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

TFRC & HTTP Streaming

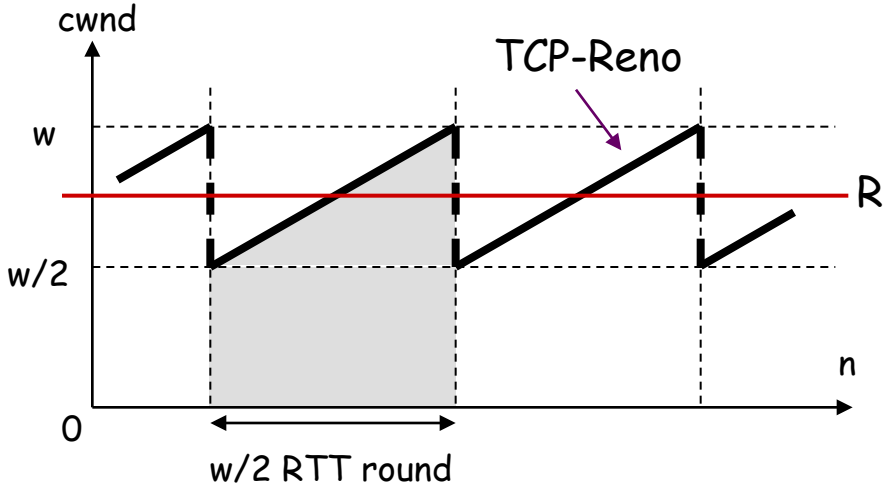
情報理工学専攻　甲藤二郎

E-Mail: katto@waseda.jp

TCP Equations

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

FSE: Fair Share Estimates

- Duplicate ACKs

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

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

- Timeout

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

$$cwnd = 1$$

in TCP-Reno case

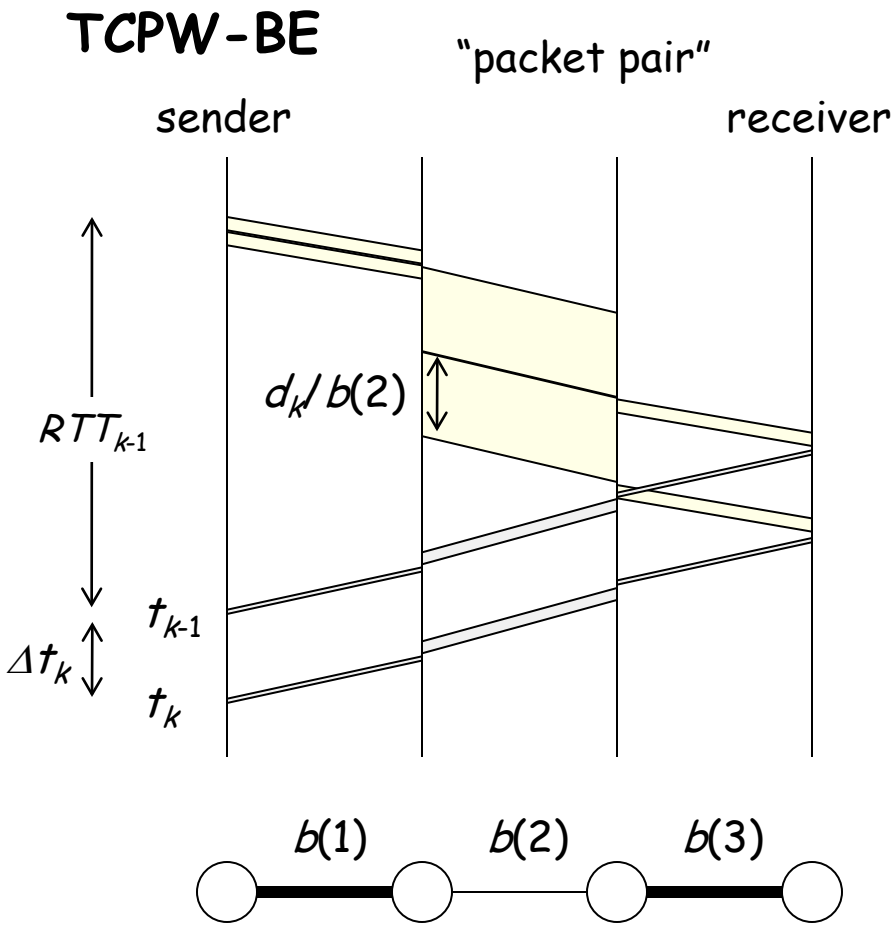
$$ssthresh = cwnd / 2$$

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

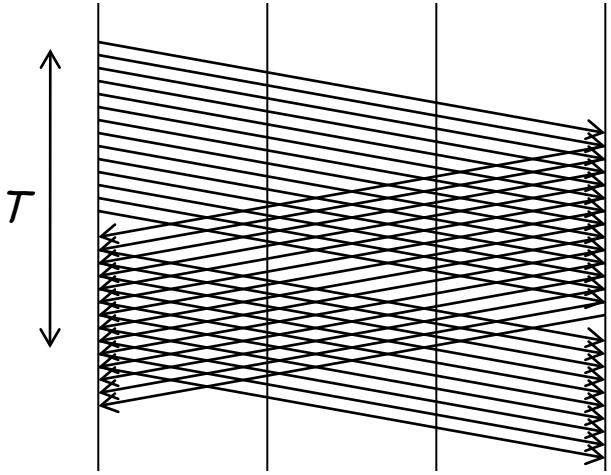
C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM MOBICOM 2001.

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= $\Sigma \Delta t_k$

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

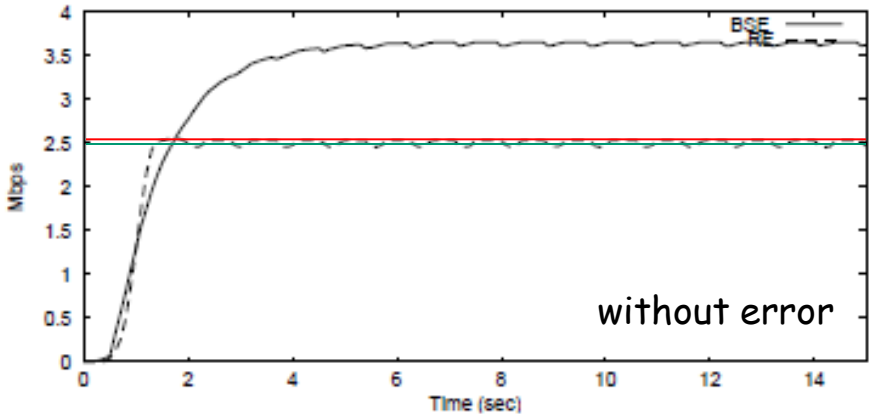
moving average: $\hat{RE}_k \rightarrow FSE$

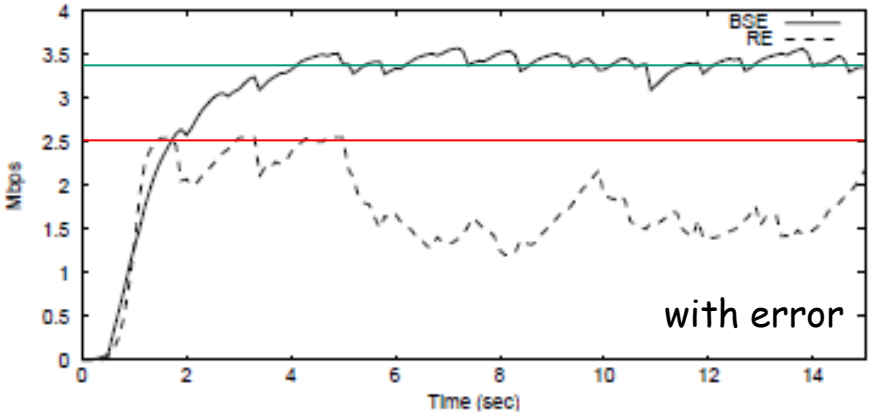
$T = n \cdot RTT$ (e.g. n=4)

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



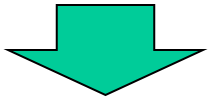


- BSE tends to overestimate (due to burstiness)
- RE tends to underestimate when losses occur

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.

HYbrid TFRC

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.

