Lecture 9
Congestion Control: Part II

EECS 122

University of California

Berkeley

TOC: Congestion Control 2

- Quick Review of TCP's CC
- Cheating TCP
- ECN
- Noisy Links
- Virtual Queues
- RED
- How Big are Router Buffers?
- Unfairness
- TCP Vegas
- Fast Links

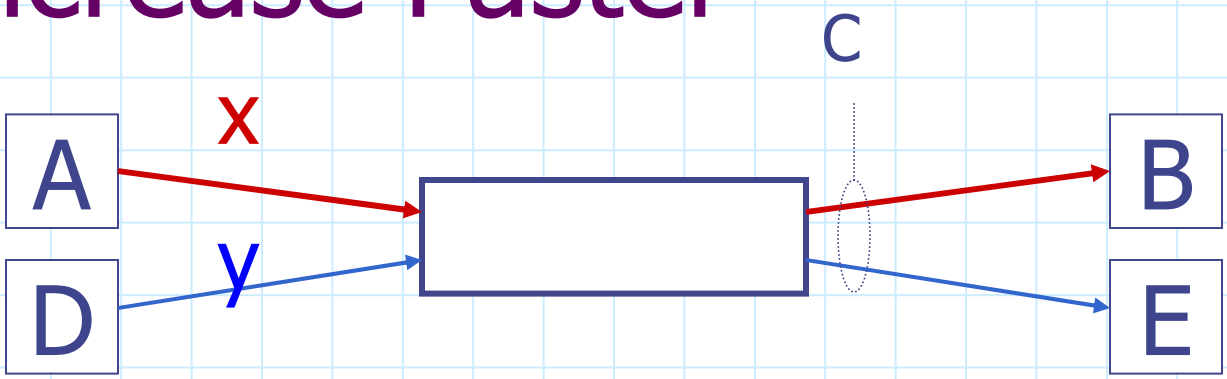
CC2: Review of TCP's CC

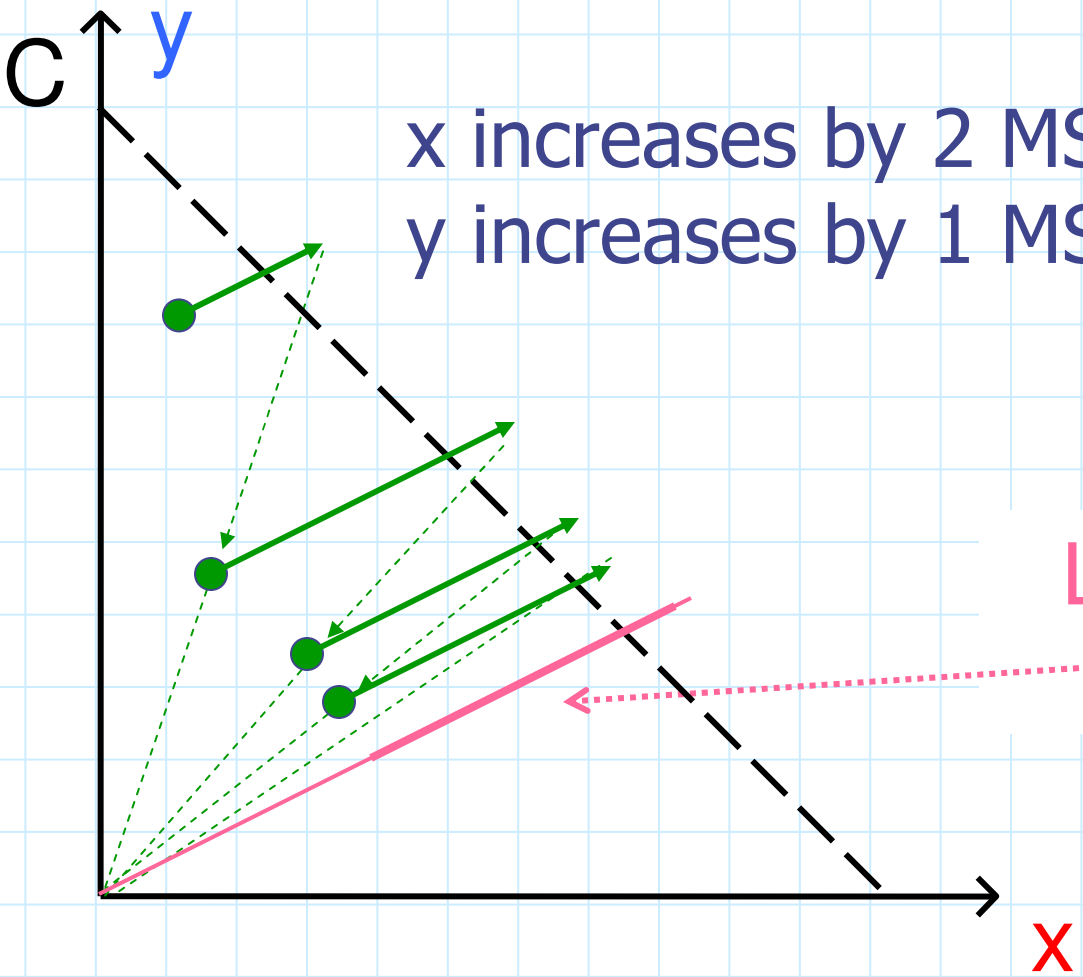
- Goal: Fair efficient sharing of links
- Steps in TCP: 3WH, Exchange, 1st ½ Close; 2nd ½ Close
- Slow Start:
  - Exp. Increase of Window to discover bdw (W = W +1 …)
  - TO → ssthresh = W/2
  - Repeat until W = ssthresh, then CA
- Congestion Avoidance:
  - AIMD: + 1 MSS/RTT (W = W + 1/W…); W/2 when 3DA
- Timeout: mean + 4 deviations
  - double when triggered, then W =1 & SS
  - reset after new ACK or new packet
- Flow Control: min{RAW – OUT, W}

CC2: Cheating TCP

- Some Methods:
  - Increase faster than 1/RTT per RTT
  - Start SS with W > 1
  - Open many connections
- Why Not?

