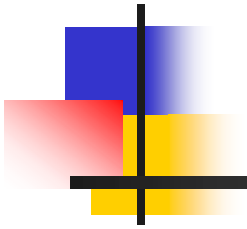
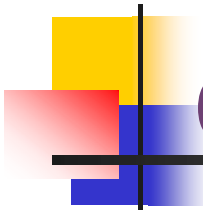
INF 5070 – Media Servers and Distribution Systems:

Protocols without QoS Support



19/9 - 2005



Overview

Non-QoS protocols:

- Transport level protocols
- Application layer protocols
- Signaling protocols

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

Basic Techniques

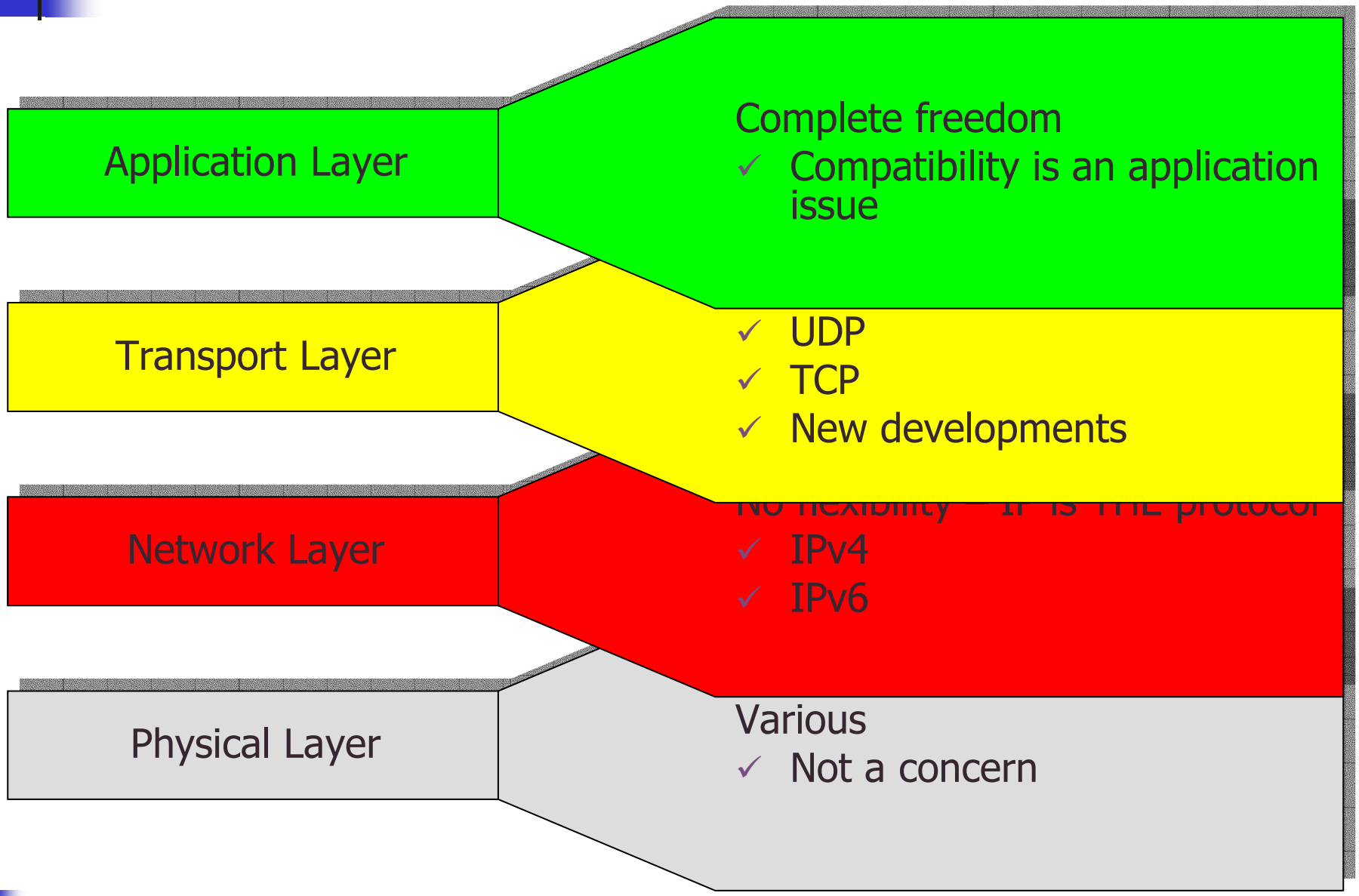
- Delay and jitter
  - Reservation (sender, receiver, network)
  - Buffering (receiver)
  - Scaling (sender)
- Continuity
  - Real-time packet re-ordering (receiver)
  - Loss detection and compensation
  - Retransmission
  - Forward error correction
  - Stream switching (encoding & server)
- Synchronity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

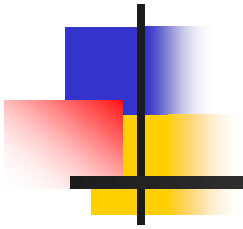
QoS vs. Non–QoS Approaches

- 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

Protocols for Non–QoS Approaches





Traditional Transport Layer Approaches

User Datagram Protocol (UDP)

- Described in
  - Internet Engineering Note 88
  - Request for Comments 768
  - Internet Standard 6
- Connection-less
- Unreliable
- Unordered
- Datagram-oriented
- Uncontrolled
- Optionally checksummed
- Often used in multimedia streaming applications – can build whatever needed on top
- Simple
  - add pseudo-header (UDP/IP)
  - calculate checksum
  - finish header
  - send to IP
- No...
  - ... guarantees
  - ... fairness
  - ...

Transmission Control Protocol (TCP)

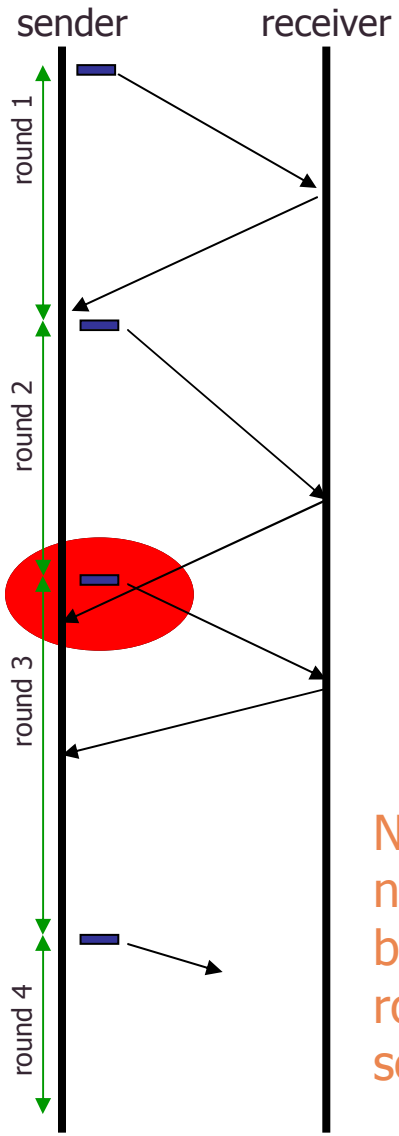
- Described in
  - Internet Engineering Note 2
  - Request for Comments 793
  - Internet Standard 7
- Connection-oriented
- Reliable
- Ordered
- Stream-oriented
- Flow-controlled
- Checksummed
- Complex compared to UDP

- **High** fraction of today's traffic is TCP-based, e.g.,
  - electronic mail
  - web
  - file transfers
  - ...
- We need to know some details about the TCP behavior/traffic → we'll briefly look at TCP ...
  - ...retransmission timeouts
  - ...congestion control
  - ...friendliness

TCP Round Trip Time Estimation

EstimatedRTT = (1 - α) * EstimatedRTT + α * SampleRTT

usually, α = 0.125 [RFC 2988]



Initially, EstimatedRTT is based on packets sent during "handshake" operation (connection setup), e.g., 3

The following RTTs are calculated using the formula above taking one SampleRTT each round:

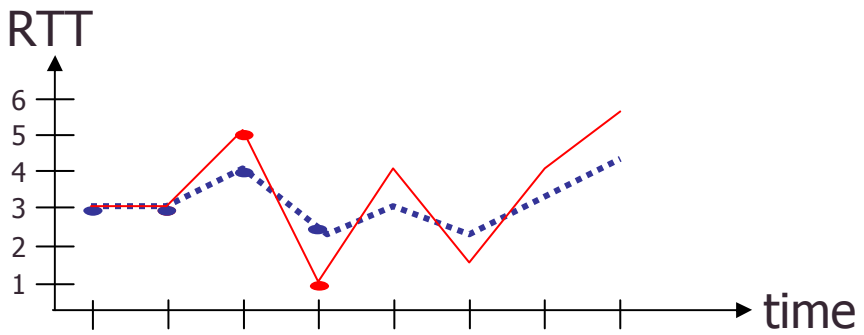
- going slightly **up** if EstimatedRTT **<** SampleRTT
- going slightly **down** if EstimatedRTT **>** SampleRTT
- e.g. (α = 0.5):

2) **EstimatedRTT** = .5 * 3 + .5 * **3** = 3

3) **EstimatedRTT** = .5 * 3 + .5 * **5** = 4

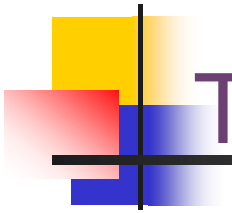
4) **EstimatedRTT** = .5 * 4 + .5 * **1** = 2.5

NOTE: the next RTT is not necessarily ready before the corresponding round starts (and we start sending the next packets)



TCP Retransmission Timeout

- The EstimatedRTT is used for the retransmission timer – timeout interval should be
  - ≥ estimatedRTT – avoiding unnecessary retransmissions
  - but not too much larger – slow retransmit, large delay
  - margin should be large when lot of fluctuations, otherwise small
- Additionally, TCP uses RTT variation – deviated RTT:
  DevRTT = (1 - β) * DevRTT + β * | SampleRTT – EstimatedRTT |, usually, β= 0.25
- Retransmission interval given by
  TimeoutInterval = EstimatedRTT + 4 * DevRTT
- Modifications
  - timeout interval doubling each timeout (form of congestion control)
  - fast retransmission – three duplicate ACKs (decrease delay)

