CN Lab (17CSL57)

Exp 03: TCP Congestion Window

Dr. Ram P Rustagi
Dept of CSE, KSIT
KRP-KSGI
rprustagi@ksit.edu.in

Ex03 Resources

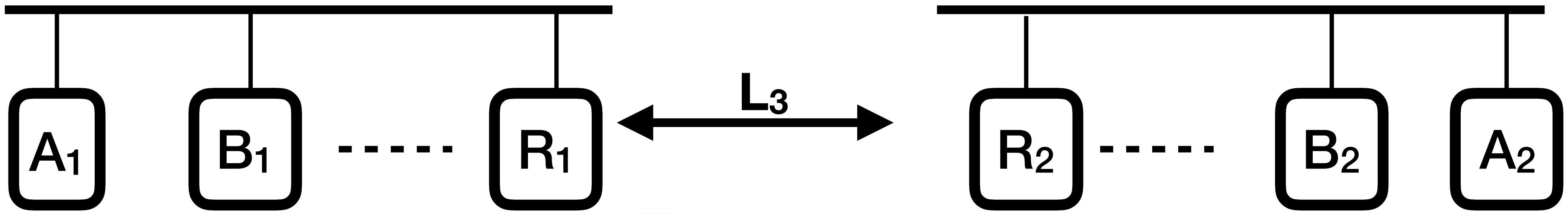
- References:
  - Local area networks
    - https://www.isi.edu/nsnam/ns/doc/node143.html
  - Transport layer : UDP and TCP agents
    - https://www.isi.edu/nsnam/ns/doc/node383.html
    - https://www.isi.edu/nsnam/ns/doc/node387.html
  - TCP Tahoe(Default) and Reno
  - TCP traffic flow
  - http://www.netlab.tkk.fi/opetus/s38148/s02/sim_harj/ns2_exers.pdf

Lab03 Program

- Program 03
  - Implement an Ethernet LAN using `n` nodes and set multiple traffic nodes and plot congestion window for different source / destination

Lab03: Design Parameters

- Design of Lab03: Basic Topology
  - Topology: A LAN can contain many hosts,
  - Two LANs are connected via WAN link



Topology

- LAN Parameters (configure different values)
  - Bandwidth: 100Mbps (or 10Mbps)
  - Delay: 0.001ms
- WAN Parameters (configure different values)
  - Bandwidth: 1Mbps
  - Delay: 500ms
  - Q Limit: 10

Lab03: Design Parameters

- TCP Config parameters
  - Initial congestion window threshold (mss): 40
  - TCP congestion mechanism: Reno, New Reno, Tahoe
  - TCP connection duration : 100s

  - TCP Payload size: 1460 byte =(1500-20-20)
    - 20 bytes of TCP header
    - 20 bytes of IP header

TCP congestion control

- Approach by sender
  - –At start, sender limits the rate of traffic
  - –When perceives no congestion
    - Increases the sending rate
  - –When perceives congestion
    - Decrease the sending rate
- Questions:
  - –How does sender limits the rate?
  - –How does it perceive congestion or no-congestion?
  - –What is the algorithm to change the sending rate?
    - Increase as well as decrease

TCP congestion control

- Consider the network when no congestion
  - Consider high delays or slow link on path
    - Acks will arrive slowly (w/o loss)
    - `cwnd` will increase/slide slowly
- Challenge for TCP Sender
  - Sending too fast will cause congestion
  - Sending too slow causes inefficient channel usage
  - How to determine the sending rate
    - Explicit coordination?
    - Distributed approach?
  - What are the guiding principles?

TCP congestion control

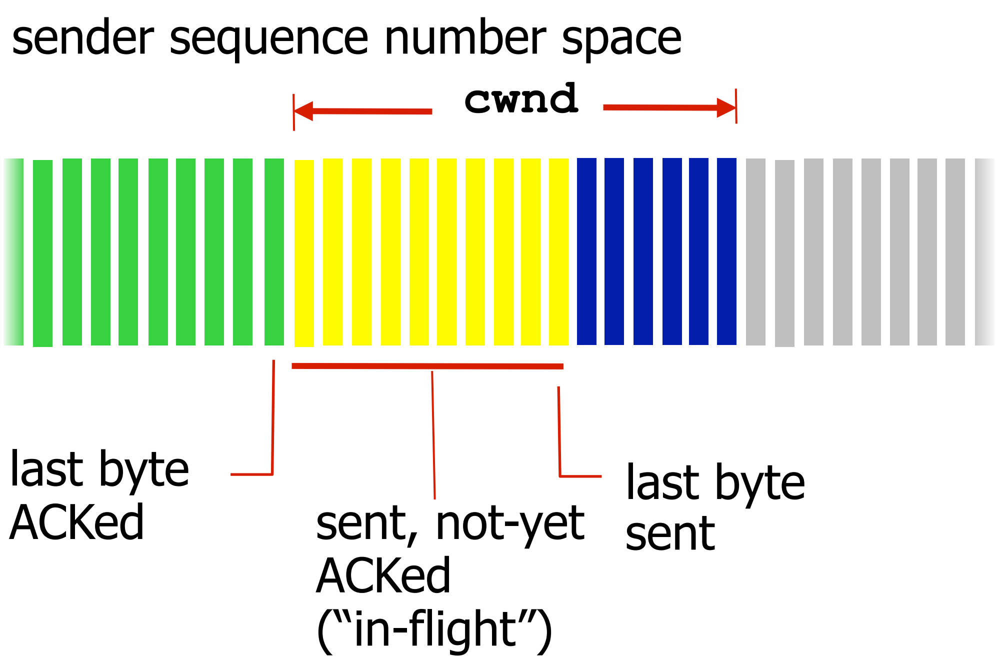
- Guiding principles to determine sending rate
- Lost segment implies congestion
  - Should result in decreased sending rate
    - How to decrease ?
- Received ack for previous unacked segment
  - Network delivered segments to receiver
    - Network is working fine
  - Sender rate can be increased
  - Congestion window can be increased
- Bandwidth probing
  - Probe the n/w for rate at which congestion begins
- Approach
  - Back off from congestion rate point
  - Start the probing process again

