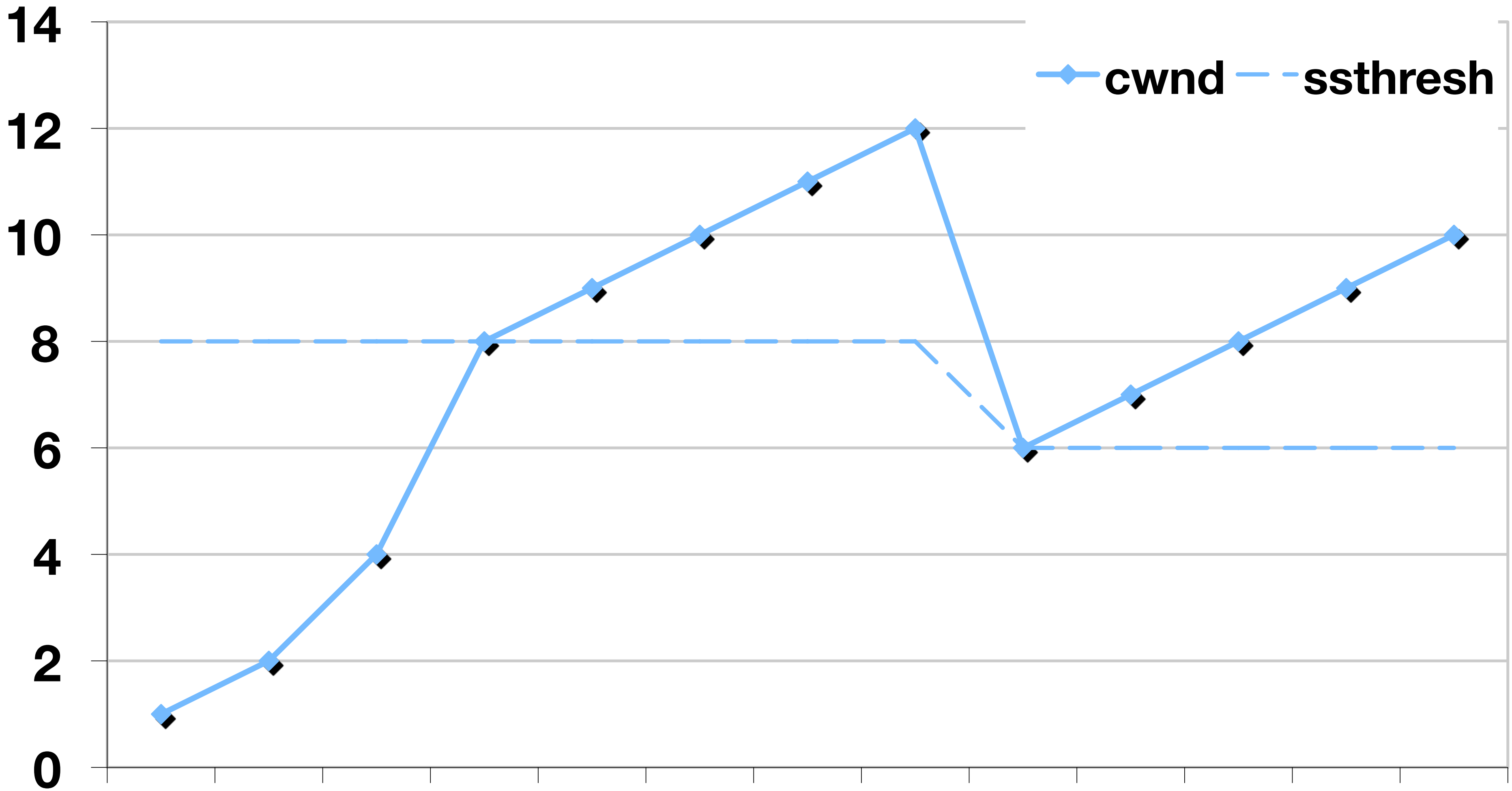
TCP

Reno, NewReno, SACK

TCP Reno


14
12
10
8
6
4
2
0
cwnd
ssthresh

new ack:

```
if (cwnd < sstresh)
   cwnd += 1
else
   cwnd += 1/cwnd
```

timeout:

retransmit 1st unacked

ssthresh = cwnd/2

cwnd = 1

3^{rd} duplicate ACK:

fast retransmission

(ie, retransmit 1^{st} unack)

fast recovery

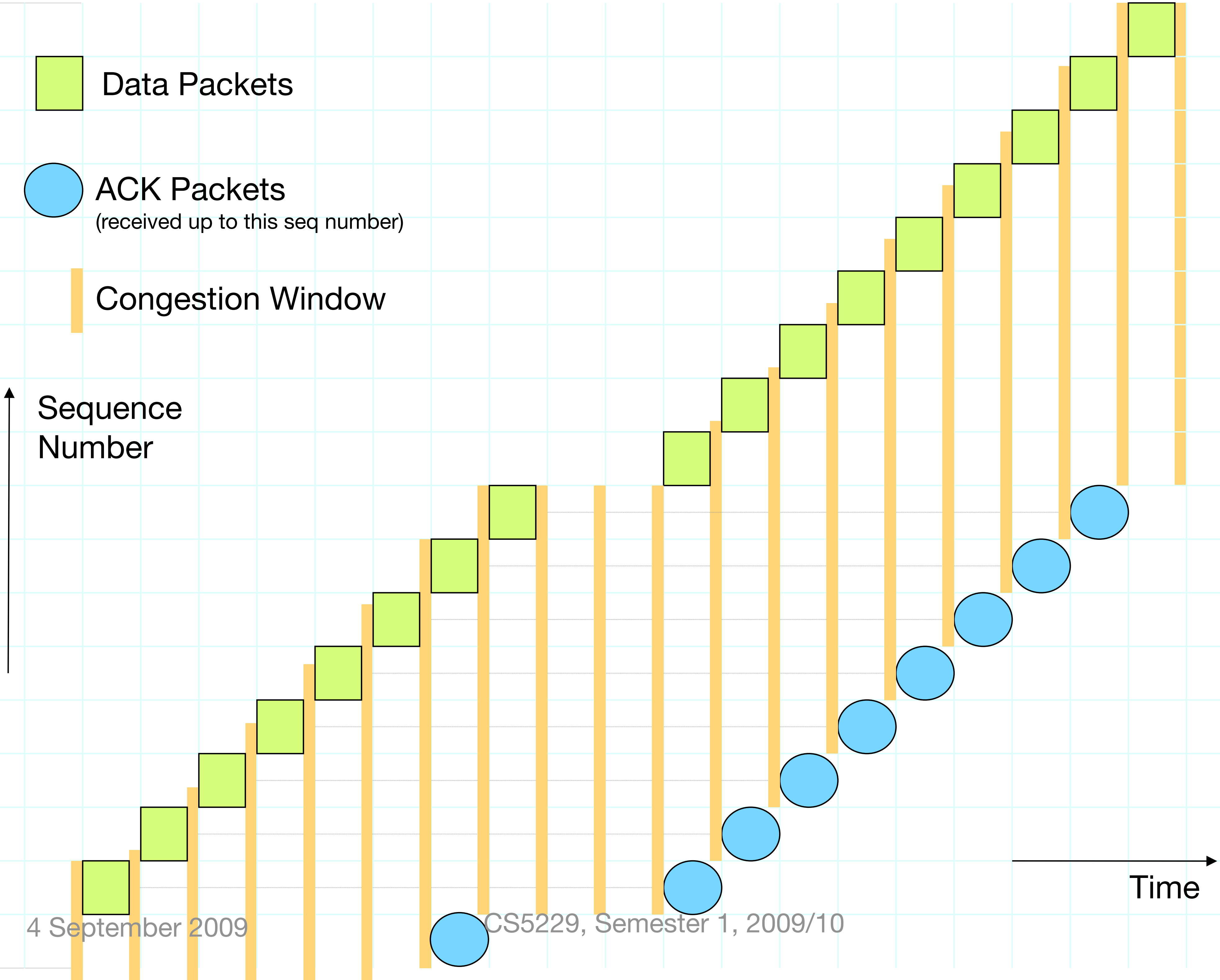
(details today)

