شبکه های کامپیوتری ۲

جلسه ۱۱ فصل ۳

Congestion Control 2

دانشگاه صنعتی اصفهان
دانشکده مهندسی برق و کامپیوتر

Chapter 3
Transport Layer

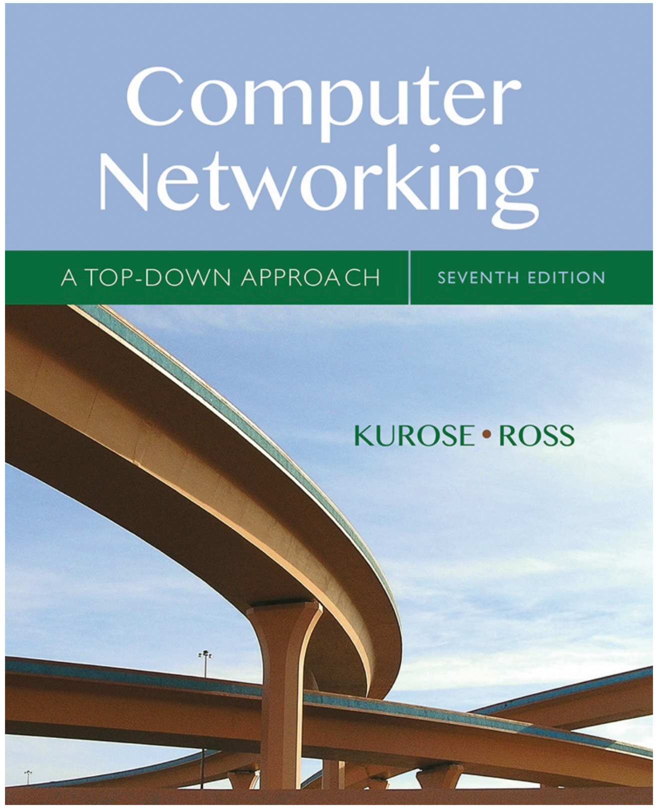
A note on the use of these Powerpoint slides:

We're making these slides freely available to all (faculty, students, readers). They're in PowerPoint form so you see the animations; and can add, modify, and delete slides (including this one) and slide content to suit your needs. They obviously represent a *lot* of work on our part. In return for use, we only ask the following:

- If you use these slides (e.g., in a class) that you mention their source (after all, we'd like people to use our book!)
- If you post any slides on a www site, that you note that they are adapted from (or perhaps identical to) our slides, and note our copyright of this material.

Thanks and enjoy! JFK/KWR

© All material copyright 1996-2016
J.F Kurose and K.W. Ross, All Rights Reserved



Computer Networking: A Top Down Approach

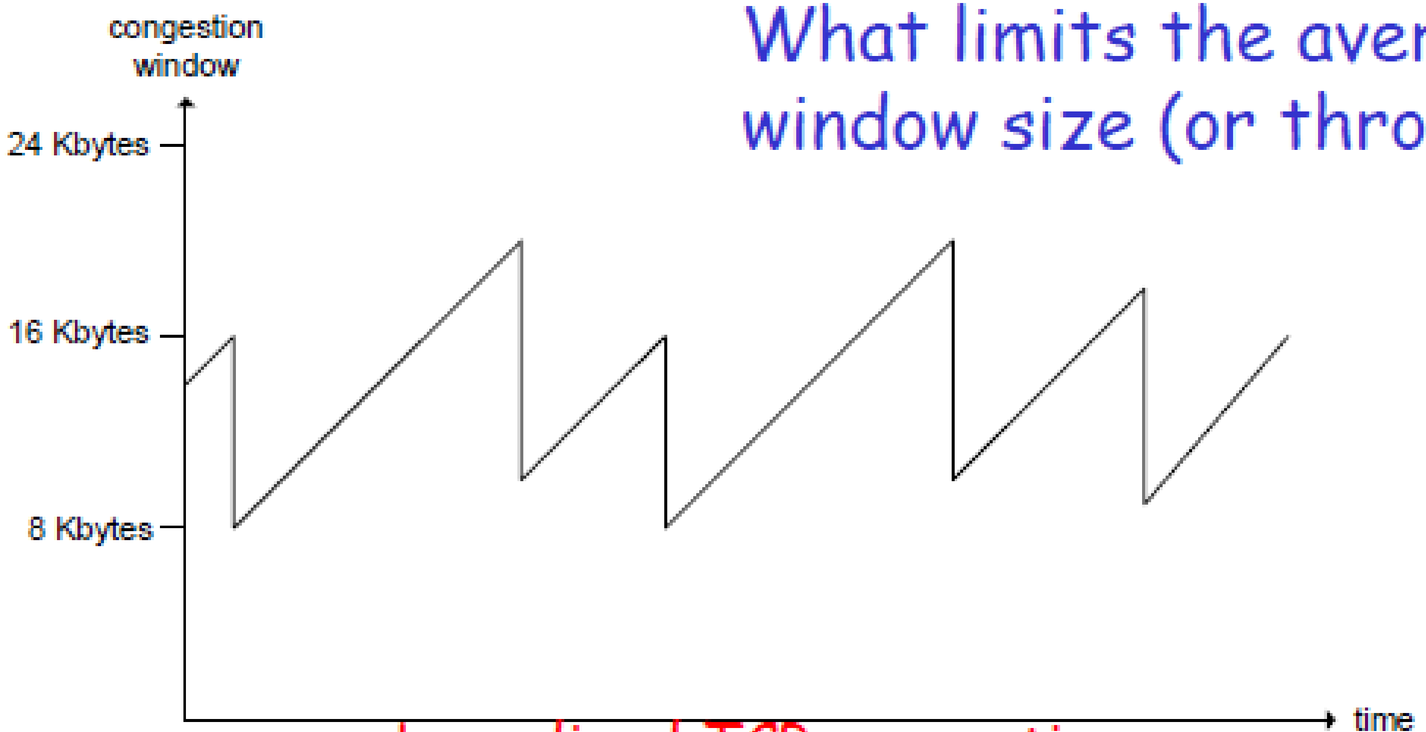
7th edition
Jim Kurose, Keith Ross
Pearson/Addison Wesley
April 2016

AIMD in steady state (when no timeout)

- additive increase:

  increase **cwnd** by 1 MSS every RTT in the absence of any loss event: probing

- multiplicative decrease:
- cut **cwnd** in half after loss event (3 dup acks)



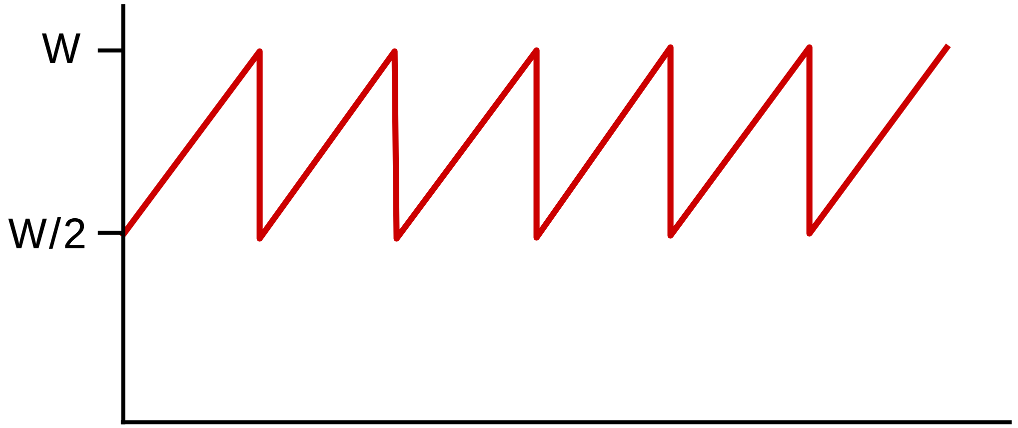
What limits the average window size (or throughput)?

