

INF5071 – Performance in Distributed Systems



Protocols

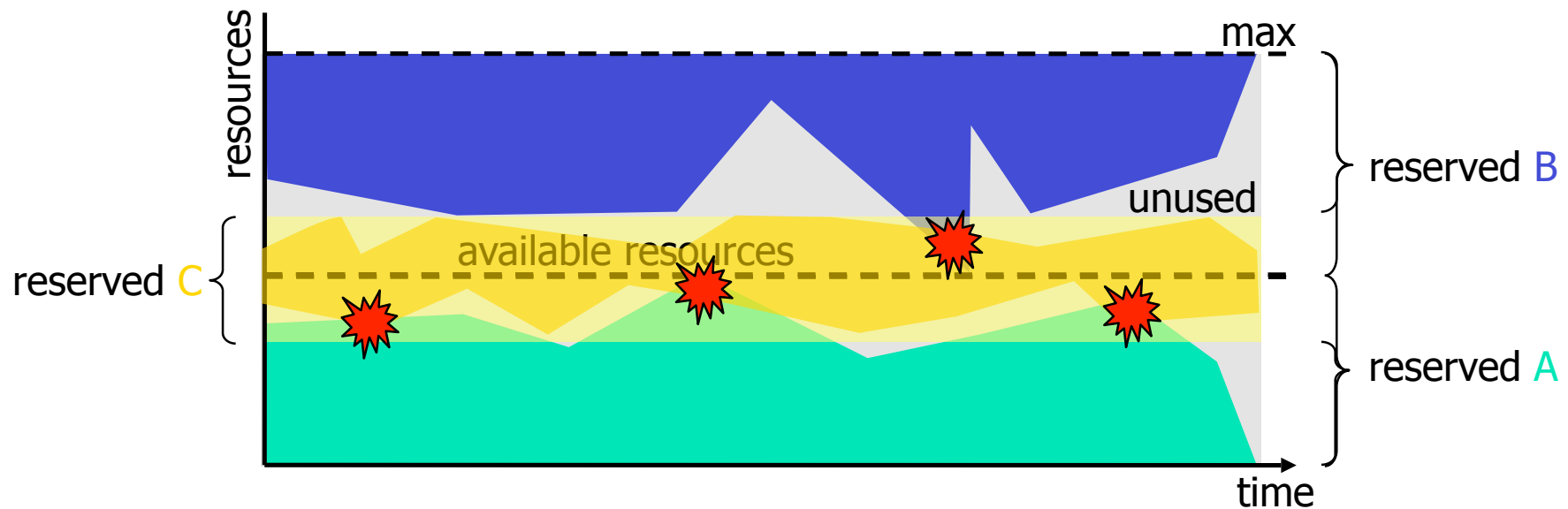
October 01, 2010



Quality-of-Service

Quality-of-Service (QoS)

- Different semantics or classes of QoS:
 - determines **reliability** of offered service
 - **utilization** of resources



Quality-of-Service (QoS)

■ Best effort QoS:

- system tries its best to give a good performance
- no QoS calculation (could be called no effort QoS)

😊 simple – do nothing

😞 QoS may be violated → unreliable service

■ Deterministic guaranteed QoS:

- hard bounds
- QoS calculation based on upper bounds (worst case)
- premium better name!??

😊 QoS is satisfied even in the worst case → high reliability

😞 over-reservation of resources → poor utilization and unnecessary service rejects

😞 QoS values may be less than calculated hard upper bound



Quality-of-Service (QoS)

■ Statistical guaranteed QoS:

- QoS values are statistical expressions (served with some probability)
- QoS calculation based on average (or some other statistic or stochastic value)
- 😊 resource capabilities can be statistically multiplexed → more granted requests
- ☹️ QoS may be temporarily violated → service not always 100 % reliable

■ Predictive QoS:

- weak bounds
- QoS calculation based previous behavior of imposed workload



Quality-of-Service

- Applicability: QoS support
 - A dream of early network researchers (lots of research topics)
 - Guarantees that distributed systems work as promised



Prof. Domenico Ferrari
CRATOS - Università Cattolica del Sacro Cuore, Piacenza, Italy
Until 1995 leader of the Tenet Group
<ftp://tenet.berkeley.edu/pub/tenet>

Quality-of-Service

- Applicability: QoS support
 - A dream of early network researchers
(lots of research topics)
 - Guarantees that distributed systems work as promised
- QoS doesn't exist?
 - IP doesn't support QoS
 - Equality is the Internet's mantra
(do you listen to the net neutrality debate?)
 - Violates Internet philosophy
(shunned by the gurus)

Network neutrality: a principle for user access networks

- ISPs cannot impose limitations on users (customers)
- ISPs cannot treat anybody in a preferred manner
- consequently: no QoS



Quality-of-Service

- Applicability: QoS support
 - A dream of early network researchers (lots of research topics)
 - Guarantees that distributed systems work as promised
- QoS doesn't exist?
 - IP doesn't support QoS
 - Equality is the Internet's mantra (do you listen to the net neutrality debate?)
 - Violates Internet philosophy (shunned)

Network


- ISPs
- ISPs
- cons

Industry demands QoS
Industry does not care about principles
End user want service guarantees
End users do not care about principles
if in doubt: build own network - see Google

KS



INF 5071 – Performance in Distributed Systems



Protocols without QoS Support

October 01, 10

Overview

Defining application for good Internet behaviour

Download applications

Total download time

Achieve low latency

Interactive applications

Fairness to download applications

On-demand **streaming** applications

Sustain application quality
after streaming start

Fairness to download applications



Requirements for Continuous and Interactive Media

- Acceptable delay
 - Seconds in asynchronous on-demand applications
 - Milliseconds in synchronous interpersonal communication
- Acceptable jitter
 - Milliseconds at the application level
 - Tolerable buffer size for jitter compensation
 - Delay and jitter include retransmission, error-correction, ...



Requirements for Continuous Media

- Acceptable synchronity
 - Intra-media: time between successive packets must be conveyed to receiver
 - Inter-media: different media played out at matching times
- Acceptable continuity
 - Streams must be displayed in sequence
 - Streams must be displayed at acceptable, consistent quality
 -



Requirements for Download

- Acceptable delay
 - receive file as soon as possible
- Acceptable fairness
 - competing streams should get the same amount of resources



Basic Techniques

Scaling

Delay and jitter

Buffering

Reservation

Real-time packet re-ordering

Forward error correction

Continuity

Stream switching

Retransmission

Loss detection and compensation

Smoothing

Synchronicity

Buffering

Multiplexing

Time-stamped packets

Fate-sharing and route-sharing



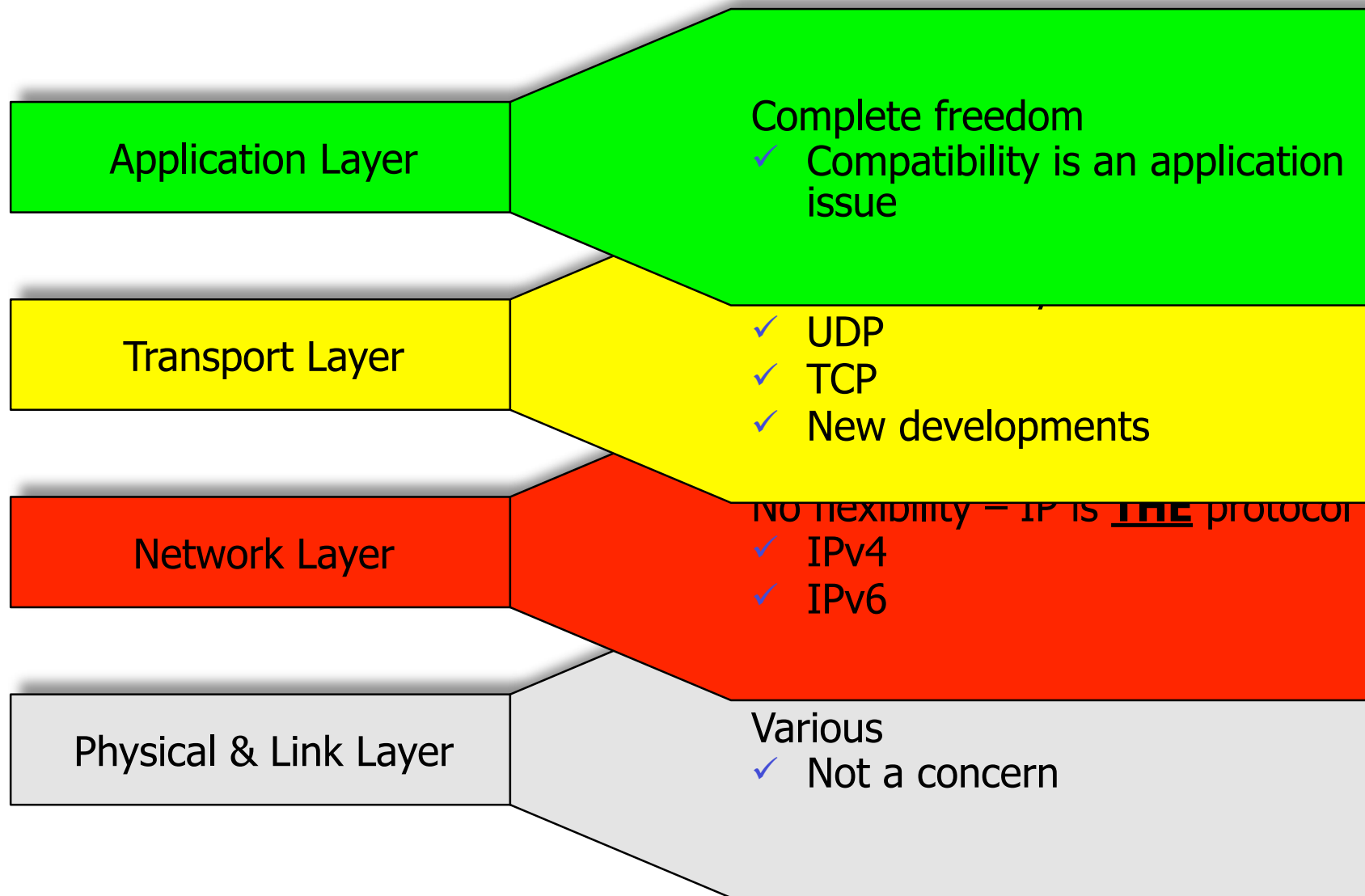
Non-QoS vs. QoS Approaches

- Internet **without** network QoS support
 - Internet applications must cope with networking problems
 - Application itself or middleware
 - "Cope with" means either ...
 - ... "adapt to" which must deal with TCP-like service variations
 - ... "don't care about" which is considered "unfair" and cannot work with TCP

- Internet **with** network QoS support
 - Application must specify their needs
 - Internet **infrastructure must change** – negotiation of QoS parameters
 - Routers need **more features**
 - Keep QoS-related information
 - Identify packets as QoS-worthy or not
 - Treat packets differently keep routing consistent



Protocols for Non-QoS Approaches



Transport Protocol Features

	UDP	DCCP	TCP	SCTP
Connection-oriented service		X	X	X
Connectionless service	X			
Ordered			X	X
Partially Ordered				X
Unordered	X	X		X
Reliable			X	X
Partially Reliable				X
Unreliable	X	X		X
With congestion control		X	X	X
Without congestion control	X			
Multicast support	X	X		
Multihoming support				X



Interactive applications

Interactive applications

- Main examples today
 - Multiplayer games
 - Audio streams
 - Audio conferencing, IP telephony
 - Signaling
 - RTSP for video stream control, SIP for 3G telephone dialing, ...
- Others
 - Remote surgery
 - Robot control
 - Sensing
 - Sensing voice, temperatures, movement, light, ...
 - Bank transactions
 - ...



