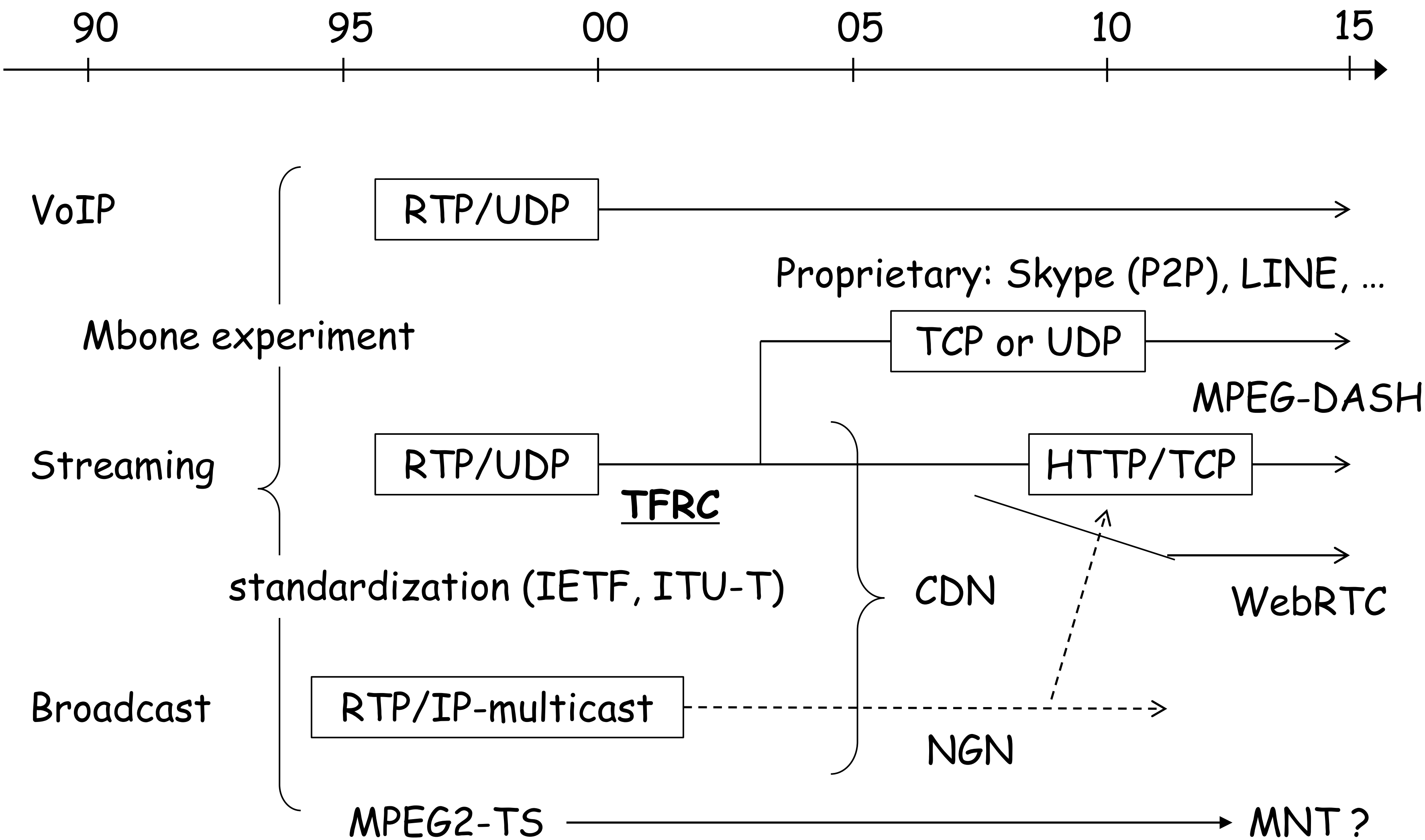
画像情報特論 (5)
Advanced Image Information (5)

TFRC and its Variants

情報理工・情報通信専攻　甲藤二郎

E-Mail: katto@waseda.jp

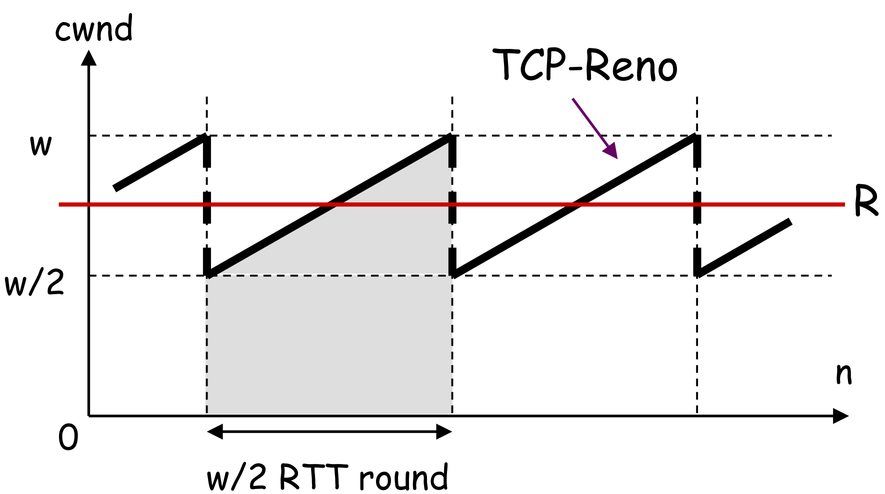
Protocol Transition



TCP Equations
→ TFRC

TCP Modeling

- TCP-Reno Equivalent Rate



w: cwnd when packet is lost

p: PLR

RTT: round trip time

R: equivalent rate

b: # of delayed ACKs

PS: packet size

with timeout consideration

$$\begin{cases} p = \dfrac{8}{3w^2} \\ R = \dfrac{PS}{RTT} \cdot \sqrt{\dfrac{3}{2p}} \end{cases}$$

$$R_{loss} = \frac{PS}{\underbrace{RTT\sqrt{\frac{2bp}{3}}}_{\text{packet loss}} + \underbrace{t_{RTO,loss} \cdot 3\sqrt{\frac{3bp}{8}} \cdot p(1+32p^2)}_{\text{time out}}}$$

J.Padhye et al: "Modeling TCP Throughput: A Simple Model and its Empirical Validation", ACM SIGCOMM 1998.

