TCP and beyond: Protocols for reliable transfer in high-bandwidth long-distance networks

Petr Holub

hopet@ics.muni.cz

Overview

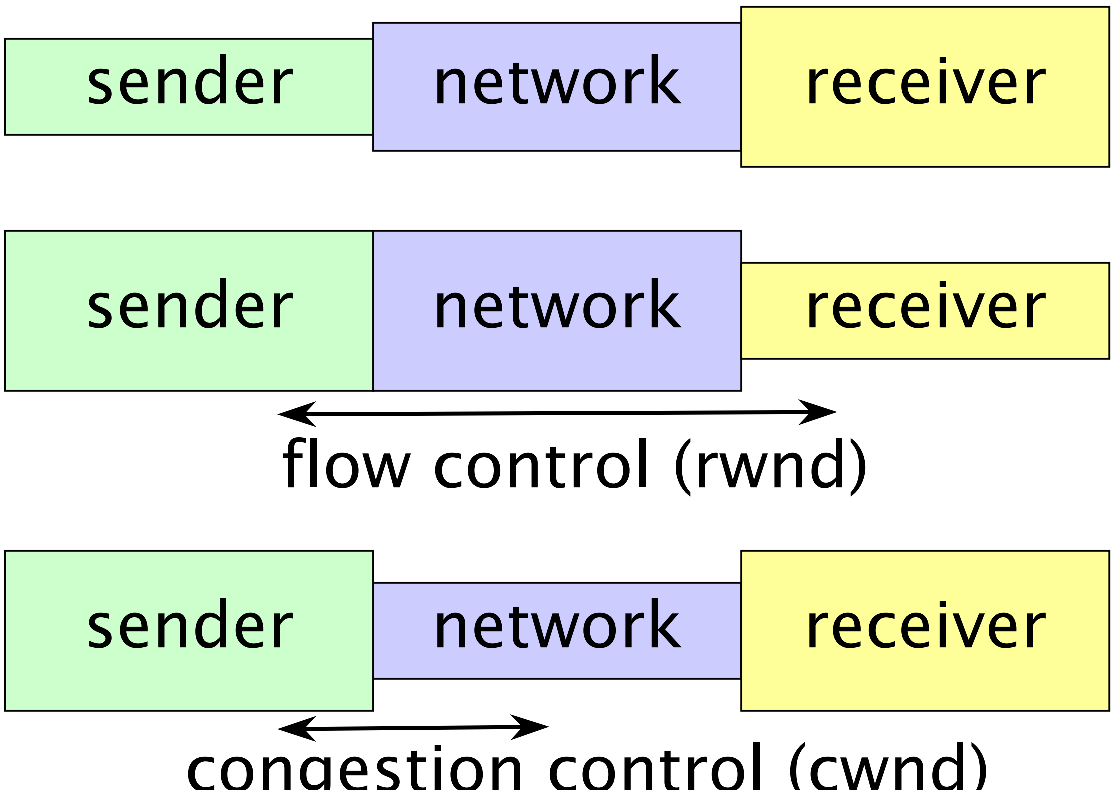
- Traditional TCP and its problems
  - multiple stream TCP
- Improvements to TCP
  - Scalable TCP
  - HS TCP
- TCP improvements based on additional information provided by the network
  - QuickStart
  - E-TCP
- Non-TCP approaches

Reliable transfer protocols

- ensuring reliability of transfer
  - retransmission of lost data
- overload prevention
  - for both network and receiver
- behavior assessment
  - aggressiveness – utilization of bandwidth available
  - responsiveness – loss recovery capability
  - fairness – obtaining fair share of bandwidth for multiple network participants
- the problem we have: fat *long* pipes

Traditional TCP

- flow control vs. congestion control



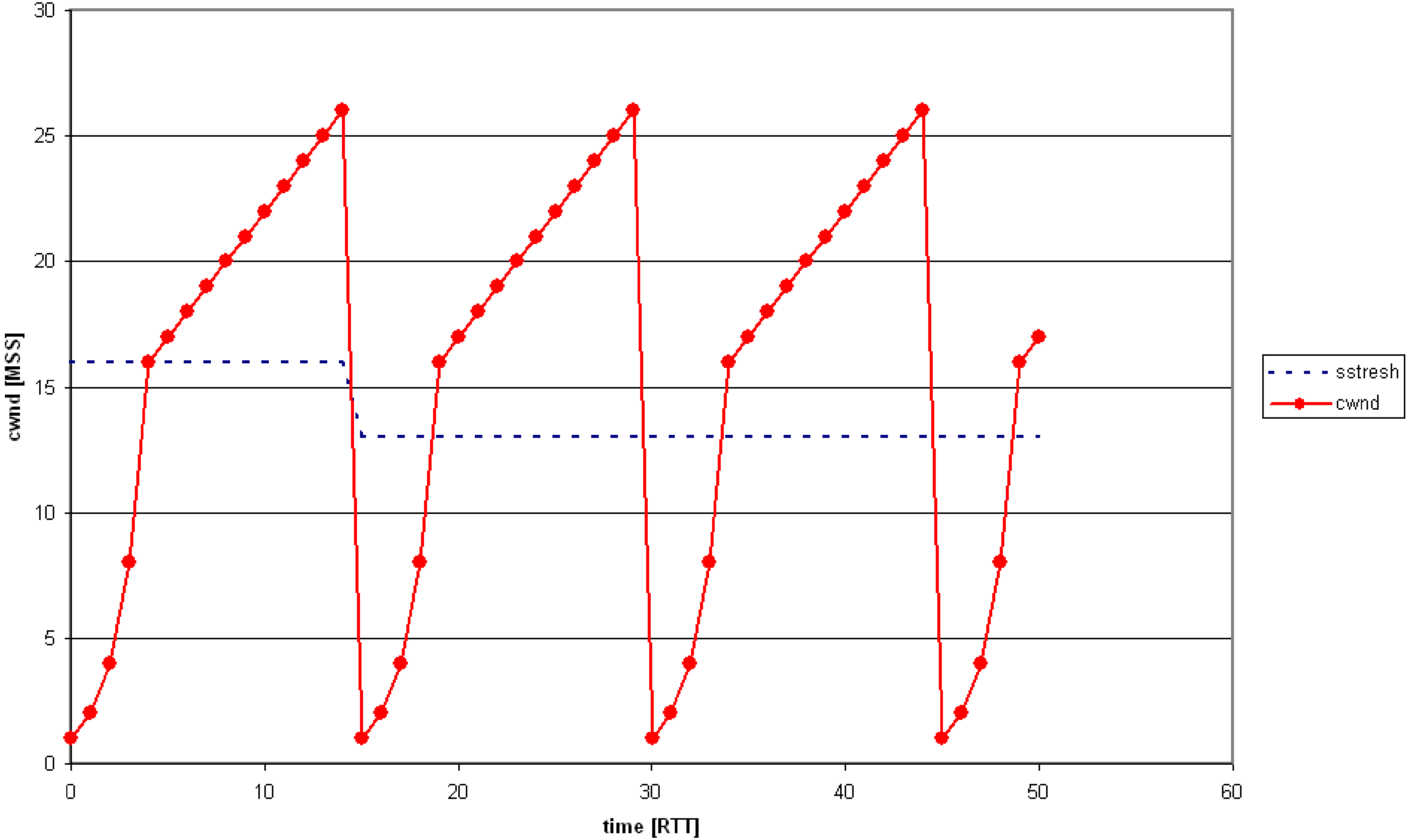
Traditional TCP (3)

- Flow control
  - deterministic, precise (rwnd)
- Congestion control
  - rough estimate
- ownd = min(cwnd,rwnd)
  - *bw=owin* 8*packet_size/ rtt*