Cheating: Increase Faster



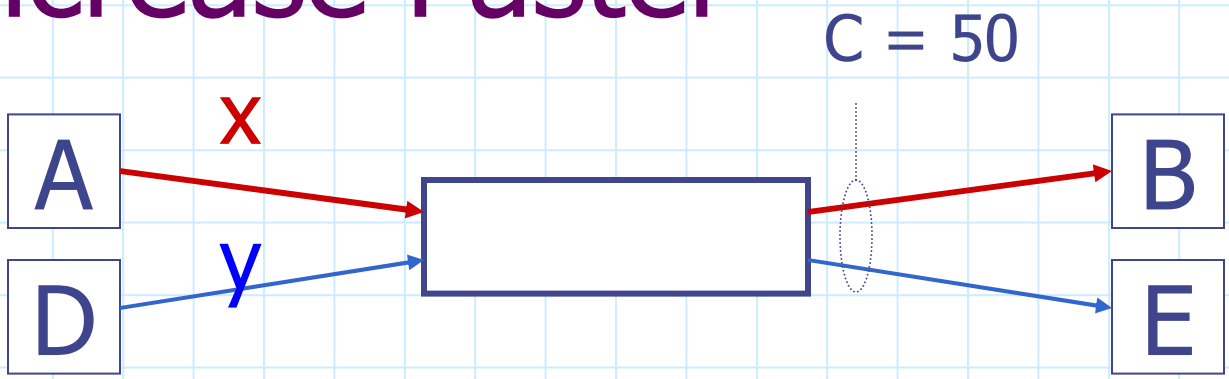


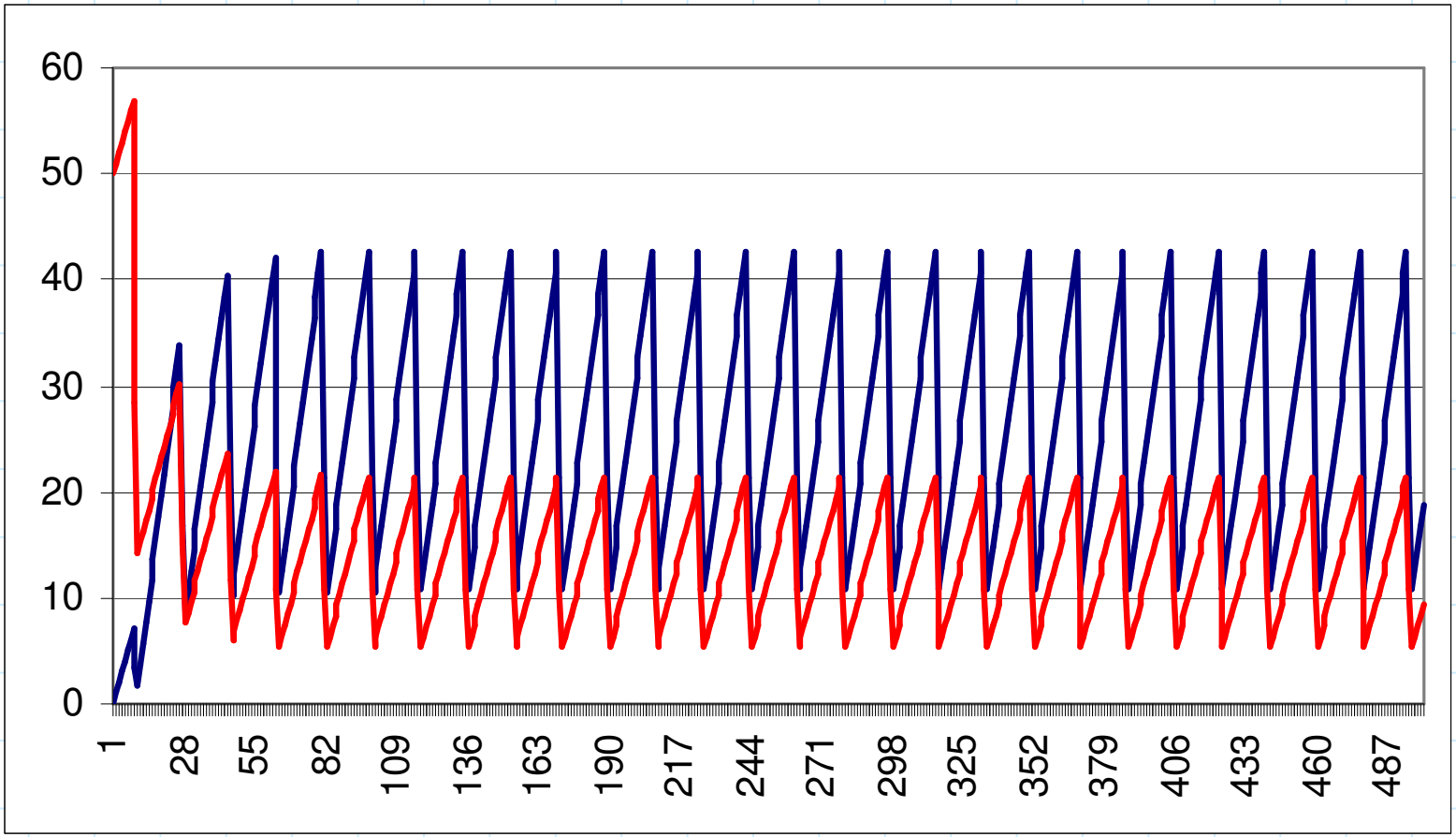
x increases by 2 MSS/RTT per RTT
y increases by 1 MSS/RTT per RTT