TCP Westwood

FSE: Fair Share Estimates

- Duplicate ACKs

$$ssthresh = FSE * RTT_{\min}$$
$$if\ (cwnd > ssthreh)\ cwnd = ssthresh$$

- Timeout

in TCP-Reno case

$$ssthresh = cwnd / 2$$

$$ssthresh = FSE * RTT_{\min}$$
$$cwnd = 1$$

- multiple versions according to FSE estimation methods

C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM MOBICOM 2001.

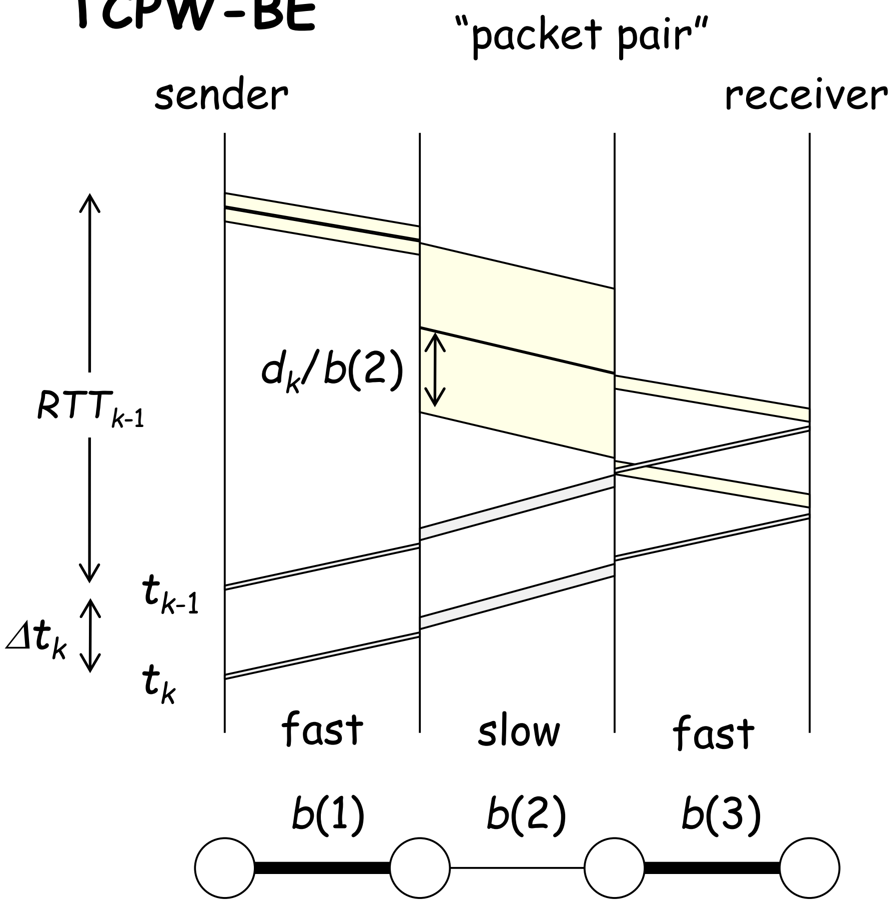
Performance tools

- Active probing (high accuracy)
  - uses probe packets
    - pathchar, pchar, …
    - iperf, netperf
- Inline measurement (low accuracy)
  - uses application data packets
    - TCP-Westwood

https://www.caida.org/tools/taxonomy/perftaxonomy.xml

Bandwidth Share Estimation

TCPW-BE



Bandwidth share: $b = \min_j \left(b(j)\right)$

t_k: ack arrival time of the k-th packet

d_k: size of the k-th packet

$$\Delta t_k = t_k - t_{k-1} \approx \frac{d_k}{b}$$

$$b_k \approx \frac{d_k}{\Delta t_k}$$

moving average: $\hat{b}_k \rightarrow FSE$

C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM MOBICOM 2001.

Rate Estimation

(reference) TCP-Vegas

$$diff = \left(\frac{cwnd}{RTT_{\min}} - \frac{cwnd}{RTT} \right) \cdot RTT_{\min}$$

expect rate

actual rate

TCPW-RE:

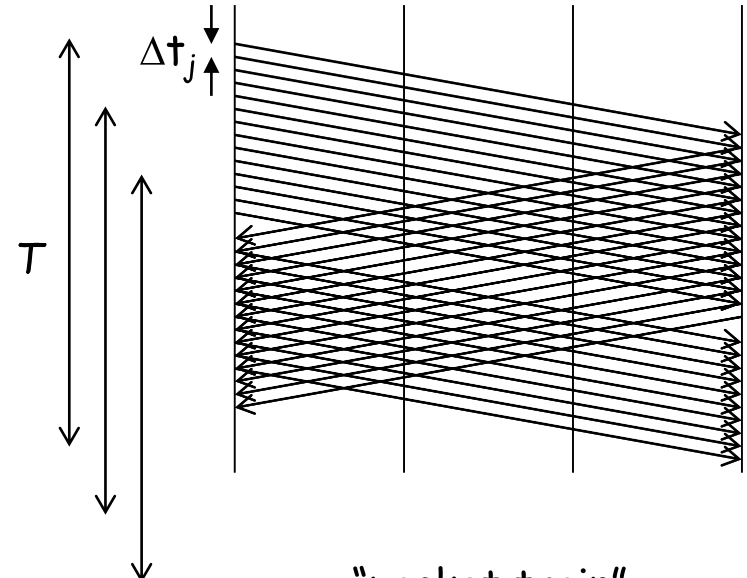
$$RE_k = \frac{\sum d_j}{T}$$

cwnd $\Rightarrow$ S=Σd_j

RTT$\Rightarrow$ T= $\Sigma \Delta t_j$

$$\hat{R}E_k \approx \frac{\sum d_j}{\sum \Delta t_j}$$

moving average: $\hat{R}E_k \rightarrow FSE$



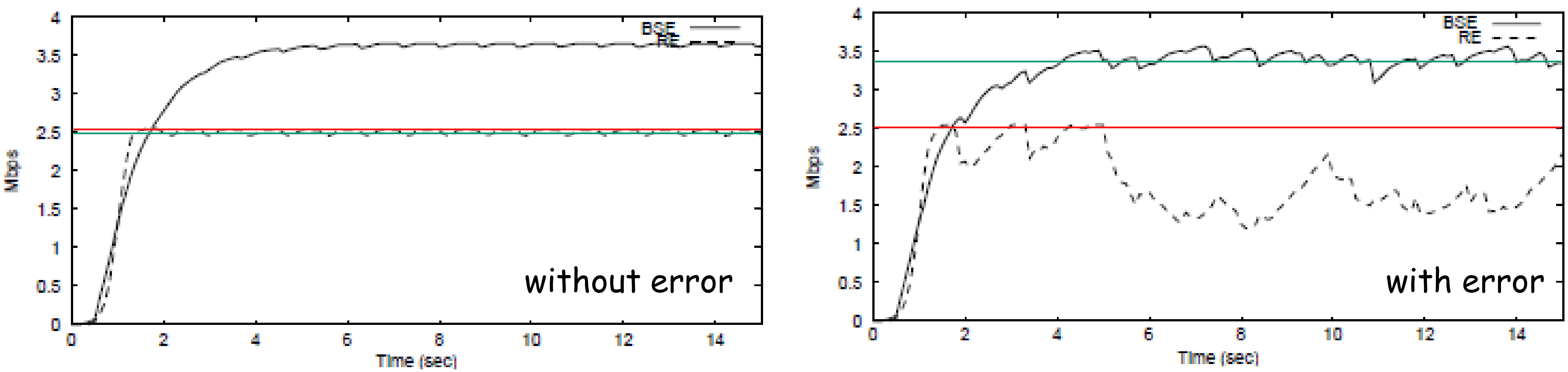
"packet train"

$T = n \cdot RTT$ (e.g. n=4)

M.Gerla et al: "TCP westwood with adaptive bandwidth estimation to improve …", Comp. & Comm., 2004.

Comparison of BSE and RE

solid: BSE, dashed: RE, red: fair share, green: capacity



- BSE tends to overestimate (due to burstiness)
- RE tends to underestimate when losses occur

M.Gerla et al: "TCP westwood with adaptive bandwidth estimation to improve …", Comp. & Comm., 2004.

Adaptive Bandwidth Share Estimation

- BSE: overestimation, RE: underestimation
- difference lies in sampling period T

- large T when congested (BSE), small T when not congested (RE)

actual rate

TCPW-ABSE:

$$T_k = \max\left(T_{\min},\ RTT \cdot \left(1 - \frac{\hat{Th} \cdot RTT_{\min}}{cwnd}\right)\right)$$

T_{min} : ACK arrival interval

$\hat{Th} \cdot RTT_{\min} < cwnd$ ⇒ congested ⇒ larger T_k

多数のパケットを送っても実レートが上がらない

M.Gerla et al: “TCP westwood with adaptive bandwidth estimation to improve …”, Comp. & Comm., 2004.

TFRC and its variants

TFRC (RFC 3448)

- TCP-Friendly Rate Control

$$R_{TFRC} = \frac{PS}{RTT\sqrt{\frac{2bp}{3}} + 4 \cdot RTT \cdot 3\sqrt{\frac{3bp}{8}} \cdot p(1+32p^2)}$$

b=1: delayed ACK (recommended)

t_{RTO} TCP retransmission timeout

  - calculate TCP-Reno equivalent rate by observing RTT and PLR p
  - assume real-time applications (voice, video, or game) by RTP/UDP or DCCP

M.Handley et al: "TCP Friendly Rate Control (TFRC): Protocol Specification", IETF RFC 3448, 2003.

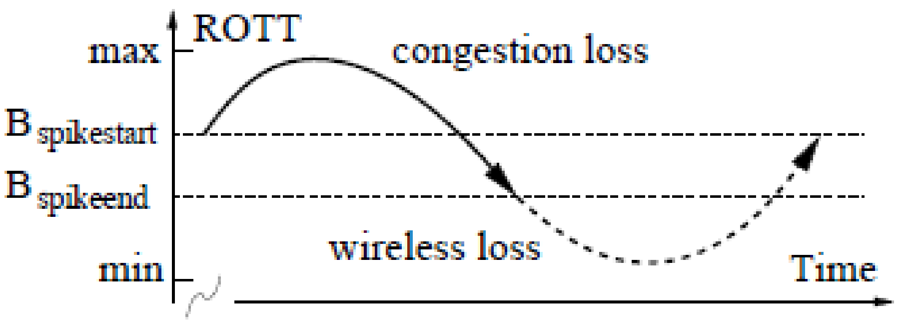
Disadvantage of TFRC

- inherits TCP-Reno's weak points
  - causes vacant capacity when buffer size is smaller than BDP (due to window halving)
  - causes unnecessary window decrease when PLS is high (e.g. wireless networks) $\Rightarrow$ LDA

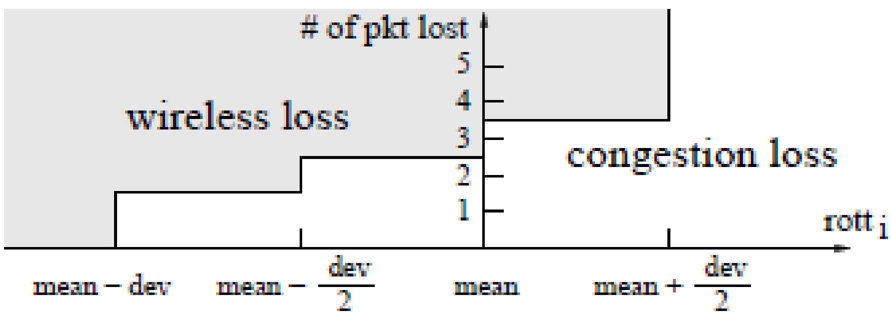
LDA (1)