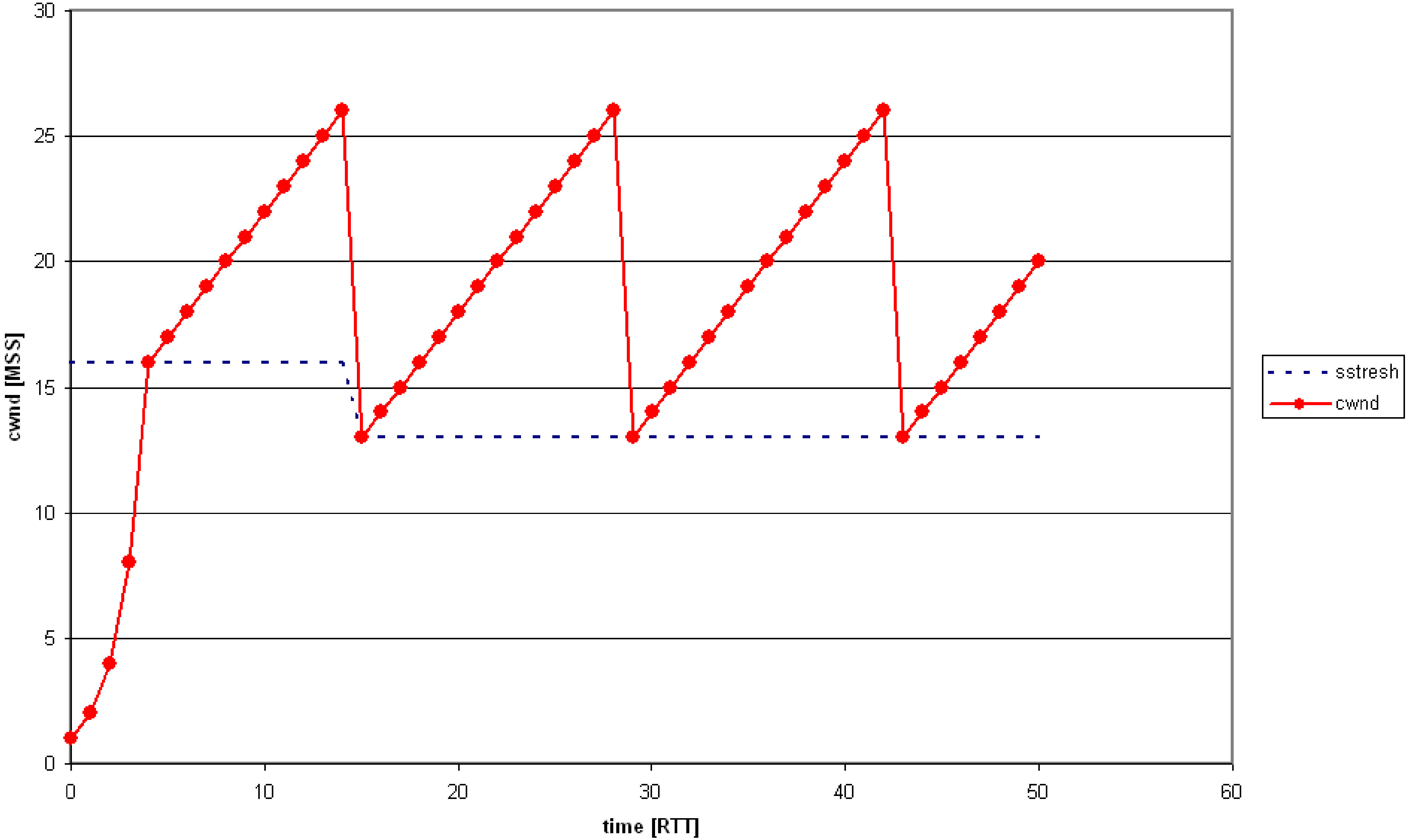
Traditional TCP (4)

- Congestion control
  - traditionally based on AIMD (additive increase multiplicative decrease)
    - *cwnd += 1 MSS*
      per successful RTT when above *sstresh*
    - *cwnd *= .5*
      on each loss event
    - Reno: fast retransmission (loss detected by receiving 3 duplicated ACKs) and fast recovery (canceling slowstart)

Tahoe


30
25
20
15
10
5
0
cwnd [MSS]
0
10
20
30
40
50
60
time [RTT]
sstresh
cwnd

Reno



30
25
20
15
10
5
0
cwnd [MSS]
0
10
20
30
40
50
60
time [RTT]
sstresh
cwnd

Traditional TCP (5)

- TCP Vegas
  - trying to avoid congestion by monitoring RTT
  - if RTT increases (suggesting that congestion is imminent) it decreases cwnd linearly
- measurement of available bandwidth based on inter-packet spacing

Traditional TCP (6)

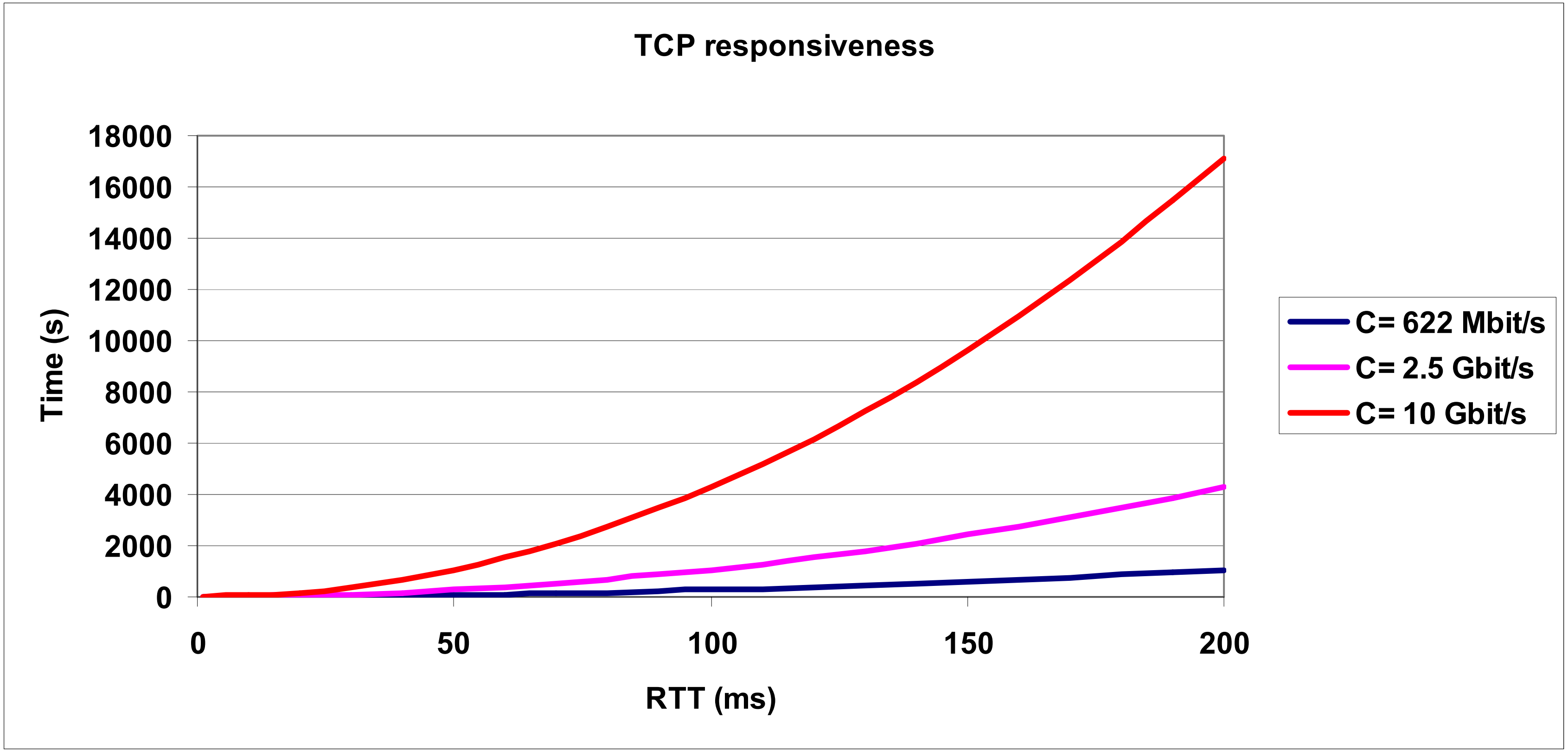
- Reaction to loss
  - TCP Tahoe (whole current windows (owin))
  - TCP Reno (one segment in „Fast Retransmit” mode)
  - TCP NewReno (more segments in “Fast Retransmit” mode)
  - TCP SACK (lost packets only)
- Issue of large enough *cwnd* for fast long distance networks

