A survey of TCP-friendly congestion control

ryyang

2002.5.27

Outline

- TCP congestion control review
- TCP-friendly congestion control
  - RAP
  - TFRP
  - RCCM
- Reference

Roundtrip time and timeout

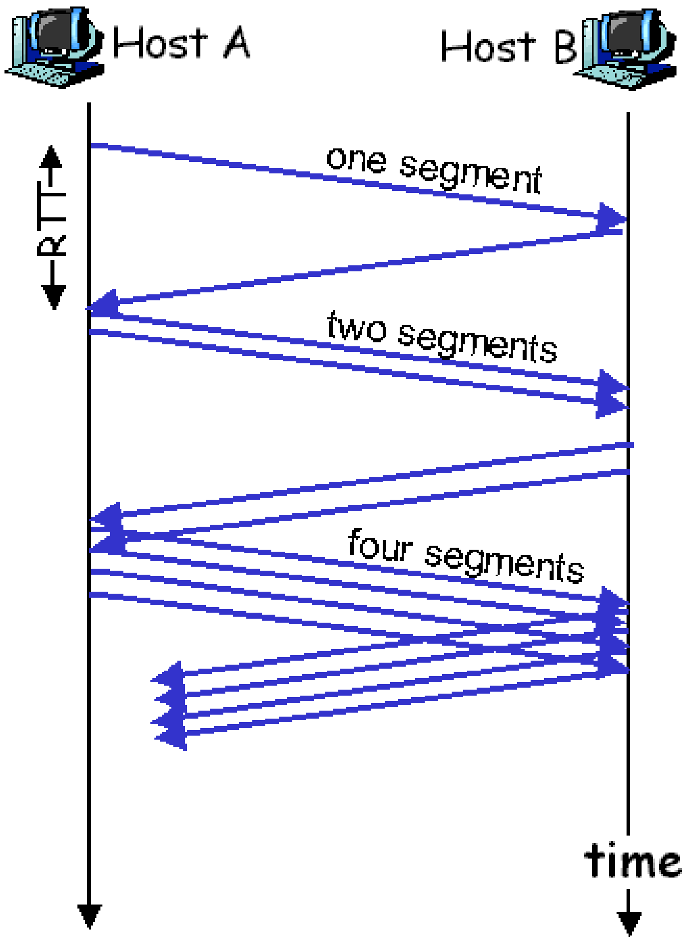
- **SampleRTT**: measured time from segment transmission until ACK receipt
- **EstimatedRTT = (1-x)*EstimatedRTT + x*SampleRTT**
- **Timeout = EstimatedRTT + 4*Deviation**
- **Deviation = (1-x)*Deviation + x*|SampleRTT-EstimatedRTT|**
- Typical value of x: 0.1

TCP Slowstart

Slowstart algorithm

initialize: Congwin = 1
for (each segment ACKed)
    Congwin++
until (loss event OR
    CongWin > threshold

- exponential increase (per RTT) in window size (not so slow!)
- loss event: timeout (Tahoe TCP) and/or or three duplicate ACKs (Reno TCP)

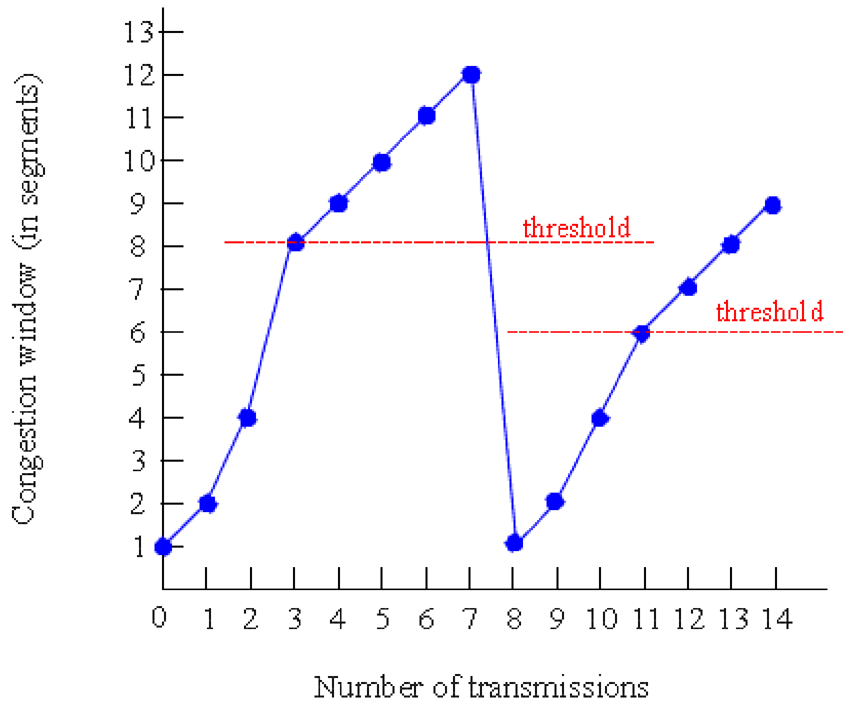


TCP Congestion Avoidance

Congestion avoidance

Until (loss event) {
  every w segments ACKed:
      Congwin++
  }
threshold = Congwin/2
Congwin = 1 (slowstart)