Limit rates:
$x = 2y$

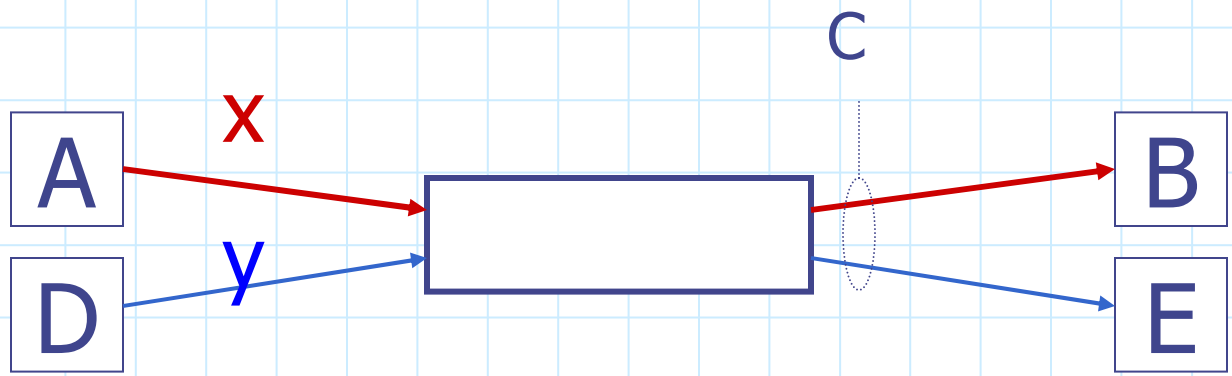
SIM

Cheating: Increase Faster

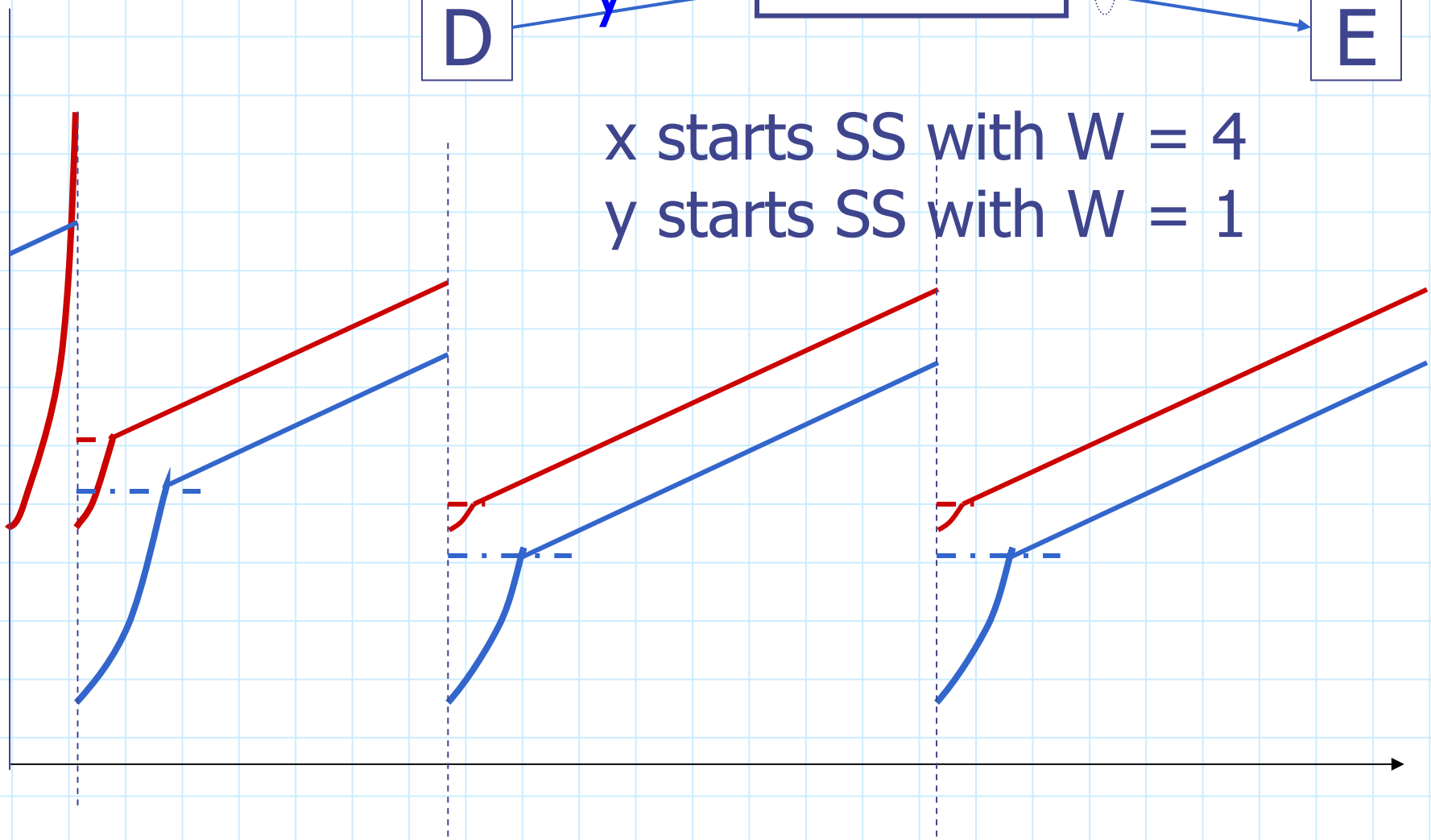




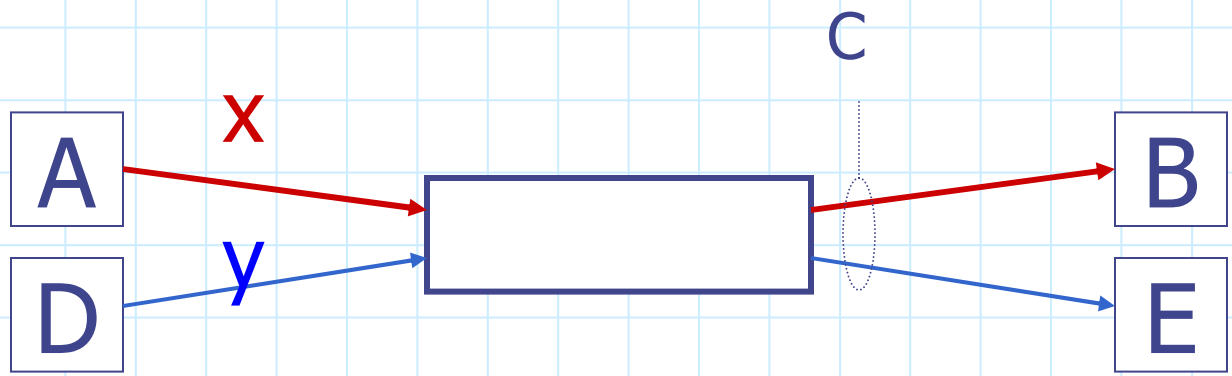
Cheating: Start SS with W > 1



x starts SS with W = 4
y starts SS with W = 1



Cheating: Open Many Connections

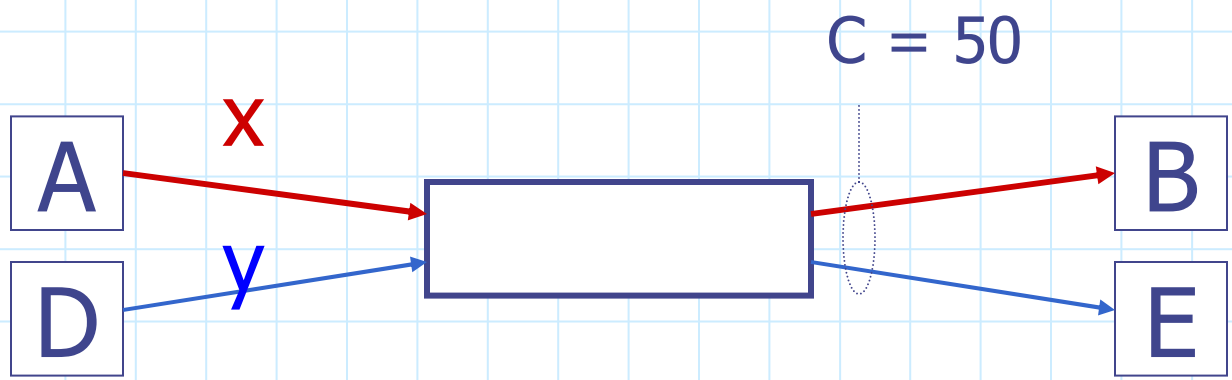


Assume

- A starts 10 connections to B
- D starts 1 connection to E
- Each connection gets about the same throughput

Then A gets 10 times more throughput than D

Cheating: Why Not?



D → Increases by 1 Increases by 5

(x, y)

A ↓

Increases by 1	22, 22	10, 35
Increases by 5	35, 10	15, 15

Too aggressive
→Losses
→Throughput falls

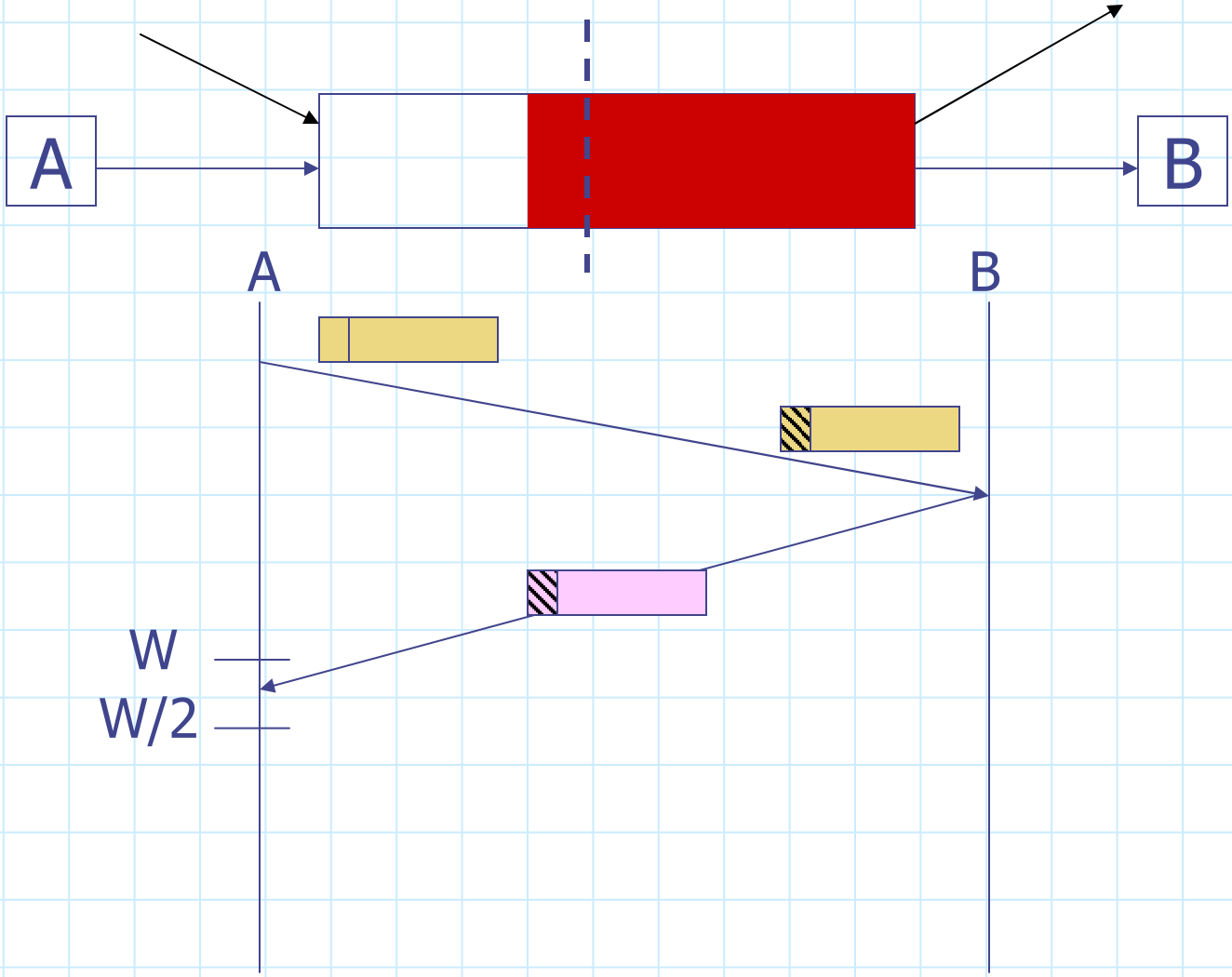
One shot: A has an incentive to "cheat"
Long Term: Users have an incentive not to cheat

CC2: Explicit Congestion Notification

- Standard TCP:
  - Losses needed to detect congestion
  - Wasteful and unnecessary
- ECN:
  - Routers mark packets instead of dropping them
  - Destination marks ACKs of marked packets
  - Source set W = W/2 when it sees mark
- Advantages:
  - No time wasted to retransmit
  - Link errors not confused with congestion
- Illustration
- Backward Compatibility

CC2: Explicit Congestion Notification

- Illustration:



CC2: Explicit Congestion Notification

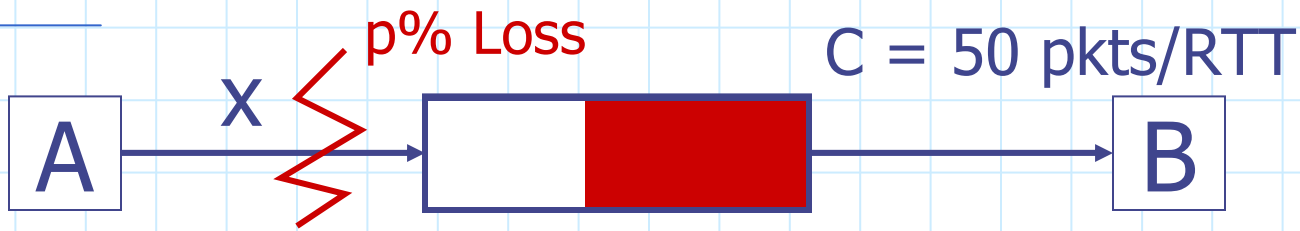
Backward Compatibility:

- Bit in Header indicates if hosts implement ECN
- If it does, router marks packet
- If it does not, router drops packet

