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

TCP Variants

情報理工・情報通信専攻　甲藤二郎
E-Mail: katto@waseda.jp

Streaming Background
(last week self study)

TCP and UDP

	Reliability	Low Delay	Congestion Control	Typical Application
TCP	◎ (ACK and lost packet retransmission)	× → ○ (thanks to **CDN** & broadband network)	○ → ◎ (**TCP versions**)	One way (on-demand) streaming
UDP	× (no ACK nor sequence number)	◎ (no ACK nor packet retransmission)	× → △ (**RTP/RTCP** and **TFRC**)	Interactive (bi-directional) phone & conference

one-way streaming in 10 to 20 years ago

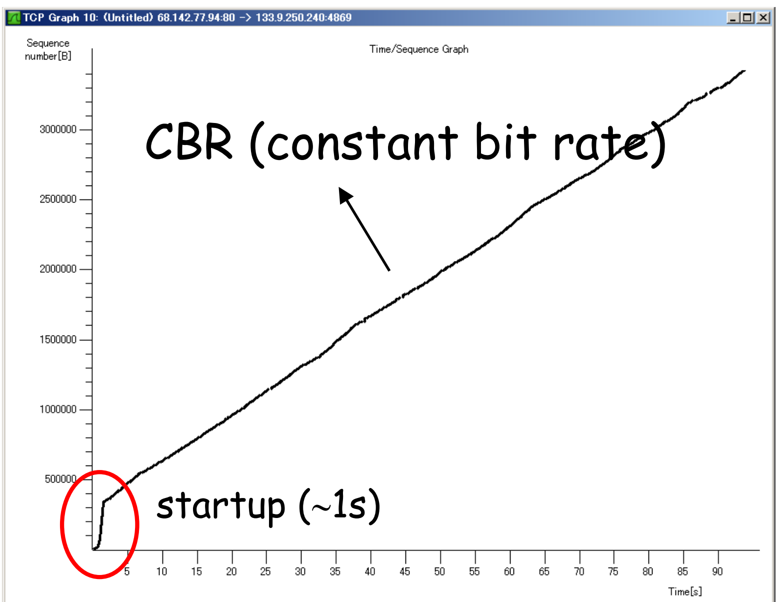
prefetching & CBR

(prefetch, then CBR)

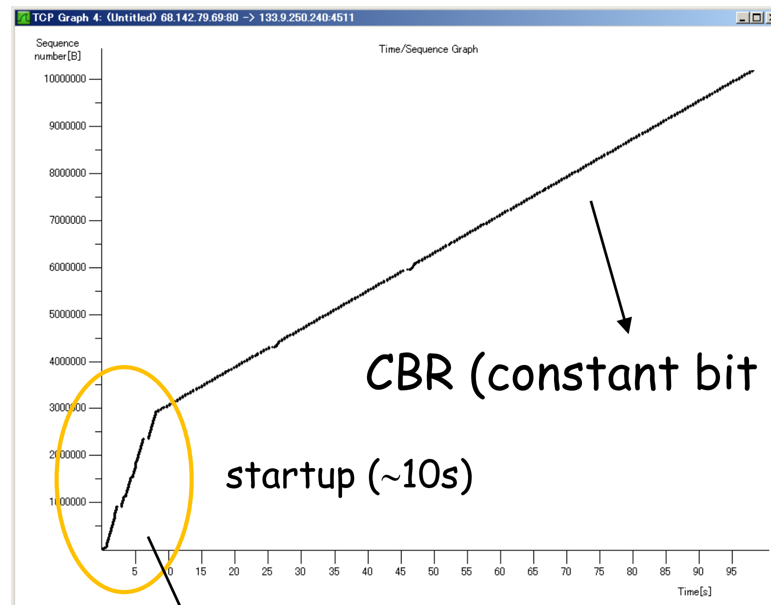
sequence number

time

Live



On-Demand



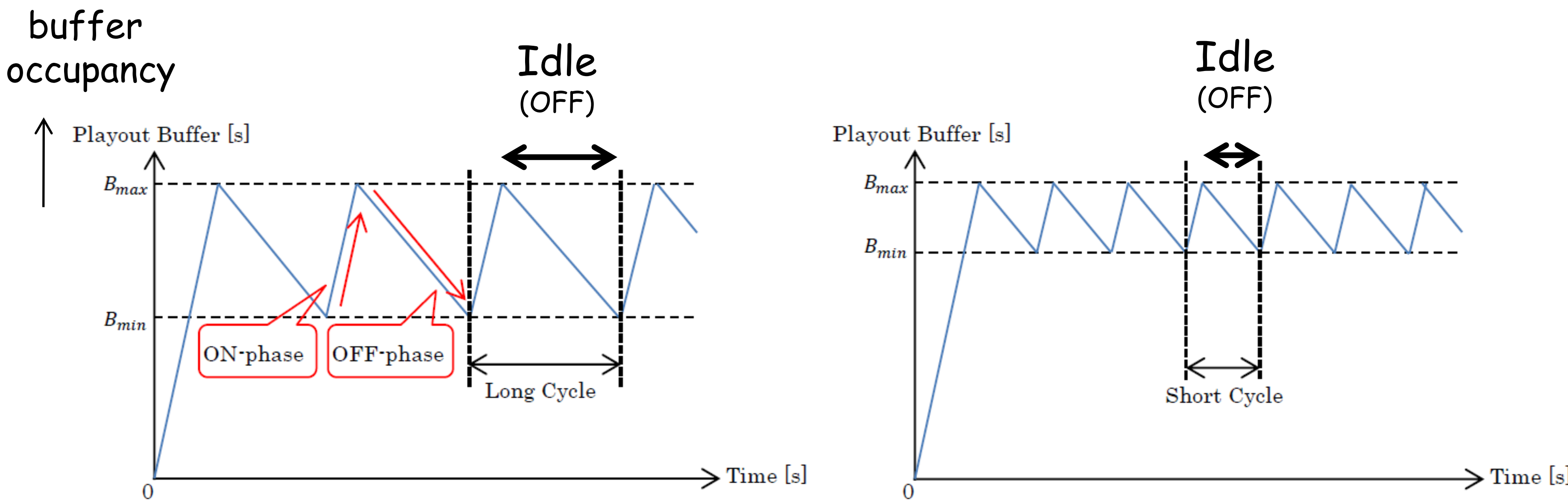
prefetching

one-way streaming nowadays

ON/OFF cycles

(prefetch & idle cycles)

- receiver buffer behaviors



(a) long ON-OFF Cycle (sawtooth)

(b) short ON-OFF Cycle (zippy pacing)

