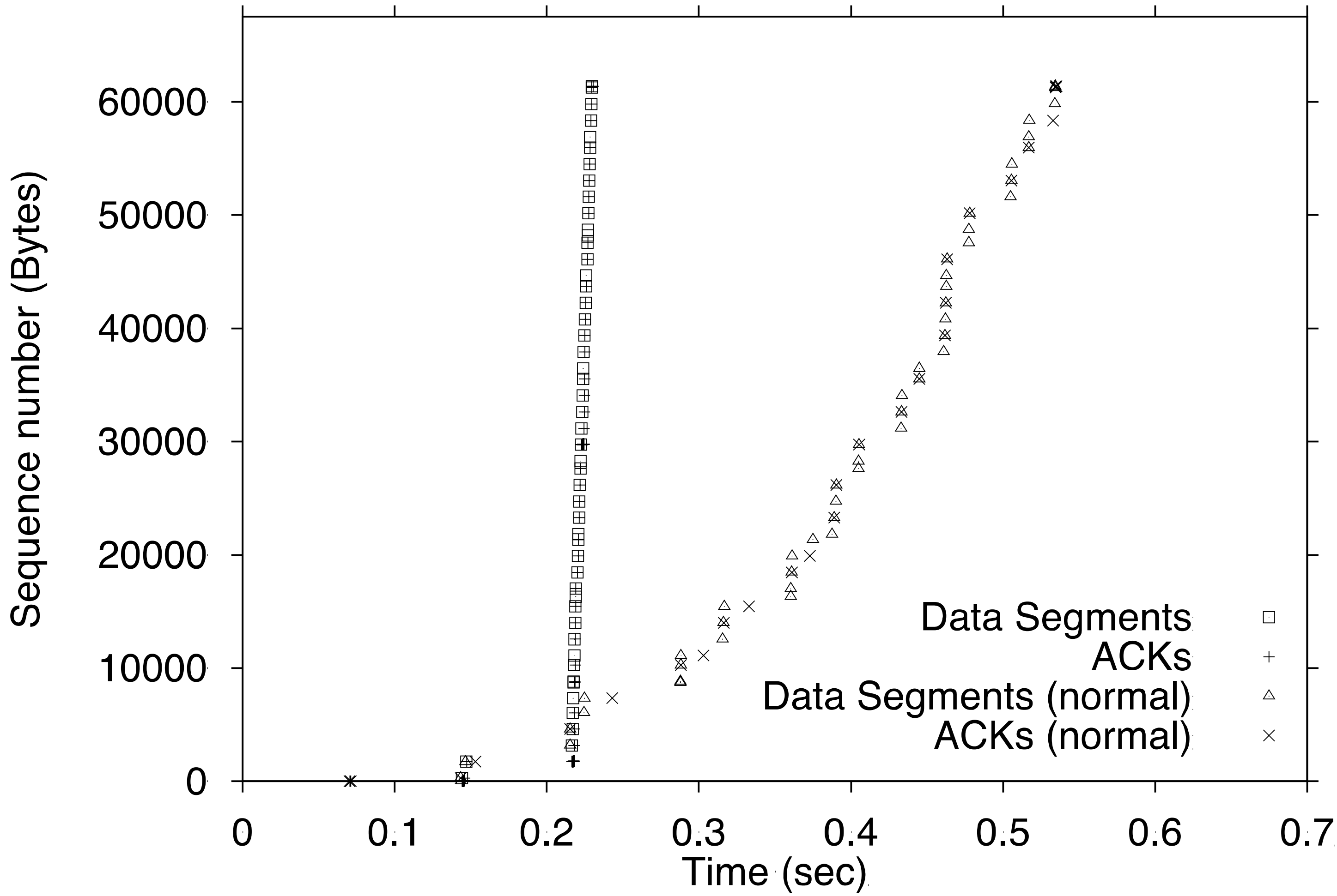
- **Savage, et al., CCR 1999:**
  - "TCP Congestion Control with a Misbehaving Receiver"
- **Exploits ambiguity in meaning of ACK**
  - ACKs can specify any byte range for error control
  - Congestion control assumes ACKs cover entire sent segments
- **What if you send multiple ACKs per segment?**

ACK Division Attack

- **Receiver: "***upon receiving a segment with N bytes, divide the bytes in M groups and acknowledge each group separately"*
- **Sender will grow window M times faster**
- **Could cause growth to 4GB in 4 RTTs!**
  - M = N = 1460



TCP Daytona!



Sequence number (Bytes)
60000
50000
40000
30000
20000
10000
0
0
0.1
0.2
0.3
0.4
0.5
0.6
0.7
Time (sec)
Data Segments
ACKs
Data Segments (normal)
ACKs (normal)

Defense

- **Appropriate Byte Counting**
  - [RFC3465 (2003), RFC 5681 (2009)]
  - In slow start, cwnd += min (N, MSS)

  where N is the number of newly acknowledged bytes in the received ACK

Next Time

- Move into the application layer
- DNS, Web, Security, and more…