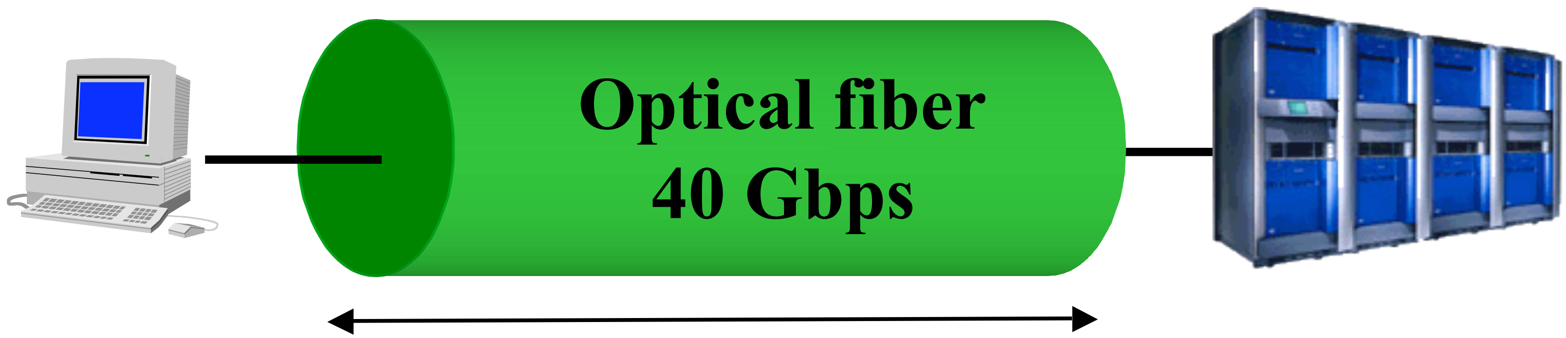
TCP on High-Speed Networks

from
"New Internet and Networking Technologies for Grids and High-Performance Computing",
tutorial given at
HiPC'04, Bangalore, India
December 22nd, 2004

C. Pham
University Lyon, France
LIP (CNRS-INRIA-ENS-UCBL)

NEW CHAPTER

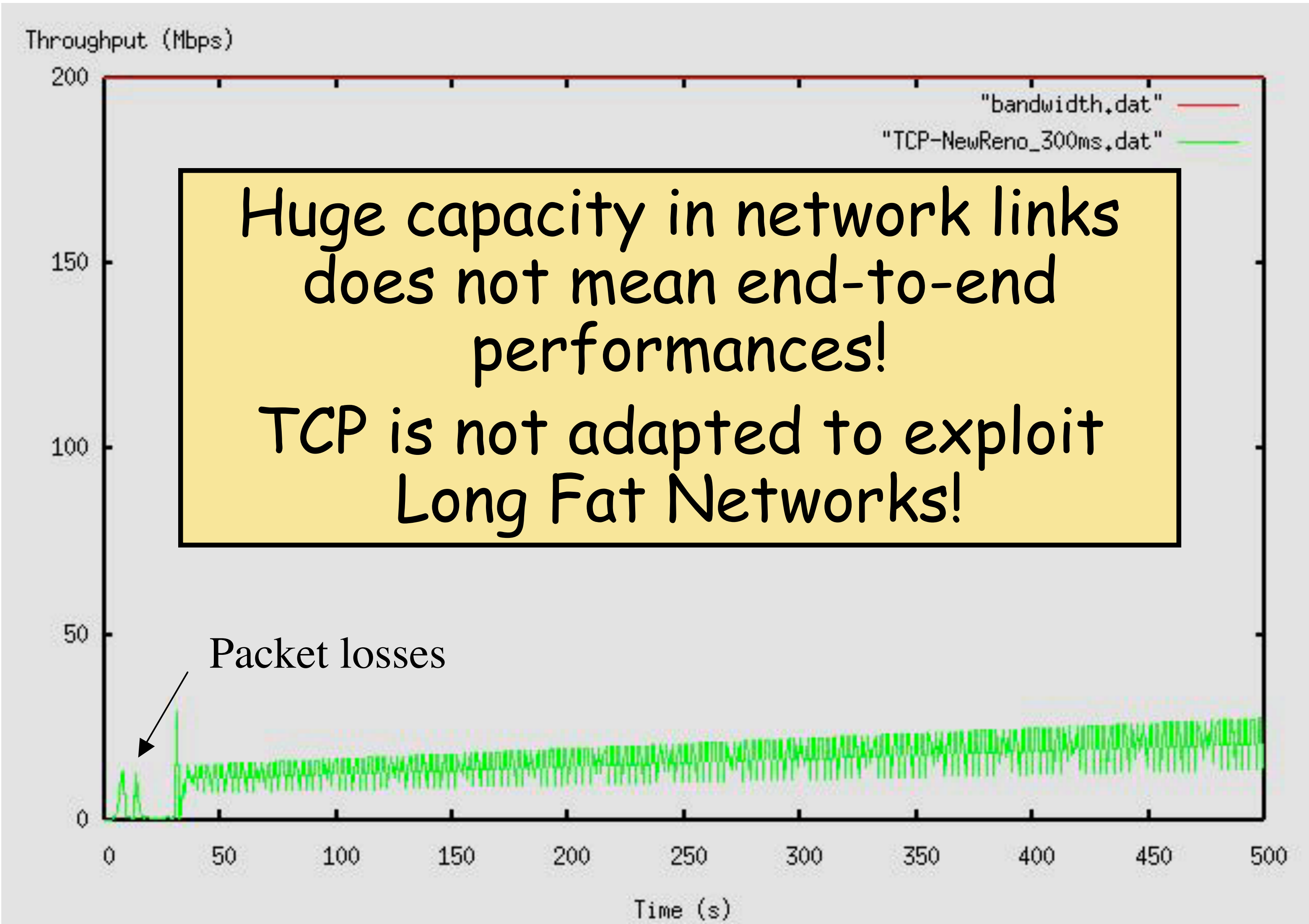
TCP & High-speed networks



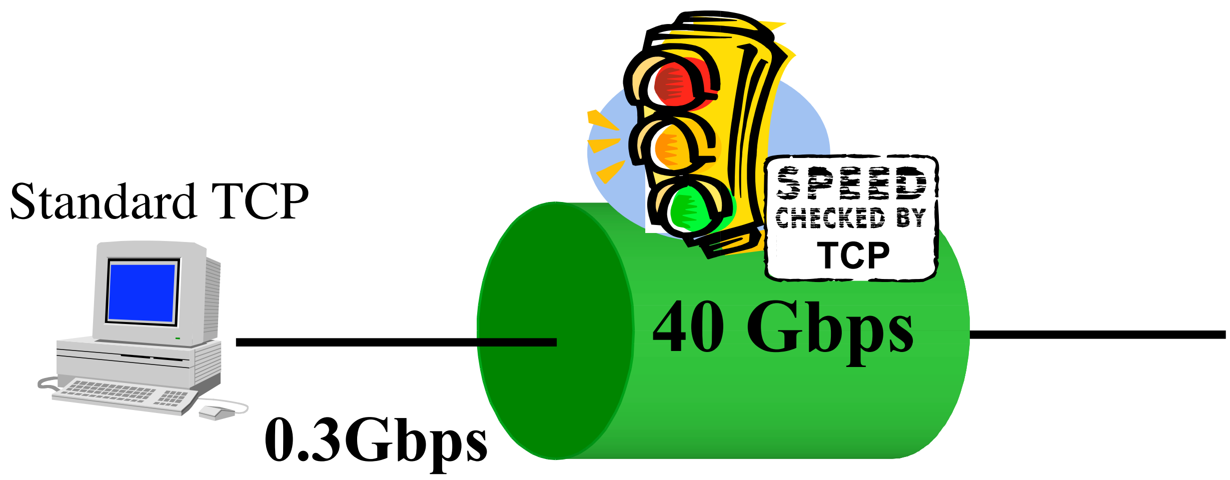
200000km/s, delay of 5ms every 1000km

- Today's backbone links are optical, DWDM-based, and offer gigabit rates
- Transmission time <<< propagation time
- Duplicating a 10GB database should not be a problem anymore

The reality check: TCP on a 200Mbps link

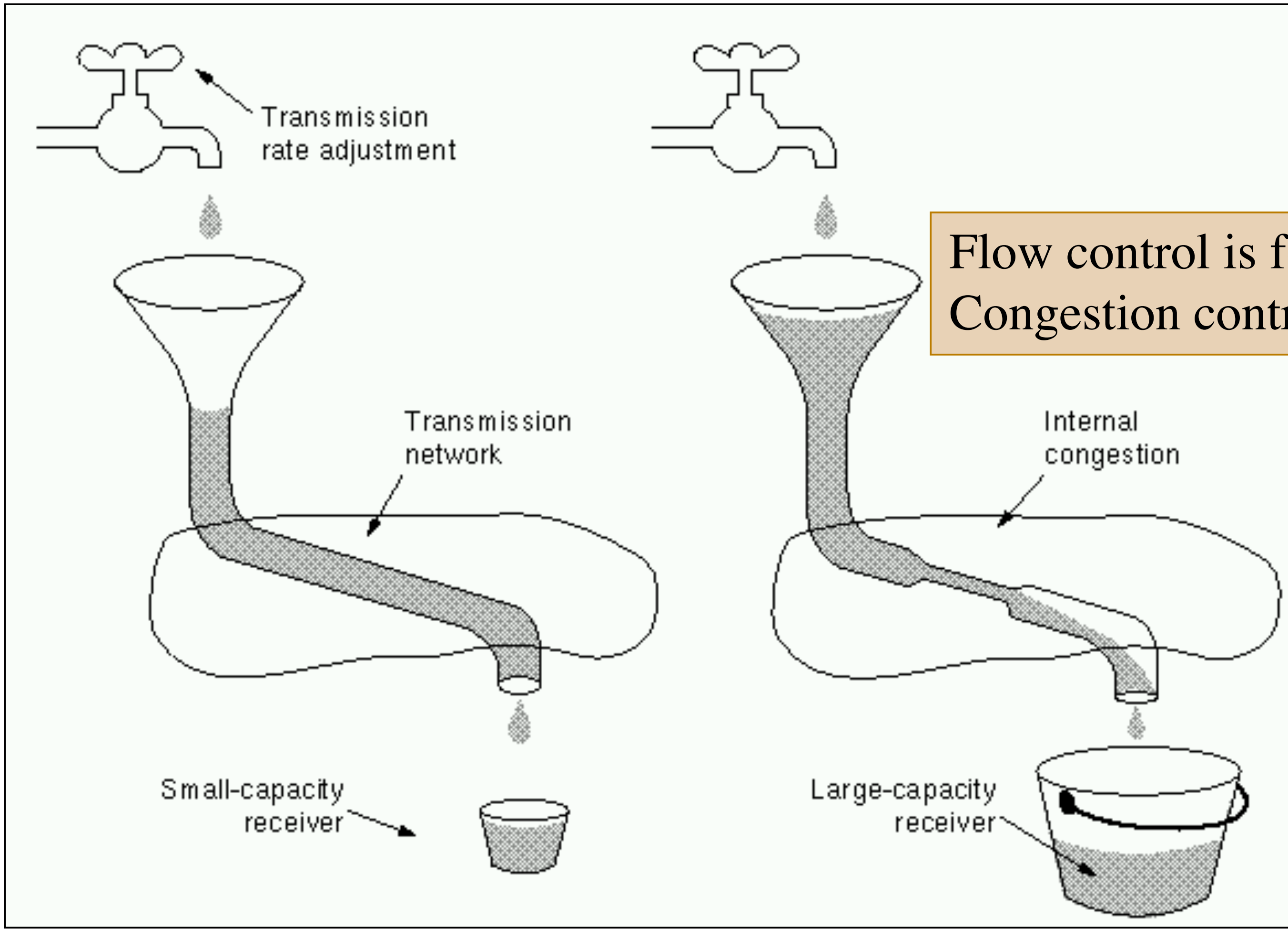


The things about TCP your mother never told you!



- If you want to transfer a 1Go file with a standard TCP stack, you will need minutes even with a 40Gbps (how much in $?) link!

Let's go back to the origin!



