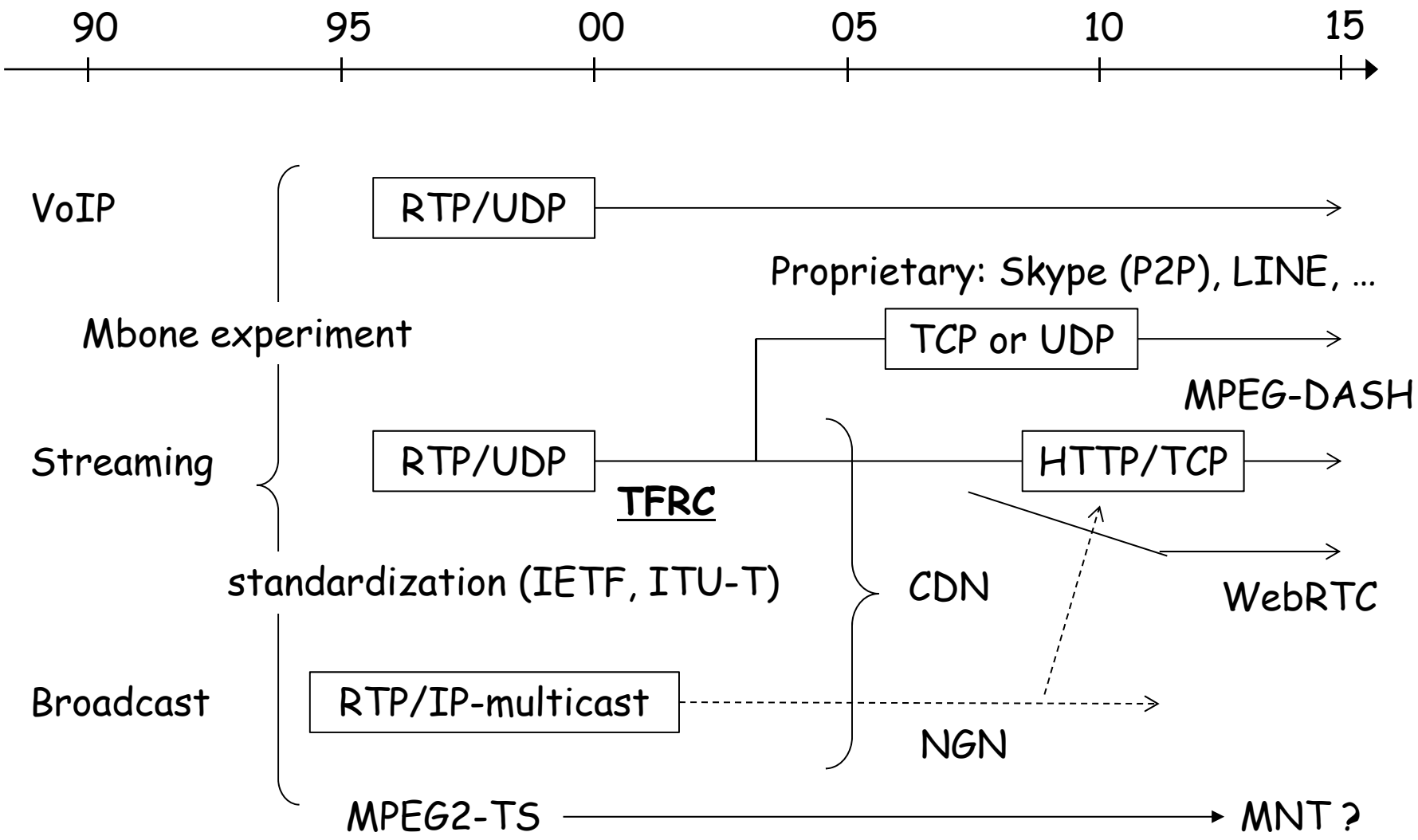
画像情報特論 (3)
Advanced Image Information (3)

TFRC and its Variants

情報理工学専攻　甲藤二郎

E-Mail: katto@waseda.jp

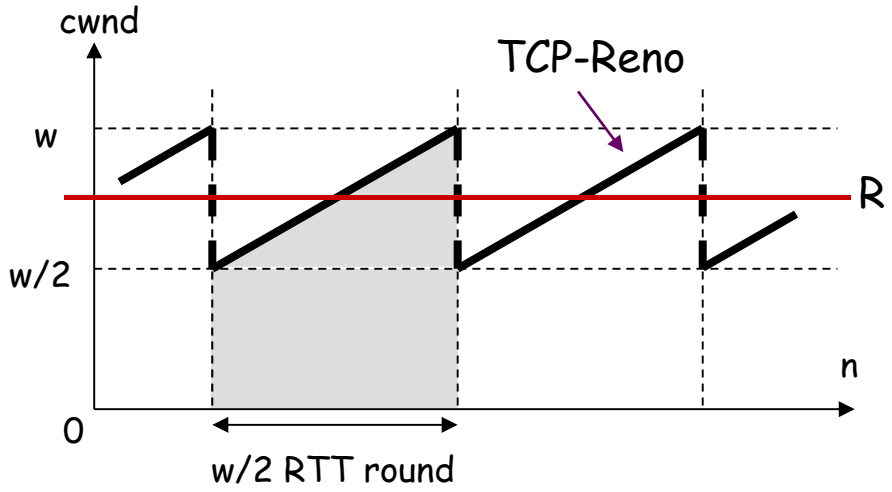
Protocol Transition



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

with timeout consideration

$$\begin{cases} p = \dfrac{8}{3w^2} \\ R = \dfrac{PS}{RTT} \cdot \sqrt{\dfrac{3}{2p}} \end{cases} \Rightarrow R_{loss} = \frac{PS}{RTT\sqrt{\frac{2bp}{3}} + t_{RTO,loss} \cdot 3\sqrt{\frac{3bp}{8}} \cdot p(1+32p^2)}$$