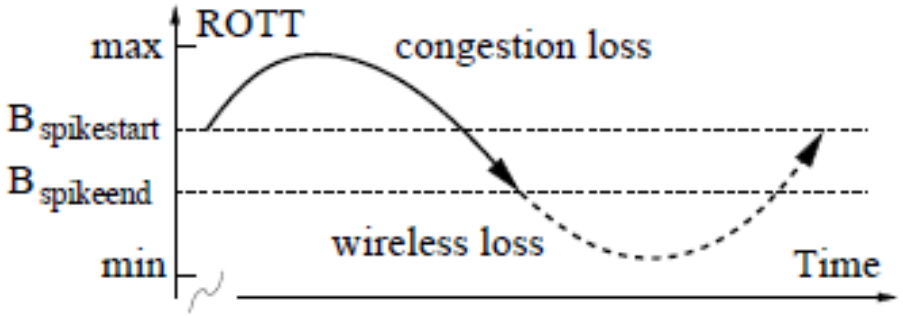
Disadvantage of TFRC

- 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

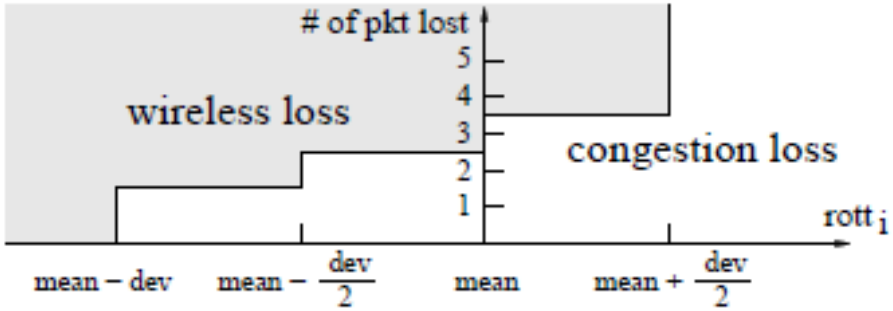
LDA (1)

"TFRC Wireless"

- Loss Differentiation Algorithm
  - Congestion loss / Wireless error loss



Spike algorithm



ZigZag algorithm

$$B_{spikestart} = rott_{\min} + \alpha \cdot (rott_{\max} - rott_{\min})$$

$$B_{spikeend} = rott_{\min} + \beta \cdot (rott_{\max} - rott_{\min})$$

$$\alpha = 1/2, \quad \beta = 1/3$$

ROTT: Relative One-way Trip Time

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

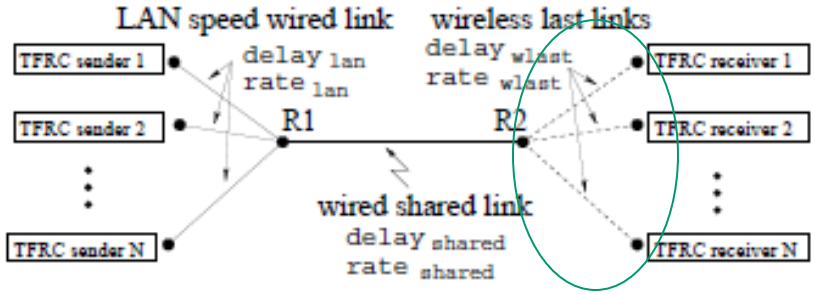
LDA (2)

"TFRC Wireless"

- Simulation results

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

Wireless last hop

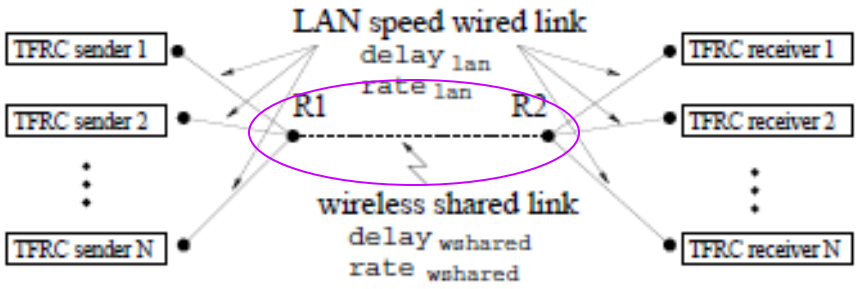


PERFORMANCE FOR WIRELESS LAST HOP, 1 FLOW

	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

Wireless backbone



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.

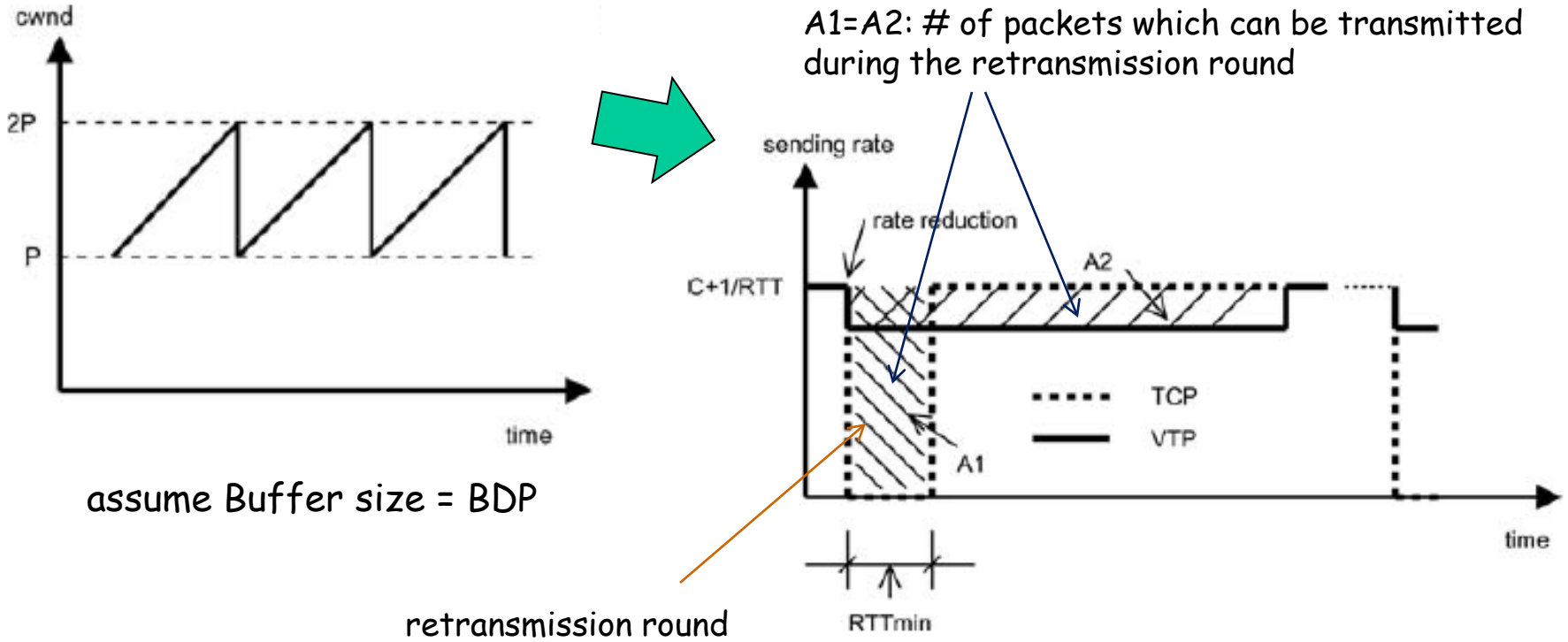
VTP (1)

- 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

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

$$ewnd_k = R_k \times RTT_k$$

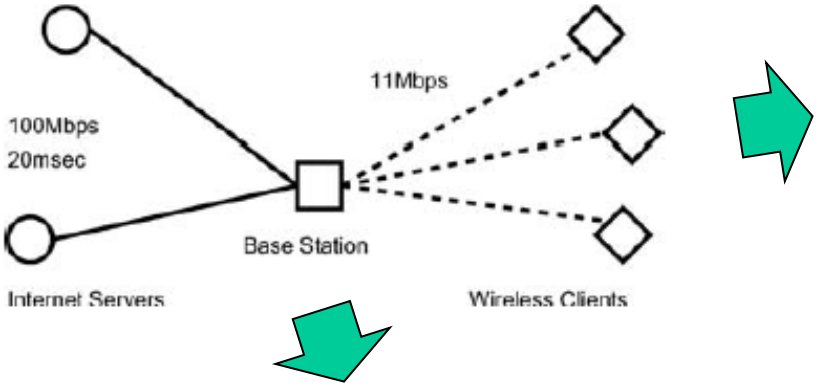
if no RTT increase, $R_{k+1} = R_k + \frac{1}{RTT_k}$