"TFRC Wireless"

- Loss Differentiation Algorithm
  - Congestion loss <u>OR</u> Wireless error loss



Spike algorithm



ZigZag algorithm

$$B_{spikestart} = rott_{\min} + \alpha \cdot (rott_{\max} - rott_{\min})$$
$$B_{spikeend} = rott_{\min} + \beta \cdot (rott_{\max} - rott_{\min})$$
$$\alpha = 1/2, \ \ \beta = 1/3$$

ROTT: Relative One-way Trip Time

S.Cen et al: "End-to-end differentiation of congestion and wireless losses", IEEE/ACM Trans. Networking, 2003.

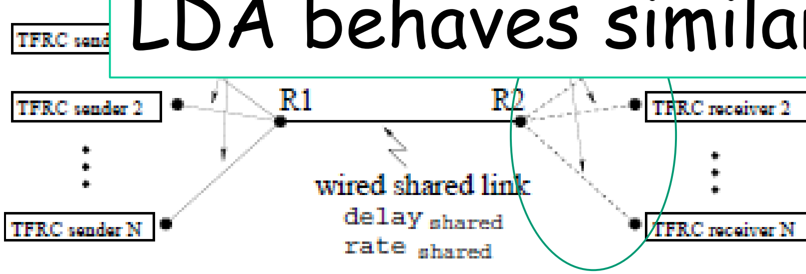
LDA (2)

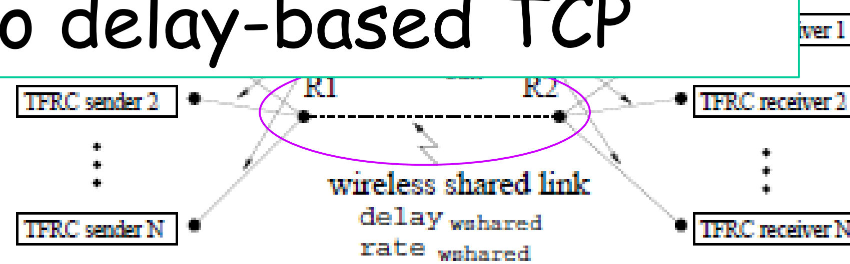
"TFRC Wireless"

- Simulation results

in Table,
- throughput
- congestion loss
- congestion loss, estimated as wireless loss
- wireless loss, estimated as congestion loss

Perform better than original TFRC because LDA behaves similar to delay-based TCP





PERFORMANCE FOR WIRELESS LAST HOP, 1 FLOW

	TCP	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	55	84	99	99	99	99	98
cong.	0.8	0.2	2.3	2.3	2.3	0.4	0.3
M_c	0	0	0	0.0	0.0	0.0	0.0
M_w	100	100	0	6.3	6.6	58	66

LDA

PERFORMANCE FOR WIRELESS BACKBONE, 1 FLOW

	TCP	TFRC	Omni	Biaz	mBiaz	Spike	ZigZag
thput	23	37	99	97	91	99	53
cong.	0.1	0.0	0.4	0.4	0.4	0.0	0.0
M_c	0	0	0	0.0	0.0	0.0	0.0
M_w	100	100	0	2.4	7.0	29	60

LDA

S.Cen et al: "End-to-end differentiation of congestion and wireless losses", IEEE/ACM Trans. Networking, 2003.

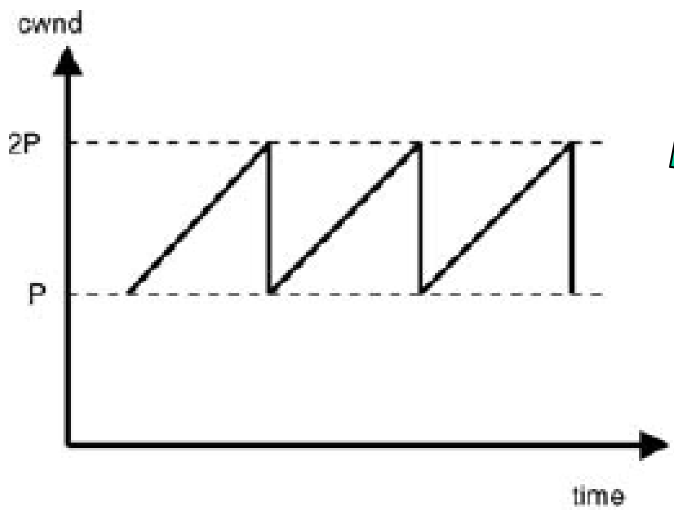
VTP (1)

- Video Transport Protocol
  - LDA（differentiation of congestion loss and wireless loss ～TFRC Wireless）
  - rate estimation similar to TCPW-RE（Achieved Rate）
  - TCP-Reno emulation（friendliness to legacy TCP）

G.Yang et al: “Smooth and efficient real-time video transport in the presence of wireless errors”, ACM Trans. MCCAP, 2006.

