INF 5071 – Performance in Distributed Systems

Protocols without QoS Support

28/9 - 2007

Requirements for Continuous Media

- Acceptable continuity
  - Streams must be displayed in sequence
  - Streams must be displayed at acceptable, consistent quality
- Acceptable delay
  - Seconds in asynchronous on-demand applications
  - Milliseconds in synchronous interpersonal communication

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 jitter
  - Milliseconds at the application level
  - Tolerable buffer size for jitter compensation
  - Delay and jitter include retransmission, error-correction, ...

Basic Techniques

- Delay and jitter
  - Reservation (sender, receiver, network)
  - Buffering (receiver)
  - Scaling (sender)
- Continuity
  - Real-time packet re-ordering (receiver)
  - Loss detection and compensation
  - Retransmission
  - Forward error correction
  - Stream switching (encoding & server)
- Synchronity
  - Fate-sharing and route-sharing (networks with QoS-support)
  - Time-stamped packets (encoding)
  - Multiplexing (encoding, server, network)
  - Buffering (client)
  - Smoothing (server)

QoS vs. Non–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

Overview

- Non-QoS protocols
  - Download applications
    - Defining application for good Internet behavior
    - Total download time
  - On-demand streaming applications
    - Fairness to download applications
    - Sustain application quality after streaming start
  - Interactive applications
    - Fairness to download applications
    - Achieve a low latency

Protocols for Non–QoS Approaches

Application Layer

Complete freedom
- ✓ Compatibility is an application issue

Transport Layer

- ✓ UDP
- ✓ TCP
- ✓ New developments

Network Layer

No flexibility – IP is THE protocol
- ✓ IPv4
- ✓ IPv6

Physical Layer

Various
- ✓ Not a concern

Download Applications

Bandwidth sharing problem

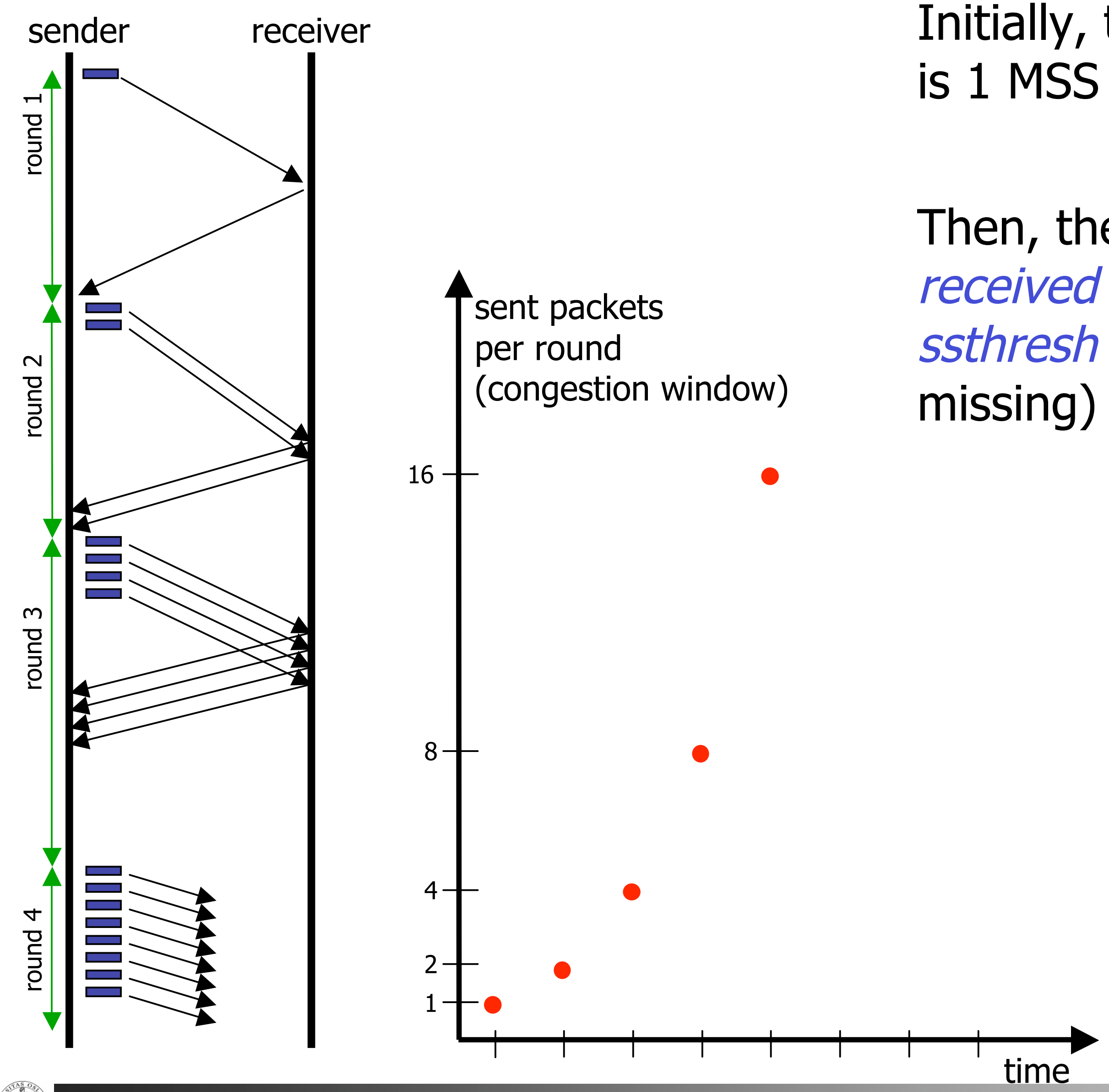
TCP Friendliness:
The definition of good Internet behavior