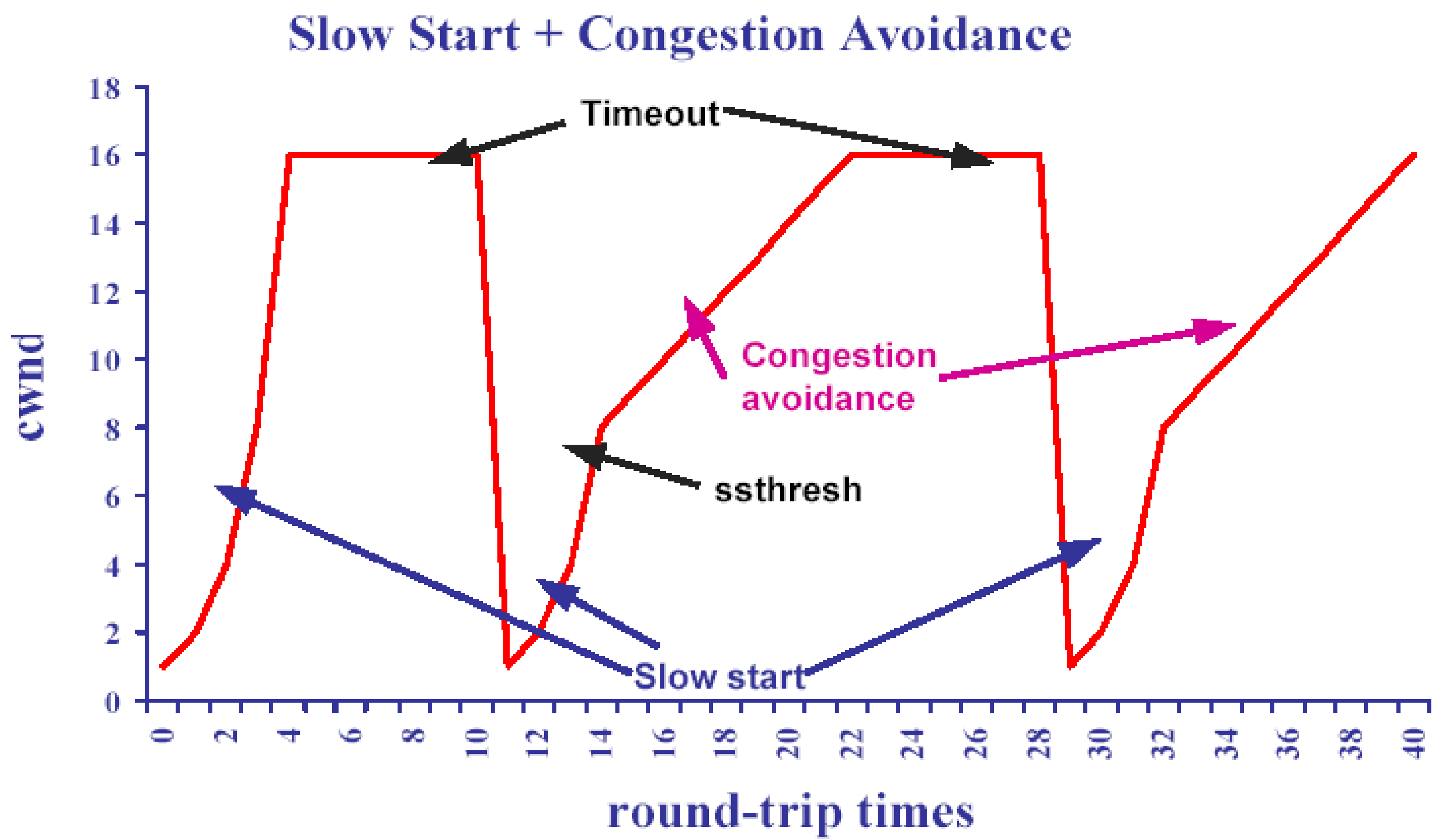
AIMD: additive increase, multiplicative decrease