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

TCP Westwood

- Duplicate ACKs

FSE: Fair Share Estimates

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

$$if\ (cwnd > ssthreh)\ cwnd = ssthresh$$

- Timeout

in TCP-Reno case

$$ssthresh = cwnd / 2$$

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

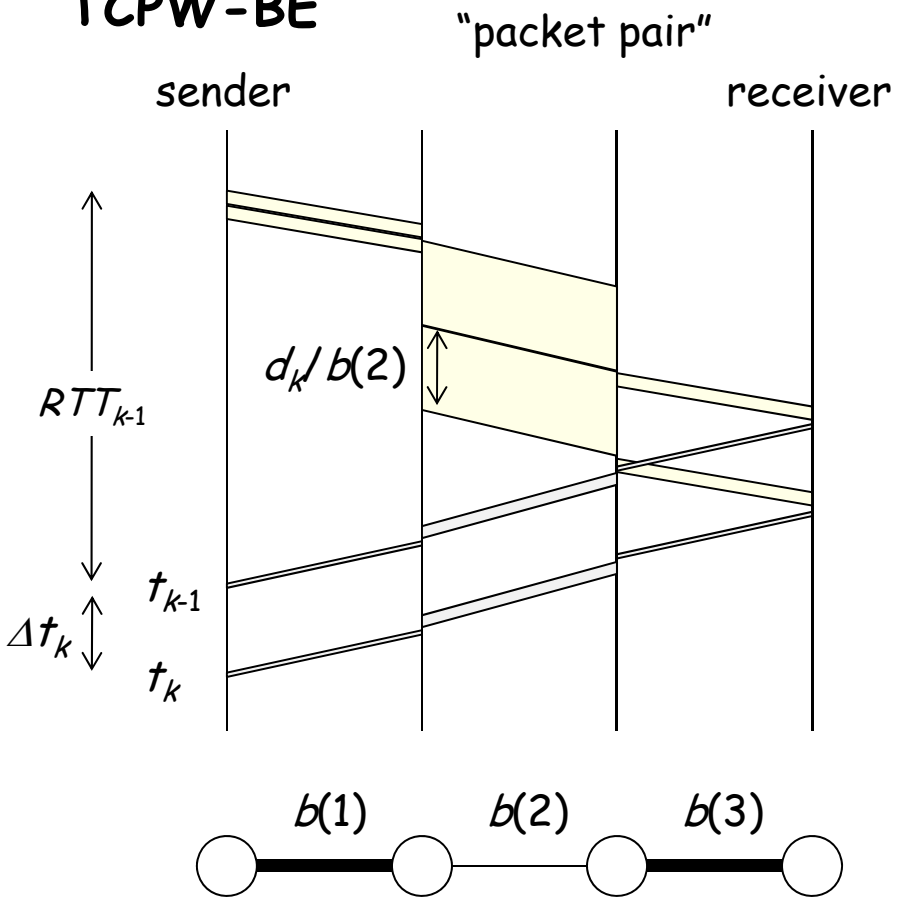
$$cwnd = 1$$

- multiple versions according to FSE estimation methods

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

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

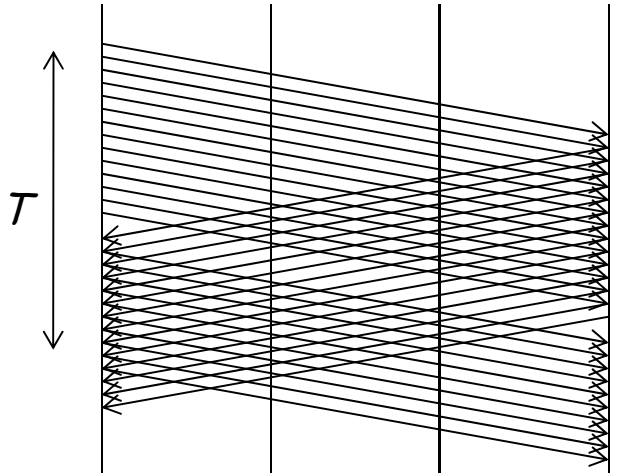
C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM MOBICOM 2001.