Thin stream applications

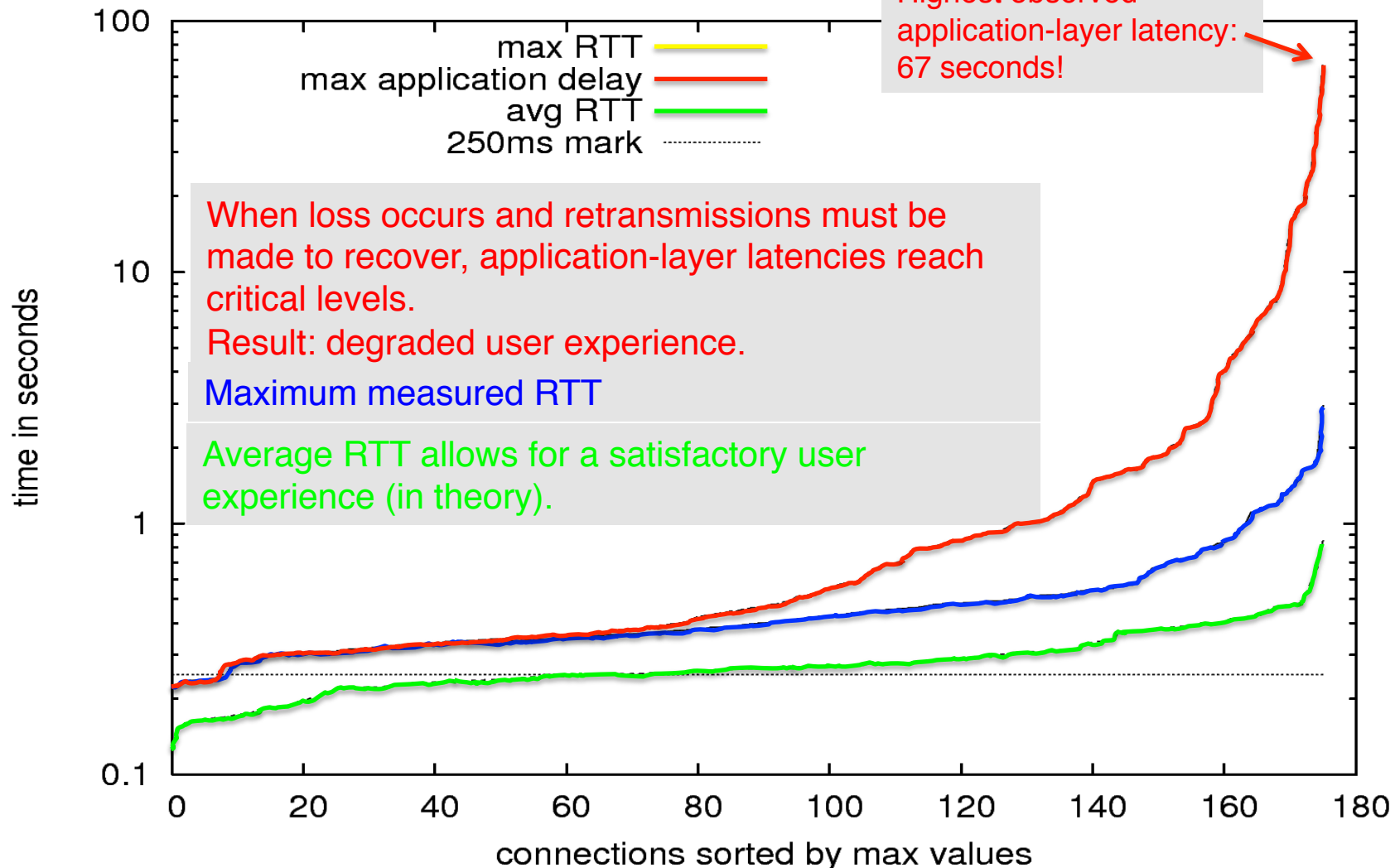
Application	Average payload size (byte)	Packet interarrival time (ms)	Bandwidth requirements (bps)
Anarchy Online	93	909	1757
Counterstrike	142	81	19764
Skype	111	30	37906
CASA (radar control)	175	7287	269
Windows remote desktop	111	318	4497
MPEG-2 streaming	1460	3	~4200000

- Analysis of traces for several applications show thin-stream properties
 - Small packets
 - High packet interarrival-time



Based on real-world observations

The traffic patterns match multiple scenarios, here an **Anarchy Online (FunCom)** example:



Problem: TCP is made for fast download



Greedy streams:

TCP retransmission mechanisms are optimised for high throughput (high packet frequency). Thus, the retransmission mechanisms need frequent feedback (ACKs) to be effective.



Thin streams:

When the throughput is application-limited (low packet frequency, small packets), we often experience very high latencies upon packet loss.

Thin stream TCP enhancements

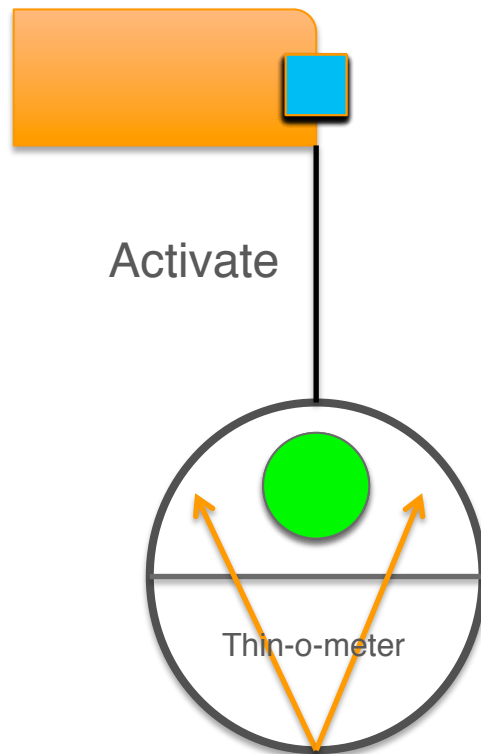
If, and only if, a thin stream is detected by the kernel, we apply three modifications to TCP (explained in hidden slides at the end):

- Modified exponential backoff
- Faster fast retransmit
- Redundant data bundling

which all are sender based, standard compliant (tested on MacOs, Linux, Windows and BSD)



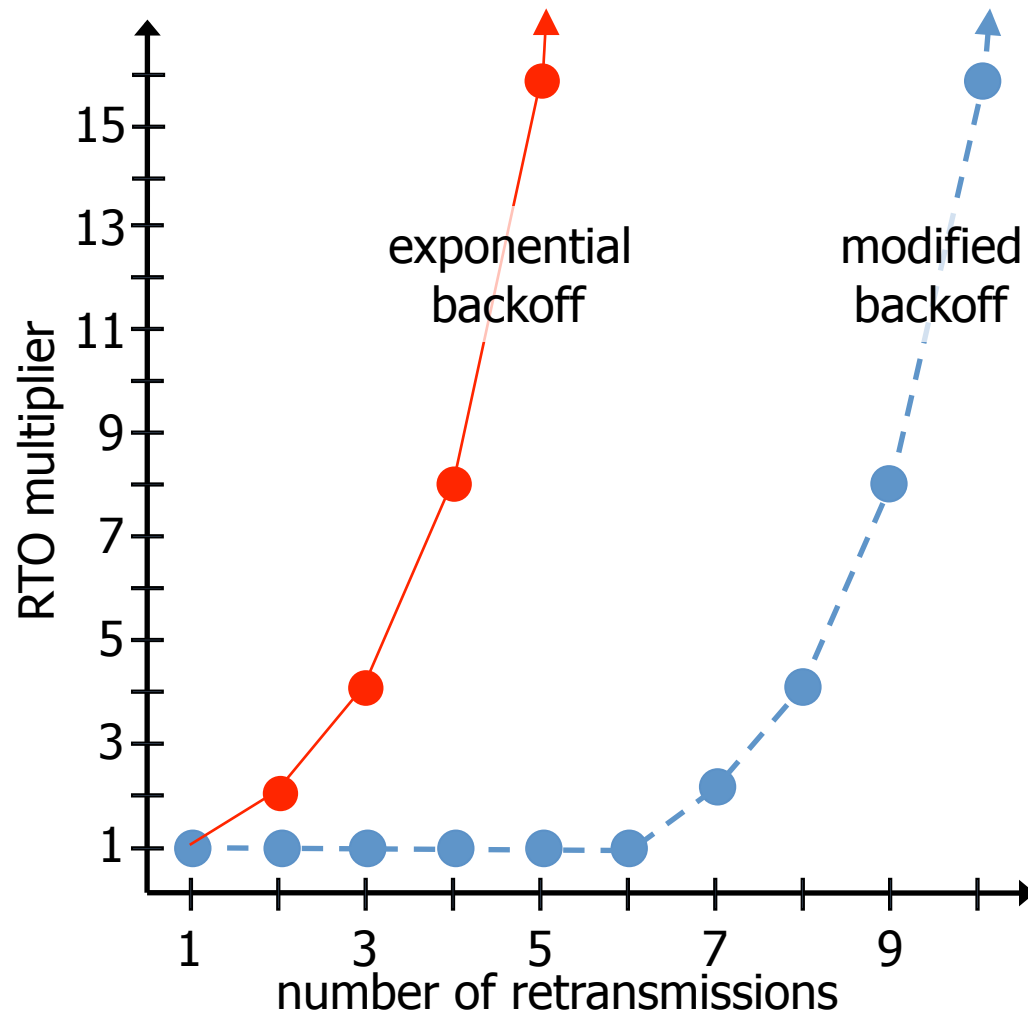
Dynamic triggering



Mechanisms are automatically triggered **only** if thin streams (small and few packets) are detected.

All modifications are **sender-side only**:
No need to update all clients.