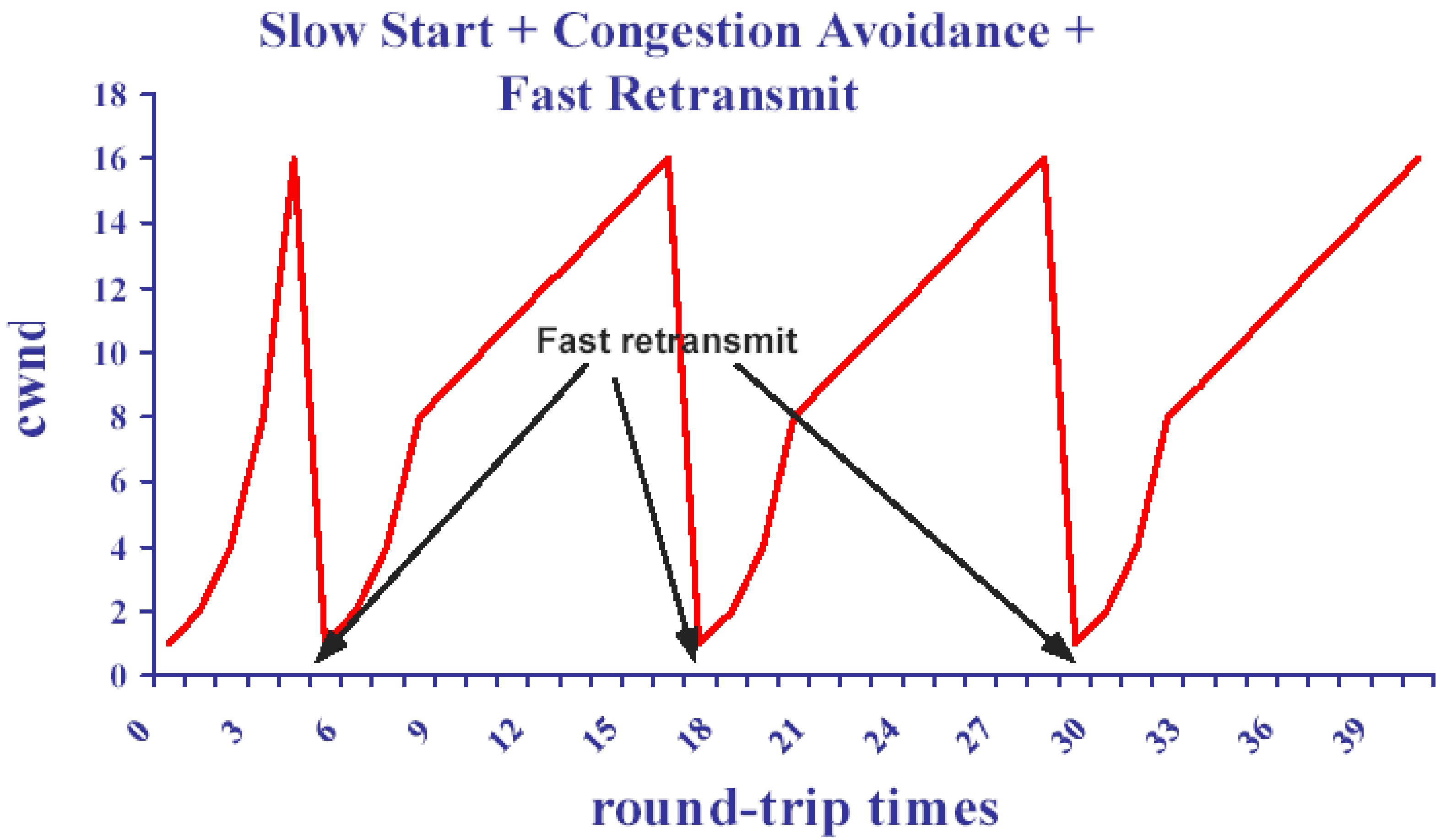
Fast retransmit & recovery

- Fast retransmit
  - Timeouts are slow (1 second is fastest timeout on most TCPs)
  - Use 3 duplicate ACKs to indicate a loss
- Fast recovery
  - If there are still ACKs coming in, then no need for slow start
  - Divide cwnd by 2 after fast retransmit