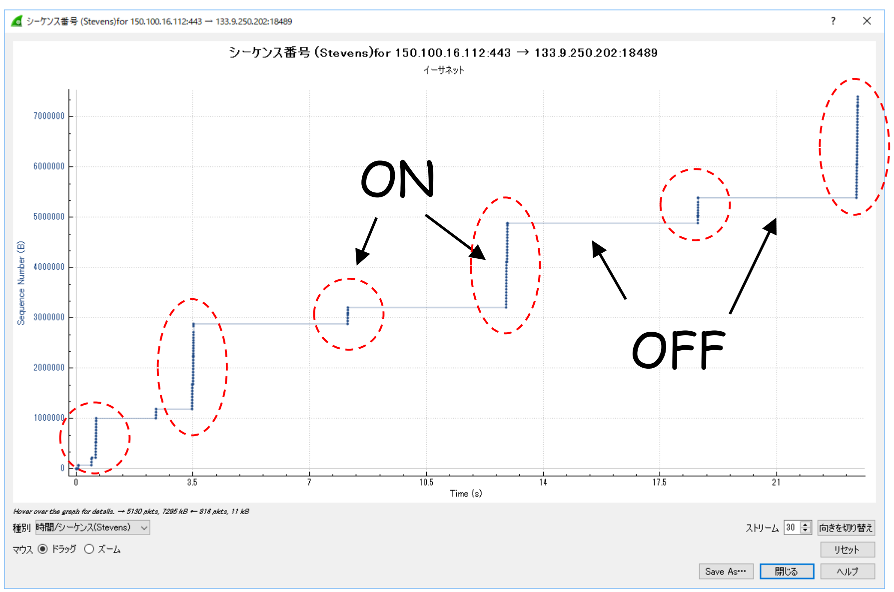
A. Rao, et al. ACM CoNEXT 2011

one-way streaming nowadays

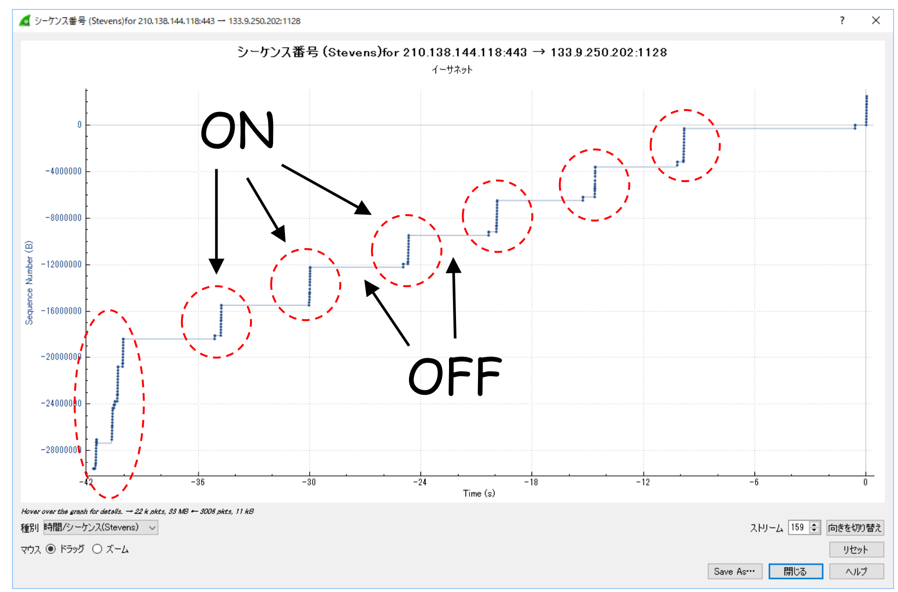
ON/OFF cycles

- sequence number behaviors

sequence number



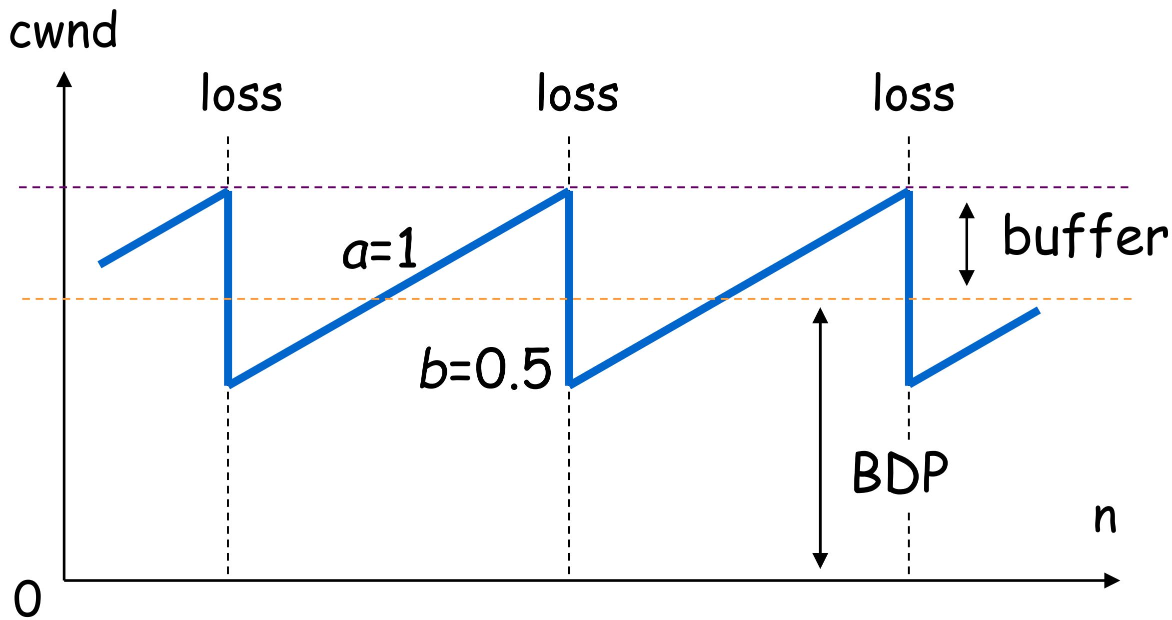
example 1 (YouTube)



example 2 (TVer)

TCP Variants

TCP-Reno (loss-based)

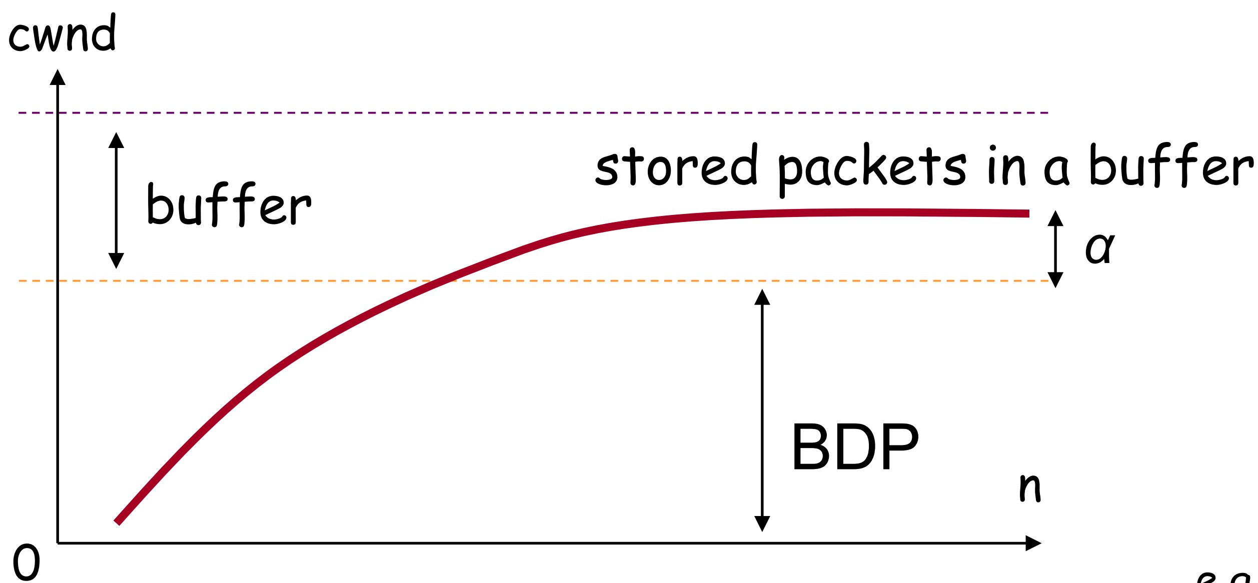


increase: cwnd = cwnd + 1/cwnd

decrease: cwnd = cwnd / 2

AIMD: additive increase multiplicative decrease

TCP-Vegas (delay-based)



e.g. α=1, β=3

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

stored packets in a buffer

increase: $$cwnd = \begin{cases} cwnd + 1 & diff < \alpha \\ cwnd & otherwise \\ cwnd - 1 & diff > \beta \end{cases}$$

decrease: $$cwnd = cwnd * 0.75$$

TCP problems, 20 years ago

- broadband wired networks
  - slow window increase (<u>inefficiency</u>)
- deployment of wireless networks
  - cannot distinguish wireless errors and buffer overflow

- TCP-Reno (NewReno, SACK) problem
  - Reno expels Vegas (<u>unfriendliness</u>)

TCP Variants in the 21th century

- **Loss-based (AIMD)**
  - TCP-Reno / NewReno / SACK
  - High-Speed TCP (IETF RFC 3649, Dec 2003)
  - Scalable TCP (PFLDnet 2003)
  - BIC-TCP / **CUBIC-TCP** (IEEE INFOCOM 2004, PFLDnet 2005) … **Linux default**
  - H-TCP (PFLDnet 2004)
  - TCP-Westwood (ACM MOBICOM 2001)
- **Delay-based (RTT Observation)**
  - TCP-Vegas (IEEE JSAC, Oct 1995)
  - FAST-TCP (INFOCOM 2004)
- **Hybrid (of loss and delay modes)**
  - Gentle High-Speed TCP (PfHSN 2003)
  - TCP-Africa (IEEE INFOCOM 2005)
  - **Compound TCP** (PFLDnet 2006) … **Windows (proposed by MSR)**
  - Adaptive Reno (PFLDnet 2006)
  - YeAH-TCP (PFLDnet 2007)
  - TCP-Fusion (PFLDnet 2007) … our lab

+ TCP-BBR (2017 by Google)

Loss-based TCPs

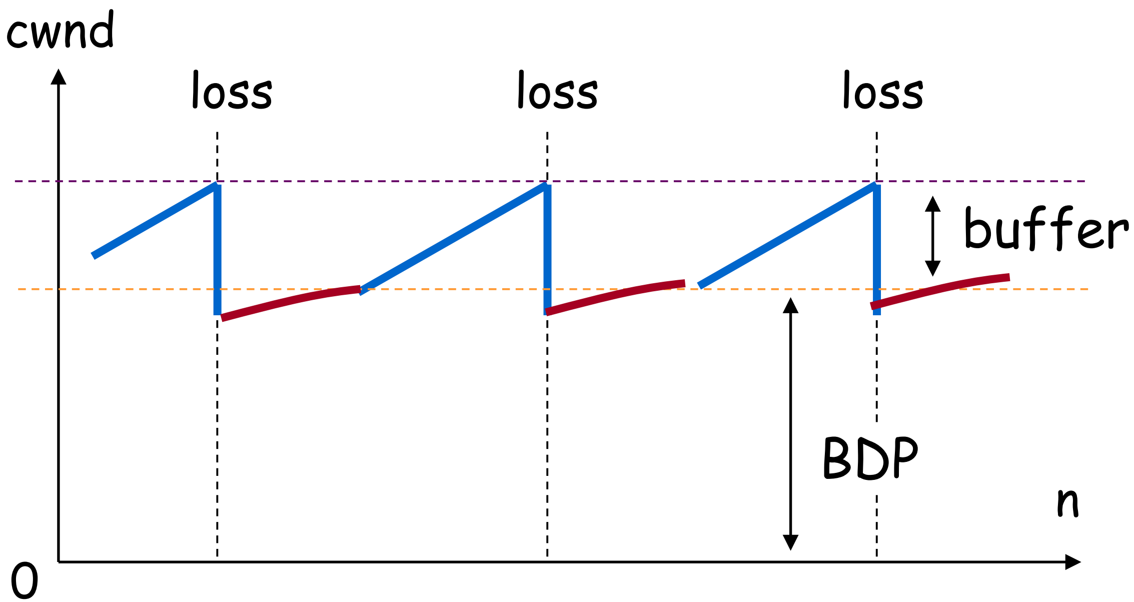
	Variants	Increase / Update (a)	Decrease (b)
	TCP-Reno	1	0.5
aggressive	HighSpeed TCP (HS-TCP)	$a(w)=\frac{2w^2\cdot b(w)\cdot p(w)}{2-b(w)}$ e.g. 70 (10Gbps, 100ms)	$b(w)=\frac{\log(w)-\log(W_{low})}{\log(W_{high})-\log(W_{low})}(b_{high}-0.5)+0.5$ e.g. 0.1 (10Gbps, 100ms)
aggressive	Scalable TCP (STCP)	0.01 (per every ACK)	0.875
adaptive	BIC-TCP	$\begin{cases} additive\ increase\ (fast) \\ binary\ search\ (slow) \\ max\ probing\ (fast) \end{cases}$	0.875
adaptive	CUBIC-TCP	$w=0.4(t-\sqrt[3]{2W_{max}})^3+W_{max}$	0.8
adaptive	H-TCP	$\alpha \leftarrow 2(1-\beta)\{1+10.5\cdot(t-TH)\}$	$\beta \leftarrow \begin{cases} 0.5 & for\ \left\lvert\frac{B(k+1)-B(k)}{B(k)}\right\rvert > 2 \\ \frac{RTT_{min}}{RTT_{max}} & otherwise \end{cases}$
adaptive	TCP-Westwood (TCPW)	1	$\begin{cases} RE * RTT_{min} / PS & (not\ congested) \\ BE * RTT_{min} / PS & (congested) \end{cases}$