Long-lived TCP connection

TCP throughput

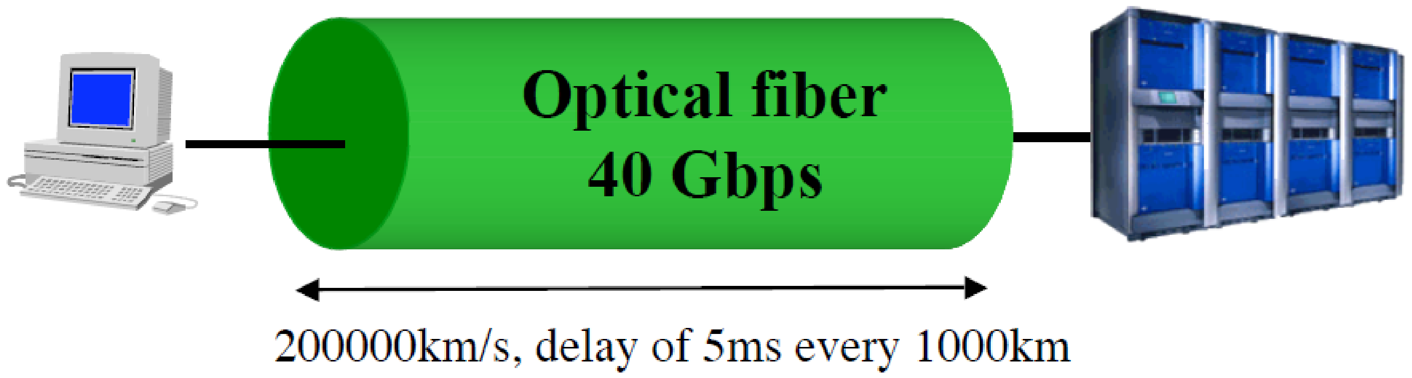
- avg. TCP thruput as function of window size, RTT?
  - ignore slow start, assume always data to send
- W: window size (measured in bytes) where loss occurs
  - avg. window size (# in-flight bytes) is ¾ W
  - avg. thruput is 3/4W per RTT

$$\text{avg TCP thruput} = \frac{3}{4}\frac{W}{RTT} \text{ bytes/sec}$$

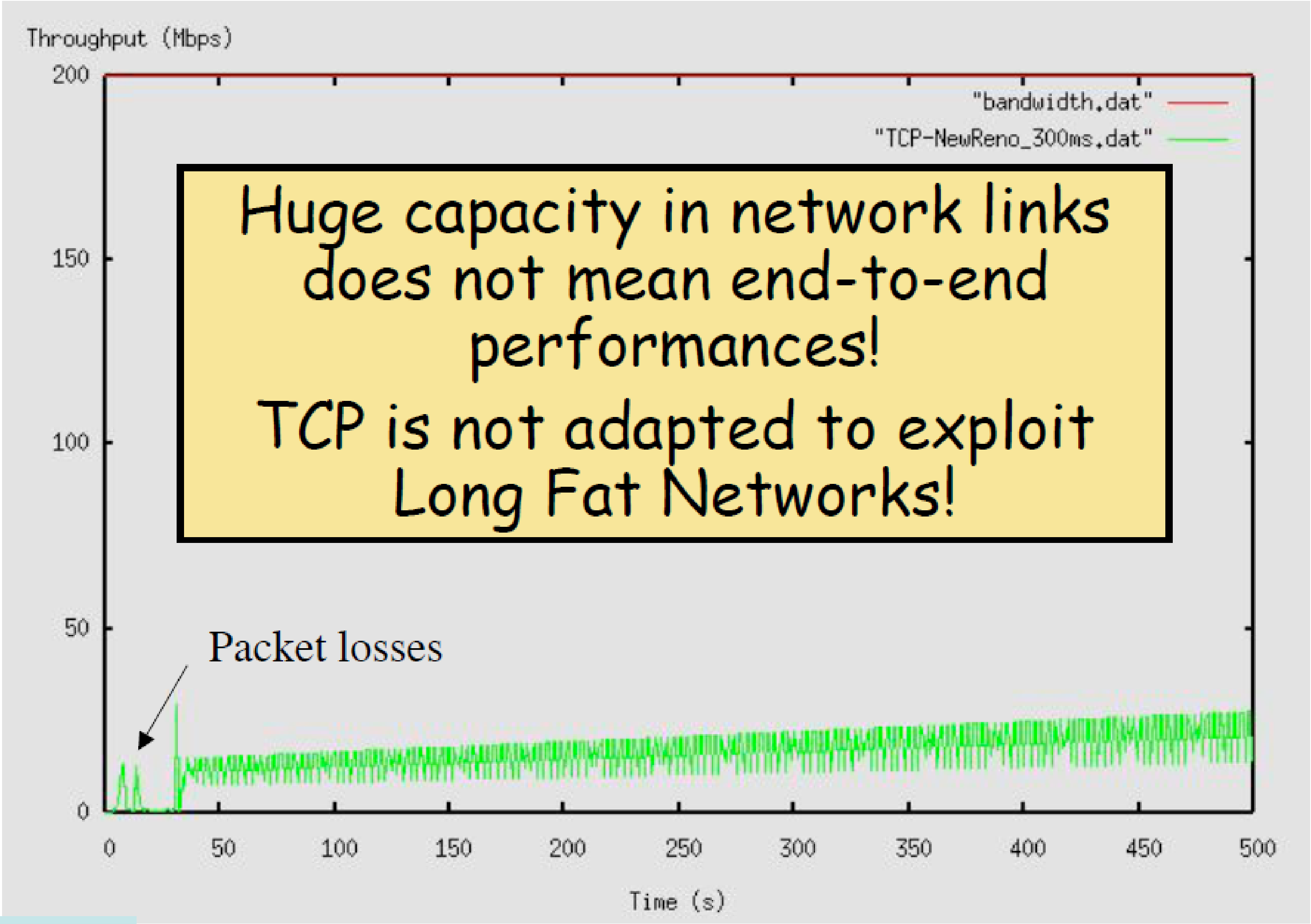


TCP in high-speed links

- Today's backbone links are optical, DWDM−based,
- High bit rates (1~100 Gbps)
- Long distances (Thousands of Km)
- Transmission time <<< propagation time