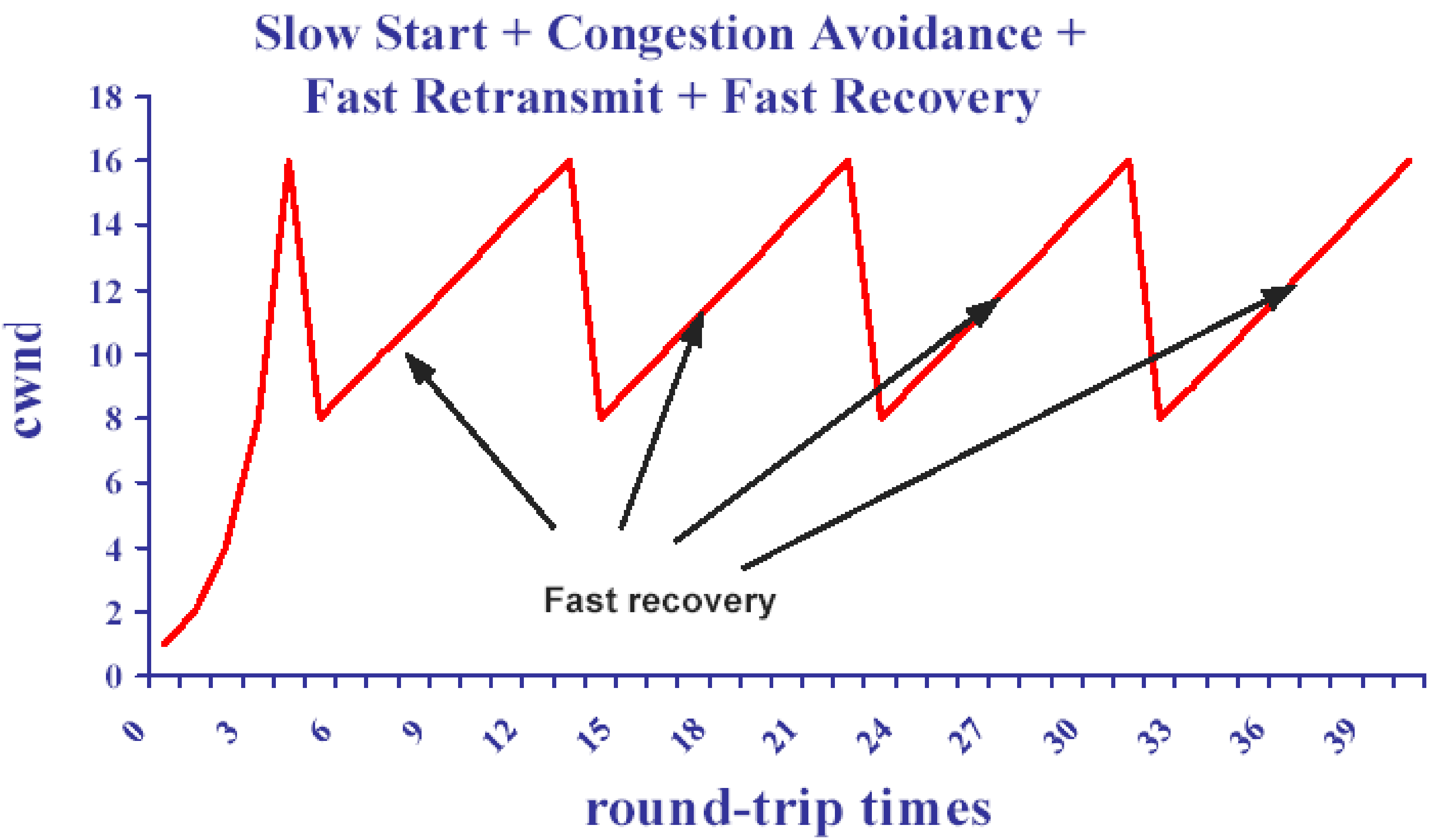
TCP Congestion Control



TCP Congestion Control



TCP Congestion Control



TCP congestion control

- TCP Tahoe
  - Slow Start, Fast Retransmission, Congestion Avoidance
- Reno
  - Tahoe + Fast Recovery
- NewReno
  - Reno + Hoe’s partial ACK change that keeps TCP in FR
- SACK
  - Selective Acknowledgments

Why TCP-friendly congestion control

- Real-time services usually transmit by UDP/RTP, but traditional UDP have not congestion control
- When more and more real-time services are deliver in the network, TCP's services can't obtain the fair bandwidth
- TCP-friendly manner using only a fair share of available bandwidth relative to other majority TCP streams
- Our target rate of real-time service can set by TCP-friendly congestion control to estimate the available bandwidth

RAP congestion control

- RAP: Rate Adaptation Protocol
  - Rate-based and single-rate
  - A simple AIMD scheme for unicast flows
  - Each data packet is acknowledged by receiver
  - The decisions on rate increase or decrease are made once per sRTT
  - Congestion detect by timeout and gap in sequence space

RAP congestion control

- Increase algorithm

$$S_i = \frac{PacketSize}{IPG_i} \qquad IPG_{i+1} = \frac{IPG_i * C}{IPG_i + C}$$

$$\alpha = S_{i+1} - S_i = \frac{PacketSize}{C}$$

- Decrease algorithm

$$S_{i+1} = \beta S_i \qquad IPG_{i+1} = IPG_i / \beta \qquad \beta = 0.5$$

IPG: inter packet gap

S: transmission rate

C: constant time(smoothed RTT)

α :step height

TFRP

- TFRP:TCP-friendly transport protocol
  - Do not address how to measure RTT (RTCP in other paper)
  - Using TCP’s steady-state throughput model to adjust rate
  - Multicast and multirate
  - Rate-based congestion control

TFRP

$$S = W * \frac{MTU}{RTT}$$

$$BW = 0.75 * W * \frac{MTU}{RTT}$$

$$Loss = \frac{1}{(\frac{W}{2} + (\frac{W}{2} + 1) + \ldots\ldots + W)}$$

$$Loss \sim \frac{8}{3 * W^2}$$

$$W \sim \sqrt{\frac{8}{3 * Loss}}$$

S:average sending rate

W:when packet dropped, the TCP connection's window size

BW:TCP connection receives average bandwidth

Loss:packet loss rate

TFRP

$$BW = 1.22 * \frac{MTU}{RTT * \sqrt{Loss}}$$

- When loss rate up to 5%, as the loss rate increases, it begins to overestimate the bandwidth received by a TCP connection
- It does not apply at all for loss rate of 15% or more. When steady-state model, W at least 4 packet when a packet drop. Thus this model does not apply for W<4, which corresponds to Loss>0.16