TCP congestion control algorithm

- RFC 5681 (algorithm by Van Jacobson)
- Three major components
  - Slow start
  - Congestion avoidance
  - Fast recovery

TCP Congestion Control: details

- Sender limits transmission:



`LastByteSent-LastByteAcked` $\leq$ `cwnd`

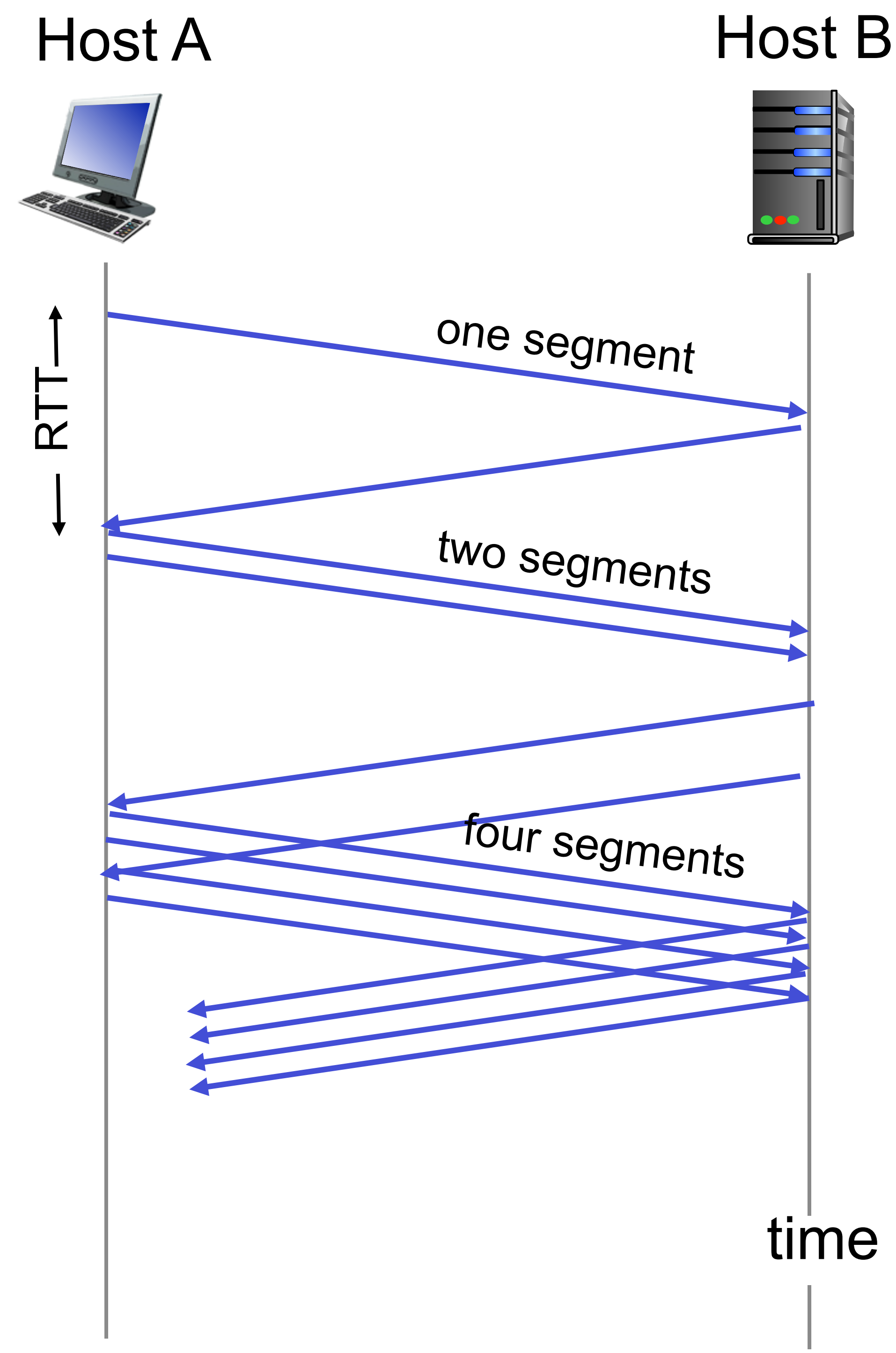
- **`cwnd`** is dynamic, function of perceived network congestion