Rate Estimation

(reference) TCP-Vegas

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

expect rate — actual rate



TCPW-RE:

$$RE_k = \frac{\sum_{t_j > t_k - T} d_j}{T}$$

cwnd $\Rightarrow$ S=Σd_k

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

$$\hat{RE}_k \approx \frac{\sum d_k}{\sum \Delta t_k}$$

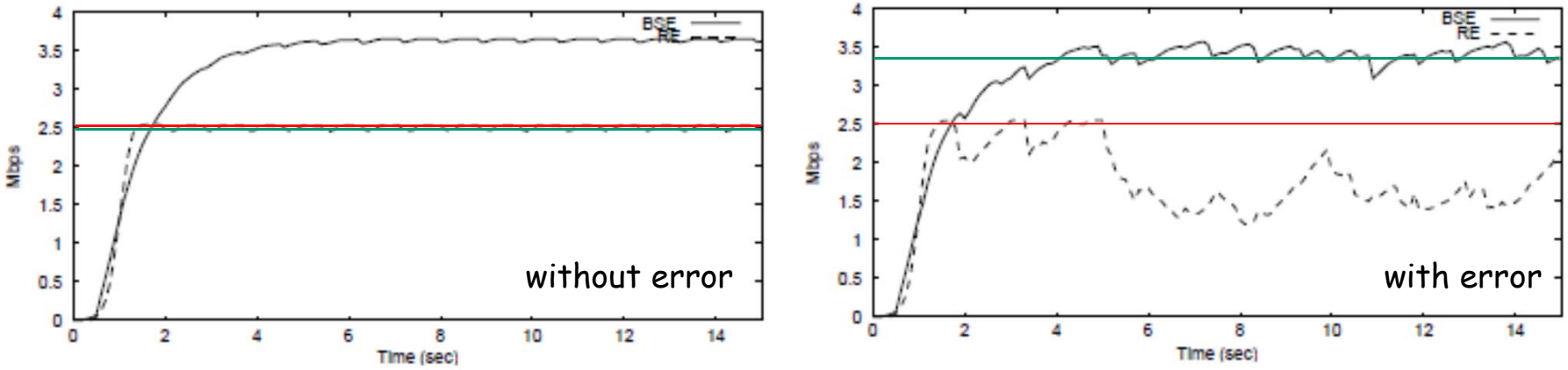
moving average: $\hat{RE}_k \rightarrow FSE$

$T = n \cdot RTT$ (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

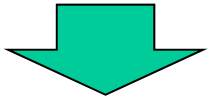


- BSE tends to overestimate (due to burstiness)
- RE tends to underestimate when losses occur

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)

TCPW-ABSE:

actual rate

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

T_{min} : ACK arrival interval

$\hat{T}h \cdot RTT_{\min} < cwnd$ → 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.

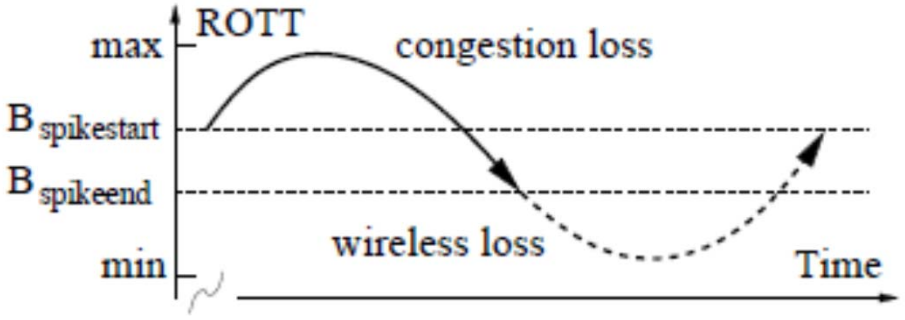
Disadvantage of TFRC

- 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

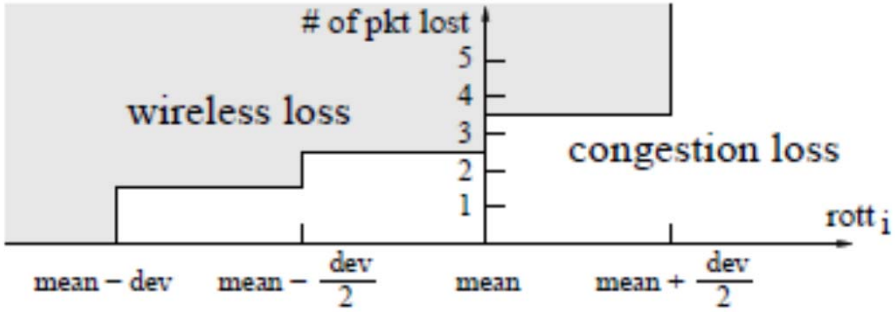
LDA (1)

"TFRC Wireless"

- Loss Differentiation Algorithm
  - Congestion loss OR 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.

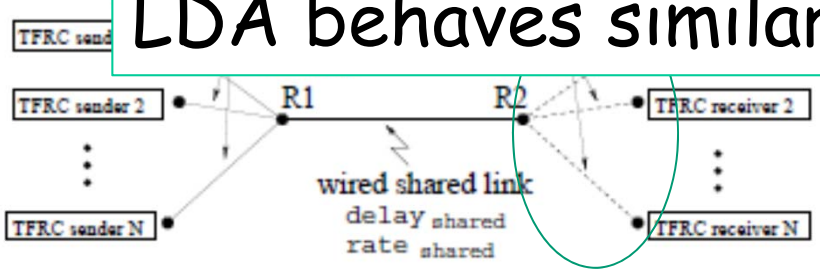
LDA (2)

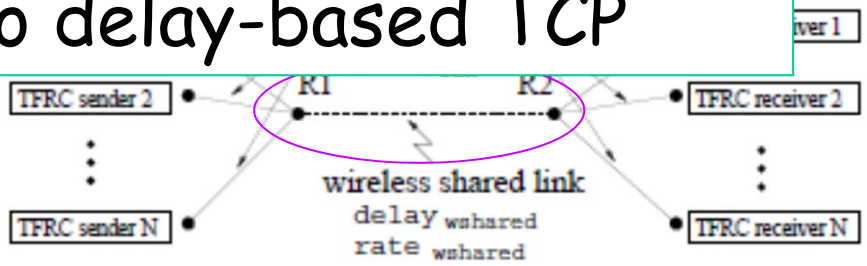
"TFRC Wireless"

- 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.