RCCM

- RCCM: Receiver-based congestion control mechanism
  - Some manner focus on TCP-friendliness but might result in inefficient usage resource
  - RTT estimate by both sender and receiver
  - RTSP and feedback by RTP/RTCP

RCCM

- $RTT_{sample} = T_{TR} - T_{FS} - T_{FE}$
- $RTT_{estimate} = \alpha * RTT_{estimate} + (1-\alpha) * RTT_{sample}$
- T_{TR} : The feedback send time at the receiver
- T_{FS} : The feedback response receive time at the receiver
- T_{FE} : The elapsed time at the sender

RCCM

- $Rate_{estimate} = Rate_{i-1} * (Cong_{degree}?(1+C_{CD}):1:(1-C_{CD}))$
- If (no packet loss)
  - $Rate_{AI} = Rate_{i-1} + PacketSize_{average}/RTT_{estimate}$
  - $Rate_{i} = max(min(Rate_{estimate}, Rate_{AI} * (1+C_{AI}), Rate_{max}), Rate_{AI} * (1-C_{AI}))$
- If (packet loss)
  - $Rate_{MD} = \beta * Rate_{i-1}$
  - $Rate_{i} = min(max(Rate_{estimate}, Rate_{MD} * (1+C_{MD}), Rate_{min}), Rate_{MD} * (1-C_{MD}))$

RCCM

- $T_j = T_{j-1} + PacketSize_j / Rate_i$
- $TargetRate_i = Rate_i + C_{slope} \times 0.5 \times (Rate_i - Rate_{i-1}) + C_{buffer} \times (Buffer_{TxRef} - Buffer_{TxLev})$
- $TargetRate_i = \mu \times TargetRate_i + (1-\mu) \times TargetRate_{i-1}$
- C_{slope} set 0.5 when increase, set 1 when decrease
- C_{buffer} assigned to 0.7

RCCM

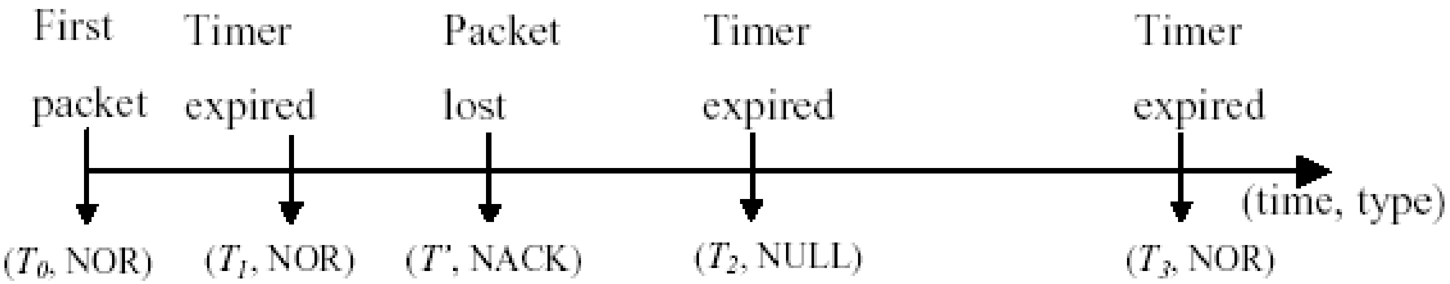
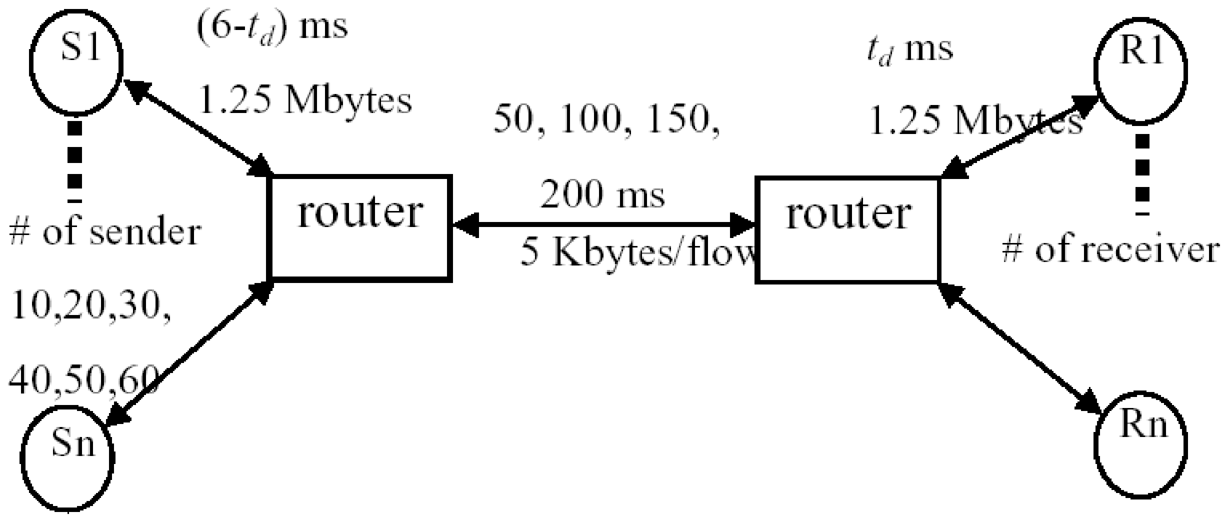