TCP Reponse function

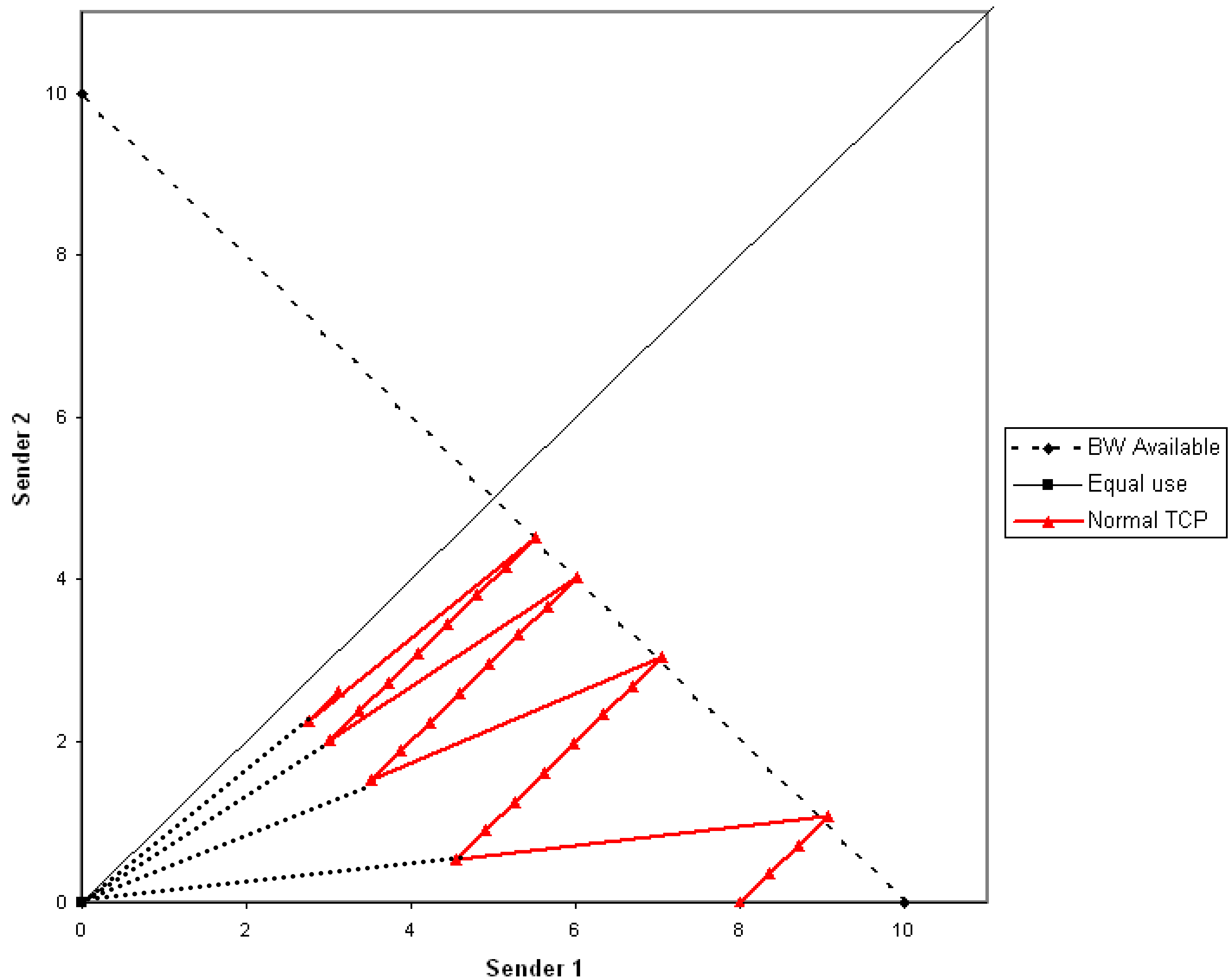
- relating throughput *bw* (or window size *owin*) and steady-state packet loss rate *p*
  - owin ~ 1.2/sqrt(p)
  - bw = 8*MSS*owin/RTT
  - bw = (8*MSS/RTT)*1.2/sqrt(p)
- Traditional TCP responsiveness
  - assume that packet loss is experienced when *cwnd = bw*RTT*

$$\rho = \frac{\mathbf{bw \, . \, RTT^2}}{\mathbf{2 \, . \, MSS}}$$

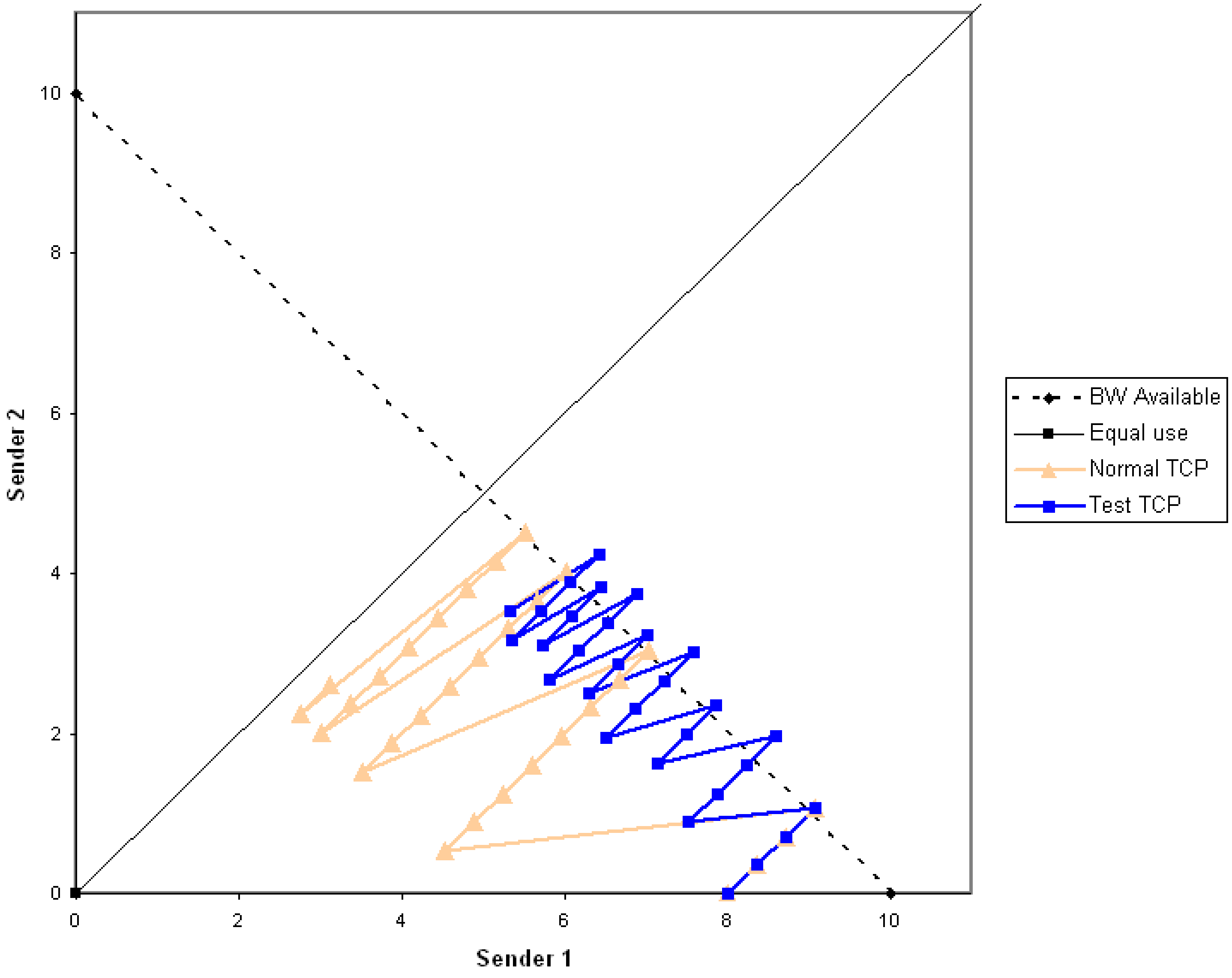
Traditional TCP responsivness



TCP – fairness



TCP – fairness (2)



Some remarks to fairness

- assessment of fairness
  - for streams with varying RTT
  - for streams with different MTU

Going multi-stream

- improves performance when single packet loss occurs
- multiple packet loss can influence all streams
- when multiple
- many real applications in use: bbftp, GridFTP, Internet Backplane Protocol

Going multi–stream (2)

- disadvantages of multiple streams
  - more complicated (usually requires several threads)
  - startup and shutdown times are not improved substantially
  - may result in synchronous overloading of router buffers

Possible implementation improvements

- Cooperation with hardware
  - TCP Checksum Offloading (both Rx and Tx)
- Zero copy
  - networking usually involves several copies: from userland process to kernel and from kernel to NIC and vice versa in case of receiving data
  - page flipping (moving pages from user to kernel space and vice versa
  - Implementations for Linux, FreeBSD, and Solaris

Web100

- TCP instrumentation for Linux kernel and userland interfaces
  - interface for monitoring kernel parameters connected with networking stack and overall performance
  - number of tunable parameters
- advanced auto-tuning support

Web100 (2)