- TCP Congestion Control

- TCP limit 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

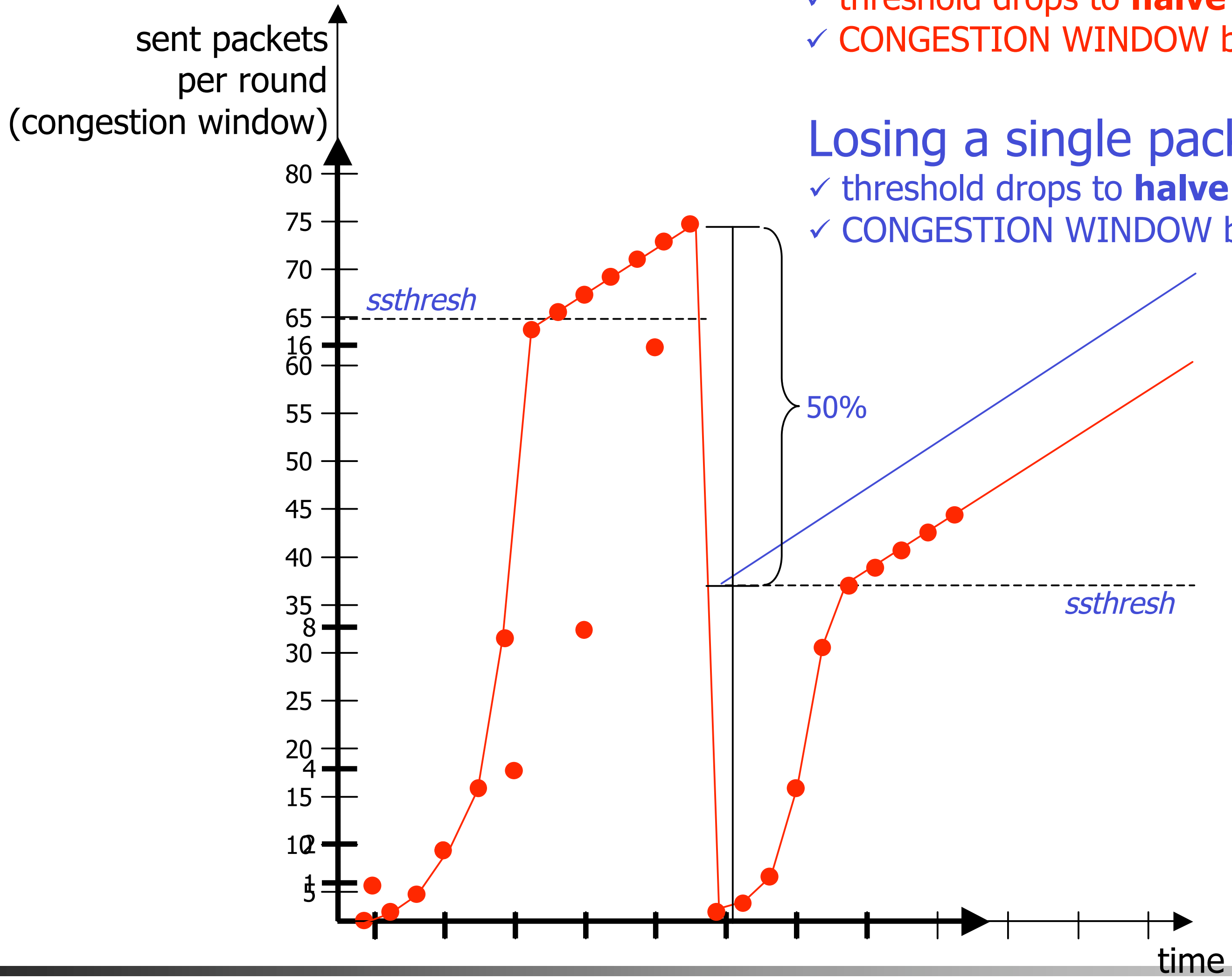
TCP Congestion Control



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

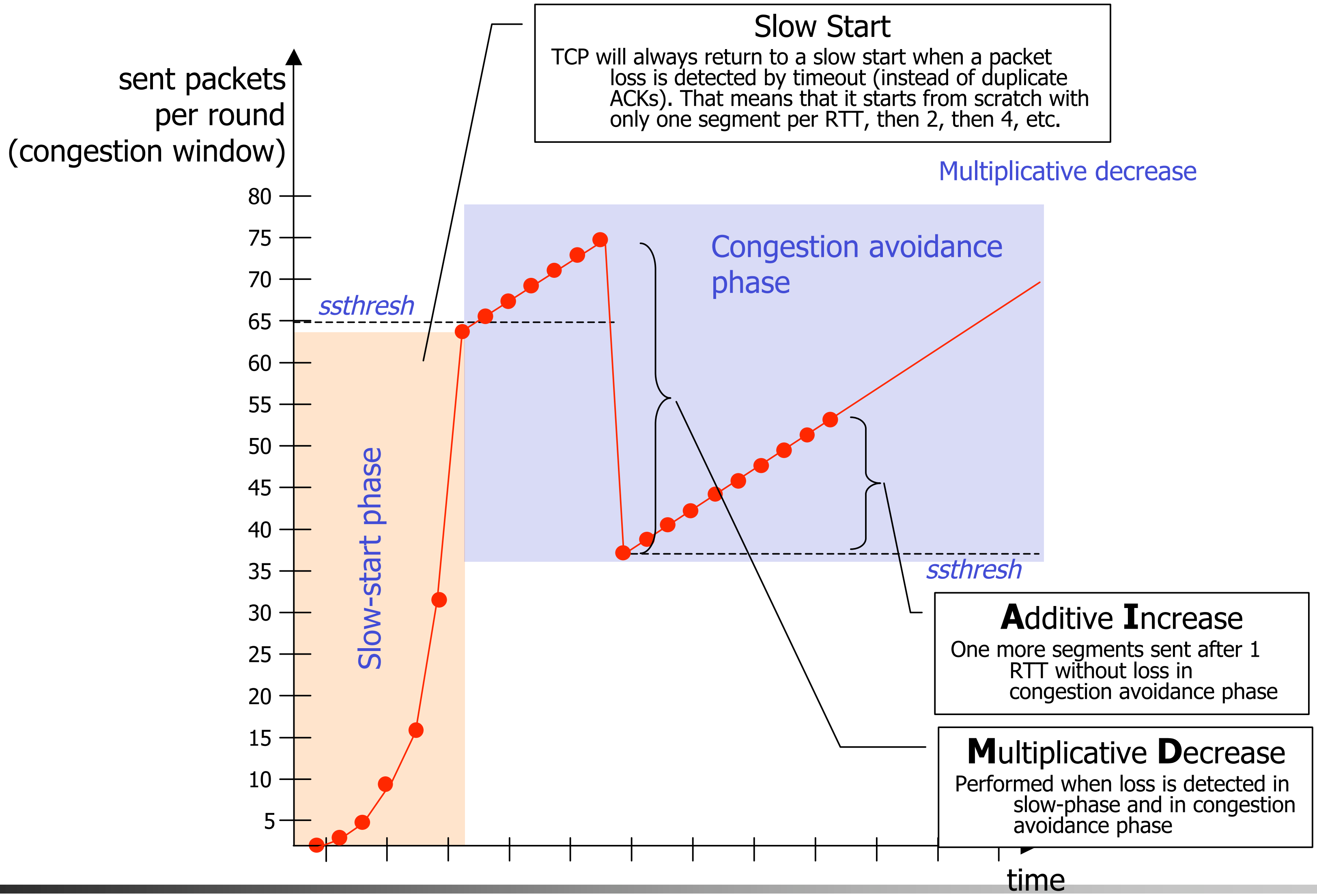
Losing a single packet (TCP Tahoe):

- ✓ 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

The TCP send rate limit is

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

In case of loss in an RTT:

$$w = \beta \cdot w, \beta = 1/2$$

In case of no loss:

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

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

- ✓ Bigger RTT
  - ➢ higher loss probability per RTT
  - ➢ slower recovery
- ✓ Disadvantage for long-distance traffic

TCP Friendliness:
The definition of good Internet behavior

- A protocol is TCP-friendly if
  - ***Colloquial:*** *It 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* …

… the AIMD algorithm with $\alpha \neq 1/2$ and $\beta \neq 1$ is still TCP-friendly

… 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
    - Loss indication forces setting new congestion window threshold to half of the last congestion window size
    - But …
      - … what else to conclude from the loss?
      - … which packets to retransmit?