Figure 2: The feedback mechanism adopted in the receiver.

- T_2-T'=T_{wait} set adaptively to handle out of order packets
- $N_{decrease}$ use for multiple packet loss

RCCM

- 6 sets of different flows numbers and 4 delay situation in the bottleneck link
- Each simulation length set to 150s



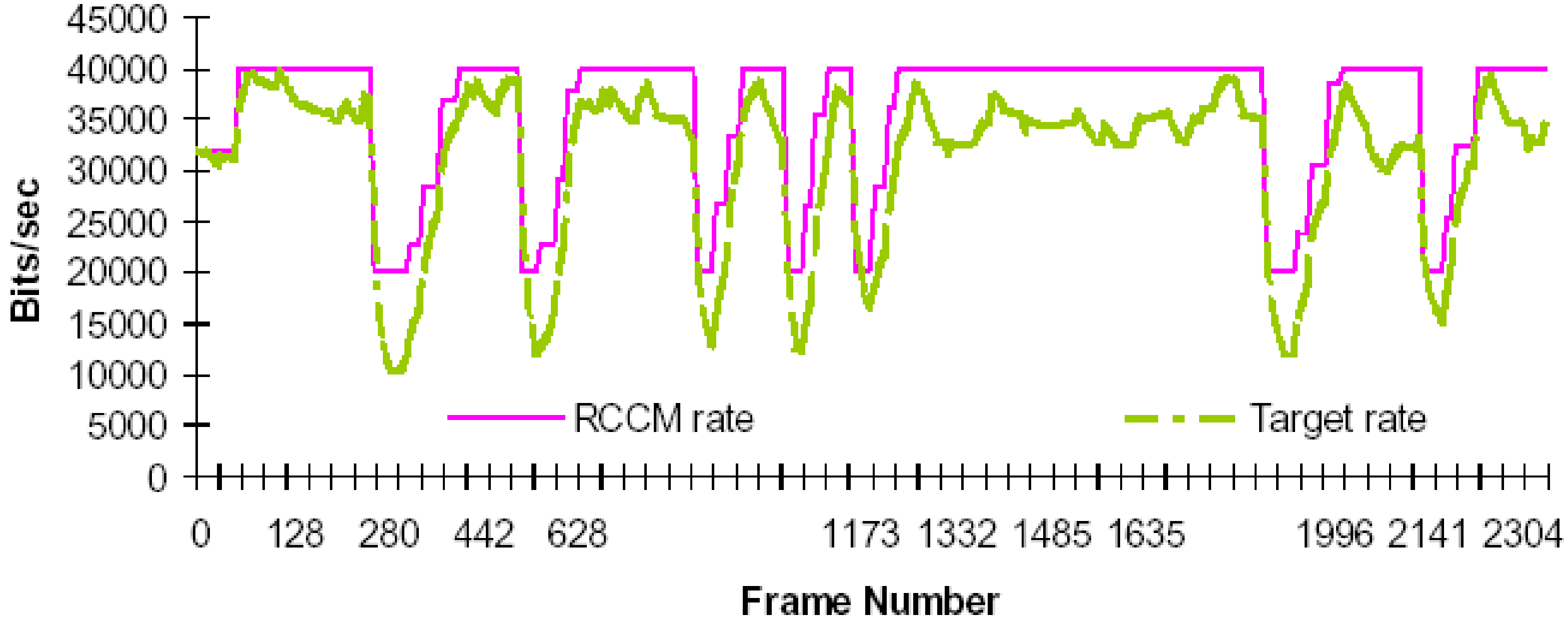
RCCM performance

C_{CD} / C_{AI} / C_{MD} (%)	$N_{decrease}$	Tahoe		Reno		NewReno		SACK	
		Avg	SD	Avg	SD	Avg	SD	Avg	SD
* / 0 / 0	1	0.91	0.30	2.03	1.16	0.90	0.31	0.74	0.24
	2	0.67	0.24	1.15	0.59	0.68	0.23	0.58	0.20
	3	0.64	0.21	1.07	0.53	0.66	0.22	0.56	0.19
	4	0.65	0.21	1.03	0.52	0.65	0.21	0.73	0.22