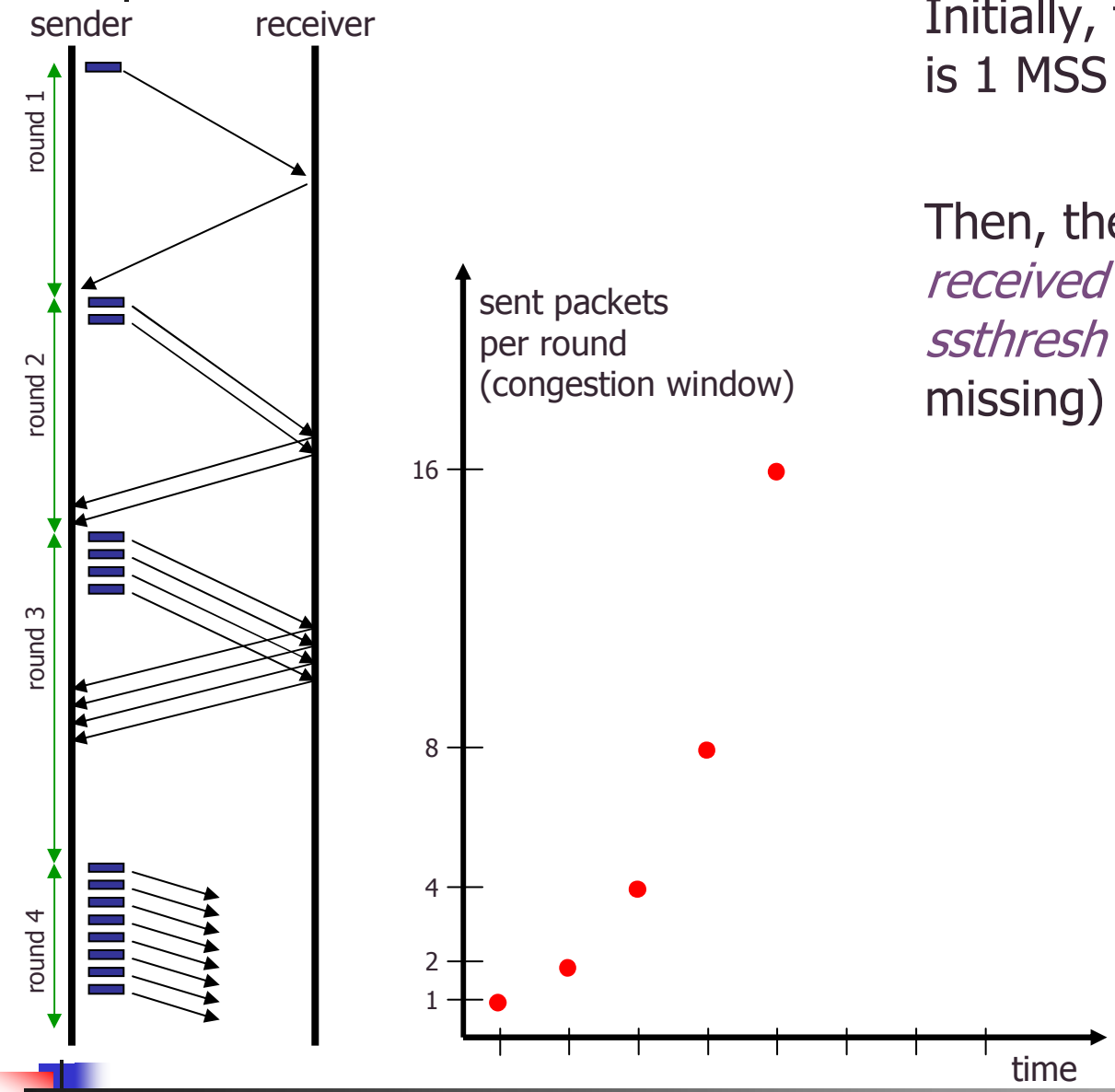


TCP Congestion Control

- 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

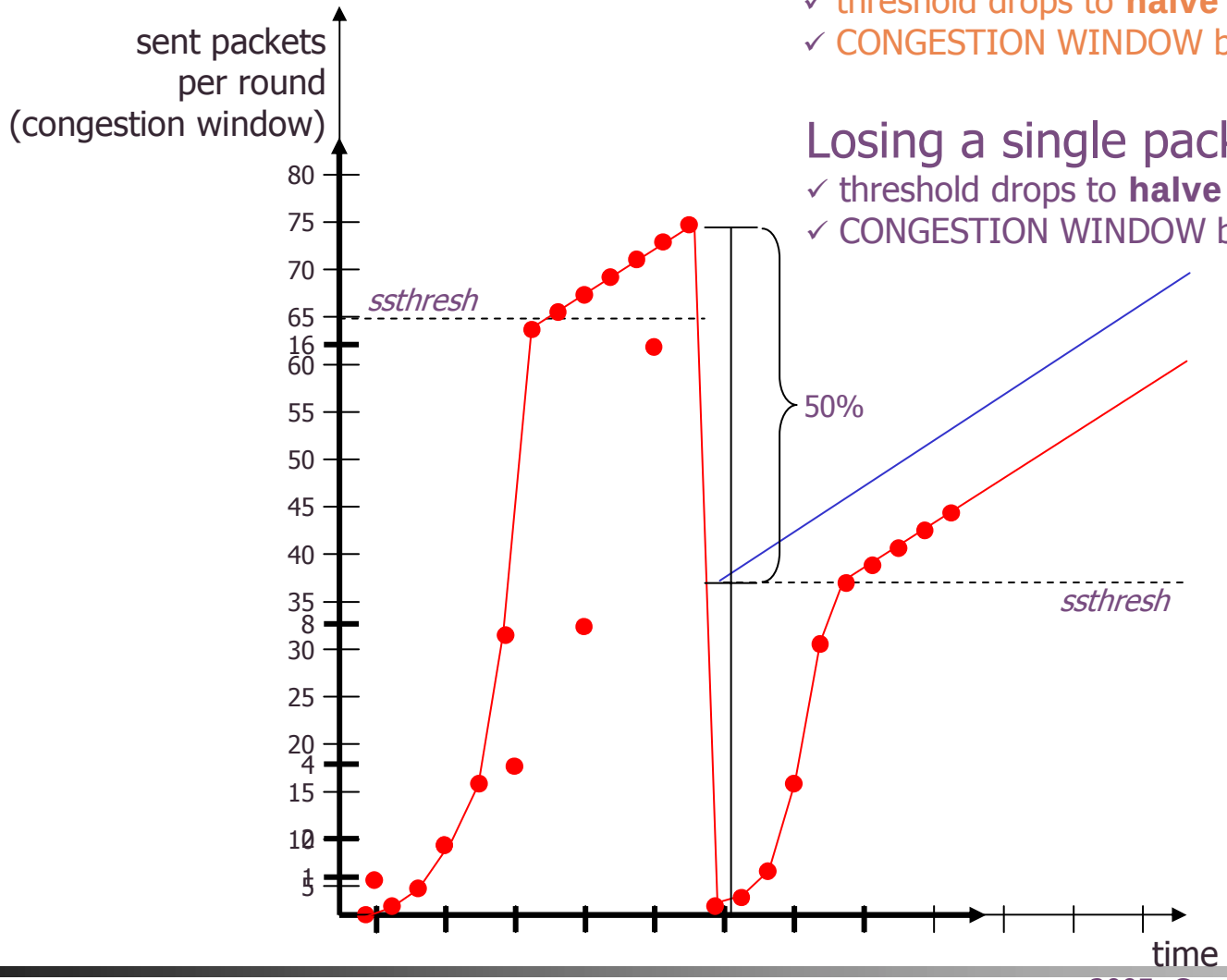
Normally, the threshold is 65 K

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 Friendliness

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

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:

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

That's not generally true

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

TCP Friendliness

- 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*

$$R_r \leq \sqrt{p} + C$$

P – packet loss rate

C – constant value

R_r – packet arrival 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

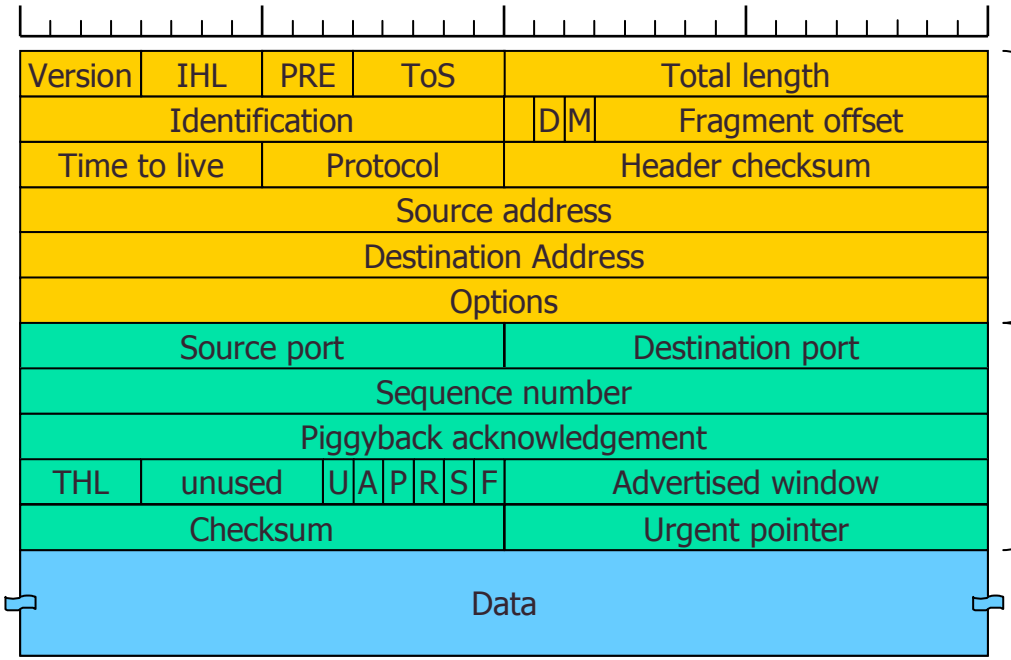
- Why alternatives?
  - Improve throughput and variance
    - Early TCP implementations did little to minimize network congestion
    - Loss indication forces setting new congestion window threshold to half of the last congestion window size
    - But …
      - … what else to conclude from the loss?
      - … 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

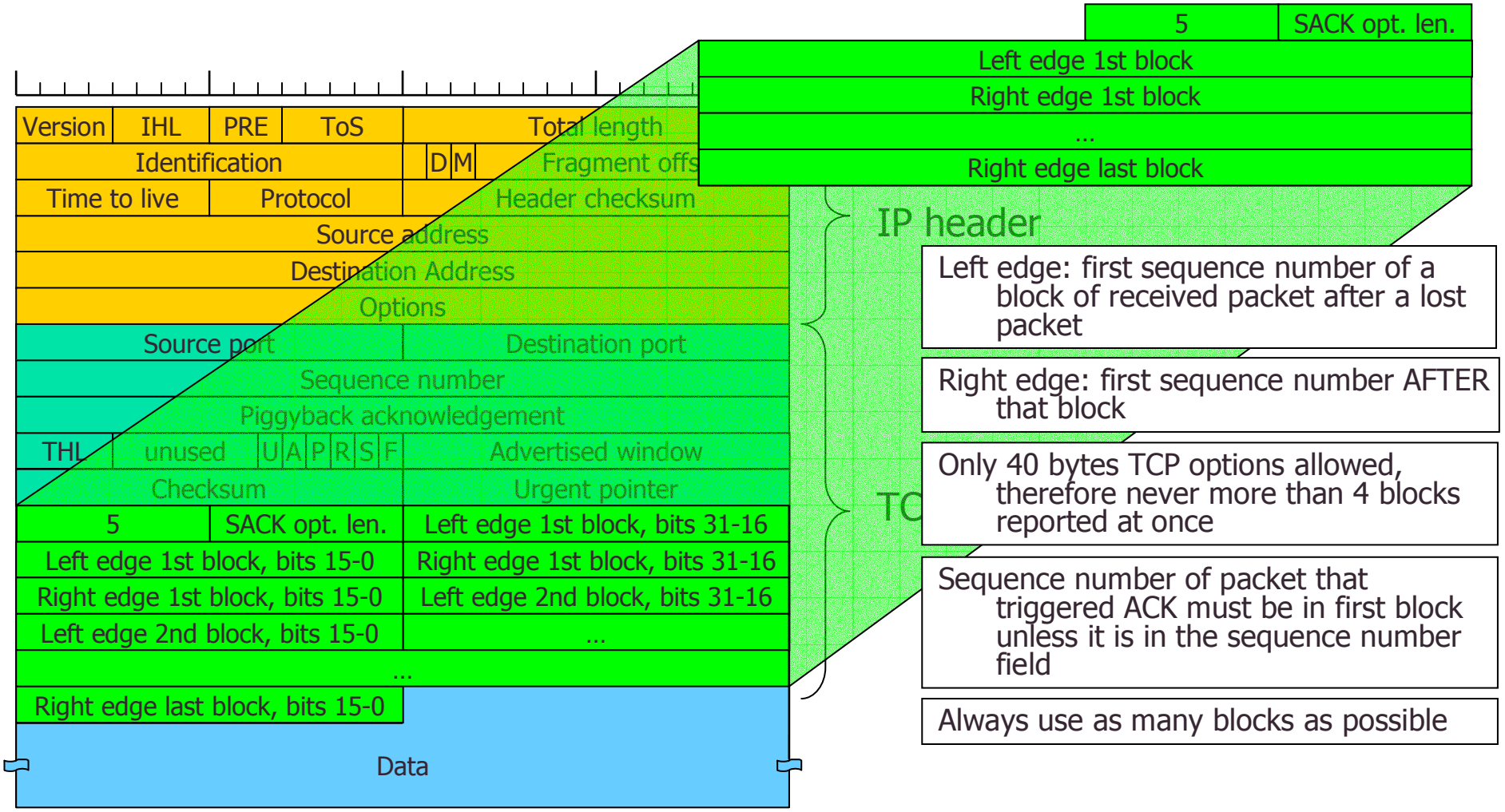
- TCP/IP Header Format for TCP Tahoe, Reno and New Reno