- *TCP sending rate:*
- *Roughly:* send cwnd bytes, wait RTT for ACKS, then send more bytes

$$\text{rate} \approx \frac{\text{cwnd}}{\text{RTT}} \text{ bytes/sec}$$

TCP Slow Start

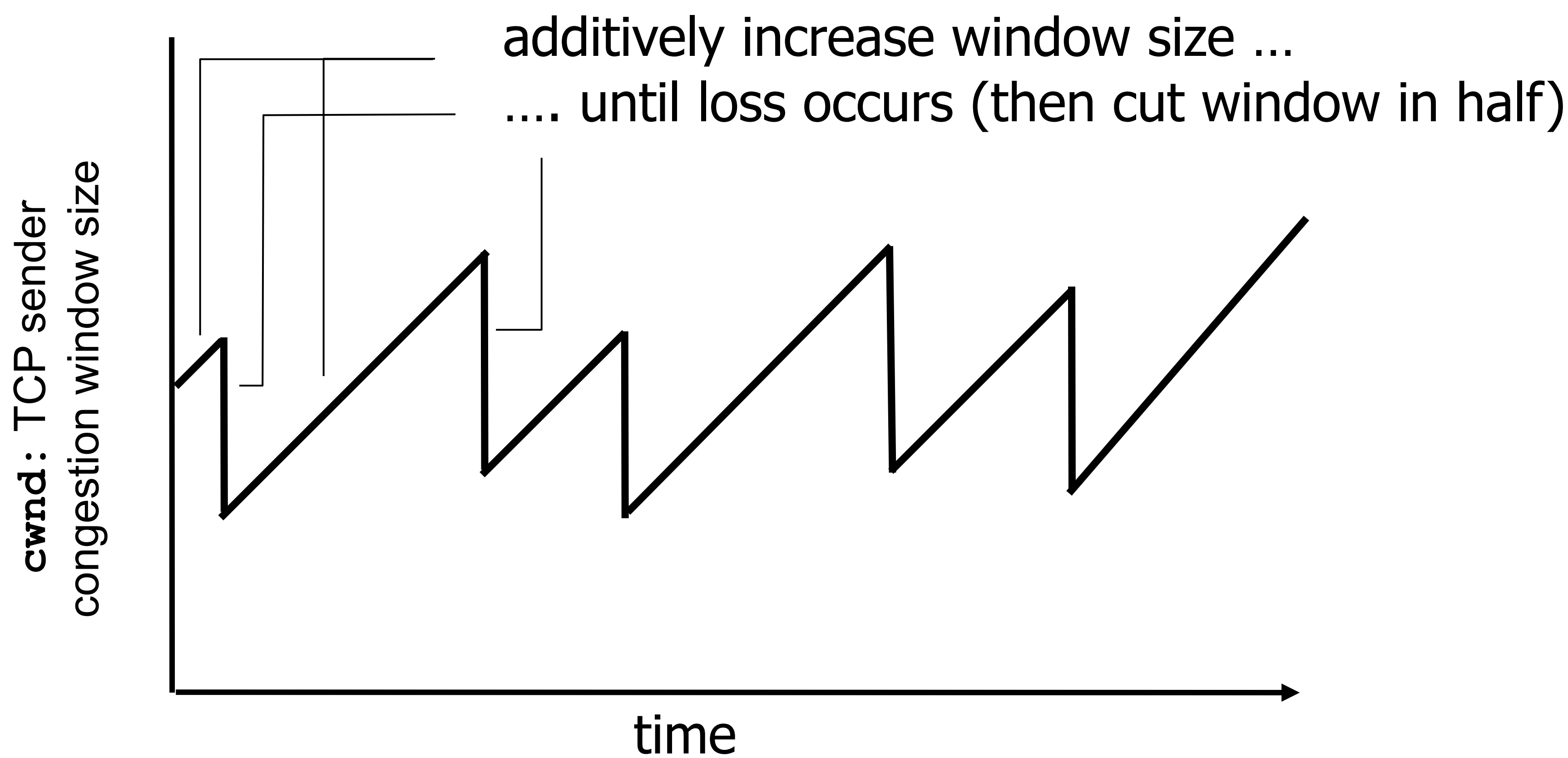
- When connection begins, increase rate exponentially until first loss event:
  - Initially **cwnd** = 1 MSS
  - Double **cwnd** every RTT
  - Done by incrementing **cwnd** for every ACK received
- *Summary:* initial rate is slow but ramps up exponentially fast



TCP congestion control:

- *Congestion Avoidance*
  - *Additive Increase Multiplicative Decrease (AIMD)*
- *Approach:* sender increases transmission rate (window size), probing for usable bandwidth, until loss occurs
- *Additive increase:* increase **`cwnd`** by 1 MSS every RTT until loss detected
- *Multiplicative decrease:* cut **`cwnd`** in half after loss