Modified Timeouts and exp. backoff

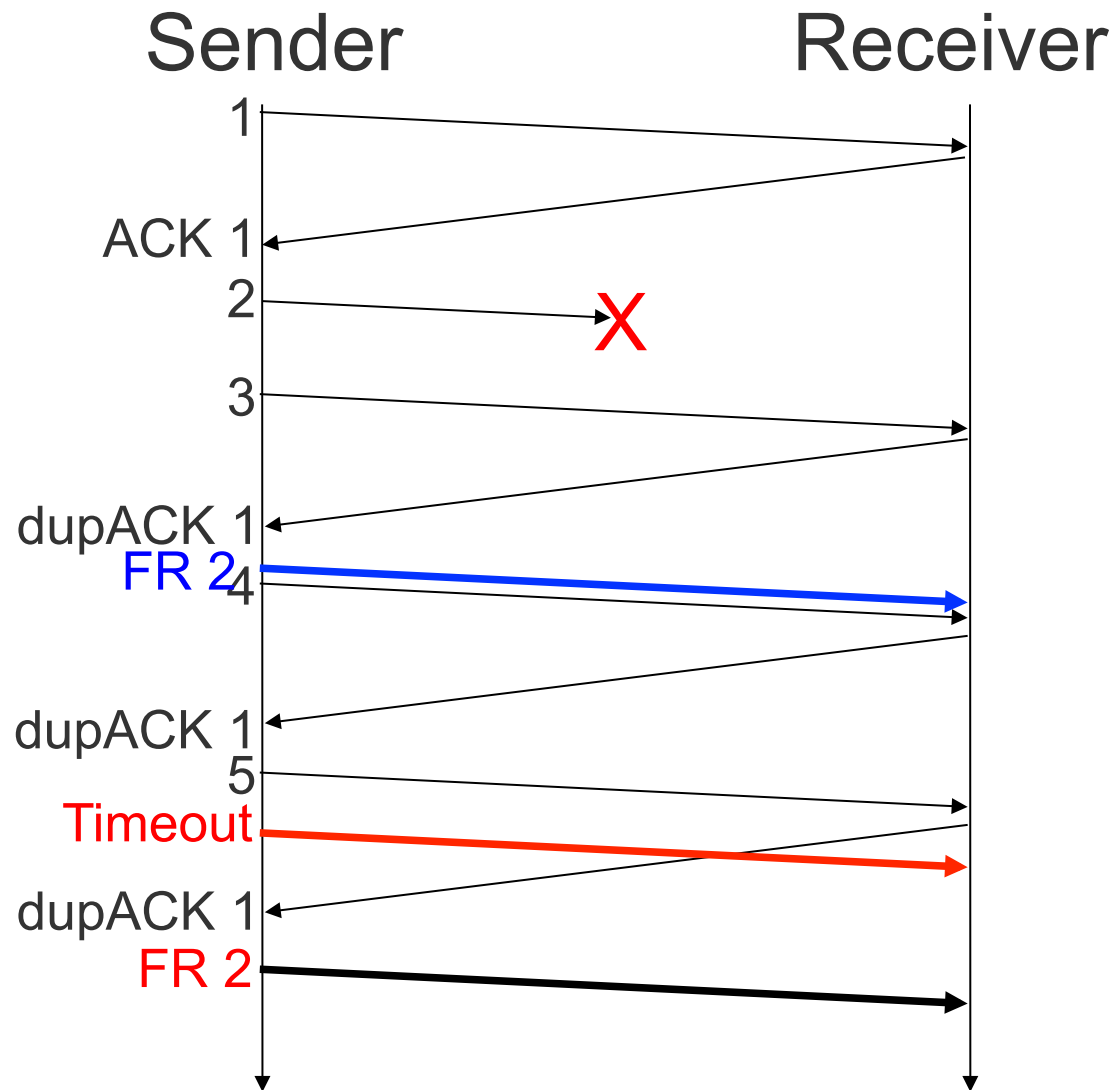


Retransmission time-out (RTO) will double for each consecutive loss.

We use linear timeouts (LT) for thin streams

Result:
Avoids the extreme latencies observed in the Anarchy Online trace.

Fast retransmit with thin-streams

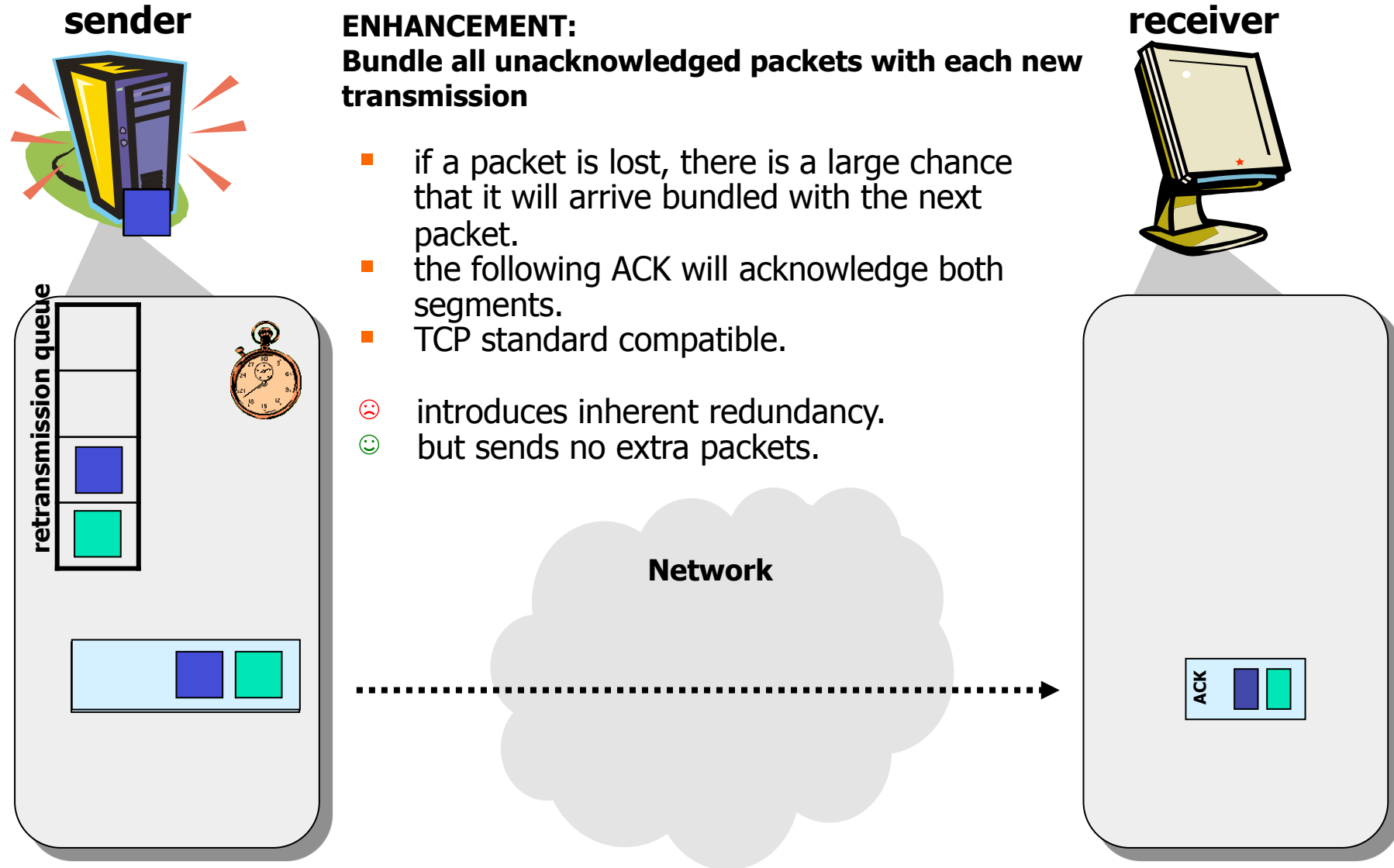


– Thin streams often have < 1 packet per RTT

– Before 3 dupACKs has arrived, a timeout will already have triggered a retransmission

– When thin streams are detected, we trigger a FR after one dupACK

Redundant Data Bundling



ENHANCEMENT:

Bundle all unacknowledged packets with each new transmission

- if a packet is lost, there is a large chance that it will arrive bundled with the next packet.
- the following ACK will acknowledge both segments.
- TCP standard compatible.
- ☹ introduces inherent redundancy.
- ☺ but sends no extra packets.

Results: VoIP audio clips

Played over Skype on a lossy connection (using TCP fallback).

No modifications



Using thin-stream modifications

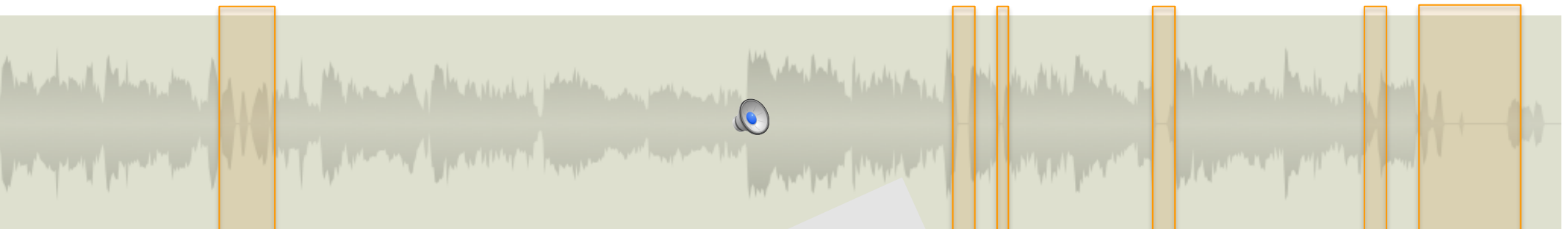


Clearly visible differences in spectrum

Results: VoIP audio clips

Played over Skype on a lossy connection (using TCP fallback).

No modifications



Using thin-stream modifications

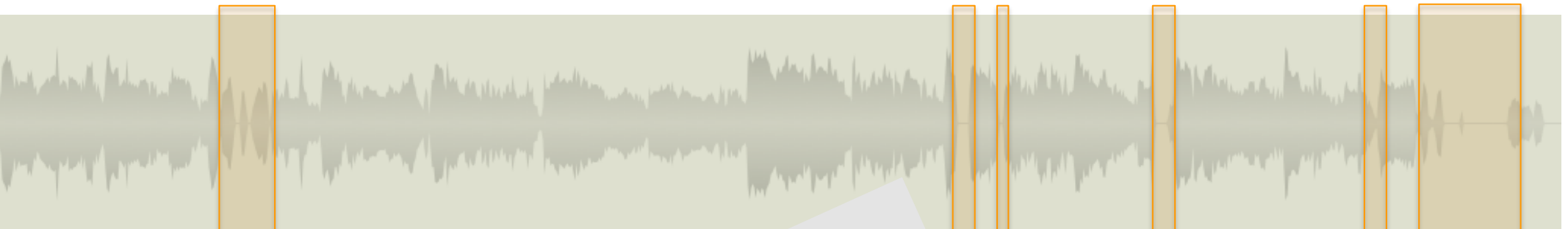


Clearly visible differences in spectrum

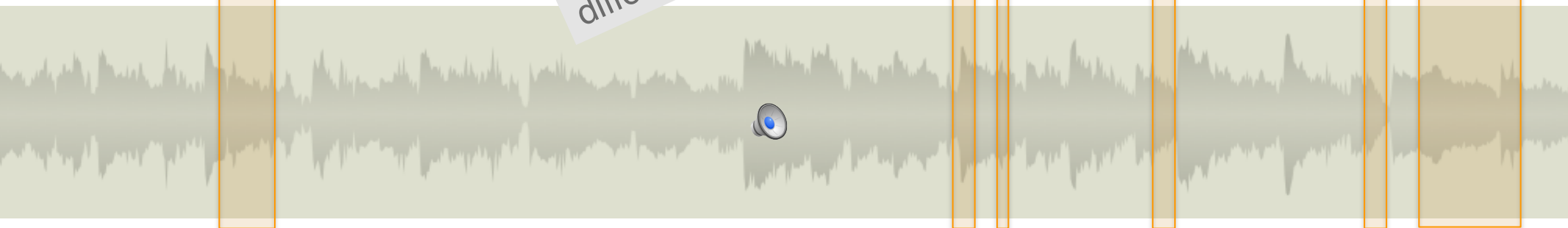
Results: VoIP audio clips

Played over Skype on a lossy connection (using TCP fallback).