ssthresh = cwnd = cwnd/2

TCP's rule

send more packets if
L + cwnd > H

[L .. H-1] are outstanding packets



Data Packets
ACK Packets
(received up to this seq number)
Congestion Window
Sequence
Number
Time

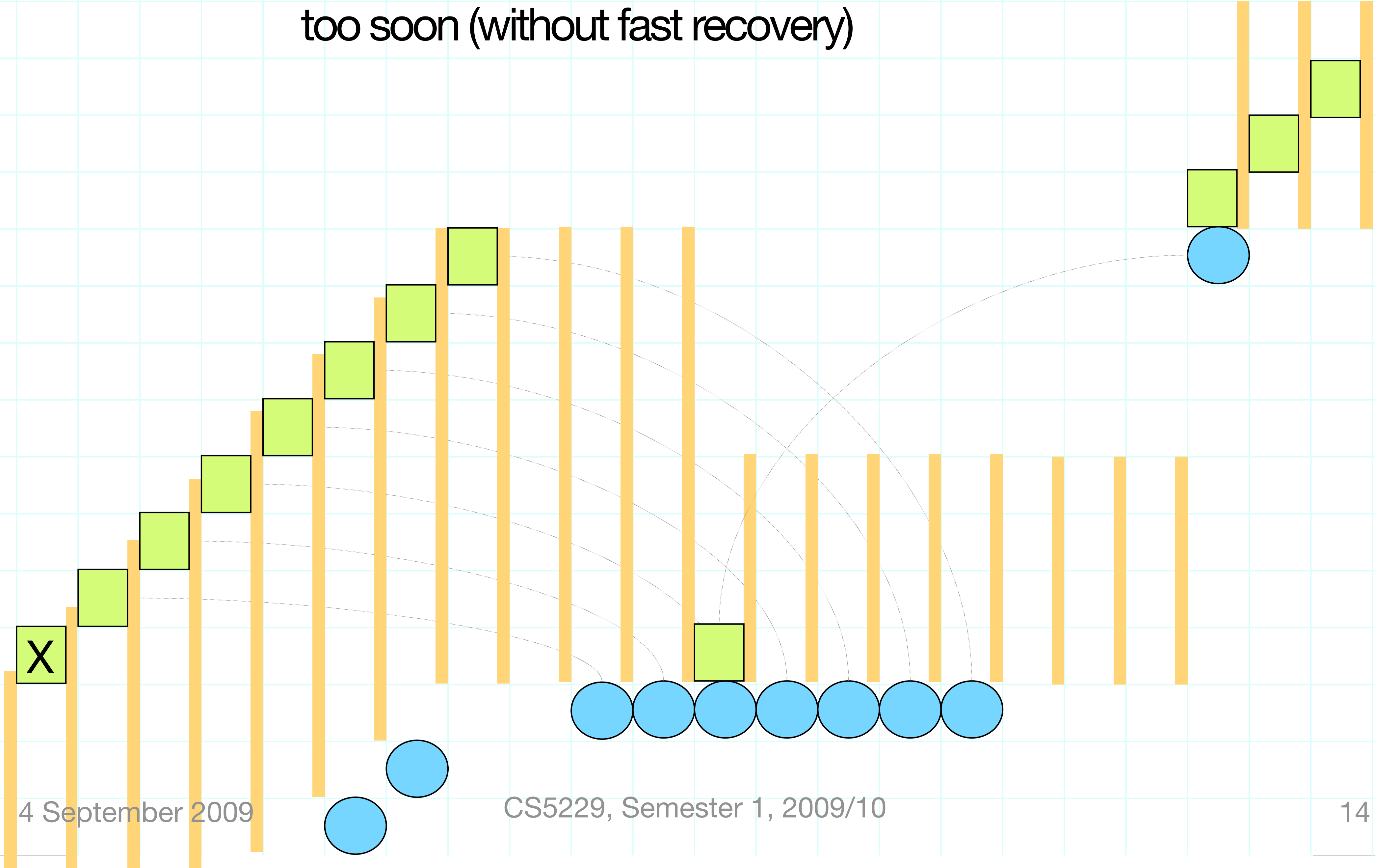
3rd dup ack:

retransmit 1st unacked

ssthresh = cwnd/2

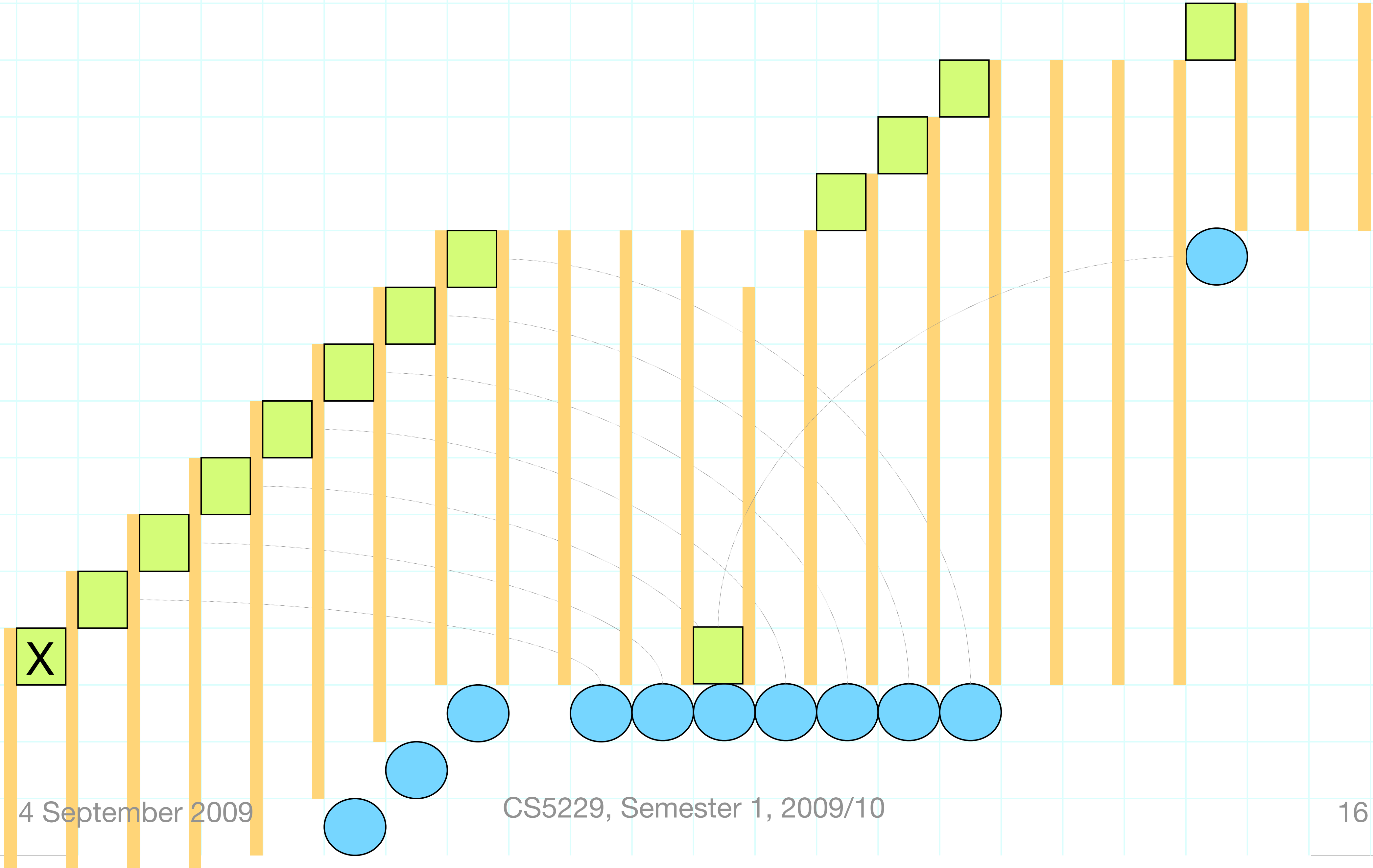
cwnd = cwnd/2

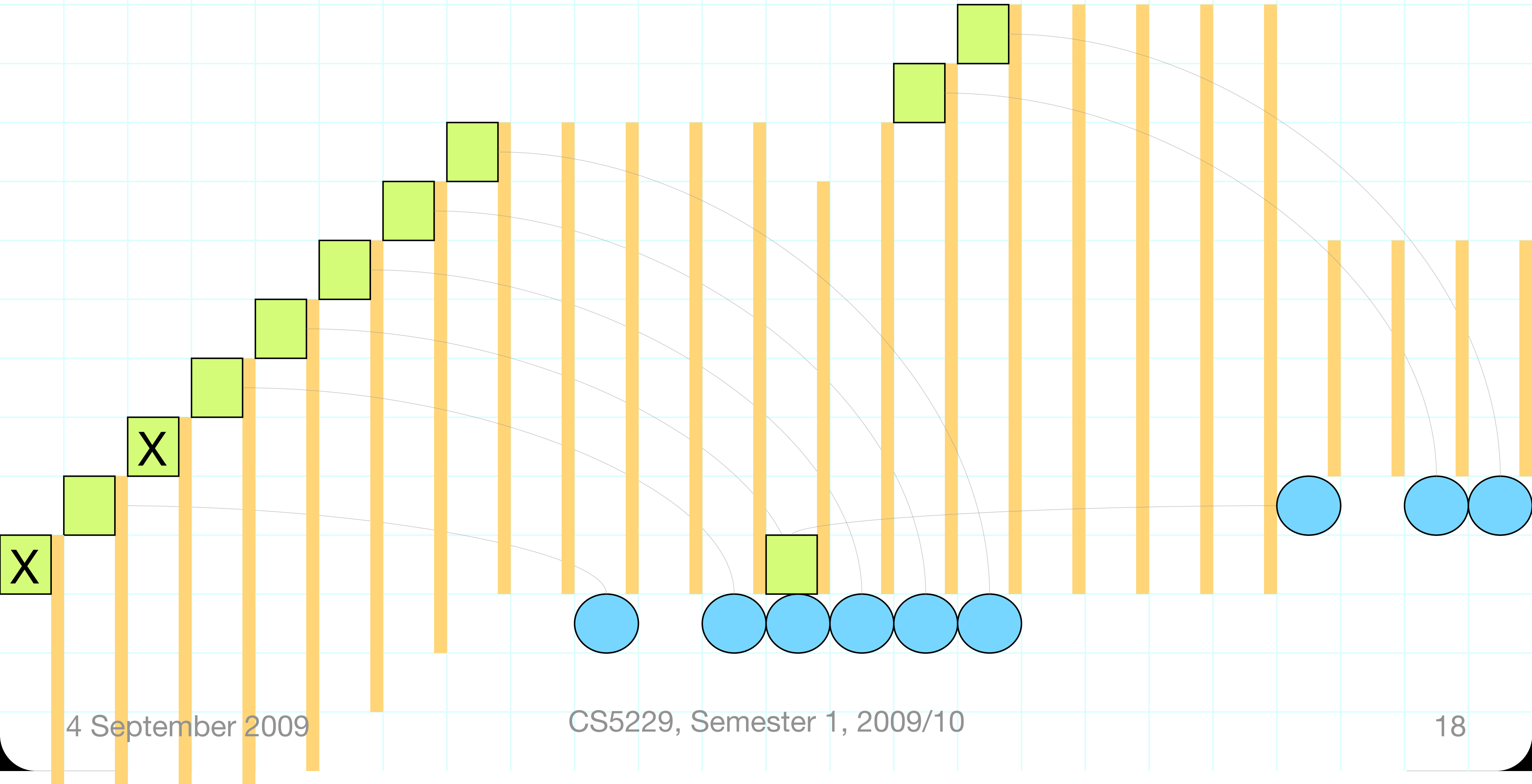
Incorrectly clamping the congestion window too soon (without fast recovery)



fast recovery:

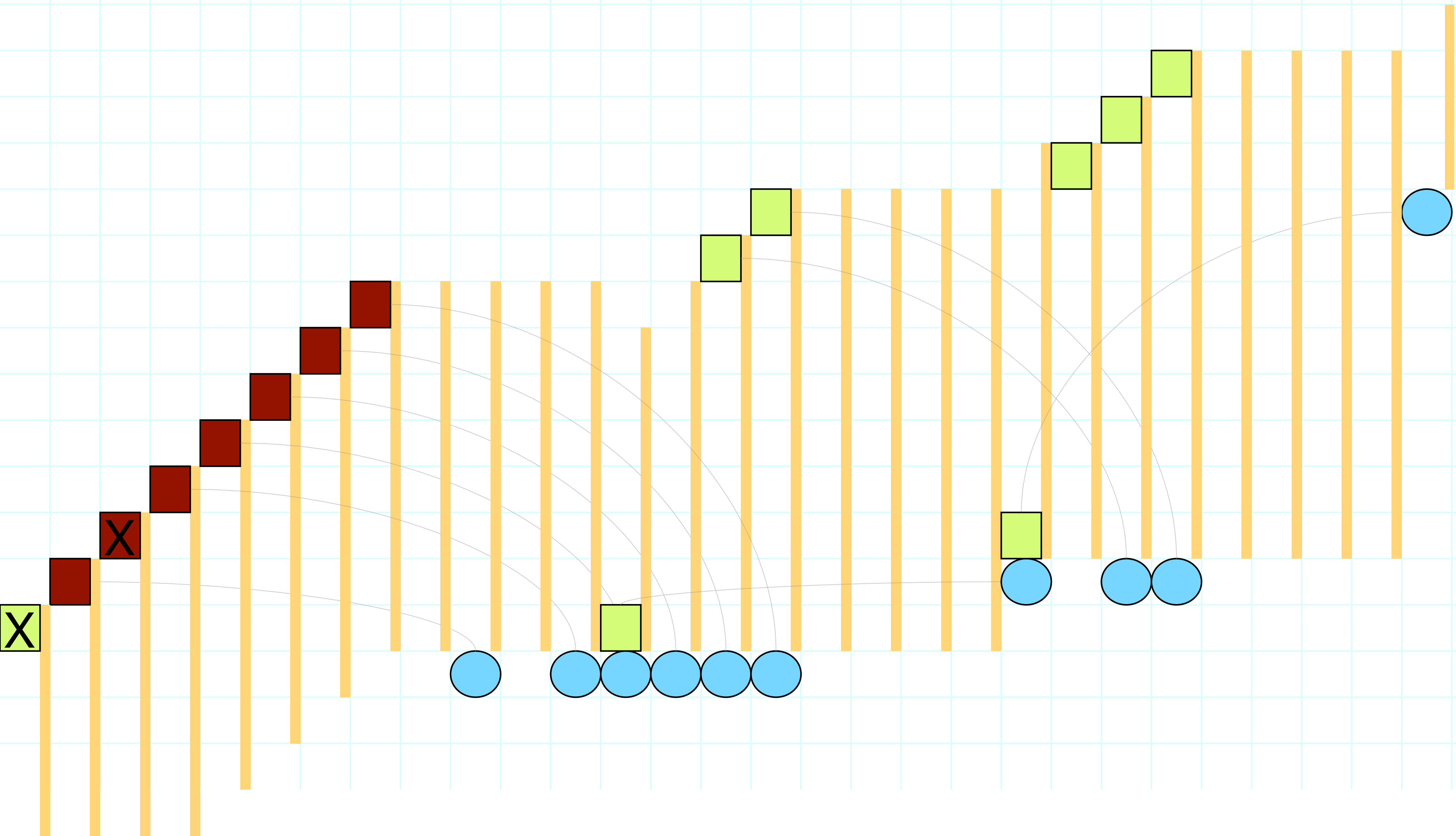
keep the pipe occupied


X


X
X

TCP Reno timeout with multiple losses in a window

TCP NewReno


X
X

3rd dup ack:

retransmit 1st unacked

ssthresh = cwnd/2

cwnd = cwnd/2 + 3

remember highest 

"complete" ack:

(all are acked)

cwnd = ssthresh

“partial” ack:

(acknowledge n packets)

retransmit

cwnd = cwnd - n + 1

Note: RFC2581/RFC2582 give the accurate/gory details. Simplified version is presented here (eg. cwnd vs FlightSize, update of cwnd upon partial ACK).

What does a dup ACK tell us?

“Coarse Feedback”

TCP SACK

Use TCP header options to report received segments.

SACK Blocks:

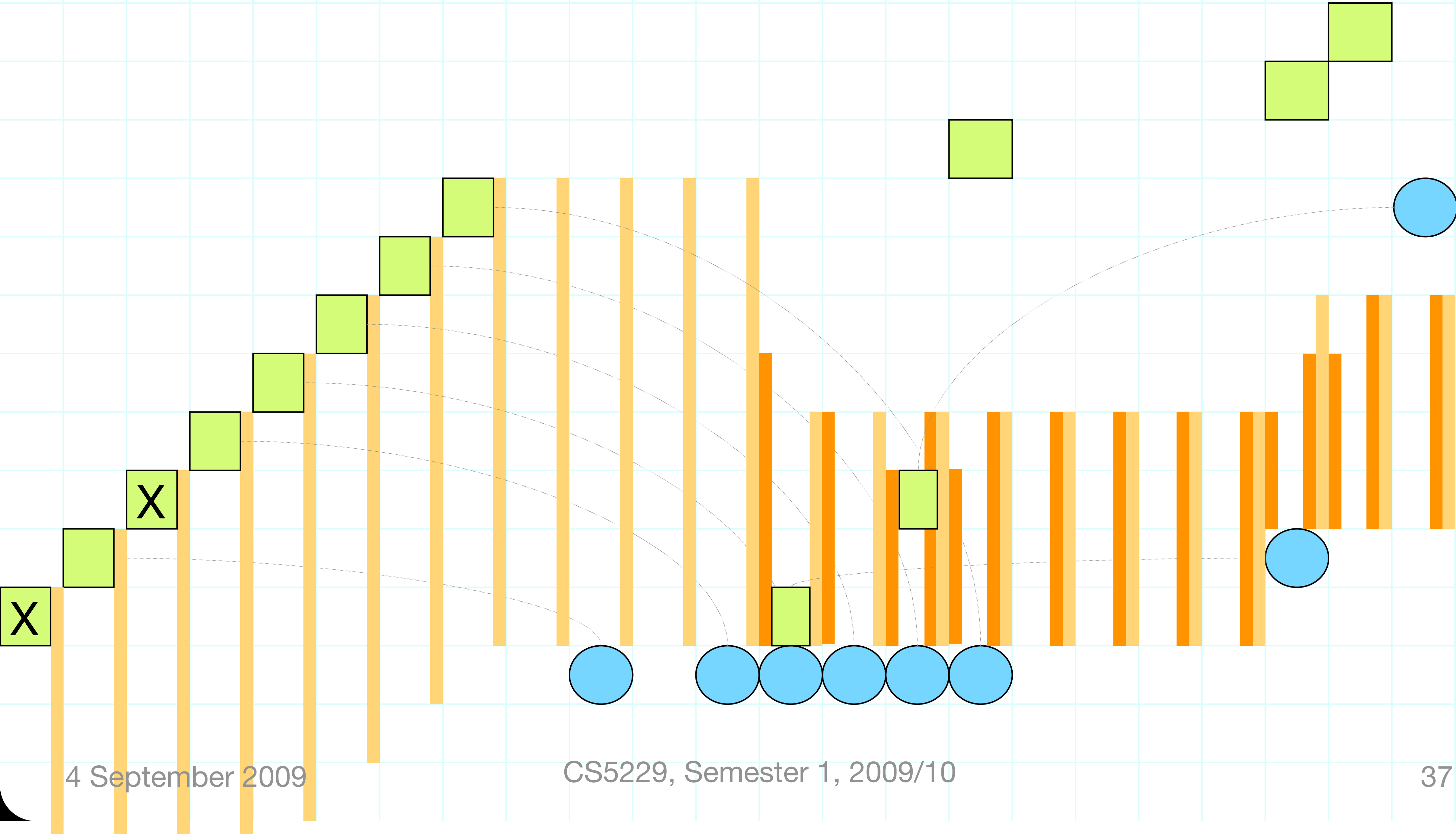
1st block - report most recently received segments