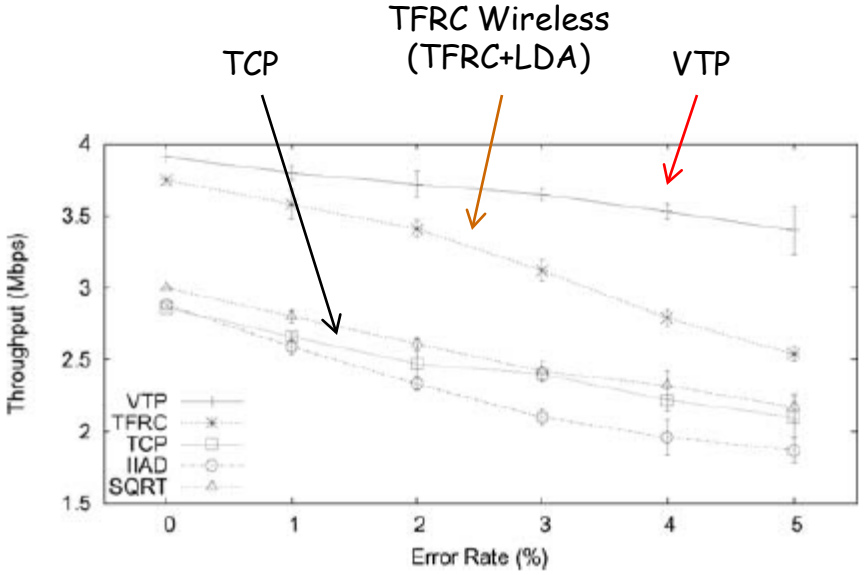
$$R_{k+1} = \frac{ewnd_{k+1}}{RTT_{k+1}} = \frac{ewnd_{k+1}}{RTT_k + \Delta RTT_k} = \frac{R_k \times RTT_k + 1}{RTT_k + (RTT_k - RTT_{k-1})}$$

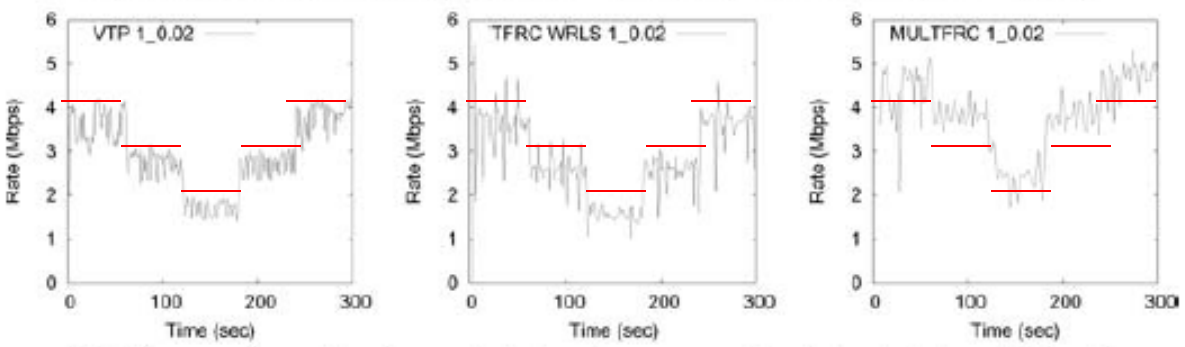
G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

VTP (4)

- simulation results







(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

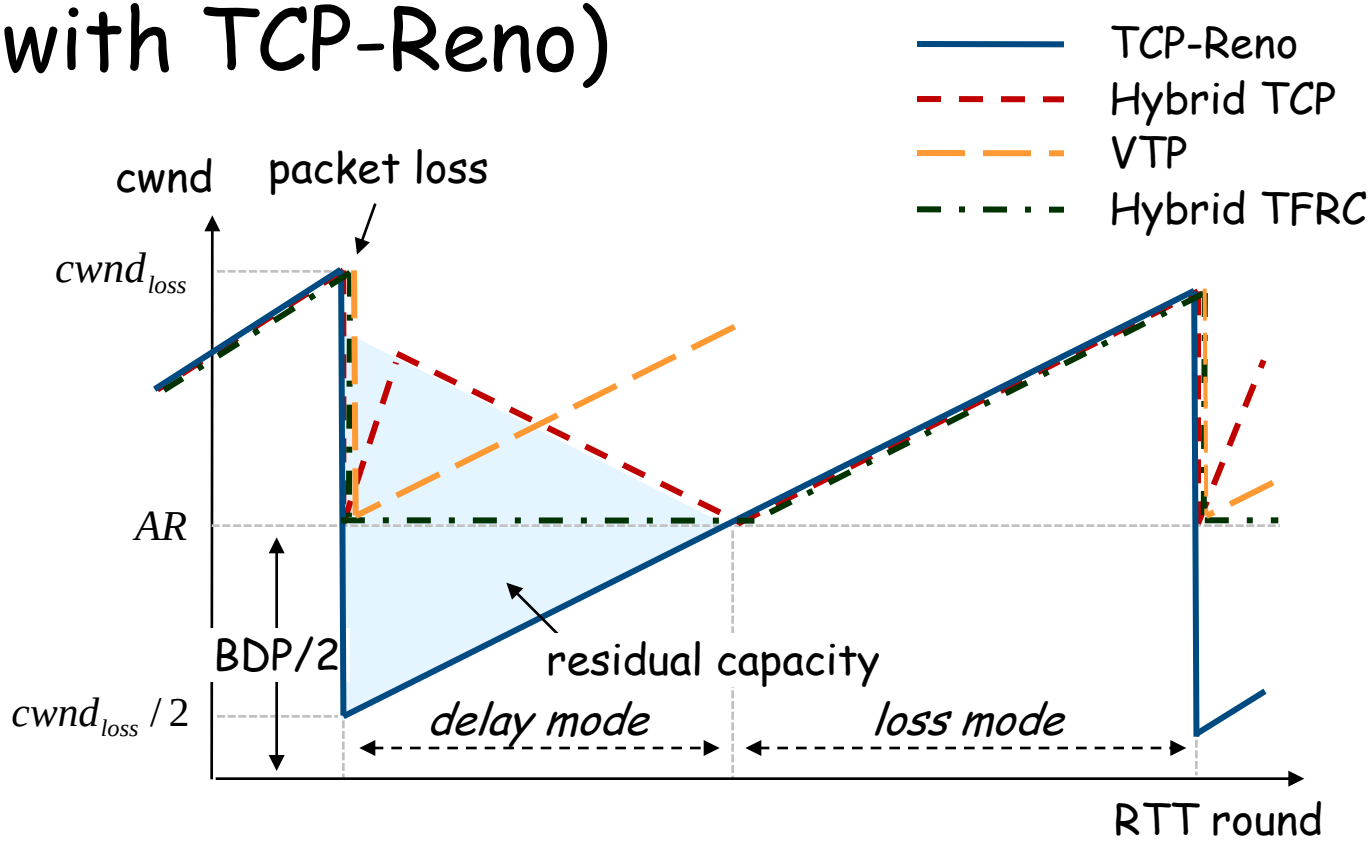
Hybrid TFRC (1)

- 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)

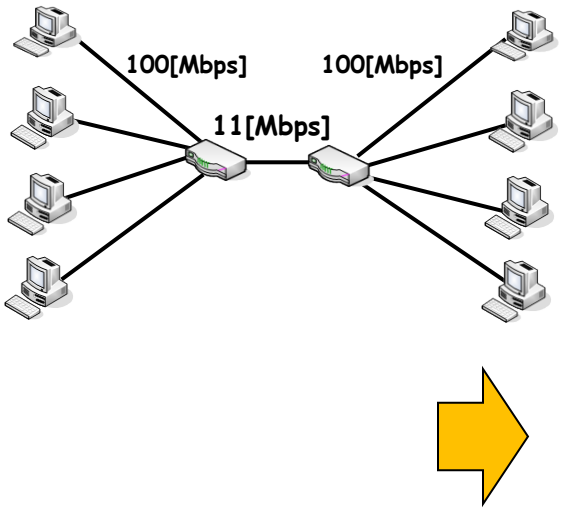
- Hybrid TCP behavior (when competing with TCP-Reno)