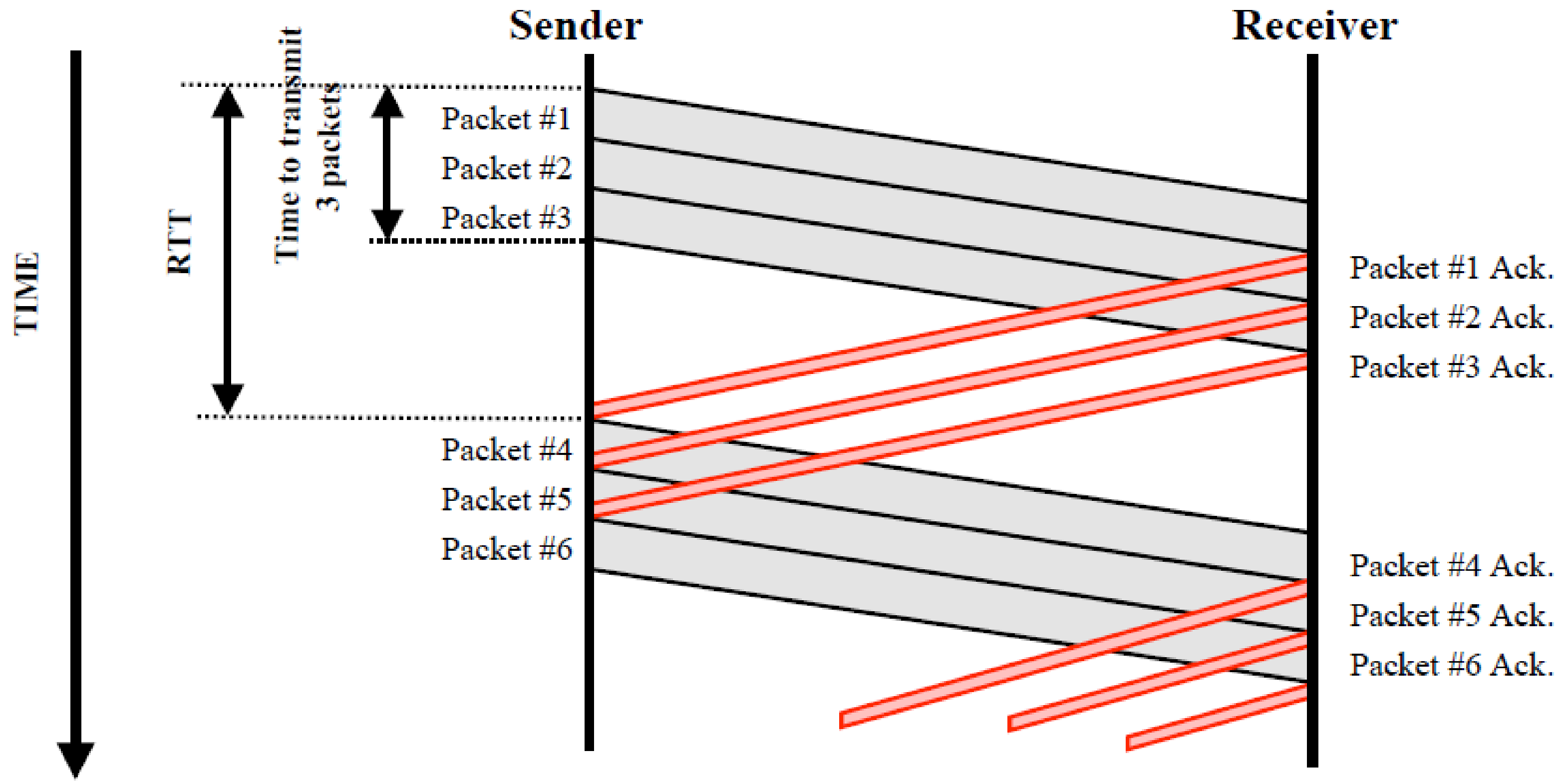


TCP throughput on a 200 Mbps link



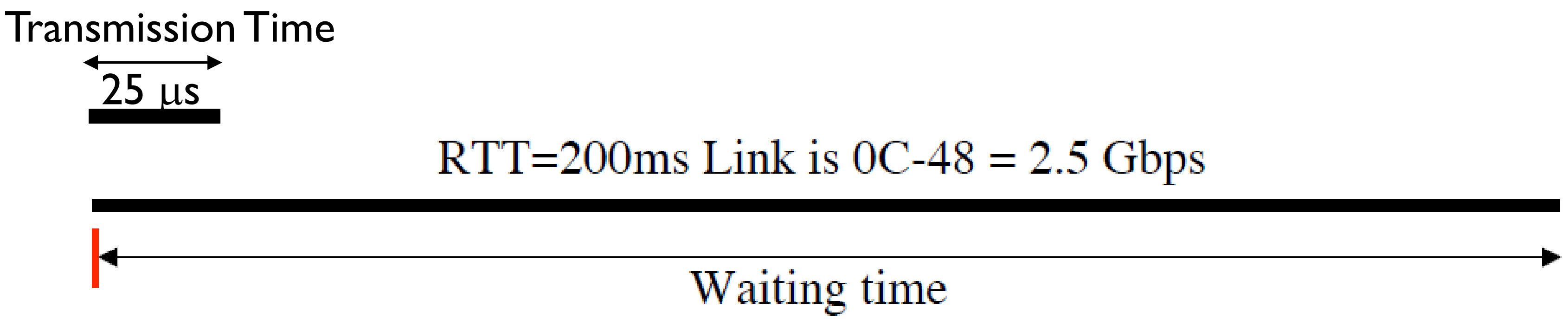
Problem: window size

- The default maximum window size is 64Kbytes.
- Then the sender has to wait for acks.



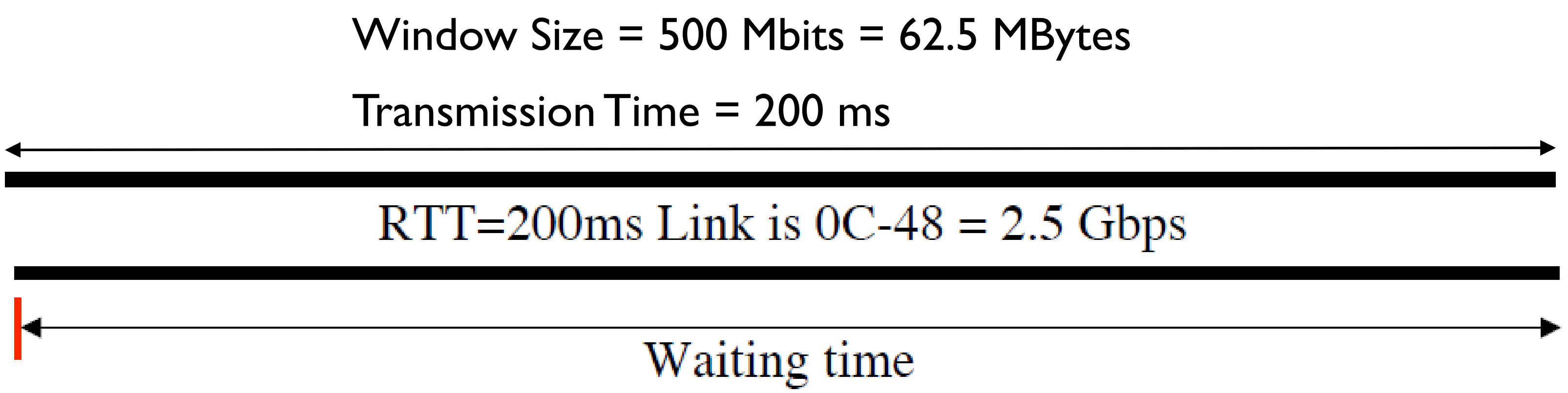
Problem: window size

- Less than 0.01% of the link bandwidth is utilized
- A big file transfer takes minutes