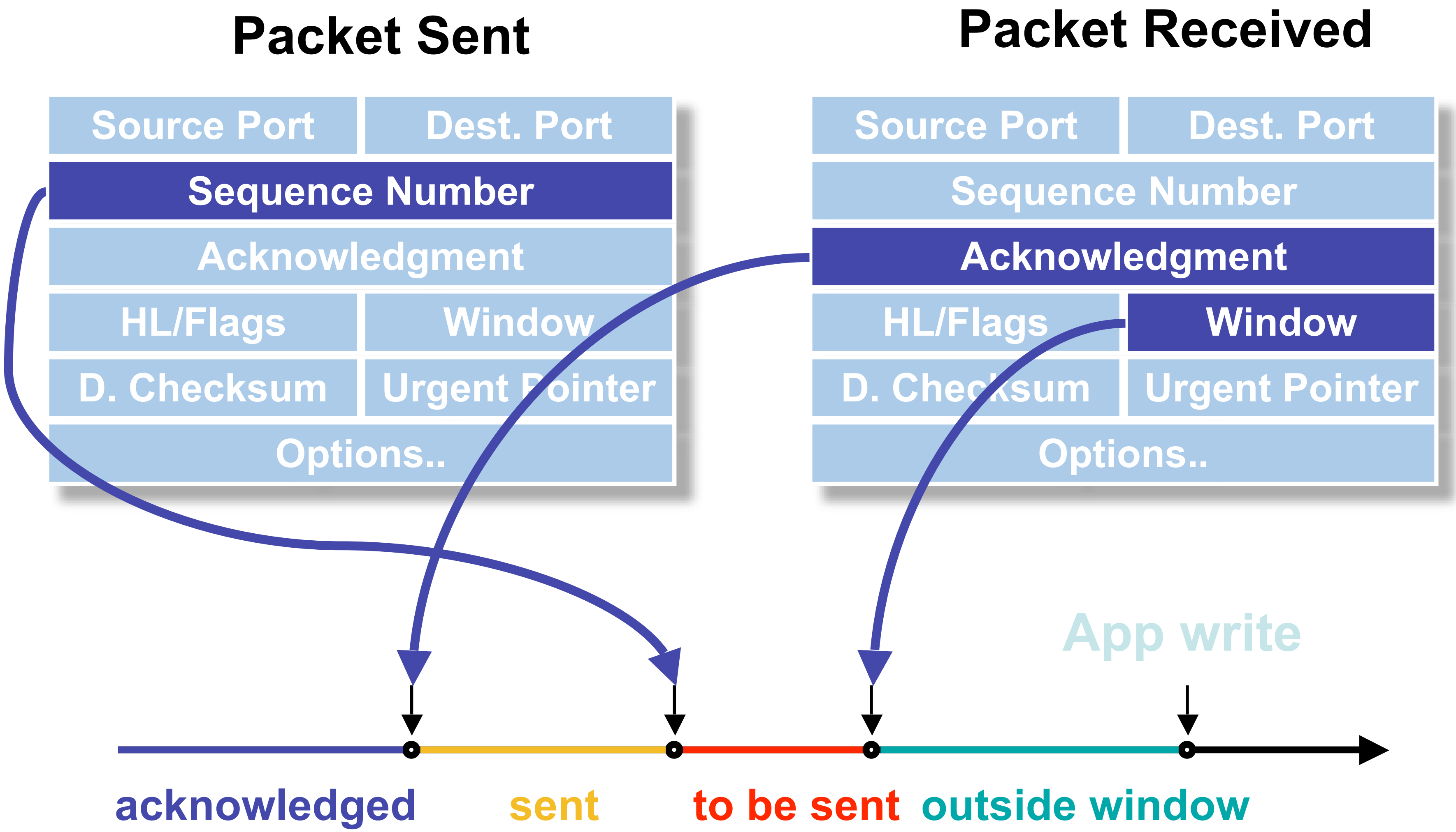
From Computer Networks, A. Tanenbaum

Flow control is for receivers
Congestion control is for the network

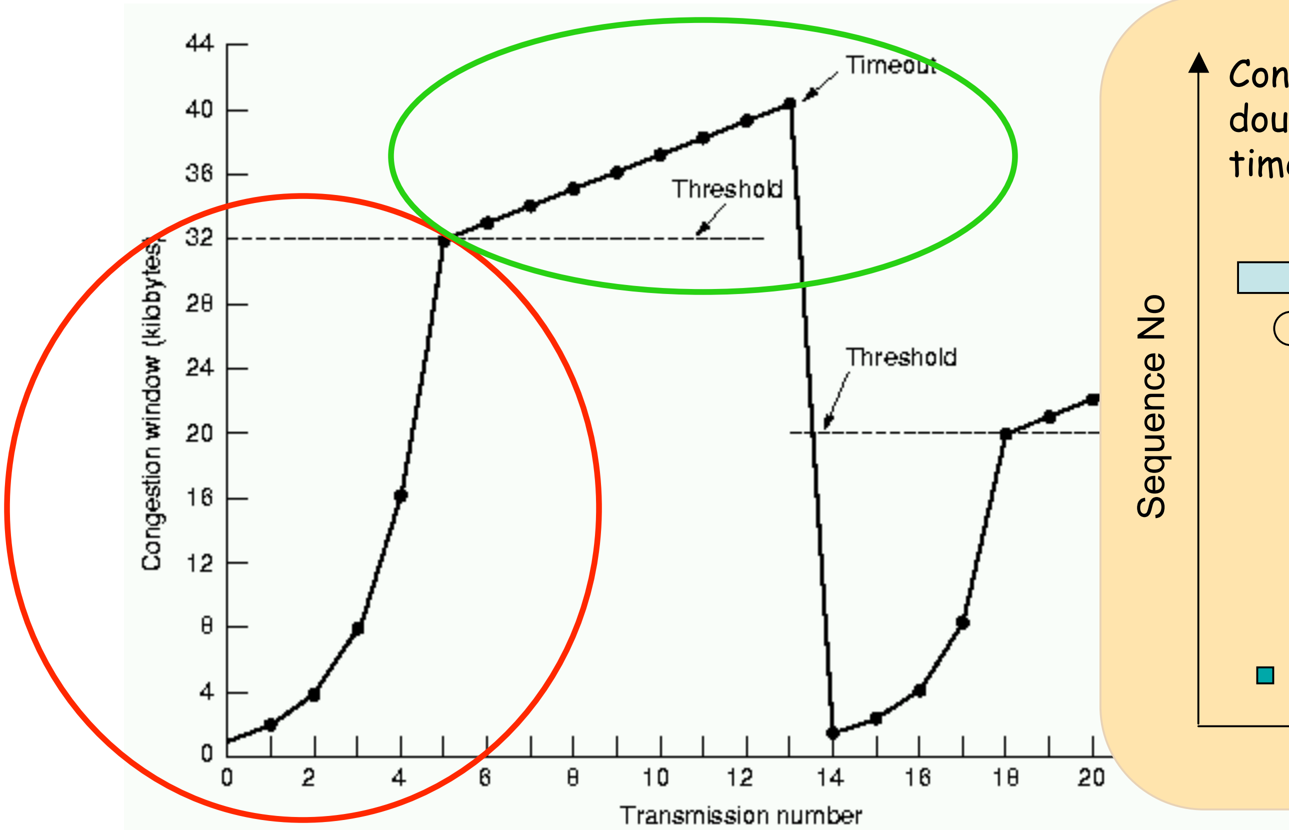
Congestion collapse was first observed in 1986 by V. Jacobson. Congestion control was added to TCP (TCP Reno) in 1988.

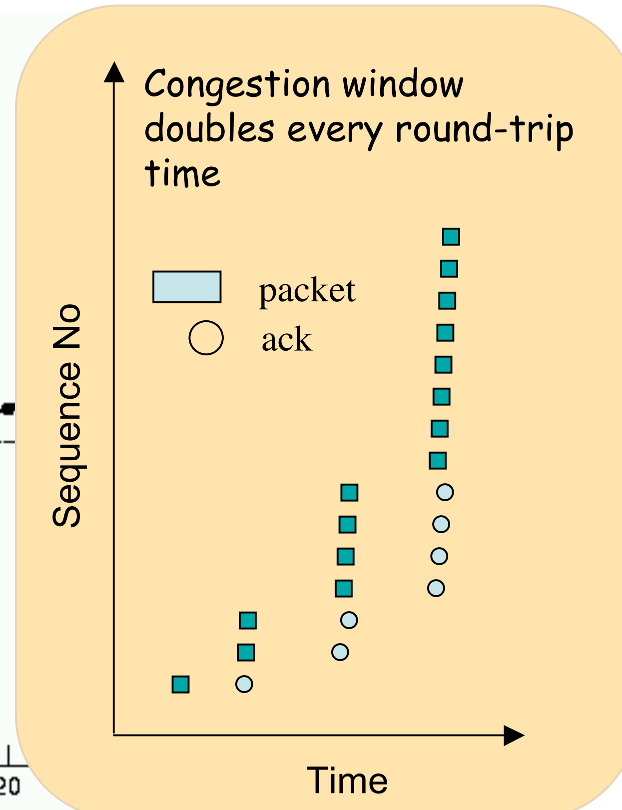
Flow control

prevents receiver's buffer overfow



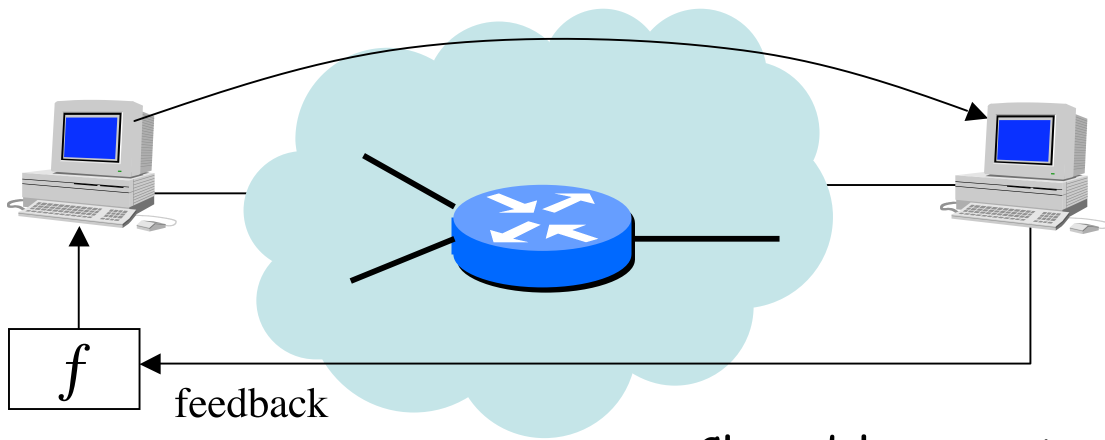
TCP congestion control: the big picture





- cwnd grows exponentially (slow start), then linearly (*congestion avoidance*) with 1 more segment per RTT
- If loss, divides threshold by 2 (multiplicative decrease) and restart with cwnd=1 packet

From the control theory point of view



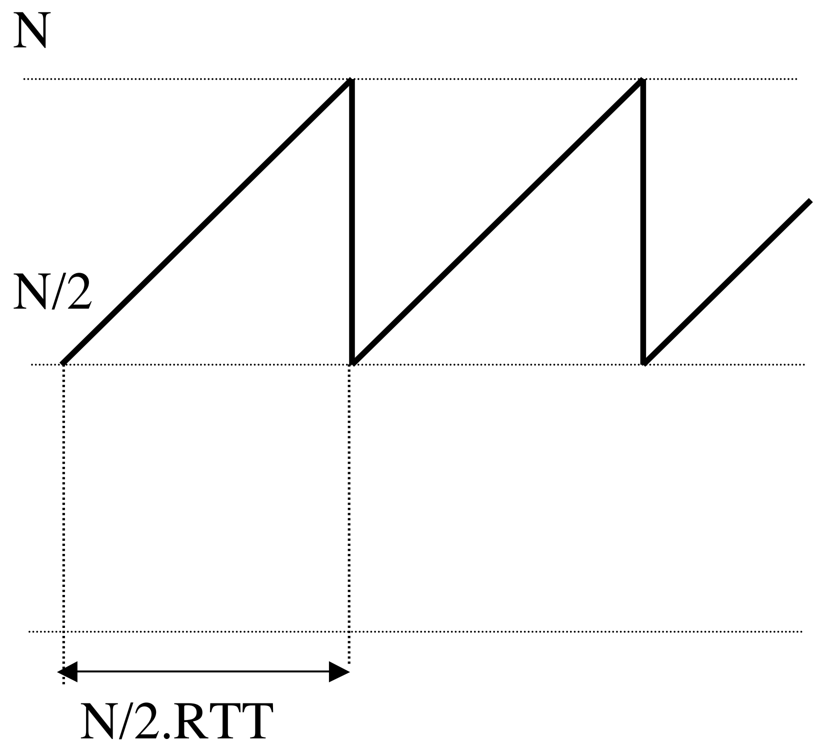
Closed-loop control

- Feedback should be frequent, but not too much otherwise there will be oscillations
- Can not control the behavior with a time granularity less than the feedback period

The TCP saw-tooth curve

TCP behavior in steady state

Isolated packet losses trigger the fast recovery procedure instead of the slow-start.