No modifications



Using thin-stream modifications

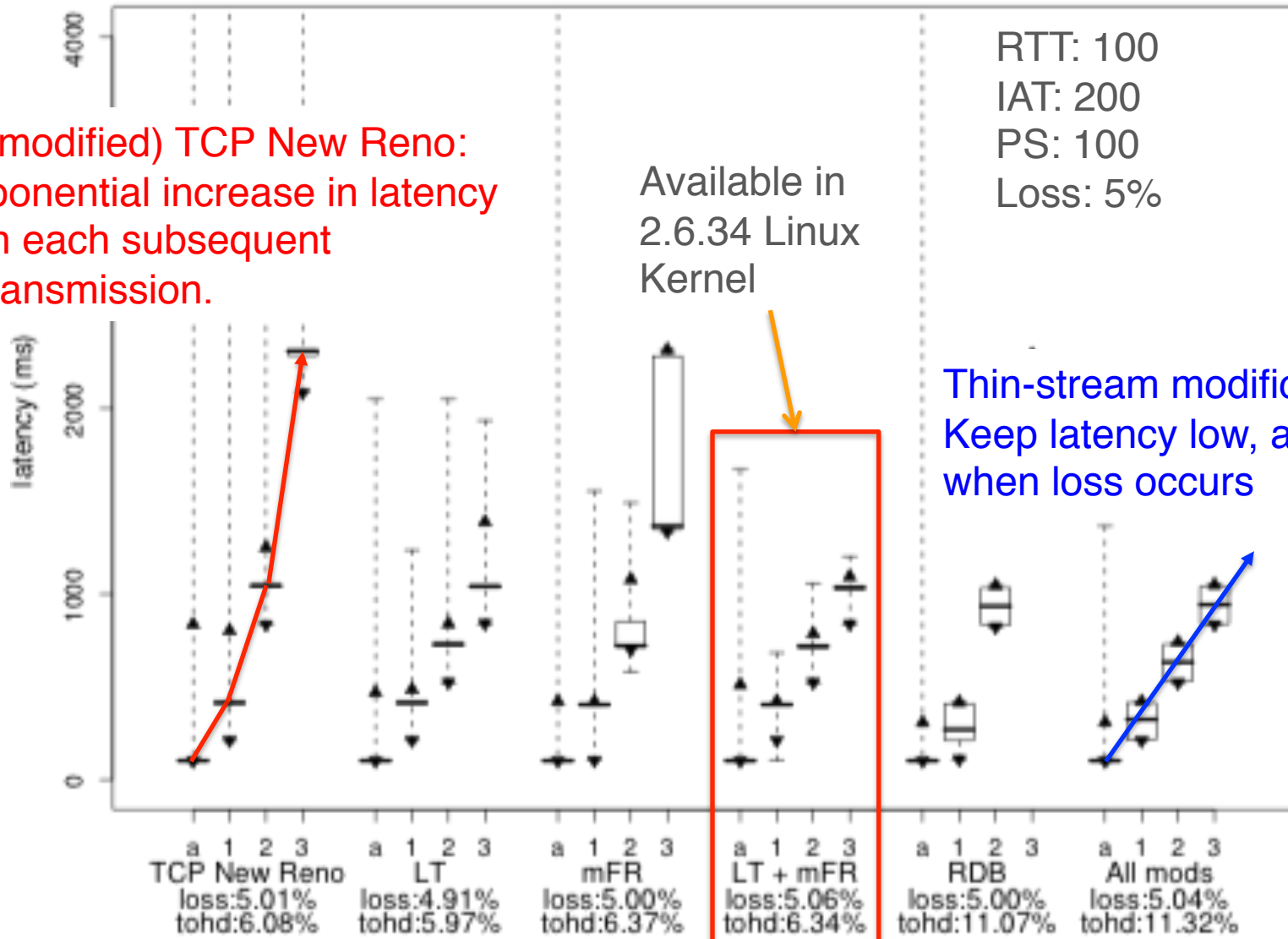


Clearly visible differences in spectrum

Results: Game packets UiO – MA, USA

Replayed Anarchy Online traffic between Oslo, Norway and Worcester, MA, USA:

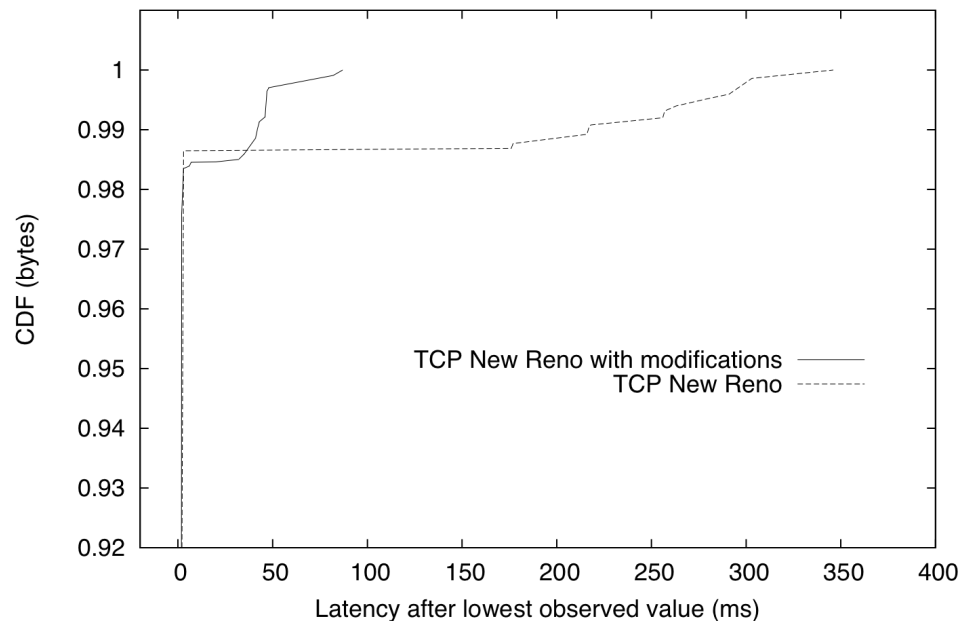
(unmodified) TCP New Reno:
Exponential increase in latency
with each subsequent
retransmission.



Results: Skype data trace analysis

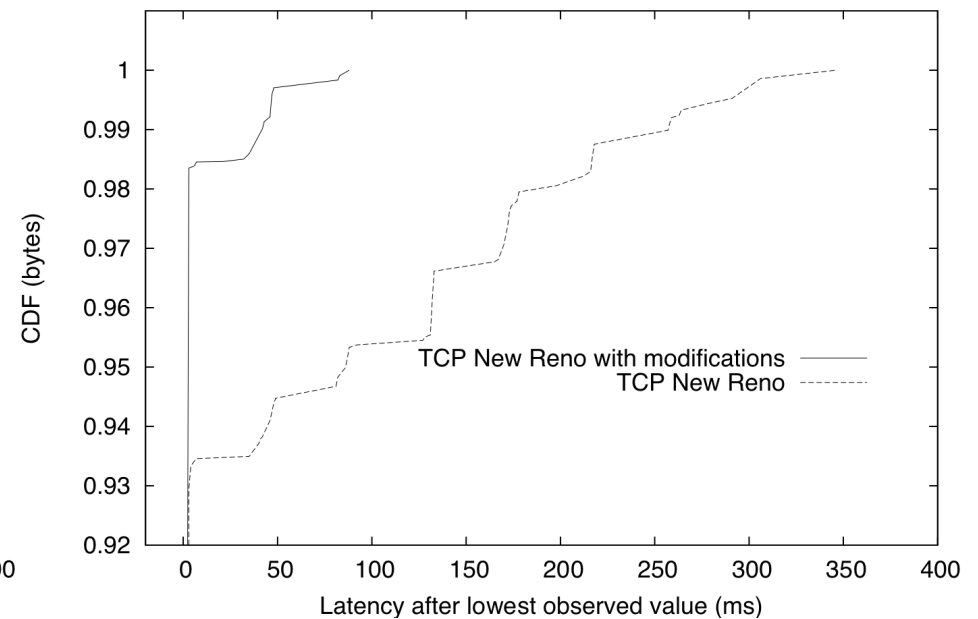
Transport layer delay

Skype CDF, 2% loss, 130ms RTT (delivery latency)







Application layer delay

Skype CDF, 2% loss, 130ms RTT (application latency)



Thin stream mechanism applicability

- From the properties we have discussed, we can derive four "classes" of streams

	Small Packets 	Large Packets 
High IA 	Typical thin stream RDB, retrans, backoff	Rare faster retransmit, backoff
Low IA 	Rare RDB	FTP, HTTP Thick

Interactive Applications

■ Summary

- Interactive applications require low latency
- Current interactive applications generate Thin Streams

- Our options
 - use UDP,
fix problems in the application
 - use TCP or SCTP,
live with high latency
 - use TCP or SCTP,
fix problems in the protocol





Download Applications

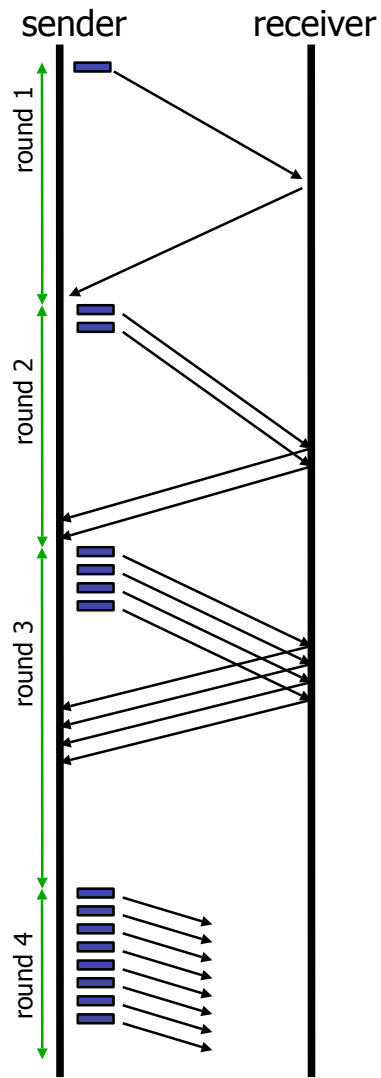
Bandwidth sharing problem

TCP Friendliness: The definition of good Internet behavior

- TCP Congestion Control
- TCP limits sending rate as a function of perceived network congestion
 - little traffic – increase sending rate
 - much traffic – reduce sending rate
- Congestion algorithm has three major “components”:
 - additive-increase, multiplicative-decrease (AIMD)
 - slow-start
 - reaction to timeout events