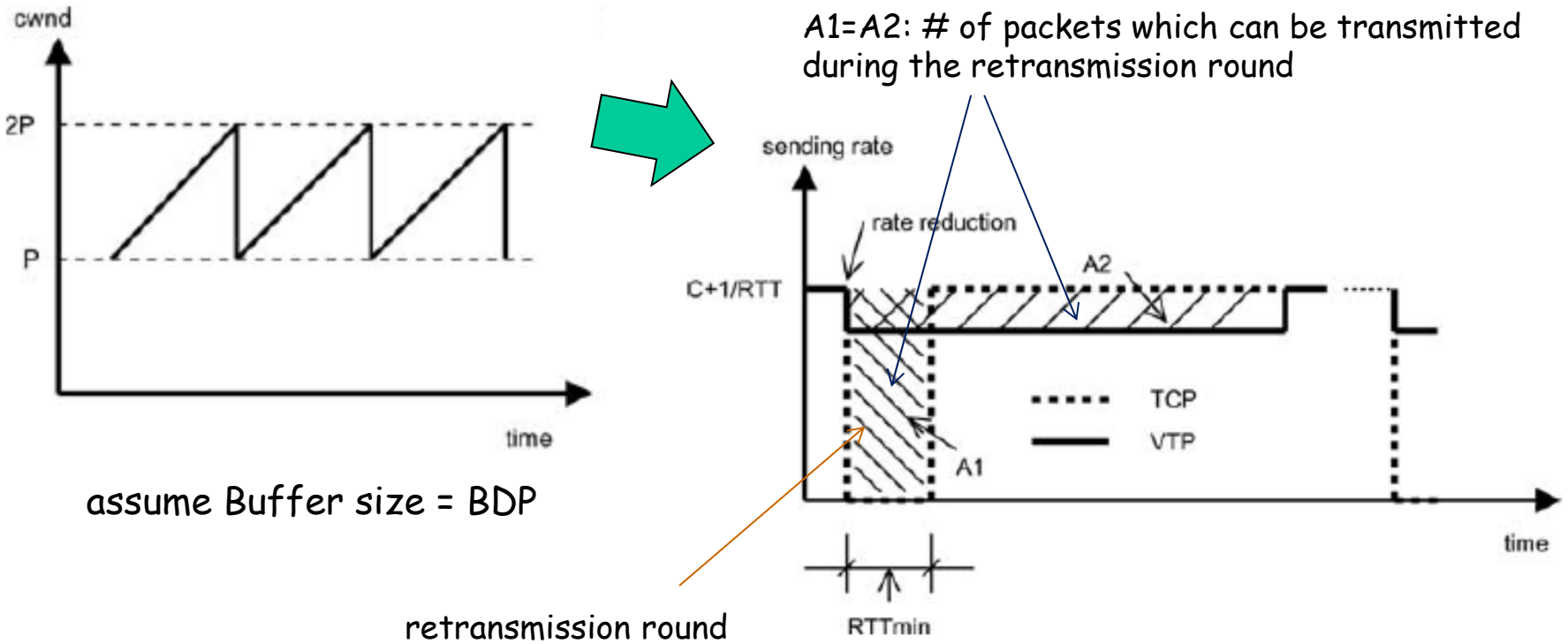
VTP (1)

- 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



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

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

$$ewnd_k = R_k \times RTT_k$$

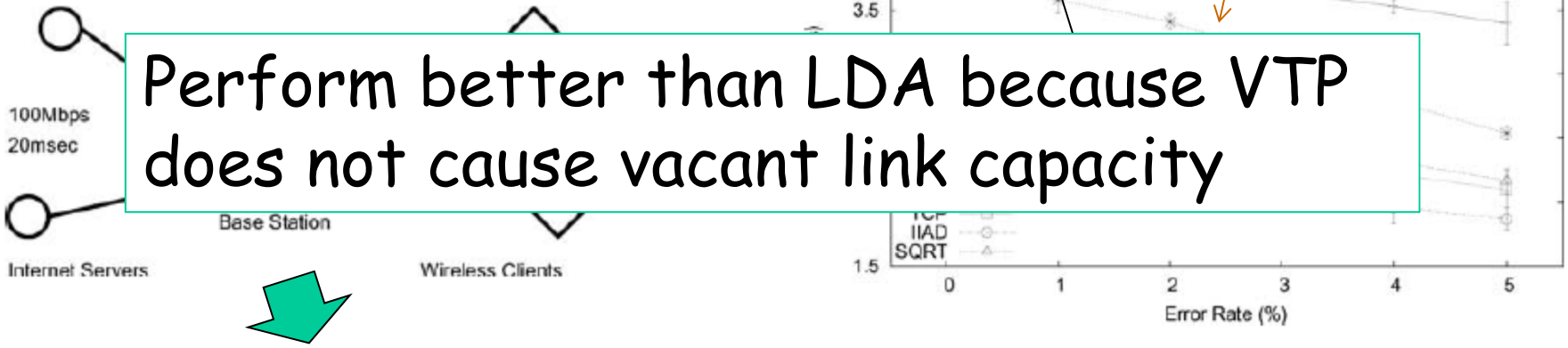
if no RTT increase, $R_{k+1} = R_k + \frac{1}{RTT_k}$