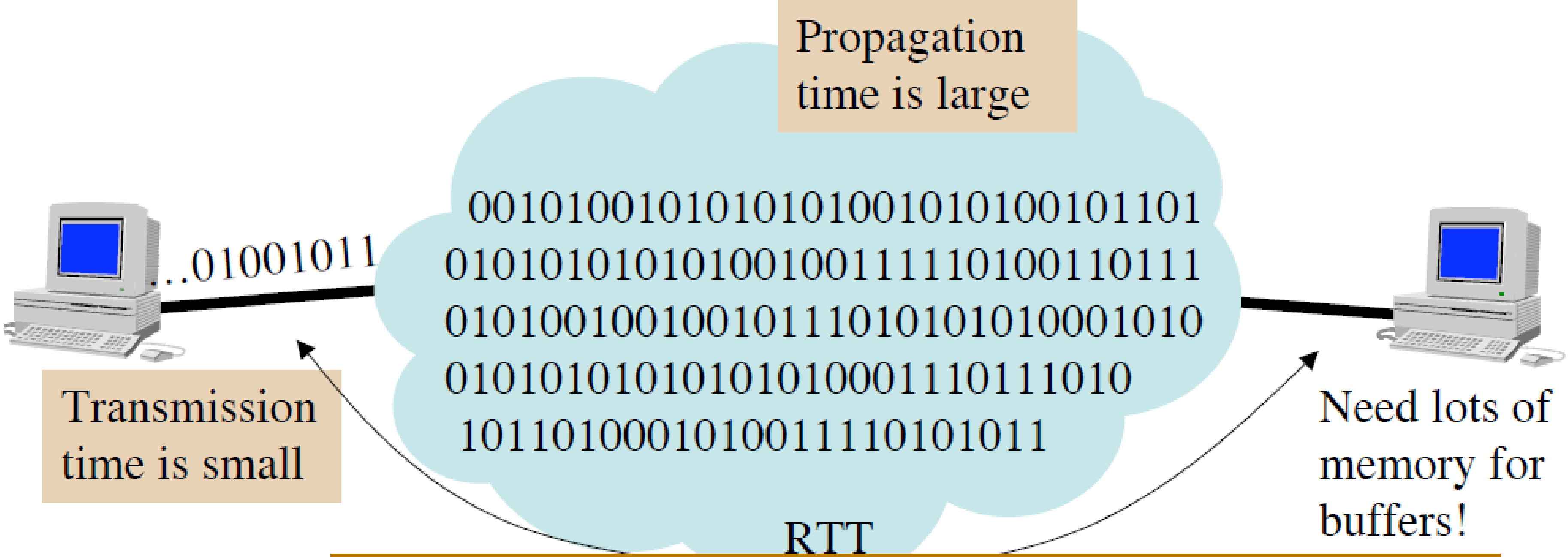
Problem: window size

- Much larger window size to utilize the link BW
- To keep sending till the first ACK
  - W = RTTx R bits



High capacity network

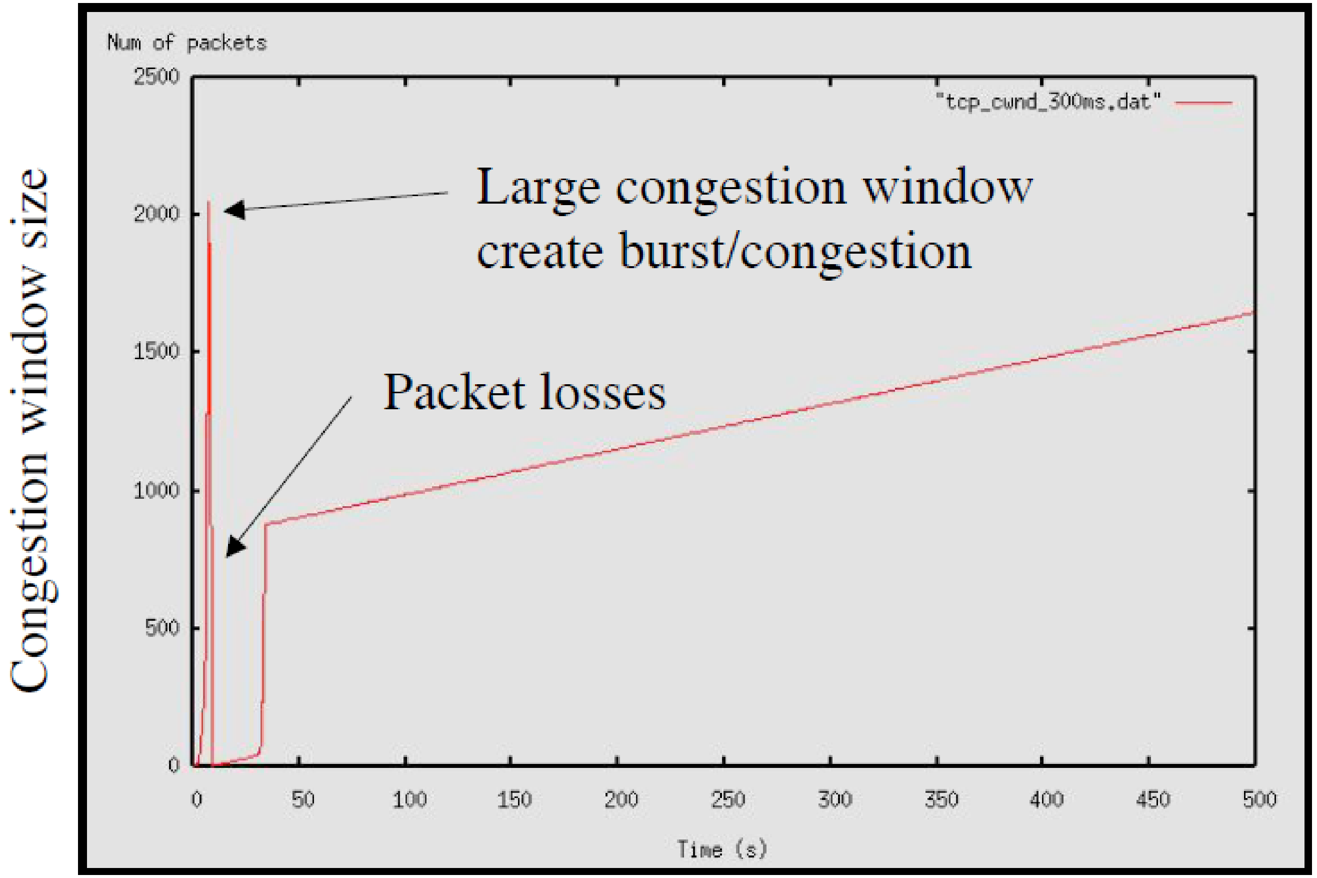
LFN: Long Fat Network



The optimal window size should be set to the bandwidthxRTT product to avoid blocking at the sender side

Side effect of large windows

- TCP becomes very sensitive to packet losses in LFN



High capacity networks

- Sustaining high congestion windows:

A Standard TCP connection with:

- 1500-byte packets;
- a 100 ms round-trip time;
- a steady-state throughput of 10 Gbps;

would require:

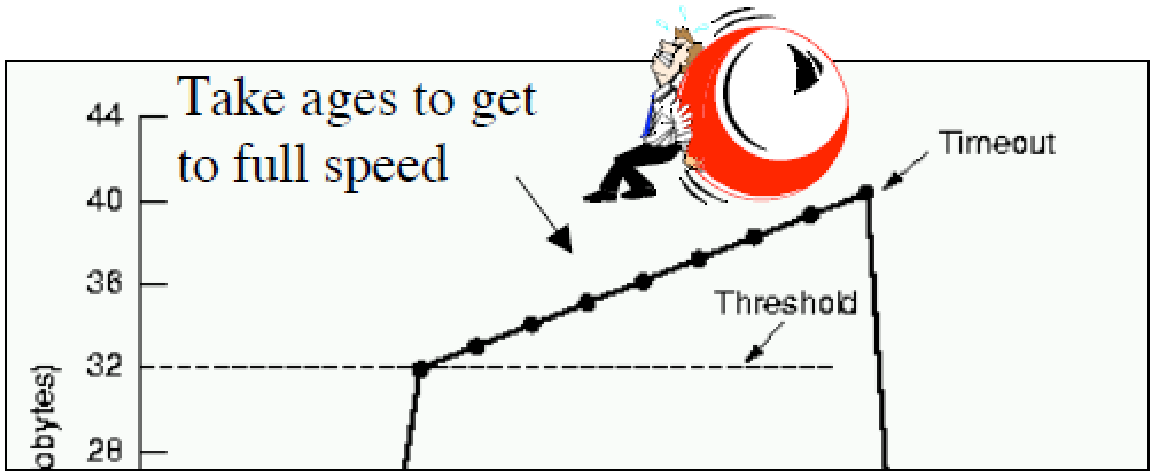
- an average congestion window of 83,333 segments;
- and at most one drop (or mark) every 5,000,000,000 packets (or equivalently, at most one drop every 1 2/3 hours).

This is not realistic.

From S. Floyd

Additive increase is still too slow

- With 100ms of round trip time, a connection needs 203 minutes (3h23) to get 1Gbps starting from 1Mbps

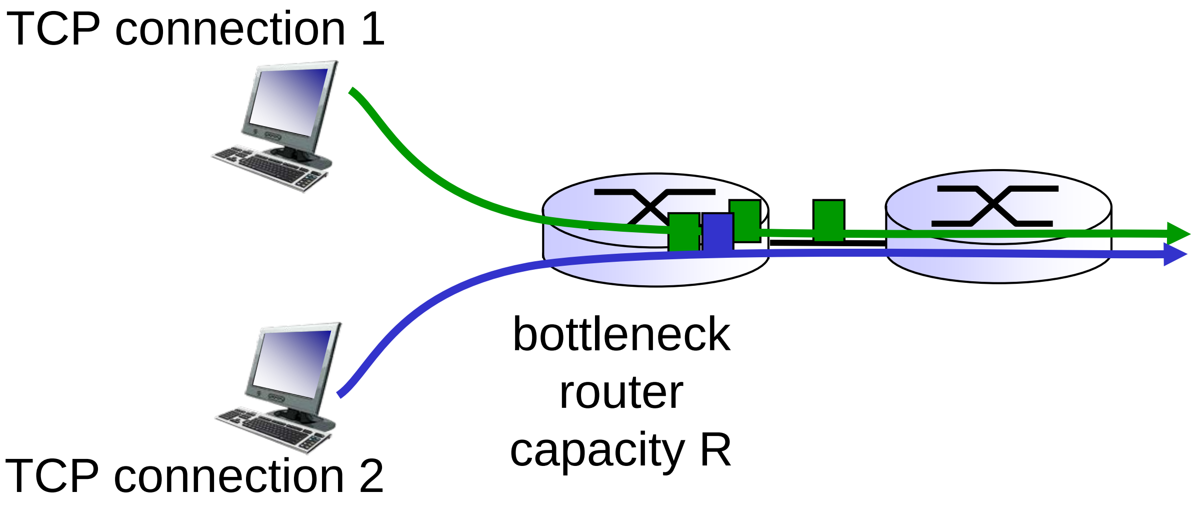


Solution?

- Several parallel TCP connections
  - Requires less configuration in the standard TCP
- New TCP Protocols for high-speed links
  - Research
    - Fast TCP
  - OS:
    - Cubic

TCP Fairness

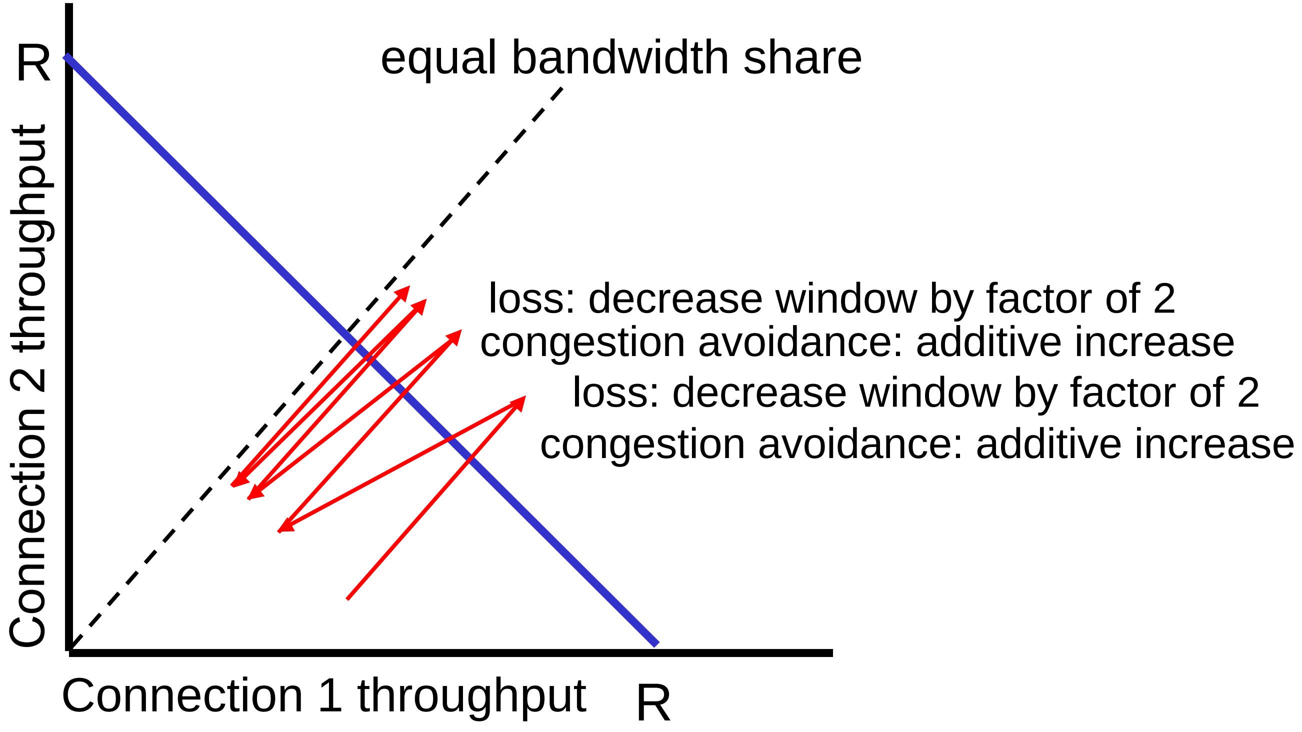
fairness goal: if K TCP sessions share same bottleneck link of bandwidth R, each should have average rate of R/K