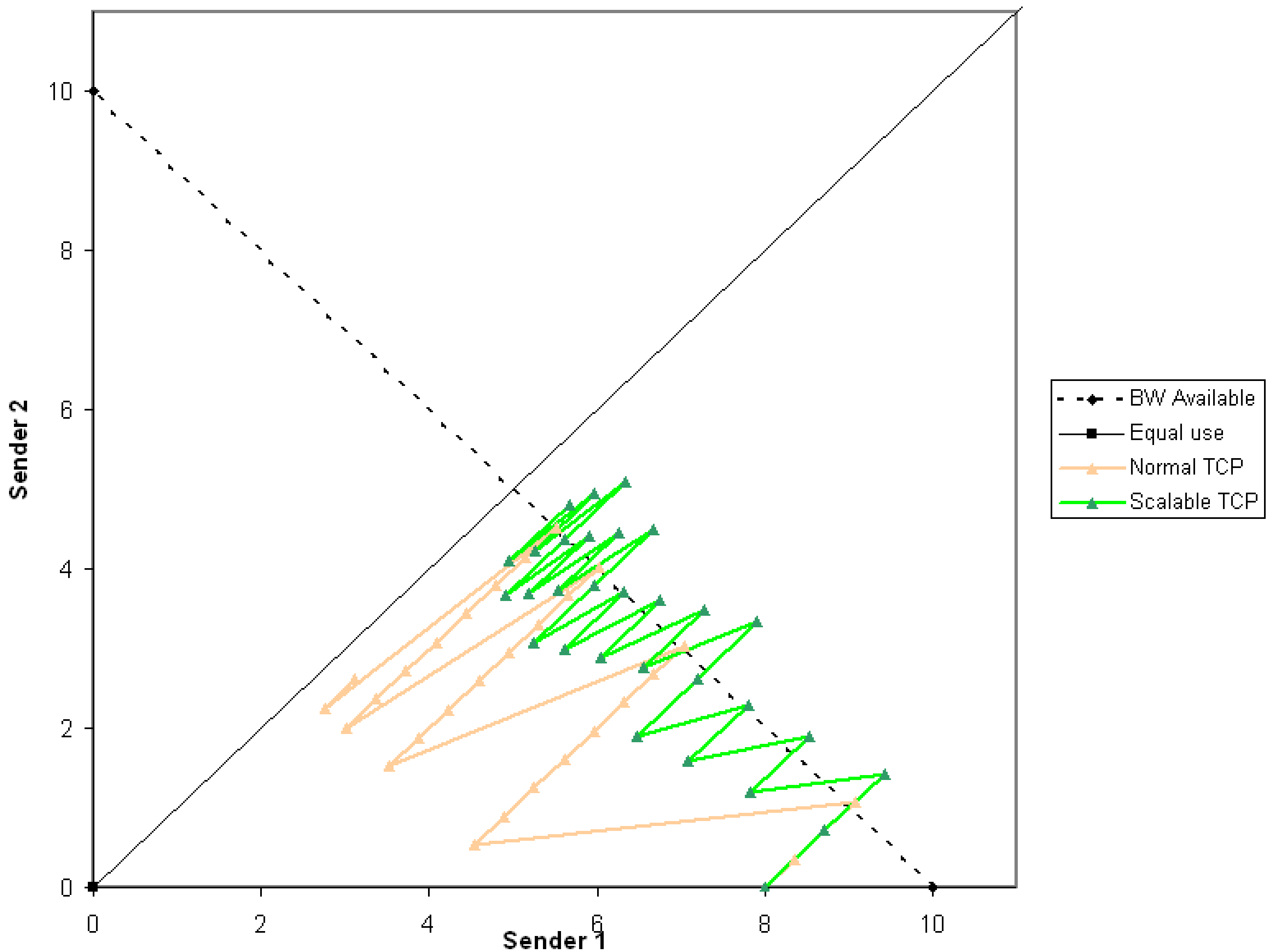
- Kernel instrumentation set (TCP-KIS)
  - instrumentation inside kernel
  - monitoring various kernel structures relevant to TCP
  - currently more than 125 “instruments”
  - exposing information via /proc
- Web100 library provides access to variables/instruments
- Userland utilities (both command-line and GUI)

Beyond the Traditional TCP

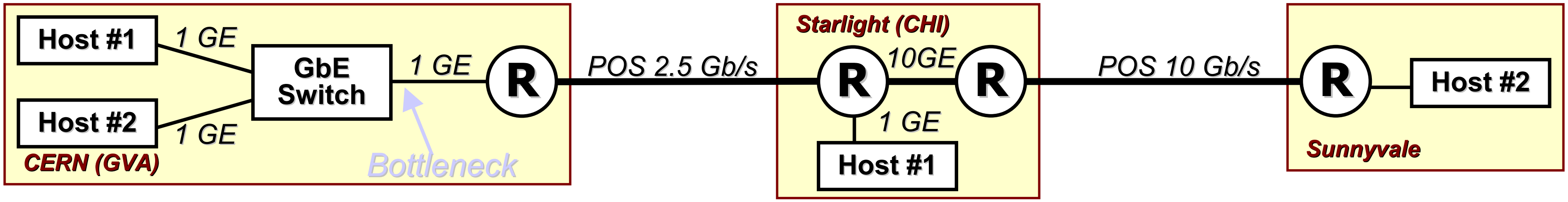
GridDT

- correction of *sstresh*
- modification of congestion control
  - *cwnd += a*
    for each RTT without packet loss
  - *cwnd *= b * cwnd*
    on each loss event
- faster slowstart
- modification of sender's stack only

GridDT fairness



GridDT example



TCP Reno performance (see slide #8):

First stream GVA <-> Sunnyvale : RTT = 181 ms ; Avg. throughput over a period of 7000s = 202 Mb/s

Second stream GVA<->CHI : RTT = 117 ms; Avg. throughput over a period of 7000s = 514 Mb/s

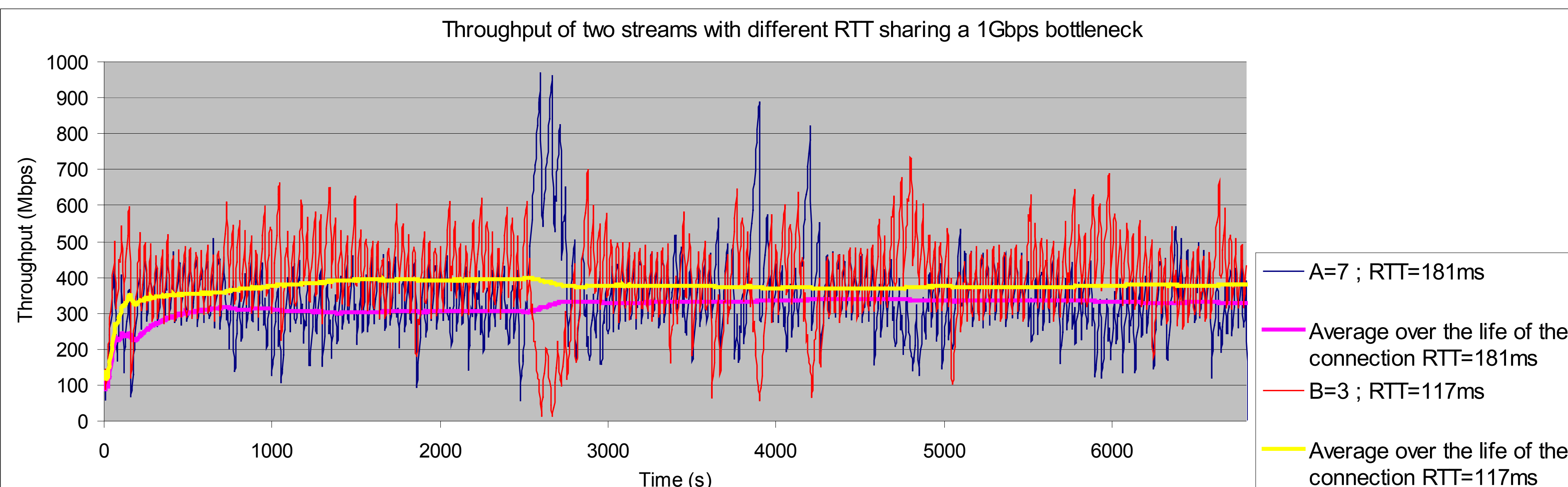
Links utilization 71,6%

Grid DT tuning in order to improve fairness between two TCP streams with different RTT:

First stream GVA <-> Sunnyvale : RTT = 181 ms, Additive increment = A = 7 ; Average throughput = 330 Mb/s

Second stream GVA<->CHI : RTT = 117 ms, Additive increment = B = 3 ; Average throughput = 388 Mb/s