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

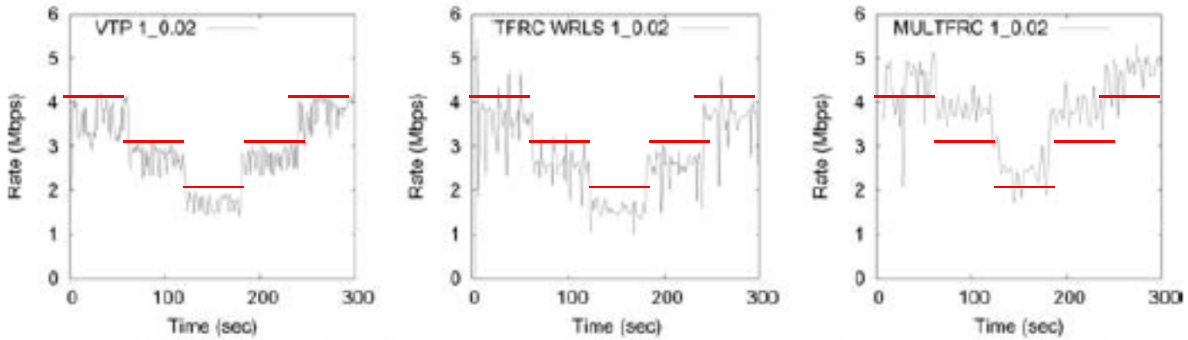
G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

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

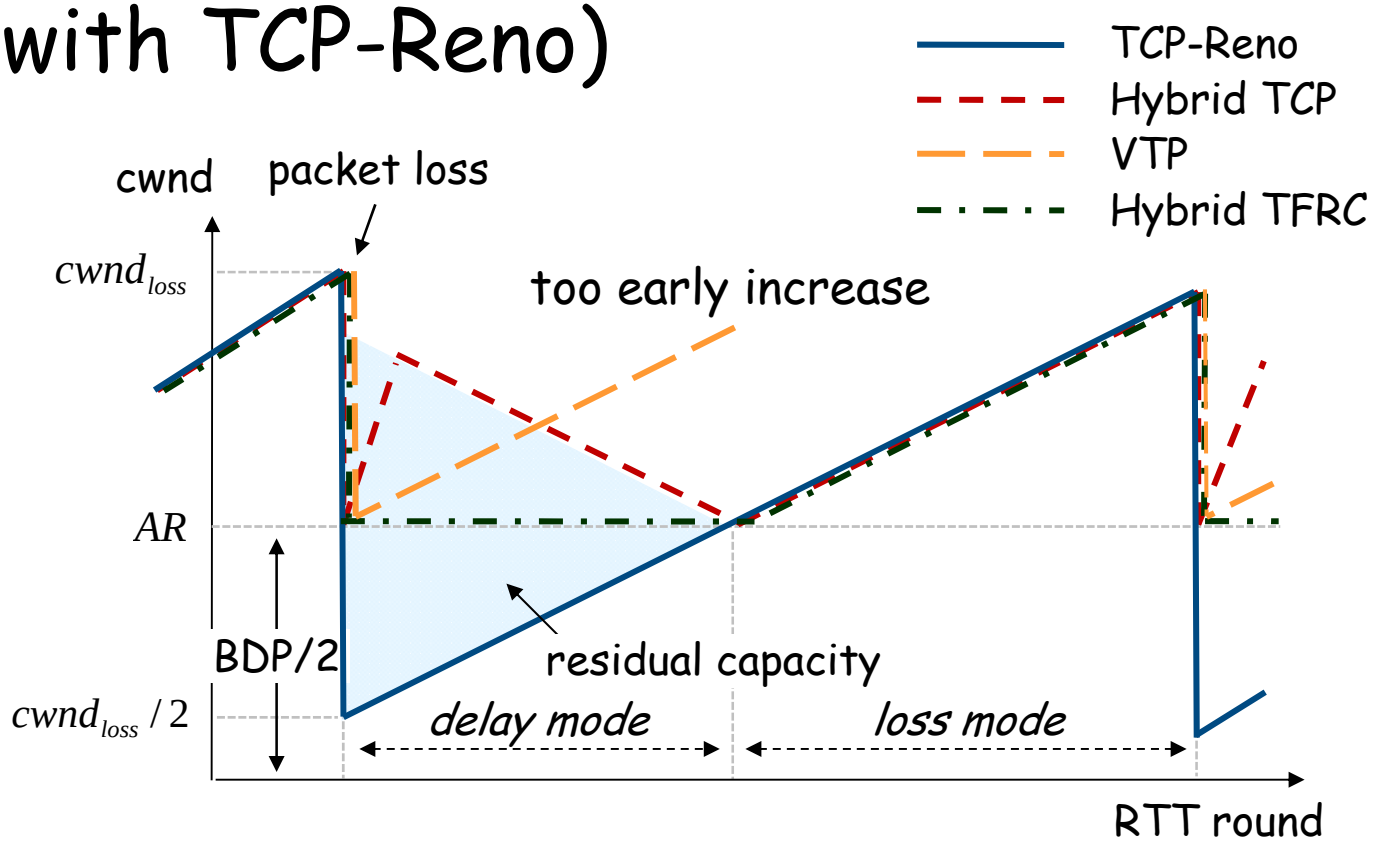
Hybrid TFRC (1)