C_{CD} / C_{AI} / C_{MD} (%)	$N_{decrease}$	Tahoe		Reno		NewReno		SACK	
		Avg	SD	Avg	SD	Avg	SD	Avg	SD
5 / 2 / 2	1	0.99	0.17	2.94	0.76	1.03	0.22	0.75	0.14
	2	0.66	0.12	1.82	0.49	0.68	0.12	0.54	0.11
10 / 2 / 2	1	1.20	0.18	3.11	0.60	1.27	0.27	0.88	0.15
	2	0.76	0.12	1.63	0.39	0.75	0.13	0.62	0.11
5 /5 /5	1	0.79	0.16	2.03	0.75	0.85	0.17	0.66	0.14
	2	0.62	0.13	1.15	0.46	0.67	0.13	0.53	0.11
10 / 5 / 5	1	1.41	0.23	3.53	0.53	1.75	0.38	1.03	0.17
	2	0.72	0.12	1.70	0.39	0.81	0.15	0.57	0.08

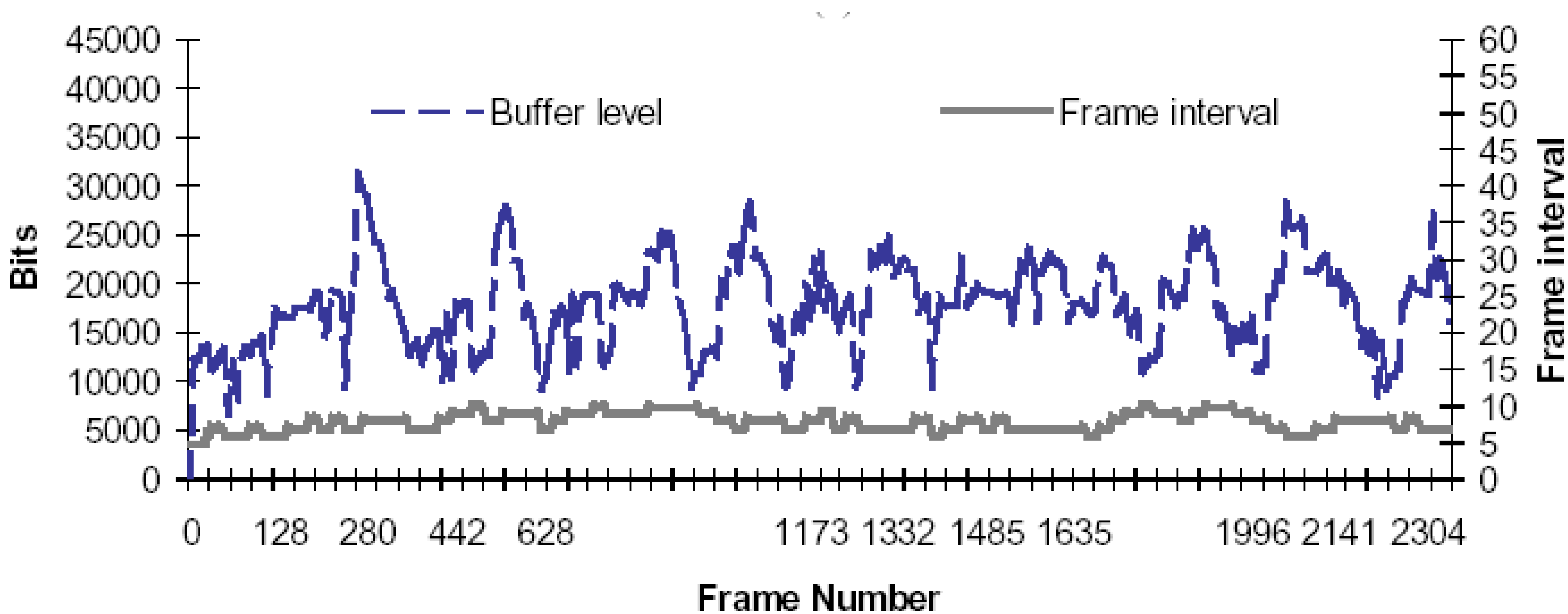
RCCM performance

- Max rate=40kbps, min rate=20kbps, initial rate=32kbps
- Foreman qcif are used



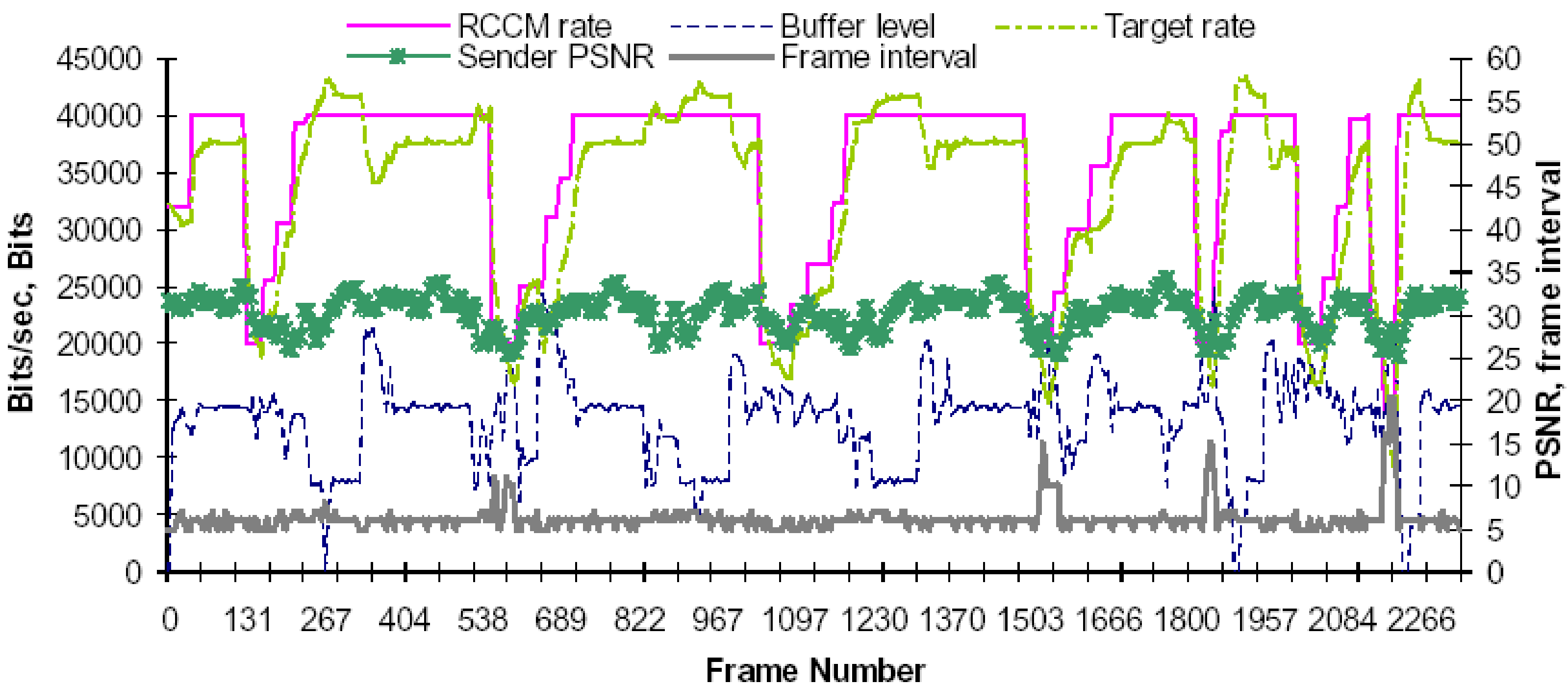
RCCM performance

- Initial frame rate = 5f/s



RCCM performance

- Constant frame rate



RCCM performance

	RCCM (Kbps)		PSNR (dB)	
	Average	Standard Deviation	Average	Standard Deviation
Foreman Variable Frame-rate	36.12	6.96	30.92	2.01
Foreman Constant Frame-rate	35.79	7.70	30.20	2.04

- The RCCM module with the available bandwidth estimator provides the bandwidth consumption level to minimize the packet loss in a TCP-friendly manner.
- The real-time variable frame-rate controller at the encoder responds by tailoring the stream to fit the available bandwidth and meet the end-to-end delay requirement.