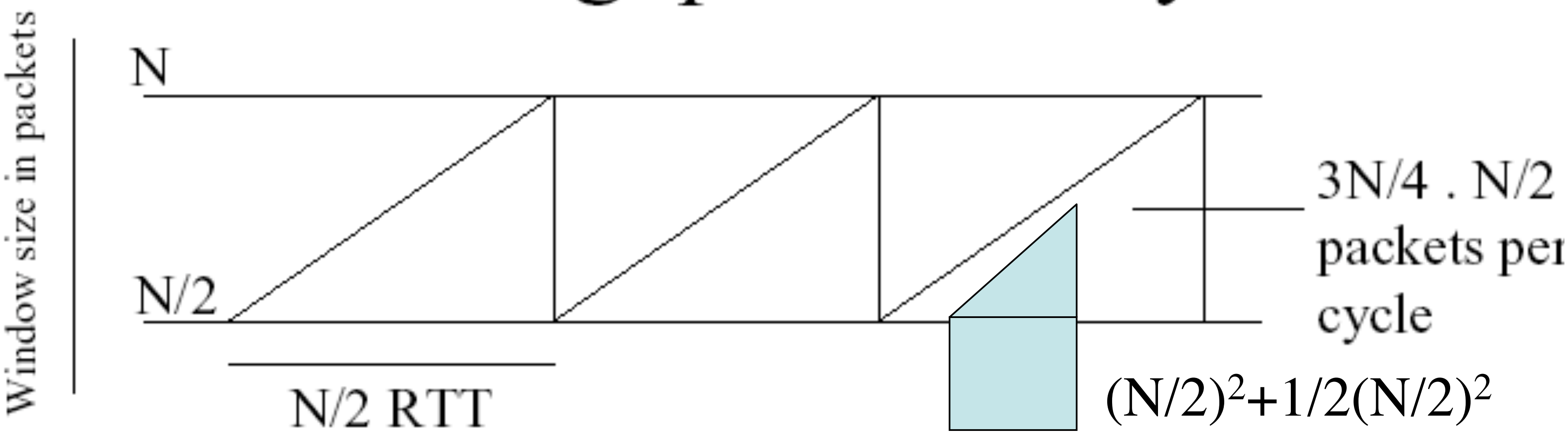


- The TCP steady-state behavior is referred to as the Additive Increase- Multiplicative Decrease process

no loss:
cwnd = cwnd + 1

loss:
cwnd = cwnd*0.5

TCP throughput in steady state



Average window size (in packets) = W = 3N/4 , from (N+N/2)/2

Number of packets per cycle = 3N/4 . N/2 = $3N^2/8$ = 1/ p

- Where p is the packet loss ratio (which should remain small enough)
- So $N = \sqrt{8/3p}$

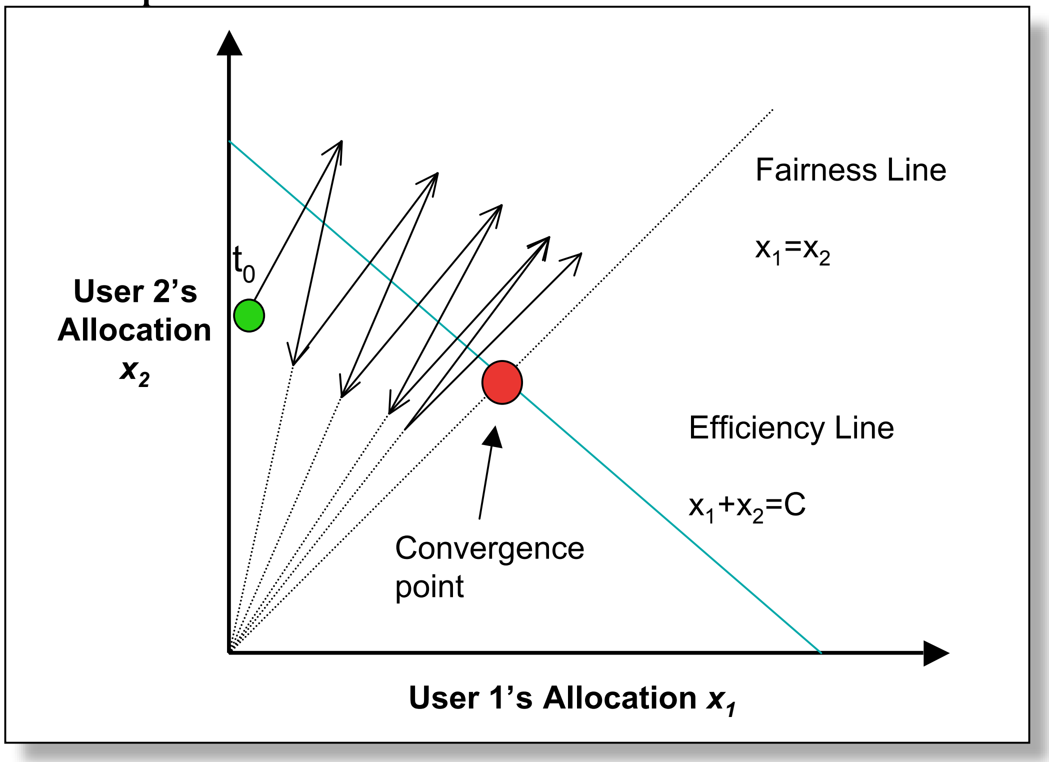
Average throughput (in packets/sec) = B = W / RTT = 3N / 4 RTT

$$= \sqrt{3/2}\,\frac{1}{RTT\sqrt{p}}$$

$$Throughput = \frac{W}{RTT} = \sqrt{\frac{3}{2}}\,\frac{MTU}{RTT\sqrt{p}}$$

AIMD

Phase plot

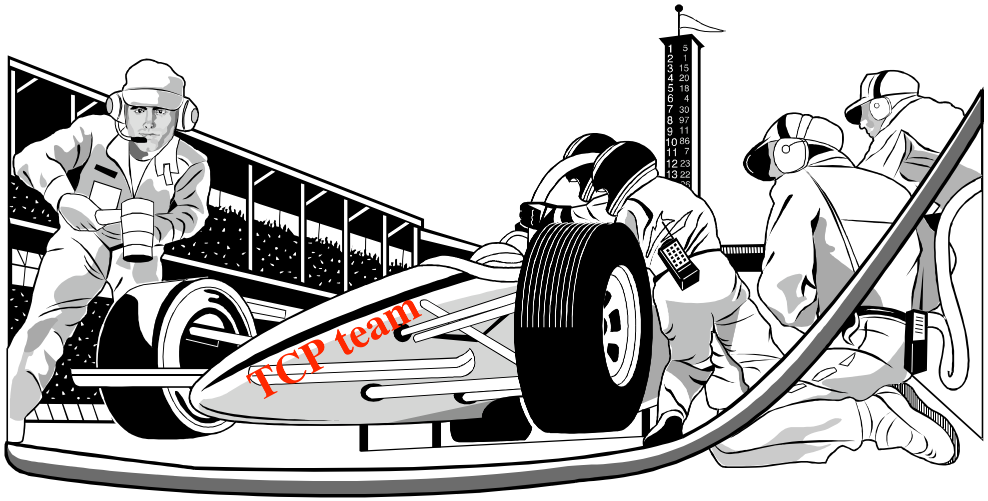


Multiplicative Decrease preserves the fairness because the user's allocation ratio remains the same

Ex: $$\frac{x_2}{x_1} = \frac{x_2.b}{x_1.b}$$

- Assumption: decrease policy must (at minimum) reverse the load increase over-and-above efficiency line
- Implication: decrease factor should be conservatively set to account for any congestion detection lags etc

Tuning stand for TCP
the dark side of speed!



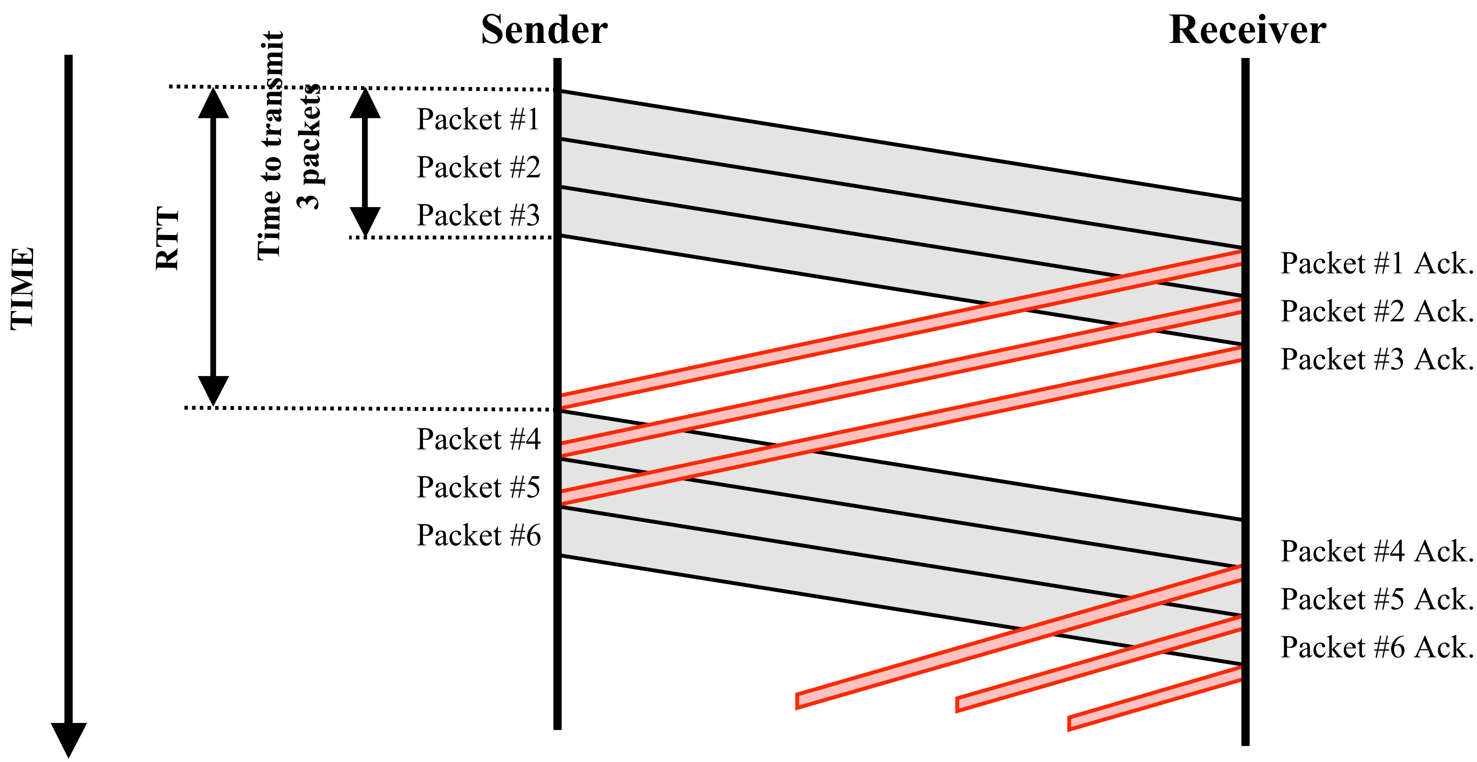
TCP performances depend on

- TCP & network parameters
  - Congestion window size, *ssthresh* (threshold)
  - RTO timeout settings
  - SACKs
  - Packet size
- System parameters
  - TCP and OS buffer size (in comm. subsys., drivers…)

NEED A SPECIALIST!