AIMD saw tooth behavior: probing for bandwidth

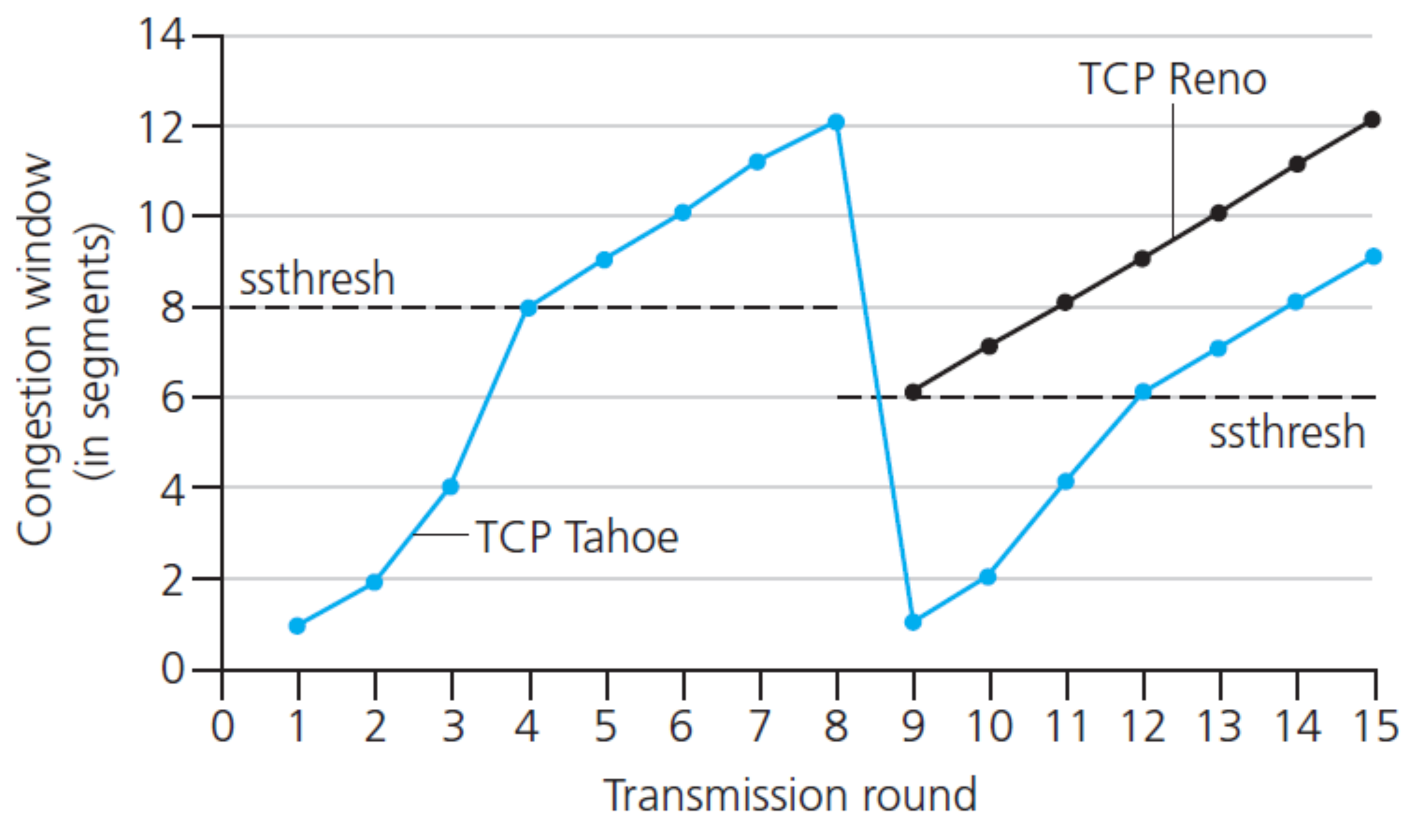


TCP: detecting, reacting to loss

- Loss indicated by timeout:
  - **cwnd** set to 1 MSS;
  - Window then grows exponentially (as in slow start) to threshold, then grows linearly
- Loss indicated by 3 duplicate ACKs: TCP RENO
  - Dup ACKs indicate network capable of delivering some segments
  - **cwnd** is cut in half window then grows linearly
    - TCP Reno
    - Add 1 to `cwnd` for each duplicate Ack
  - TCP Tahoe always sets **cwnd** to 1
    - Timeout or 3 duplicate acks

TCP: switching from slow start to

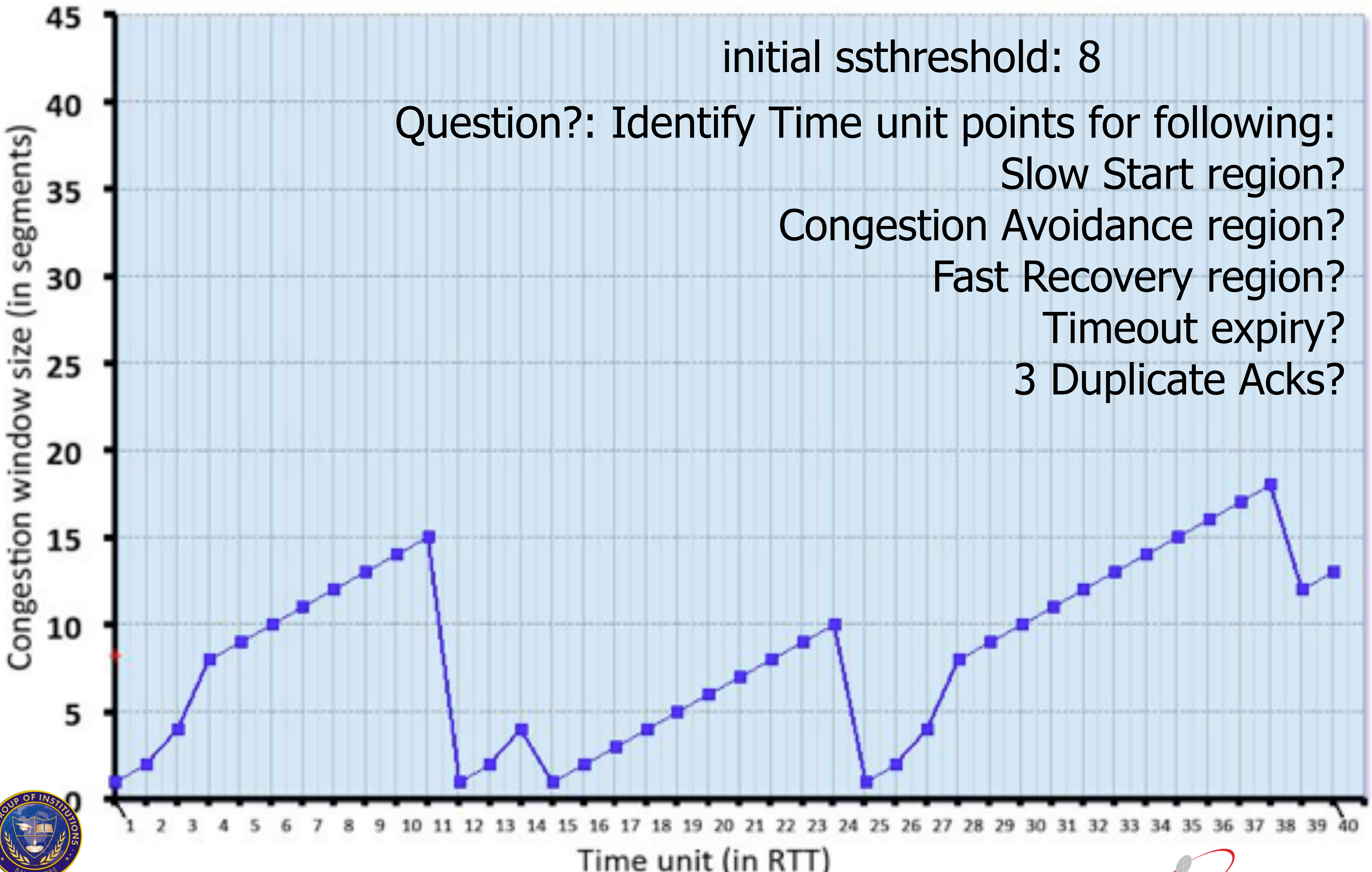
Q: When should the exponential increase switch to linear?