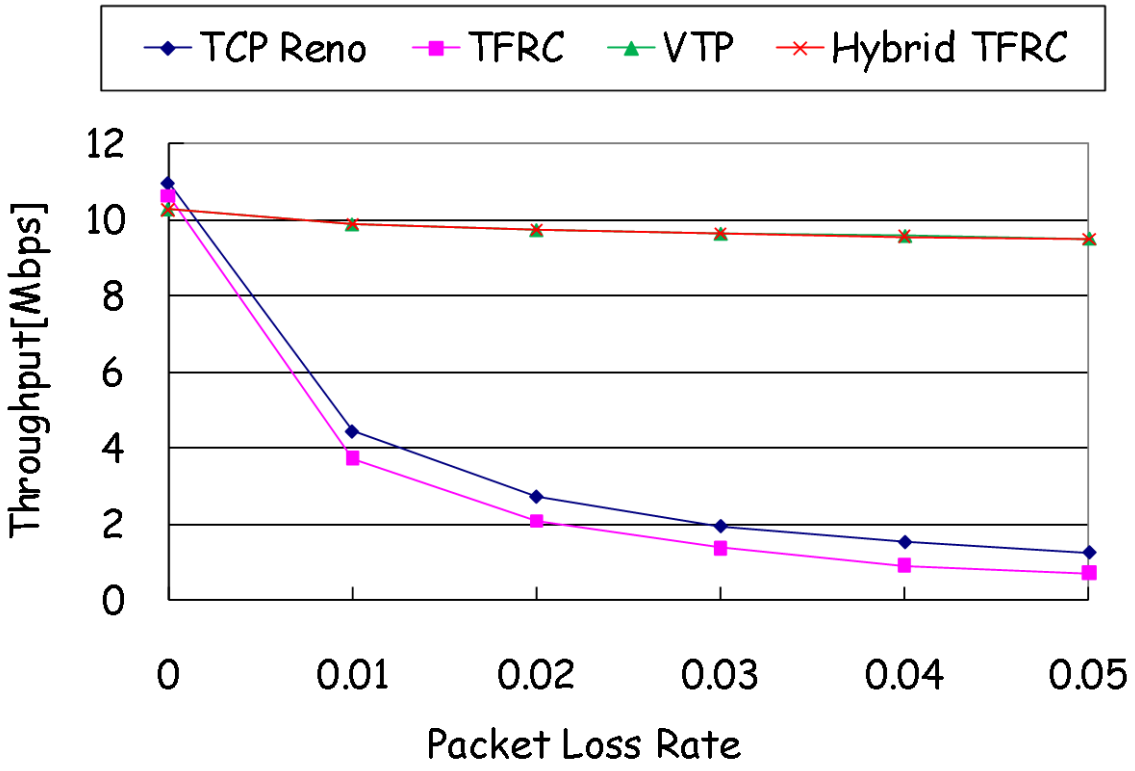


Hybrid TFRC (3)

- simulation result (1)

Buffer size = BDP

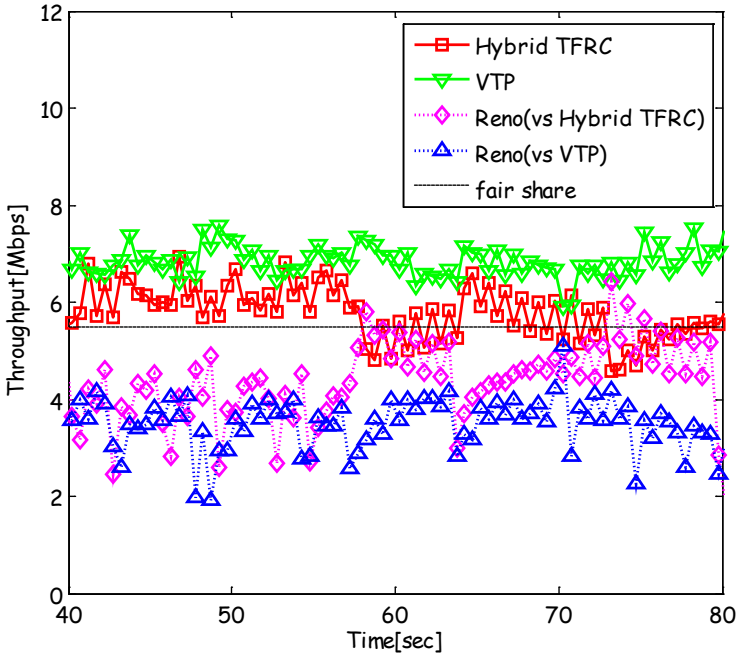




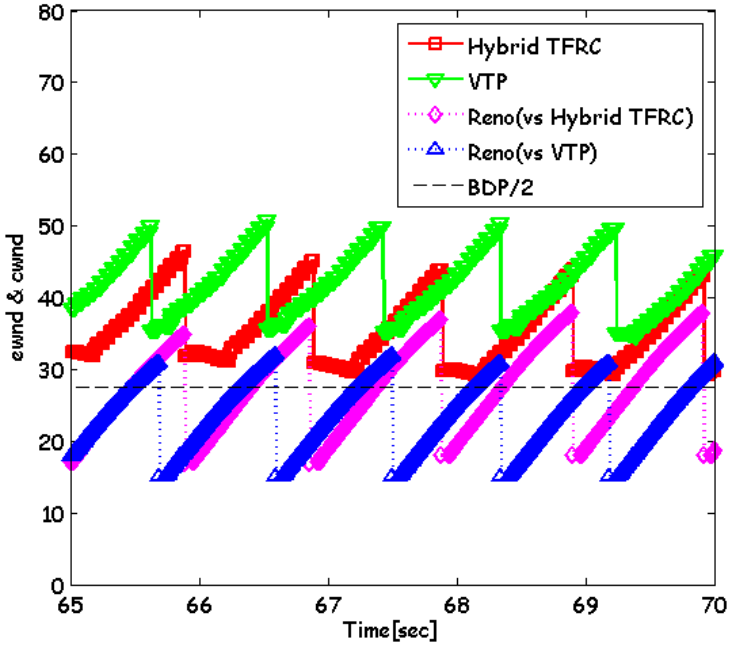
Hybrid TFRC (4)

- simulation result (2)

Buffer size = BDP/2



Throughput



ewnd & cwnd

TFWC & Relentless 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

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

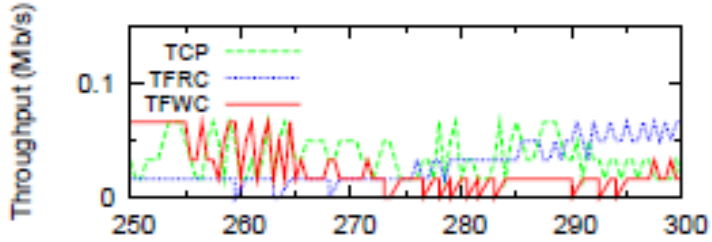
  - stable because RTT is eliminated

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

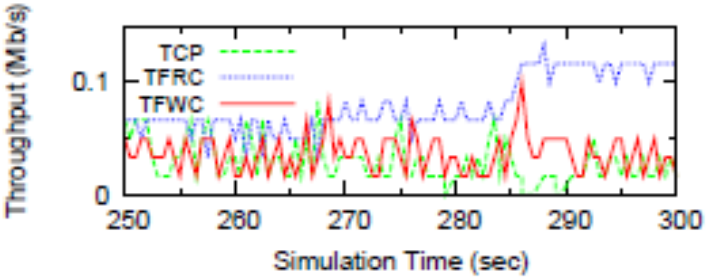
TCP-Friendly Window-based Congestion Control (3)

- Hybrid Window & Rate
  - too small cwnd causes timeout

  - operates in TFRC when cwnd is too small



(a) operated by window-based only



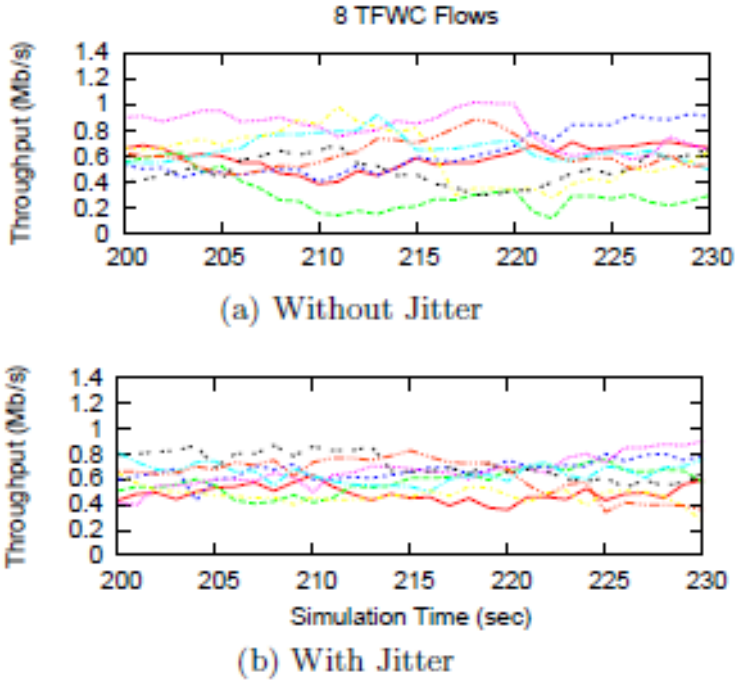
(b) operated by window/rate-based

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

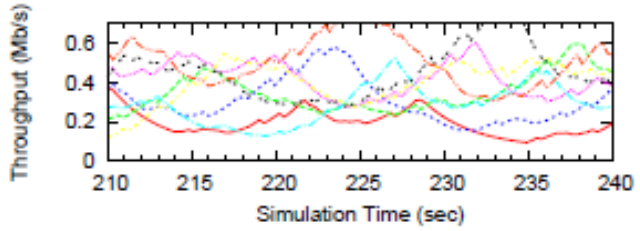
- Analogy: add small "random number" to cwnd