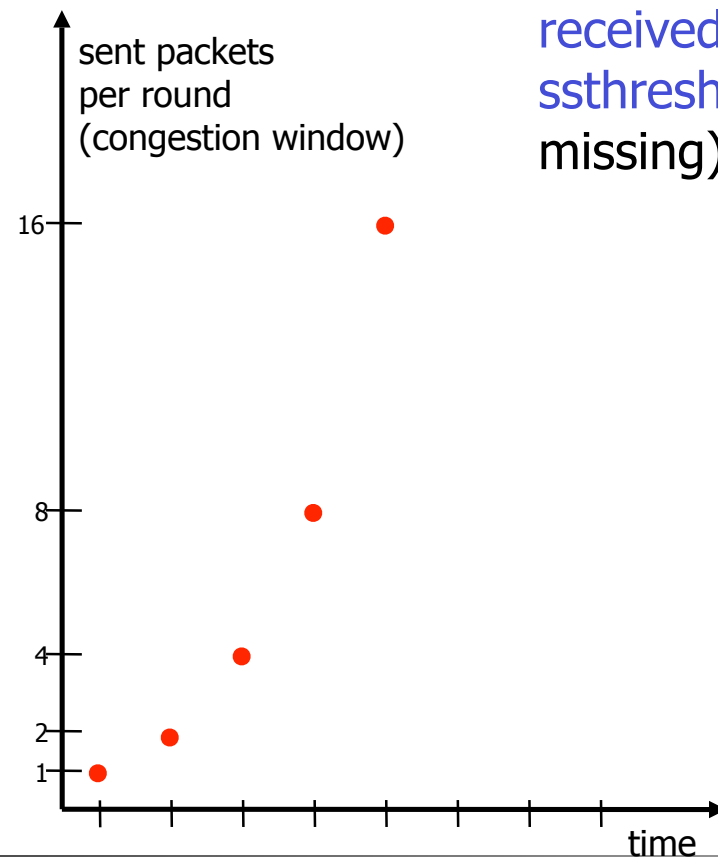


TCP Congestion Control



Initially, the CONGESTION WINDOW is 1 MSS (message segment size)

Then, the size increases by 1 for each received ACK (until threshold `ssthresh` is reached or an ACK is missing)



TCP Congestion Control

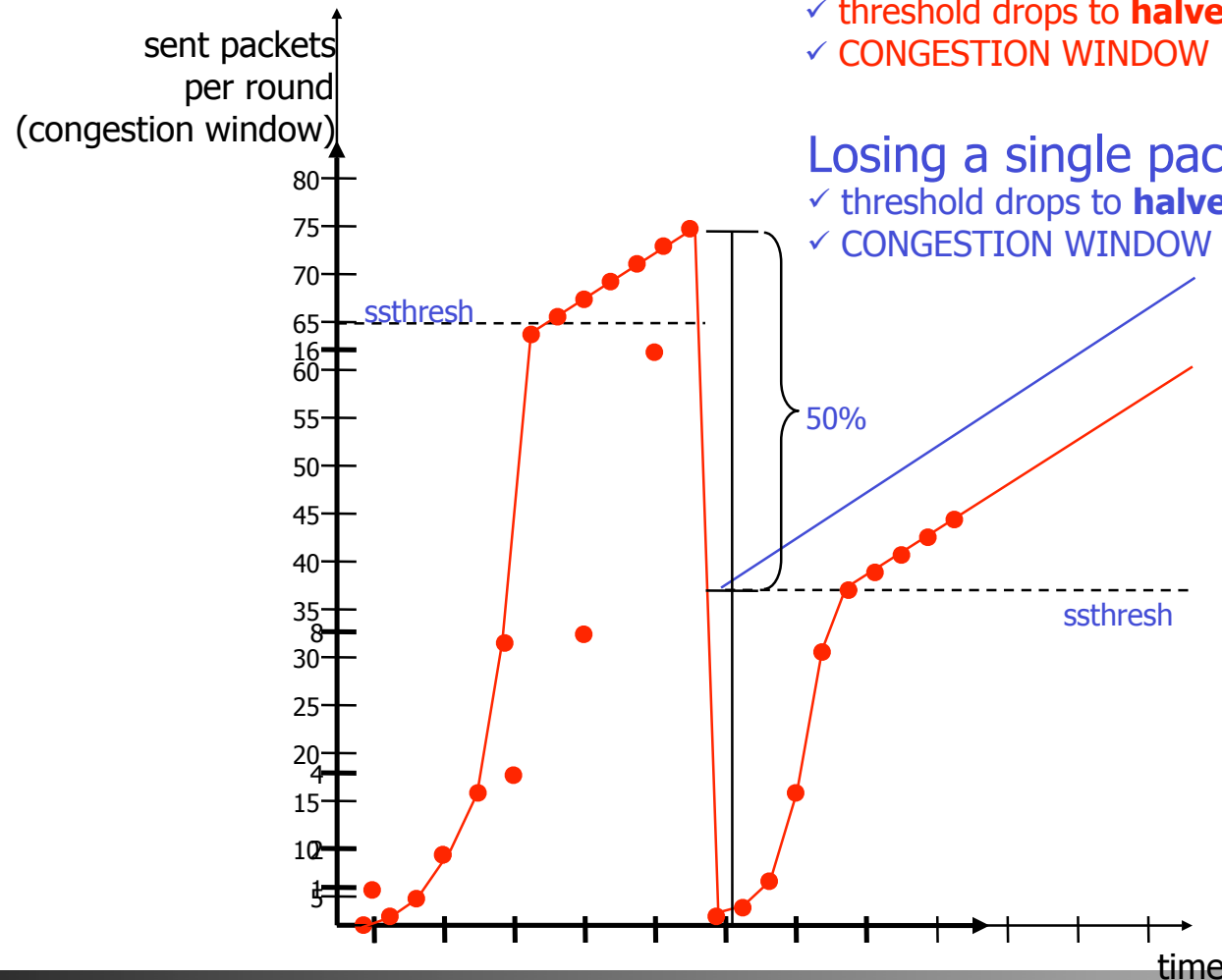
Normally, the threshold is 65 K

Losing a single packet (TCP Tahoe):

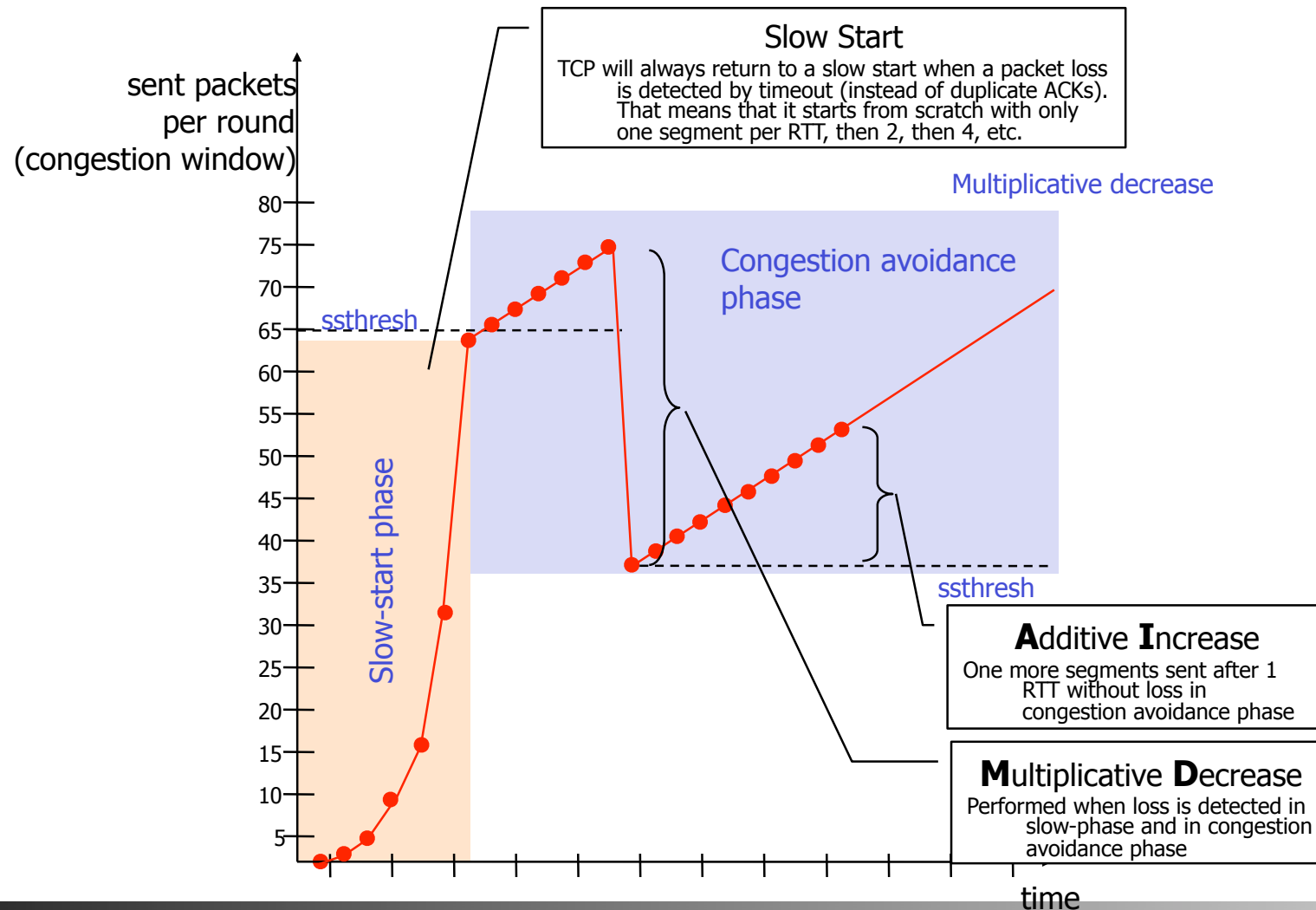
- ✓ threshold drops to **halve** CONGESTION WINDOW
- ✓ CONGESTION WINDOW back to **1**

Losing a single packet (TCP Reno):

- ✓ threshold drops to **halve** CONGESTION WINDOW
- ✓ CONGESTION WINDOW back to **new threshold**



TCP Congestion Control



TCP Friendliness: The definition of good Internet behavior

A TCP connection's throughput is bounded

- ✓ W_{\max} - maximum retransmission window size
- ✓ RTT - round-trip time

Congestion windows size changes

- ✓ **AIMD** algorithm
- ✓ additive increase, multiple decrease

TCP is said to be fair

- ✓ Streams that share a path will reach an equal share

That's not generally true

- ✓ Disadvantage for long-distance traffic → bigger RTT
 - higher loss probability per RTT
 - slower recovery

The TCP send rate limit is

$$R_s = \frac{W_{\max}}{RTT}$$

In case of **loss** in an RTT:

$$w = \beta \times w, \beta = \frac{1}{2}$$

In case of **no loss**:

$$w = w + \alpha, \alpha = 1$$

TCP Friendliness: The definition of good Internet behavior

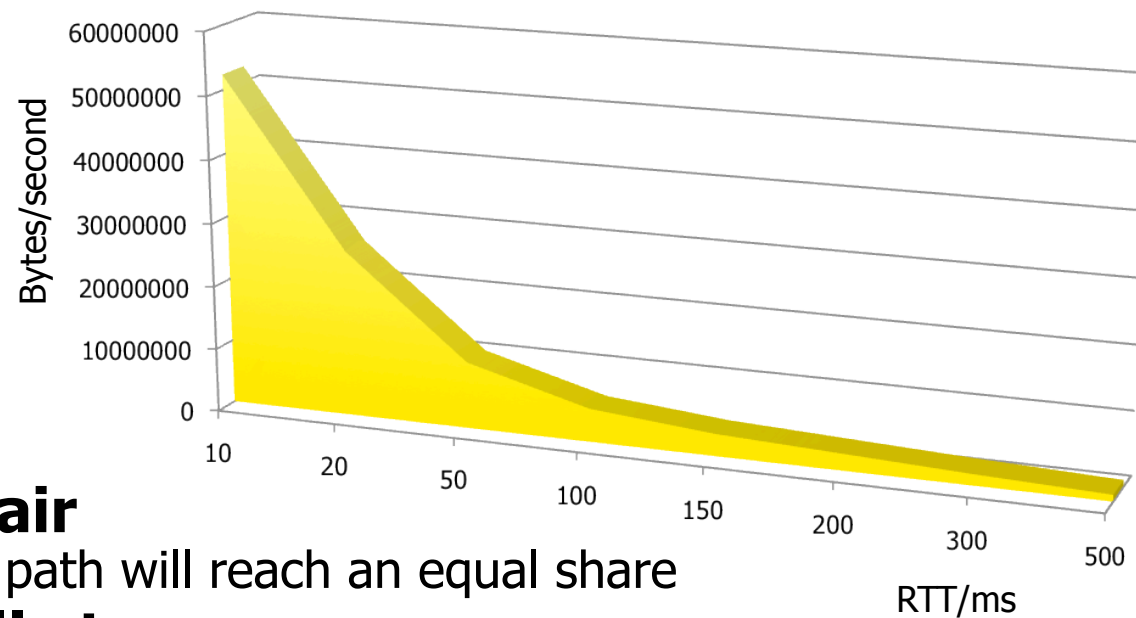
A TCP connection's throughput is bounded