subsequent blocks - repeat most recent previous blocks

pipe: num of outstanding packets in the pipe.

send only if pipe < cwnd

scoreboard: which packets have been received?


X
X

3rd dup ack:

pipe = cwnd - 3

retransmit 1st unacked

ssthresh = cwnd/2

cwnd = cwnd/2 ~~+ 3~~

subsequent dup ack:

~~cwnd++~~

pipe--

if pipe < cwnd

send packet, pipe++

“partial” ack:

~~retransmit~~

~~cwnd = cwnd - n + 1~~

pipe -= 2

if pipe < cwnd

 send packet, pipe++

Idea of SACK:

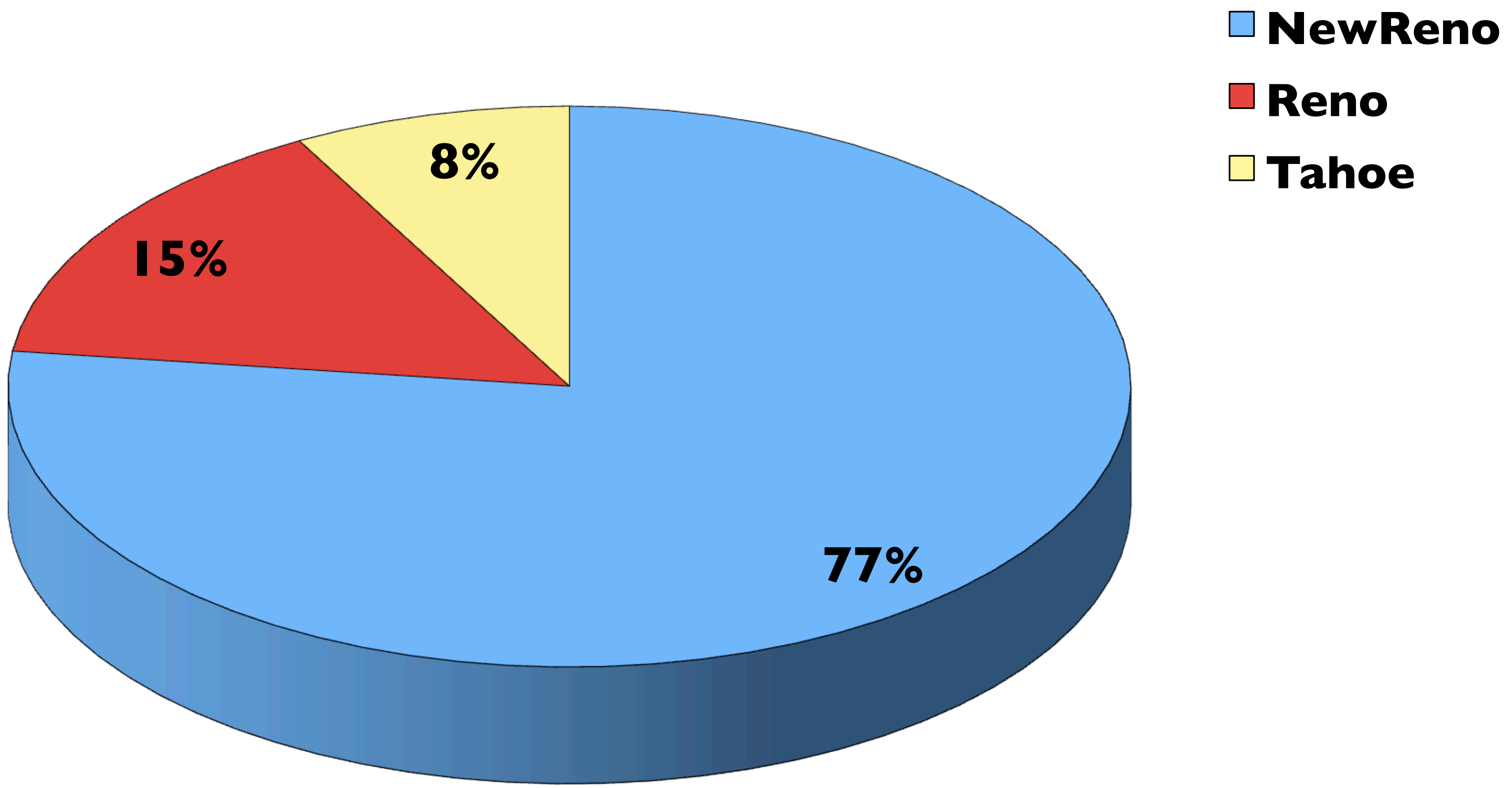
Which packet has left the network?
Where is the gap?

Decouple *when* to send and *what* to send.

TCP SACK recovers faster than NewReno with multiple losses in a window.

Deployment

Feb 2004



70%

SACK capable

TCP BIC/CUBIC

Linux 2.6.x

Compound **TCP**

MS Windows Vista

TFRC

equation-based
congestion control

Not everyone uses TCP

UDP:

Media streaming
Gaming
VoIP

Why not congestion controlled?

1. UDP has low delay, no need full reliability. UDP is not congestion controlled.

Why not congestion controlled?

2. No incentive. OTOH, there are incentives NOT to use congestion control.

- Flash Networks BoosterWare: "For the Internet community at large, NetBooster exploits the capacity of the modem to maintain a constant data flow at its maximum rated speed, regardless of the network traffic load." (Flash Networks Press Release)

  From their White Paper on The BoosterWare Advantage: Enhancing TCP/IP: "BoosterWare, by contrast, abandons the effort to optimize the window size (a key source of bottlenecks) during transmissions; instead, window sizes are fixed according to pre-defined parameters negotiated between the client and the server once a connection has been established. BoosterWare can be viewed as a reliable, "no overhead" UDP (user datagram protocol)..."

- RUN Inc. ("RUN Inc. has found a way to squeeze more bandwidth out of existing TCP/IP networks without changing the network protocols or the applications that run over them.... In field tests over the Internet, runTCP has accelerated data transfers by as much four times." - PC Week Online, Sept. 4, 1997.)