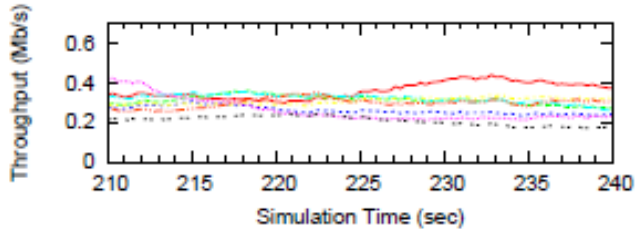
S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

TCP-Friendly Window-based Congestion Control (5)

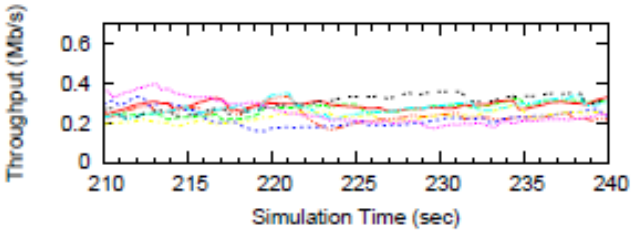
- Smoothness



(a) TCP's Abrupt Sending Rate Change

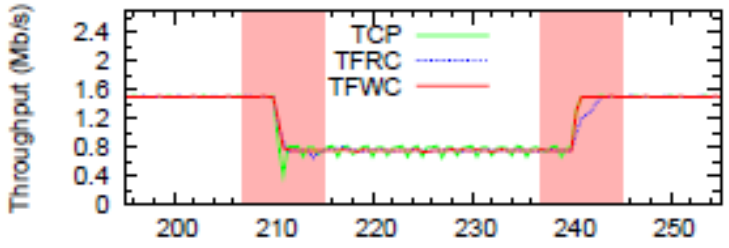


(b) TFRC Smoothness

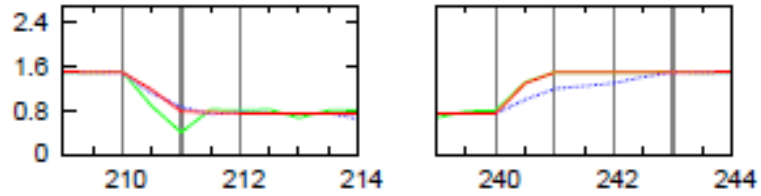


(c) TFWC Smoothness

- Responsiveness



(a) Impulse Response



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

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

Relentless Congestion Control (1)

- Packet conservation
  - when a packet is received, send a new packet (SIGCOMM 1988)

  - when packets are lost, reduce cwnd by the number of lost packets (no 1/2)

$$cwnd = cwnd - \text{loss counts}$$

$$ssthresh = cwnd - \text{loss counts}$$

  - use slow start until stable state

M.Mathis: "Relentless Congestion Control", PFLDNet 2009.

Relentless Congestion Control (2)

- Scalability
  - In classical AIMD, packet loss interval (PIR) changes according to bandwidth
  - Relentless CC tries to keep PIR to be constant independent of bandwidth (e.g. 3*RTT)
    - analogous to CUBIC-TCP
  - End-to-end ⇒ Network assist (baseline AQM: active queue management)

M.Mathis: "Relentless Congestion Control", PFLDNet 2009.

Relentless Congestion Control (3)

- Relentless CC + Baseline AQM
  - without loss, cwnd=cwnd+1
  - forces packet drop per 3*RTT (AQM)
  - update cwnd by reducing # of lost packets
  - repeat

- TCP-unfriendly paradigm

M.Mathis: "Relentless Congestion Control", PFLDNet 2009.

MPEG DASH
(Dynamic Adaptive Streaming over HTTP)

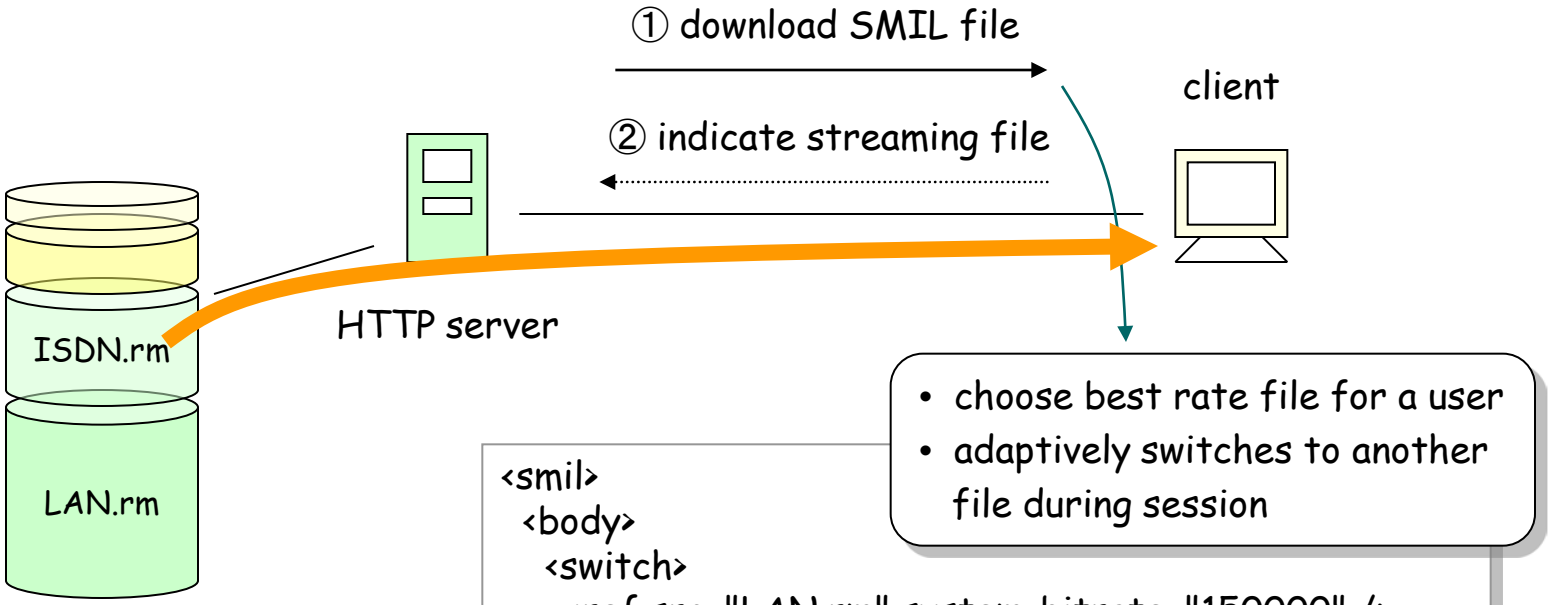
TJ. Ohm: "MPEG Developments in Multi-view Video Coding and 3D Video", Apr. 2009.

Old Comparison (before CDN's deployment)

	HTTP streaming	Proprietary streaming (RTSP, MMS etc)
method	common web server	proprietary server for streaming
pros	simple firewall (because uses well-known port 80)	can apply smart congestion control Real: SureStream, TurboPlay MS: Intelligent Streaming, FastStreaming can support live streaming and multicast
cons	cannot apply smart congestion control (depends on underlying TCP) cannot support live streaming and multicast	firewall
assumption	client is intelligent	server is intelligent

Old work in "SMIL"

- Dynamic Streaming using SMIL (1998)



multiple files encoded in different bitrates

multiple bit rates

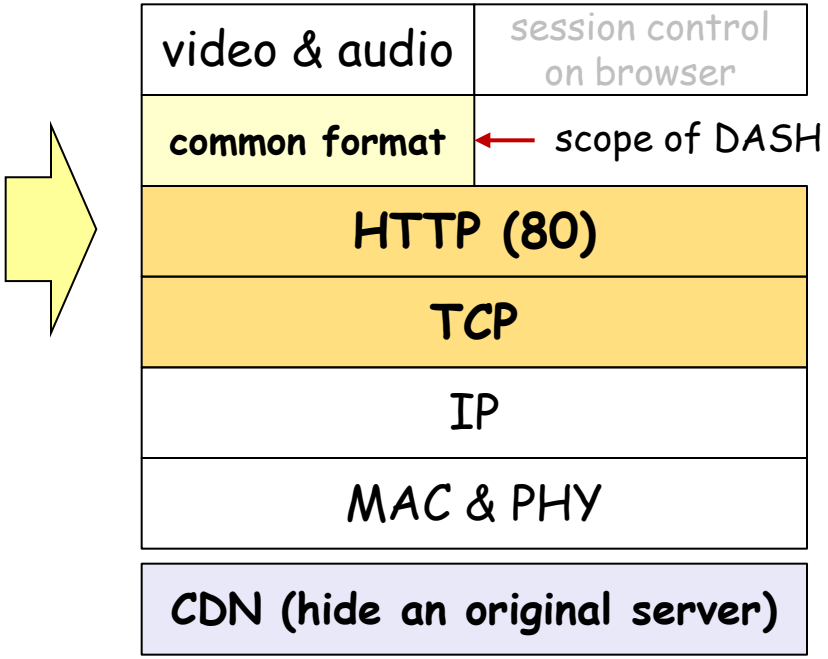
SMIL: "Synchronized Multimedia Integration Language" (1998)