Delay-based TCPs

a b

Variants	Update	Decrease
TCP-Vegas	$w \leftarrow \begin{cases} w+1 & (no\ congestion) \\ w & (stable) \\ w-1 & (early\ congestion) \end{cases}$	0.75
FAST-TCP	$w \leftarrow \min\left\{2w, (1-\gamma)w + \gamma\left(\frac{RTT_{\min}}{RTT} w + \alpha\right)\right\}$	0.5 (?)

Hybrid TCP



- RTT increase: loss mode ⇒ improvement of friendliness
- no RTT increase: delay mode ⇒ improvement of efficiency

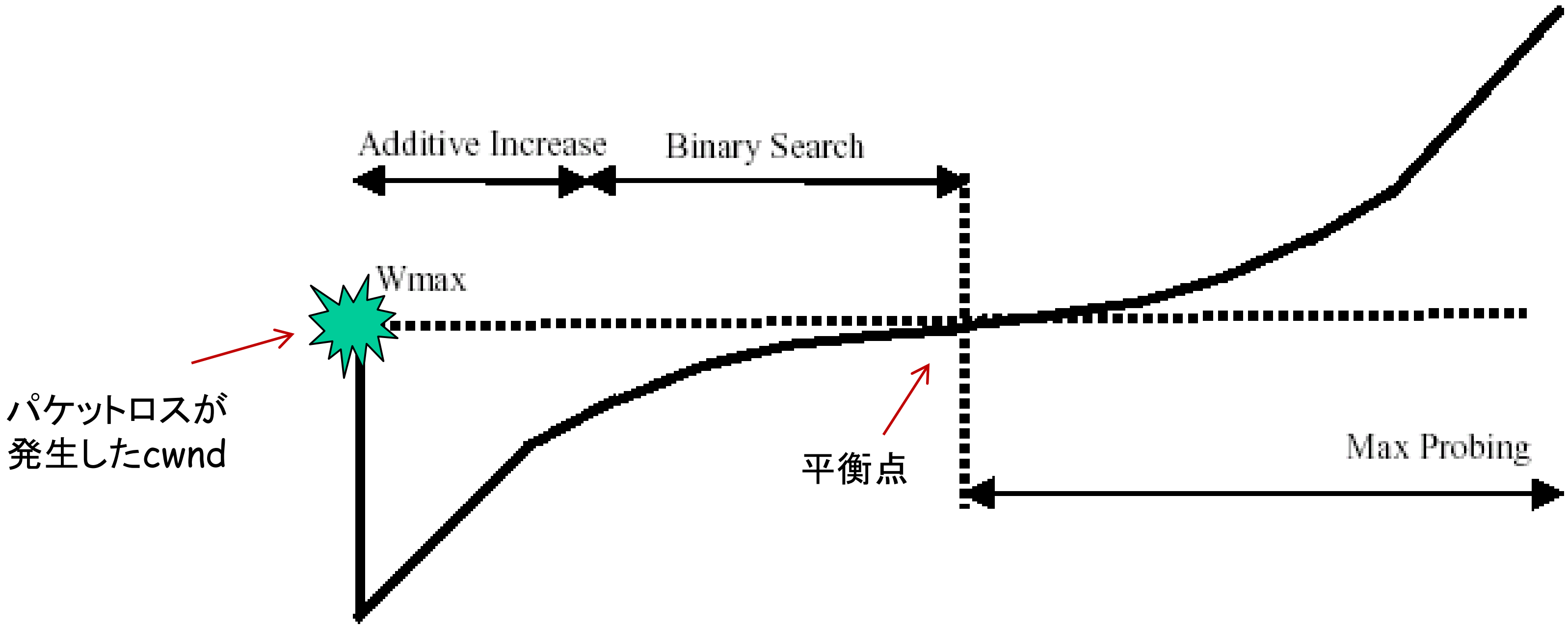
Hybrid TCPs

	Variants	Increase (a)	Decrease (b)
simple	Gentle HS-TCP	HS-TCP / Reno	HS-TCP
simple	TCP-Africa	HS-TCP / Reno	HS-TCP
adaptive	Compound TCP (CTCP)	$0.125 \cdot cwnd^{0.75}$ / Reno	0.5
adaptive	Adaptive Reno (ARENO)	$B/10\text{Mbps}$ / Reno	$\begin{cases} 1 & (non\ congestion\ loss) \\ 0.5 & (congestion\ loss) \end{cases}$
adaptive	YeAH-TCP	STCP / Reno	$\max\left(\frac{RTT_{\min}}{RTT}, 0.5\right)$
adaptive	TCP-Fusion	$\frac{B * D_{\min}}{PS}$ / Reno	$\max\left(\frac{RTT_{\min}}{RTT}, 0.5\right)$

CUBIC-TCP
(Linux default)

BIC-TCP (1)

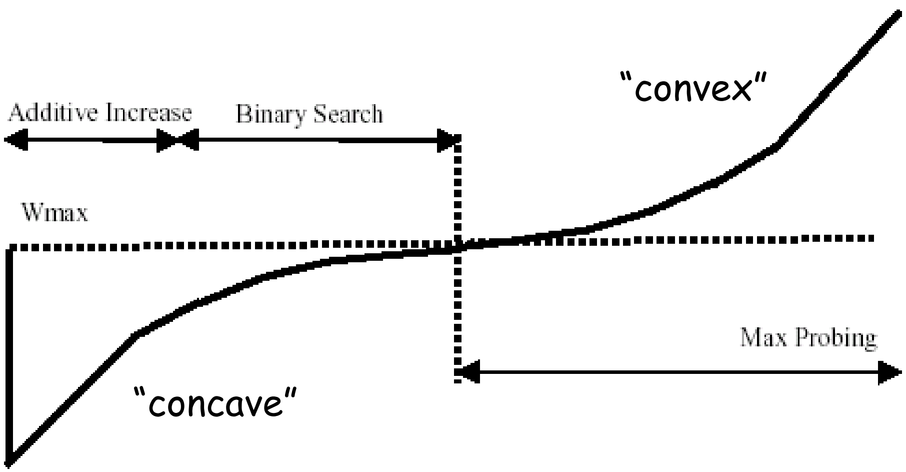
- Binary Increase Congestion Control



L.Xu et al: "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," IEEE INFOCOM 2004.

BIC-TCP (2)

- Window Increase



binary search

```
if (cwnd < Wmax )
    Winc = (Wmax – cwnd) / 2;
else
    Winc = (cwnd - Wmax) / 2;

if (Winc > Smax)
    Winc = Smax;
elseif (Winc < Smin)
    Winc = Smin;

cwnd = cwnd + Winc / cwnd;
```

additive increase
(linear increase)

Wmax: cwnd when a last loss happened

Smax: maximum increase rate (e.g. 32)

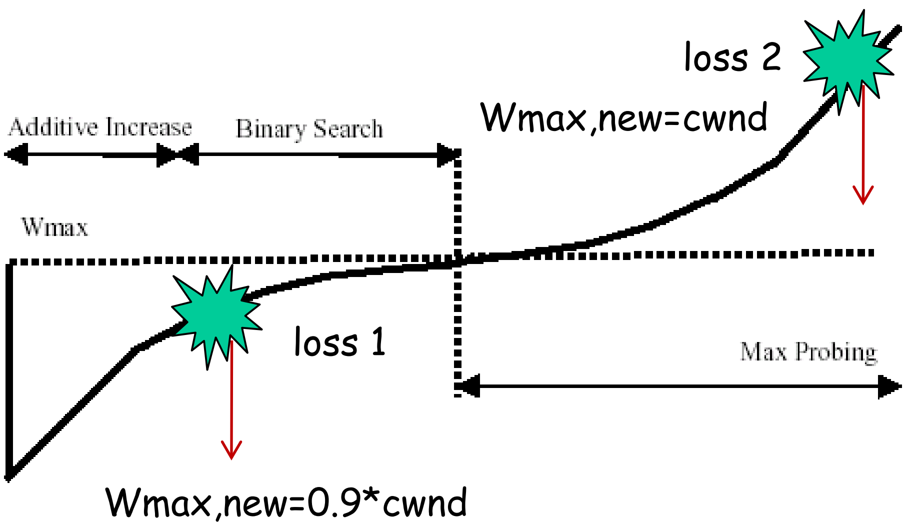
Smin: minimum increase rate (e.g. 0.01)

L.Xu et al: "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," IEEE INFOCOM 2004.