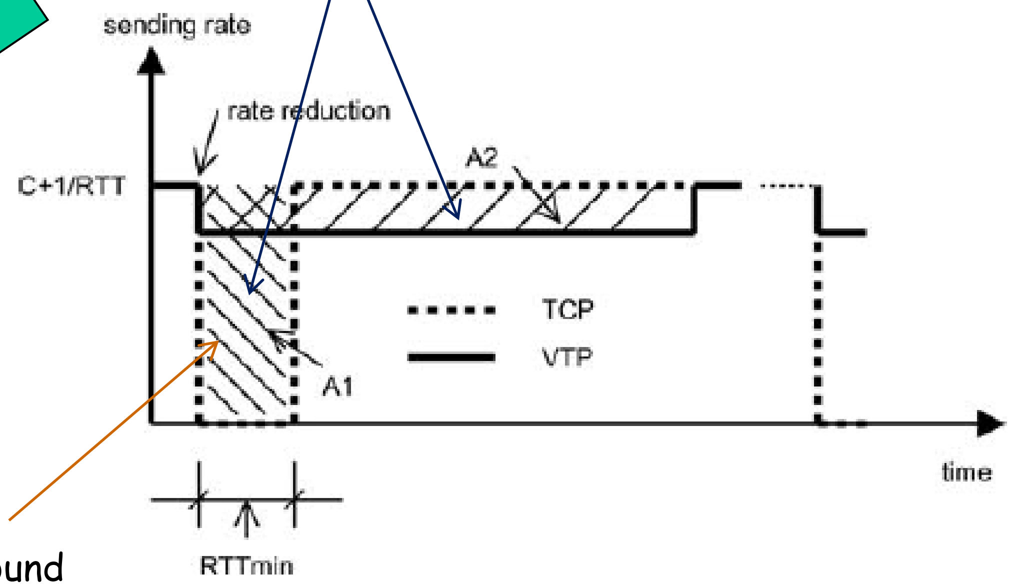
VTP (2)

- VTP overview



assume Buffer size = BDP

A1=A2: # of packets which can be transmitted during the retransmission round



retransmission round

G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

VTP (3)

- VTP's window control
  - init: Achieved Rate by TCPW-RE
  - update: 1 packet increase per RTT

$$R_0 = \text{Achived Rate}$$

$$ewnd_k = R_k \times RTT_k$$

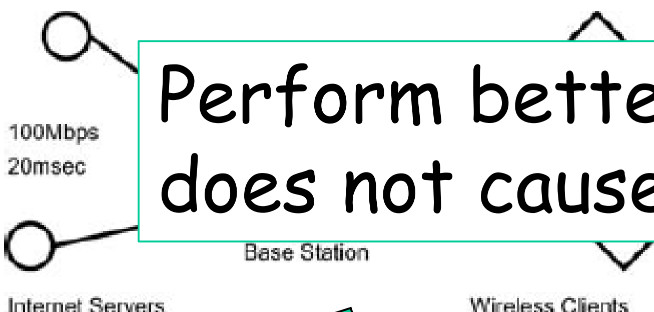
if no RTT increase, $R_{k+1} = R_k + \dfrac{1}{RTT_k}$

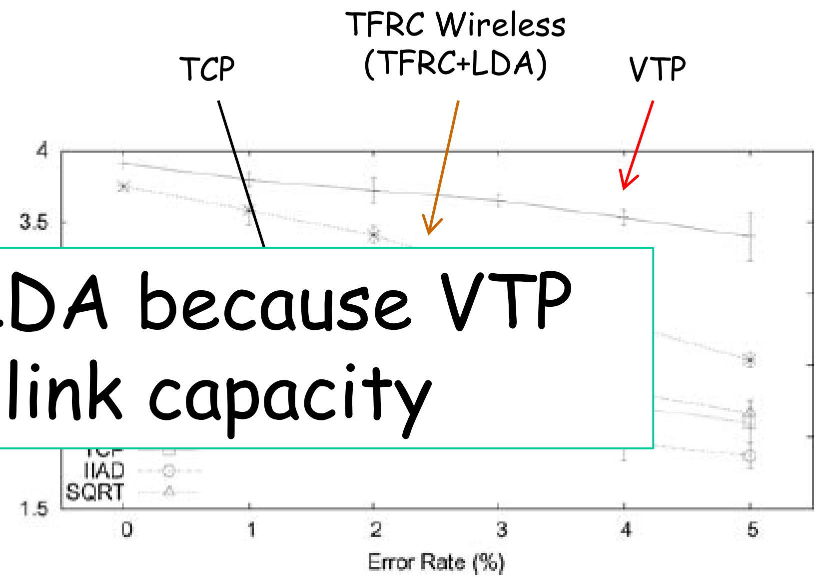
$$R_{k+1} = \frac{ewnd_{k+1}}{RTT_{k+1}} = \frac{ewnd_{k+1}}{RTT_k + \Delta RTT_k} = \frac{R_k \times RTT_k + 1}{RTT_k + (RTT_k - RTT_{k-1})}$$

G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

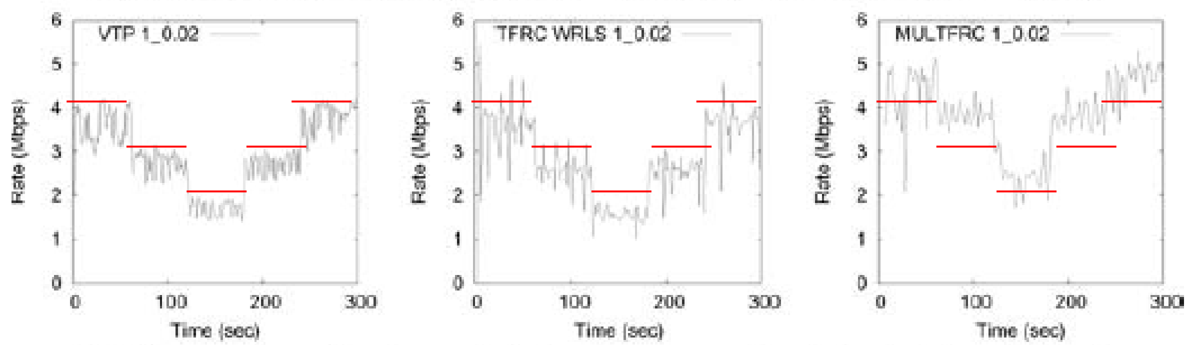
VTP (4)

- simulation results





Perform better than LDA because VTP does not cause vacant link capacity



(b) 2% errors (avg. time in good state = 1 sec, avg. time in bad state = 0.02 sec).