TCP Congestion Control Alternatives

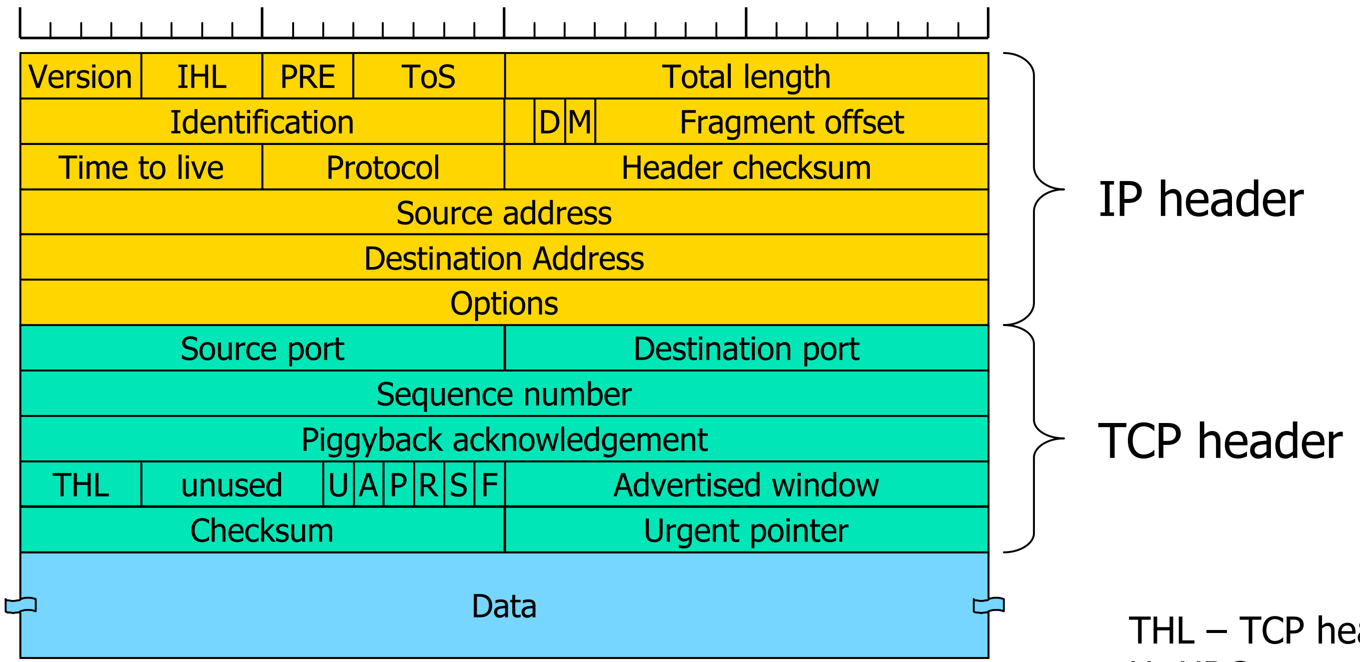
- Original TCP
  - not in use

- 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

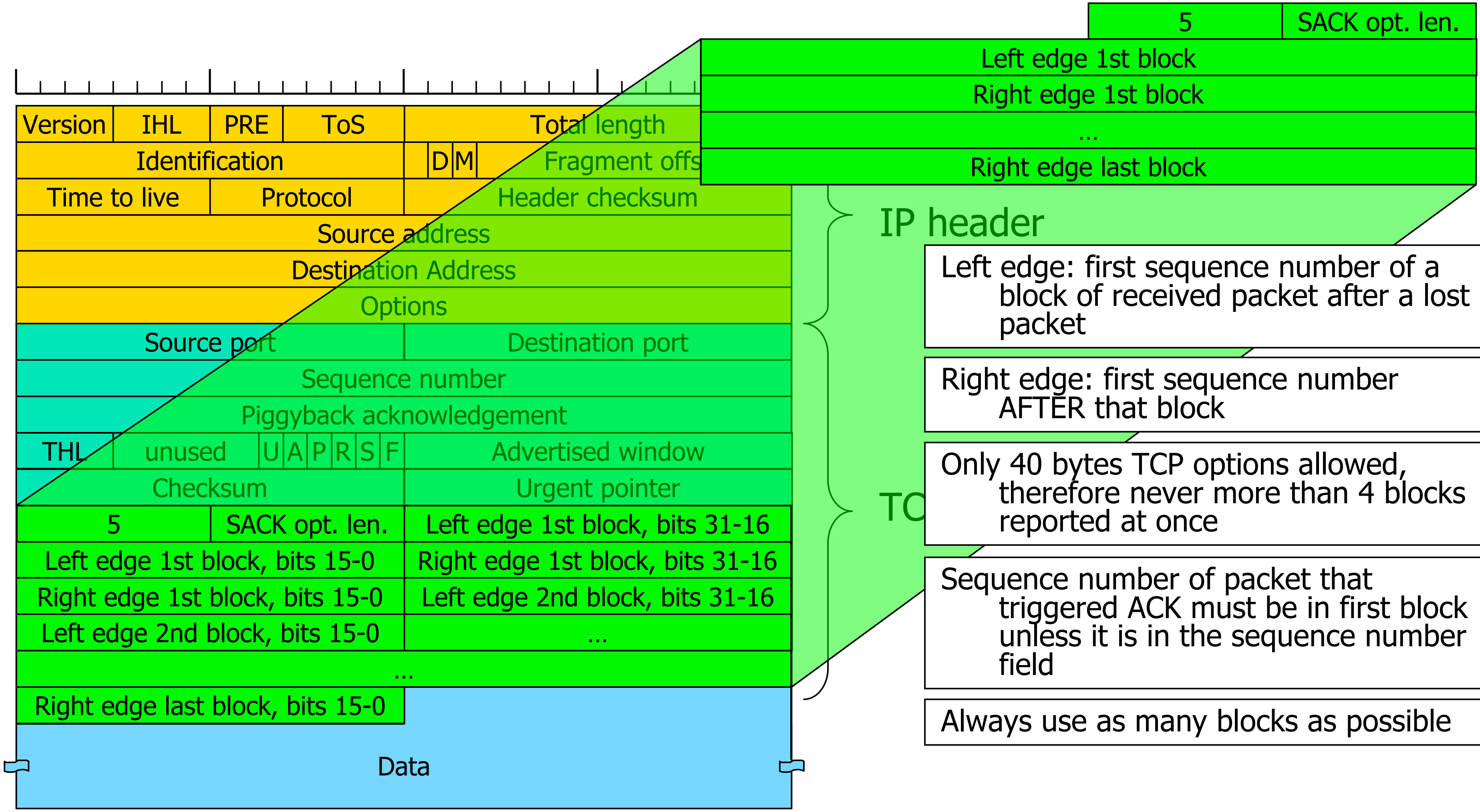
- TCP/IP Header Format for TCP Tahoe, Reno and New Reno



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



Left edge: first sequence number of a block of received packet after a lost packet

Right edge: first sequence number AFTER that block

Only 40 bytes TCP options allowed, therefore never more than 4 blocks reported at once

Sequence number of packet that triggered ACK must be in first block unless it is in the sequence number field

Always use as many blocks as possible

TCP Congestion Control Alternatives

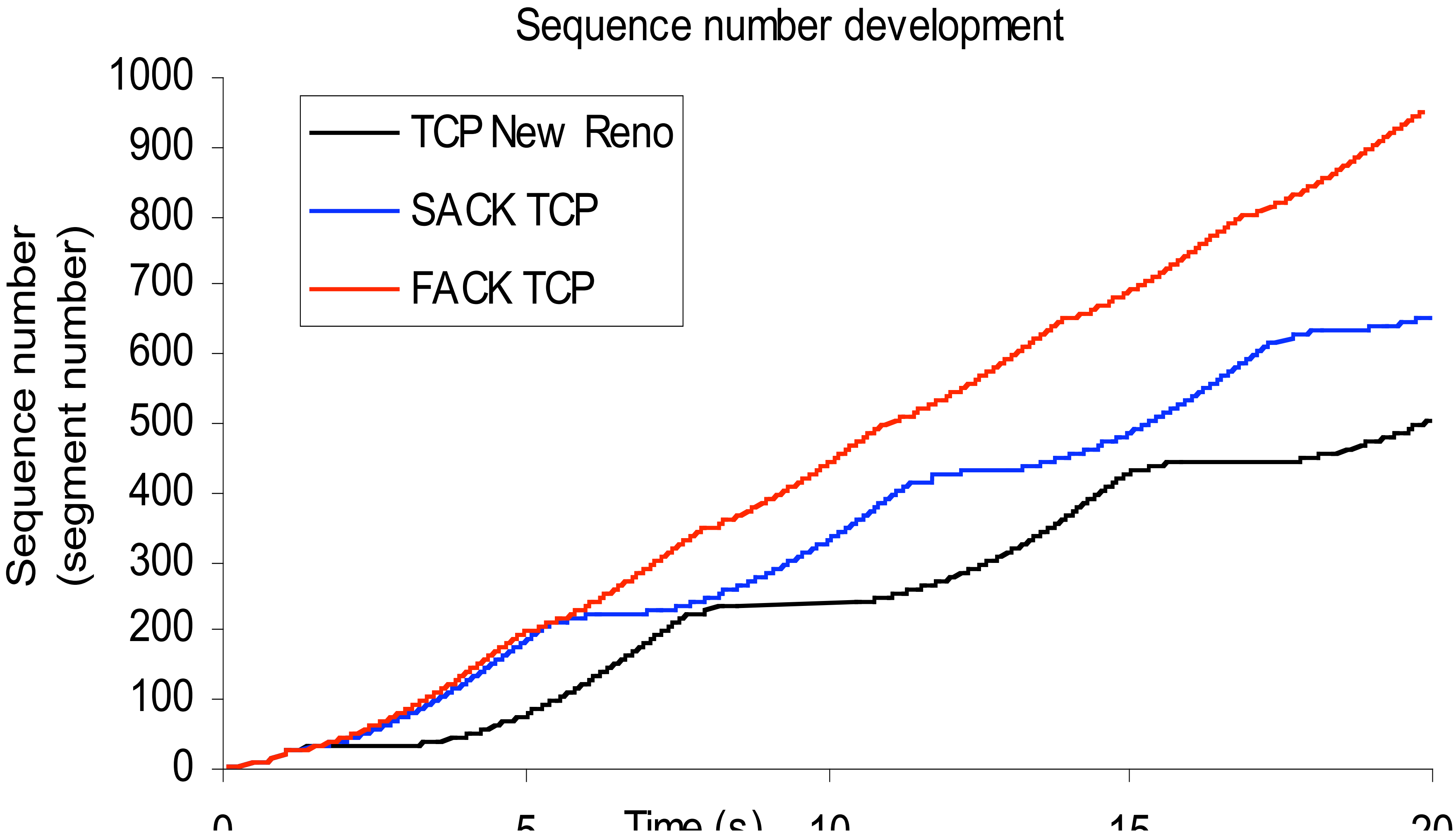
Feature	Original TCP	Tahoe	Reno	New-Reno	SACK	FACK
Retransmission strategy						
Slow start						
Congestion avoidance						
Fast retransmit						
Fast recovery						
Stay in f. rec.						
In flight packet estimation						
Cong. window halving						

TCP Congestion Control Alternatives

Feature	Original TCP	Tahoe	Reno	New-Reno	SACK	FACK
Retransmission strategy	Go back-n		Retransmit lost packet, continue after last sent		By SACK blk	
Slow start	No	Yes	Yes	Yes	Yes	Yes
Congestion avoidance	No	Yes	Yes	Yes	Yes	Yes
Fast retransmit	No	Yes	Yes	Yes	Yes	Yes
Fast recovery	No	No	Yes (3 duplicate ACKs)			
Stay in f. rec.	No	No	No	Yes	Yes	Consider gaps
In flight packet estimation	By TCP sequence number					By 1st SACK blk
Cong. window halving	Immediately					Spread out