Protocol stack

- RTSP Streaming

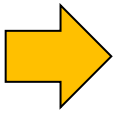
video & audio	session control
RTP/RTCP	RTSP (554)
UDP	TCP
IP	
MAC & PHY	

- HTTP Streaming (supported by CDN)



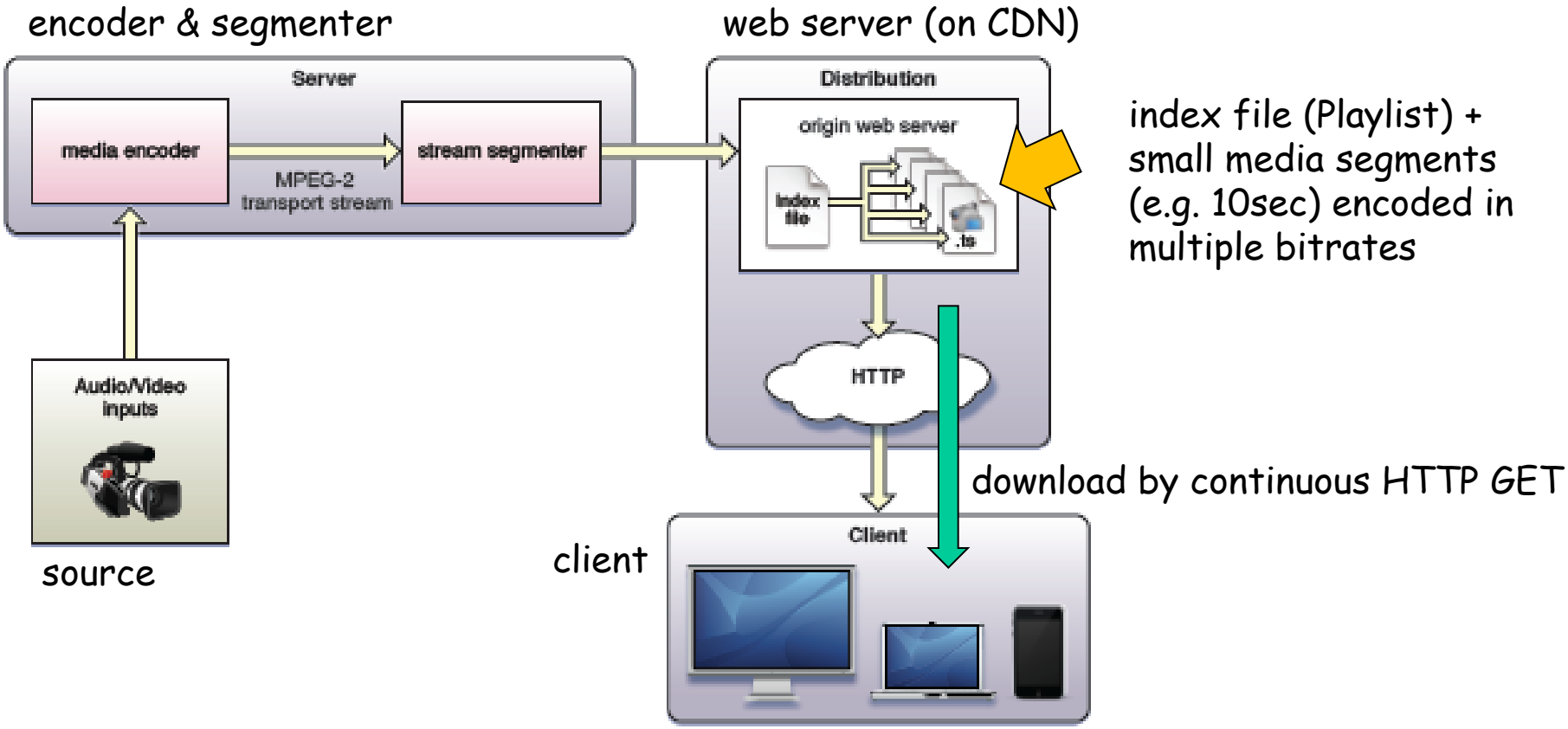
Proprietary HTTP Streaming

- Apple's HTTP Live Streaming
  - https://developer.apple.com/resources/http-streaming/
- Adobe's Dynamic HTTP Streaming
  - http://www.adobe.com/products/httpdynamicstreaming/
- Microsoft's Smooth Streaming
  - http://www.microsoft.com/silverlight/smoothstreaming/



activate international standardization in MPEG and 3GPP (2009)

Apple's HTTP Live Streaming (1)



Apple.com: StreamingMediaGuide.pdf

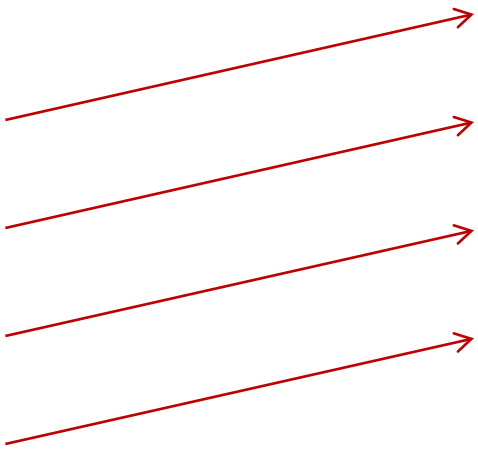
Apple's HTTP Live Streaming (2)

- Playlist at the beginning

```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:1
#EXTINF:10,
fileSequence1.ts
#EXTINF:10,
fileSequence2.ts
#EXTINF:10,
fileSequence3.ts
#EXTINF:10,
fileSequence4.ts
#EXTINF:10,
fileSequence5.ts
```

- Playlist after updating

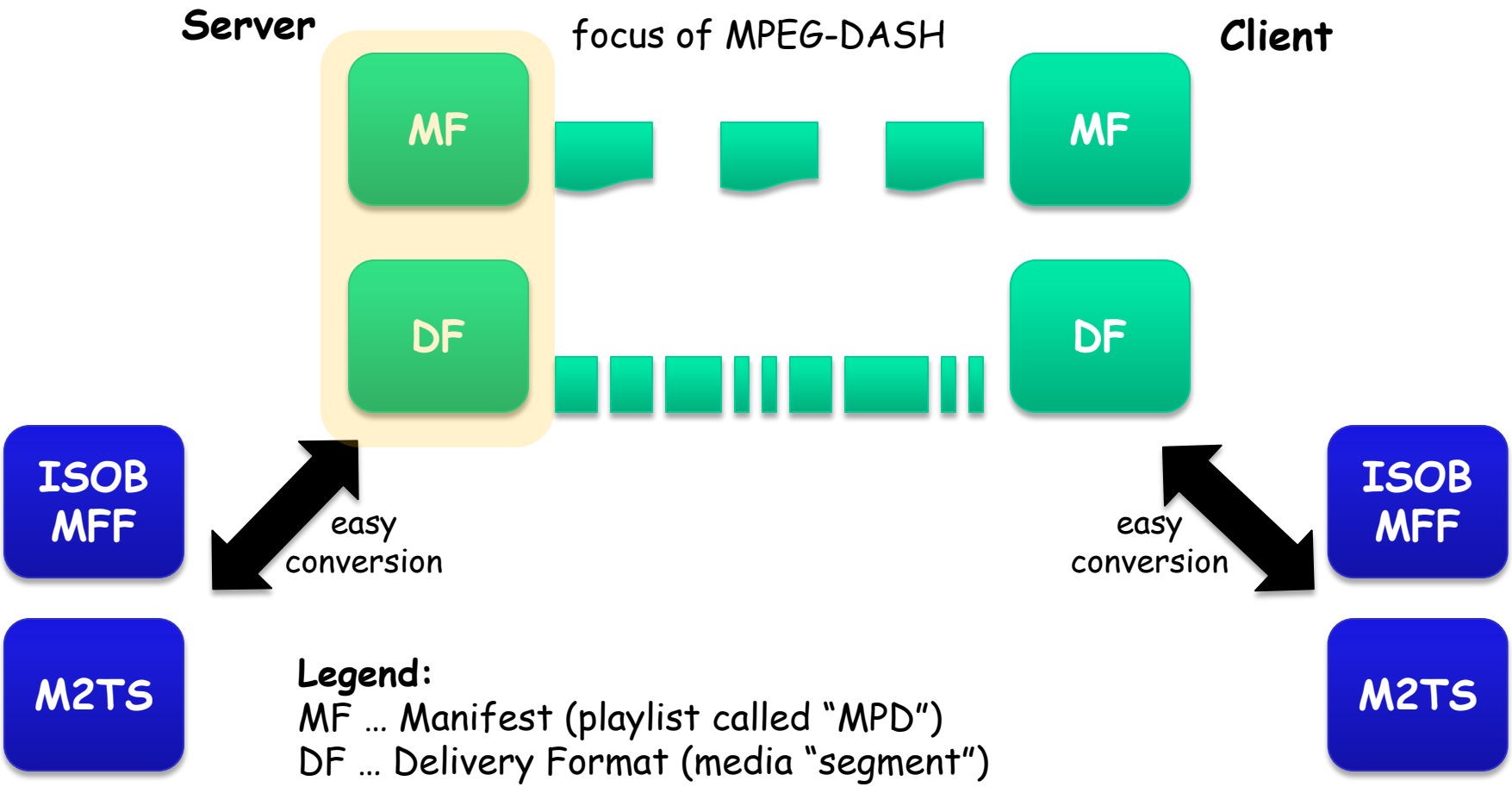
```
#EXTM3U
#EXT-X-TARGETDURATION:10
#EXT-X-VERSION:3
#EXT-X-MEDIA-SEQUENCE:2
#EXTINF:10,
fileSequence2.ts
#EXTINF:10,
fileSequence3.ts
#EXTINF:10,
fileSequence4.ts
#EXTINF:10,
fileSequence5.ts
#EXTINF:10,
fileSequence6.ts
```