First problem: window size

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

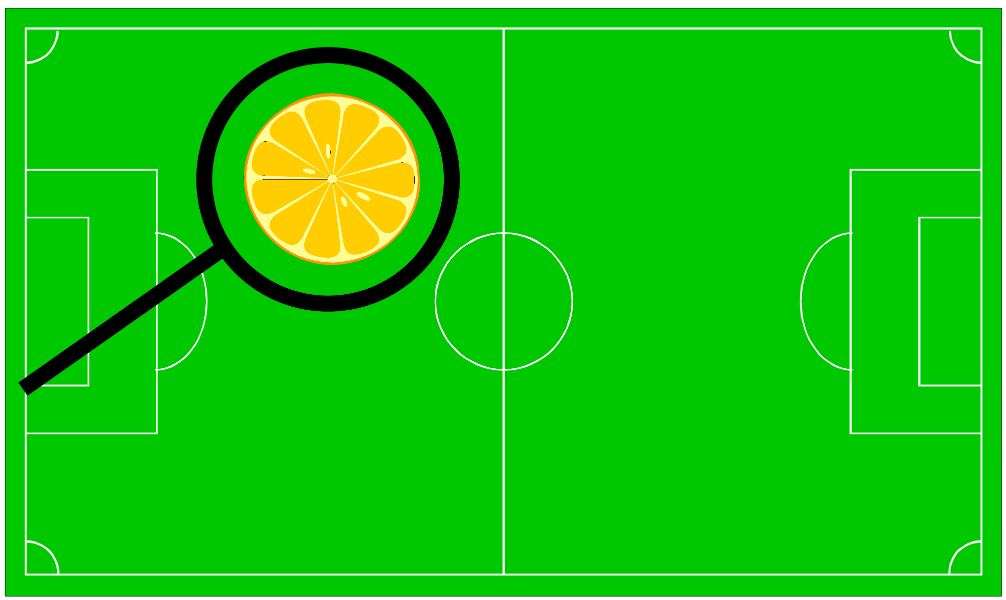


First problem: window size

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

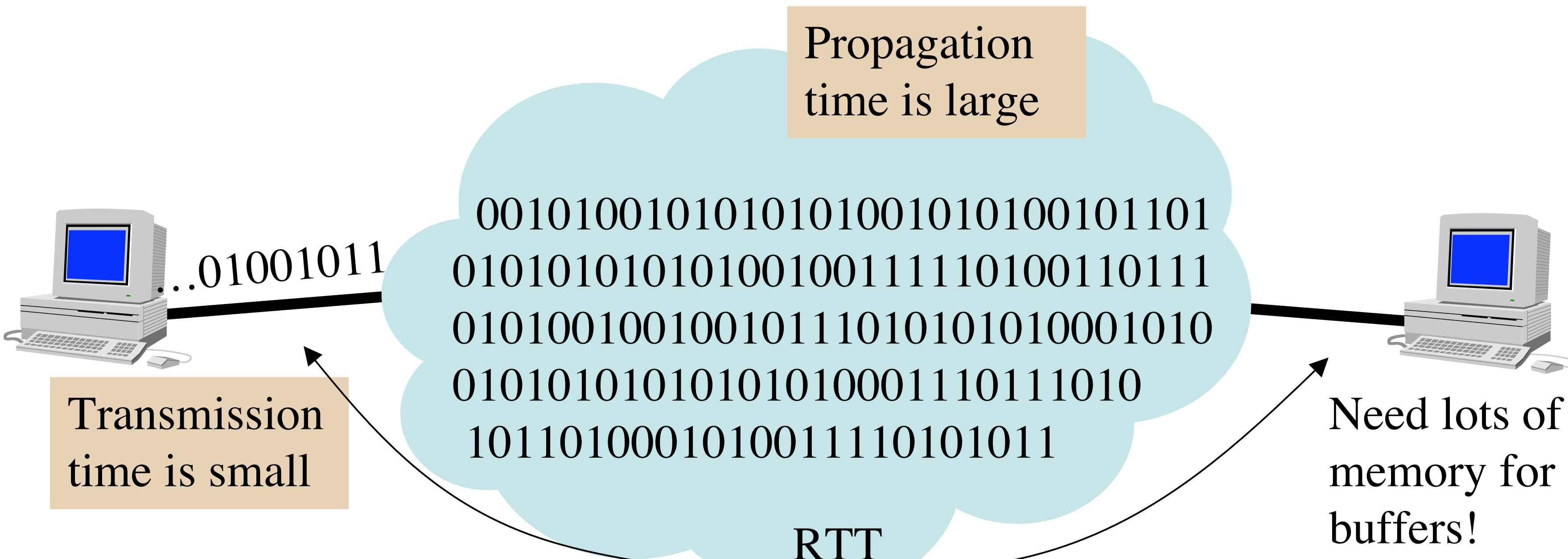
RTT=200ms Link is 0C-48 = 2.5 Gbps

Waiting time



Rule of thumb on LFNs

capacity
- High-~~speed~~ network

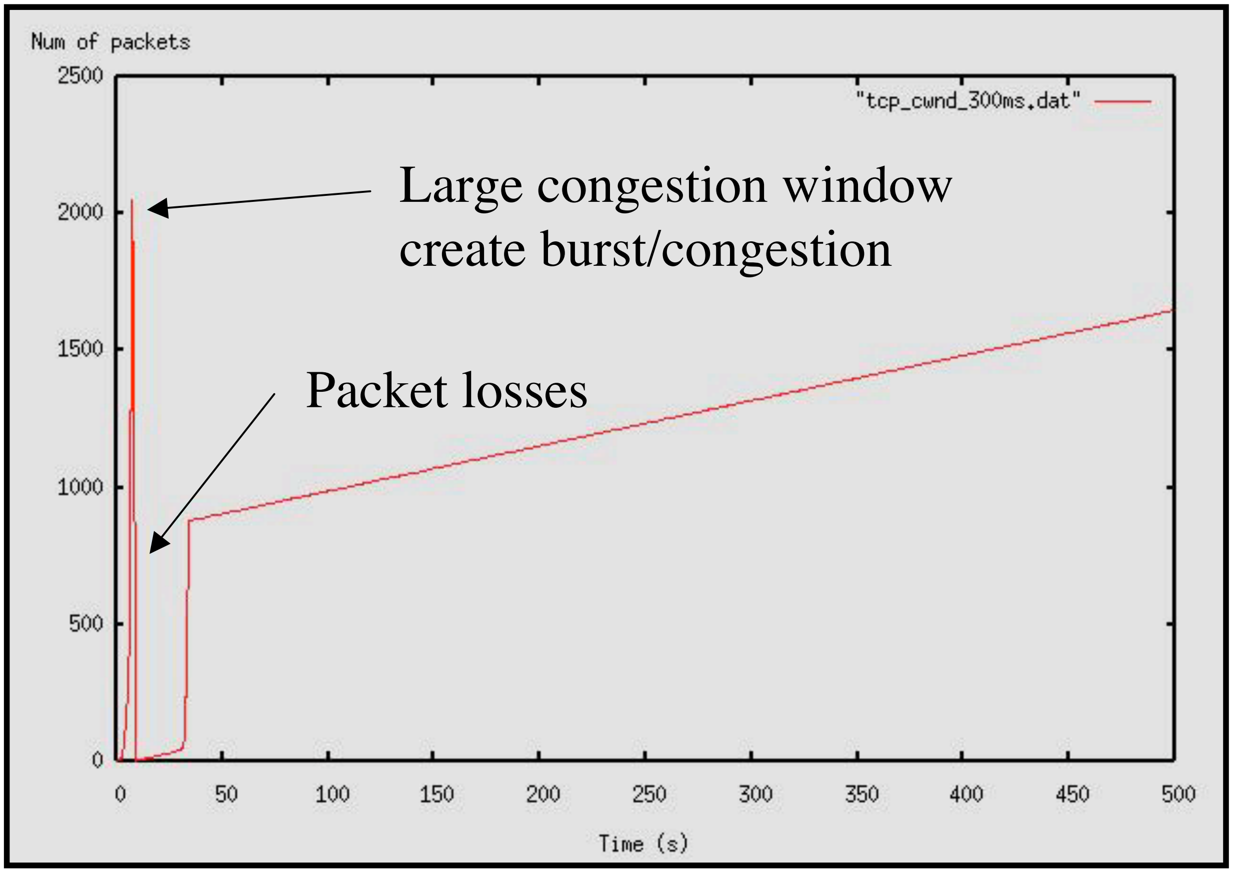


The optimal window size should be set to the bandwidthxRTT product to avoid blocking at the sender side

Side effect of large windows

TCP becomes very sensitive to packet losses on LFN

Congestion window size

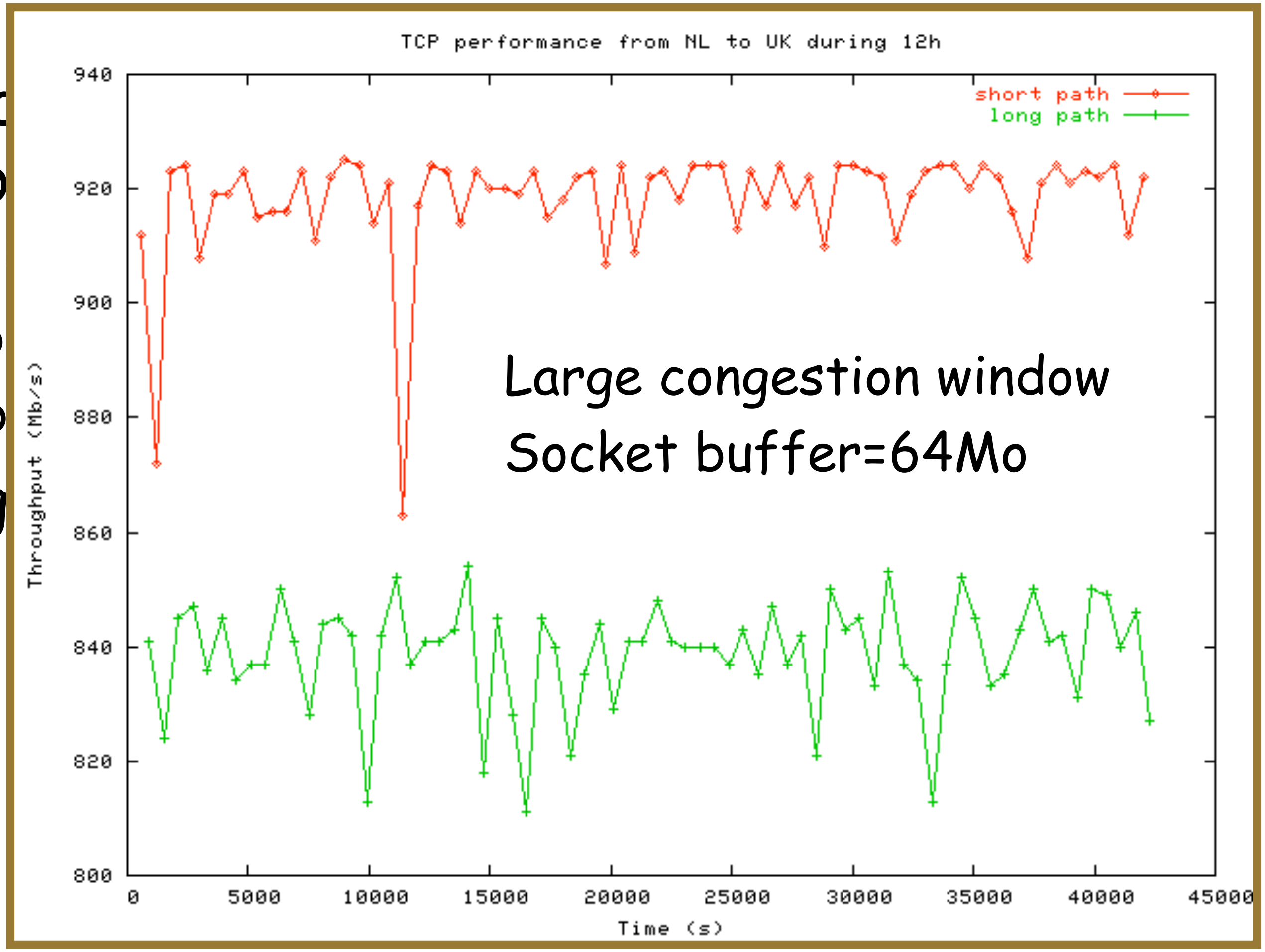


Pushing the limits of TCP

- Standard configuration (vanilla TCP) is not adequate on many OS, everything is under-sized
  - Receiver buffer
  - System buffer
  - Default block size
- Will manage to get near 1Gbps if well-tuned

Pushing the limits of TCP

- Standard c
  adequate o
  - Receiver
  - System b
  - Default b
- Will manag

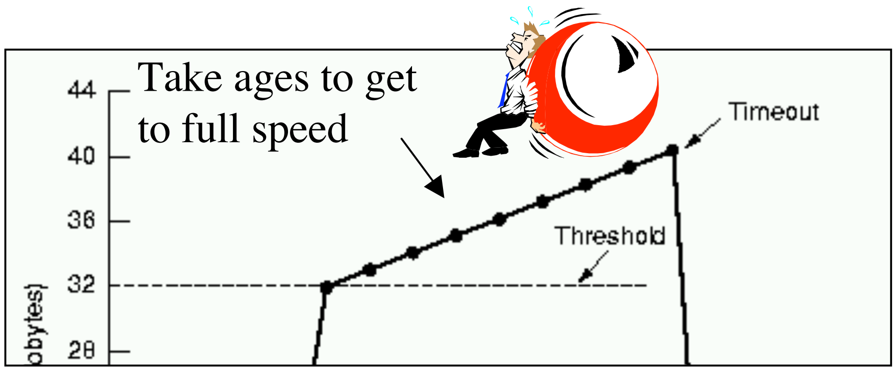


Source: M. Goutelle, GEANT test campaign

Some TCP tuning guides

- http://www.psc.edu/networking/projects/tcptune/
- http://www.web100.org/
- http://rdweb.cns.vt.edu/public/notes/win2k-tcpip.htm
- http://www.sean.de/Solaris/soltune.html
- http://datatag.web.cern.ch/datatag/howto/tcp.html

The problem on high capacity link? Additive increase is still too slow!



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

Once you get high throughput, maintaining it is difficult too!

- 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:

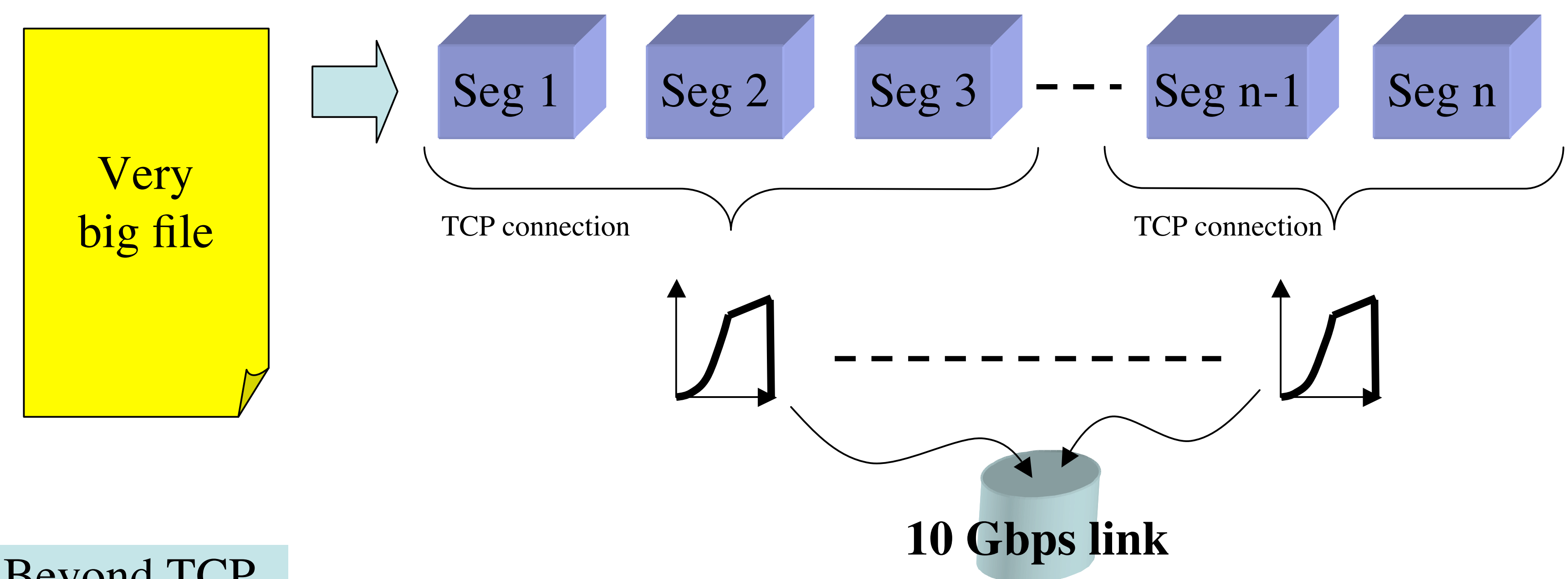
- an average congestion window of 83,333 segments;
- and at most one drop (or mark) every 5,000,000,000 packets (or equivalently, at most one drop every 1 2/3 hours).