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



TCP Congestion Control Alternatives

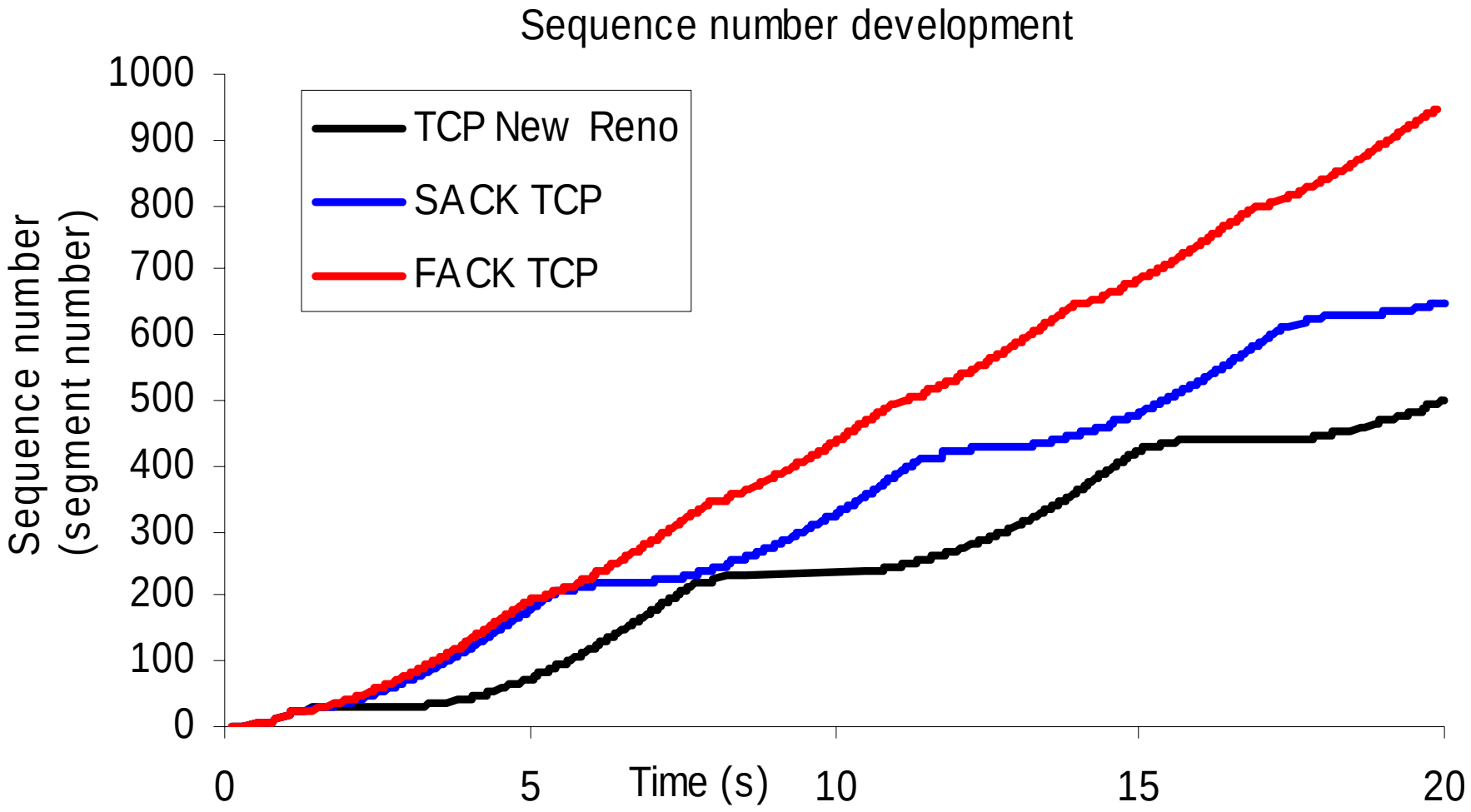
Feature	Original TCP	Tahoe	Reno	New-Reno	SACK	FACK
Retransmission strategy						
Slow start						
Congestion avoidance						
Fast retransmit						
Fast recovery						
Stay in f. rec.						
In flight packet estimation						
Cong. window halving						

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):

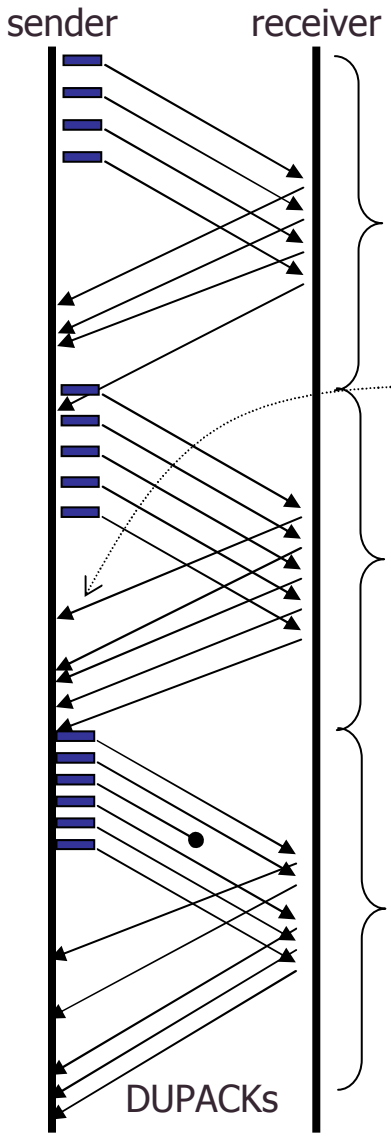


TCP Westwood+

- Very recent
- Developed for wireless networks with many losses
  - Losses in wireless networks are often ***non-congestion losses*** but corruption losses
- Side effect
  - Less unfair against long round-trip times
- Implemented in Linux

- 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+



$$b_k = \frac{seg\_size}{\Delta_k} \qquad \Delta_k = \frac{\sum_{packets} t_{ack} - t_{send}}{|packets|}$$

new *RTTmin*

$$b_{k+1} = \frac{seg\_size}{\Delta_{k+1}} \qquad \Delta_{k+1} = \frac{\sum_{packets} t_{ack} - t_{send}}{|packets|}$$

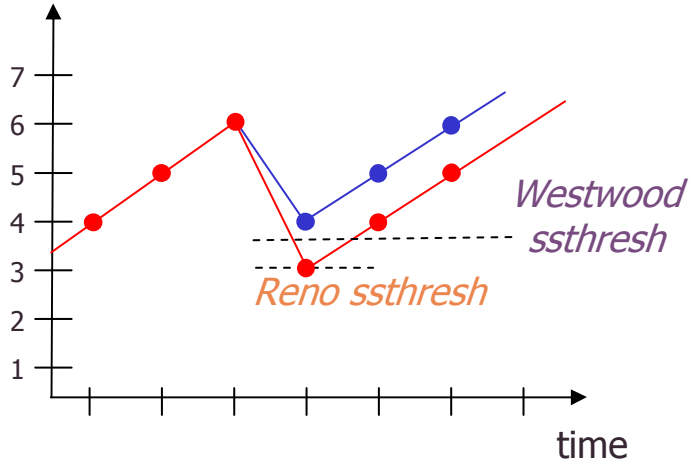
$$ssthresh = \frac{\dot{b}_{k+1} * RTT\min}{seg\_size}$$

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

Low pass filter for bandwidth estimate:

$$\dot{b}_k = \frac{\frac{2\tau}{\Delta_k} - 1}{\frac{2\tau}{\Delta_k} + 1}\dot{b}_{k-1} + \frac{b_k + b_{k-1}}{\frac{2\tau}{\Delta_k} + 1} \qquad \tau = 1s$$

$$if\ (\Delta_k > \tau/2) : b_0 = 0$$



TCP Westwood+

- 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+

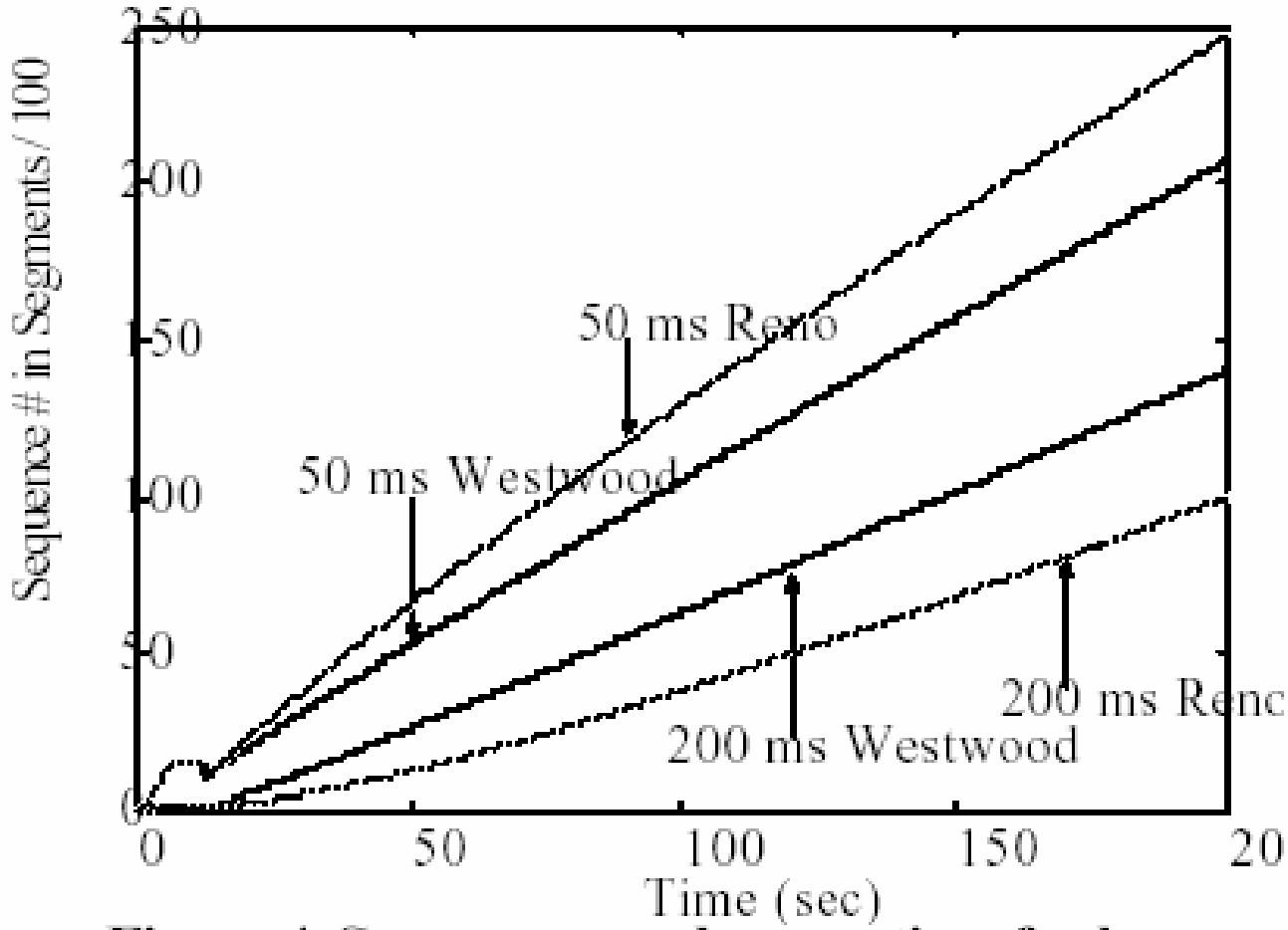


Figure 4. Sequence numbers vs. time for long and short RTT connections without RED