MULTFRC: use multiple TFRC connections

G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

VTP (5)

- disadvantage of VTP
  - assumes only the case that Buffer size = BDP
  - no consideration on the vacant capacity which happens when Buffer size < BDP

Google Congestion Control (GCC)

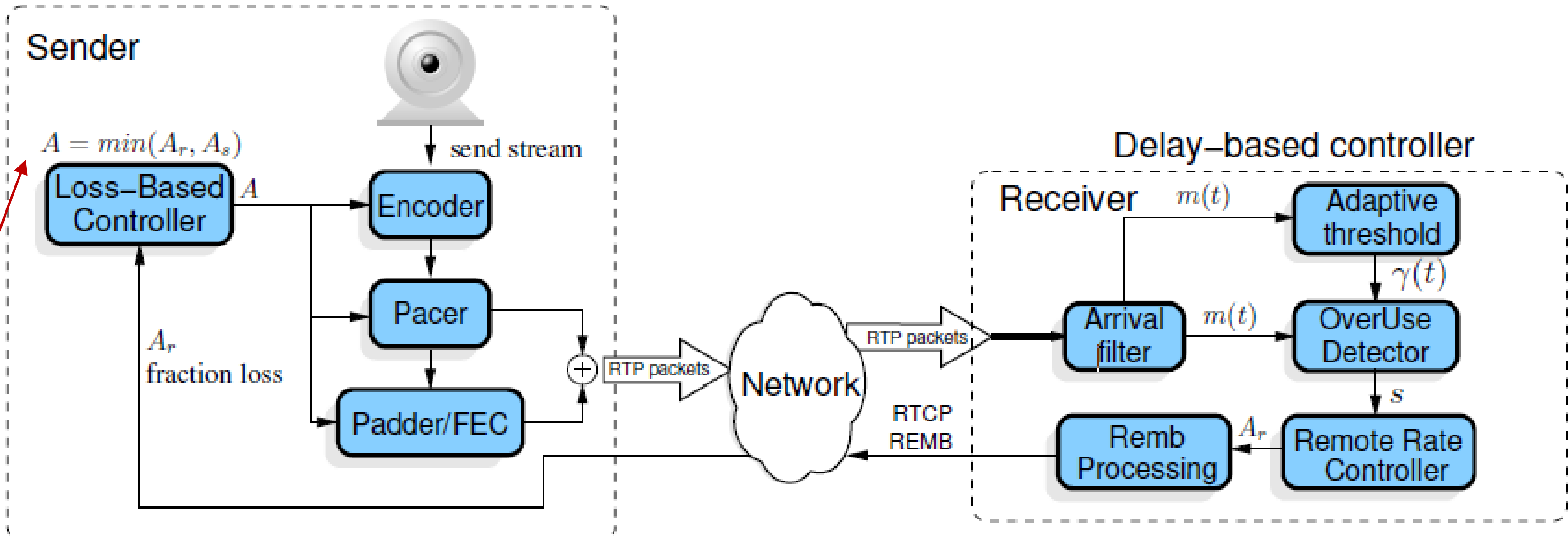
used in Google Hangout and WebRTC, …

G.Carlucci et al: "Analysis and Design of the Google Congestion Control for WebRTC", ACM MMSys, 2016.

Google Congestion Control

- designed for RTP/RTCP content delivery over UDP
  - hybrid congestion control by delay-based controller and loss-based controller
- used by Google Hangout and WebRTC in Chrome browser
- one of e2e congestion control algorithms discussed in IETF RMCAT (RTP Media Congestion Avoidance Techniques) WG
  - NADA (Network Assisted Dynamic Adaptation)
  - SCREAM (Self-Clocked Rate Adaptation for Multimedia)
  - GCC (Google Congestion Control)

Google Congestion Control

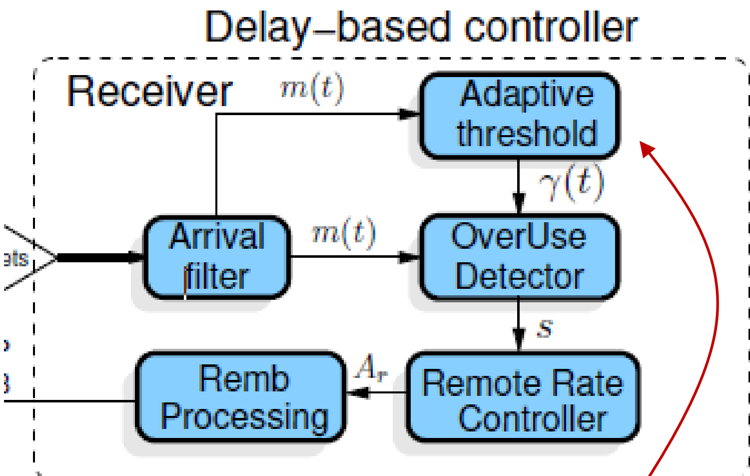


REMB: Receiver Estimated Maximum Bitrate
(RTP/RTCP feedback extension)

A_r: expected sending rate calculated by <u>delay-based controller at a receiver side</u>

A_s: target sending rate calculated by <u>loss-based controller at a sender side</u>

Delay-based Controller

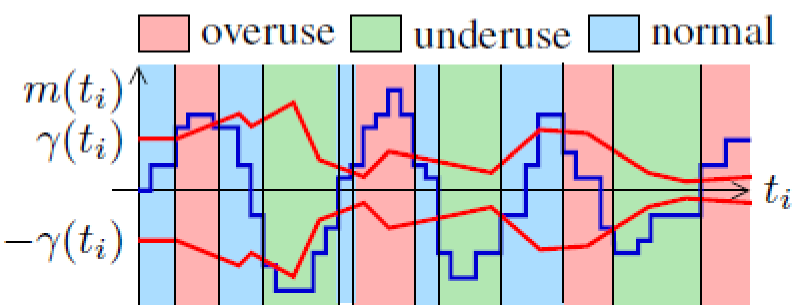


$m(t)$: estimated delay gradient at time t by Kalman filter

$\gamma(t)$: threshold to judge three states at time t; "overuse" ($m(t) > \gamma(t)$), "underuse" ($m(t) < -\gamma(t)$), and "normal" (otherwise)