Simulation results

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

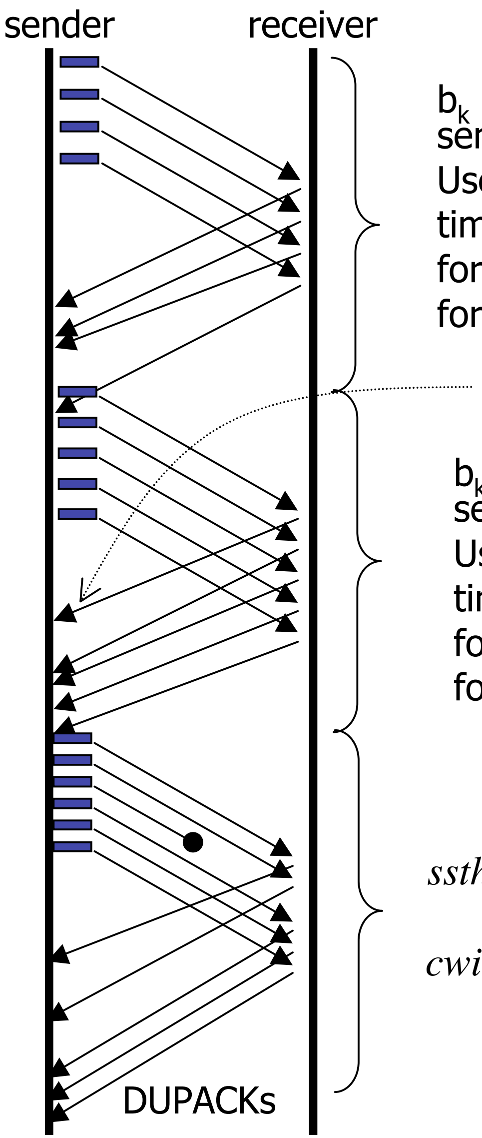


TCP Westwood+

- 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+



new $RTTmin$

b_k = estimate number of bytes sent in this RTT.
Uses average difference of time(sent) and time(ack'd) for every packet for this RTT

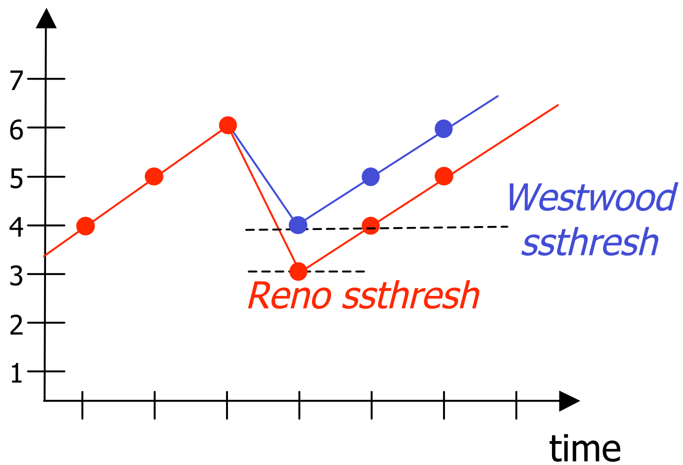
b_{k+1} = estimate number of bytes sent in this RTT.
Uses average difference of time(sent) and time(ack'd) for every packet for this RTT

$$ssthresh = \frac{l_{k+1} * RTT\min}{seg\_size}$$

$$cwin = \min(cwin, ssthresh)$$

l_k = estimate bytes that can be sent per times unit (e.g. second) uses a low pass filter (aging) to estimate longer-term development of bytes per RTT

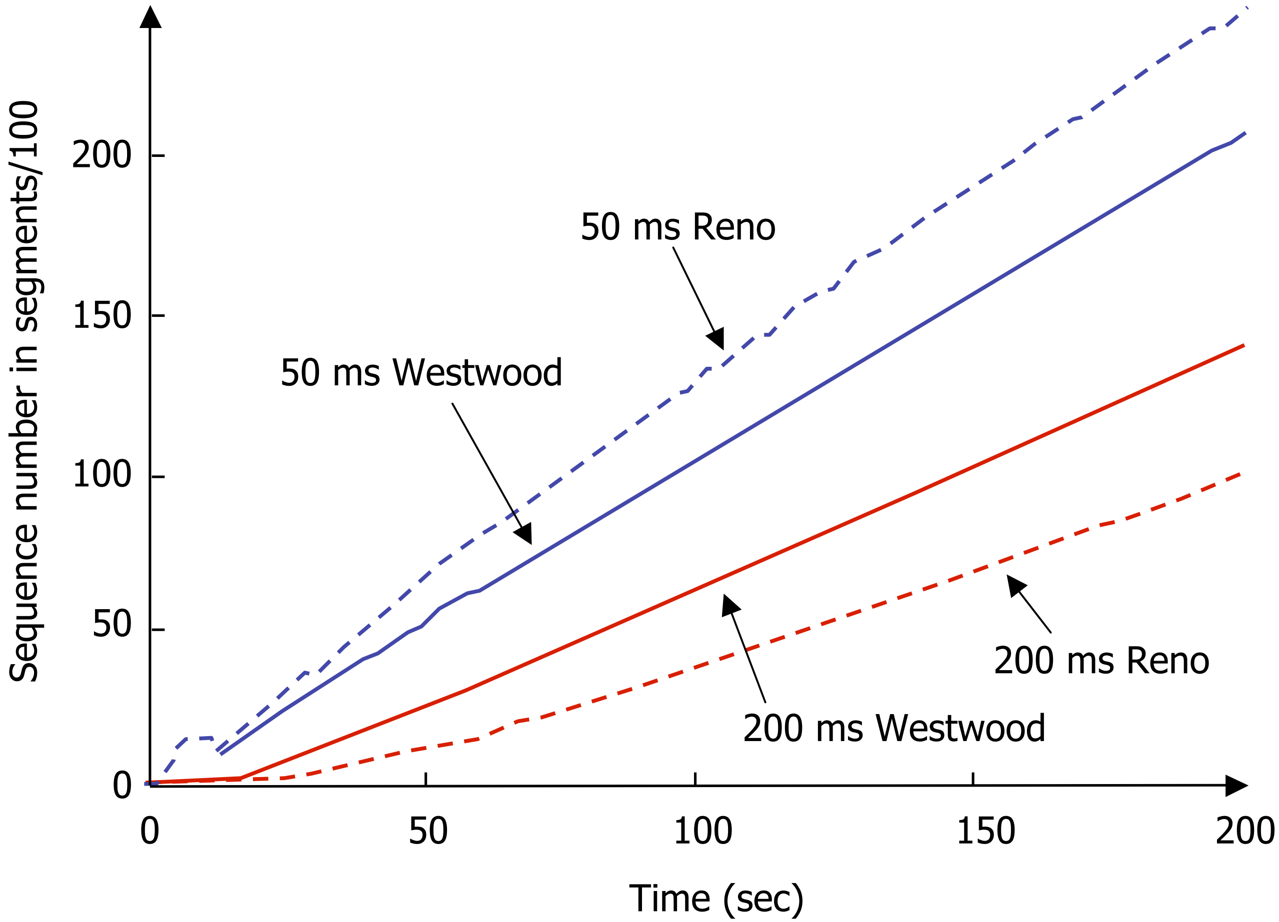
ssthresh = in case of loss, multiply l_k with the minimum RTT to get a minimum of bytes that have been supported per RTT.
Divide by segment size to get number of segments/RTT that should be supportable.



TCP Westwood+

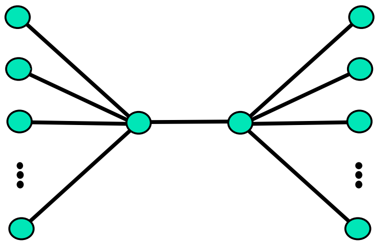
- TCP Westwood assumption
  - Immediately before the loss, TCPW was very close to its fair share. Therefore, on triple ACKs and DUPACKs, a state of congestion is reached and the previously used bandwidth is used for the congestion window size (not halving!)