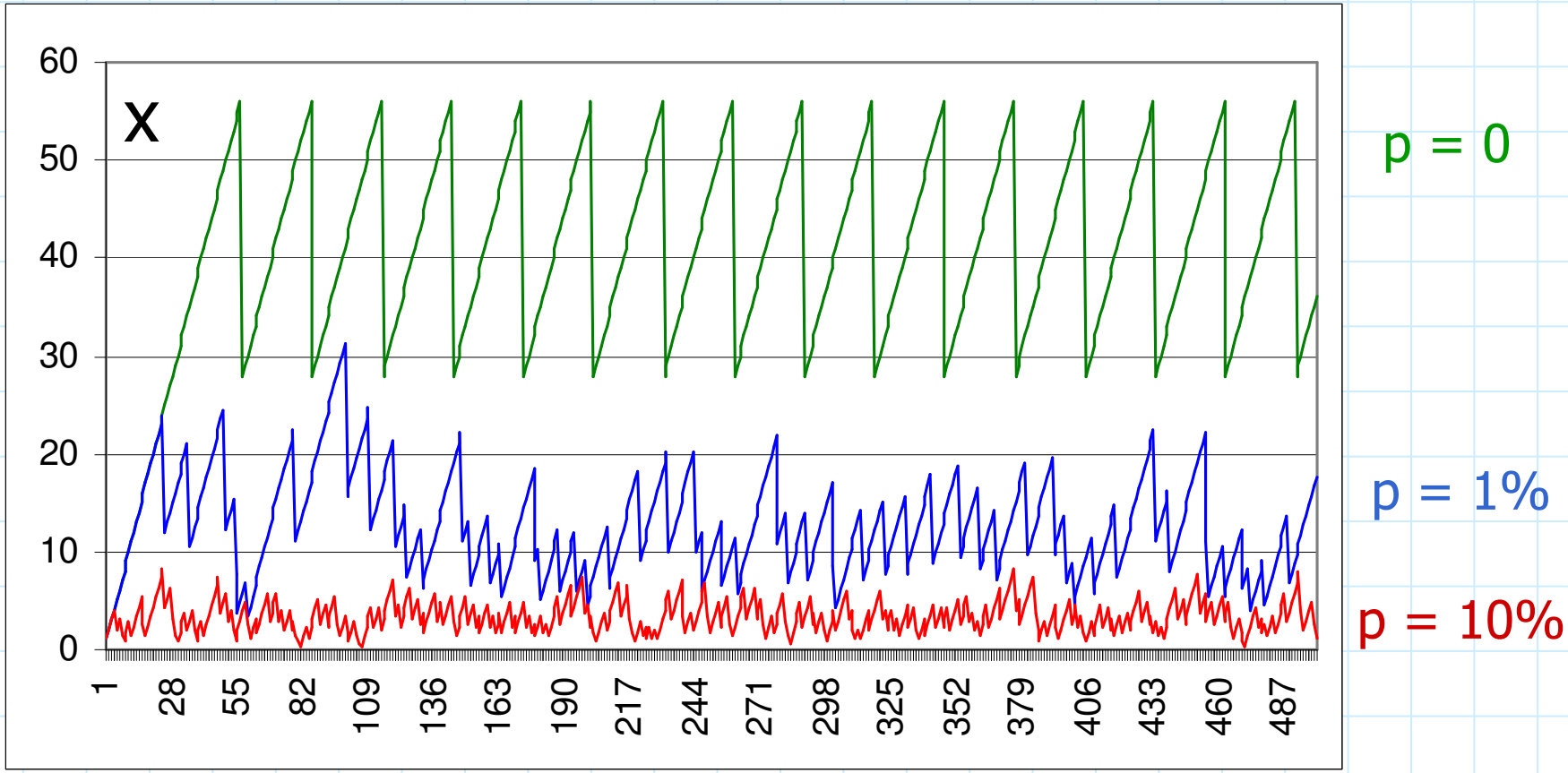
CC2: Noisy Link

- Basic TCP assumption:
  - Losses indicate congestion
- What is losses come from link errors?
  - Slowing down does not help; in fact it hurts
- Illustration
- Solutions

Noisy Link: Illustration



- No congestion → x increases by one packet/RTT every RTT
- Congestion → decrease x by factor 2

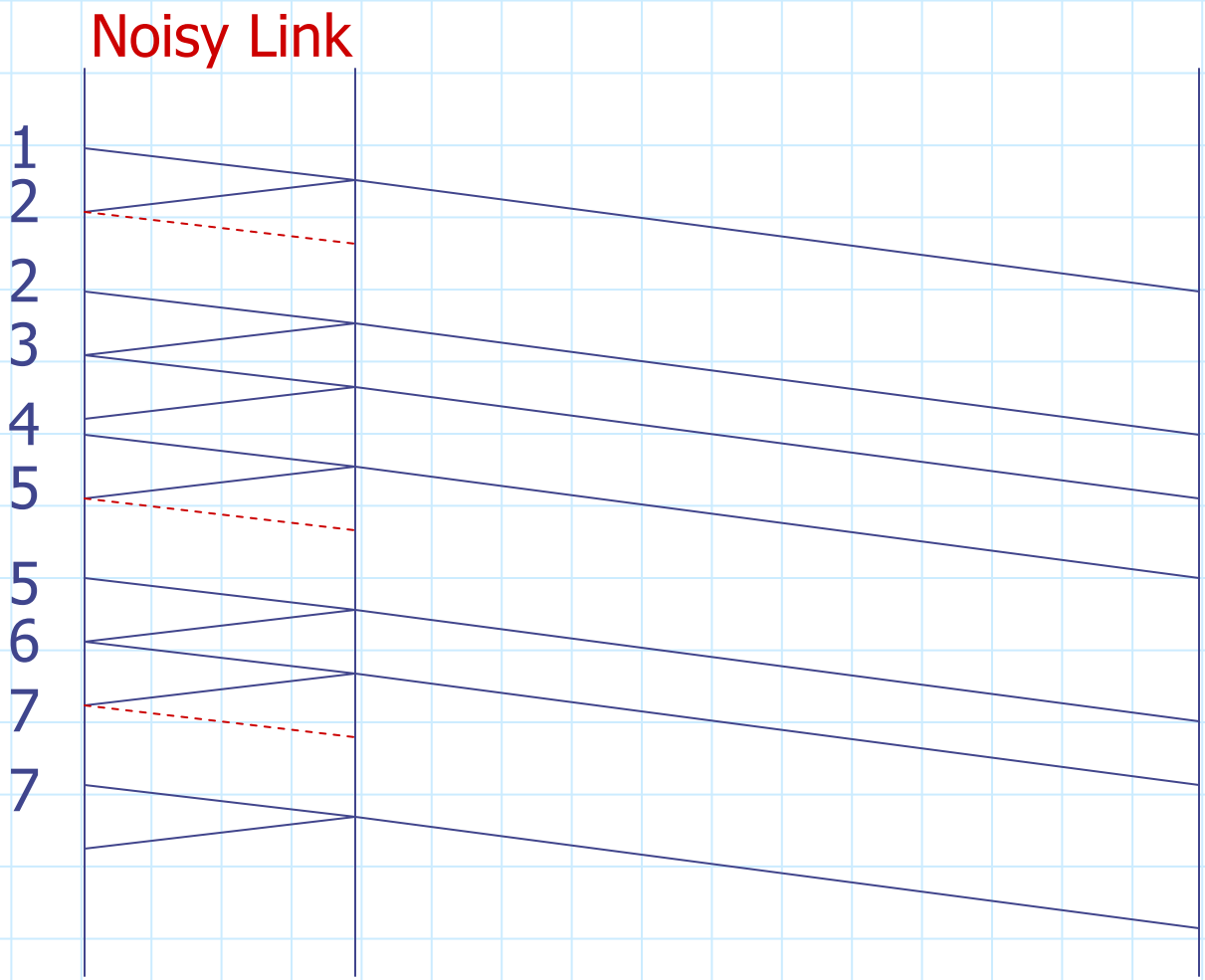


Noisy Link: Solutions

- Link Error Control
- ECN
- Link Layer Hint
- Various other schemes …

Noisy Link: Solutions: Link Error Control

- Noisy link implements a retransmission protocol

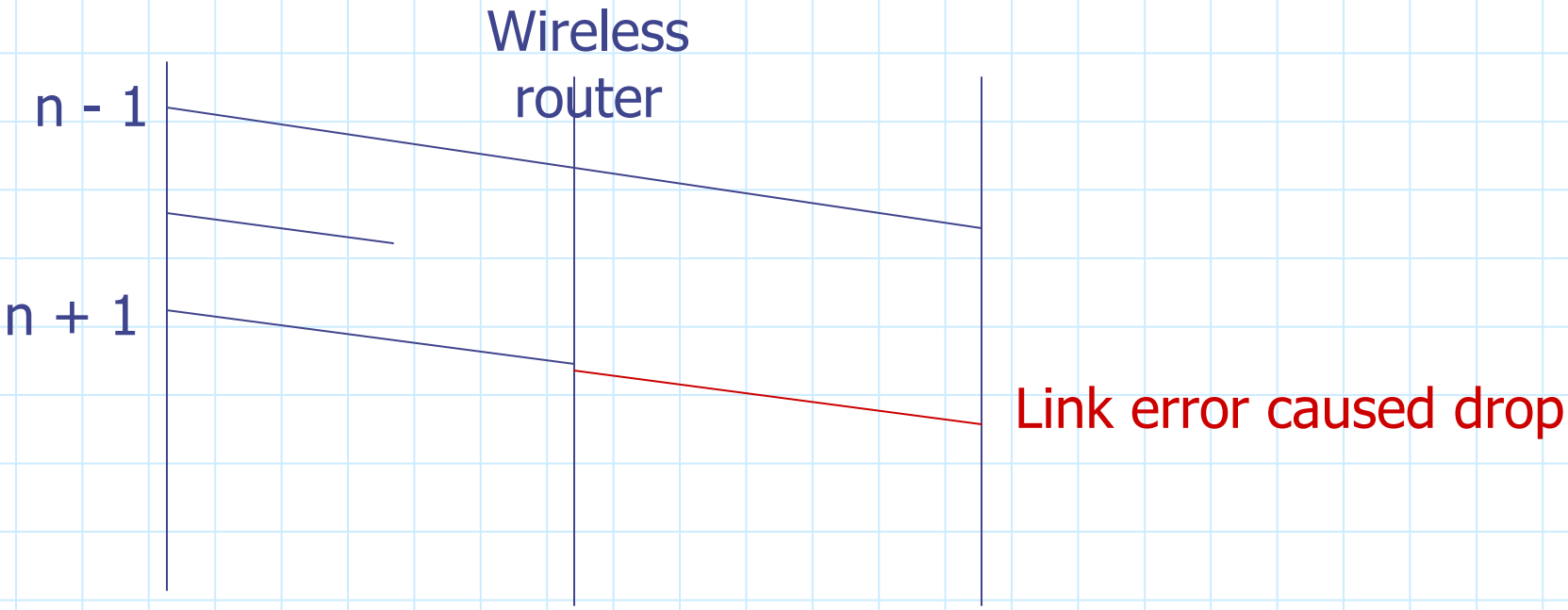


Noisy Link: Solutions: ECN

- If all the routers implement ECN, then a link error is not confused with congestion.
- In that case, a 3DA would indicate a link error and the source should not reduce the window size
- Not applicable today ….