s: one of three states

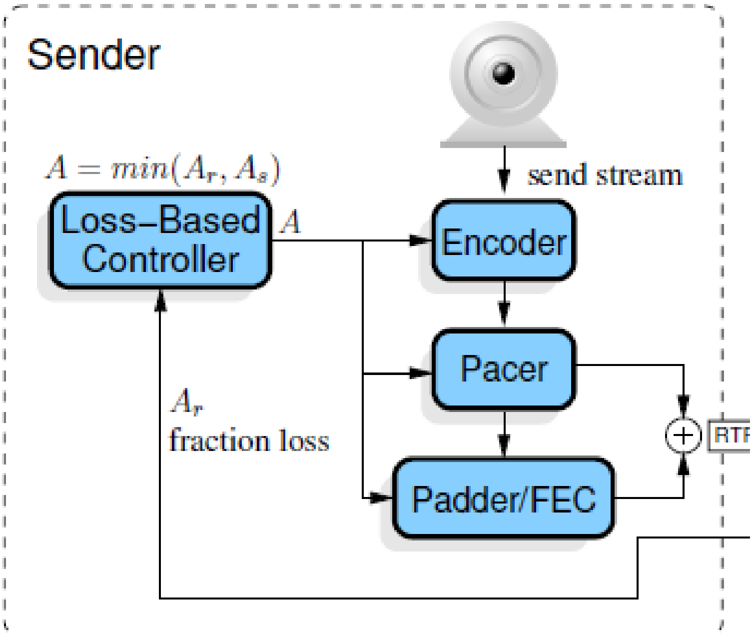
A_r: expected sending rate (feedback to a sender)



receiving rate in last 500ms

$$A_r(t) = \begin{cases} 0.85 \cdot R_r(t-1) & \text{(overuse)} \\ 1.05 \cdot A_r(t-1) & \text{(underuse)} \\ A_r(t-1) & \text{(normal)} \end{cases}$$

Loss-based Controller



RTCP feedback (A_r, PLR)

PLR: Packet Loss Rate

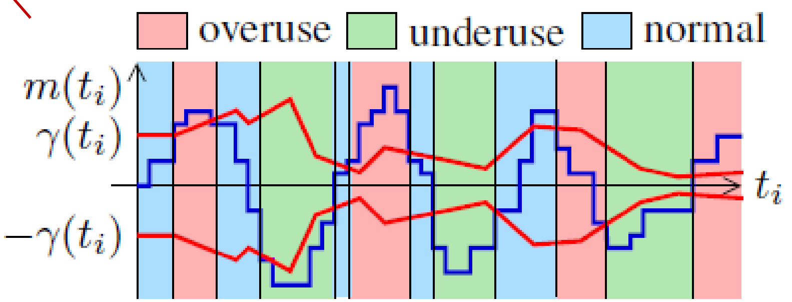
$$A_s(t) = \begin{cases} (1 - PLR) \cdot A_s(t-1) & (PLR > 0.1) \\ 1.05 \cdot A_s(t-1) & (PLR < 0.02) \\ A_s(t-1) & (\text{otherwise}) \end{cases}$$

$$A = \min(A_r, A_s)$$

Adaptive Threshold

$$\gamma(t) = \gamma(t-1) + \Delta T \cdot k_r(t) \cdot \big(|m(t)| - \gamma(t-1)\big)$$