- **Implementation:**
- Variable **`ssthresh`**
- On loss event, **`ssthresh`** is set to 1/2 of **`cwnd`** just before loss event

Interactive activity:
http://gaia.cs.umass.edu/kurose_ross/interactive/tcp_evolution.php

TCP Congestion Control: Example



initial ssthreshold: 8

Question?: Identify Time unit points for following:

Slow Start region?

Congestion Avoidance region?

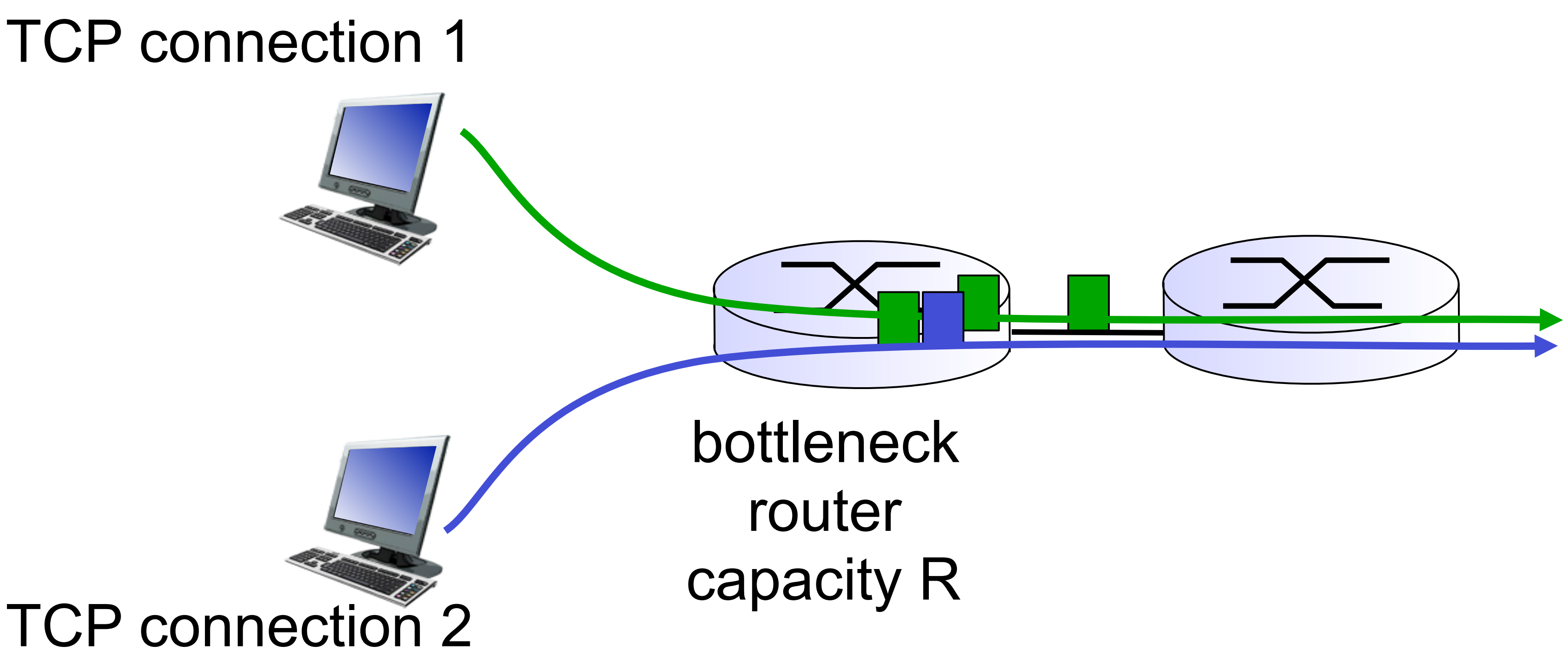
Fast Recovery region?

Timeout expiry?