- Sitara Networks Inc. ("Everyone talks about the "World Wide Wait", but no one does anything about it."). As discussed in IP Acceleration Software: Torquing Up TCP/IP, DataCommunications, January 1998: "Speedseeker can selectively suspend the TCP/IP congestion control mechanism when sending audio and video." See About Sitara in the News.

- RealAudio. "RealAudio 3.0 encoding algorithms have four different fixed data rates which can be used depending on the bandwidth requirements." (Audio Bandwidth)

- Jae Chung, Yali Zhu, and Mark Claypool, FairPlayer or FoulPlayer?--Head to Head Performance of RealPlayer Streaming Video Over UDP versus TCP, Technical Report N. WPI-CS-tr-02-17, Worchester Polytechnic Institute Computer Science Department, May, 2002.
  "In times of congestion, most RealVideo over UDP does respond to Internet congestion by reducing the application layer encoding rate, often achieving a TCP-Friendly rate. In times of severe congestion, RealVideo over UDP gets a proportionately larger share of the available bandwidth than does the same video over TCP."

Sally Floyd, http://www.icir.org/floyd/tcp_unfriendly.html

“Unresponsive Flows”

Bad: lead to unfairness and congestion collapse.

Unfairness: unresponsive flows consume more bandwidth than congestion controlled flows.

Congestion Collapse:

wasting bandwidth by sending packets that will be dropped

Today:
a TCP-friendly unreliable protocol

Idea:

send at a rate that a TCP flow would send we can do the AIMD-thing at the source, or

$$B_{TCP} = \frac{MSS}{RTT\sqrt{\frac{2p}{3}} + min(1, 3\sqrt{\frac{3p}{8}})T_O p(1 + 32p^2)}$$