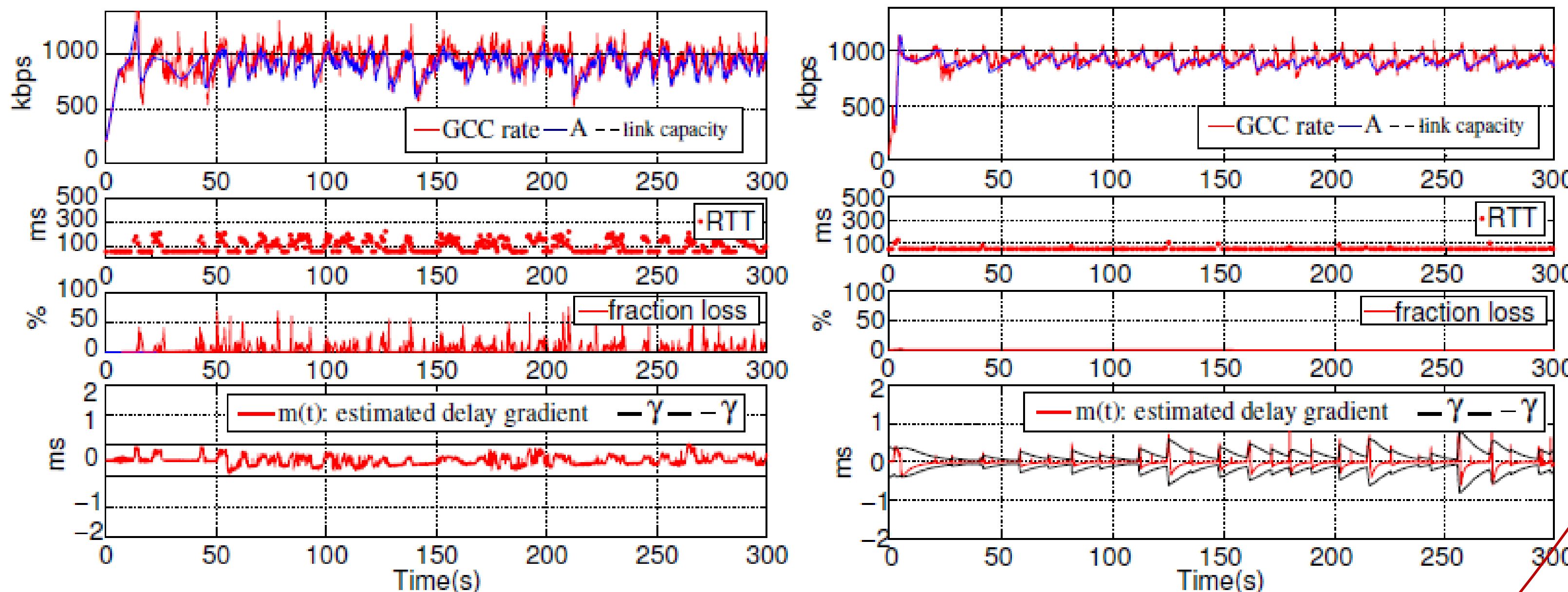
time interval   gain



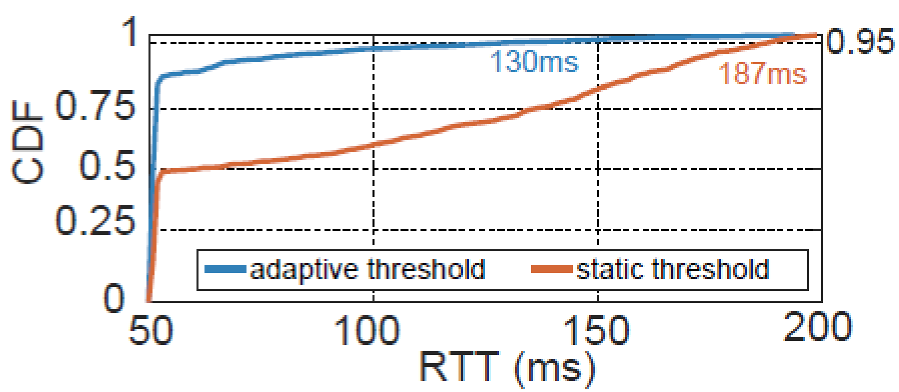
analogous to adaptive filter

- threshold γ(t) gradually converges to estimated delay gradient |m(t)|
- |m(t)| is gradually controlled to be smaller than threshold γ(t)

Single GCC Flow



(a) Static threshold (b) Adaptive threshold



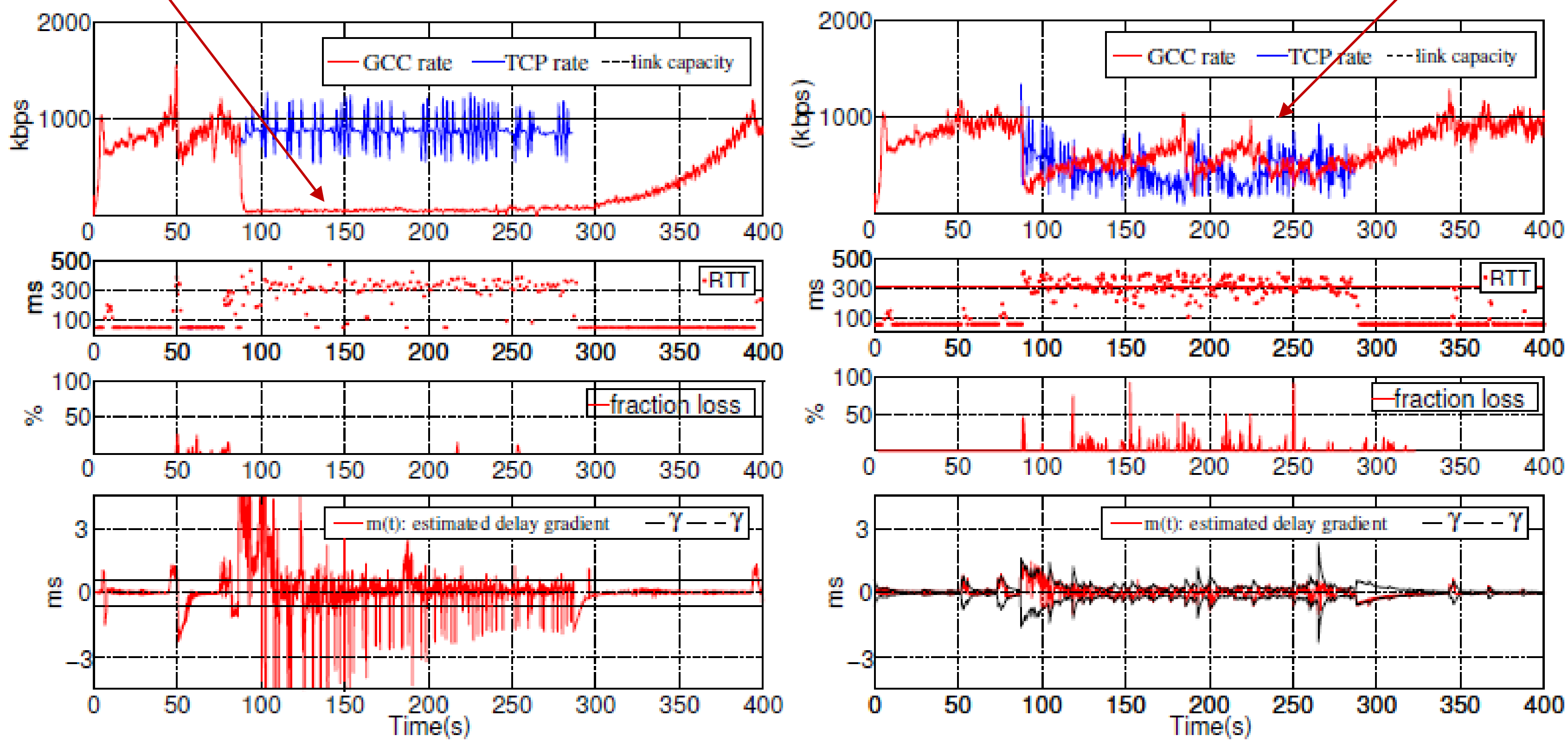
adaptive threshold contributes to

- smaller RTTs (low delay)
- smaller PLRs (high QoE)

GCC vs CUBIC TCP

starved

shared



(a) Static threshold

(b) Adaptive threshold

adaptive threshold contributes to co-existence of GCC flow (over RTP/UDP) with CUBIC TCP flow