BIC-TCP (3)

- Window Decrease

Wmax=target cwnd update




```
if (cwnd < Wmax )
    Wmax,new = cwnd * (2-β) / 2;
else
    Wmax,new = cwnd;

cwnd = cwnd * (1- β);
```

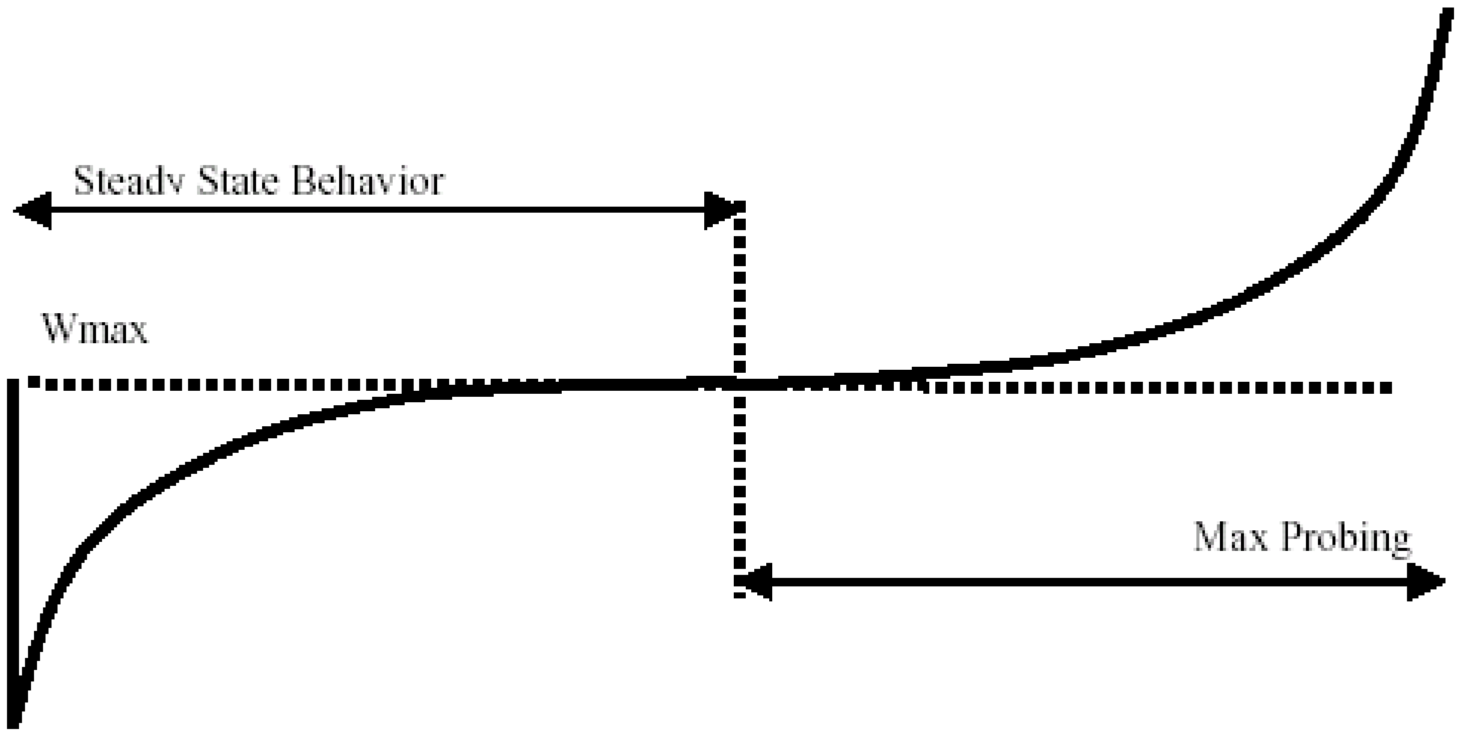
β: decrease rate (e.g. 0.2)

*0.9: give bandwidth to newly-coming flows
... "Fast Convergence"

L.Xu et al: "Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks," IEEE INFOCOM 2004.

CUBIC-TCP (1)

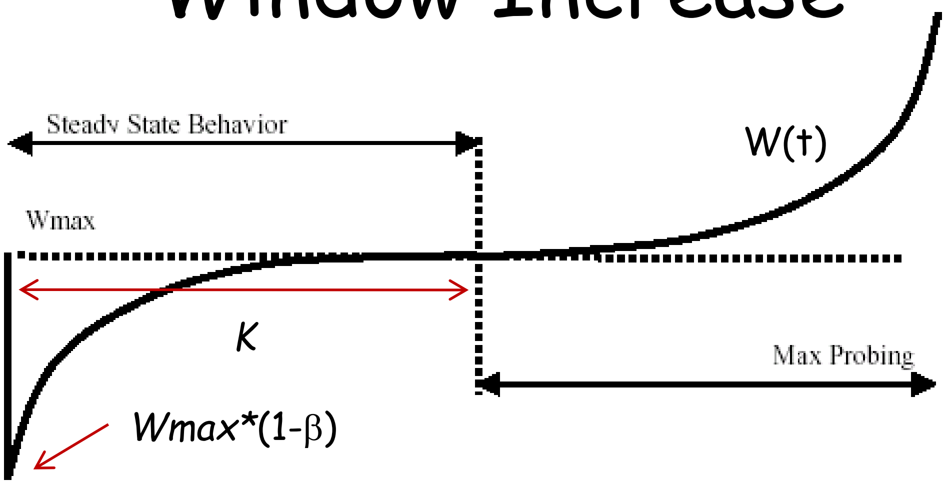
- Cubic approximation of BIC-TCP



S.Ha et al: "CUBIC: A New TCP Friendly HighSpeed TCP Variant", ACM SIGOPS Review, 2008.

CUBIC-TCP (2)

- Window Increase




```
/* cubic function */
Winc = W(t+RTT) – cwnd;

cwnd = cwnd + Winc / cwnd;

/* TCP mode */
if ( Wtcp > cwnd )
   cwnd = Wtcp;
```

$$W(t) = C * (t - K)^3 + W_{max}$$

$$K = \sqrt[3]{\frac{W_{max}\beta}{C}}$$

equivalent to Reno

$$W_{tcp}(t) = W_{max}(1-\beta) + 3\frac{\beta}{2-\beta}\frac{t}{RTT}$$

※ window decrease is the same as BIC

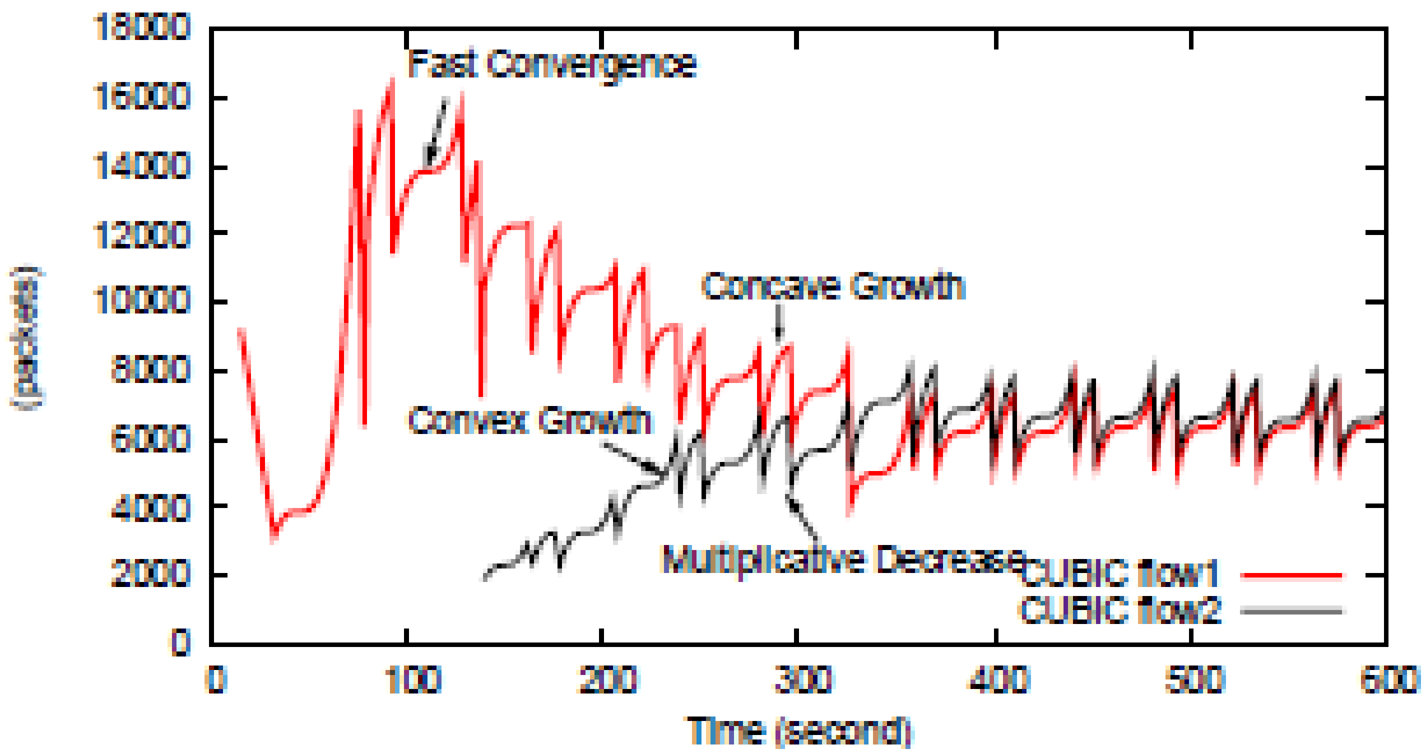
β: decrease rate (e.g. 0.2)

C: constant (e.g. 0.4)

S.Ha et al: "CUBIC: A New TCP Friendly HighSpeed TCP Variant", ACM SIGOPS Review, 2008.