*.ts: MPEG-2 Transport Stream (same as DVD)

Apple.com: StreamingMediaGuide.pdf

MPEG-DASH specification (1)

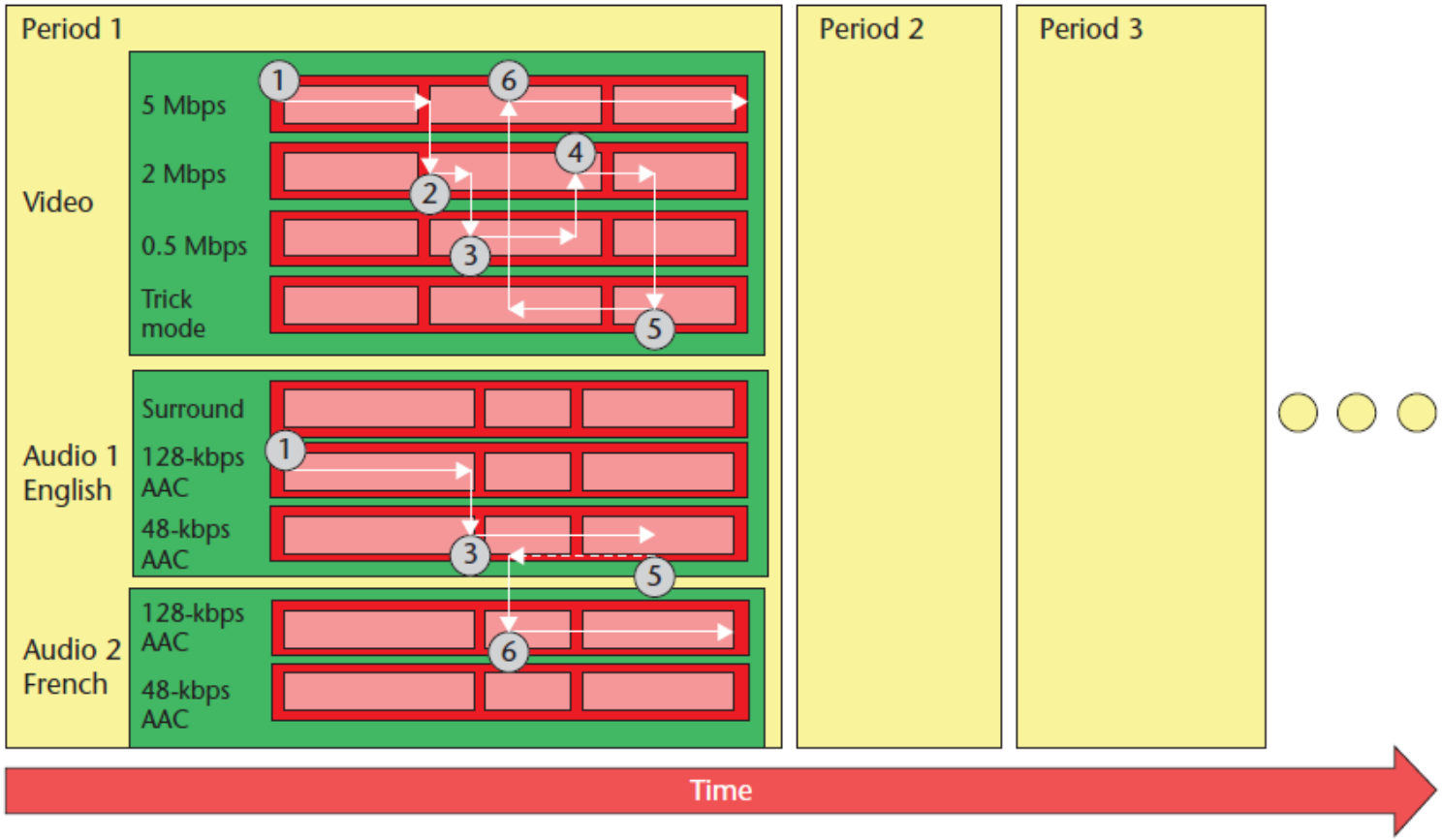


Legend:
MF … Manifest (playlist called "MPD")
DF … Delivery Format (media "segment")
ISOBMFF … File Format (MP4) specified by MPEG-4
M2TS … Transport Stream (TS) specified by MPEG-2

C.Timmerer: "HTTP Streaming of MPEG Media", Sep.2010.

MPEG-DASH specification (2)

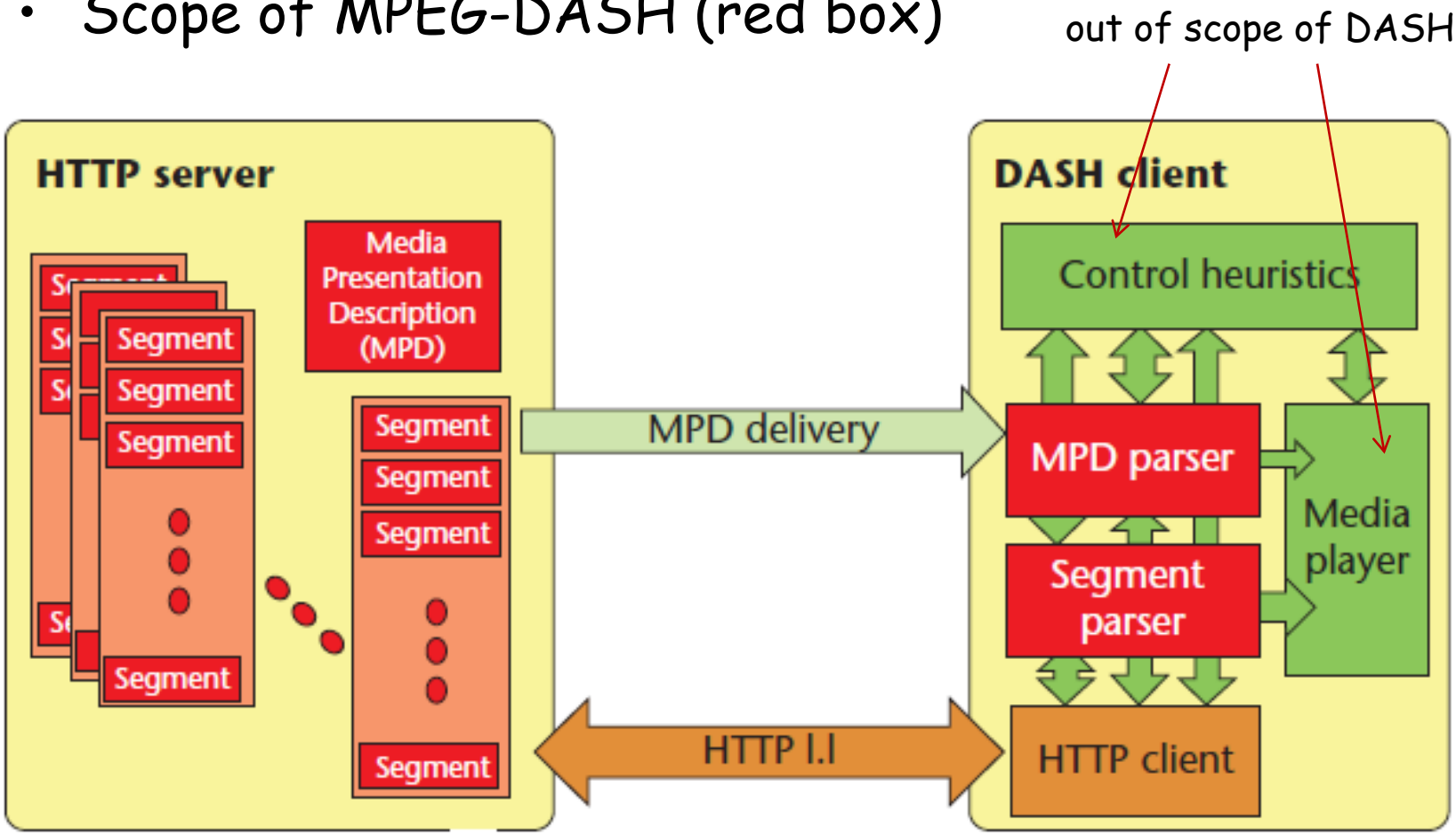
- Example of "dynamic adaptive" streaming



A.Vetro: "The MPEG-DASH Standard for Multimedia Streaming over the Internet", 2011.