3 Duplicate Acks?

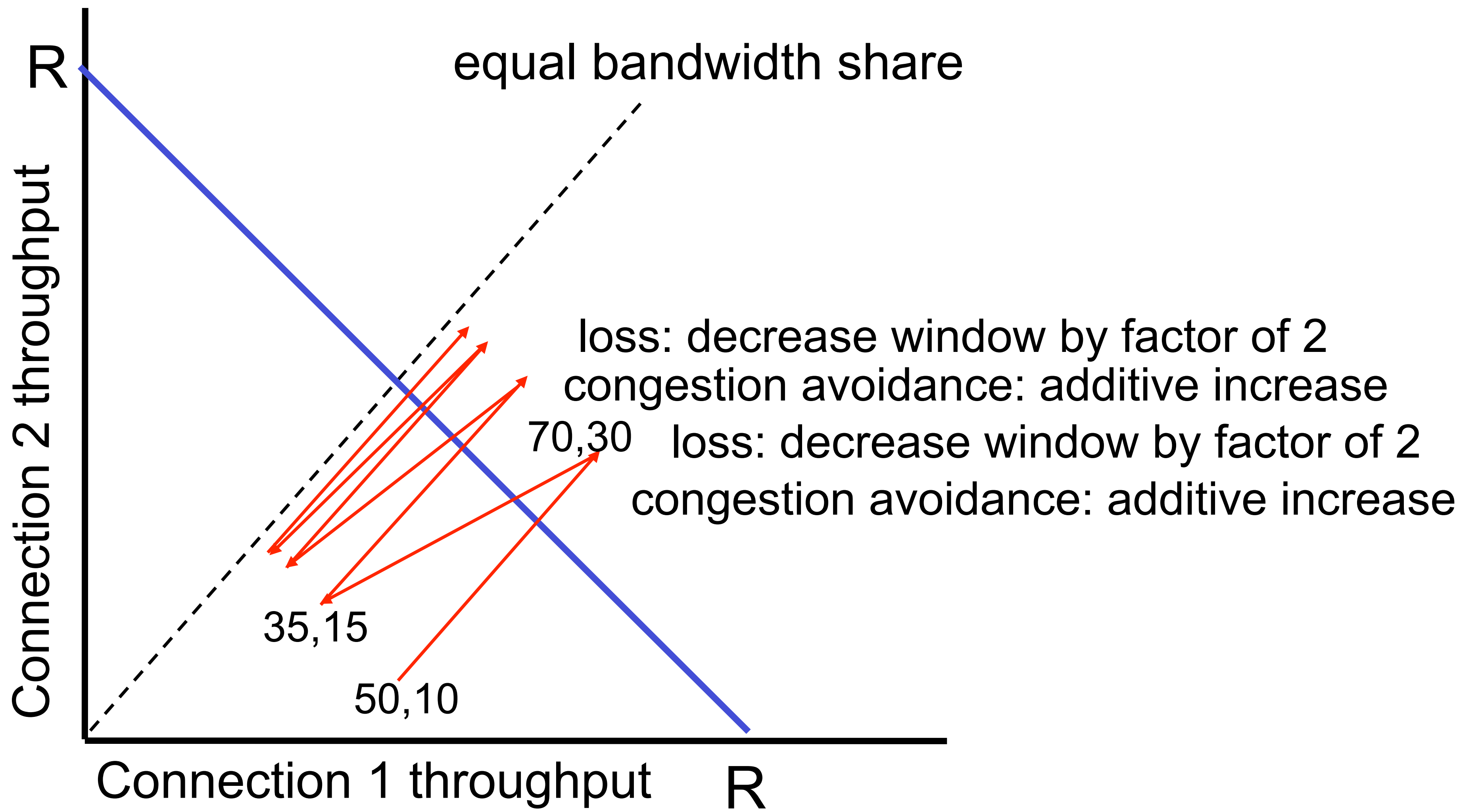
TCP Fairness

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



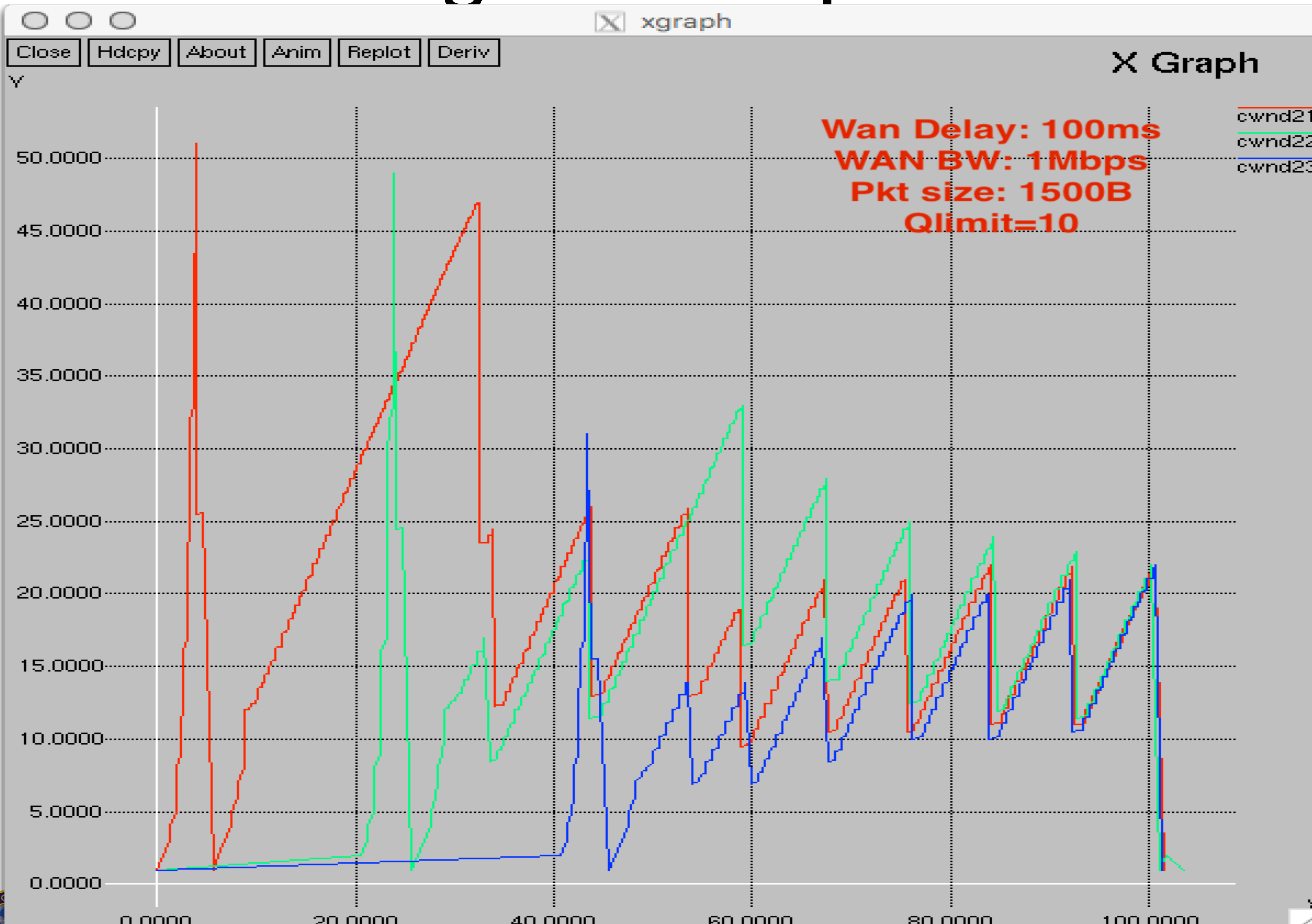
Why is TCP fair?

- Two competing sessions:
- Additive increase gives slope of 1, as throughout increases
- Multiplicative decrease decreases throughput proportionally



Full paper: http://www.cs.wustl.edu/~jain/papers/ftp/cong_av.pdf

TCP Congestion Experiment 03b



Lab Program Snippets

- Topology parameters
  - `set lanbw 100Mb`
  - `set landelay 0.001ms`
  - `set wanbw 1Mb`
  - `set wandelay 500ms`
  - `set qlimit 10`
- TCP config params
  - `set tcppktsize 1460` # Add 40 for TCP+IP hdrs
  - `set cwindow 40`
  - `Agent/TCP set packetSize_ 960`
- Initial value of threshold
  - `Agent/TCP set window_  60`
- Application parameters
  - `set conntime 200`

Lab Program Snippets

- Make a LAN : syntax
  - ns make-lan nodelist bw delay LL ifq MAC channel phy
- Tcl code example

```
$ns make-lan "$n0 $n1 $n2 $n3 $n4" $lanbw
$landelay LL Queue/DropTail Mac/802_3
$ns make-lan "$n0 $n1 $n2 $n3 $n4" $lanbw
$landelay LL Queue/DropTail Mac/Csma/Cd
```

- Create TCP Source and Sink
  - TCP agent : Tahoe (default), Reno, NewReno, …
- Tcl code example

```
set tcp0 [new Agent/TCP]
$ns attach-agent $n0 $tcp0
set tcp1 [new Agent/TCP/Reno]
$ns attach-agent $n1 $tcp1
```

Lab Program Snippets