This is not realistic.

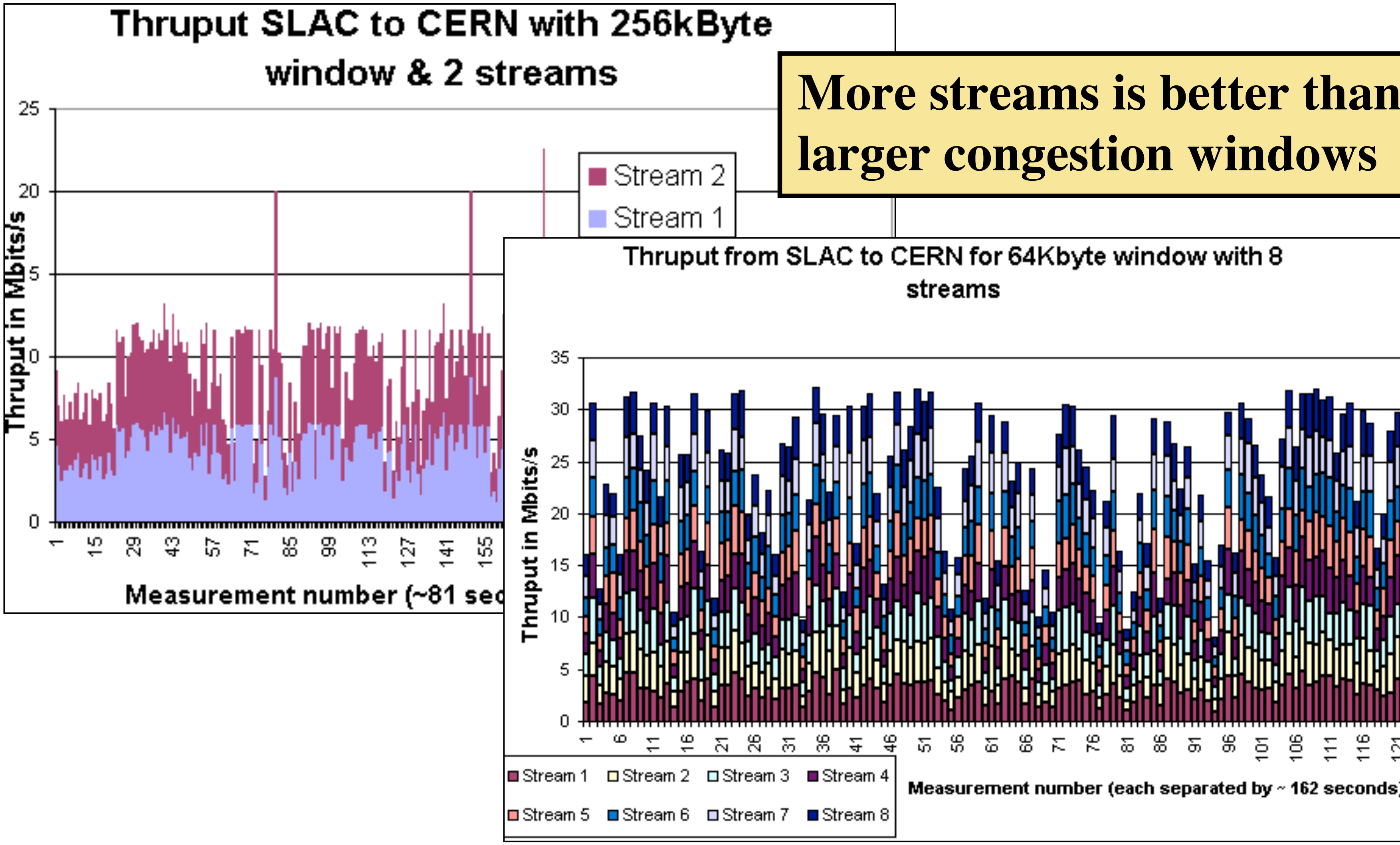
From S. Floyd

Going faster (cheating?)
n flows is better than 1

- The CC limits the throughput of a TCP connection: so why not use more than 1 connection for the same file?



Some results from IEPM/SLAC



More streams is better than larger congestion windows

Beyond TCP

http://www-iepm.slac.stanford.edu/monitoring/bulk/window-vs-streams.html

Multiple streams

- No/few modifications to transport protocols (i.e. TCP)
  - Parallel socket libraries
  - GridFTP (http://www.globus.org/datagrid/gridftp.html)
  - bbFTP (http://doc.in2p3.fr/bbftp/)

New transport protocols

- New transport protocols are those that are not only optimizations of TCP
- New behaviors, new rules, new requirements! Everything is possible!
- New protocols are then not necessarily TCP compatible!

The new transport protocol strip

High Speed TCP [Floyd]

- Modifies the response function to allow for more link utilization in current high-speed networks where the loss rate is smaller than that of the networks TCP was designed for (at most 10^{-2})

TCP Throughput (Mbps)	RTTs Between Losses	W	P
1	5.5	8.3	0.02
10	55.5	83.3	0.0002
100	555.5	833.3	0.000002
1000	5555.5	8333.3	0.00000002
10000	55555.5	83333.3	0.0000000002

Table 1: RTTs Between Congestion Events for Standard TCP, for 1500-Byte Packets and a Round-Trip Time of 0.1 Seconds.

From draft-ietf-tsvwg-highspeed-01.txt

Modifying the response

Packet Drop Rate P	Congestion Window W	RTTs Between Losses
10^-2	12	8
10^-3	38	25
10^-4	120	80
10^-5	379	252
10^-6	1200	800
10^-7	3795	2530
10^-8	12000	8000
10^-9	37948	25298
10^-10	120000	80000

Table 2: TCP Response Function for Standard TCP. The average congestion window W in MSS-sized segments is given as a function of the packet drop rate P.

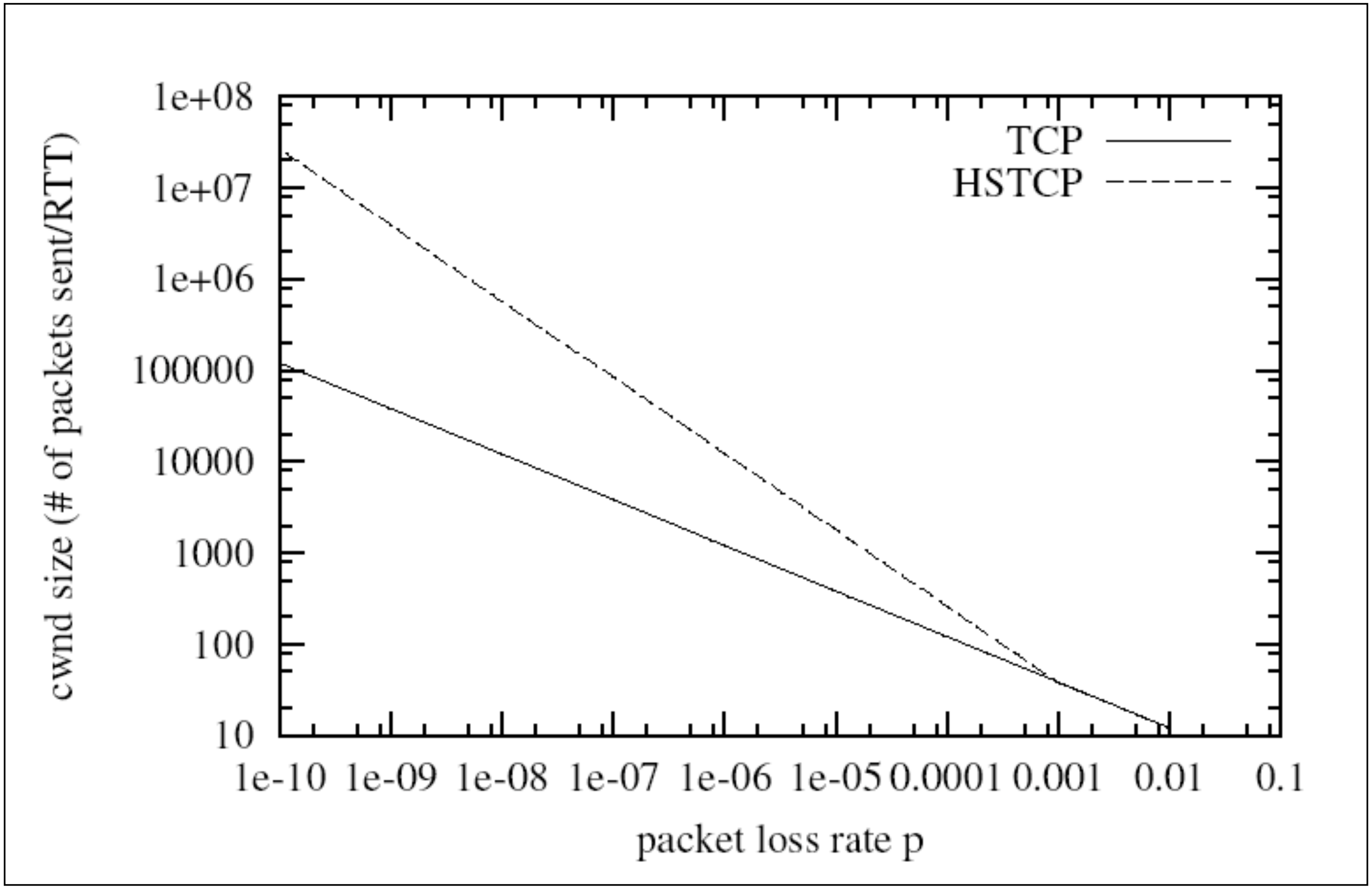
From draft-ietf-tsvwg-highspeed-01.txt

To specify a modified response function for HighSpeed TCP, we use three parameters, Low_Window, High_Window, and High_P. To Ensure TCP compatibility, the HighSpeed response function uses the same response function as Standard TCP when the current congestion window is at most Low_Window, and uses the HighSpeed response function when the current congestion window is greater than Low_Window. In this document we set Low_Window to 38 MSS-sized segments, corresponding to a packet drop rate of 10^-3 for TCP.

Packet Drop Rate P	Congestion Window W	RTTs Between Losses
10^-2	12	8
10^-3	38	25
10^-4	263	38
10^-5	1795	57
10^-6	12279	83
10^-7	83981	123
10^-8	574356	180
10^-9	3928088	264
10^-10	26864653	388

Table 3: TCP Response Function for HighSpeed TCP. The average congestion window W in MSS-sized segments is given as a function of the packet drop rate P.