Reference

- "Computer networking", James F.Kurose, Keith W. Ross
- "A survey on TCP-Friendly congestion control", Joerg Widmer, Robert Denda, and Martin Mauve, Praktische Informatik IV, University of Mannheim, Germany
- TFRP
  - TCP-Friendly unicast Rate-Based flow control, Jamshid Mahdavi & Sally Floyd, January 1997 http://www.psc.edu/networking/papers/tcp_friendly.html
  - Real-Time internet video using error resilience scalable compression and TCP-Friendly transport protocol, Wai-tian Tan & Avideh Zakhor, IEEE transactions on multimedia,1999

Reference

- RAP
  - RAP:An End-to-End Rate-based congestion control mechanism for realtime streams in the Internet, Reza Rejaie, Mark Handley, Deborah Estrin, Proc. IEEE INFORCOM, Mar. 1999
- RCCM
  - An end-to-end congestion control mechanism for internet video, Young-Gook Kim, Yon Jun Chung, Jong Won Kim, C.-C. Jay Huo,
  - TCP-friendly internet video streaming employing variable frame-rate encoding and interpolation, IEEE transactions on circuits and systems for video technology, John-Won Kim, Young-Gook Kim, Hwang-Jun Song, Tien-Ying Kuo, Yon Jun Chung, and C.-C. Jay Kuo