equation-based congestion control

steady

fair

responsive

how to determine

RTT

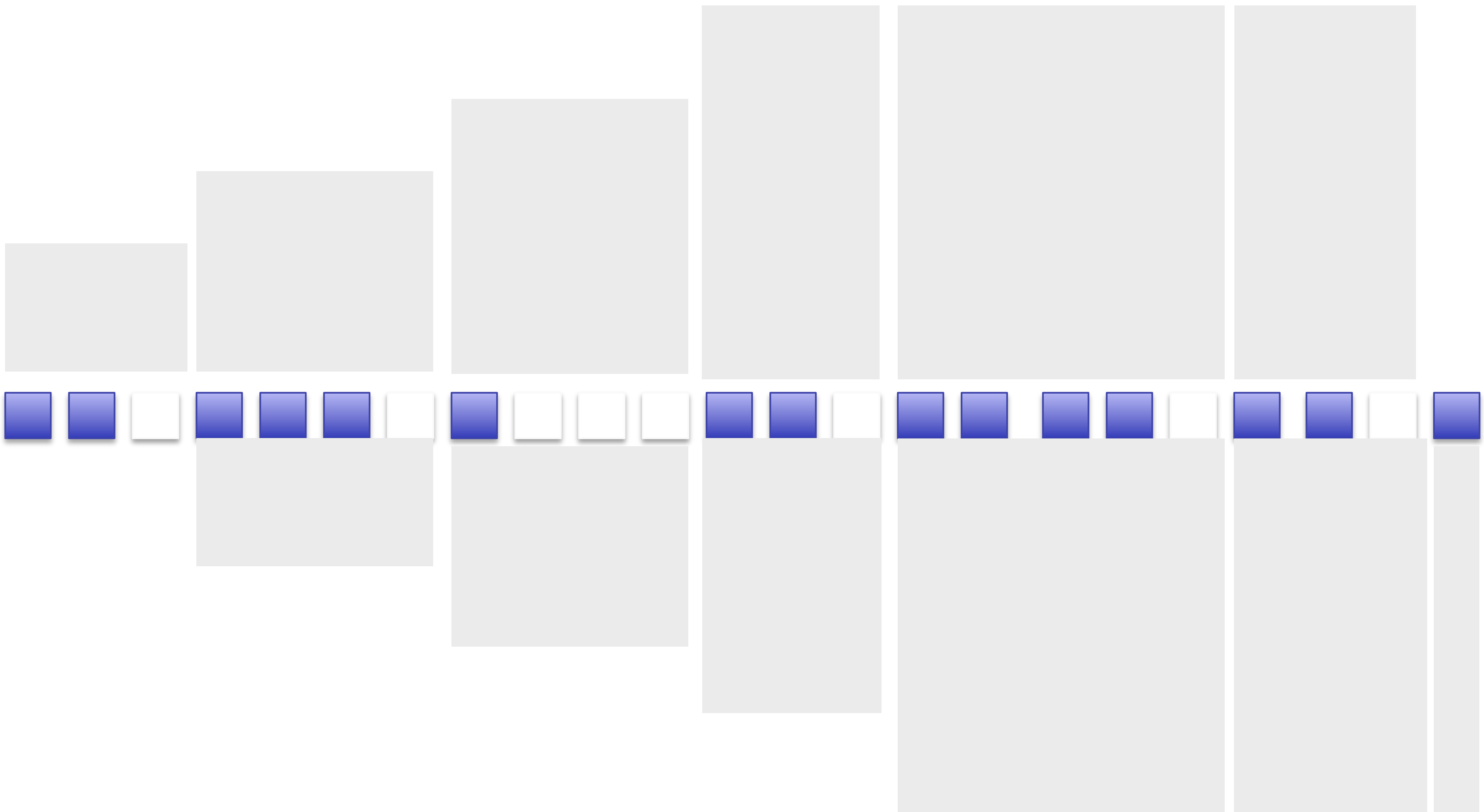
t_{RTO}

p

p is not packet loss rate

but

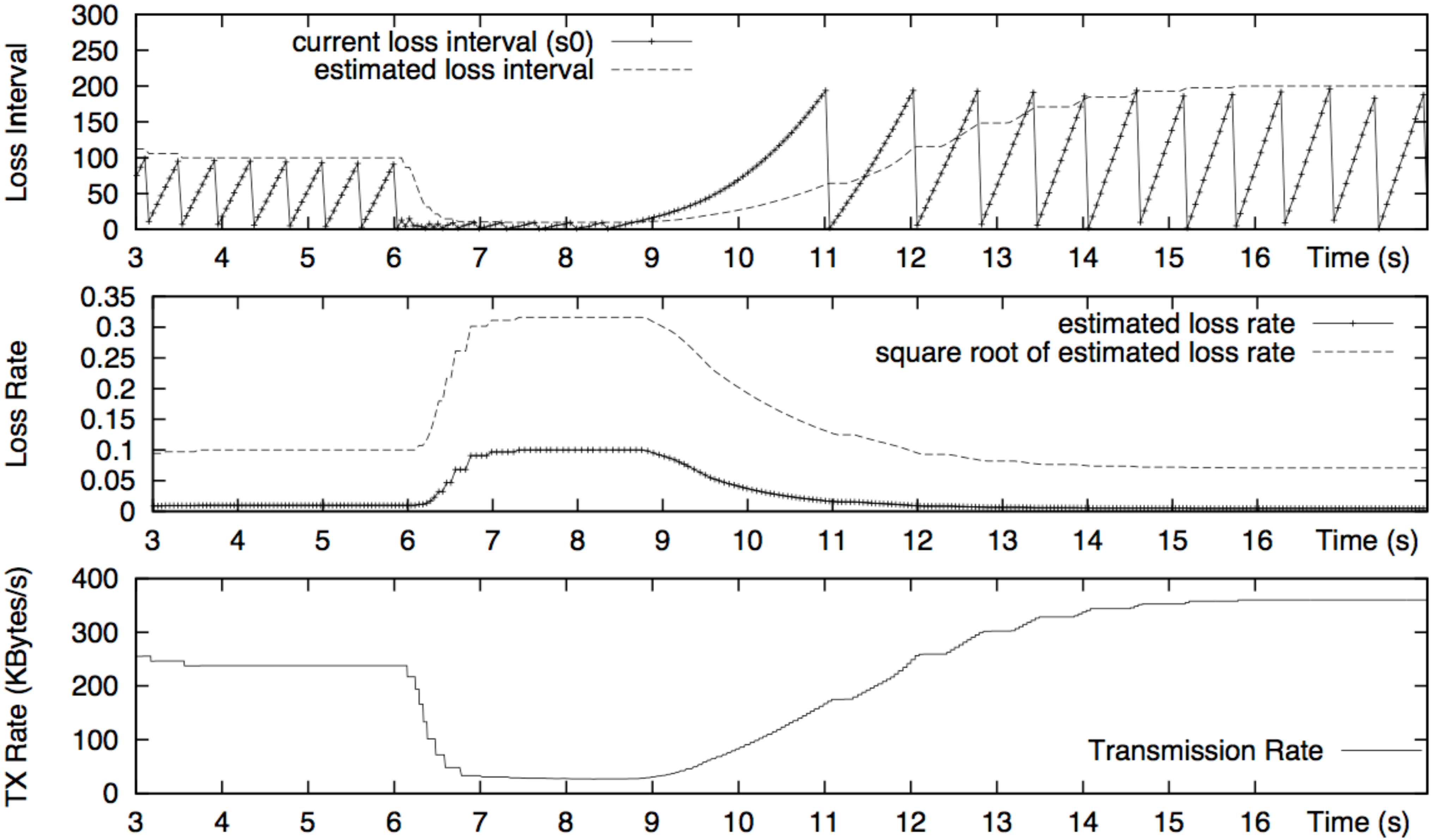
loss event rate



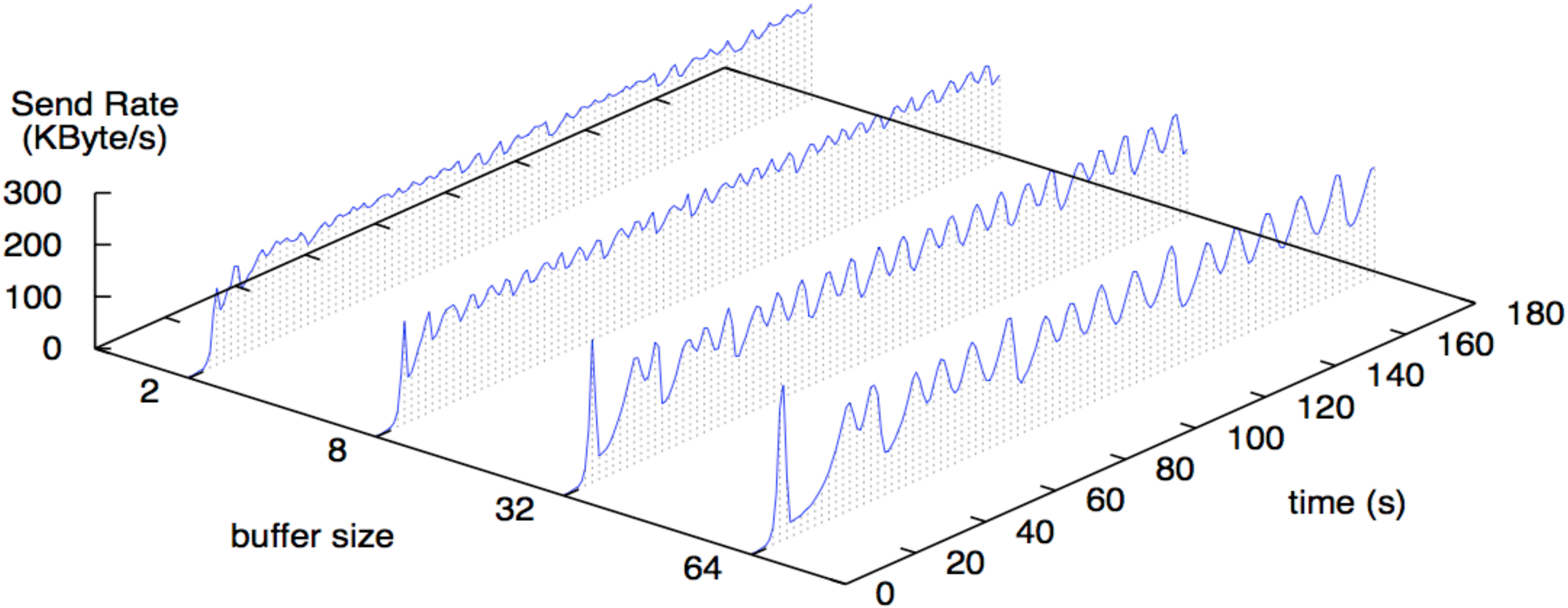
$$\hat{s}_{(0,n-1)} = \frac{\sum_{i=0}^{n-1} w_{i+1} s_i}{\sum_{i=1}^{n} w_i}$$