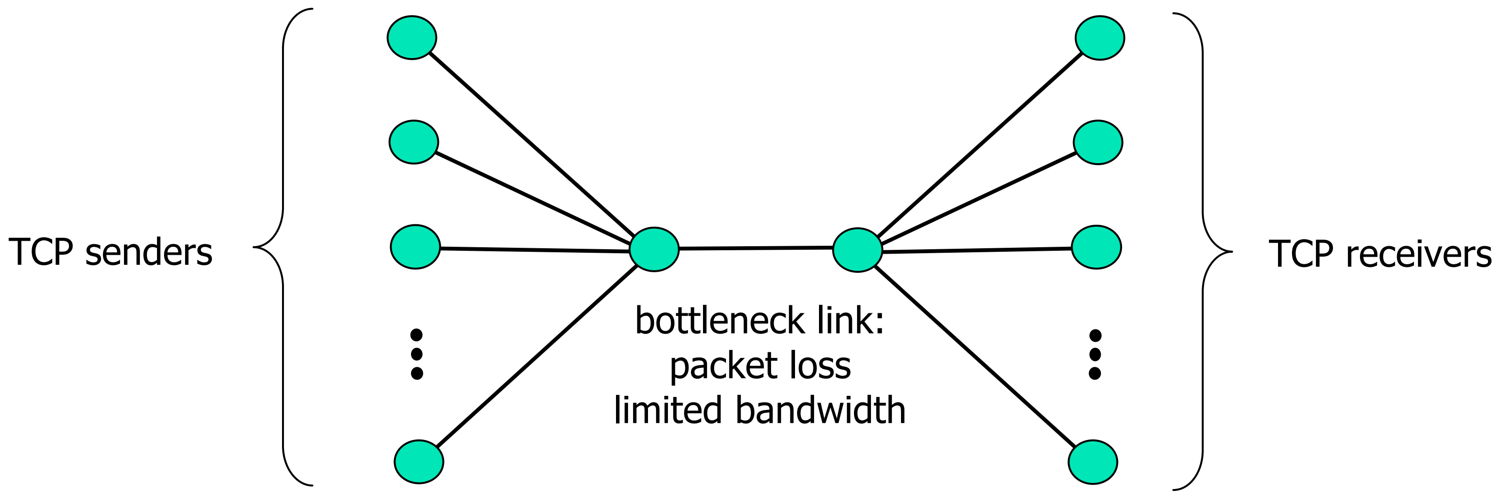
TCP Westwood+



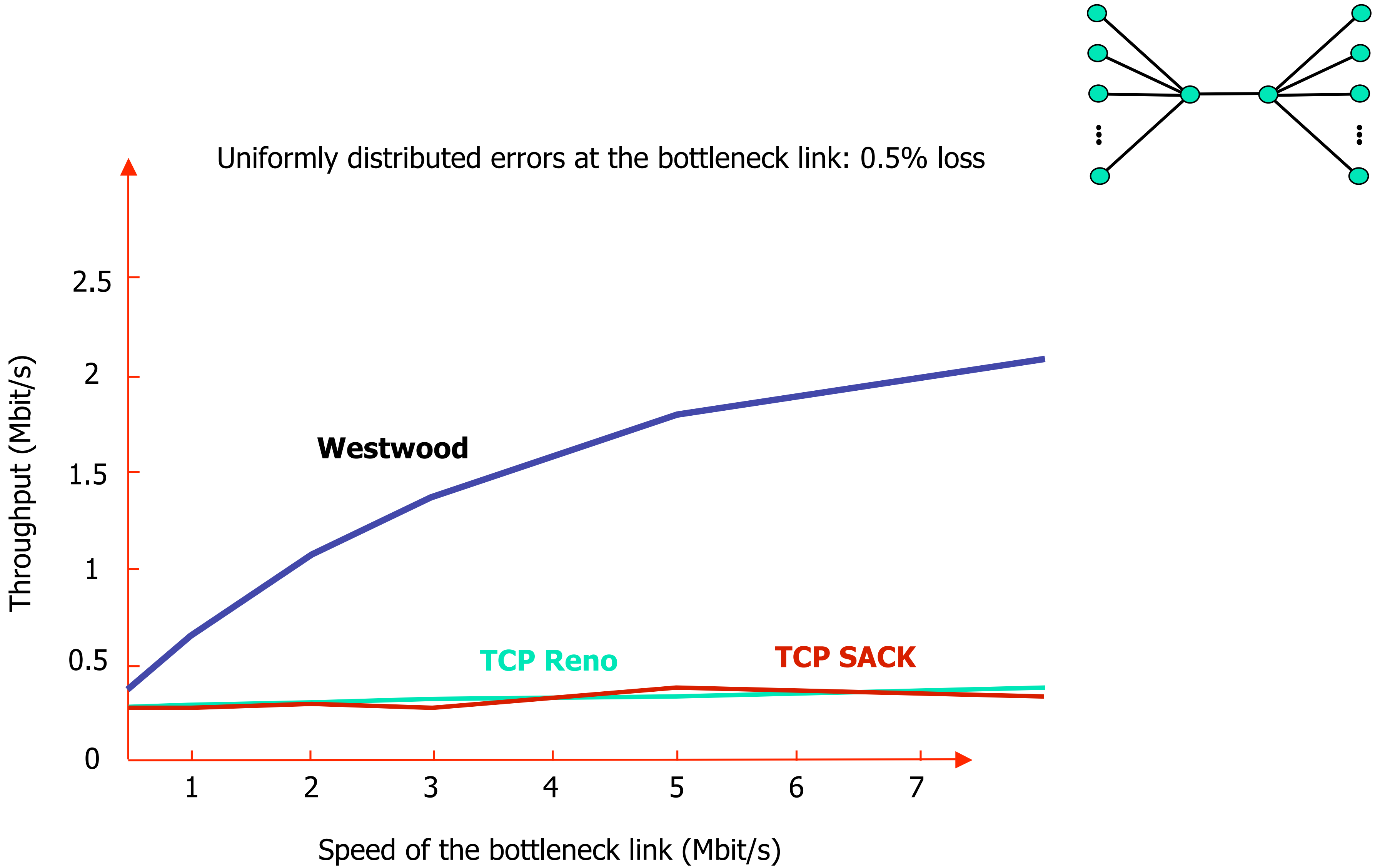
(approximation of a perf. eval. figure)

TCP Westwood+





TCP Westwood+



(approximation of a perf. eval. figure)

Random Early Detection (RED)

- Random Early Detection (discard/drop) (RED) uses active queue management

- Drops packet in an intermediate node based on average queue length exceeding a threshold
  - TCP receiver reports loss in ACK
  - sender applies MD

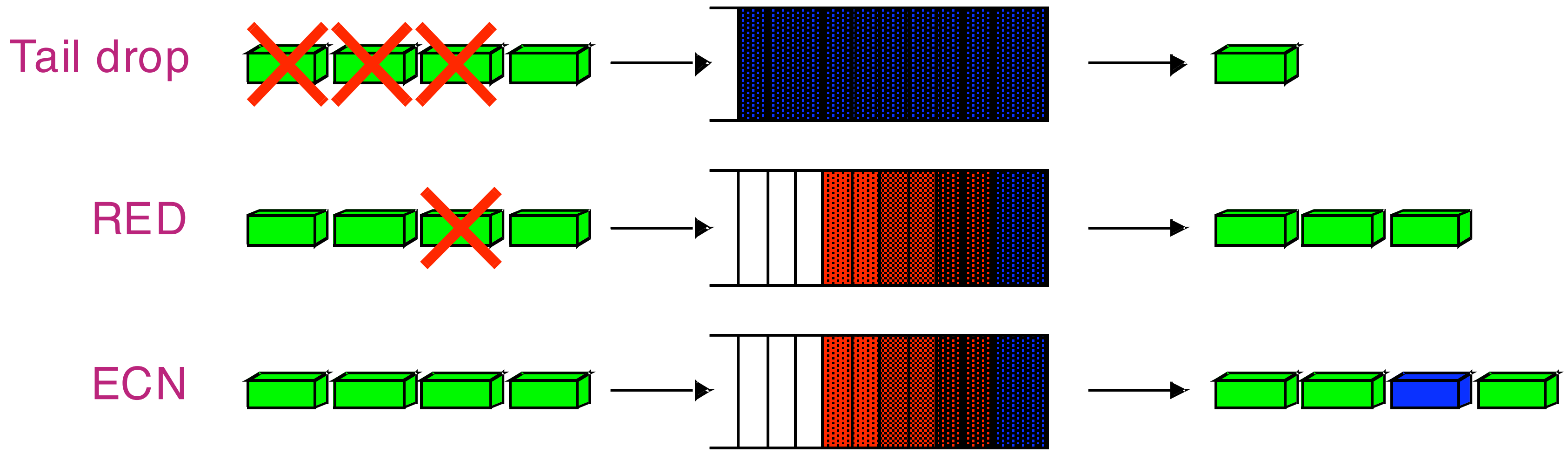
- Why?
  - if not, many TCPs loose packets at the same time
  - many TCP streams probe again at the same time
  - oscillating problems

Early Congestion Notification (ECN)

- Early Congestion Notification (ECN) - RFC 2481
  - an end-to-end congestion avoidance mechanism
  - implemented in routers and supported by end-systems
  - not multimedia-specific, but very TCP-specific
  - two IP header bits used
    - ECT - ECN Capable Transport, set by sender
    - CE - Congestion Experienced, may be set by router

- Extends RED
  - if packet has ECT bit set
    - ECN node sets CE bit
    - TCP receiver sets ECN bit in ACK
    - sender applies multiple decrease (AIMD)
  - else
    - Act like RED

Early Congestion Notification (ECN)



- (brief reminder of INF3190)
- Effects
  - Congestion is not oscillating - RED & ECN
  - ECN-packets are never lost on uncongested links
  - Receiving an ECN mark means
    - TCP window decrease
    - No packet loss
    - No retransmission

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
  - …

On–demand Streaming Applications

Stable bandwidth problem

TCP Congestion Control

- TCP congestion control is based on the notion that the network is a “black box” – congestion indicated by a loss

- Sufficient for best-effort applications, but losses might severely hurt traffic like audio and video streams
  → *congestion indication can enable features like quality adaptation*

UDP

- The classical solution
  - Send data at playout speed
  - Write loss-tolerant audio-video codecs
  - Ignore all kinds of loss

- Problem
  - Does not back off at bandwidth bottlenecks
  - TCP connections suffer