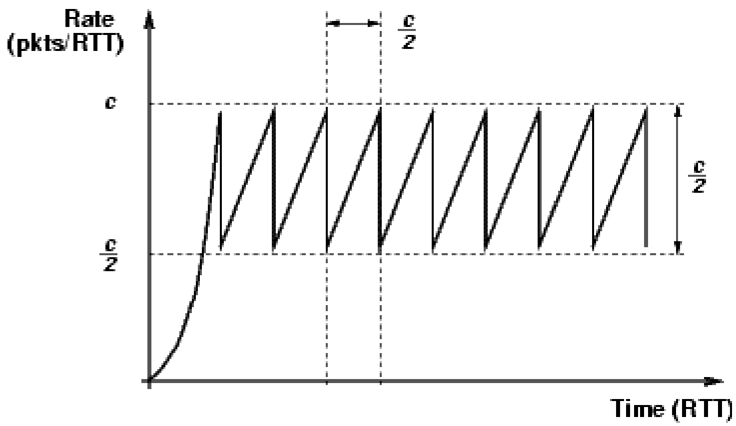
Links utilization 71.8%

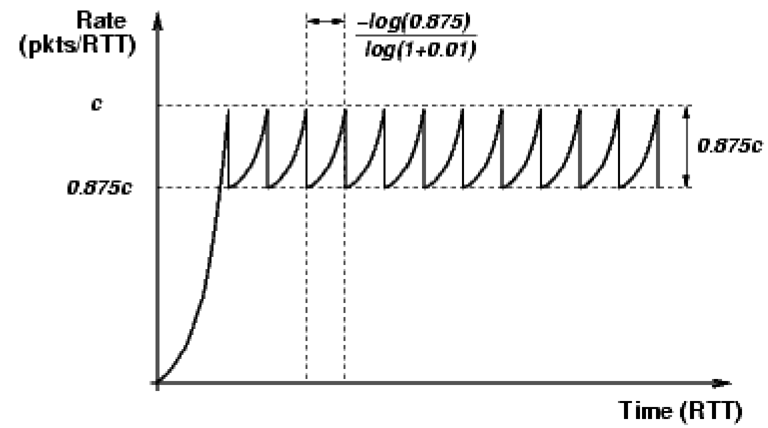


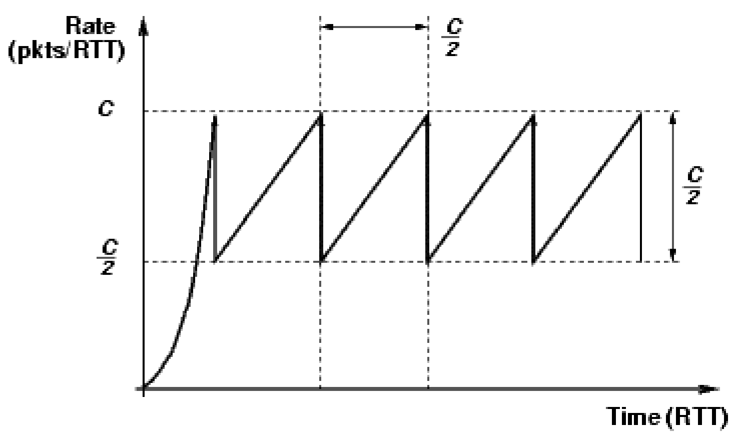
Scalable TCP

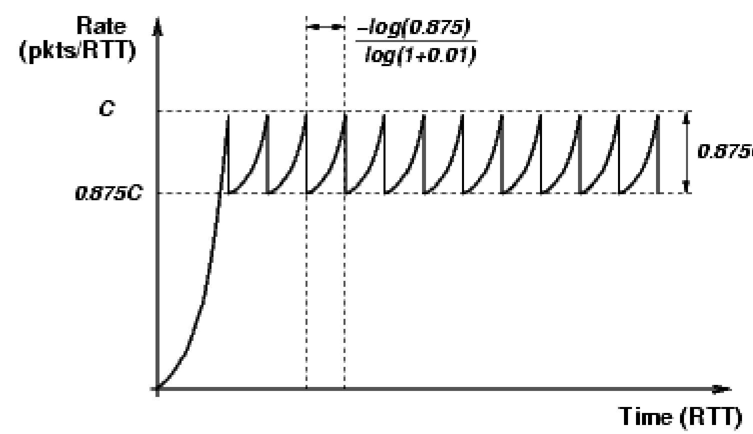
- Modification of congestion control mechanism
  - *cwnd += 0.01 * cwnd*
    ... for each RTT without loss
  - *cwnd = 0.875 * cwnd*
    ... on each loss event
  - with constant number of RTTs between losses independently of bandwidth
- no longer AIMD => MIMD
- switches to AIMD for smaller window size and occurrence of more losses

Scalable TCP (2)