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

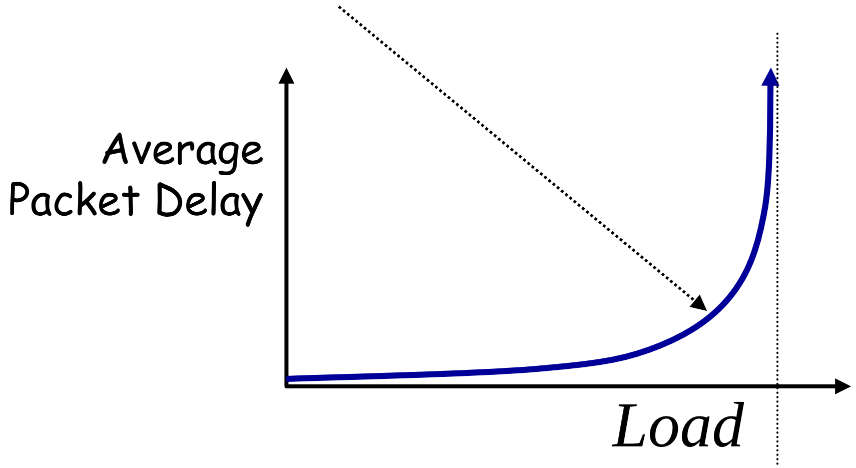
- multimedia apps often do not use TCP
  - do not want rate throttled by congestion control
- instead use UDP:
  - send audio/video at constant rate, tolerate packet loss

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

Congestion Avoidance

- TCP reacts to congestion *after* it takes place. The data rate changes rapidly and the system is barely stable (or is even unstable).
- Can we *predict* when congestion is about to happen and avoid it? E.g. by detecting the knee of the curve.



Congestion Avoidance Schemes

- Router-based Congestion Avoidance:
  - DECbit:
    - Routers explicitly notify sources about congestion.
  - Random Early Detection (RED):
    - Routers implicitly notify sources by dropping packets.
    - RED drops packets at random, and as a function of the level of congestion.
- Host-based Congestion Avoidance
  - Source monitors changes in RTT to detect onset of congestion.

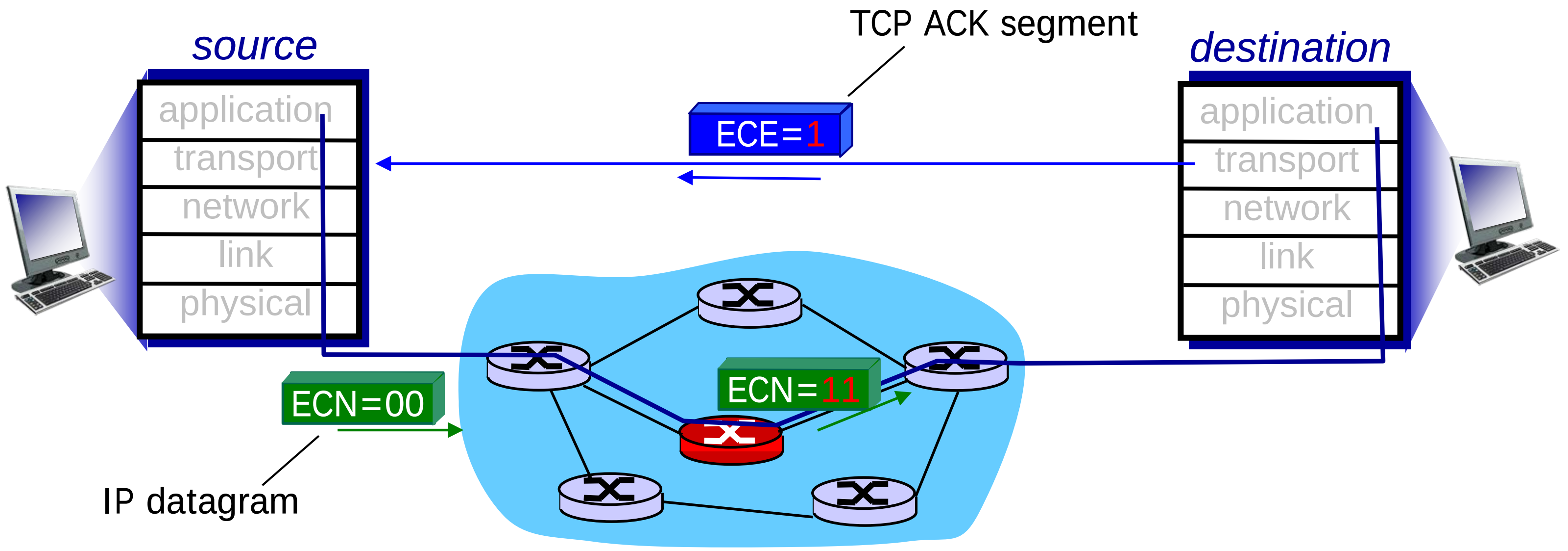
TCP Vegas

- Not much in use but continues to be studied
  - Goals:
    - Detect congestion in routers between source and destination before packet loss occurs
    - Lower the rate linearly when this imminent packet loss is detected
      Predicted by observing the RTTs
    - Longer RTT than expected == greater congestion in routers

Explicit Congestion Notification (ECN)

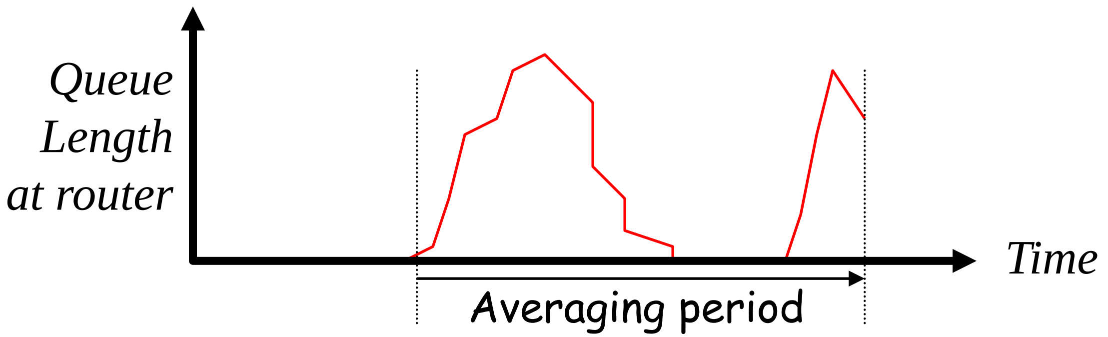
network-assisted congestion control:

- two bits in IP header (ToS field) marked *by network router* to indicate congestion
- congestion indication carried to receiving host
- receiver (seeing congestion indication in IP datagram)) sets ECE bit on receiver-to-sender ACK segment to notify sender of congestion



DECbit

- Each packet has a "Congestion Notification" bit called the DECbit (Destination. Experiencing Congestion Bit) in its header.
- If any router on the path is congested, it sets the DECbit.
  - Set if average queue length >= 1 packet, averaged since the start of the previous busy cycle.
- To notify the source, the destination copies DECbit into ACK packets.
- Source adjusts rate to avoid congestion.
  - Counts fraction of DECbits set in each window.
  - If <50% set, increase rate additively.
  - If >=50% set, decrease rate multiplicatively.