⇒ Approach is no longer accepted

Comparison of Non-QoS Philosophies

Pro UDP	Pro TCP

Using Standard Protocols

<table>
<tr><th>Over UDP</th><th>Over TCP</th><th>Alternative Transport</th></tr>
<tr><td>RTP
Real Time Protocol
IETF std, supported by ITU-T & Industry</td><td>RTP in RTSP over TCP
standardized worst-case fallback
firewall-friendly</td><td>SCTP
Stream Control Transmission Protocol
IETF RFC, supported by telephone industry</td></tr>
<tr><td>RLM
TCP-friendly, needs fine-grained layered video</td><td rowspan="2">"Progressive Download" or "HTTP Streaming"
application-level prefetching and buffering
trivial, cheap, firewall-friendly</td><td rowspan="2">DCCP
Datagram Congestion Control Protocol
IETF RFC, driven by TCP-friendliness researchers</td></tr>
<tr><td>SR-RTP
TCP-friendly with RTP/UDP
needs special encoding (OpenDivX)</td></tr>
<tr><td>VDP
Video Datagram Protocol
Research, for Vosaic</td><td rowspan="2">Priority Progress Streaming
needs special encoding
needs special routers for ’multicast’</td><td rowspan="2">PRTP-ECN
Partially reliable transport protocol using ECN
Research, Univ. Karlstad</td></tr>
<tr><td>MSP
Media Streaming Protocol
Research, UIUC</td></tr>
</table>

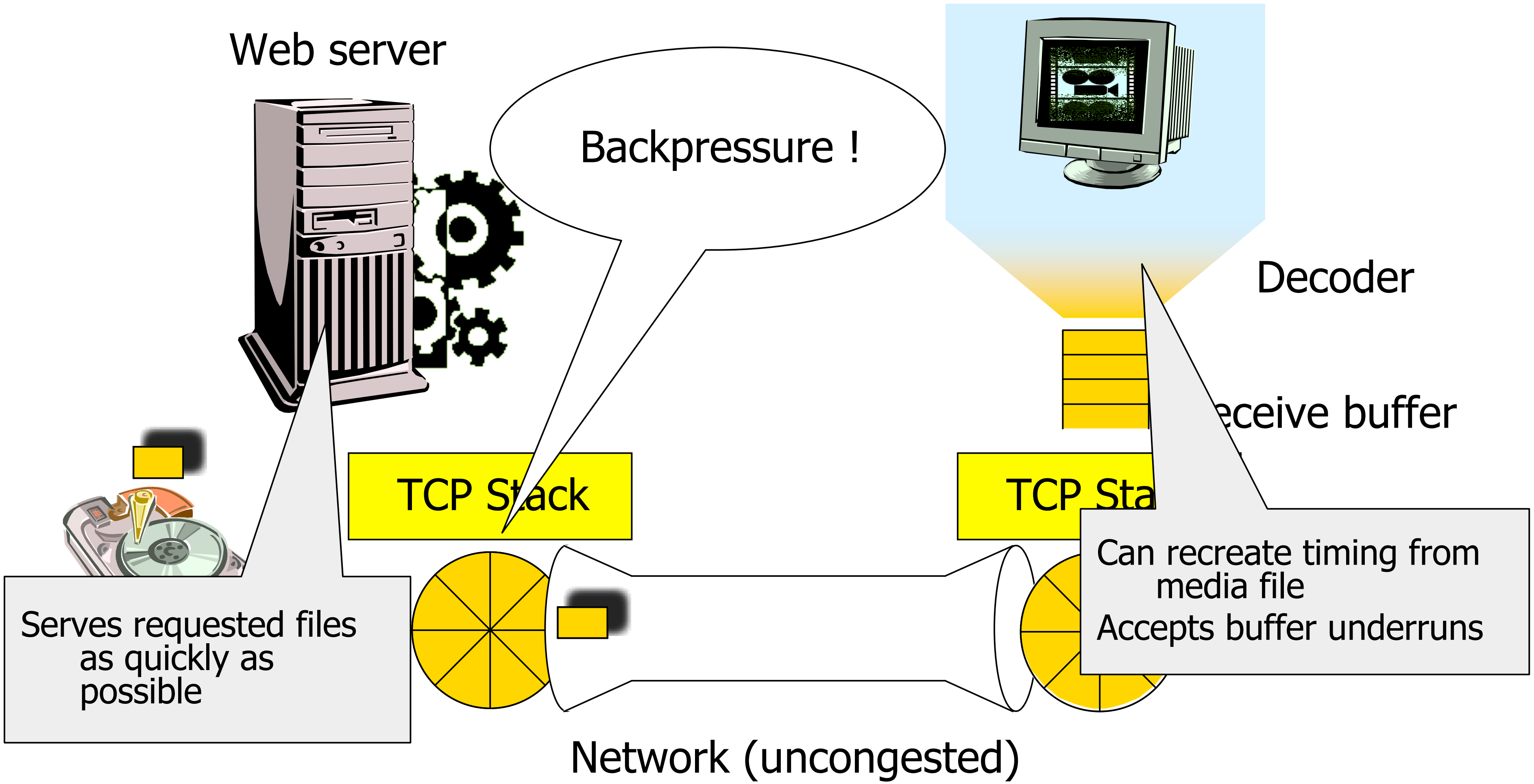
Progressive Download

- In-band in long-running HTTP response
  - Plain file for the web server
  - Even simpler than FTP
  - No user interactions – start, stop, ...

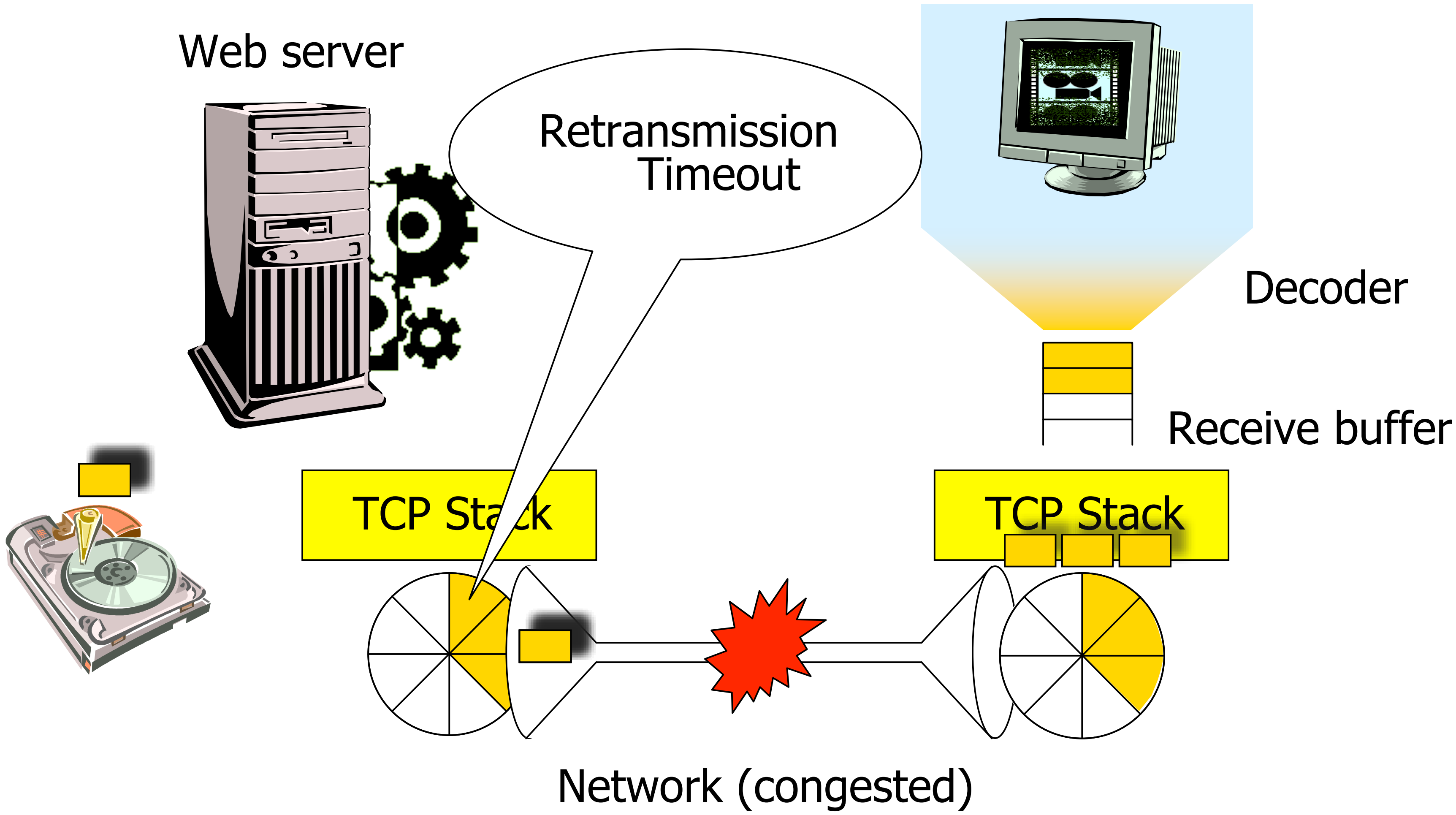
- If packet loss is ...
  - ... low – rate control by back-pressure from client
  - ... high – client’s problem