- TFRC extension of Hybrid TCP
  - uses Achieved Rate when no RTT increase is observed(efficiency)
  - uses VTP-like window control when RTT increase is observed (friendliness)
  - works in small router buffer $\Rightarrow$ small RTT

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

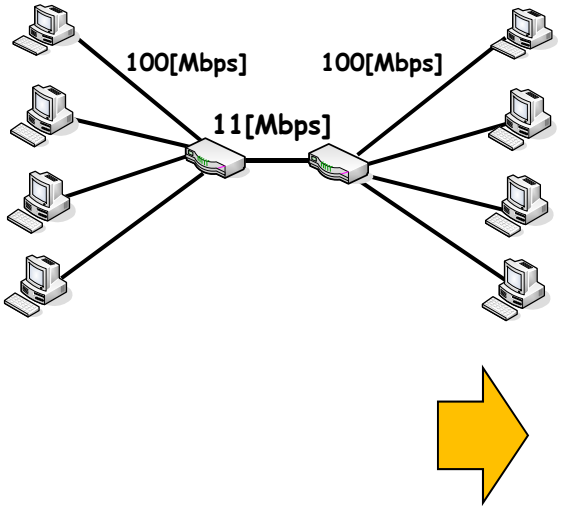
Hybrid TFRC (2)

- Hybrid TCP behavior (when competing with TCP-Reno)

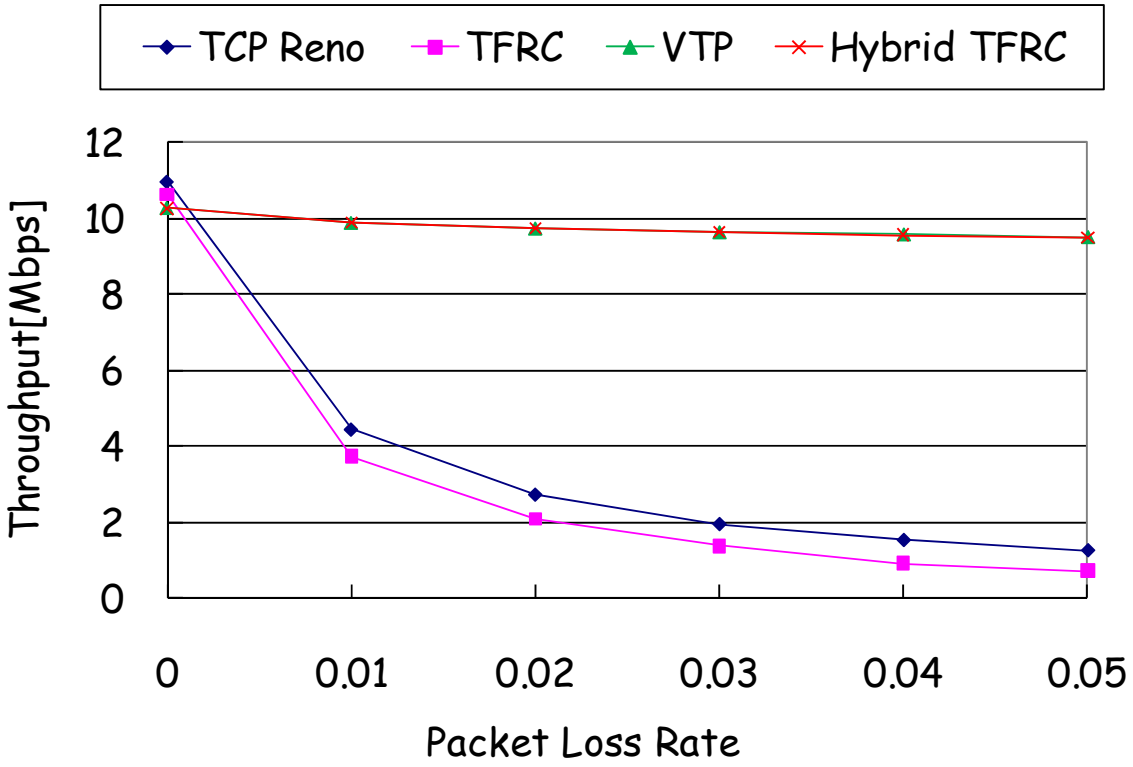


Hybrid TFRC (3)

- simulation result (1)