TCP Westwood+

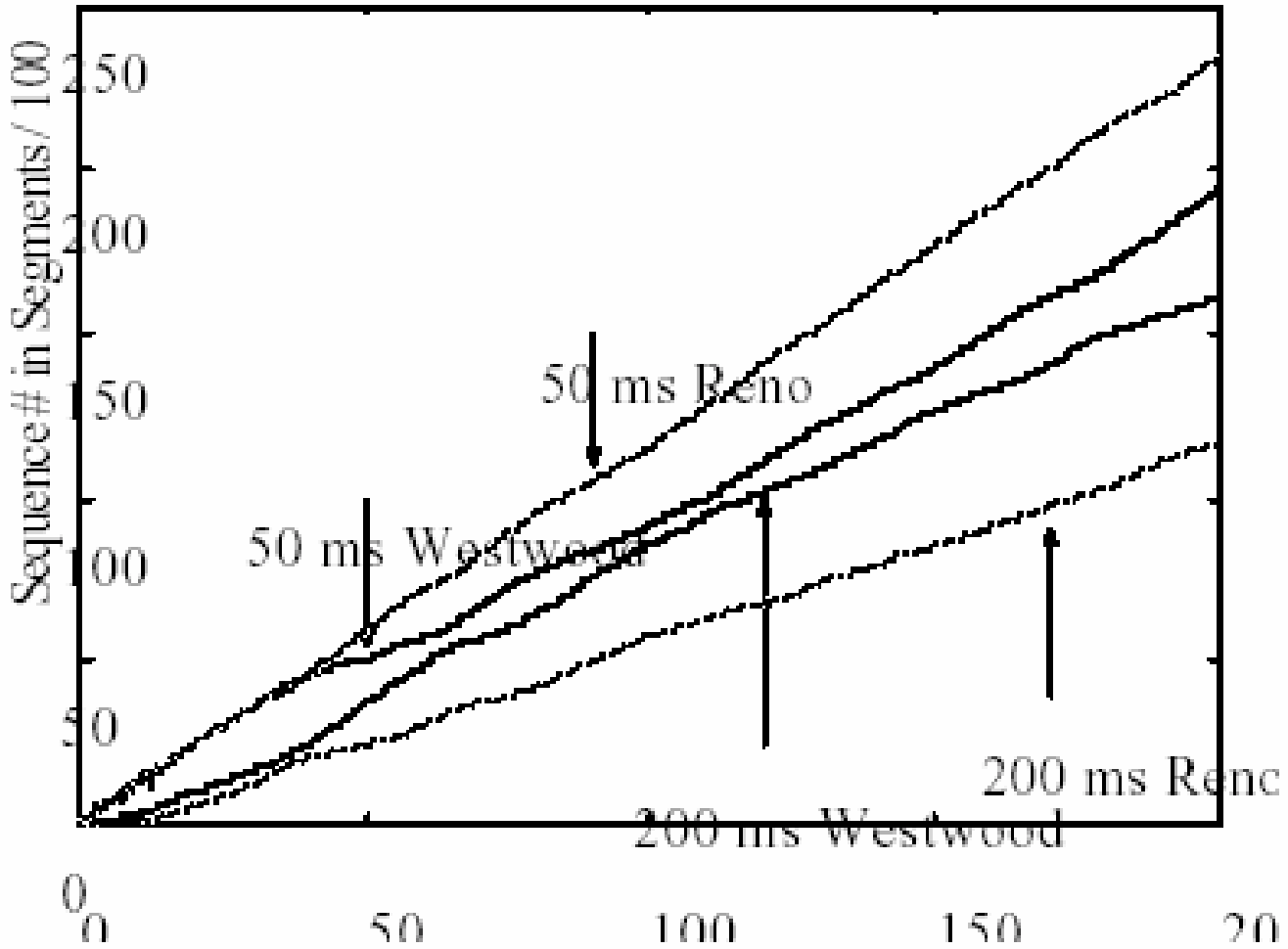


Figure 5. Sequence numbers vs. time for long and short RTT connections with RED

TCP Westwood+

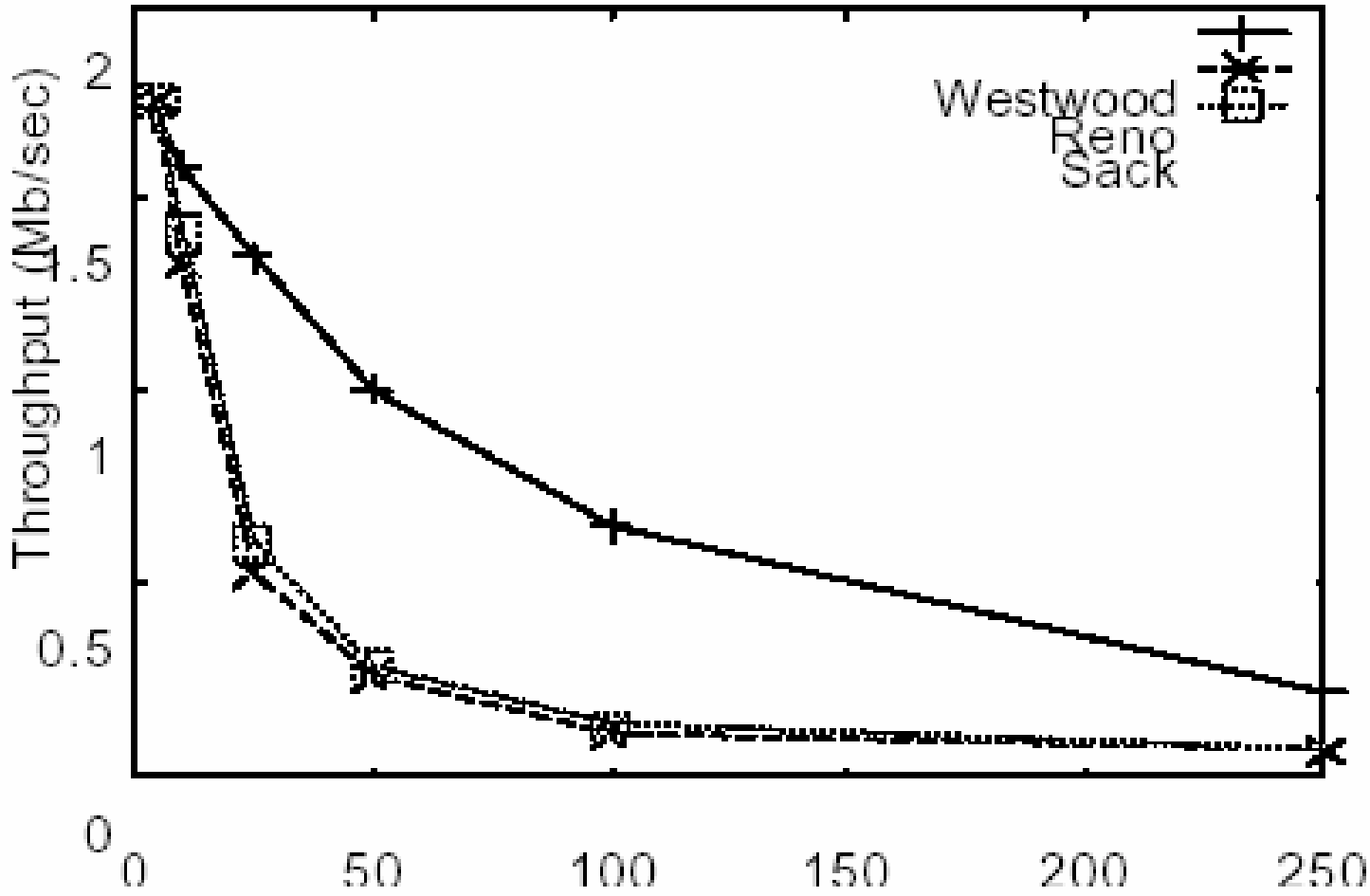


Figure 10. Throughput vs. one-way propagation delay

TCP Westwood+

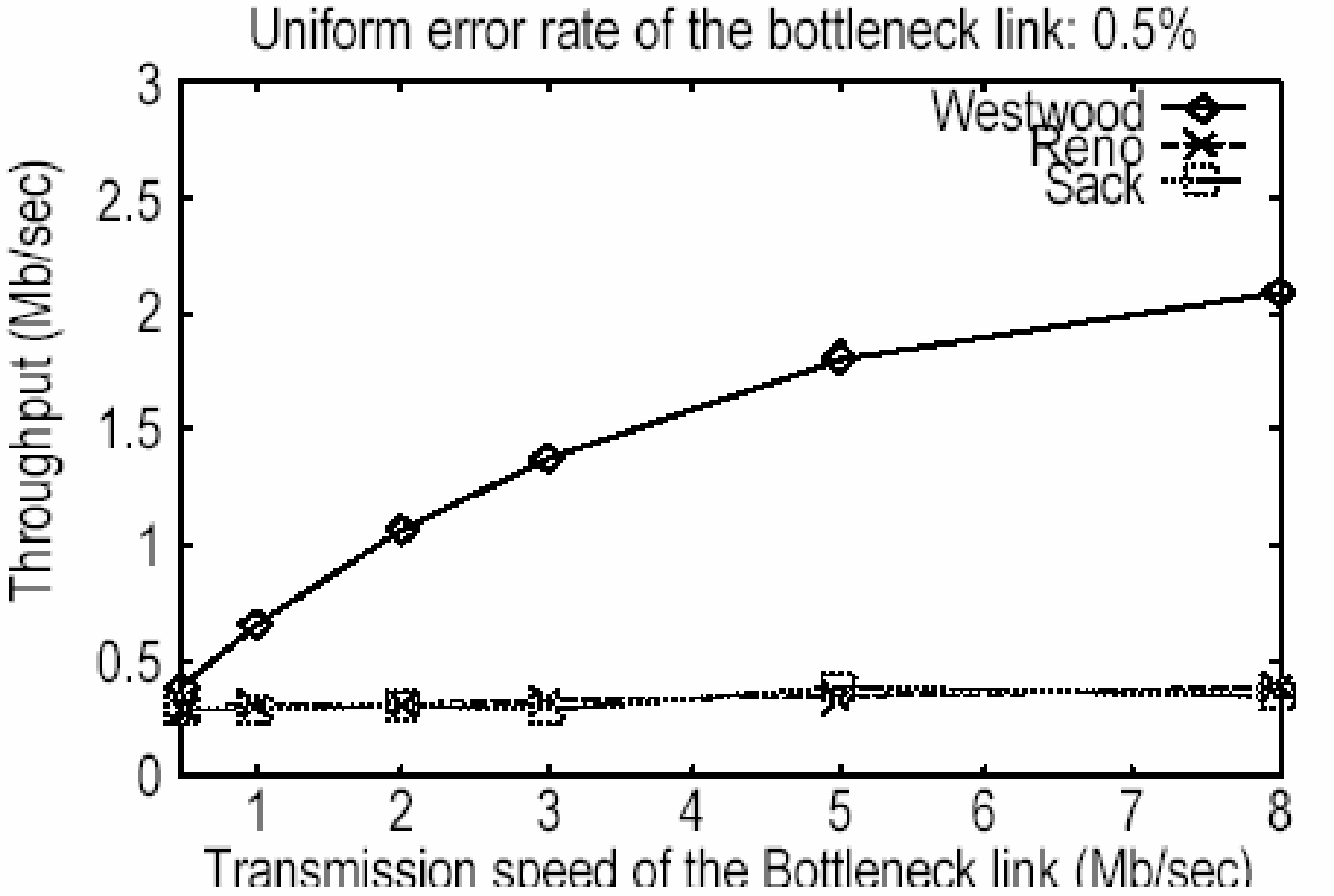


Figure 11 Throughput vs. link capacity

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*

- Use *active queue management* to detect congestion

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

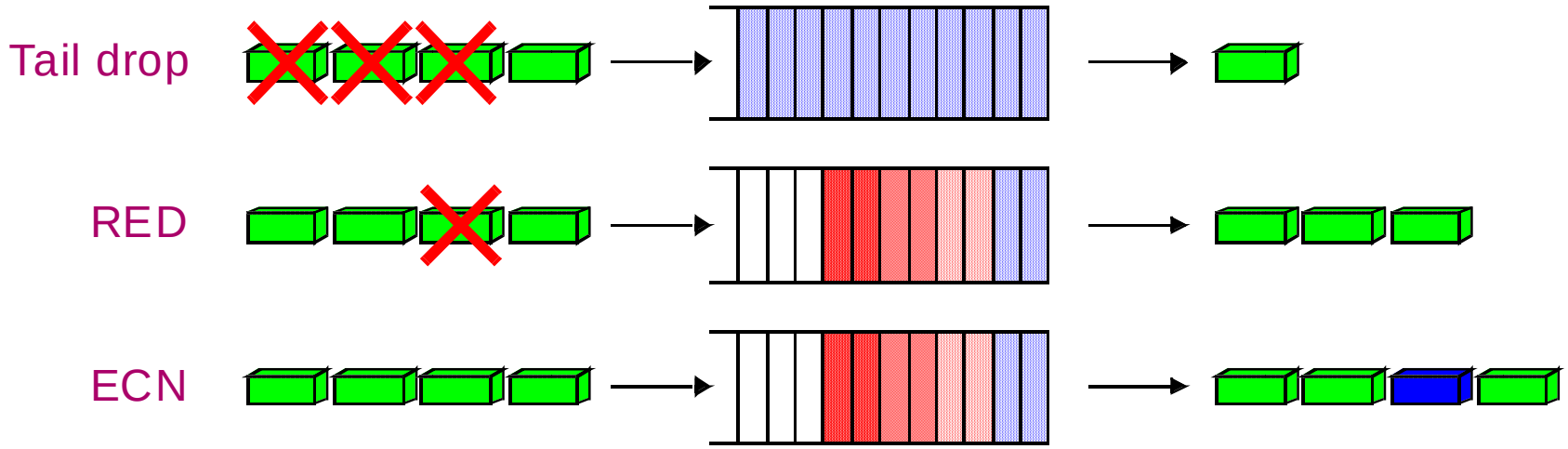
- Current Internet RED
  - restricted to packet drop as congestion indication, but
  - could also only indicate congestion setting congestion experienced (CE) bit in packet header

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

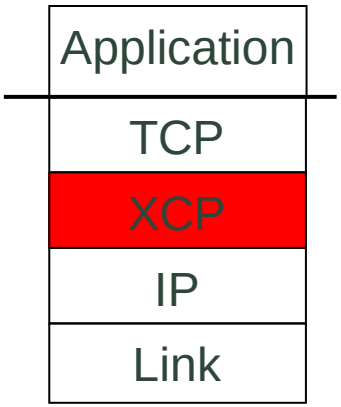
Early Congestion Notification (ECN)



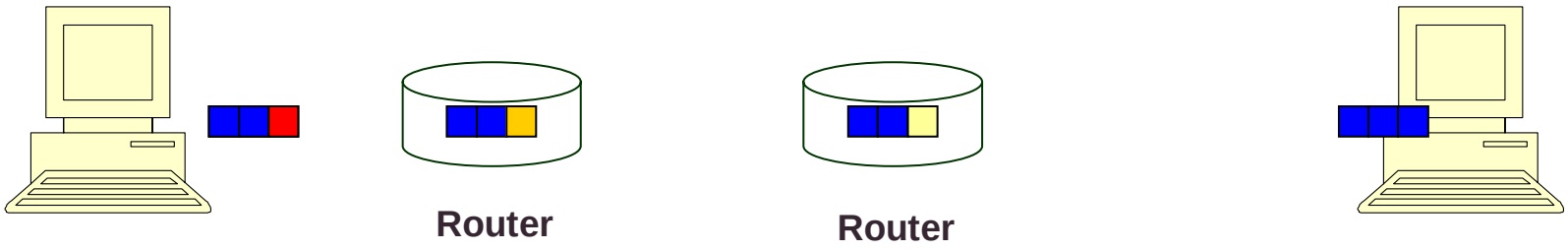
- Effects
  - Congestion is not oscillating - RED & ECN
  - ECN-packets are never lost on uncongested links
  - Receiving an ECN mark means
    - TCP window decrease
    - No packet loss
    - No retransmission