MPEG-DASH specification (2)

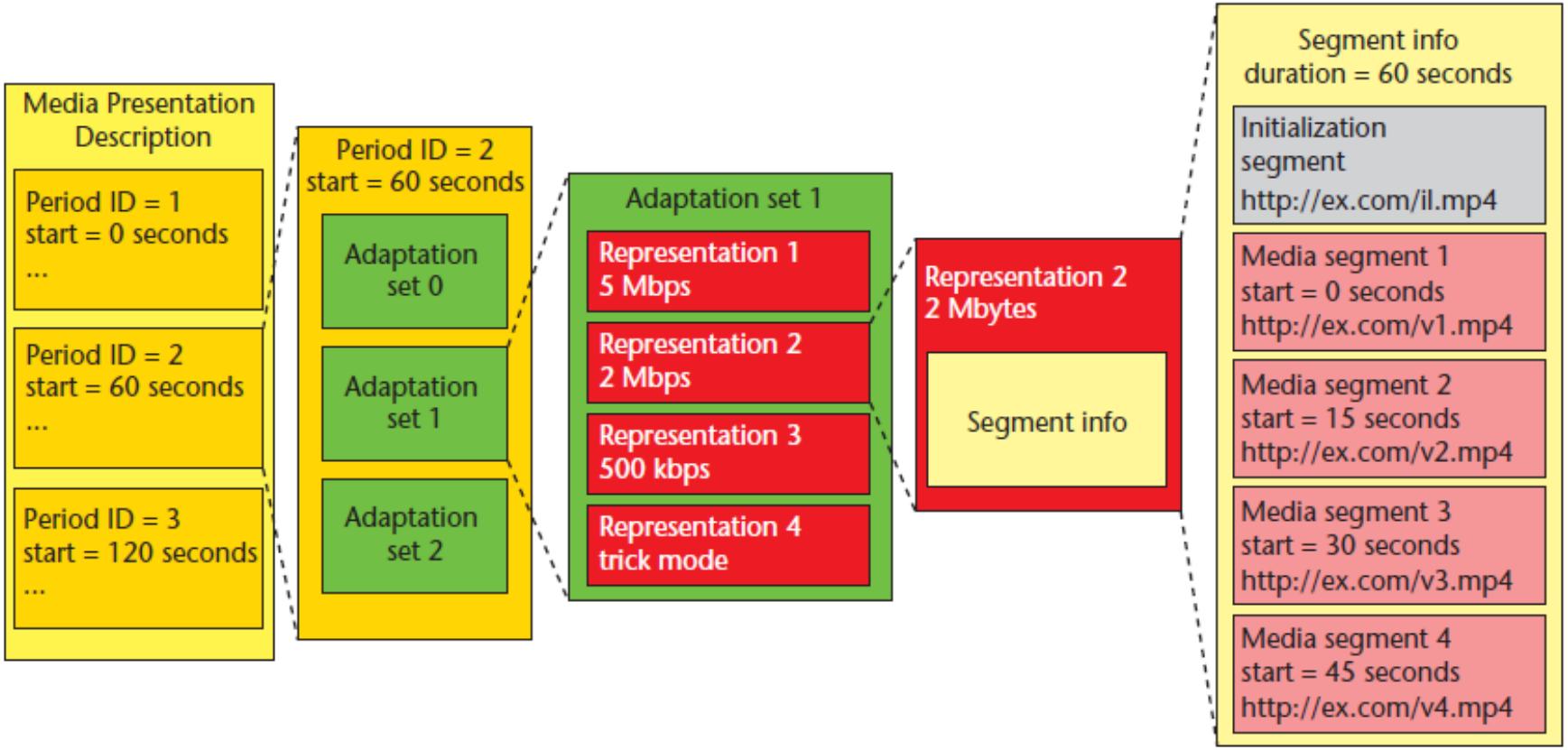
- Scope of MPEG-DASH (red box)



A.Vetro: "The MPEG-DASH Standard for Multimedia Streaming over the Internet", 2011.

MPEG-DASH specification (3)

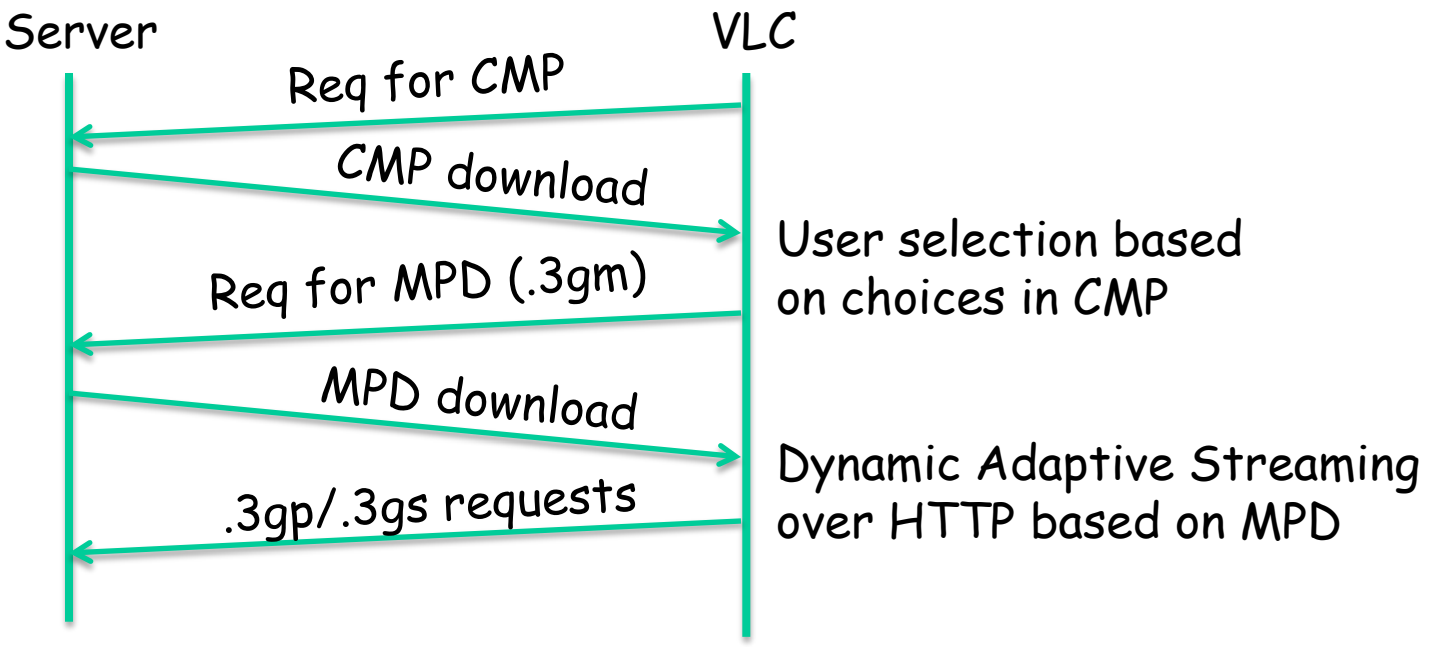
- Example of MPD (Media Presentation Description)



A.Vetro: "The MPEG-DASH Standard for Multimedia Streaming over the Internet", 2011.

MPEG-DASH implementation

- Video Lan Plugin:
  - http://www-itec.uni-klu.ac.at/dash/



C.Timmerer: "HTTP Streaming of MPEG Media", Sep.2010.

DASH's contributions and open issues

- MPEG-DASH assumes:
  - HTTP streaming over CDN
  - client based session control (rate control, trick mode, and so on)
- Open issues:
  - development of smart session control by client (network monitoring, segment request scheduling and segment switching)
  - evaluation of effect of underlying protocols

Example of Rate Control using DASH