Buffer size = BDP

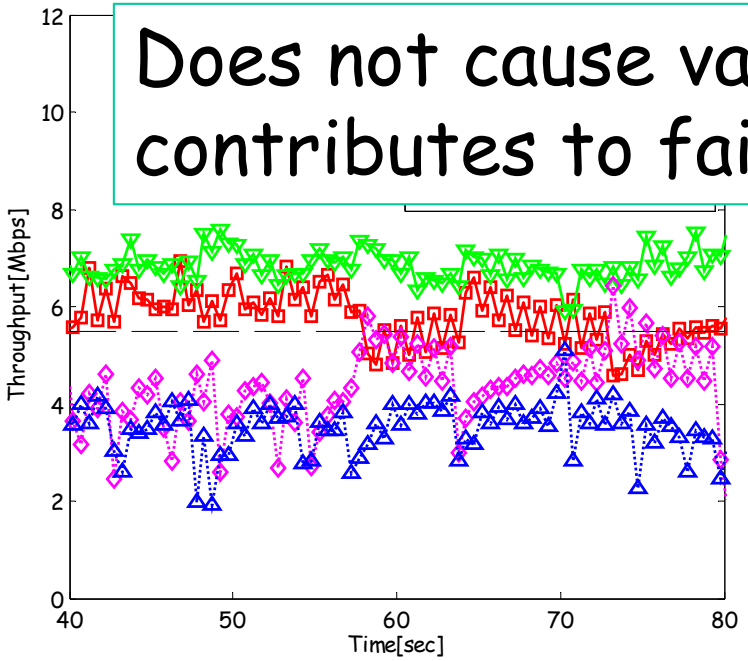


Hybrid TFRC (4)

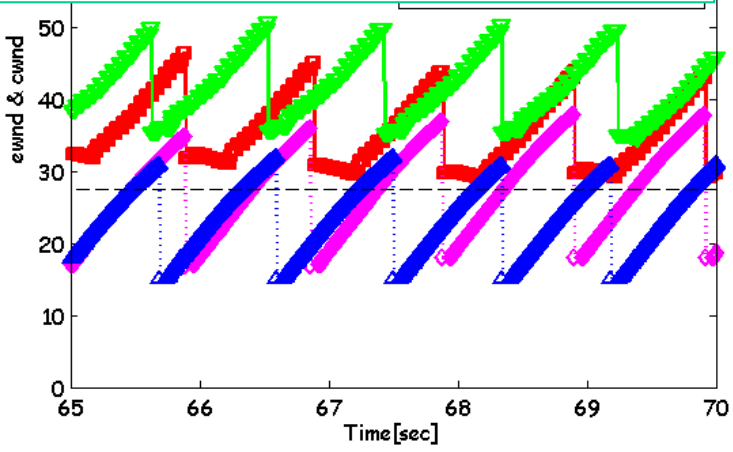
- simulation result (2)

Buffer size = BDP/2

Does not cause vacant link capacity, and contributes to fairness improvement



Throughput



ewnd & cwnd

TFWC: TCP Friendly Window-based CC

TCP-Friendly "Window-based" Congestion Control (1)

- disadvantages of TFRC
  - Friendliness: small buffer causes unfairness (expels competing TCPs)
  - Smoothness (1): transmission rate fluctuates (due to RTT instability)
  - Smoothness (2): rate calculation fluctuates when RTT is very small
  - Responsiveness: convergence speed is slow against flows' ON/OFF characteristics
- Rate-based $\Rightarrow$ Window-based

S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

TCP-Friendly Window-based Congestion Control (2)

- Ack Vector:
  - A receiver returns an aggregated ACK packet (Ack Vector) to a sender
  - A sender calculates an average packet loss interval (1/p) and updates cwnd by

$$W_{TWRC} = \frac{1}{\sqrt{\frac{2p}{3}} + 12\sqrt{\frac{3p}{8}} \cdot p(1+32p^2)} \quad \Leftrightarrow \quad R_{TFRC} = \frac{PS}{RTT\sqrt{\frac{2p}{3}} + 12 \cdot RTT\sqrt{\frac{3p}{8}} \cdot p(1+32p^2)}$$

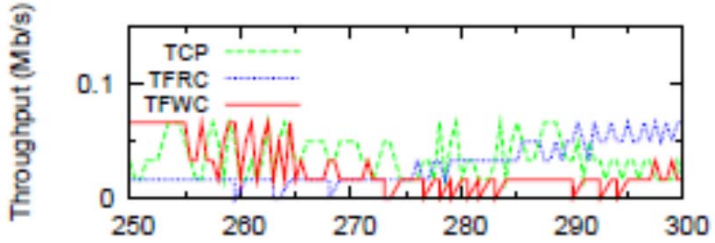
  - stable because RTT is eliminated

S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

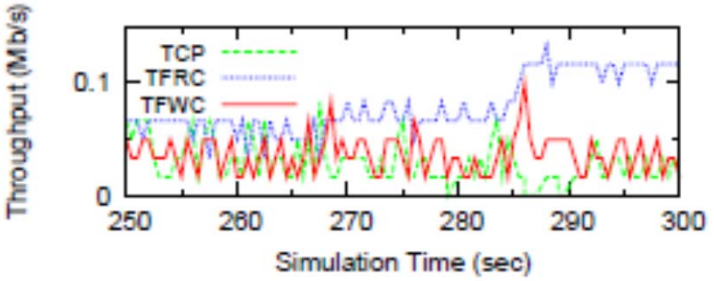
TCP-Friendly Window-based Congestion Control (3)