- ✓ W_{\max} - maximum retransmission window size
- ✓ RTT - round-trip time

The TCP send rate limit is

$$R_s = \frac{W_{\max}}{RTT}$$

Example:
TCP throughput when
the Window Scaling options
is not used



TCP is said to be fair

- ✓ Streams that share a path will reach an equal share

That's not generally true

- ✓ Disadvantage for long-distance traffic → bigger RTT
 - higher loss probability per RTT
 - slower recovery



TCP Friendliness: The definition of good Internet behavior

- A protocol is TCP-friendly if
 - **Colloquial:** Its long-term average throughput is not bigger than TCP's
 - **Formal:** Its arrival rate is at most some constant over the square root of the packet loss rate
- Thus, *if the rule is not violated ...*
 - ... TCP-friendly protocols may ...
 - ✓ ...probe for available bandwidth faster than TCP
 - ✓ ...adapt to bandwidth changes more slowly than TCP
 - ✓ ...use different equations or statistics, i.e., not AIMD
 - ✓ ...not use slow start (i.e., don't start with $w=0$)



TCP Congestion Control Alternatives

- Why alternatives?
 - Improve throughput and variance
 - Early TCP implementations did little to minimize network congestion
 - Every indication of a packet loss forces reduction of the congestion window threshold to 50% of the last congestion window size
 - But ...
 - ... what else to conclude from the loss?
 - ... which packets to retransmit?



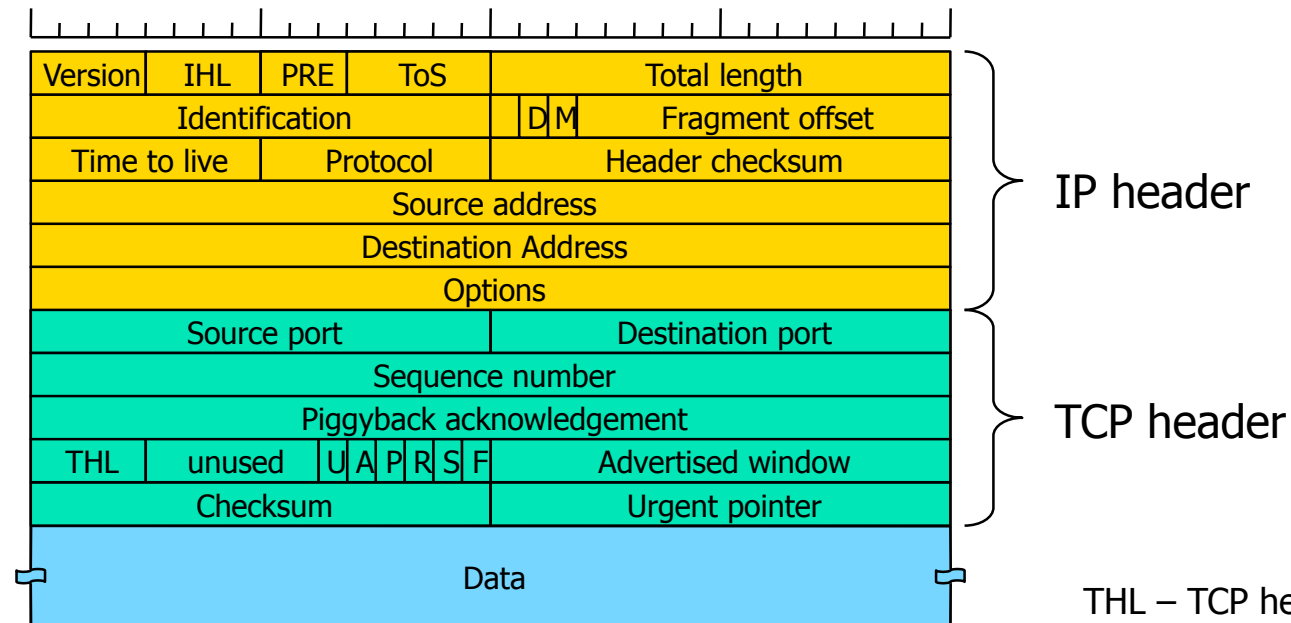
Some TCP Congestion Control Alternatives

- Original TCP
 - not in use
- TCP Tahoe
- TCP Reno
- TCP New-Reno
 - standard TCP headers
- TCP SACK (Selective Acknowledgements)
- TCP FACK (Forward Acknowledgements)
 - must use a TCP option
 - RFC 2018 “TCP Selective Acknowledgment Options”
- TCP Westwood+
 - use bandwidth estimate for congestion events



TCP Congestion Control Alternatives

- TCP/IP Header Format for TCP Tahoe, Reno and New Reno

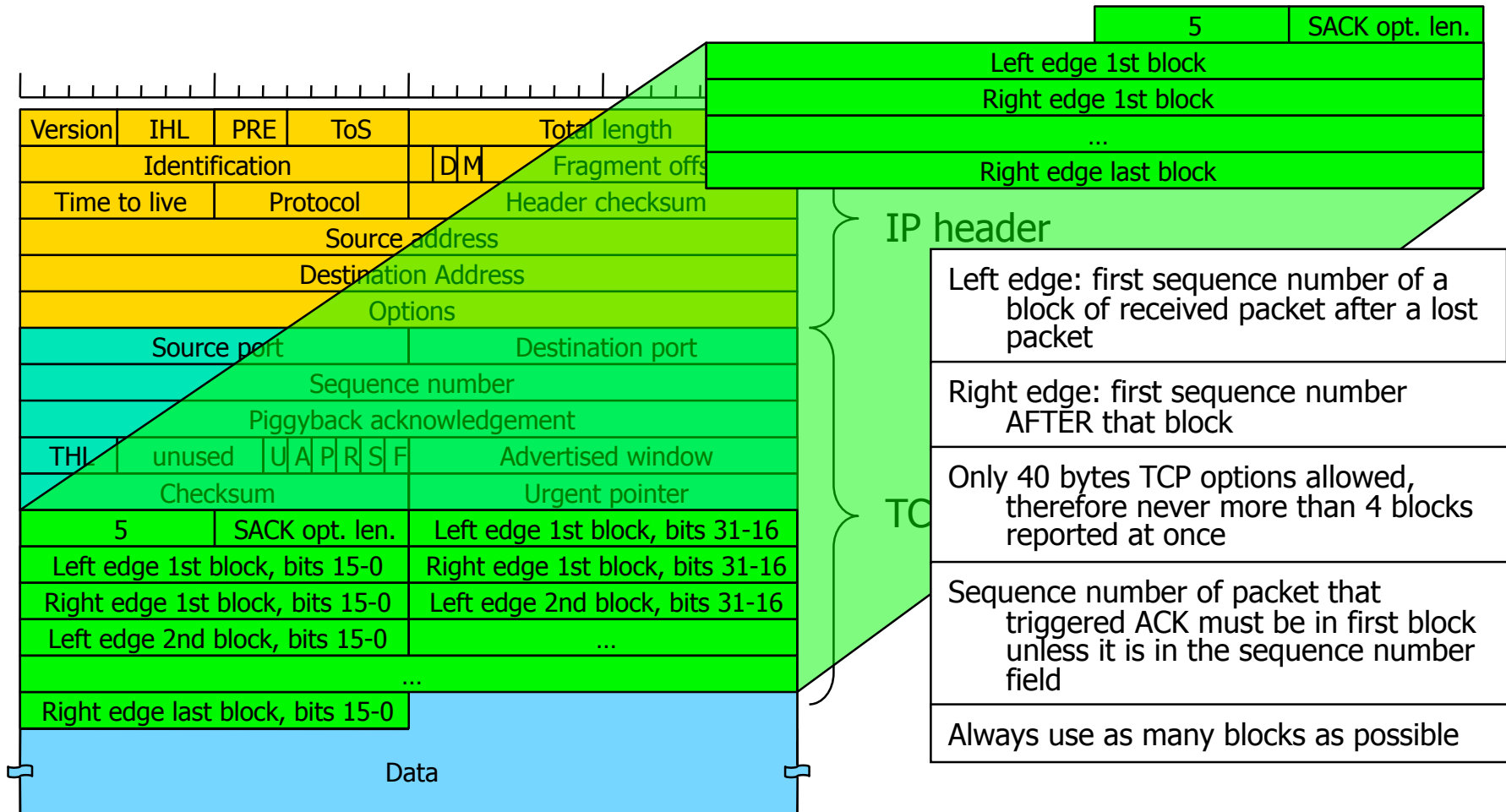


THL – TCP header length
U: URG – urgent
A: ACK – acknowledgement
P: PSH – push
R: RST – reset
S: SYN – sync
F: FIN – finalize