- Applicability
  - Theoretical
    - For very low-bit-rate codecs
    - For very loss-intolerant codecs
  - Practical
    - All low-volume web servers

Progressive Download



Progressive Download



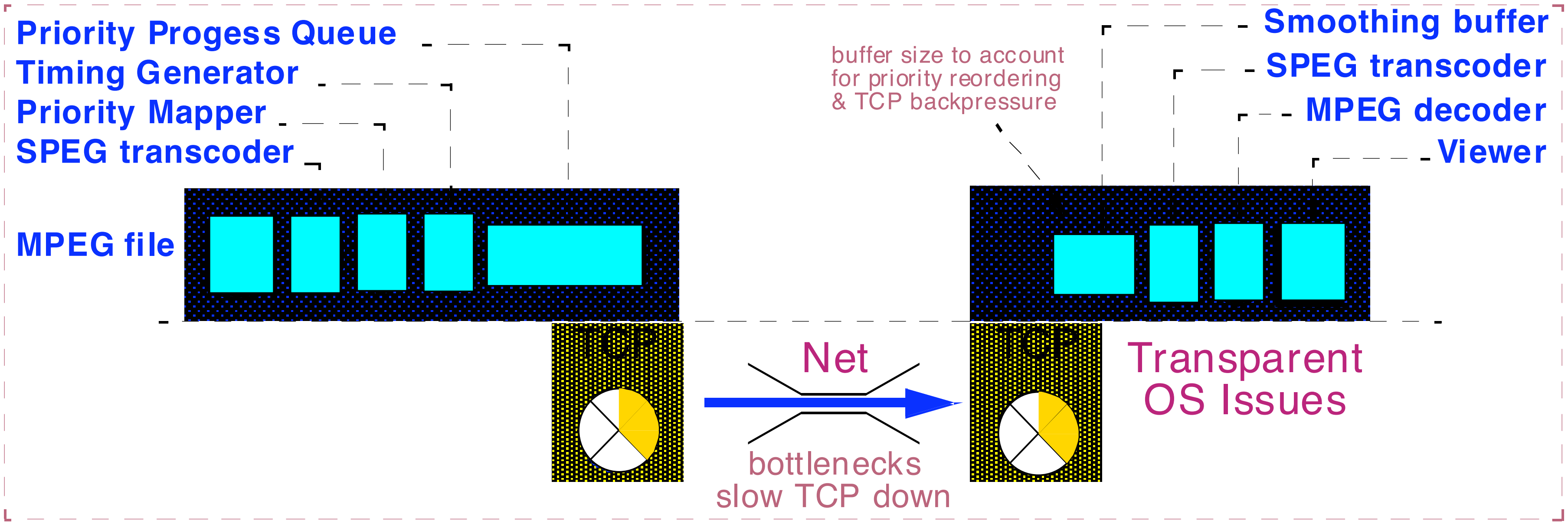
Priority Progress Streaming

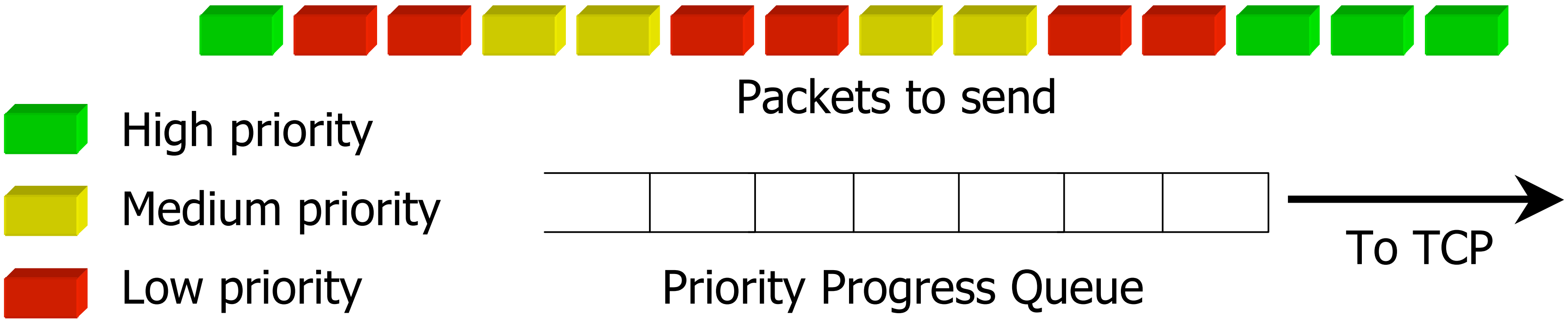
- Unmodified TCP (other transports conceivable)
- Unmodified MPEG-1 video-in (other encoding formats conceivable)

- Real-time video processing
  - Convert MPEG to Spatially Scalable MPEG (SPEG) – 10-25% overhead
  - Packetize SPEG to separate by frame and by SNR quality step
    - More variations conceivable: color, resolution
  - Assign priorities to SPEG packets
    - Dynamic utility curves indicate preference for frame or SNR dropping
  - Write SPEG packets in real-time into reordering priority progress queue

- Queues are long
  - Much longer than TCP max window
  - Dynamically adjustment allows fast start and dynamic growth
  - With longer queues
    - Total delay is increased
    - High priority packets win more often

Priority Progress Streaming





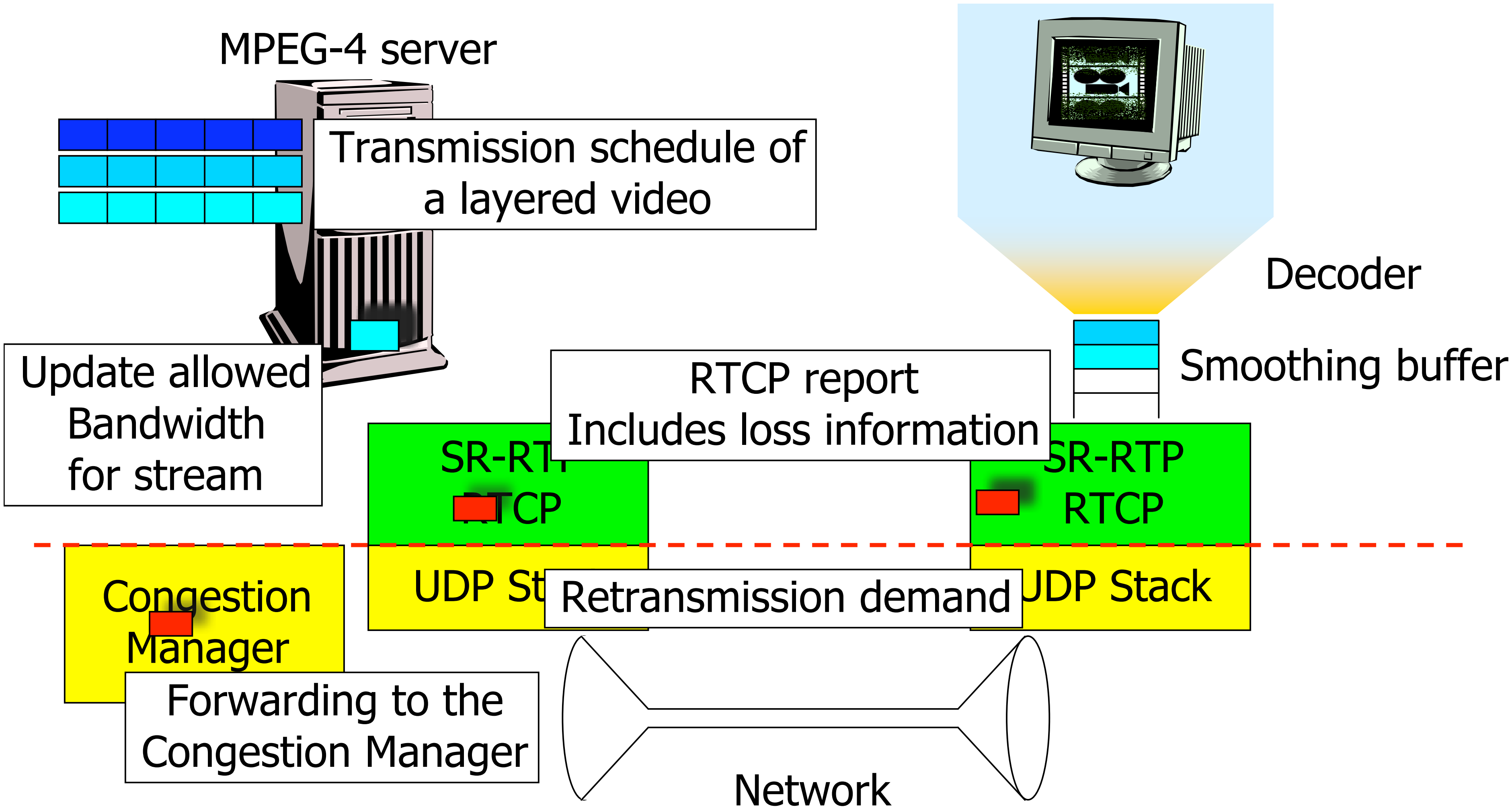
Selective Retransmission–RTP (SR–RTP)