- Hybrid Window & Rate
  - too small cwnd causes timeout

  - operates in TFRC when cwnd is too small



(a) operated by window-based only



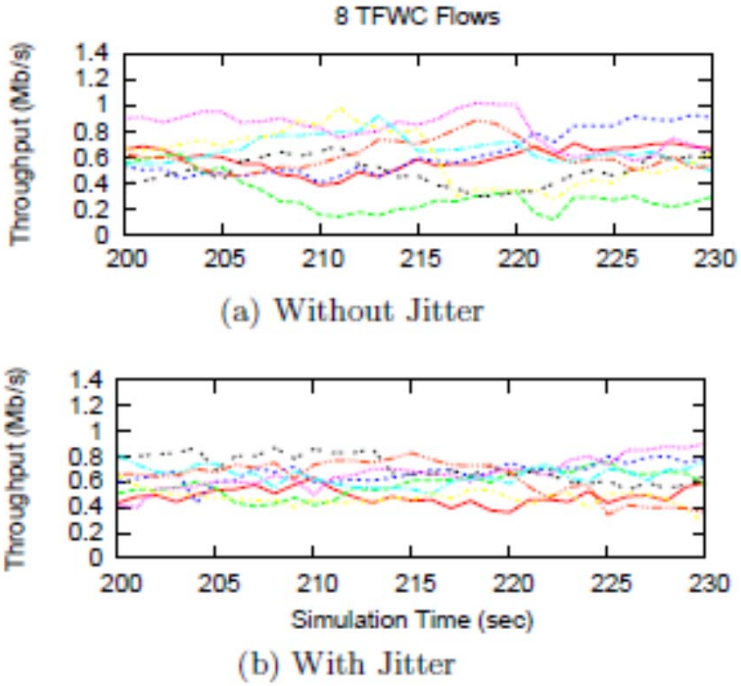
(b) operated by window/rate-based

S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

TCP-Friendly Window-based Congestion Control (4)

- Drop tail and RED
  - Drop tail sometimes causes PLR increase of competing flow ("synchronization" effect)
  - RED (Random Early Drop) enables random drop

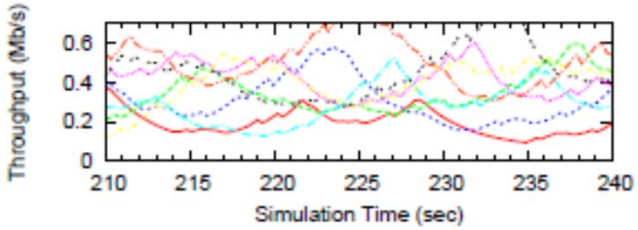
- Analogy: add small "random number" to cwnd



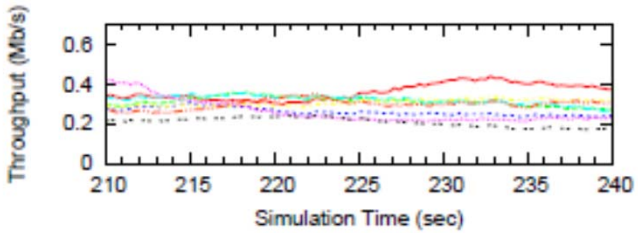
S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

TCP-Friendly Window-based Congestion Control (5)

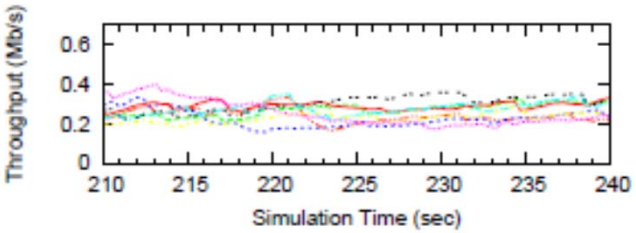
- Smoothness



(a) TCP's Abrupt Sending Rate Change

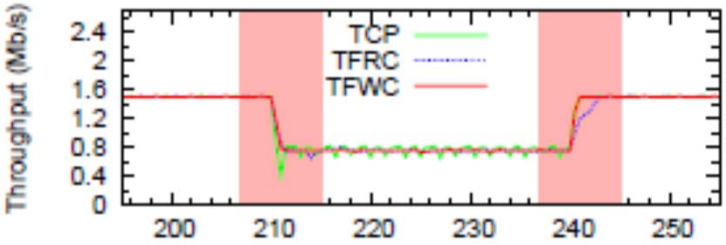


(b) TFRC Smoothness

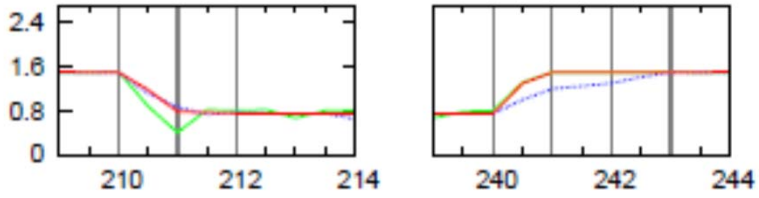


(c) TFWC Smoothness

- Responsiveness



(a) Impulse Response



(b) Zoom-In (high-lighted area)

S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.