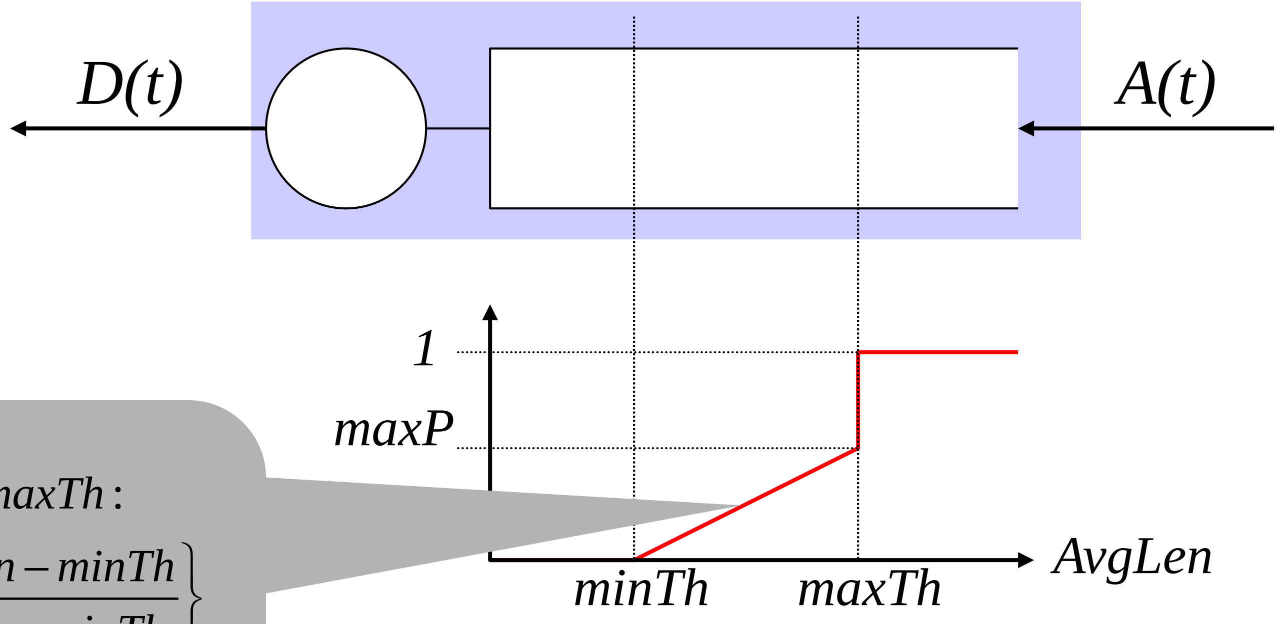
Random Early Detection (RED)

- RED is similar to DECbit, and was designed to work well with TCP.
- RED implicitly notifies sender by dropping packets.
- Drop probability is increased as the *average* queue length increases.
- (Geometric) moving average of the queue length is used so as to detect long term congestion, yet allow short term bursts to arrive.

$$AvgLen_{n+1} = (1-\alpha) \times AvgLen_n + \alpha \times Length_n$$

i.e. $AvgLen_{n+1} = \sum_{i=1}^{n} Length_i (\alpha)(1-\alpha)^{n-i}$

RED Drop Probabilities



If $minTh < AvgLen < maxTh$:

$$\hat{p}_{AvgLen} = maxP\left\{\frac{AvgLen - minTh}{maxTh - minTh}\right\}$$

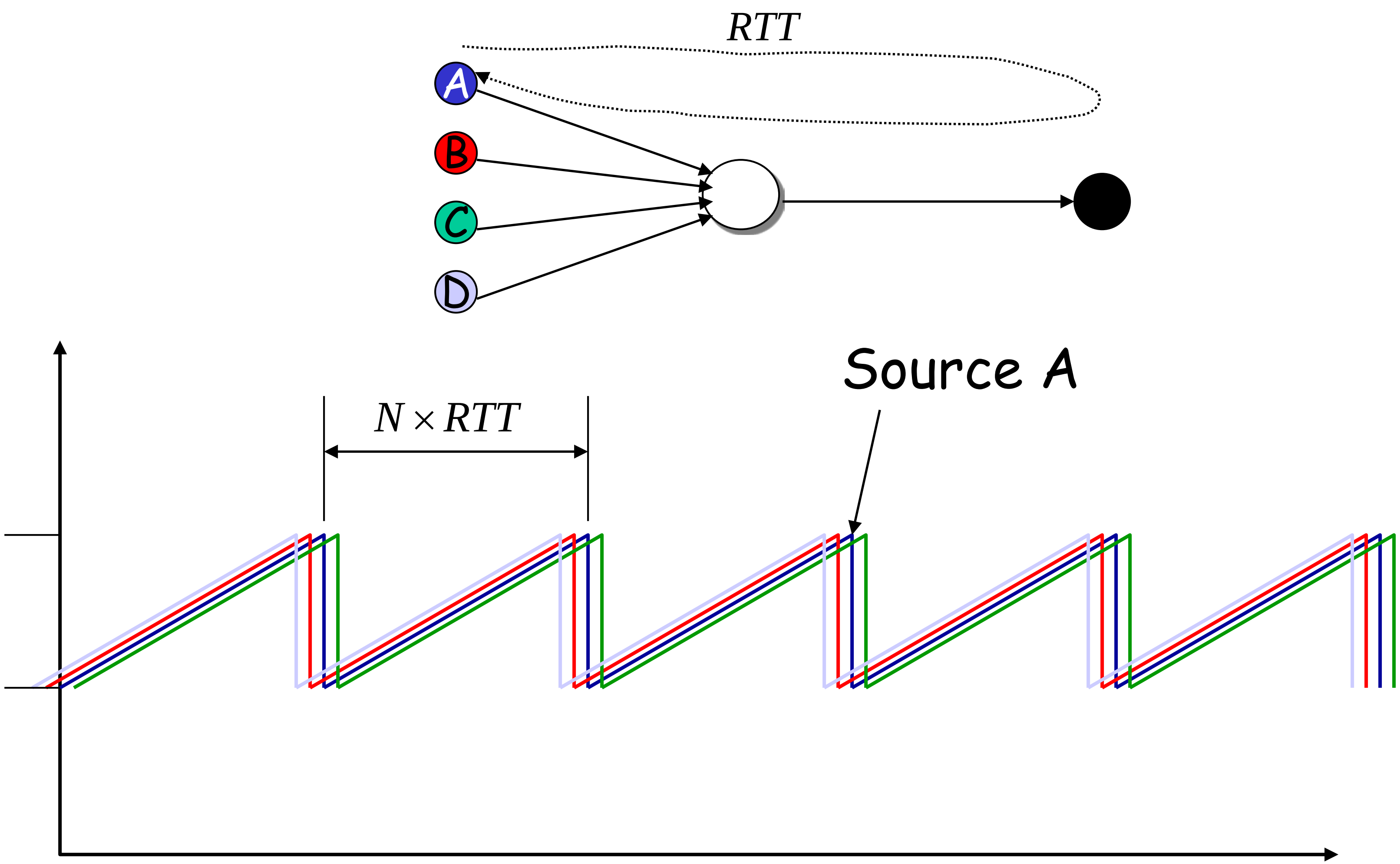
$$\Pr(\text{Drop Packet}) = \frac{\hat{p}_{AvgLen}}{1 - count \times \hat{p}_{AvgLen}}$$

$count$ counts how long we've been in $minTh < AvgLen < maxTh$ since we last dropped a packet. i.e. drops are spaced out in time, reducing likelihood of re-entering slow-start.