XCP – eXplicit Congestion Notification

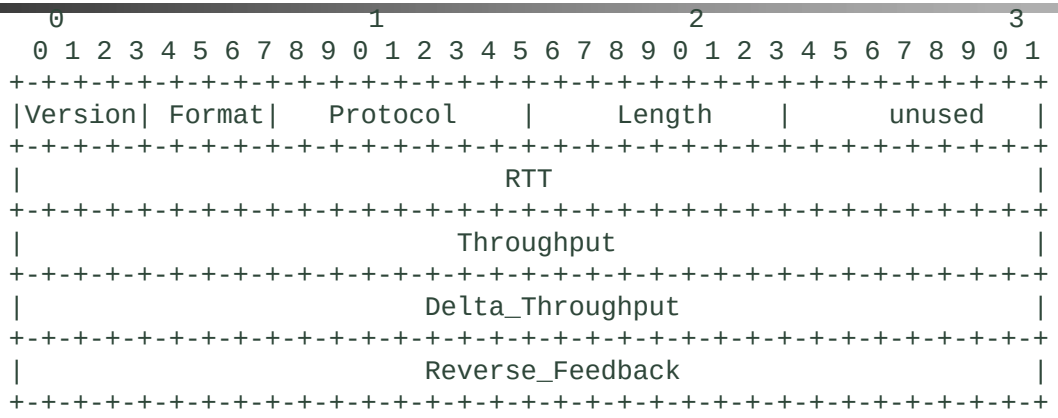
- Protocol for connections with high bandwidth-delay product
- Routers return explicit feedback to the host
  - 20 bytes header between IP and TCP
  - Router does not need to know flows
- Operation
  - Routers adjust sending speed of hosts continously
  - Adjustments are done by changing headers
  - Host use feedback from routers to change their congestion window



The TCP/XCP/IP stack



XCP Header



Version : 1

Format : 1 (Full header), 2 (Minimal header)

Protocol : Next level protocol (6 for TCP)

Length : 20

Unused : 0

RTT : Round Trip Time as measured from the sender in milliseconds

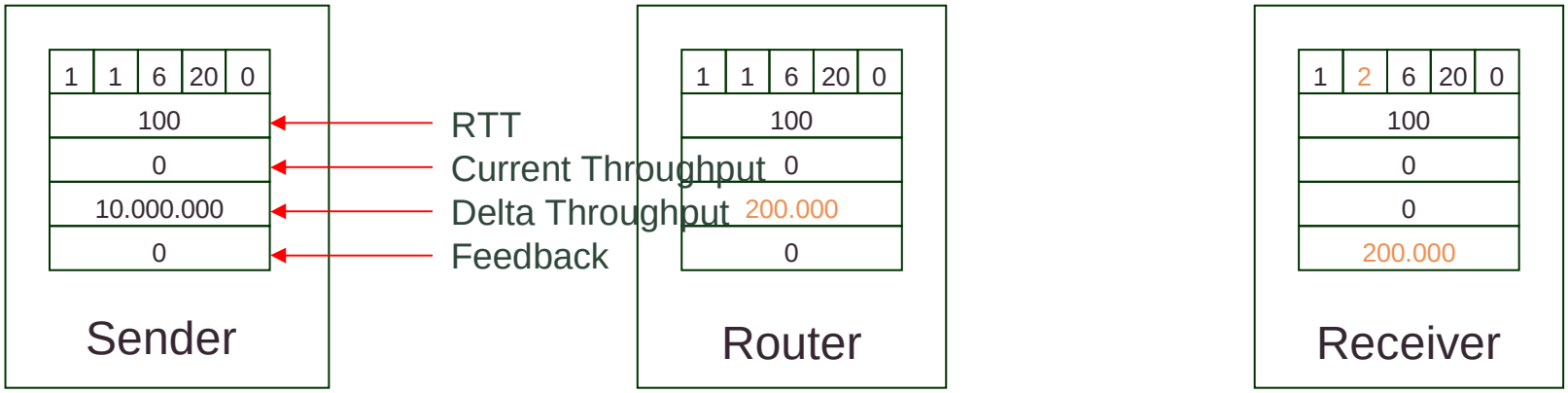
Throughput: The current throughput in bytes/ms in use by the sender

Delta_Throughput: The wanted change in throughput wanted by the sender, measured in bytes/second.

Reverse_Feedback: The contents of the Delta_Throughput field when the message is returned back to the sender from the receiver.

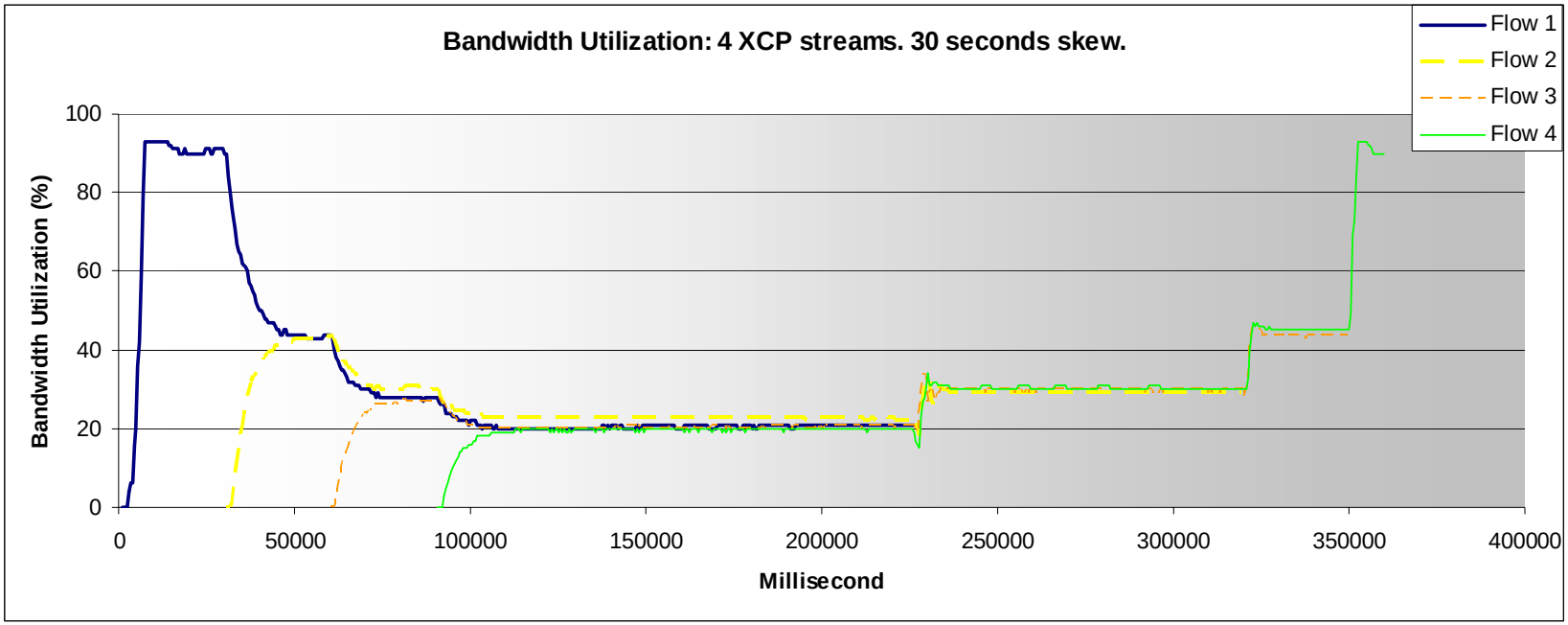
An XCP network (simplified)

Network RTT: 100 ms
Router's capacity: 200.000 B/s (available 200.000 B/s)
Sender's capacity: 10.000.000 B/s (available 10.000.000 B/s)
Sender's current throughput: 0 B/s (or 0 B/ms)