See it in image



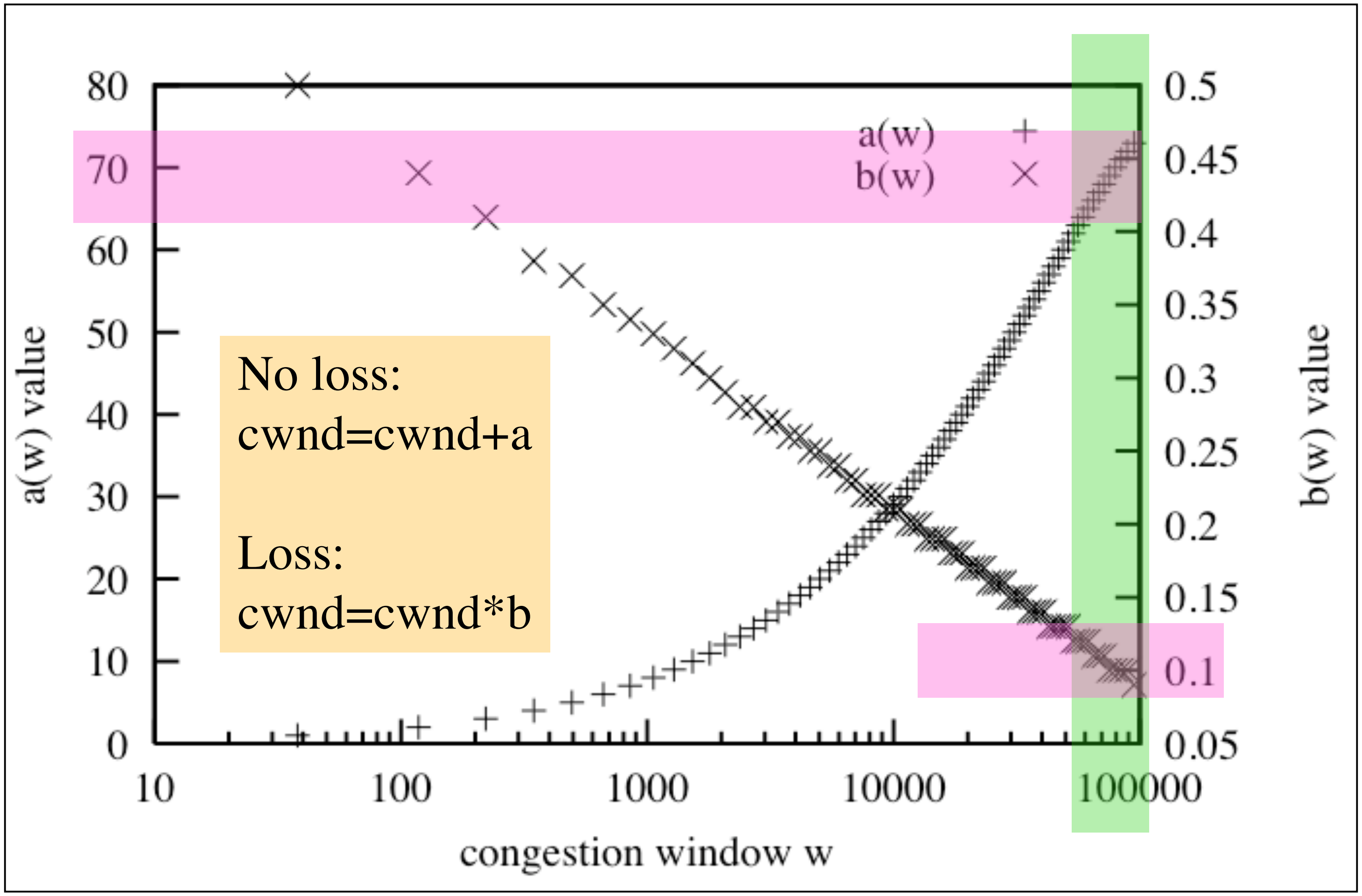
Relation with AIMD

- TCP-AIMD
  - Additive increase: a=1
  - Multiplicative decrease: b=1/2
- HSTCP-AIMD
  - Link a & b to congestion window size
  - a = a(cwnd), b=b(cwnd)

no loss:
cwnd = cwnd + 1

loss:
cwnd = cwnd*0.5

Quick to grab bandwidth, slow to give some back!



Scalable TCP [Kelly]

Let a and b be constants and $cwnd$ be the congestion window

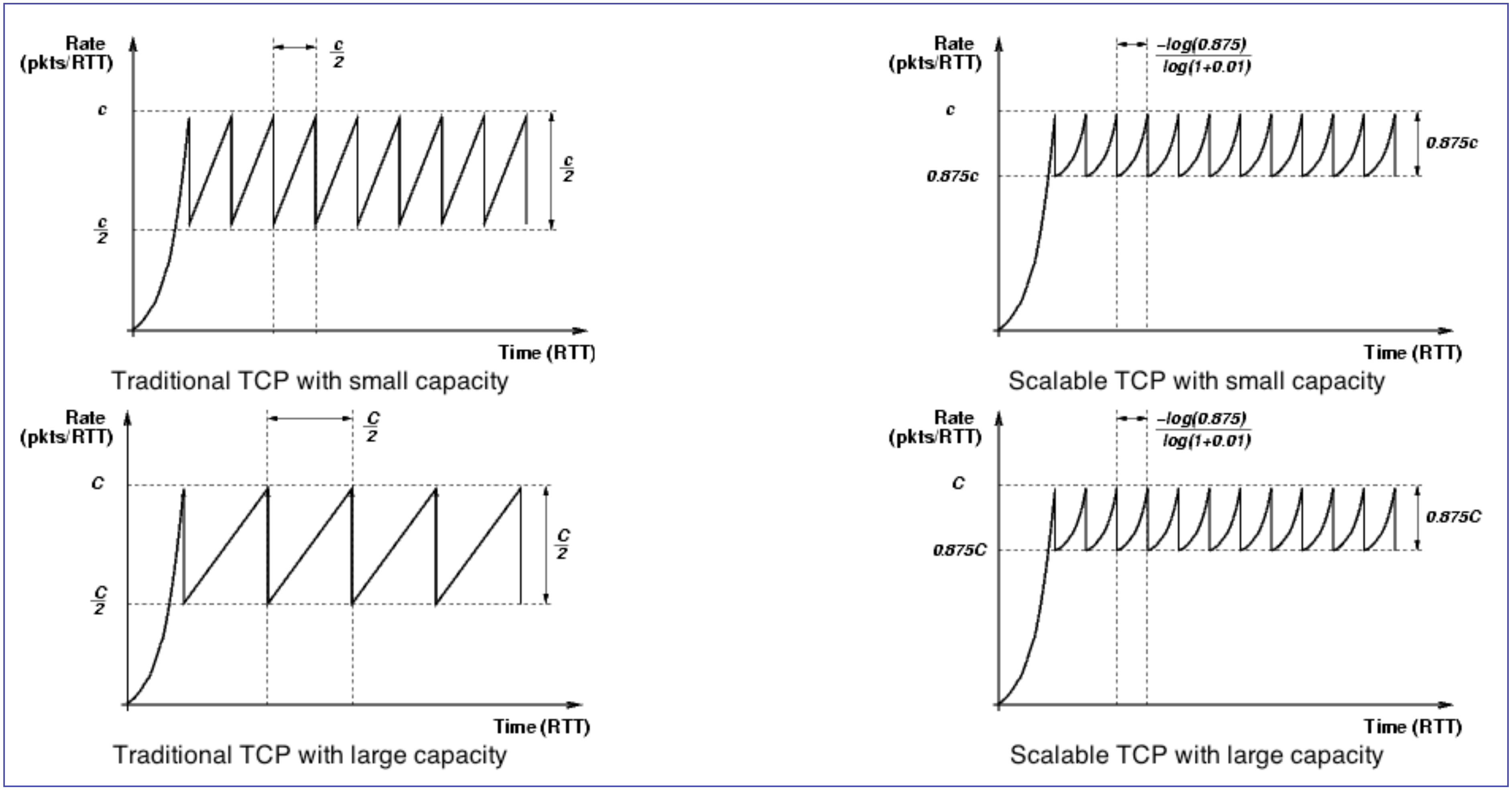
- for each ack in a RTT without loss:
  $cwnd \mapsto cwnd + a$
- for each window experiencing loss:
  $cwnd \mapsto cwnd - b \times cwnd$

Loss recovery times for RTT 200ms and MTU 1500bytes

- Scalable TCP: $\frac{log(1-b)}{log(1+a)}$ RTTs
  e.g. if $a = 0.01, b = 0.125$ then it is about 2.7s
- Traditional: at 50Mbps about 1min 38s, at 500Mbps about 27min 47s!

From 1st PFLDnet Workshop, Tom Kelly

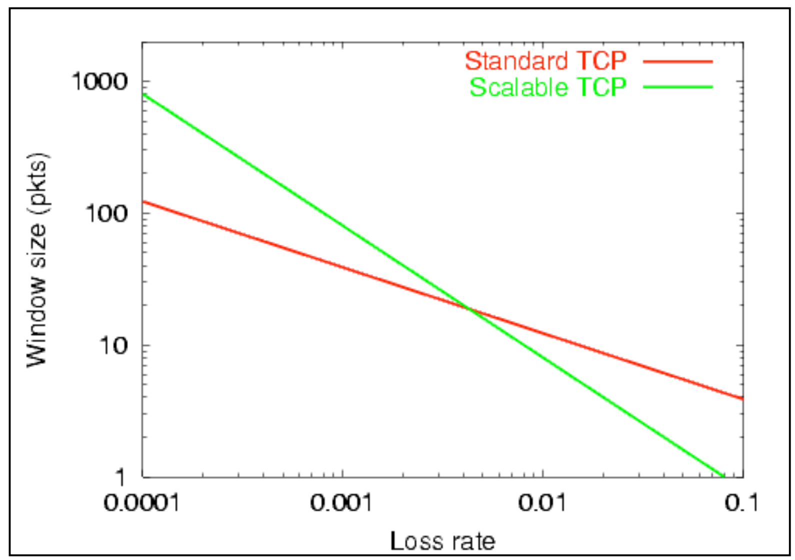
STCP in images



From 1st PFLDnet Workshop, Tom Kelly

Fairness of STCP

- Farness is achieved by having the same AIMD parameters for small congestion window values: same solution than HS-TCP
- Threshold: lcwnd=16



http://www-lce.eng.cam.ac.uk/~ctk21/scalable/

STCP: some results

b	a	Rate CoV	Loss recovery time	Rate halving time	Rate doubling time
$\frac{1}{2}$	$\frac{2}{50}$	0.50	$17.7T_r$ (3.54s)	T_r (0.20s)	$17.7T_r$ (3.54s)
$\frac{1}{4}$	$\frac{1}{50}$	0.35	$14.5T_r$ (2.91s)	$2.41T_r$ (0.48s)	$35T_r$ (7.00s)
$\frac{1}{8}$	$\frac{1}{100}$	0.25	$13.4T_r$ (2.68s)	$5.19T_r$ (1.04s)	$69.7T_r$ (13.9s)
$\frac{1}{16}$	$\frac{1}{200}$	0.18	$12.9T_r$ (2.59s)	$10.7T_r$ (2.15s)	$139T_r$ (27.8s)

Number of flows	2.4.19 TCP	2.4.19 TCP with gigabit kernel modifications	Scalable TCP
1	7	16	44
2	14	39	93
4	27	60	135
8	47	86	140
16	66	106	142

Table 3: Number of 2 Gigabyte transfers completed in 1200 seconds.

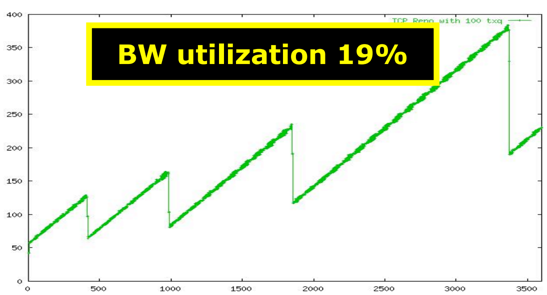
From 1st PFLDnet Workshop, Tom Kelly

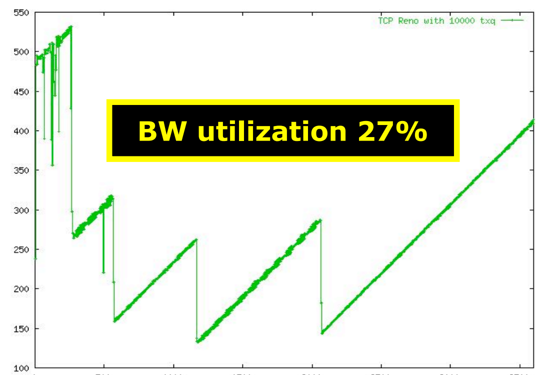
FAST TCP [Low04]

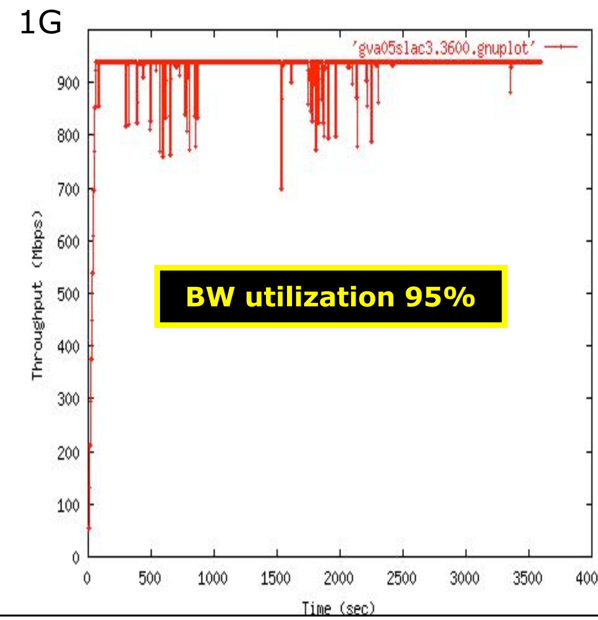
- Based on TCP Vegas
- Uses end-to-end delay and loss to dynamically adjust the congestion window
- AIMD reduces throughput