TCP Congestion Control Alternatives

- TCP/IP Header Format for TCP SACK and FACK



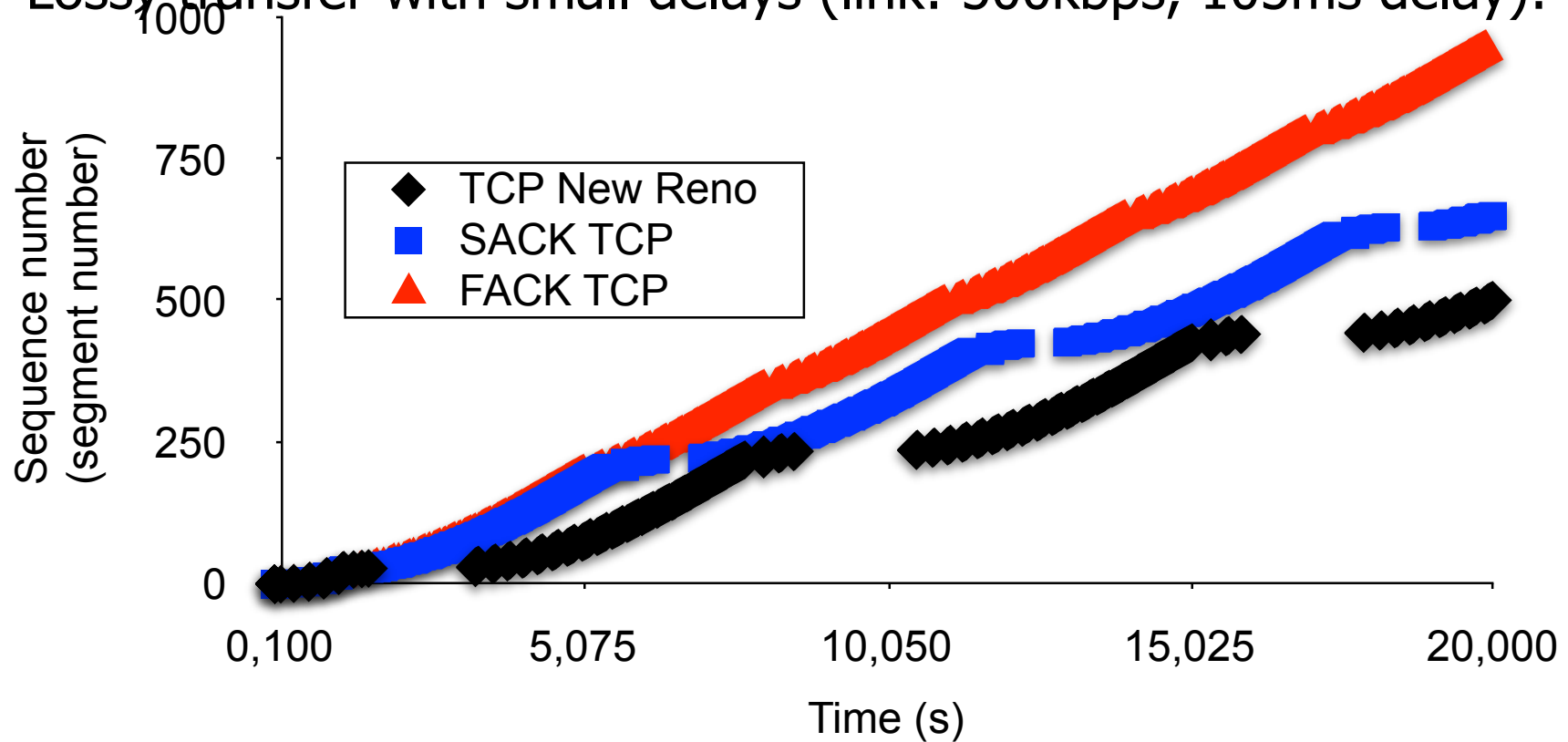
TCP Congestion Control Alternatives

Feature	Original TCP	Tahoe	Reno	New-Reno	SACK	FACK
Retransmission strategy						
Slow start						
Congestion avoidance						
Fast retransmit						
Fast recovery						
Stay in f. rec.						
In flight packet estimation						
Cong. window halving						



Simulation results

Sequence number development
Lossy transfer with small delays (link: 500kbps, 105ms delay):

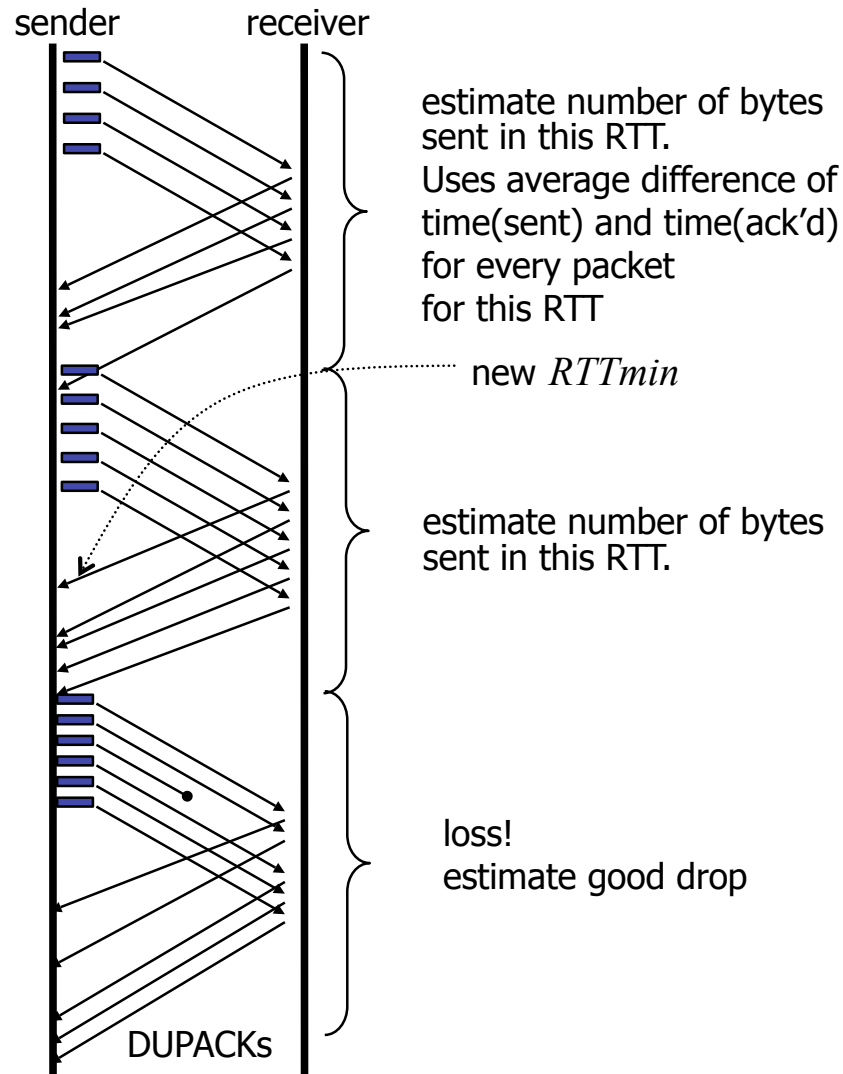


TCP Westwood+

- Very recent
- Developed for wireless networks with many losses
 - Losses in wireless networks are often **non-congestion losses**: corruption losses
- Side effect
 - Less unfair against long round-trip times
- Implemented in Linux
 - With SACK extensions
- Procedure
 - TCP Westwood uses ACK packets
 - provide a bandwidth estimate
- “Faster recovery”
 - After loss indication by a triple-ACK go into faster recovery
 - Use bandwidth estimate to set new congestion window size and new slow start threshold

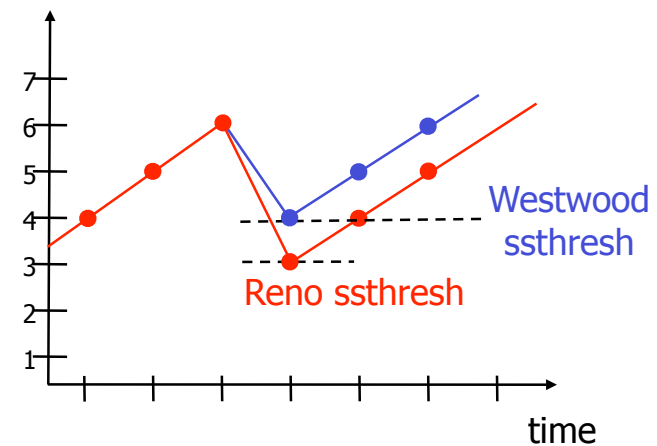


TCP Westwood+



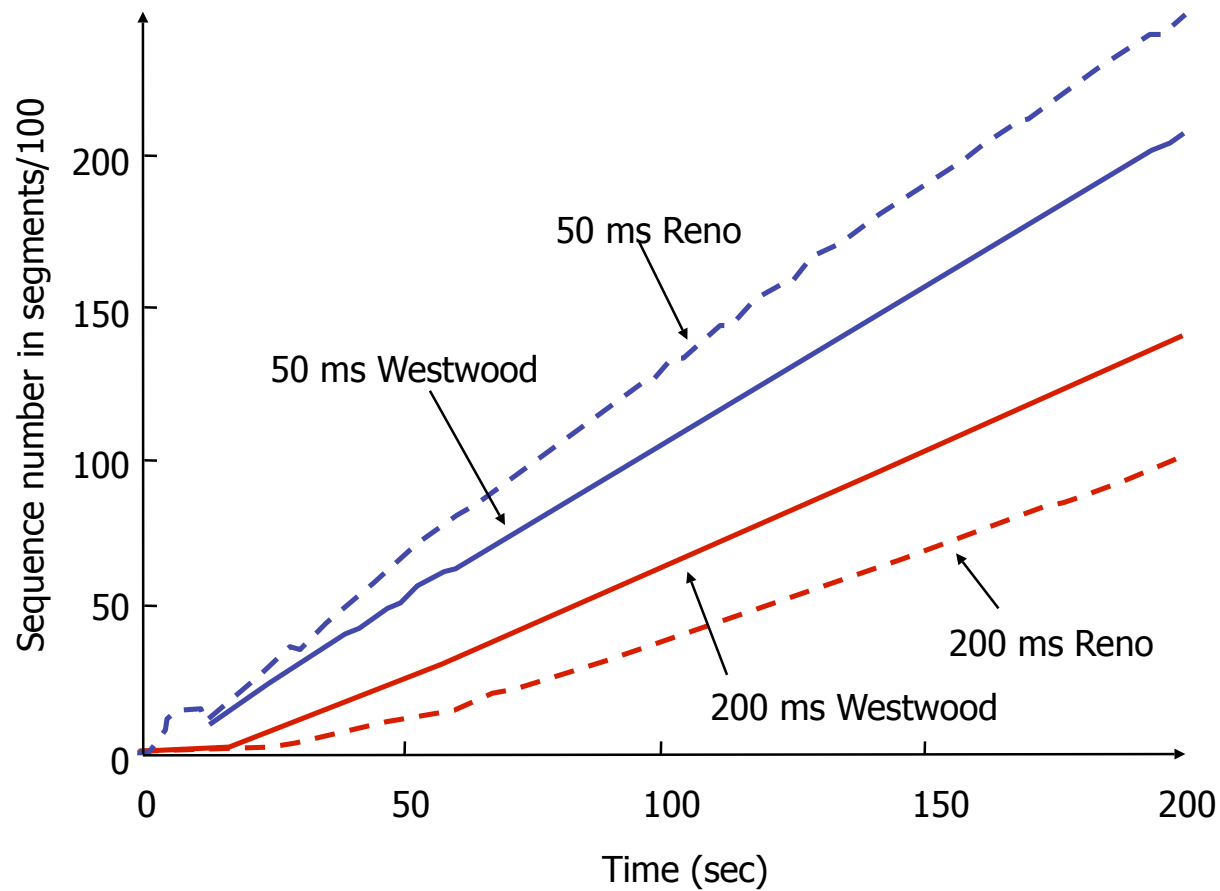
estimate bytes that can be sent per time unit (e.g. second)

ssthresh = in case of loss, get a minimum of bytes that have been supported per RTT