XCP bandwidth distribution

- XCP distributes bandwidth fairly
- Bandwidth does not oscillate

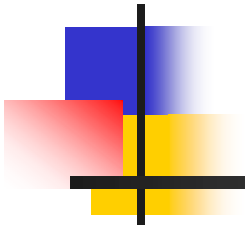


Comparison of Non-QoS Philosophies

Pro UDP	Pro TCP

Using Standard Protocols

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

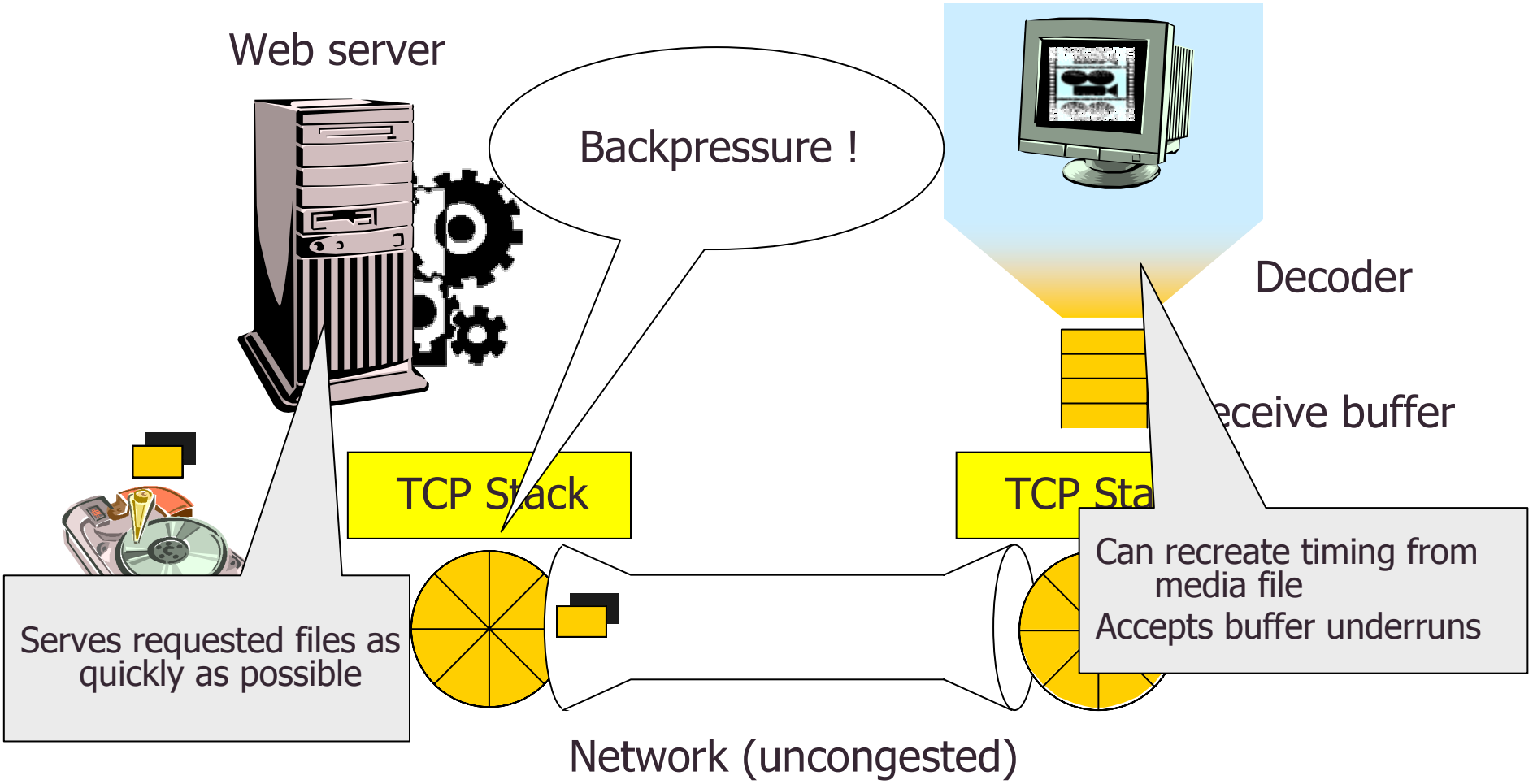


Application Layer Approaches

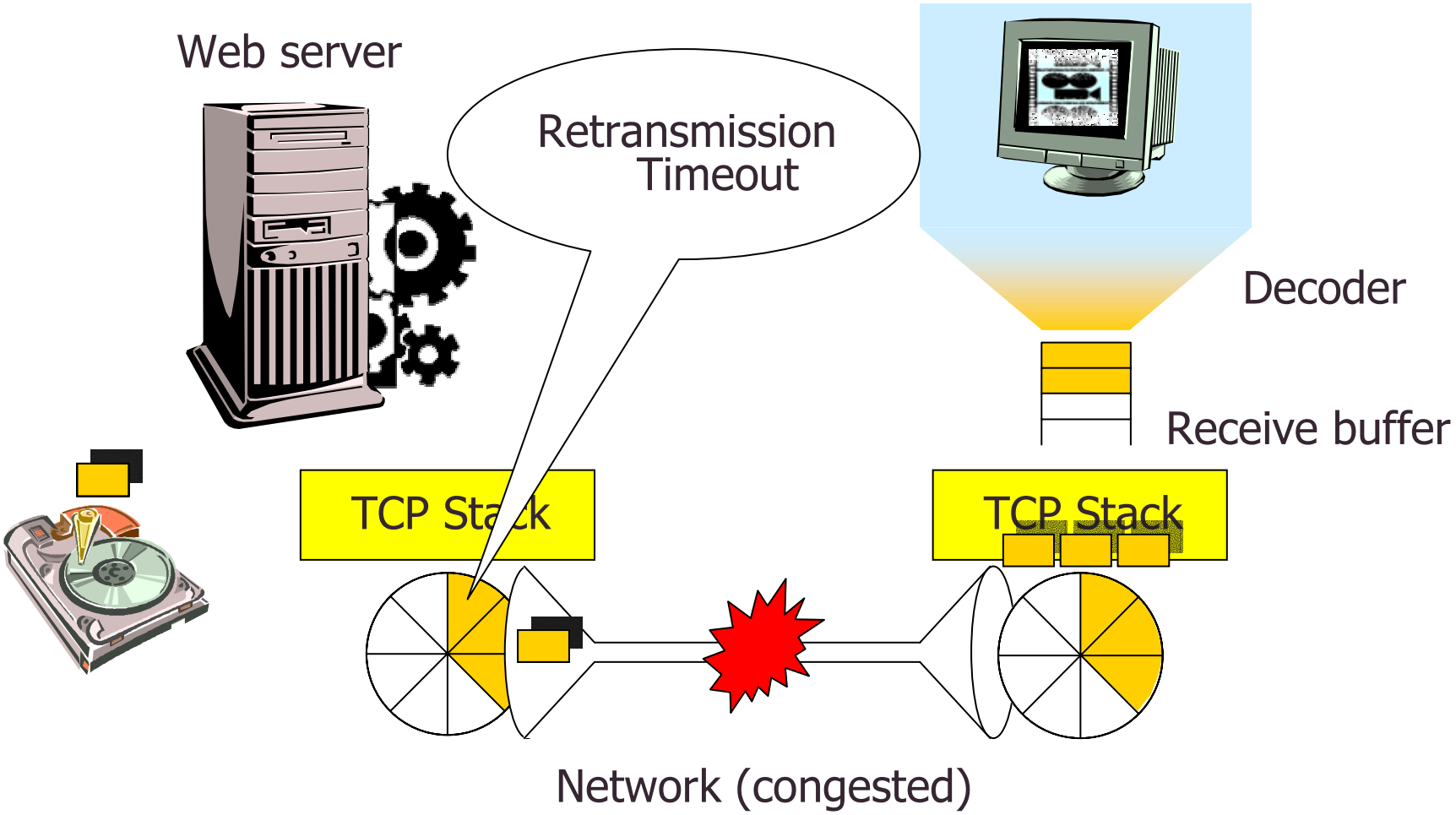
Progressive Download

- 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



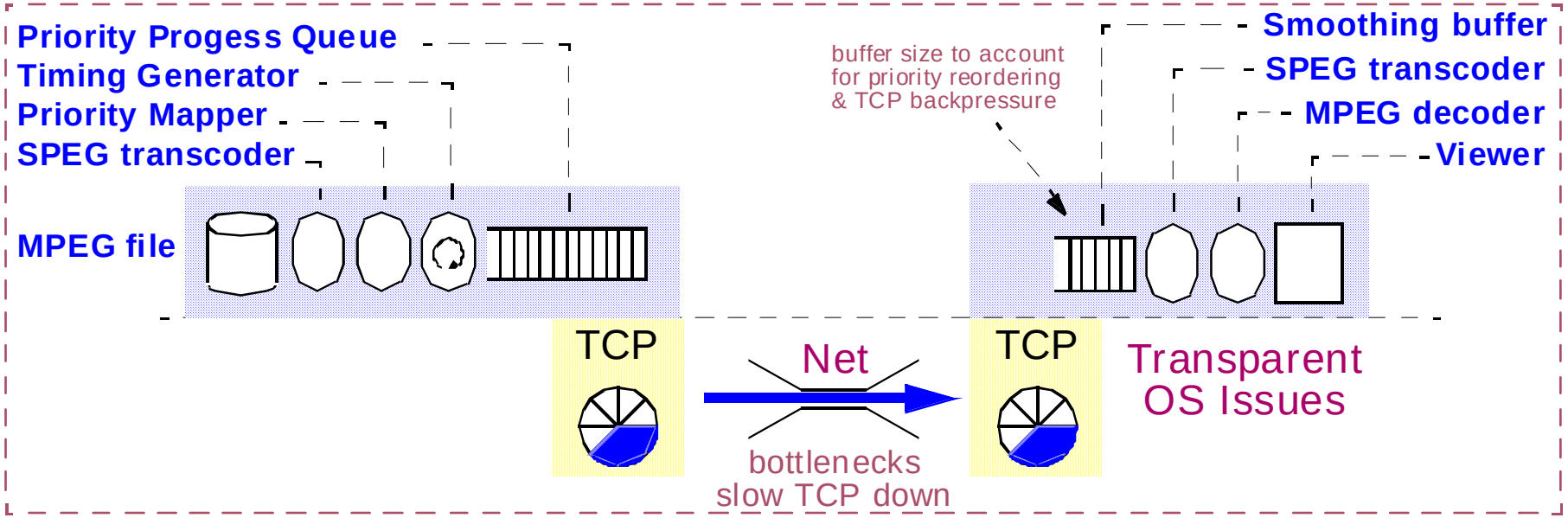
Priority Progress Streaming

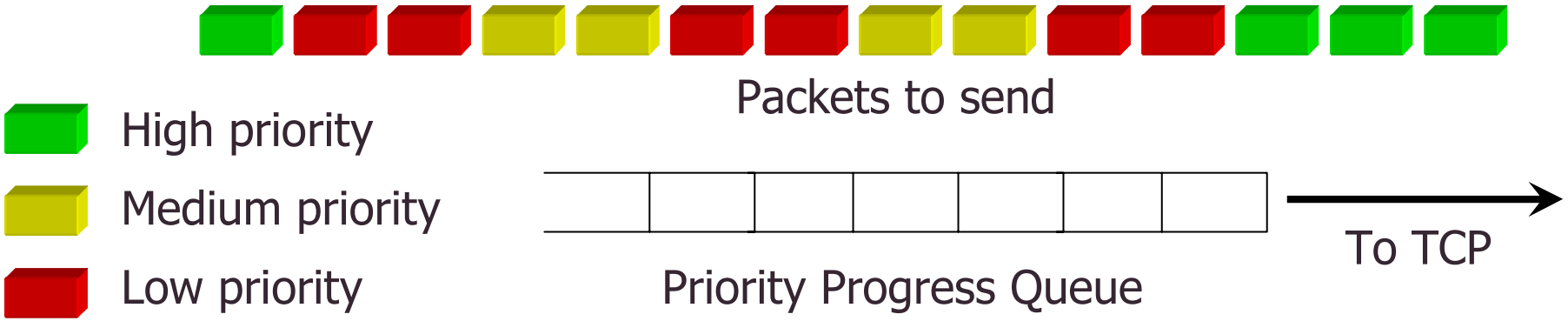
- Unmodified TCP (other transports conceivable)
- Unmodified MPEG-1 video-in (other encoding formats conceivable)

- 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

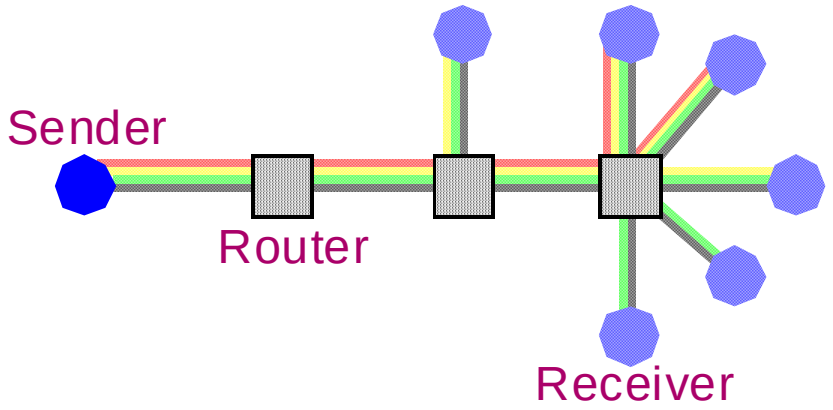
Priority Progress Streaming





Receiver-driven Layered Multicast (RLM)

- Requires
  - IP multicast
  - layered video codec (preferably exponential thickness)

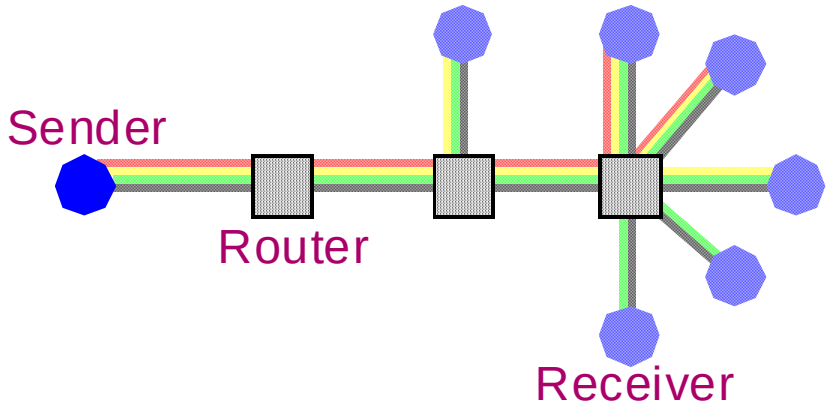


- Operation
  - Each video layer is one IP multicast group
  - Receivers join the base layer and extension layers
  - If they experience loss, they drop layers (leave IP multicast groups)
  - To add layers, they perform "join experiments"

- Advantages
  - Receiver-only decision
  - Congestion affects only sub-tree quality
  - Multicast trees are pruned, sub-trees have only necessary traffic

Receiver-driven Layered Multicast (RLM)

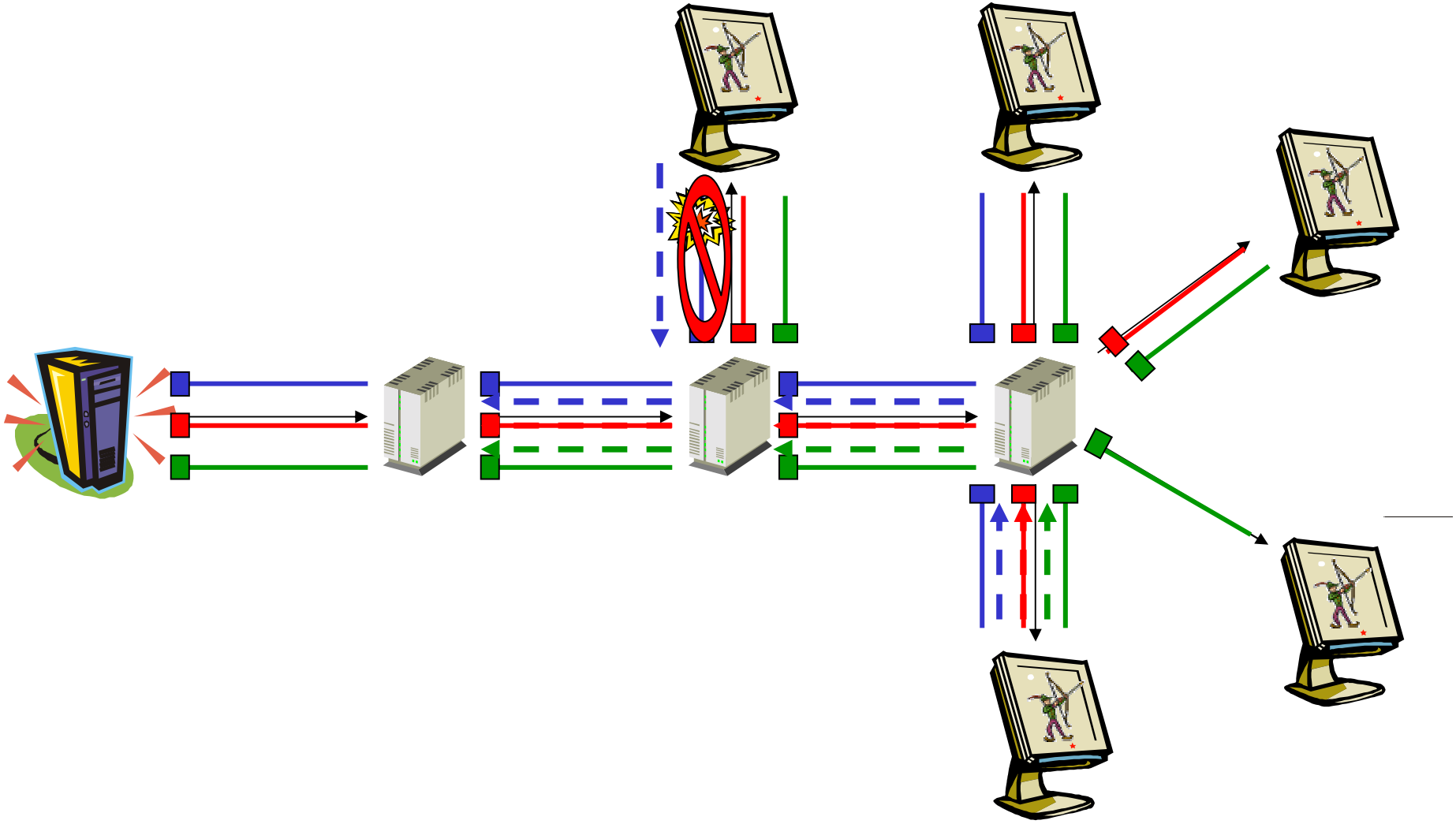
- Requires
  - IP multicast
  - layered video codec (preferably exponential thickness)