FAST TCP: some results

capacity = 1Gbps; 180 ms round trip latency; 1 flow





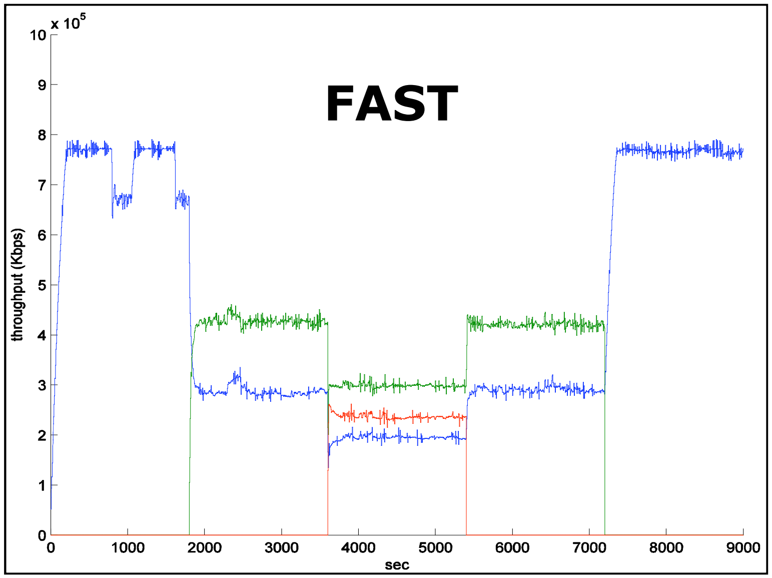


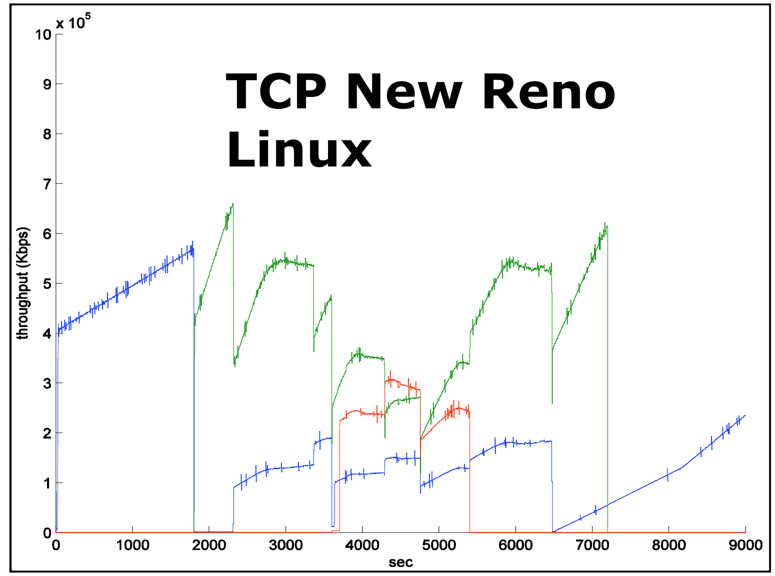
Linux TCP	Linux TCP (Optimized)	FAST

From Sylvain Ravot

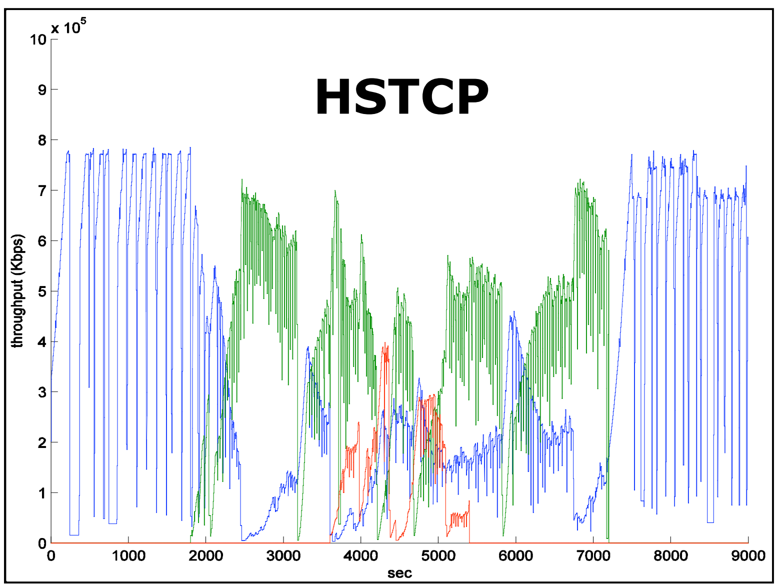
Beyond TCP

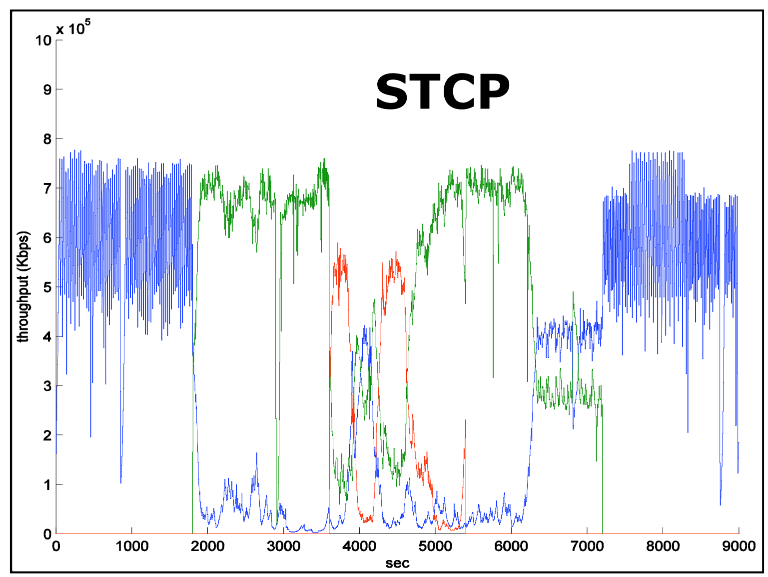
Comparisons





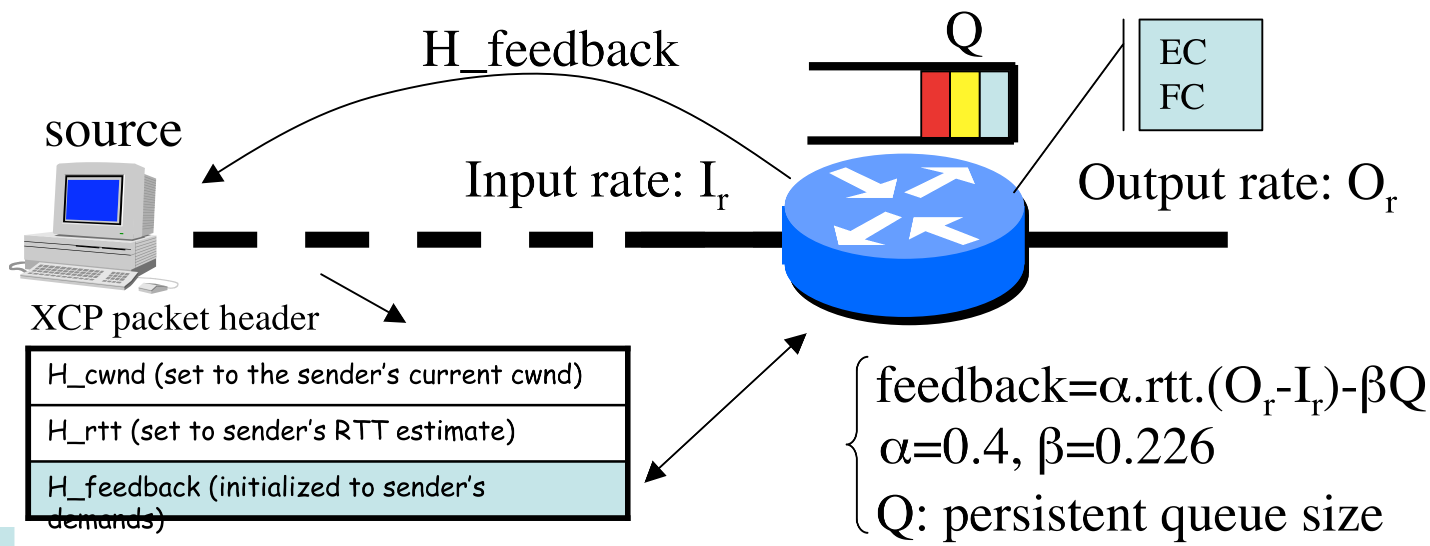
From Sylvain Ravot





XCP [Katabi02]

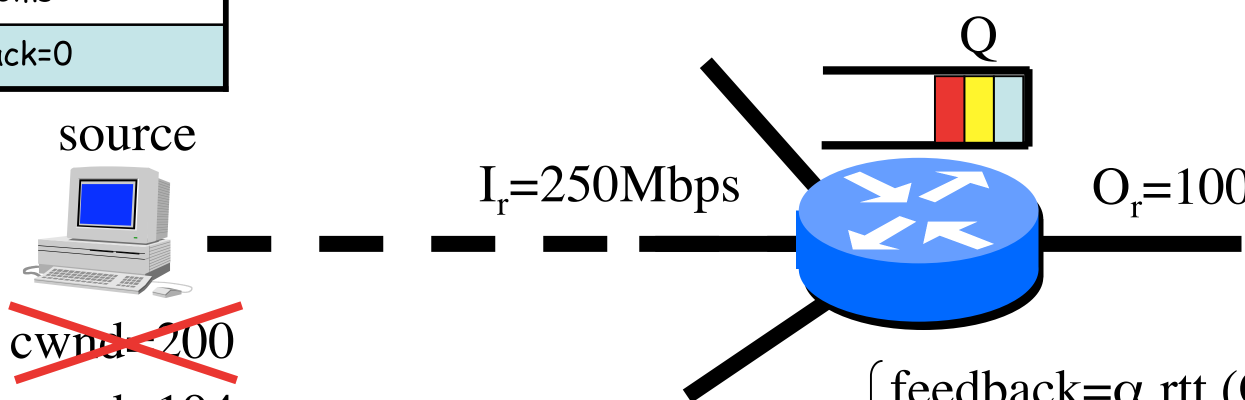
- XCP is a router-assisted solution, generalized the ECN concepts (FR, TCP-ECN)
- XCP routers can compute the available bandwidth by monitoring the input rate and the output rate
- Feedback is sent back to the source in special fields of the packet header