- Features
  - Relies on a layered video codec
  - Supports selective retransmission
  - Uses congestion control to choose number of video layers

- Congestion Manager
  - Determines the permitted send rate at the sender
  - Uses TCP-friendly algorithm for rate computation

- Knowledge about encoding
  - Required at sender to select video layers to send
  - Required at receiver to
    - decode at correct rate
    - send NAKs

Selective Retransmission–RTP (SR–RTP)



Selective Retransmission–RTP (SR–RTP)

- Binomial Congestion Control
  - Provides a generalization of TCP AIMD

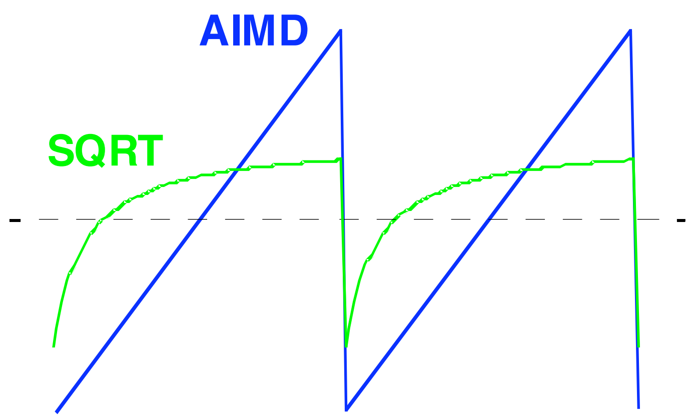
Increase	Decrease
$w_{t+RTT} = w_t + \alpha / w_t^k, \alpha > 0$	$w_{t+RTT} = \beta \cdot w_t^l, 0 < \beta < 1$

  - Congestion window size w_t depends on losses per RTT
  - TCP's AIMD: α = 1, β = .5, k = 0 and l = 1
  - k + l = 1: binomial congestion control is TCP friendly

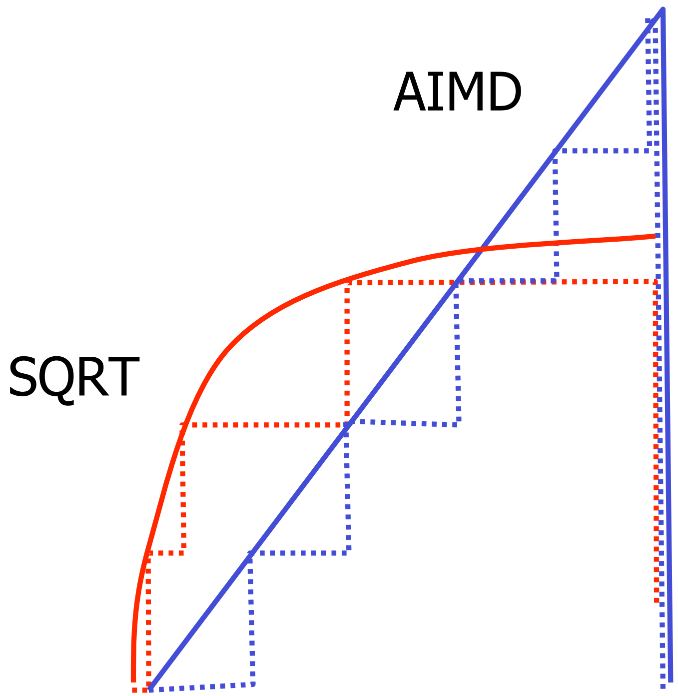
Nick Feamster and Hari Balakrishnan

Selective Retransmission–RTP (SR–RTP)

- **SQRT**
  - Special case of binomial congestion control
  - k=0.5, l=0.5
  - Name because $w^{0.5}$ = sqrt(w)



- **Effect of SQRT**
  - Average bandwidth is like TCP's
  - Maximum is lower
  - SQRT covers a step function with less steps



Datagram Congestion Control Protocol (DCCP)

- Datagram Congestion Control Protocol

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

- 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

Steady state TCP send rate

Loss probability

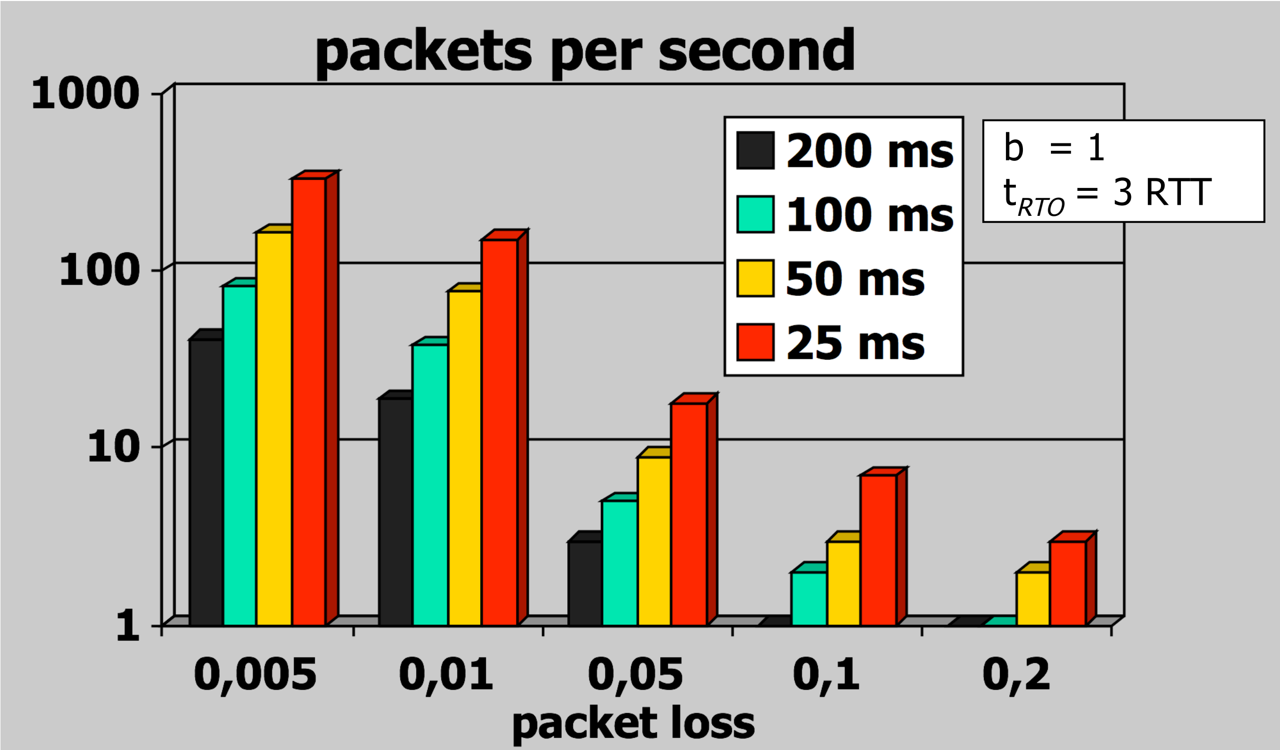
$$T = \frac{1}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO}\min(1, 3\sqrt{\frac{3bp}{8}})p(1+32p^2)}$$

Retransmission timeout

Number of packets ack'd by one ACK

Padhye's TCP New Reno estimation formula

Datagram Congestion Control Protocol (DCCP)



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

On-demand streaming applications

- Smoothness is key
  - Use a lot of buffering
  - Don’t surprise the application
  - Consume a limited amount of buffers
  - Try to make congestion control as smooth as possible

- Adaptive applications
  - Can by improved by this

Some References

- Charles Krasic, Jonathan Walpole, Wu-chi Feng: "Quality-Adaptive Media Streaming by Priority Drop", 13th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2003), June 2003
- Charles Krasic, Jonathan Walpole: "Priority-Progress Streaming for Quality-Adaptive Multimedia", ACM Multimedia Doctoral Symposium, Ottawa, Canada, October 2001
- Kurose, J.F., Ross, K.W.: "Computer Networking – A Top-Down Approach Featuring the Internet", 2nd ed. Addison-Wesley, 2003
- The RFC repository maintained by the IETF Secretariat can be found at http://www.ietf.org/rfc.html
  The following RFCs might be interesting with respect to this lecture:
  - RFC 793: Transmission Control Protocol
  - RFC 2988: Computing TCP's Retransmission Timer
  - RFC 768: User Datagram Protocol
  - RFC 2481: A Proposal to add Explicit Congestion Notification (ECN) to IP
  - RFC 1889: RTP: A Transport Protocol for Real-Time Applications
  - RFC 1890: RTP Profile for Audio and Video Conferences with Minimal Control
  - RFC 2960: Stream Control Transmission Protocol
  - RFC 2326: Real Time Streaming Protocol