- Operation
  - Each video layer is one IP multicast group
  - Receivers join the base layer and extension layers
  - If they experience loss, they drop layers (leave IP multicast groups)
  - To add layers, they perform "join experiments"

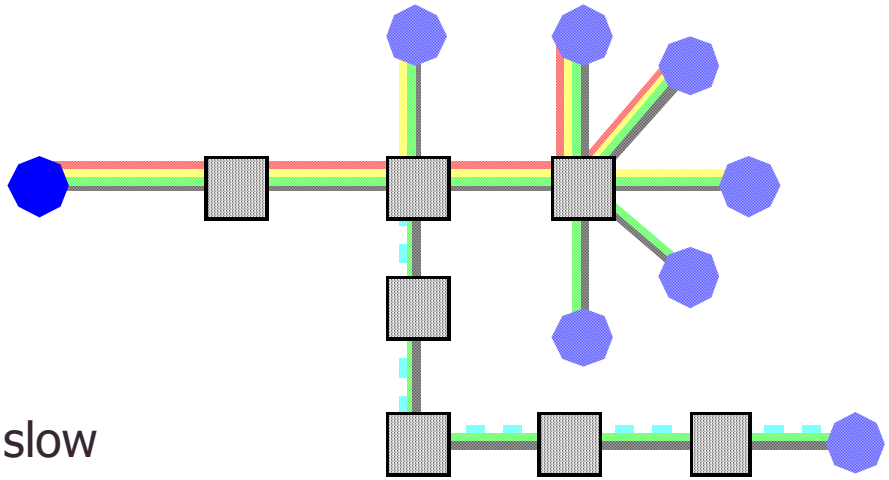
- Advantages
  - Receiver-only decision
  - Congestion affects only sub-tree quality
  - Multicast trees are pruned, sub-trees have only necessary traffic

Receiver-driven Layered Multicast (RLM)

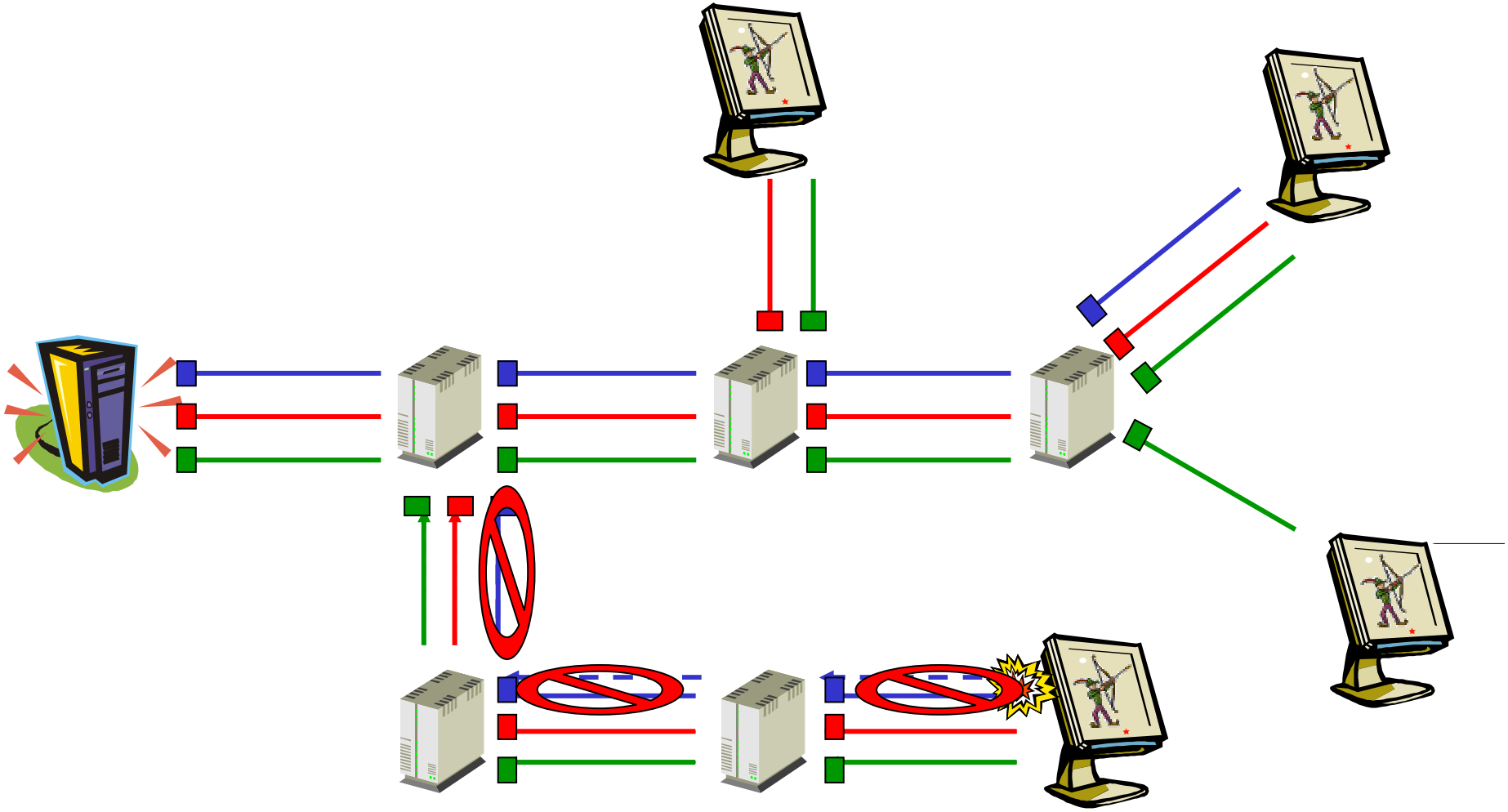


Receiver-driven Layered Multicast (RLM)

- Layer size considerations
  - Adaptations are in steps of 1 layer
  - Big layers
    - Join experiments have huge impact
    - Quality changes are very visible
  - Small layers
    - Many addresses are used
    - Multicast routing effort is high
    - Fair share is hard to achieve (don't release bandwidth quickly enough)
  - Exponential layer sizes
    - Bad enough
    - Best possible mix
- Other problems
  - Synchronization problems
  - PIM-SM removes sub-trees quickly
    - Join and leave operations are very slow



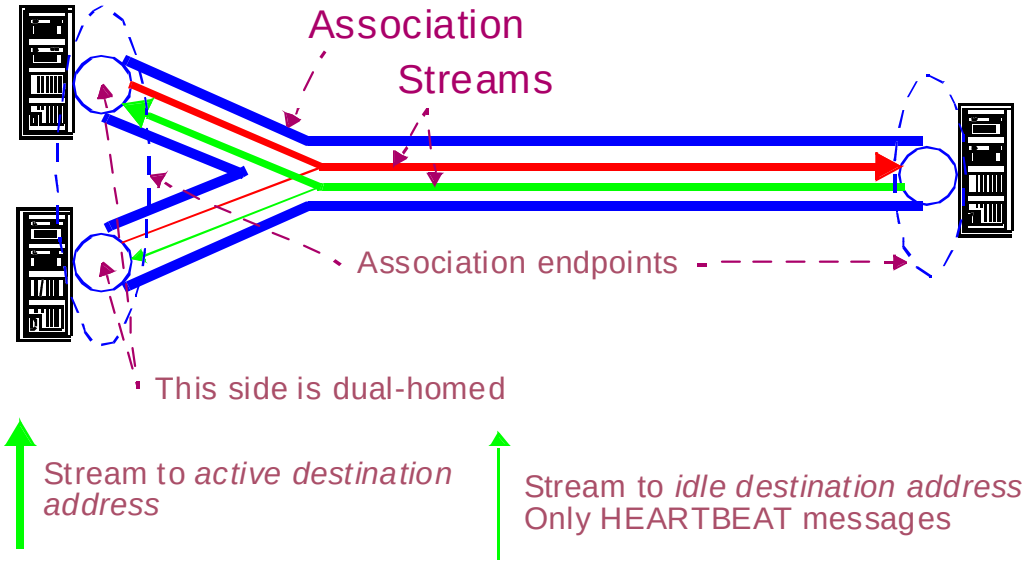
Receiver-driven Layered Multicast (RLM)



Stream Control Transmission Protocol (SCTP)

- Stream Control Transmission Protocol
  - RFC2960, IETF Standards Track &
    SCTP Unreliable Data Mode Extension (draft-ietf-tsvwg-usctp-00.txt)
  - Features
    - Connection-oriented
    - Message-oriented
    - Reliable (with extension also: unreliable, partially reliable)
    - Fully ordered, unordered, partially ordered delivery
    - Multi-homed
- Origin
  - Signaling protocol for SS7 transport over IP networks
  - Players
    - Motorola, Cisco, Siemens, Nortel Networks, Ericsson, Telcordia, UCLA, ACIRI

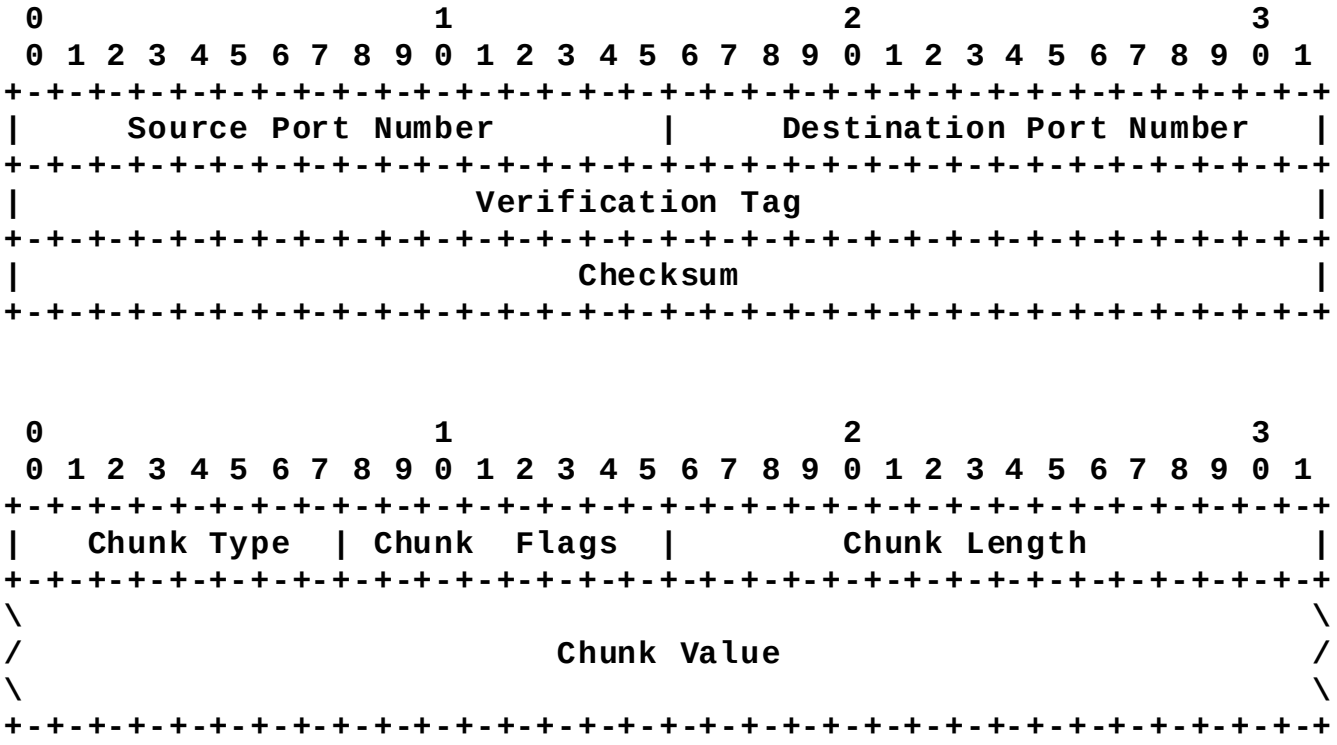
Stream Control Transmission Protocol (SCTP)