$$\hat{s}_{(1,n)} = \frac{\sum_{i=1}^{n} w_{i+1} s_i}{\sum_{i=1}^{n} w_i}$$

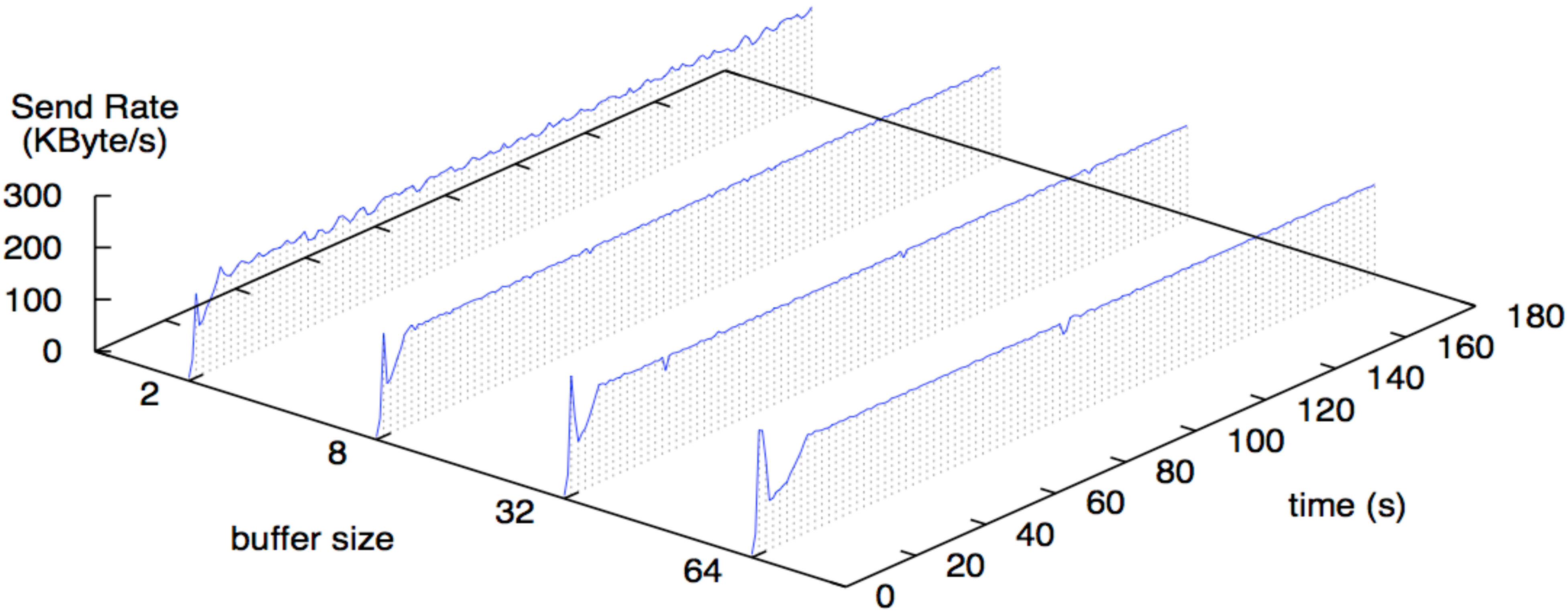
RTT can fluctuates



Loss Interval
current loss interval (s0)
estimated loss interval
Time (s)
Loss Rate
estimated loss rate
square root of estimated loss rate
Time (s)
TX Rate (KBytes/s)
Transmission Rate
Time (s)


Send Rate
(KByte/s)
300
200
100
0
2
8
32
64
buffer size
0
20
40
60
80
100
120
140
160
180
time (s)

$$t_{\text{inter-packet}} = \frac{MSS}{B_{TCP}}$$


Send Rate
(KByte/s)
300
200
100
0
2
8
32
64
buffer size
0
20
40
60
80
100
120
140
160
180
time (s)

how to initialize?

slow start

(until loss occur)

$$T_{now} = \min(2T_{prev}, 2T_{recv})$$

no loss history, how?

solve p given T, RTT

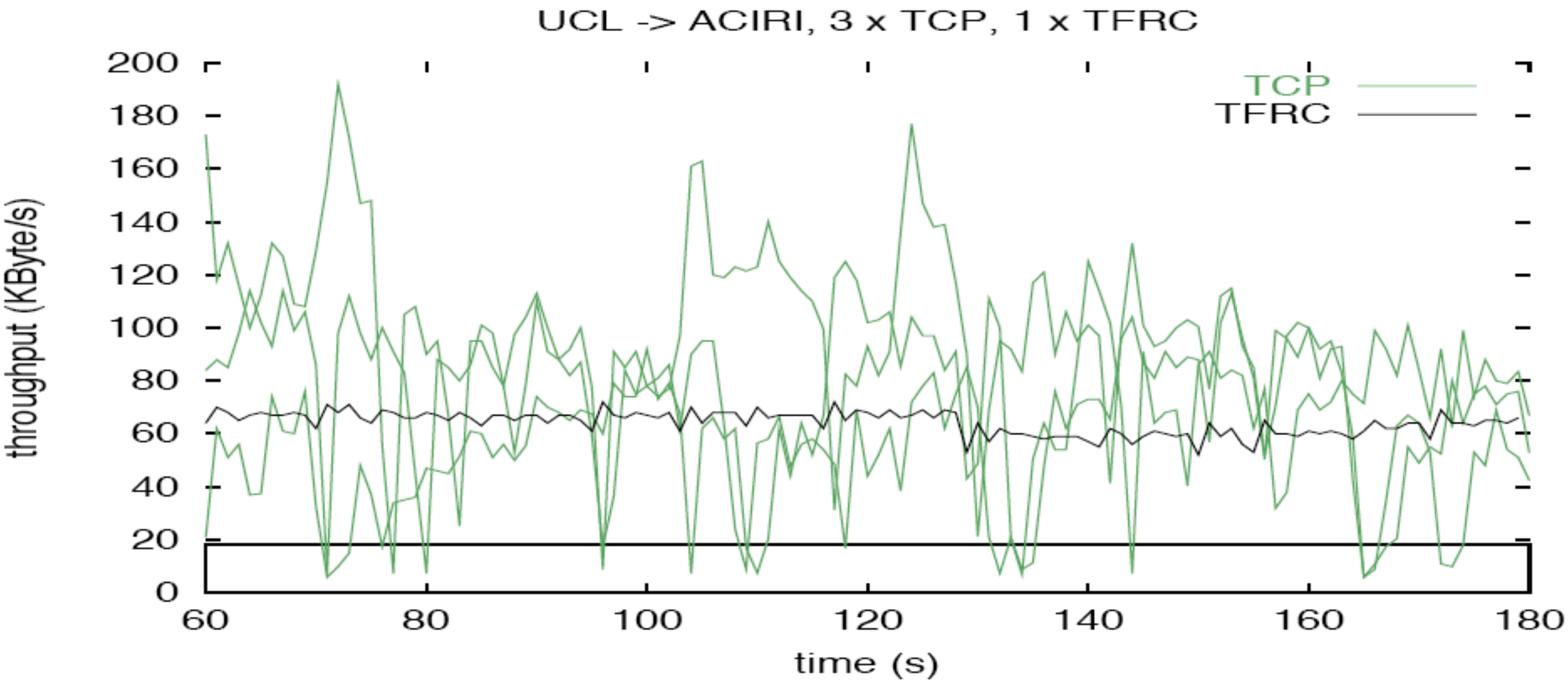


Figure 15: Three TCP flows and one TFRC flow over the Internet.

Equation-Based Congestion Control for Unicast Applications*

Sally Floyd, Mark Handley
AT&T Center for Internet Research at ICSI (ACIRI)

Jitendra Padhye
University of Massachusetts at Amherst

Jörg Widmer
International Computer Science Institute (ICSI)

TFRC is now part of DCCP