CUBIC-TCP (3)

- CUBIC's cwnd behavior



S.Ha et al: "CUBIC: A New TCP Friendly HighSpeed TCP Variant", ACM SIGOPS Review, 2008.

CUBIC-TCP (4)

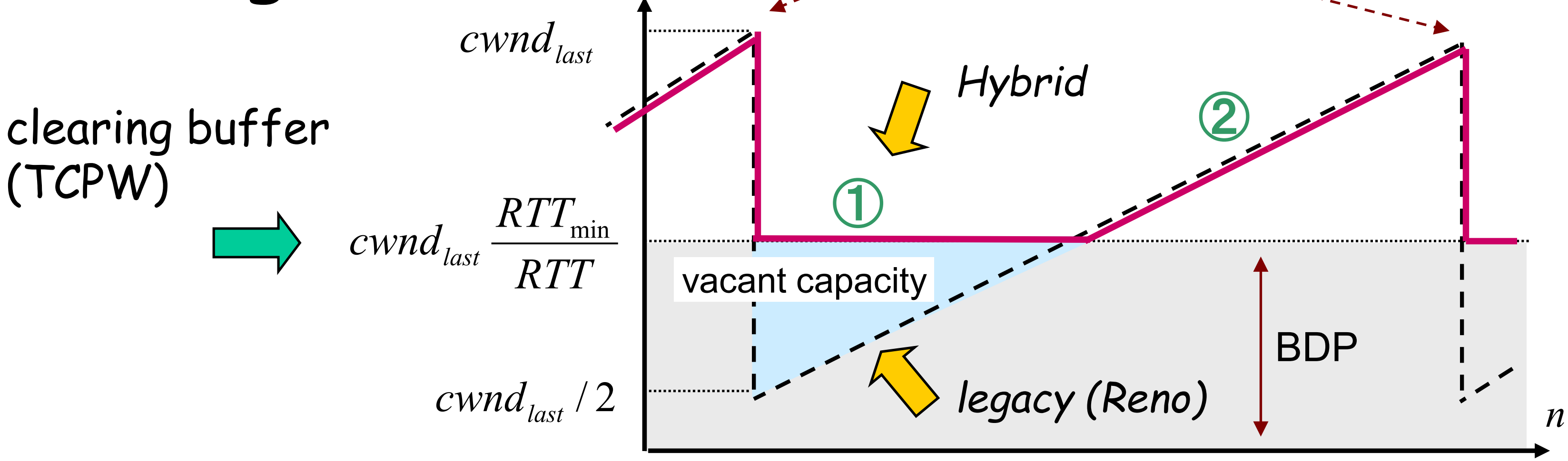
- Advantages
  - stability
  - "intra-protocol fairness" among multiple CUBIC flows
- Disadvantages
  - heavy buffer occupancy and delay increase (⇔ delay-based)
  - "inter-protocol unfairness" against other TCP flows
    - "Linux beats Windows!" (vs. Compound TCP)

K.Munir et al: "Linux beats Windows! or the Worrying Evolution of TCP...", PFLDNet 2007.

Hybrid TCPs

Hybrid TCP (1)

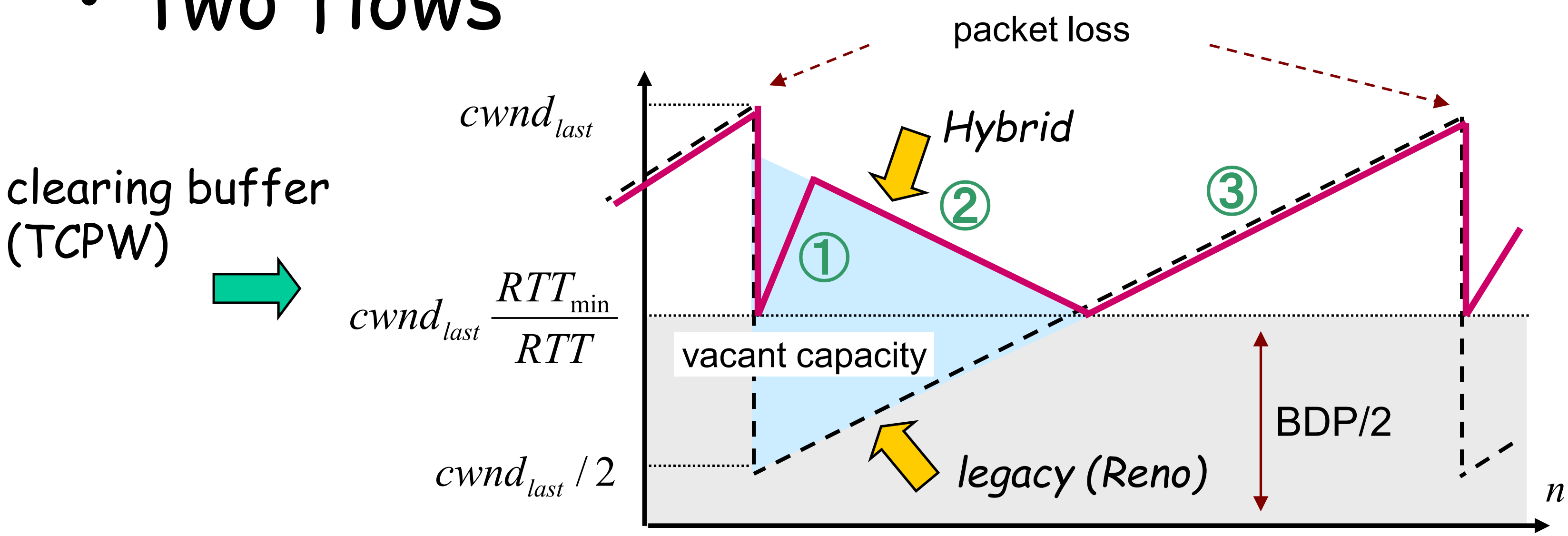
- single flow



adaptive switching of two modes (loss & delay):
① constant rate until RTT increases (delay mode) : “efficiency” and “low delay”
② performs as Reno when RTT increases (loss mode) : “friendliness”

Hybrid TCP (2)

- two flows



adaptive switching of two modes (loss & delay):

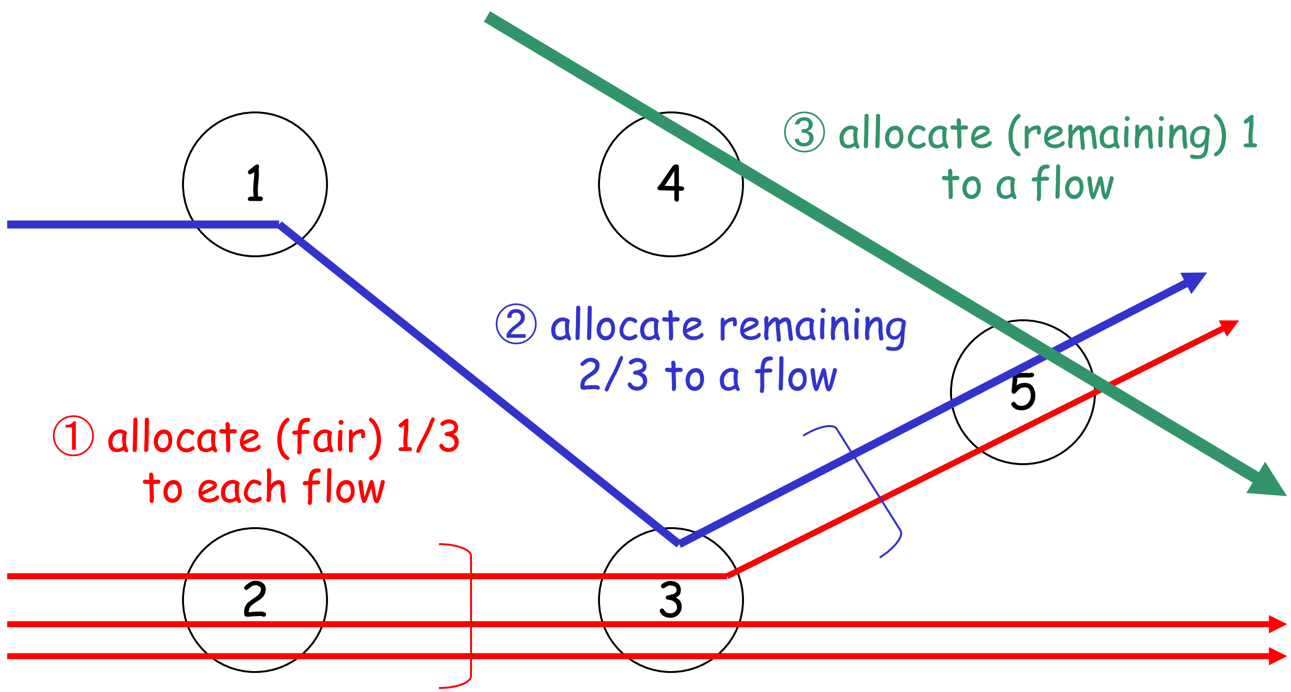
① fast cwnd increase (delay ... "efficiency")

② mild cwnd decrease (delay ... congestion avoidance)

③ performs as Reno when RTT increases (loss ... "friendliness")

Min-Max Fair (ideal case)