Noisy Link: Solutions: Link Layer Hint

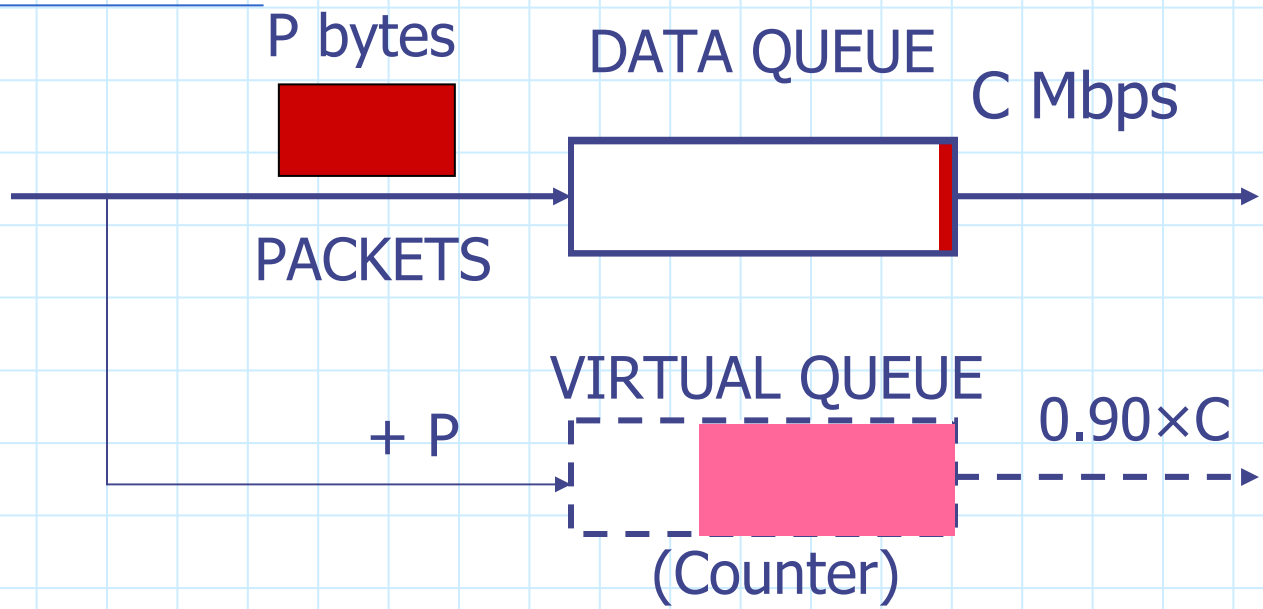
- Link Layer knows when an error occurred
- It indicates that event in next packet
- Destination reflects that fact in ACK
- Source retransmits but does not reduce W



CC2: Virtual Queues

- Motivation:
  - Detect impending congestion before it leads to a queue build-up
  - Reduce losses, delays, need for storage in routers
- Mechanism:
  - Construct a virtual queue that mimics the real queue when served with a fraction of the link rate
- Illustration

CC2: Virtual Queues: Illustration



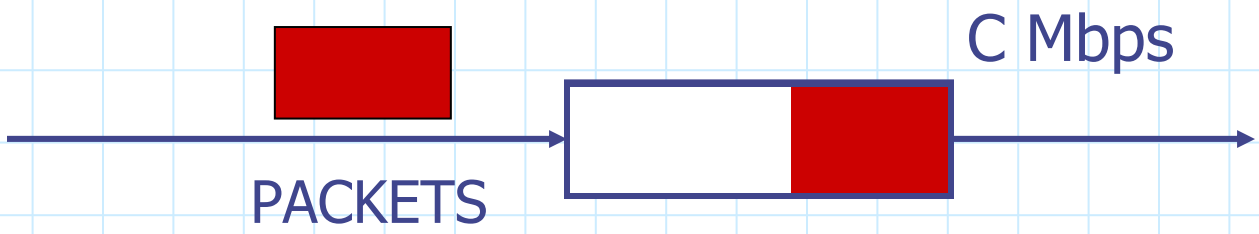
- Virtual queue "detects" when link utilization exceeds 90%, even though the data queue is empty
- The router can use this indication to mark packets in ECN
- Data queues remain almost empty, links used at 90%

CC2: RED

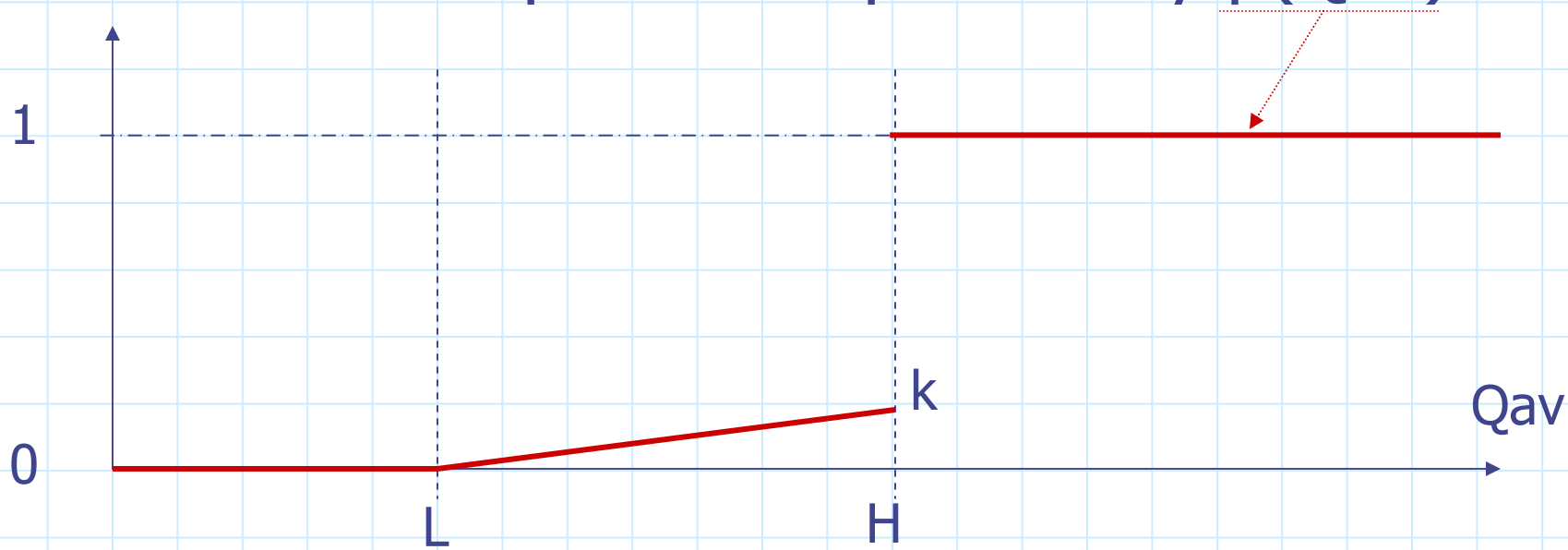
Random Early Detection:
As queue builds up, drop or mark packets with increasing probability (before queue gets full)

- Advantages:
  - Avoids penalizing streams with large bursts
  - De-synchronizes the source behaviors
- Illustration