TCP Westwood+

Smaller difference between streams having short and long RTTs

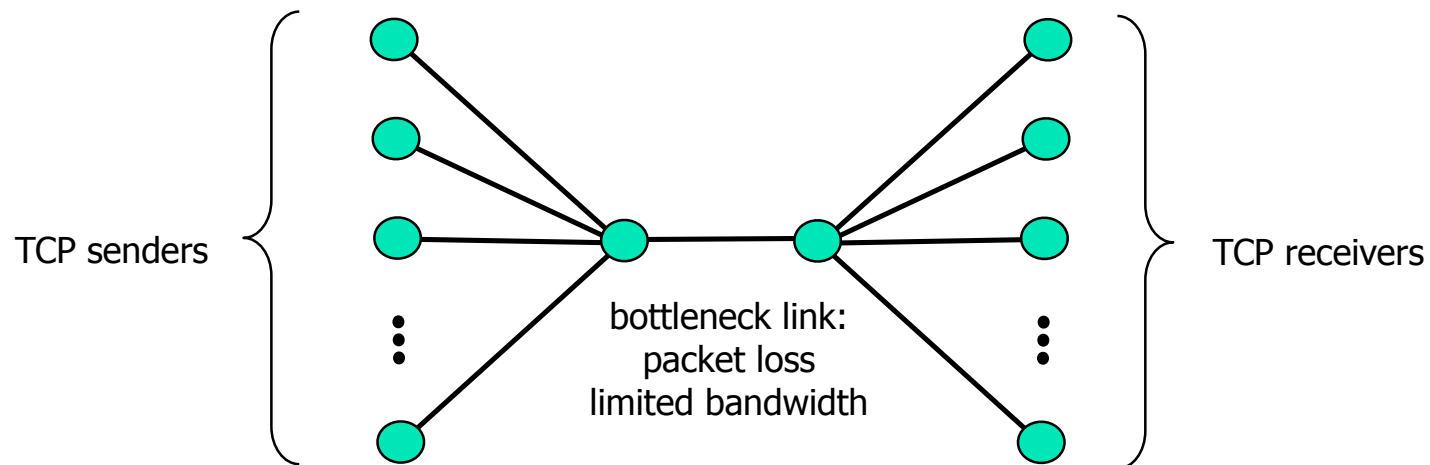


(approximation of a perf. eval. figure)



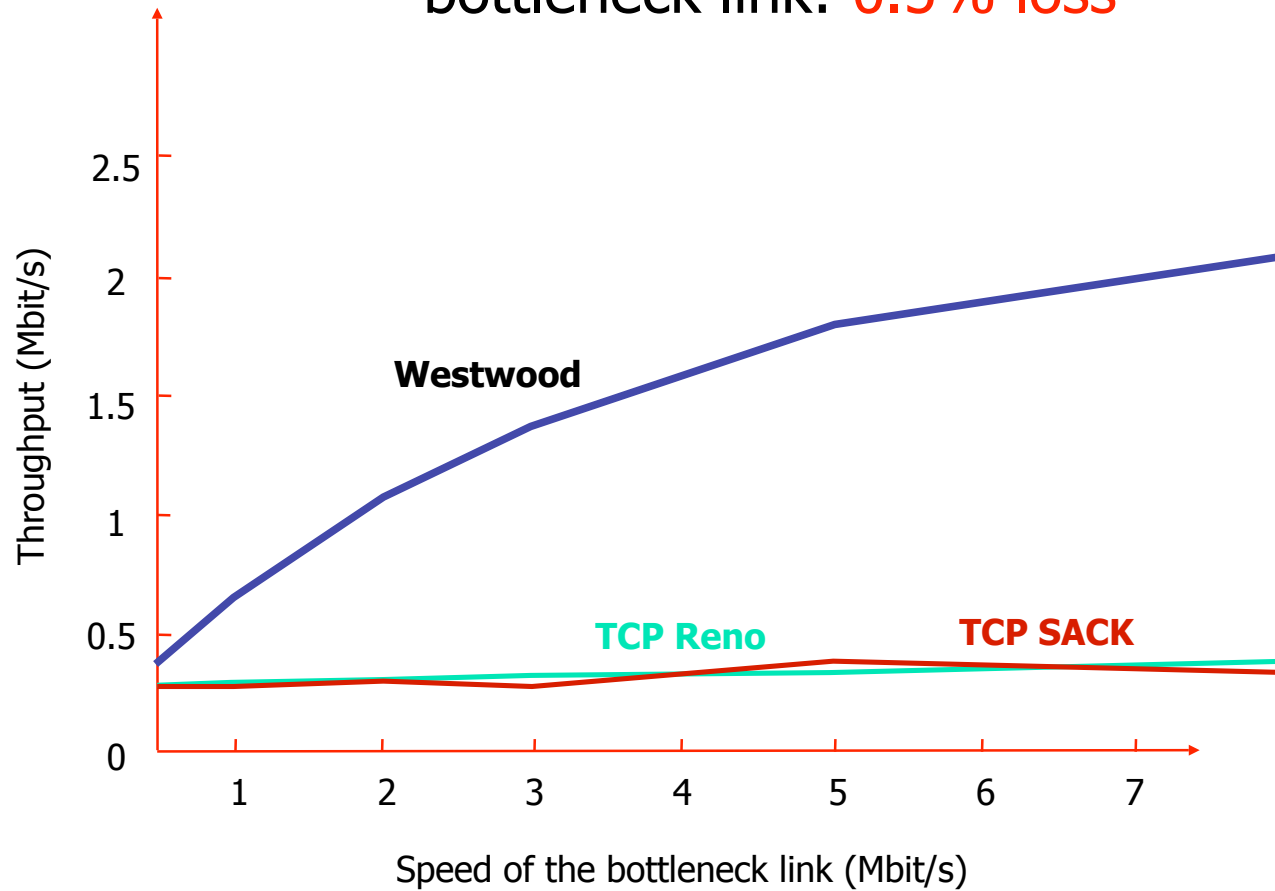
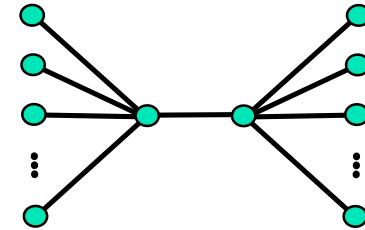
TCP Westwood+

Uniformly distributed errors at the bottleneck link: **0.5% loss**



TCP Westwood+

Uniformly distributed errors at the bottleneck link: **0.5% loss**



(approximation of a perf. eval. figure)



Datagram Congestion Control Protocol (DCCP)

- Transport Protocol
 - Offers unreliable delivery
 - Low overhead like UDP
 - Applications using UDP can easily change to this new protocol
 - Uses ACKs and ECN

- Accommodates different congestion control
 - Congestion Control IDs (CCIDs)
 - Add congestion control schemes on the fly
 - Choose a congestion control scheme
 - TCP-friendly Rate Control (TFRC) is included
 - Half-Connection
 - Data Packets sent in one direction



Datagram Congestion Control Protocol (DCCP)

- Congestion control is pluggable
 - One proposal is **TCP-Friendly Rate Control (TFRC)**
 - Equation-based TCP-friendly congestion control
 - Receiver sends rate estimate and loss event history
 - Sender uses models of SACK TCP to compute send rate

$$T = \frac{\text{Steady state TCP send rate}}{RTT \sqrt{\frac{2bp}{3}} + t_{RTO} \min(1, 3\sqrt{\frac{3bp}{8}}) p(1 + 32p^2)}$$

Loss probability

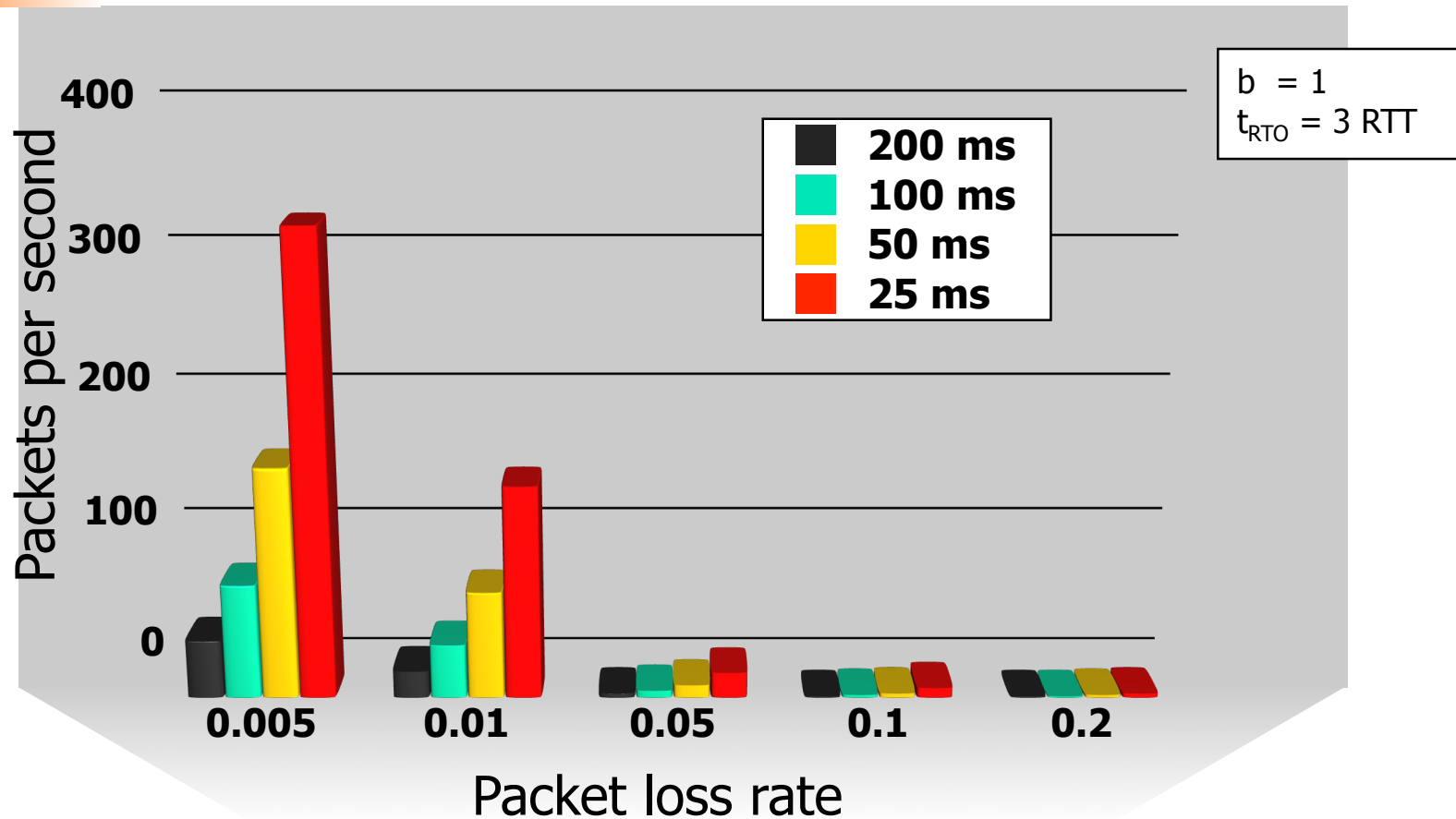
Retransmission timeout

Number of packets ack'd by one ACK

Padhye's TCP New Reno estimation formula



Datagram Congestion Control Protocol (DCCP)



$$T = \frac{1}{RTT \sqrt{\frac{2bp}{3}} + t_{RTO} \min(1, 3\sqrt{\frac{3bp}{8}}) p(1 + 32p^2)}$$



Download applications

- Loss is worst ...
 - ... because it must be corrected
 - ... because it must be interpreted as congestion, and
 - TCP-friendliness demands that bandwidth consumption is reduced

- Non-QoS problem
 - Transport layer can share bandwidth only fairly
 - End-users can tweak this: performance isolation

- Other TCP variants that you find in Linux ...
 - BIC
 - CUBIC
 - Vegas
 - High-speed TCP
 - Fast TCP
 - H-TCP
 - ...

- ... and in Windows 7
 - Default (BSD)
 - Compound TCP