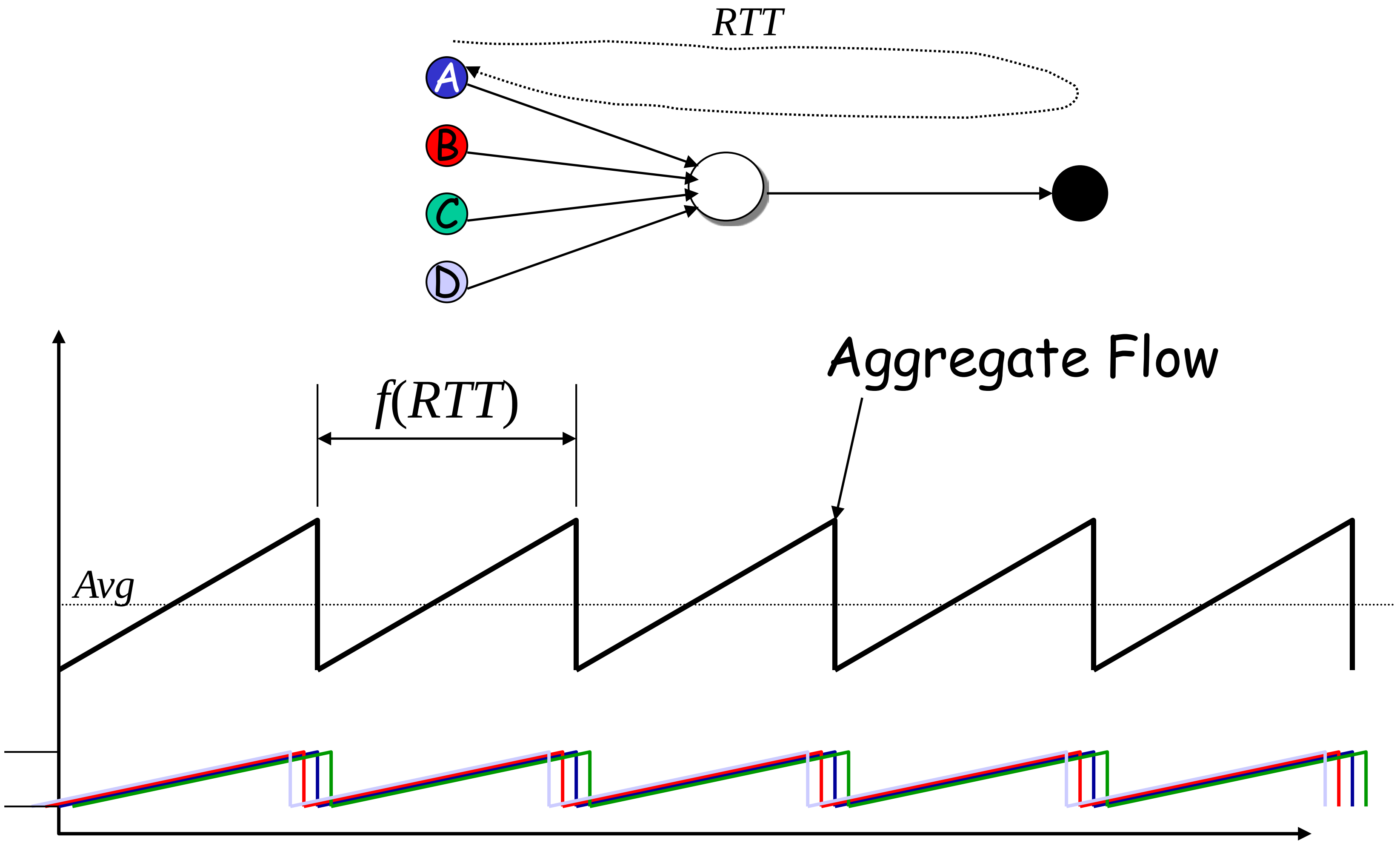
Properties of RED

- Drops packets before queue is full, in the hope of reducing the rates of some flows.
- Drops packet for each flow *roughly* in proportion to its rate.
- Drops are spaced out in time.
- Because it uses average queue length, RED is tolerant of bursts.
- Random drops hopefully desynchronize TCP sources.

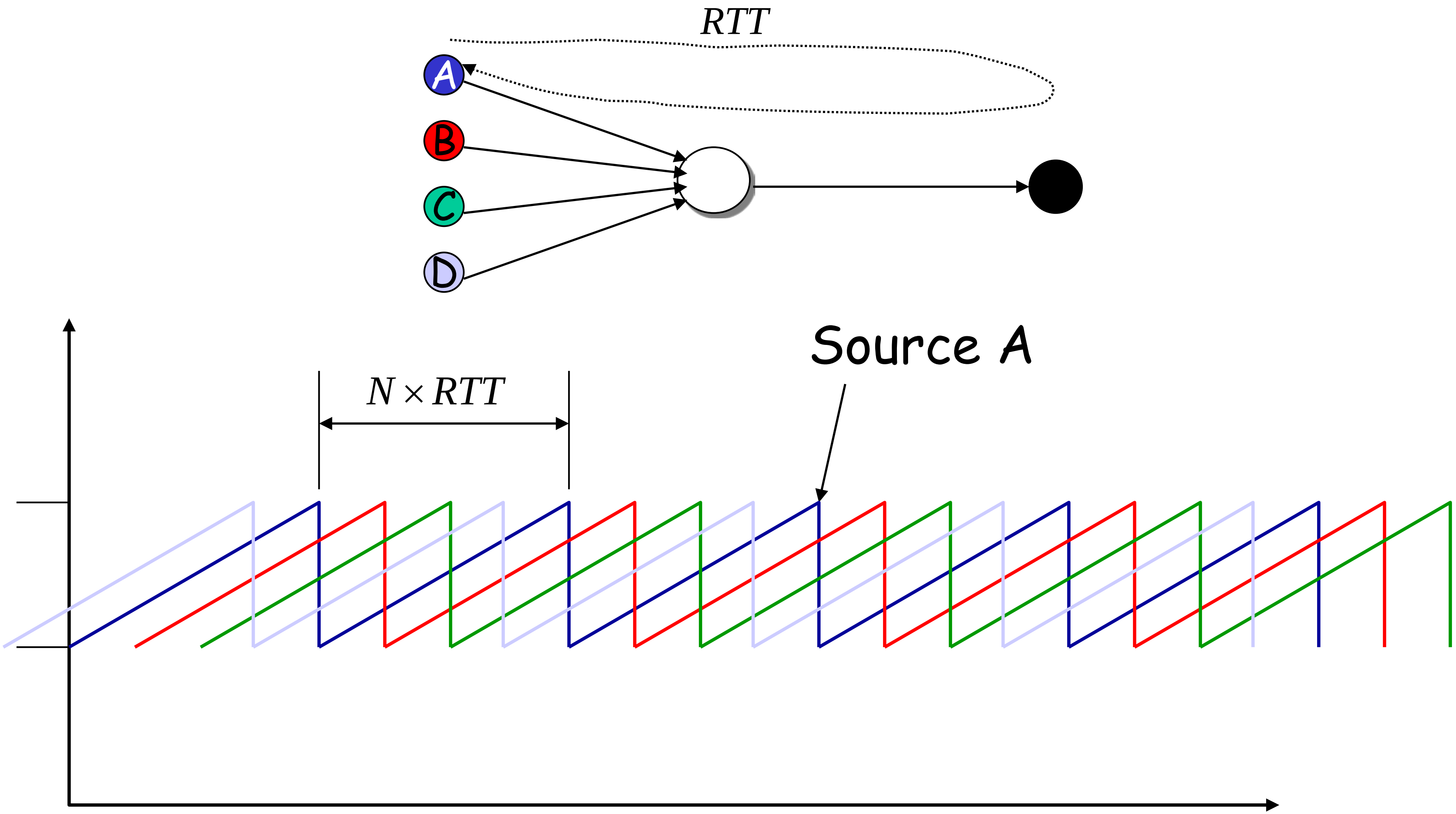
Synchronization of sources



Synchronization of sources



Desynchronized sources



Desynchronized sources