- TCP Sink agents

```
set sink0 [new Agent/TCPSink]
set sink1 [new Agent/TCPSink]
$ns attach-agent $n2 $sink0
$ns attach-agent $n3 $sink1
```

- TCP Application agents e.g. FTP

```
set ftp0 [new Application/FTP]
set ftp1 [new Application/FTP]
$ftp0 attach-agent $tcp0
$ftp1 attach-agent $tcp1
```

- Connect TCP applications end points

```
$ns connect $tcp0 $sink0
$ns connect $tcp1 $sink1
```

Lab Program Snippets

- TCP traffic statistics recording

```
set cf0 [open conn_0.tr w]
$tcp0 attach $cf0
$tcp0 trace cwnd_
$tcp0 trace dupacks_
$tcp0 trace t_seqno_

set cf1 [open conn_1.tr w]
$tcp1 attach $cf1
$tcp1 trace cwnd_
$tcp1 trace dupacks_
$tcp1 trace t_seqno_

set cf2 [open conn_2.tr w]
:
```

Lab Program Snippets

- Finish procedure

```
proc finish { } {
  global ns nf tf file1 file2
  $ns flush-trace
  close $tf
  close $nf
  close $file1
  close $file2
  exit 0
}
```

Lab Program Snippets

- Actual traffic generation

```
$ns at 1.0 "$ftp0 start"
$ns at 5.0 "$ftp0 stop"
$ns at 5.0 "$ftp1 start"
$ns at 9.0 "$ftp1 stop”
$ns at 10.5 "$ftp0 start”
$ns at 11.0 "$ftp1 start"
$ns at 19.0 "$ftp0 stop"
$ns at 20.0 "$ftp1 stop”

$ns at 21 "finish"
$ns run
```

Generating Congestion Window

- *awk* script to process *cwnd*
  - Consider following sample trace

```
0.00000  0  0  5  0  cwnd_ 1.000
0.10767  0  0  5  0  cwnd_ 2.000
0.13126  0  0  5  0  cwnd_ 3.000
```