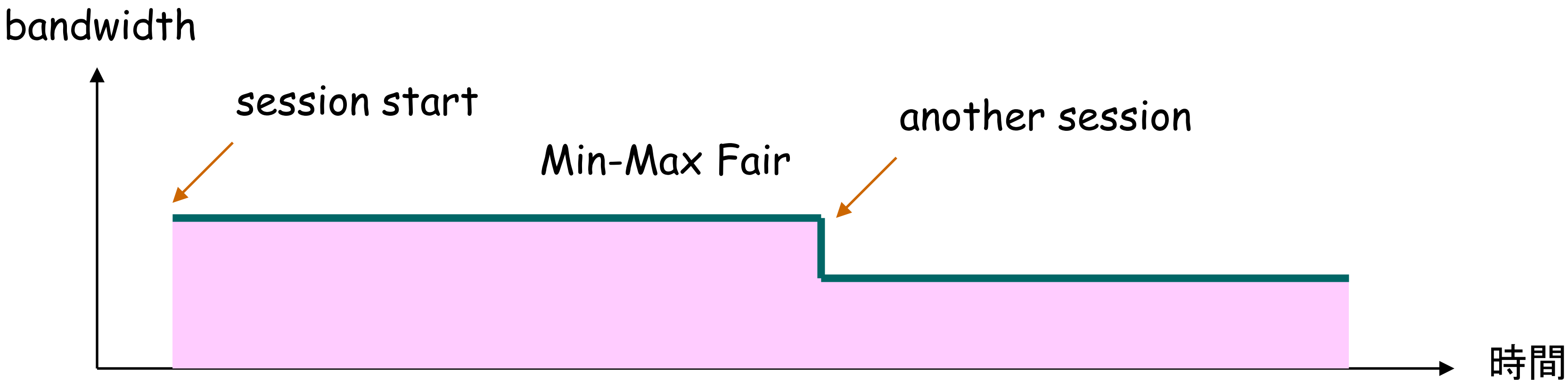
• Min-Max-Fair: allocate "maximum bandwidth" to a user who has "minimum bandwidth"



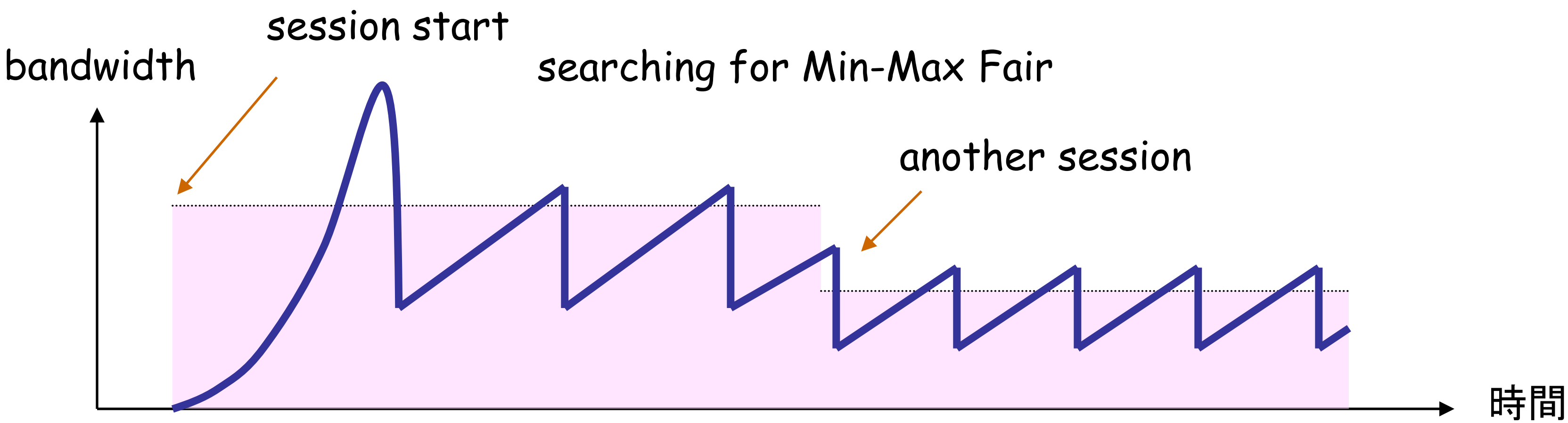
D.Bertsekas and R.Gallager: "Data Networks," Prentice Hall.

TCP's objective

Ideal:



TCP Reno



TCP behavior model (1)

- model definition
  - Loss-mode (TCP-Reno) :
    - cwnd += 1 (per "RTT round")
    - cwnd *= 1/2 (when a packet loss is detected)
  - Delay-mode :
    - fill a "pipe" (fully utilize a link) without causing RTT increase
  - Hybrid :
    - works in delay mode when RTT is not increased
    - works in loss mode when RTT is increases (i.e. when packets are buffered)
    - mode selection: cwnd = max($cwnd_{loss}$, $cwnd_{delay}$)

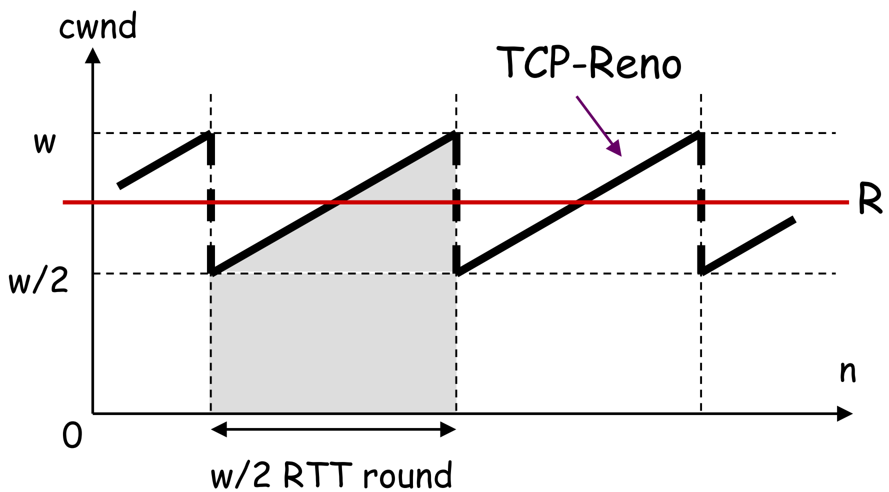
TCP behavior model (2)

- parameter definition
  - w : cwnd when a packet loss is detected
  - W : cwnd which just fills a pipe ~ BDP
  - p : packet loss rate
- assumption
  - packet loss due to buffer overflow is equivalent to packet loss due to random error

$$p = \frac{8}{3w^2}$$ (in case of TCP-Reno)

TCP behavior model (3)