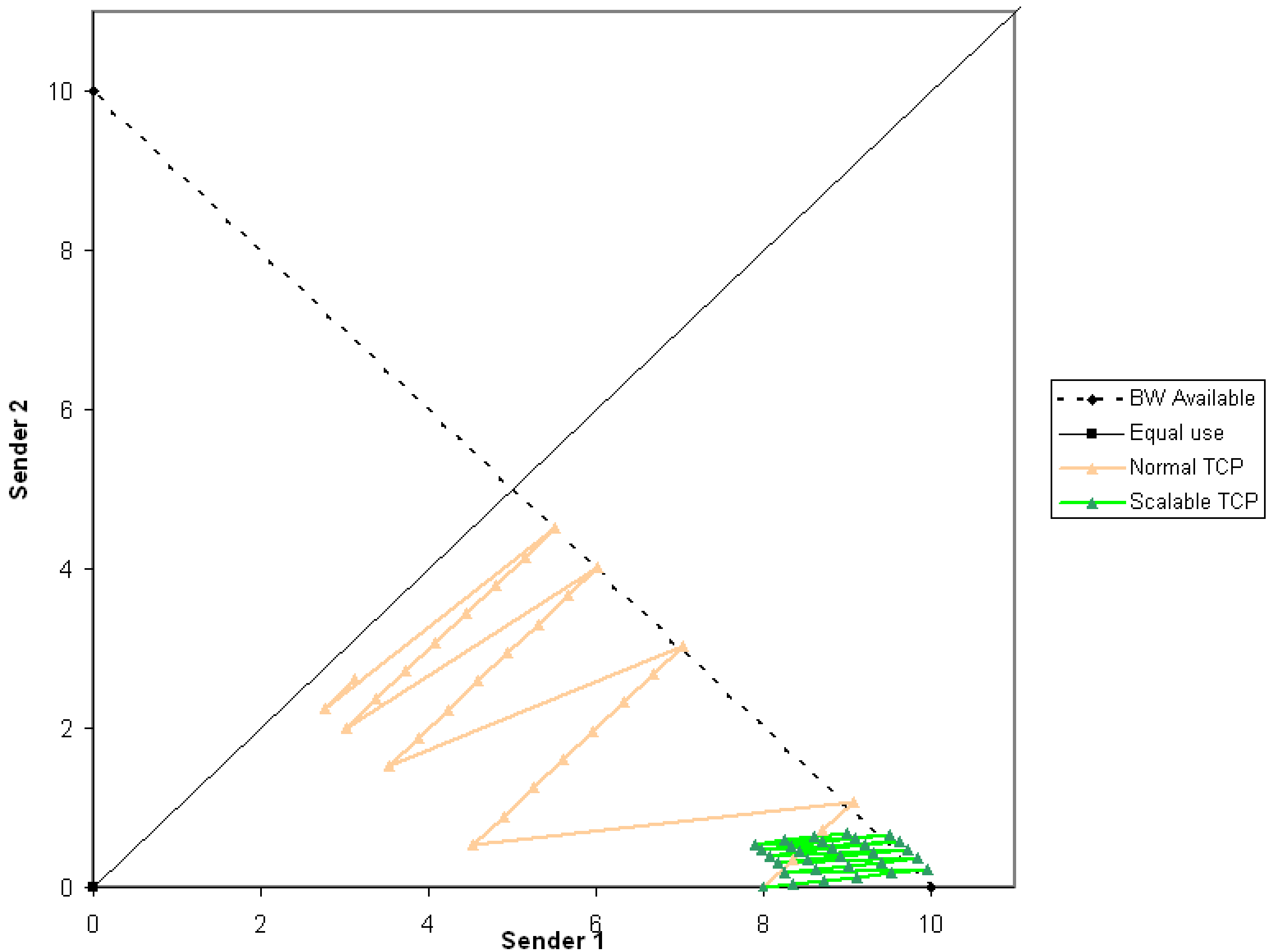




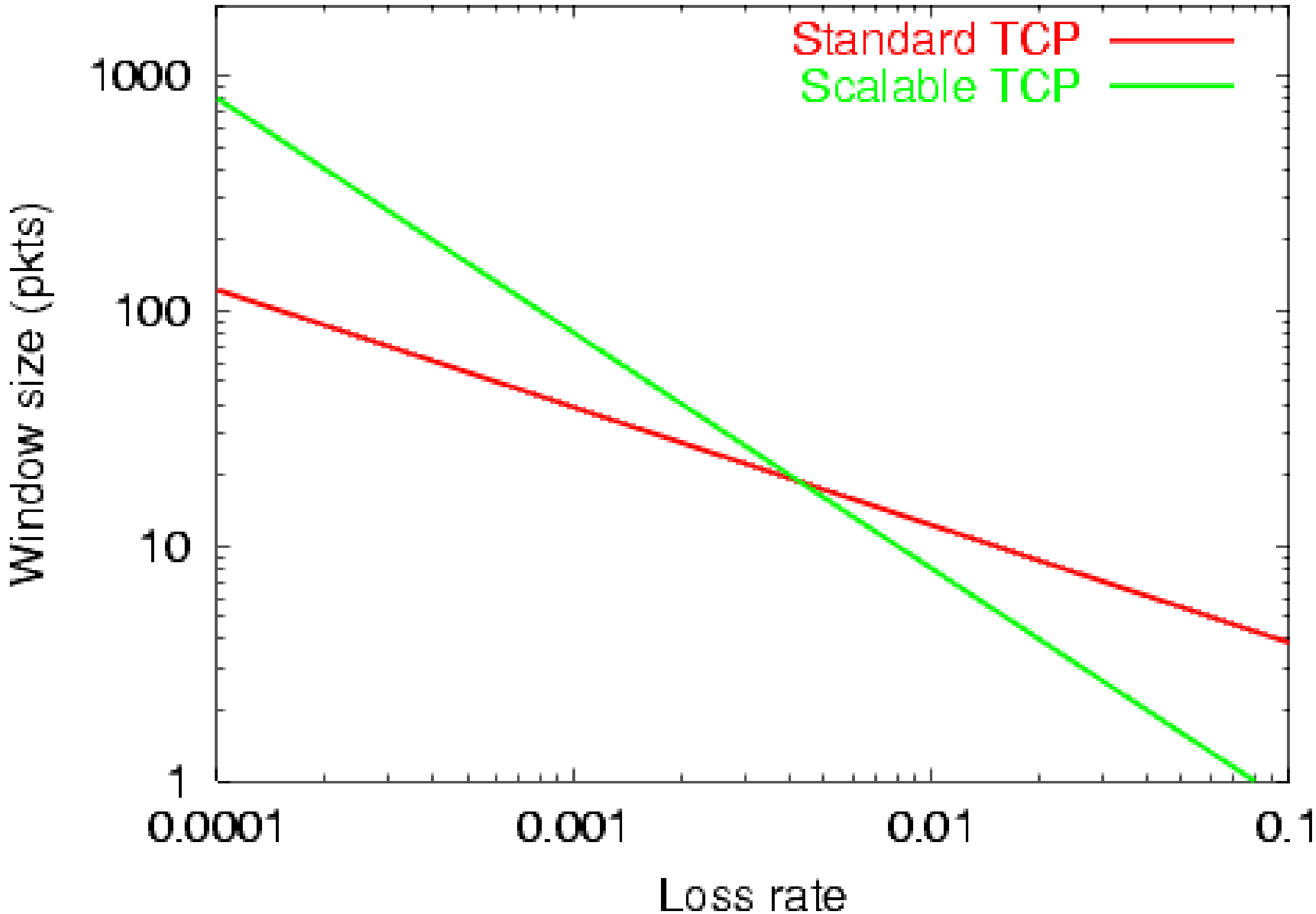




Scalable TCP – fairness



Scalable TCP – response curve



Scalable TCP

- http://www-lce.eng.cam.ac.uk/~ctk21/scalable/
- Tom Kelly, *Scalable TCP: Improving Performance in Highspeed Wide Area Networks* Submitted for publication, December 2002. http://www-lce.eng.cam.ac.uk/~ctk21/papers/scalable_improve_hswan.pdf

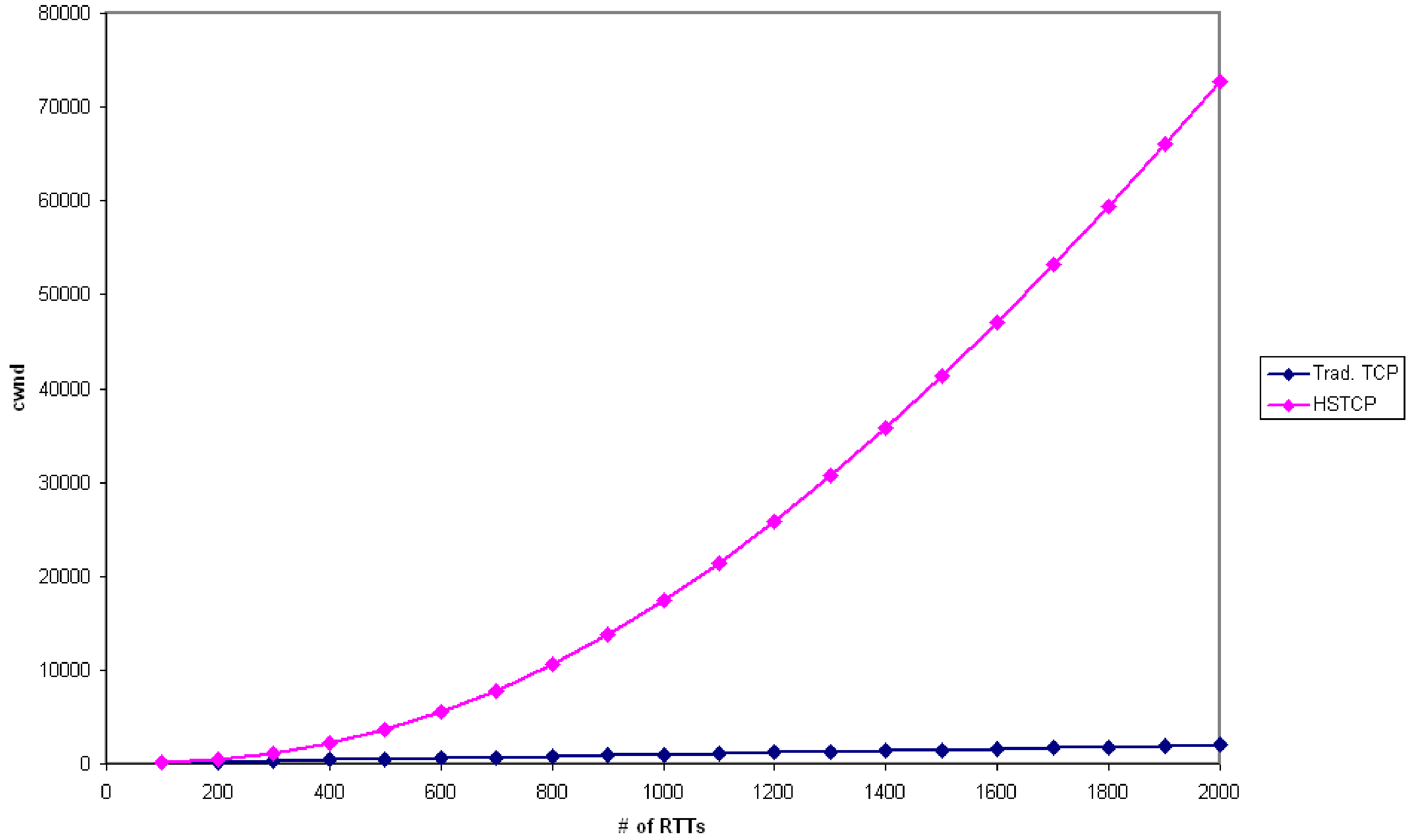
HSTCP

- Emulates behavior of standard TCP for congested (=high packet loss rate) and low bandwidth networks
- Modification of congestion control mechanism
  - *cwnd += a(cwnd)*
    ... for each RTT without loss
  - *cwnd = (1-b(cwnd)) * cwnd*
    ... on each loss event
- RFC 3649

HSTCP (2)

- suggested parameterization
  - b(w) ~ −0.4 * (log(w) − 3.64) / 7.69 + 0.5
  - a(w) ~ (2*w^2*b(w))/((2−b(w))*w^1.2*12.8)
- possible Linear High Speed equivalent to Scalable TCP
- comparison with multiple streams
  - N(W) ~ 0.23 * W^0.4