- program *cong.awk*

```
BEGIN { } {
  if($6=="cwnd_")
   printf("%f\t%f\t\n",$1,$7);
}
END { }
```

Analyzing Cwnd

- Processing cwnd trace file

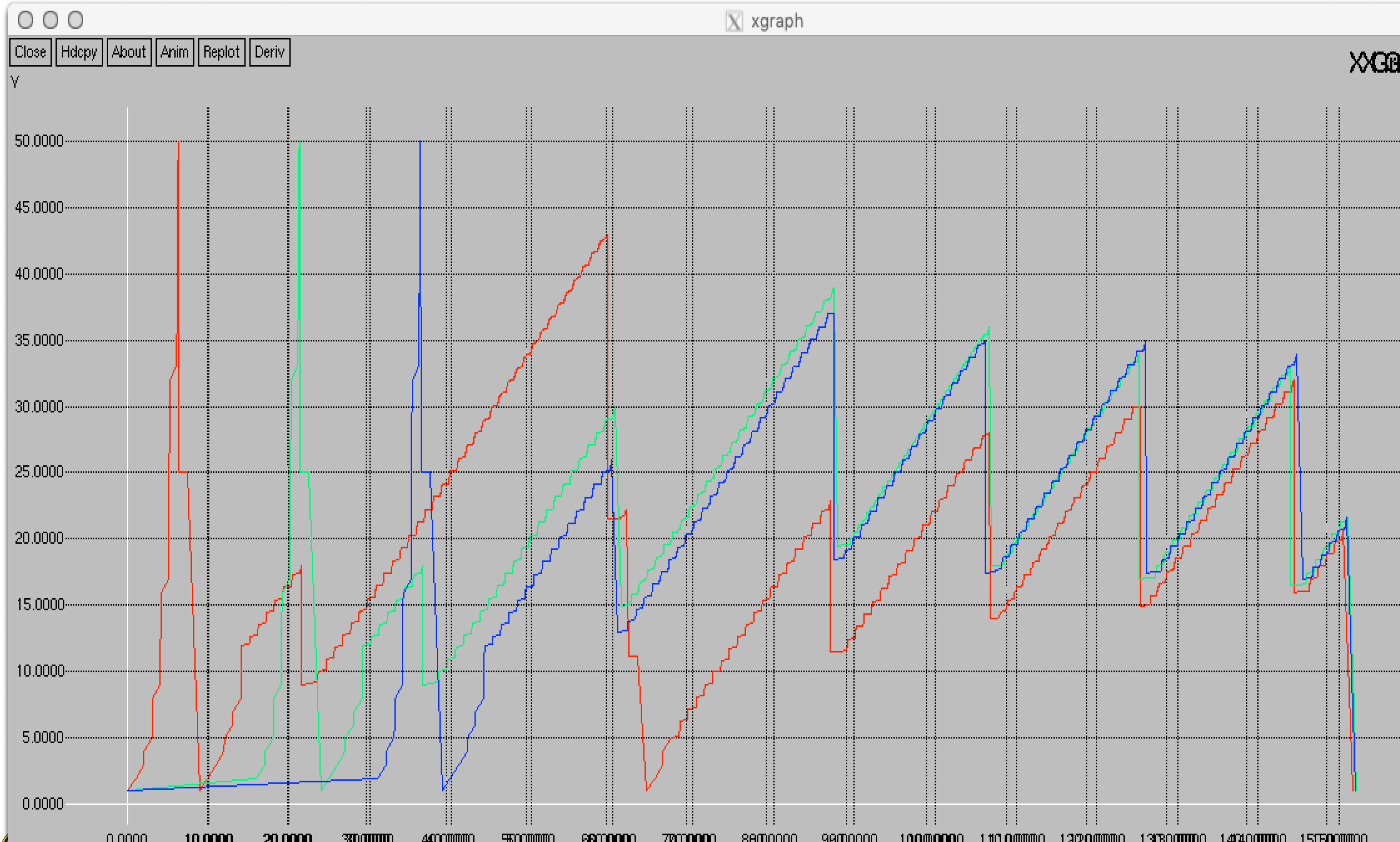
```
awk -f cong.awk conn_0.tr >xgr.conn_0
awk -f cong.awk conn_1.tr >xgr.conn_1
awk -f cong.awk conn_2.tr >xgr.conn_2
```

- Viewing graph

```
xgraph xgr.conn*
```

- Analyze the graph for following
  - AIMD pattern of congestion window size
  - TCP Fairness among 3 connections
-

Sample graph - Congestion window



Lab exercise

- Lab 3a:
  - Generate FTP traffic with 1 src and 1 sink in one LAN itself
  - Analyze TCP congestion window
- Lab 3b:
  - Generate non-overlap FTP traffic
    - With 2 srcs and 2 sinks in one LAN
  - Analyze TCP cwnd
- Lab 3c
  - Generate overlapping traffic of 2 srcs and 2 sinks in one LAN
  - Analyze TCP cwnd, and TCP Fairness
- Lab 3d
  - Create two LANs each of 3 nodes, connected via a link.
  - Generate traffic across LANs
  - Analyze congestion window for each conn, and TCP fairness

Summary

- TCP congetion control
- TCP fairness
- Generating ethernet LAN
- Connecting two ethernet LAN with a WAN link
- Generate FTP traffic between sources in 1st LAN and destination in 2nd LAN
- Analysis of TCP Congestion window and TCP Fairness.