- TCP friendly model



w: cwnd when a packet loss is detected
p: packet loss rate
RTT: round trip time

R: TCP equivalent rate

# of transmitted packets until a packet loss is detected

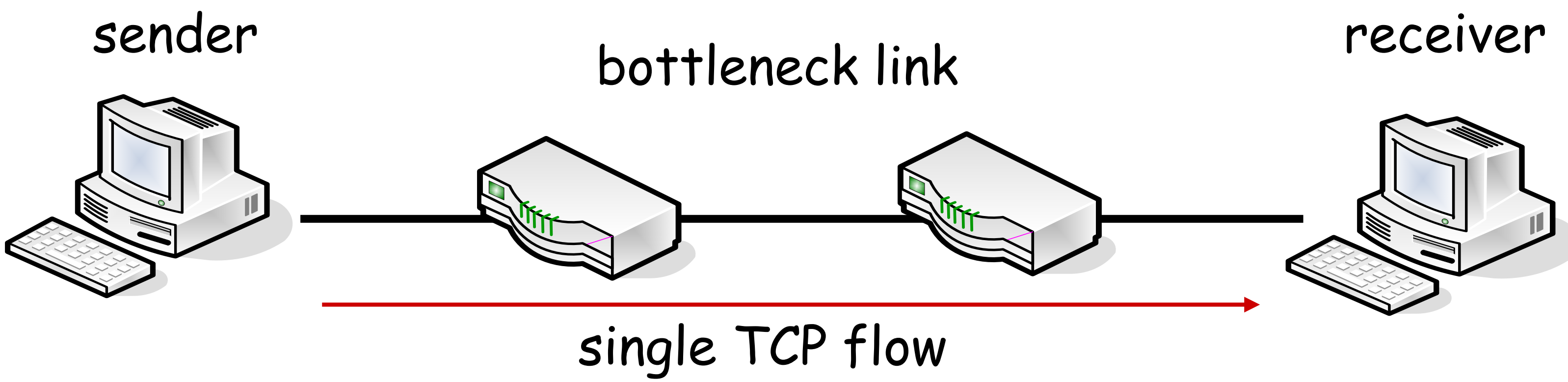
= area of a trapezoid

$$\frac{1}{2}\cdot\left(\frac{w}{2}+w\right)\cdot\frac{w}{2}=\frac{3w^2}{8}$$

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

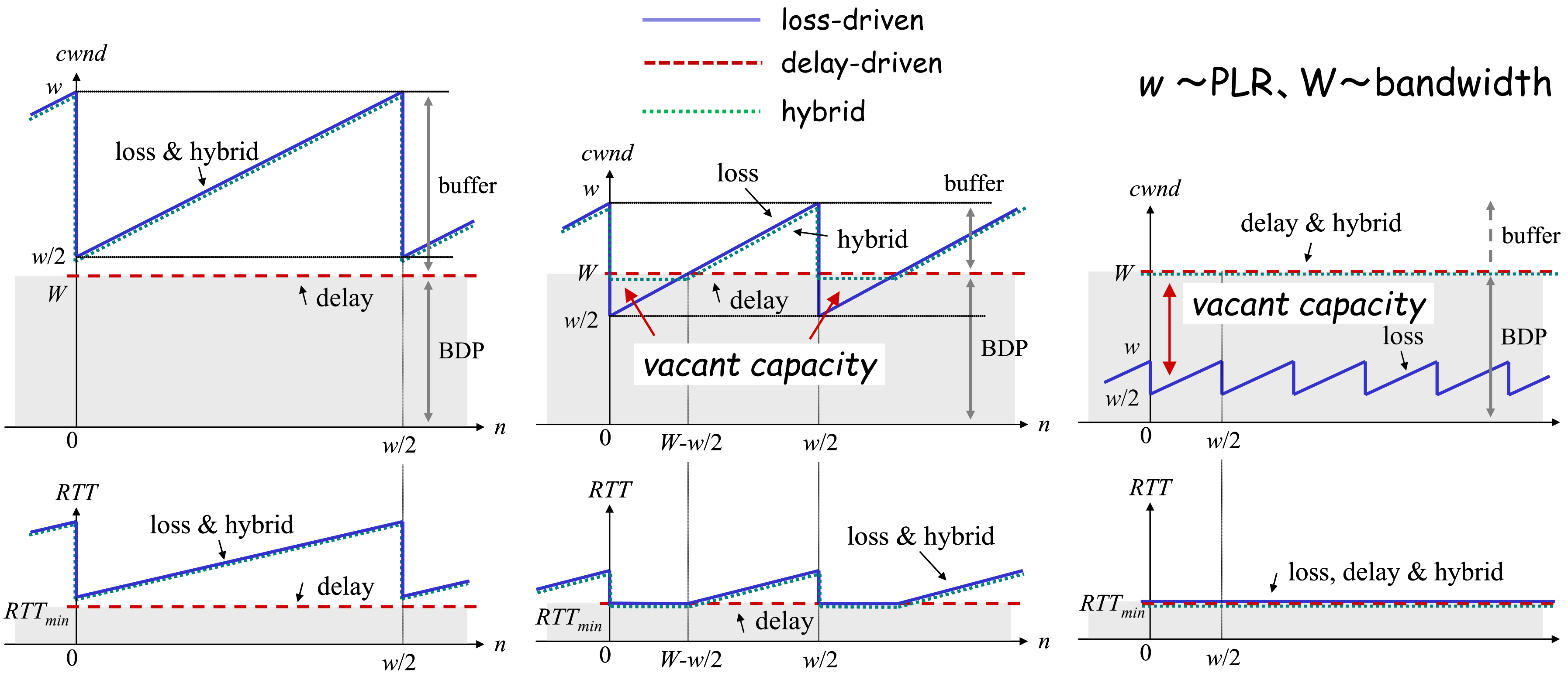
TCP behavior model (4)

- single flow



TCP behavior model (5)

- cwnd & RTT behaviors of ideal models (single flow case)



(i) $W < w/2$

large buffer, small PLR
(always loss-mode)

(ii) $w/2 < W < w$

small buffer, medium PLR
(delay ↔ loss)

(iii) $w < W$

large PLR, always vacant
(always delay-mode)

TCP behavior model (6)

- formulation

TCP	CA round	(i) $W < w/2$	(ii) $w/2 \le W < w$	(iii) $w \le W$
Loss	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$
	elapsed time	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{8}(3w^2 - 4wW) \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{2}(w - W)^2 \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min}$
Delay	transmitted packets	$\frac{1}{2}w \cdot W$	$\frac{1}{2}w \cdot W$	$\frac{1}{2}w \cdot W$
	elapsed time	$\frac{1}{2}w \cdot RTT_{\min}$	$\frac{1}{2}w \cdot RTT_{\min}$	$\frac{1}{2}w \cdot RTT_{\min}$
Hybrid	transmitted packets	$\frac{3}{8}w^2$	$\frac{1}{2}w \cdot W + \frac{1}{2}(w - W)^2$	$\frac{1}{2}w \cdot W$
	elapsed time	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{8}(3w^2 - 4wW) \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{2}(w - W)^2 \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min}$

PS: Packet size, B: Link bandwidth

TCP behavior model (7)

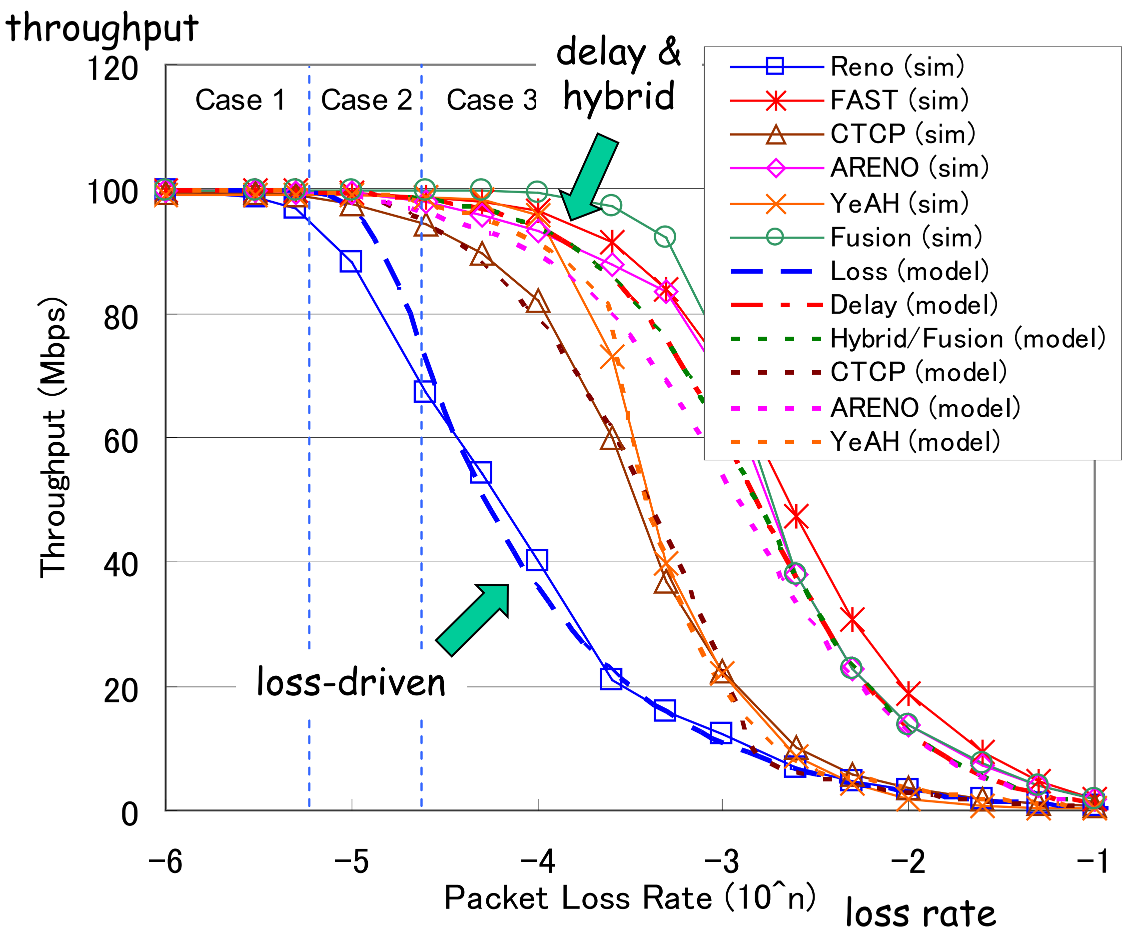
- abstraction of actual hybrids

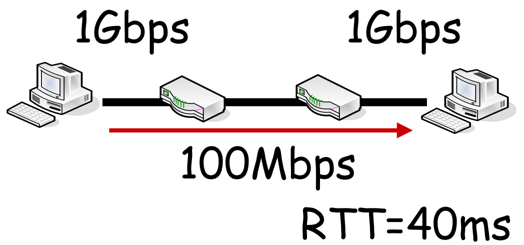
Hybrids	Window increase	Window decrease
Compound TCP	$0.125 * cwnd^{0.75}$	1/2
ARENO	B/10Mbps	1/2～1
YeAH-TCP	Scalable TCP (1.01)	1/2, RTT_{min}/RTT, 7/8
TCP-Fusion	$B * D_{min}/(N * PS)$	RTT_{min}/RTT

D_{min}: timer resolution, N: # of flows

TCP behavior model (8)

- evaluation by models and simulations





buffer size = BDP (constant)