- **Connection Management**
  - 4-way handshake for setup
  - Always bi-directional association (no one-way teardown)
  - Up to 216-1 streams per direction and association
  - Cookie exchange against man-in-the-middle attack

Stream Control Transmission Protocol (SCTP)

- SCTP packets
  - Consist of one or more *chunks*
  - Chunks are bundled automatically



Chunks:

- ✓ DATA
- ✓ SACK
- ✓ INIT
- ✓ ABORT
- ✓ SHUTDOWN
- ✓ HEARTBEAT
- ✓ ERROR
- ✓ ECNE
- ✓ CWR
- ✓ & responses

Stream Control Transmission Protocol (SCTP)

- Deadlines
  - Applications can give deadline for packet sending
  - Once sent, delivery is guaranteed (retransmission)
- Reliability
  - Sender and receiver window are computed
    - per stream
    - in bytes
    - changed per stream as in TCP
  - Arrival is reported by SACK chunks
  - SACK chunks
    - are piggybacked
    - contain ranges of packets
  - Retransmission
    - not before 4th missing report
    - always before new packets
- Delivery
  - Sender can specify
    - unordered delivery
    - per packet
- Unreliable transport – V1
  - Proposed extension
  - Max. number of retransmissions
    - specified per stream
    - 0 is legal
  - Ordered and unreliable is possible
- Unreliable transport – V2
  - Proposed extension
  - Sender demands ACKs
    - Receiver must ACK
    - Even if packets were not received

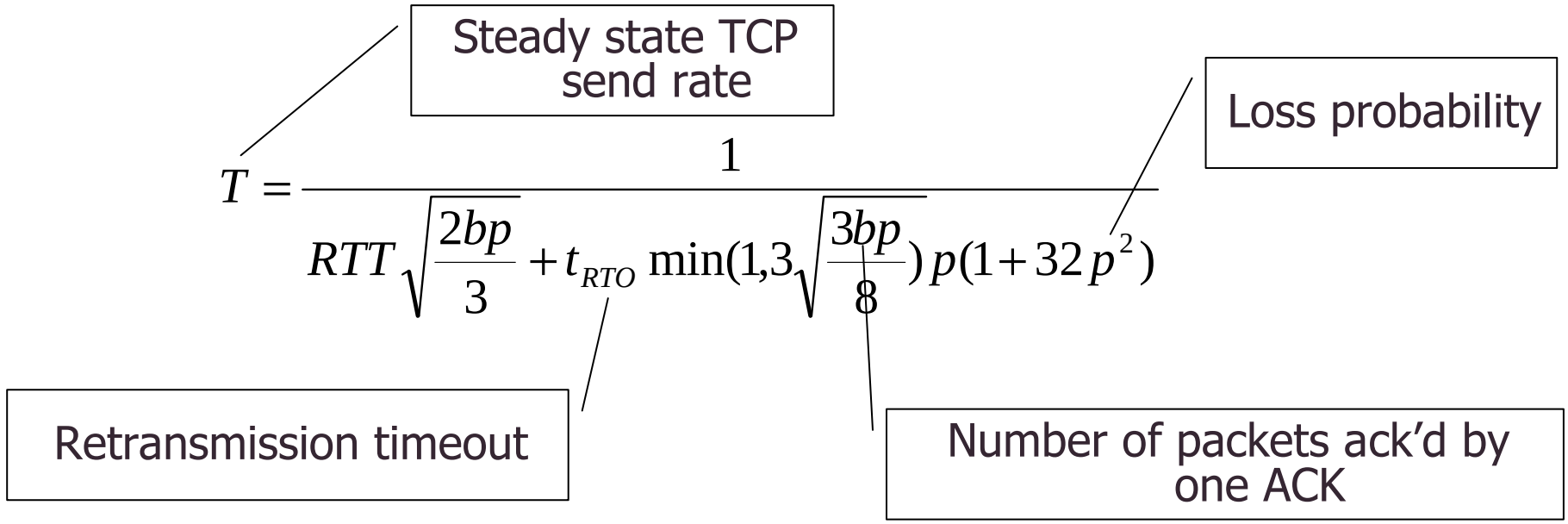
✓ Unreliable, unordered, implicitly TCP-friendly transport protocol

Datagram Congestion Control Protocol (DCCP)

- Datagram Congestion Control Protocol
  - Under development
  - http://www.ietf.org/html.charters/dccp-charter.html
- 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)

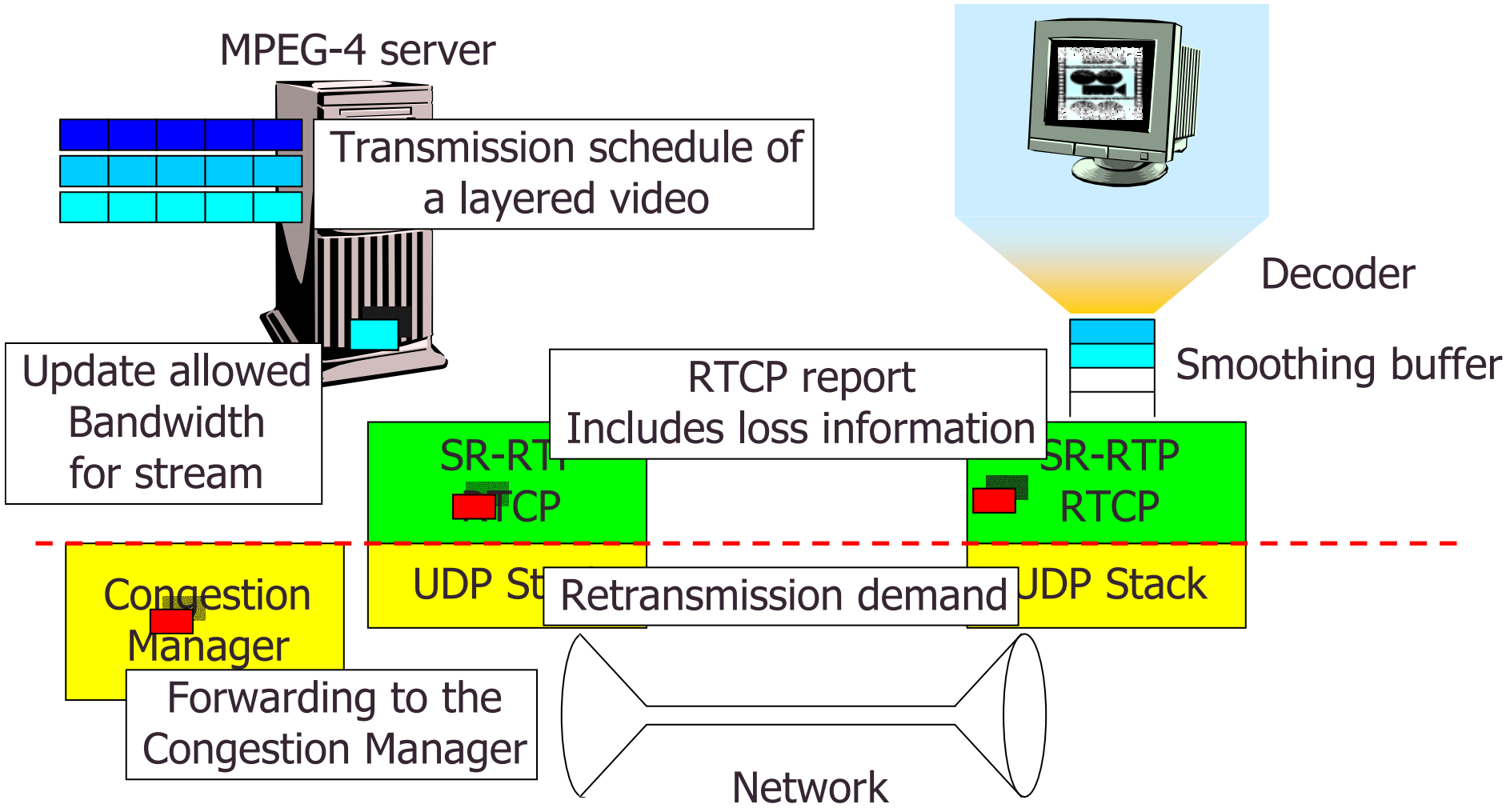
- Congestion control is plugable
  - One proposal is TCP-Friendly Rate Control (TFRC)
    - Equation-based TCP-friendly congestion control
    - Receiver sends rate estimate and loss event history
    - Sender uses models of SACK TCP to compute send rate



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

$$w_{t+RTT} = w_t + \alpha / w_t^k, \alpha > 0$$

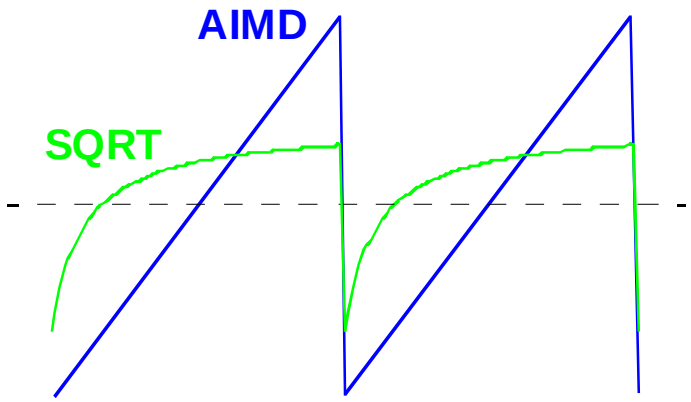
Decrease

$$w_{t+RTT} = \beta \cdot w_t^l, 0 < \beta < 1$$

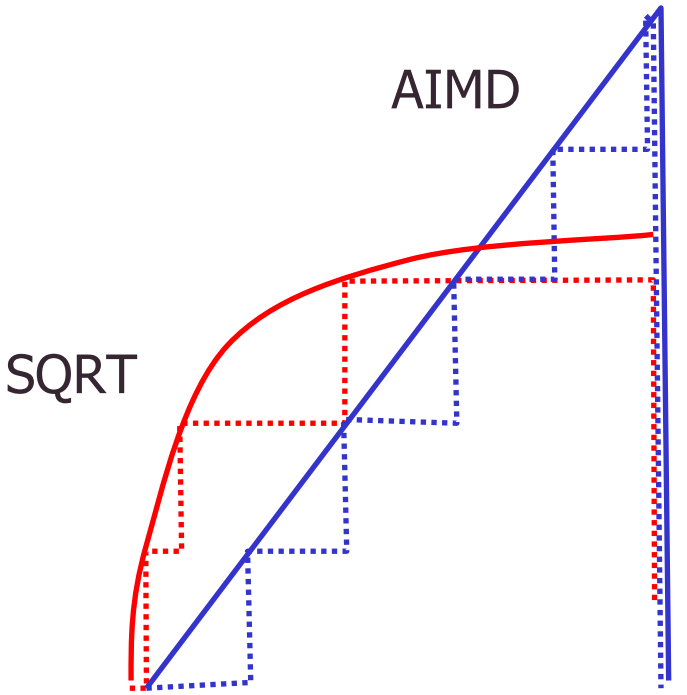
  - Congestion window size w_t depends on losses per RTT
  - TCP's AIMD: $\alpha = 1$, $\beta = .5$, $k = 0$ and $l = 1$
  - $k + l = 1$: binomial congestion control is TCP friendly

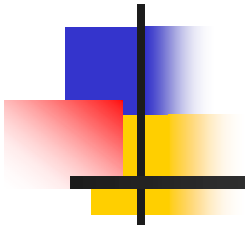
Selective Retransmission-RTP (SR-RTP)

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

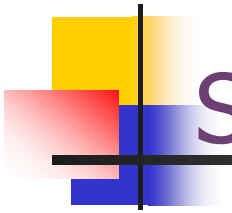


- **Effect of SQRT**
  - Average bandwidth is like TCP's
  - Maximum is lower
  - SQRT covers a step function with less steps





The End: Summary



Summary

- Non-QoS protocols:
  - Transport level protocols
    - UDP
    - TCP
    - ...
  - Application layer protocols
    - RTP
    - Priority progress streaming
    - RLM
    - DCCP
    - ...
  - Signaling protocols
    - RTSP
- Next time, we look at protocols supporting QoS

Some References

1. 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
2. Charles Krasic, Jonathan Walpole: "Priority-Progress Streaming for Quality-Adaptive Multimedia", ACM Multimedia Doctoral Symposium, Ottawa, Canada, October 2001
3. 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