CC2: RED: Illustration

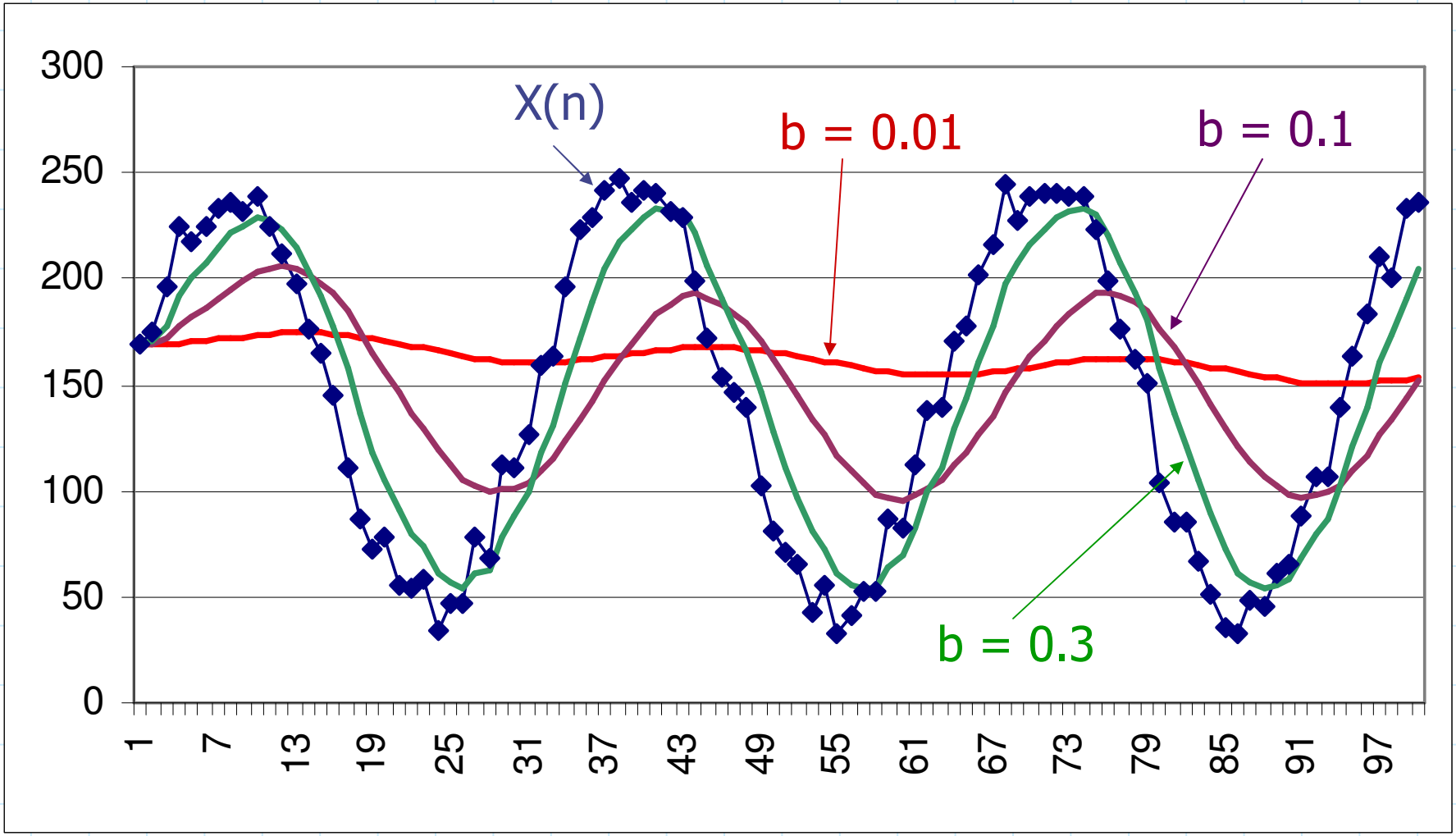


- Calculate recent average of queue length: Qav
  $Qav(n+1) = (1 - b)Qav(n) + Q(n)$
- Determine drop or mark probability p(Qav):



Note: Exponential Averaging

$$A(n+1) = (1 - b)A(n) + bX(n)$$

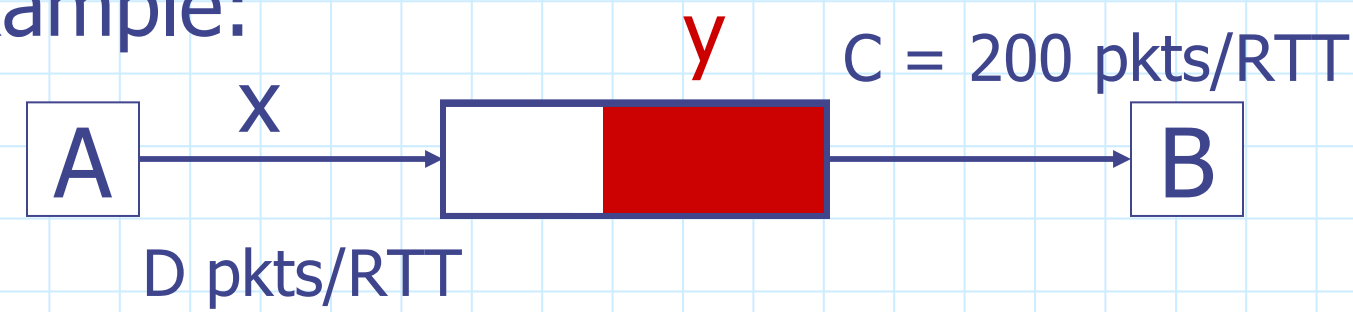


CC2: Router Buffers

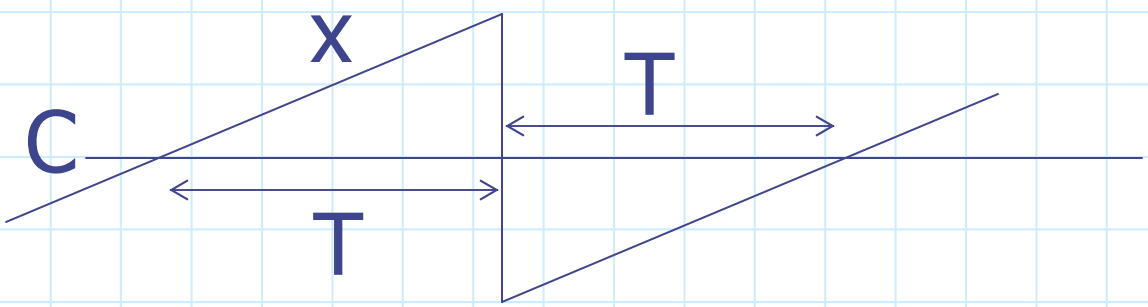
- Our objective:
  - Understand dynamics of queues in routers
- One connection
- Many connections
- Rule of thumb ….

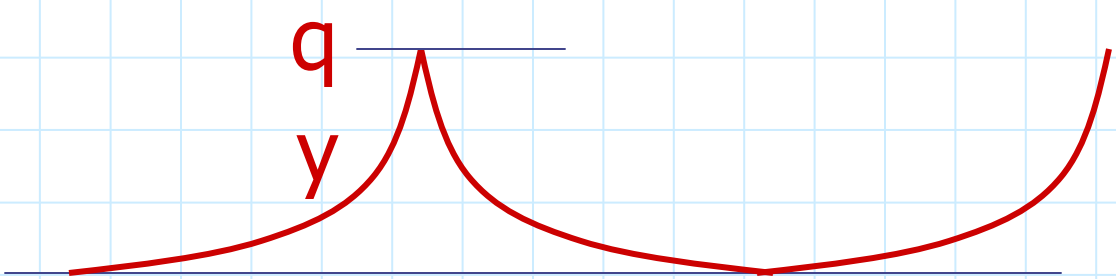
Router Buffers: One Connection

- Example:



Full throughput buffer:





Assume D >> C

$T = C - (C + T)/2$

$\rightarrow T = C/3$

$\rightarrow Q = T^2/2 = C^2/18$

RTT = 200ms

Pkt = 10^4bits

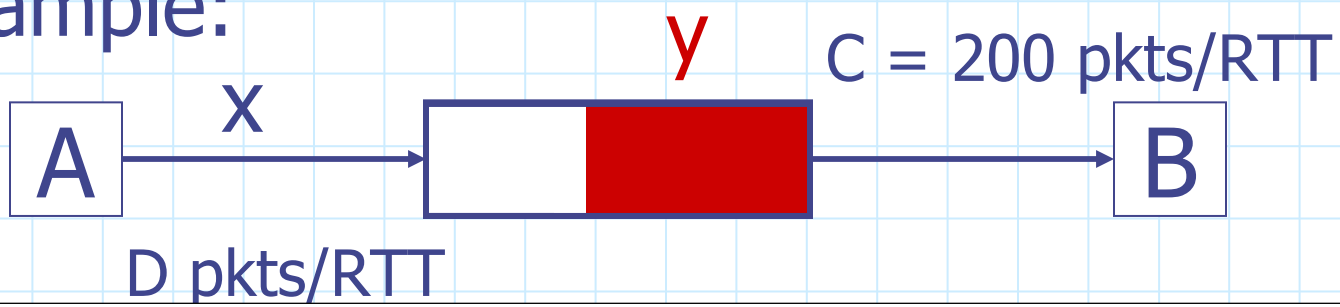
C = 10Mbps = 200

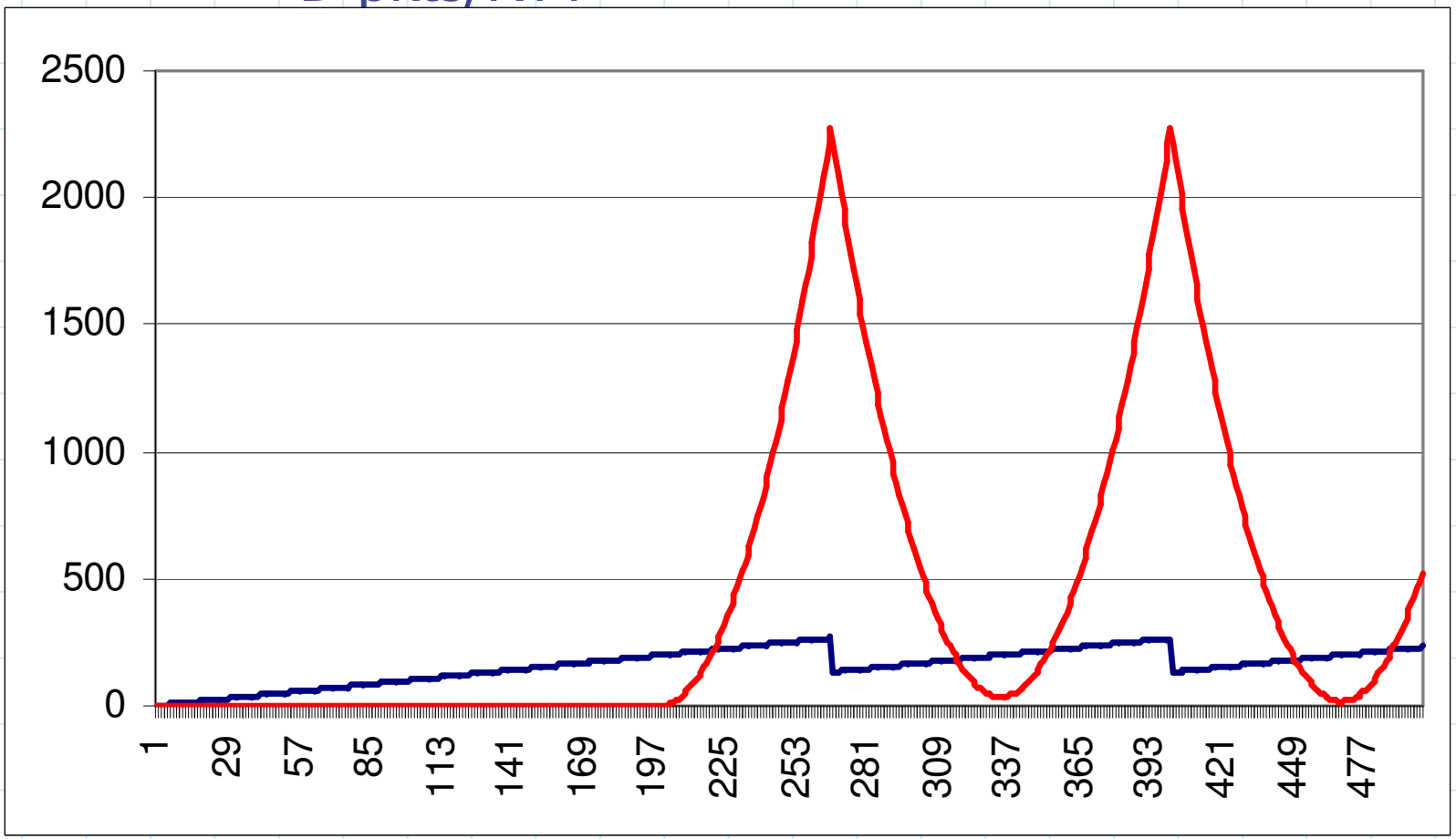
q = 2,200 pkts

= 2.8MBytes

Router Buffers: One Connection

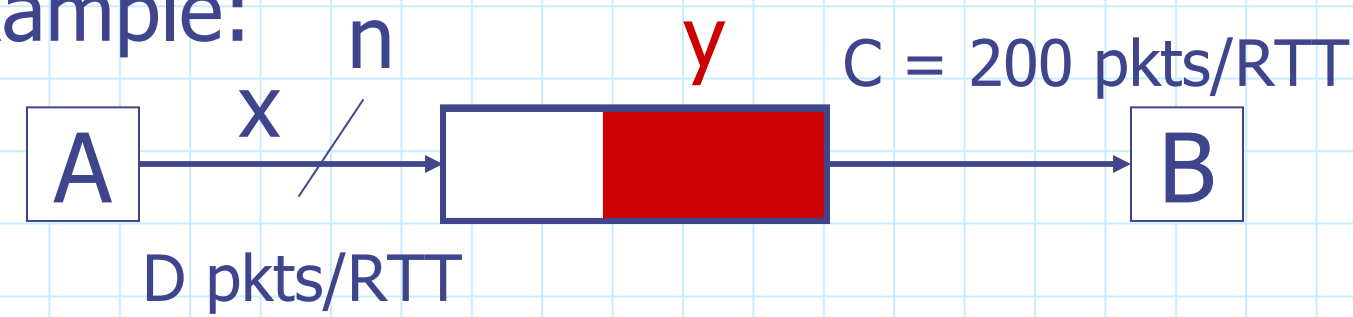
- Example:





Router Buffers: Many Connections

- Example:



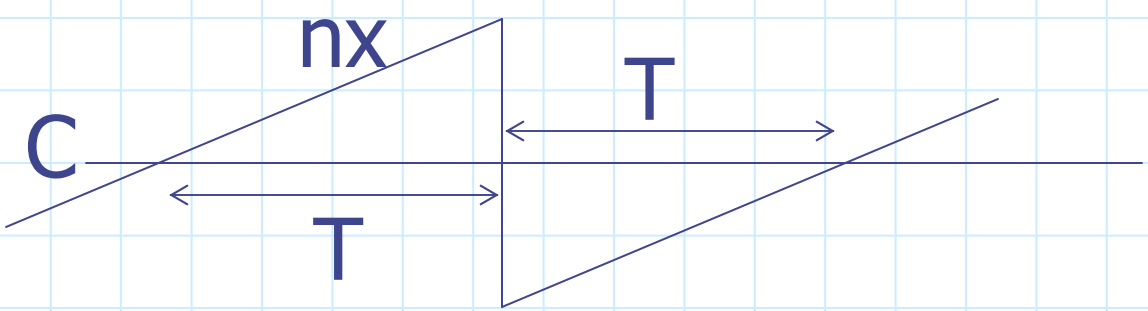
A fraction 2h of sources reduce their rate by 50% [RED]

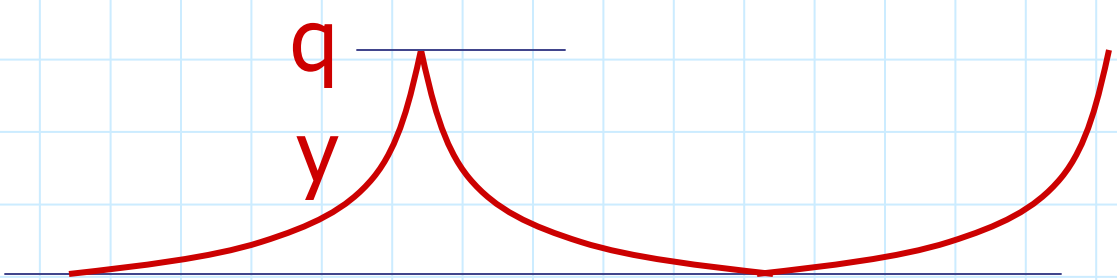
Assume D >> C

$nT = C - hn(C/n + nT)$

$\rightarrow Q = C^2(1 - h)^2/[n(1 + nh)^2]$

Full throughput buffer:





RTT = 200ms

Pkt = 10^4bits

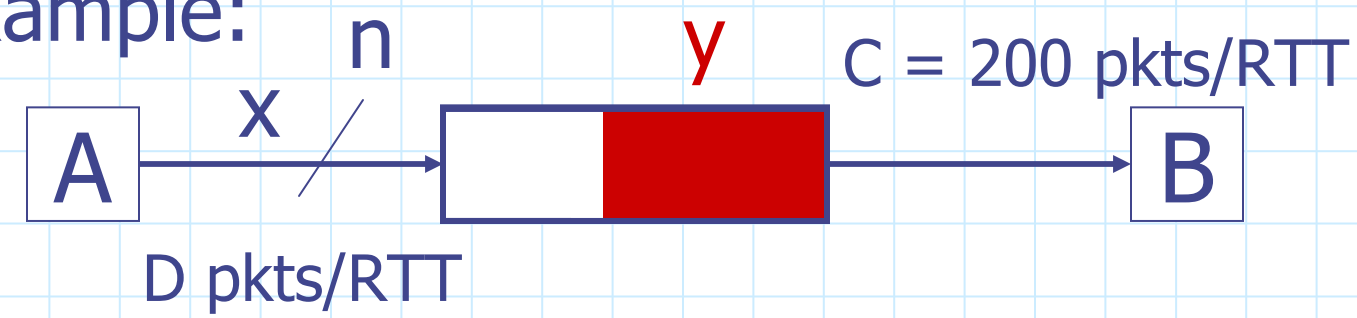
C = 10Mbps = 200

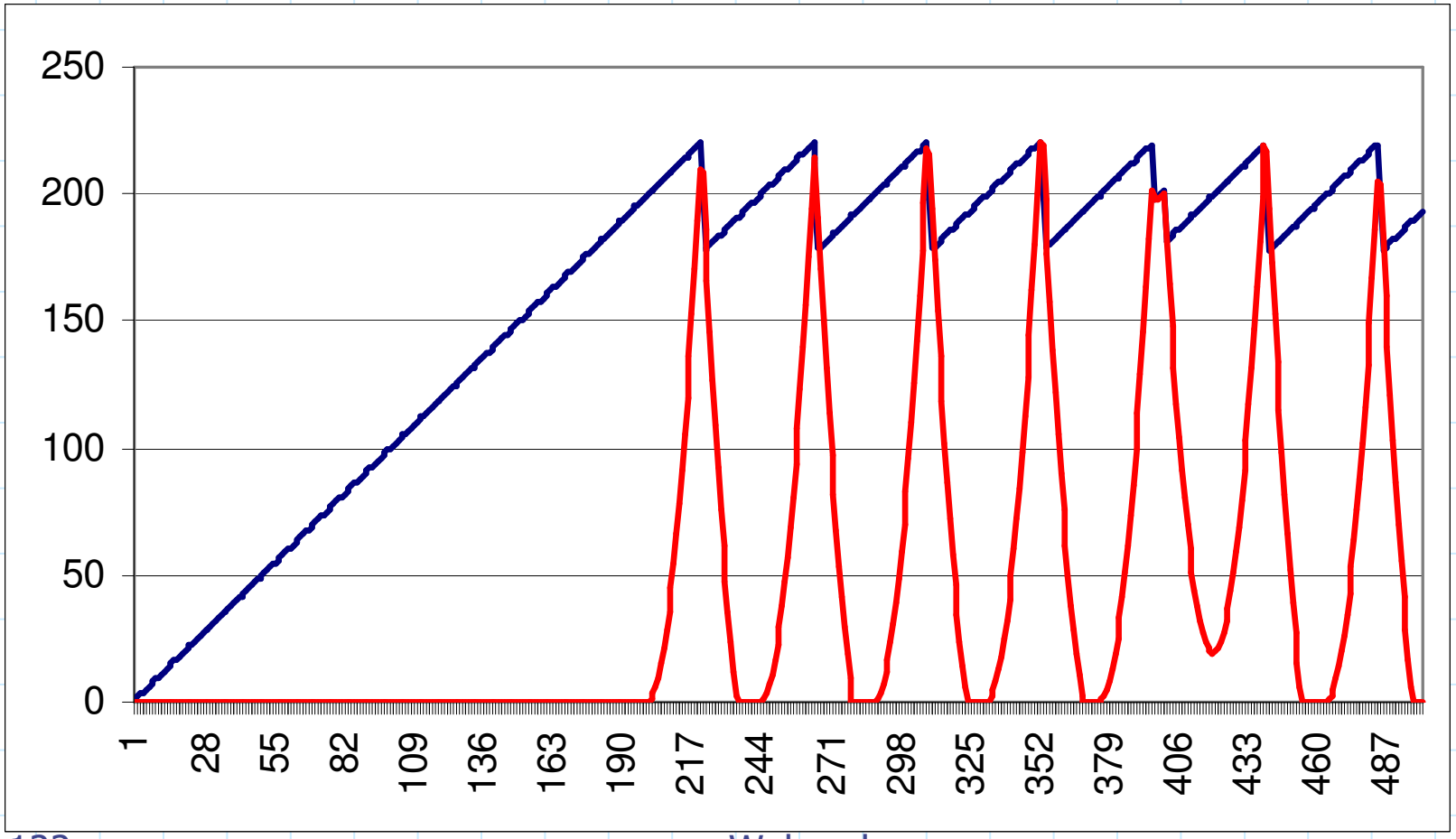
n = 20, h = 10%

q = 220KBytes

Router Buffers: Many Connections

- Example:





Router Buffers: Rule of Thumb

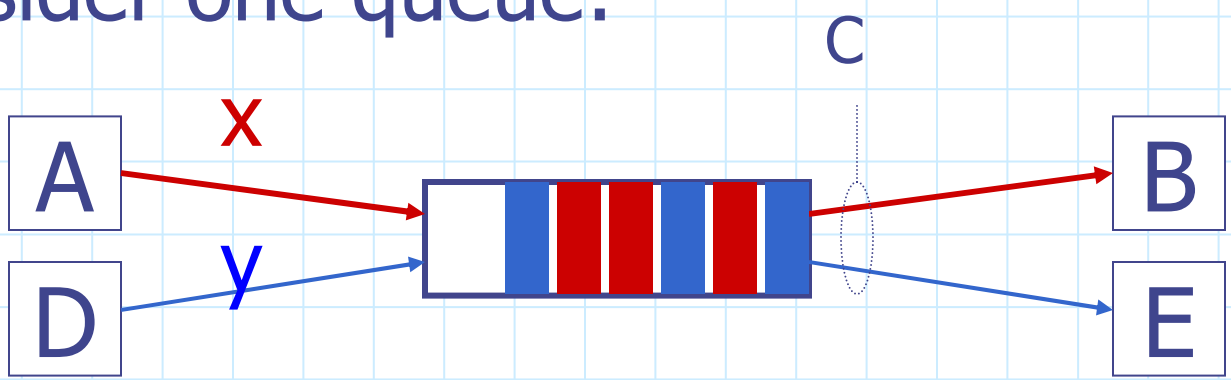
- Imagine that all connections on input port with rate R "burst" for RTT seconds (until stopped by RED)
- Router must store RxRTT for each port
- Example:
  - 40 Gbps throughput (sum of port rates)
  - RTT = 200 ms (worst case?)
  - Then storage = 8 Gbits = (about) 1 GByte
- Question: Is this reasonable?

CC2: Unfairness

- Fact:
  - TCP favors connections with short RTT
- Cause:
  - Increase rate is 1 MSS/RTT, so that it is faster for connections with small RTT
  - Recall our discussion of "Cheating RTT"
- It is quite possible for a connection to get only a few percent of its fair share
- Solutions:
  - Modify TCP to increase in proportion to RTT? Problem: Estimate of RTT is noisy
  - TCP Vegas (see next)

CC2: TCP Vegas

- Consider one queue:



Assume both connections have the same backlog
Then they have the same throughput …

Vegas Algorithm:
  Estimate backlog by Q = OUT – rate.RTT
  If Q > 3 MSS, then slow down; otherwise, speed up

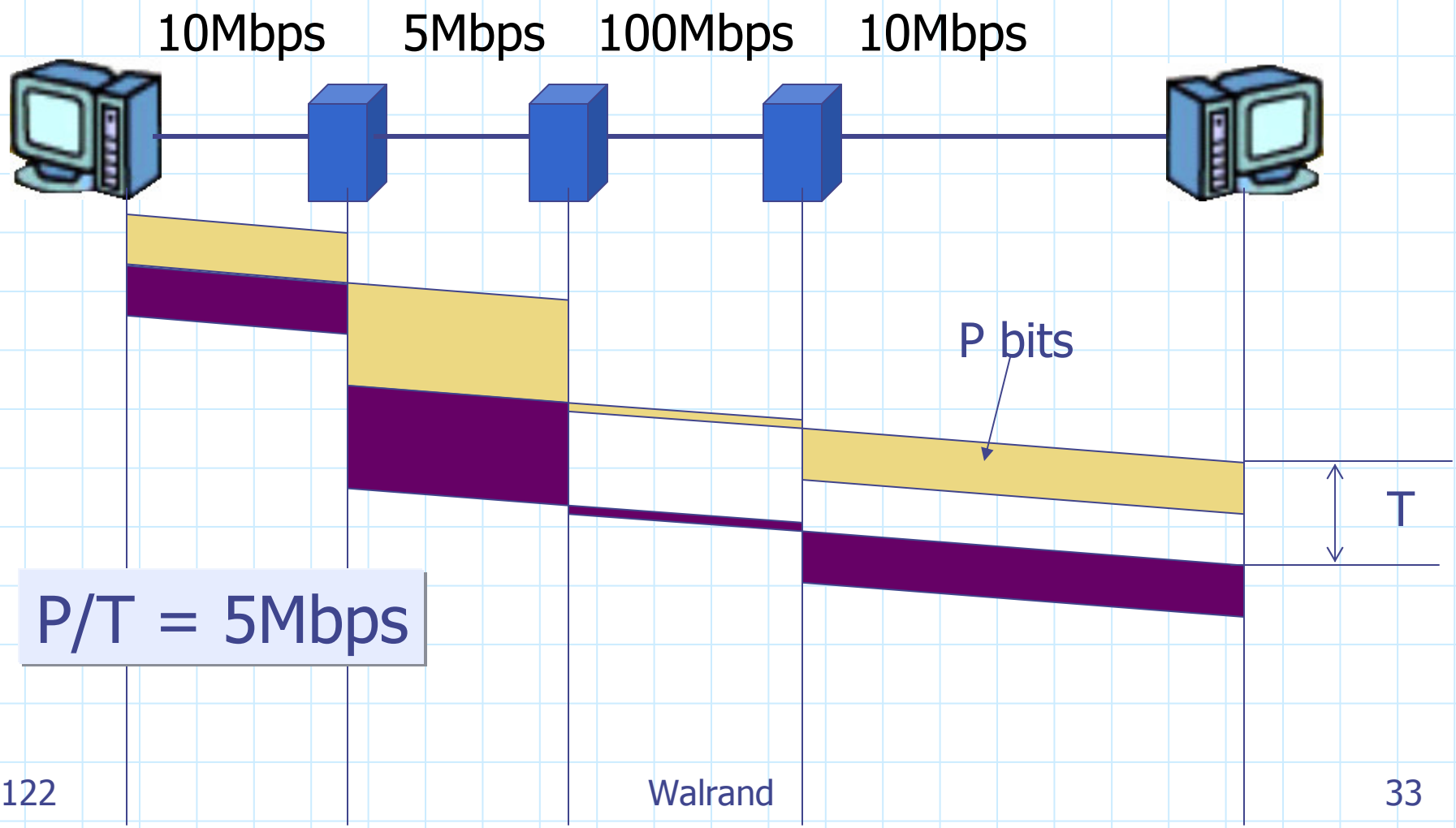
Problem: Reno clobbers Vegas (unlike in real life)

CC2: Fast Links

- Imagine a 10 Gbps link
- Assume MSS = 10kbits, RTT = 100 ms
- Slow Start:
  - After n RTT, window = 2^{n-1}10kbits, so that the rate is 2^{n-1}100kbps
  - The rate reaches 10Gbps when $2^{n-1} = 10^5$, i.e., when n = 17, which takes about 2 seconds
- CA:
  - If rate must increase by 5 Gbps after 3DA, this takes an increase of k MSS where k.10kbits/0.1 = 5Gbps →k = 5.10^4, which takes forever
- Some proposed solutions: Vegas, probe

CC2: Fast Links - Probe

- Probe for bandwidth of slowest link
- Example: Packet Pair:



P/T = 5Mbps