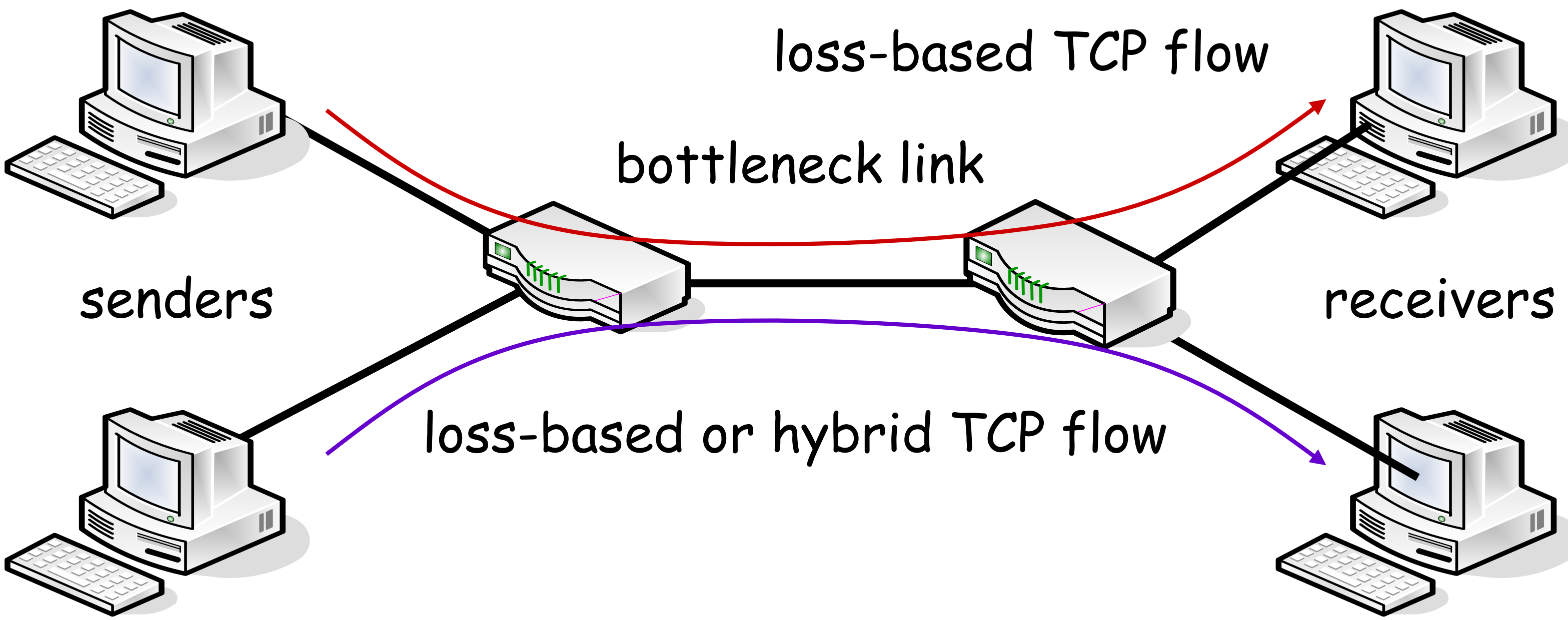
Packet loss rate : variable

when PLR is large (w/2<W), throughputs of delay & hybrid are much larger than that of loss-mode (i.e. <u>efficiency</u>)

degradation of Compound & YeAH is due to fixed window decrease

TCP behavior model (9)

- two flows (competing)

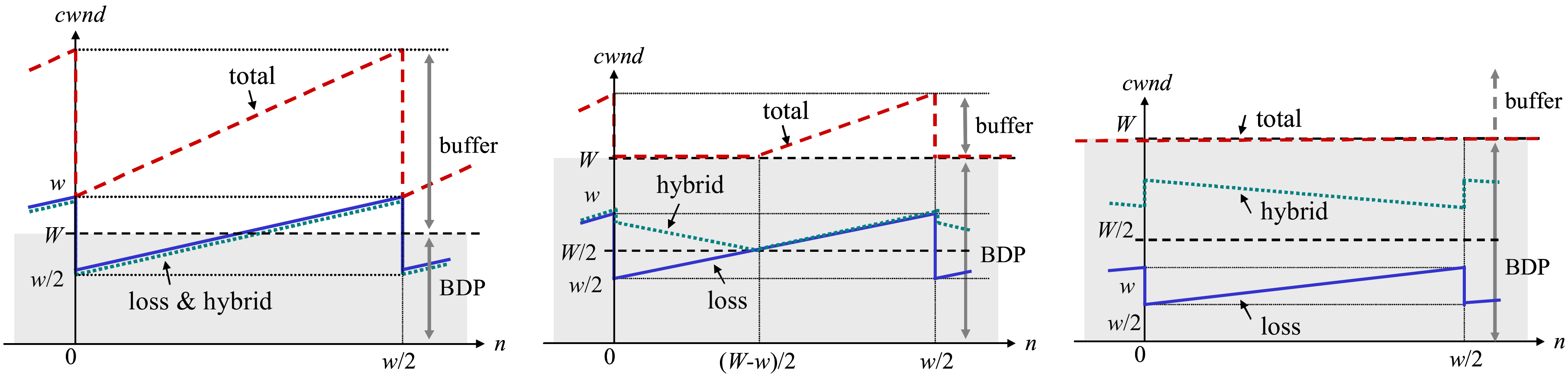


TCP behavior model (10)

- cwnd behavior of ideal models (two flow case)

loss-driven
hybrid
total (loss + hybrid)

w ~PLR、W~bandwidth



(i) $W < w$ (low PLR) | (ii) $w < W < 2*w$ (medium PLR) | (iii) $2*w < W$ (high PLR)

always buffered
(loss mode)

vacant → buffered
(delay → loss)

always vacant
(delay mode)

large buffer, small PLR

small buffer, medium PLR

large PLR, always vacant

TCP behavior model (11)

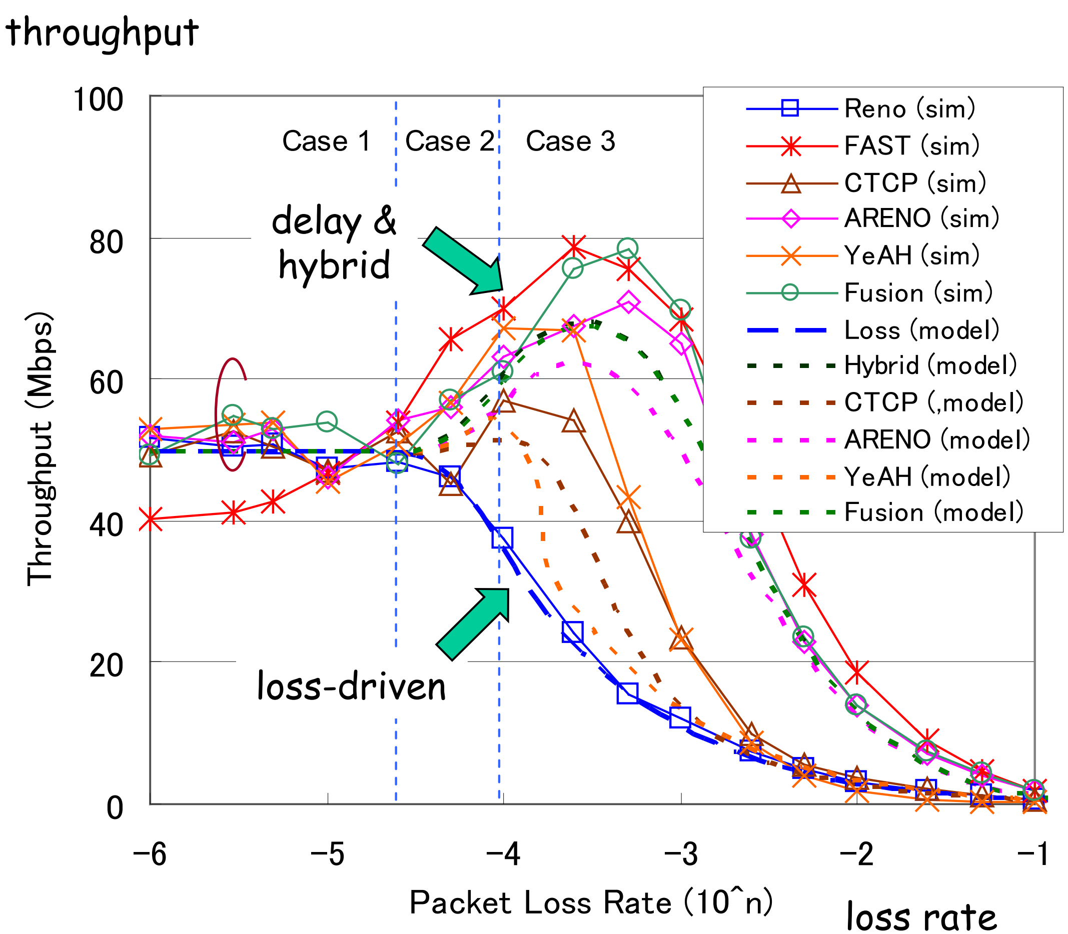
- formulation

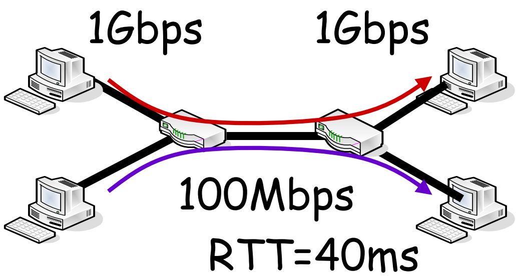
TCP	CA round	(i) $W < w$	(ii) $w \le W < 2w$	(iii) $2w \le W$
Loss	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$
Hybrid	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2 + \frac{1}{4}(W-w)^2$	$\frac{1}{2}w \cdot W - \frac{3}{8}w^2$
(common)	elapsed time	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{4}w(3w-2W) \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min} + \frac{1}{4}(2w-W)^2 \cdot \frac{PS}{B}$	$\frac{1}{2}w \cdot RTT_{\min}$

PS: Packet size, B: Link bandwidth

TCP behavior model (12)

- evaluation by models and simulations





buffer size = BDP (constant)

Packet loss rate : variable

when PLR is large (w<W), throughputs of delay & hybrid are much larger than that of loss-mode (**efficiency**)

when PLR is low (w>W), hybrid behaves similar to loss-mode (**friendliness**)

TCP behavior model (13)

- Advantages of Hybrid TCP
  - when vacant capacity exists (or PLR is large), throughput efficiency is greatly improved (advantage of delay-mode)
  - when no vacant capacity exists (or buffer size is large), friendliness to legacy TCP (i.e. Reno) is achieved (advantage of loss-mode)
- Disadvantages of Hybrid TCP
  - when buffer size is large, delay-mode is never activated …

Summary of Hybrid TCP

- "Efficiency", "Friendliness" and "Low delay"
  - can be applied to real-time streaming and large file download
  - might be effective in wireless networks
  - friendliness to CUBIC-TCP or Compound-TCP
    - CUBIC-TCP: Linux default
    - Compound-TCP: Windows
  - other metrics
    - RTT fairness, mice/elephant (short-lived or long-lived), convergence speed, etc…

  - efficiency is brought by delay-mode

Summary

Summary of TCP versions

- CUBIC-TCP provides "efficiency", but tends to increase latency because router buffers are filled up
- Compound-TCP provides "low delay" thanks to its delay mode, but suffers from unfriendliness against CUBIC-TCP

- Some community discusses redesign of TCP