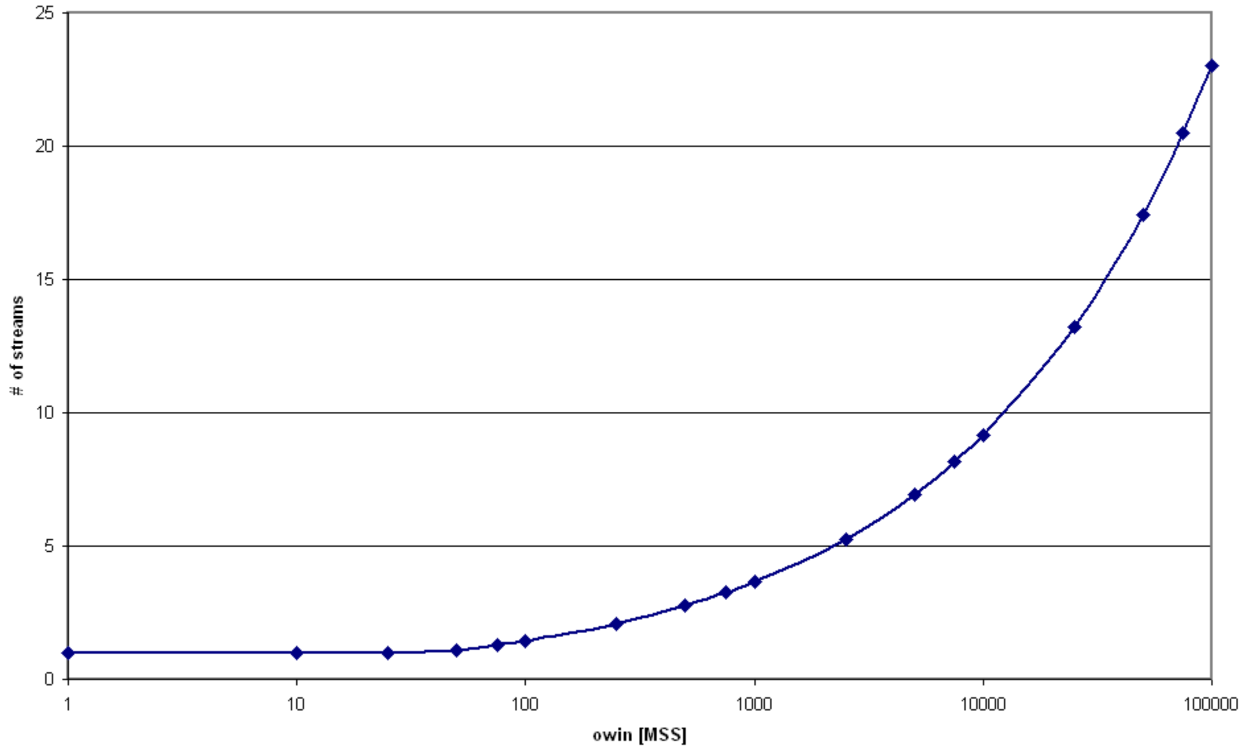
cwnd: Traditional TCP vs. HSTCP



HSTCP (2)

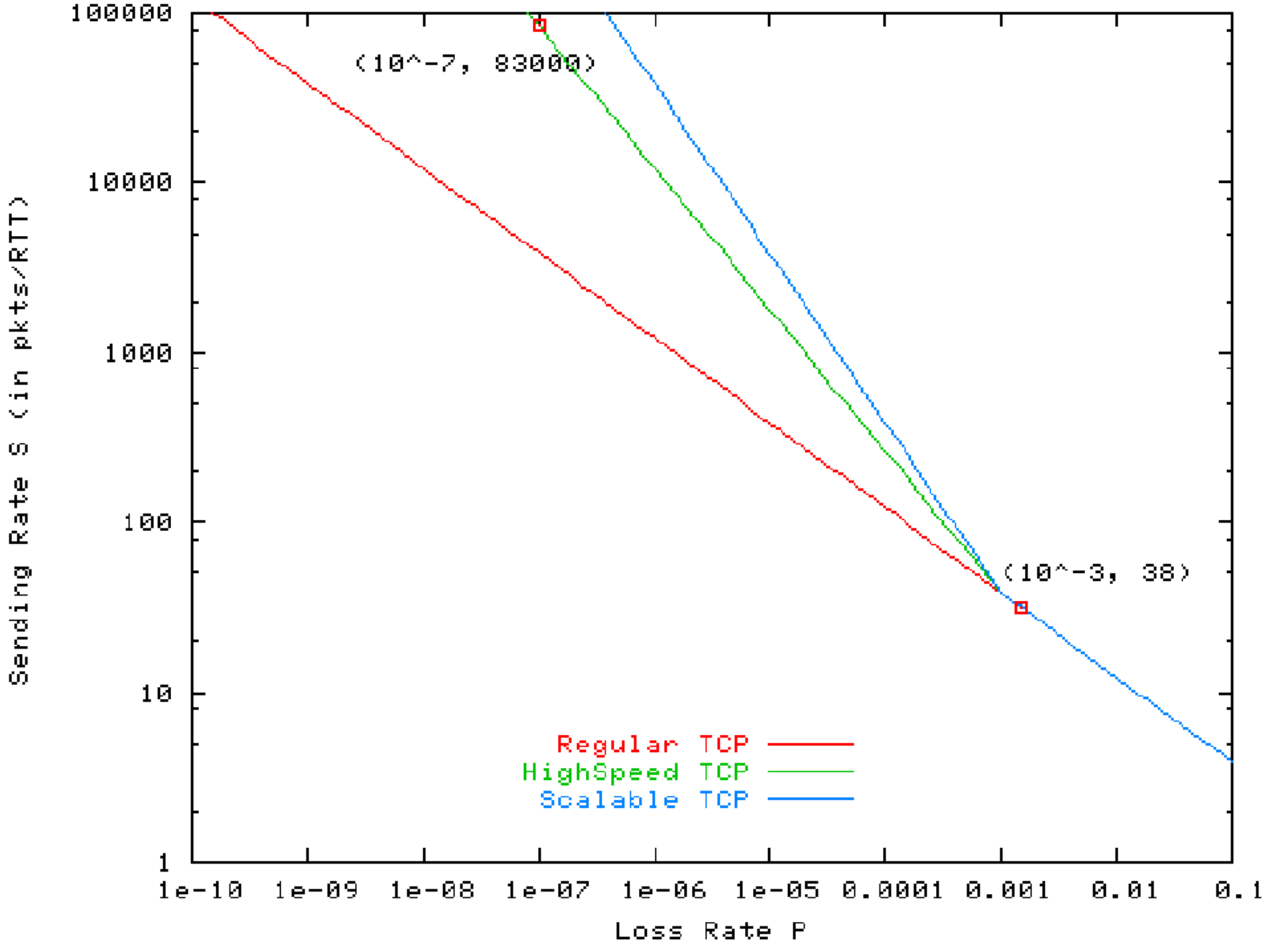
- suggested parameterization
  - b(w) ~ -0.4 * (log(w) - 3.64) / 7.69 + 0.5
  - a(w) ~ (2*w^2*b(w))/((2-b(w))*w^1.2*12.8)
- possible Linear High Speed equivalent to Scalable TCP
- comparison with multiple streams
  - N(W) ~ 0.23 * W^0.4

HSTCP (3)



- neither ScalableTCP nor HSTCP handles slowstart phase in some advanced way...

HSTCP (4)



100000
10000
1000
100
10
1
Sending Rate S (in pkts/RTT)
(10^-7, 83000)
(10^-3, 38)
Regular TCP
HighSpeed TCP
Scalable TCP
1e-10
1e-09
1e-08
1e-07
1e-06
1e-05
0.0001
0.001
0.01
0.1
Loss Rate P

Quickstart or Limited Slowstart

- strong suspicion there is no reasonable way for improving slowstart phase without interaction with the network
- proposes 4 byte option for IP header comprising two fields: `QS TTL` and `Initial rate`

Quickstart (2)

- Sender willing to use QS sets QS TTL to random value and Initial rate to desired value and sends SYN packet.
- All routers on the way to receiver that understand and approve QS decrement QS TTL by 1 and decrease Initial rate if needed.

Quickstart (3)

- Receiver sends feedback in SYN/ACK packet so sender knows if all the routers on the way participated, has RTT measurement.
- Sender sets initial adequate congestion window and than uses AIMD as usual.
- *Requires changing IP layer* :-((

E–TCP

- Early Congestion Notification
  - the bit is set by routers before reaching line/buffer saturation
  - ECN flag must be reflected by receiver
  - TCP was excepted to react in the same way as to congestion
- E–TCP
  - suggest to reflect ECN flag just once
  - freeze cwnd when ECN marked ACK arrives

E–TCP (2)

- requires modification of Active Queue Management to allow small losses to reintroduce multiplicative decrease for fair temporal behavior
- problems
  - with ECN: most of routed admins don't bother to configure it
  - with AQM: the same as above
  - change ECN behavior on receivers

Other protocols

- FAST – Fast AQM Scalable TCP
  - uses end-to-end delay, ECN, and loss as congestion avoidance/detection
  - http://netlab.caltech.edu/FAST/
    http://netlab.caltech.edu/pub/papers/FAST-infocom2004.pdf