XCP in action

Feedback value represents a window increment/decrement

H_cwnd=200
H_rtt=100ms
H_feedback=0



~~cwnd=200~~

cwnd=194

H_cwnd=200
H_rtt=100ms
H_feedback=-6

feedback=α.rtt.(O_r-I_r)-βQ

α=0.4, β=0.226

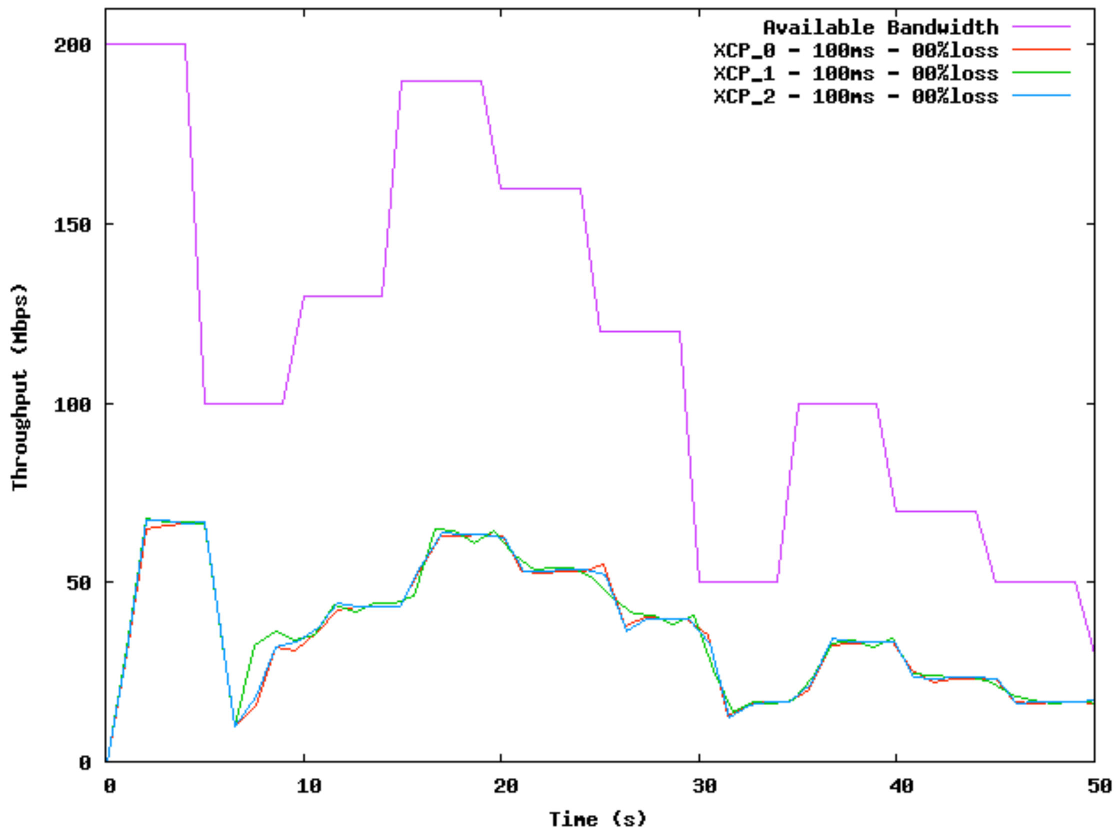
Q: persistent queue size

Case without βQ contribution

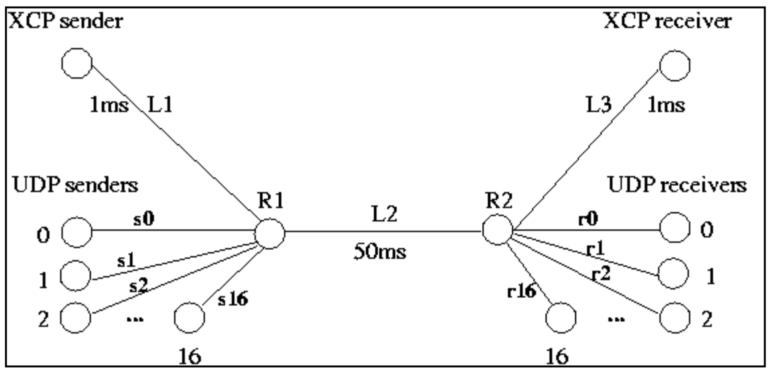
O_r-I_r=100-250=-150

feedback=-6

XCP
Variable bandwidth environments



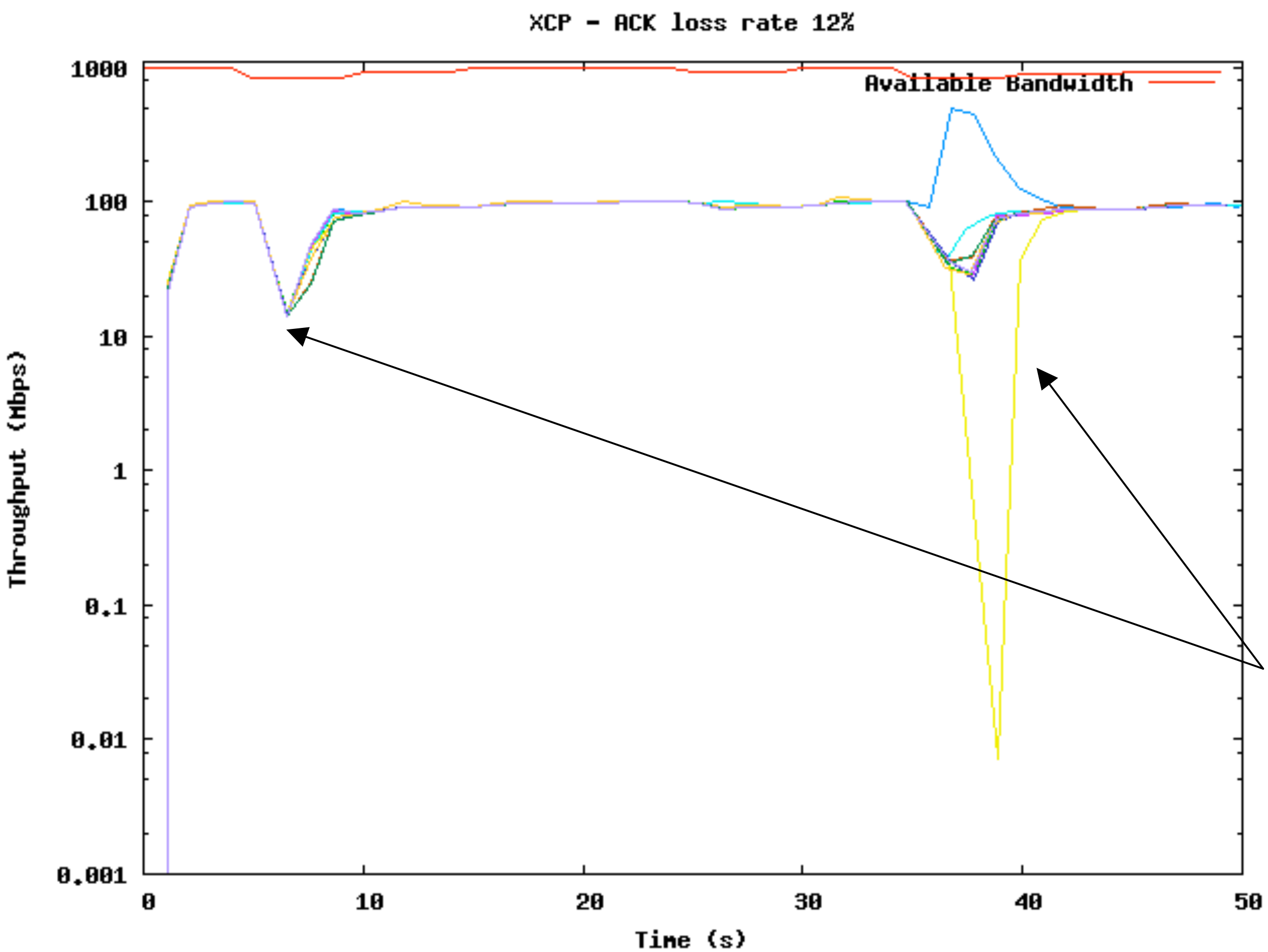
Good fairness and stability even in variable bandwidth environments



Beyond TCP

XCP-r [Pacheco&Pham05]

A more robust version of XCP

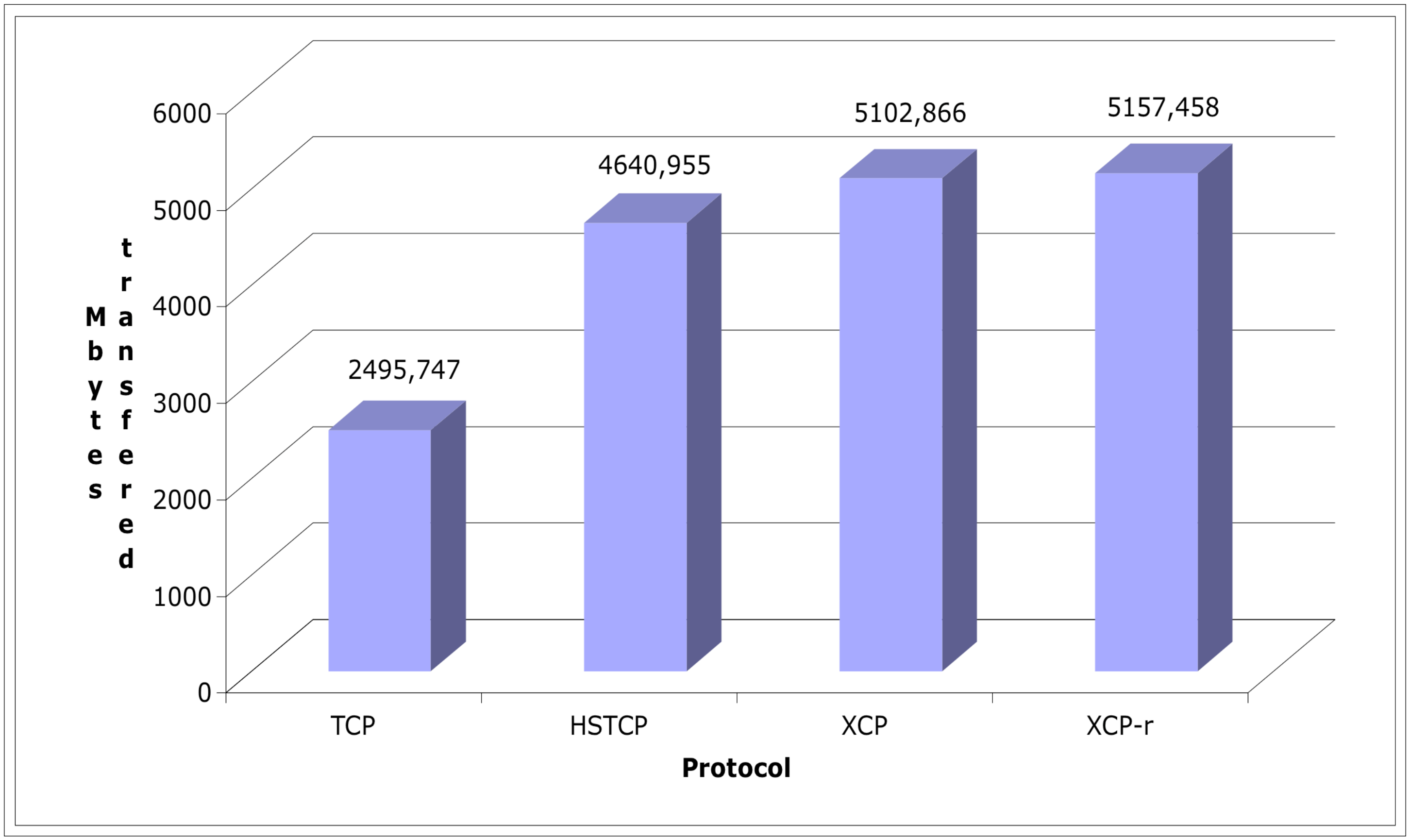


10 flows sharing a 1Gbps link

Fast recovery after the timeouts and better fairness level

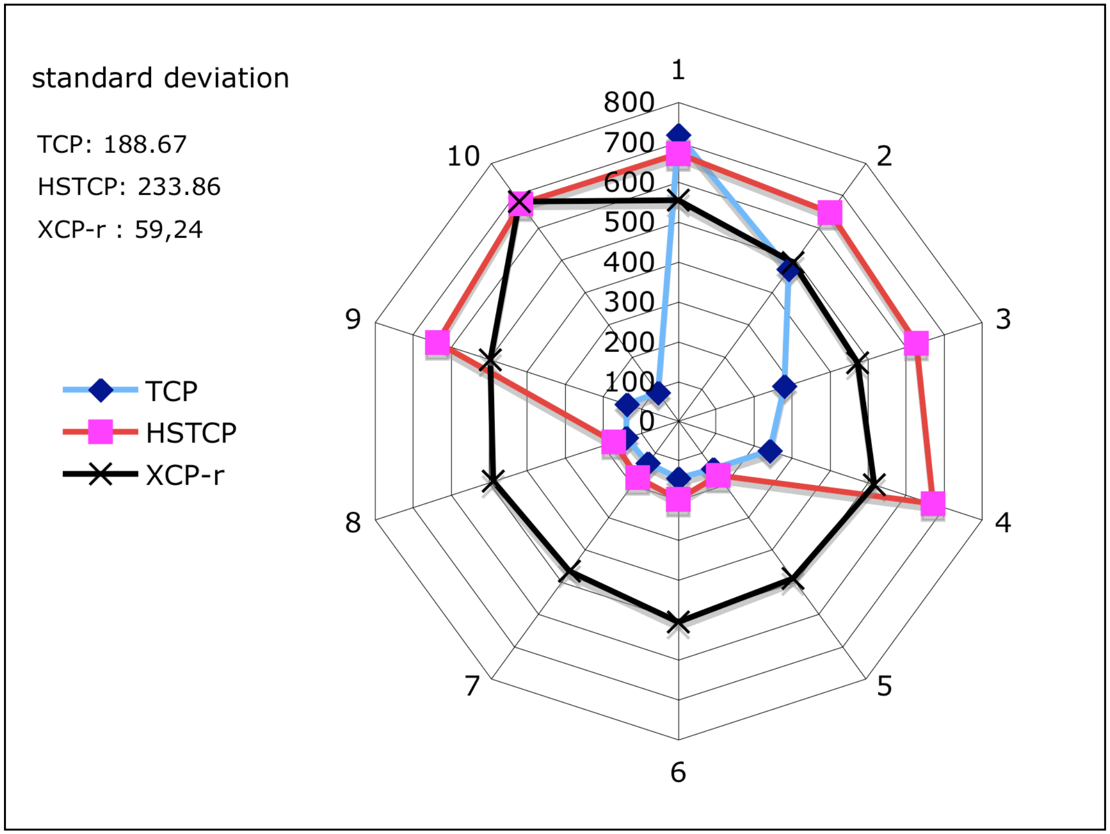
XCP-r performance

Amount of data transfered in 50s, 10 flows, 1Gbps link, 200ms RTT



XCP-r fairness

TCP and HSTCP are not really fair...



Nothing is perfect :-(

- Multiple or parallel streams
  - How many streams?
  - Tradeoff between window size and number of streams
- New protocol
  - Fairness issues?
  - Deployment issues?
  - Still too early to know the side effects

Where to find the new protocols?

- HSTCP
  - http://www.icir.org/floyd/hstcp.html
- STCP on Linux 2.4.19
  - http://www-lce.eng.cam.ac.uk/~ctk21/scalable/
- FAST
  - http://netlab.caltech.edu/FAST/
- XCP
  - http://www.ana.lcs.mit.edu/dina/XCP/
- BIC TCP on Linux 2.6.7
  - http://www.csc.ncsu.edu/faculty/rhee/export/bitcp/

Web100 project

- www.web100.org
- « The Web100 project will provide the software and tools necessary for end-hosts to automatically and transparently achieve high bandwidth data rates (100 Mbps) over the high performance research networks »
- Actually it's not limited to 100Mbps!
- Recommended solution for end-users to deploy and test high-speed transport solutions