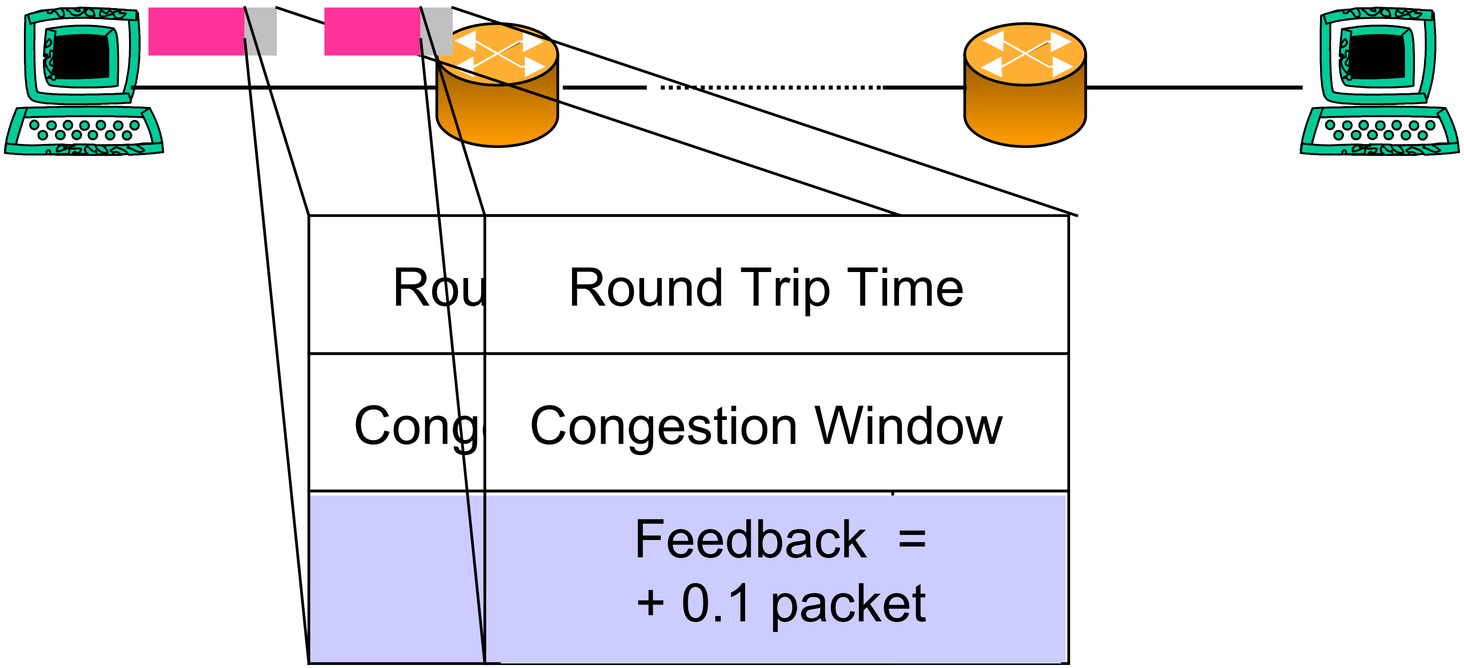
Non-TCP based protocols

tsunami

- TCP connection for out-of-band control channel
  - transfer parameters negotiation, retransmission requests, end of transmission negotiation
  - NACKs instead of ACKs
- UDP data channel
  - exponential increase, exponential back-off
  - highly tunable: speedup/slowdown factors, error threshold, maximum retransmission queue, retransmission request interval.
- http://www.anml.iu.edu/anmlresearch.html

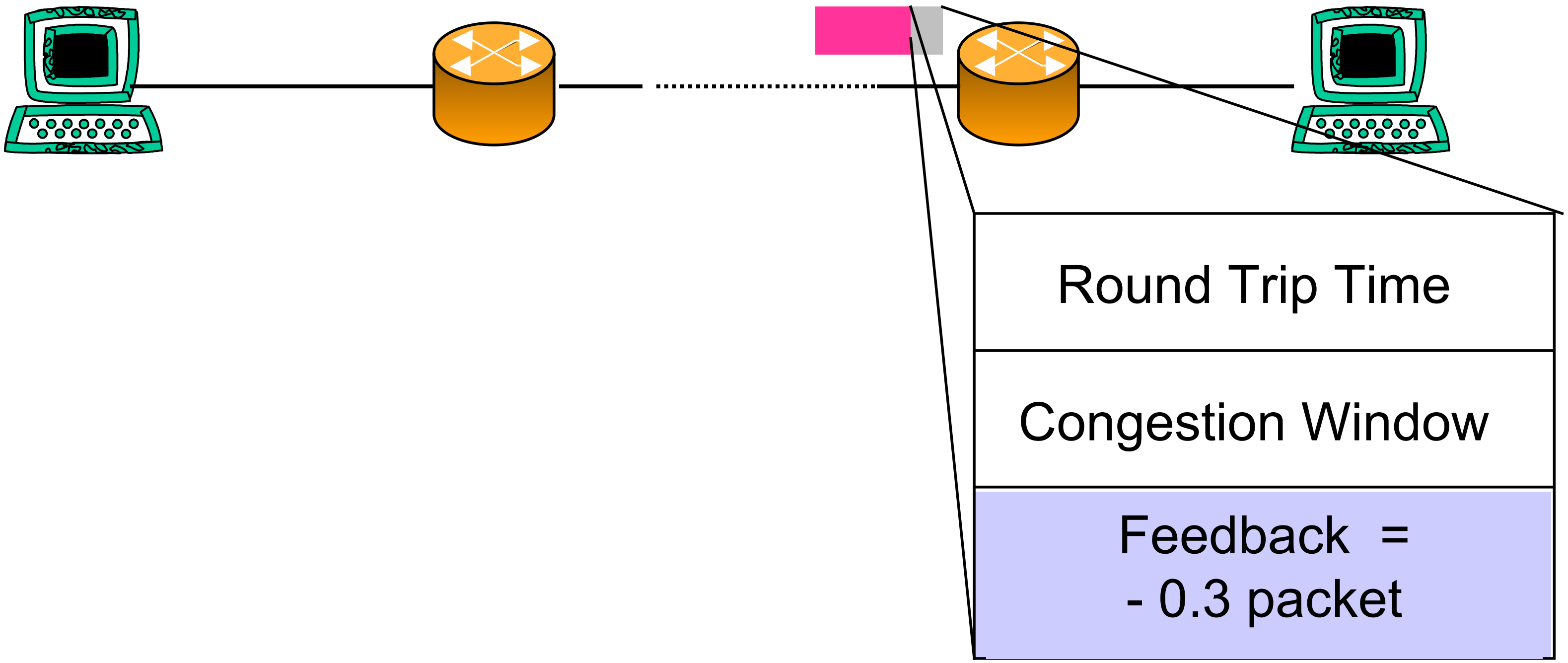
XCP

- per packet feedback from routers

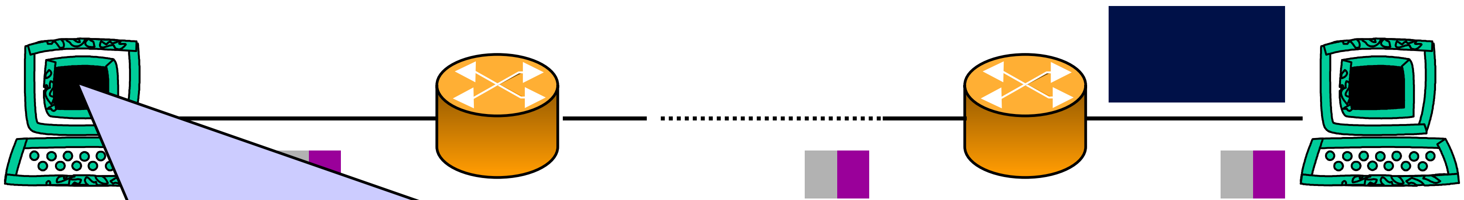


Congestion Header

XCP (2)



XCP (3)



Congestion Window = Congestion Window + Feedback

Other approaches

- SCTP
  - multistreaming, multi-homed transport
  - http://www.sctp.org/
- DCCP
  - unreliable protocol with congestion control mechanisms
  - http://www.ietf.org/html.charters/dccp-charter.html
  - http://www.icir.org/kohler/dcp/

- STP
  - simple protocol easily implemented in hw; no sophisticated congestion etc.
  - http://lwn.net/2001/features/OLS/pdf/pdf/stlinux.pdf
- Reliable UDP
  - ensures reliable and in-order delivery
  - why??? Cisco folks needed some job perhaps...
  - http://www.watersprings.org/pub/id/draft-ietf-sigtran-reliable-udp-00.txt
- XTP and bunch of other guys...

Concluding remarks

- Current status
  - multi-stream TCP in heavy use in Grid computing
  - when going for improving TCP, {HS,Scalable}TCP et al. provide least dangerous way to go
  - use of aggressive protocols in private virtual circuits based e.g. on lambdas (CA*net4 already supports establishing these at user request!)

Concluding remarks (2)

- Interactions with link layer
  - wireless with varying delay and throughput
  - optical burst switching
- Flow-specific state in routers
  - flow-specific maring or dropping
  - may help also finishing short and high-bw transmissions
  - scalability & cost :-(

Other References

- RFC 3426 General Architectural and Policy Considerations

Thank you for your attention!