画像情報特論 (2)
Advanced Image Information (2)

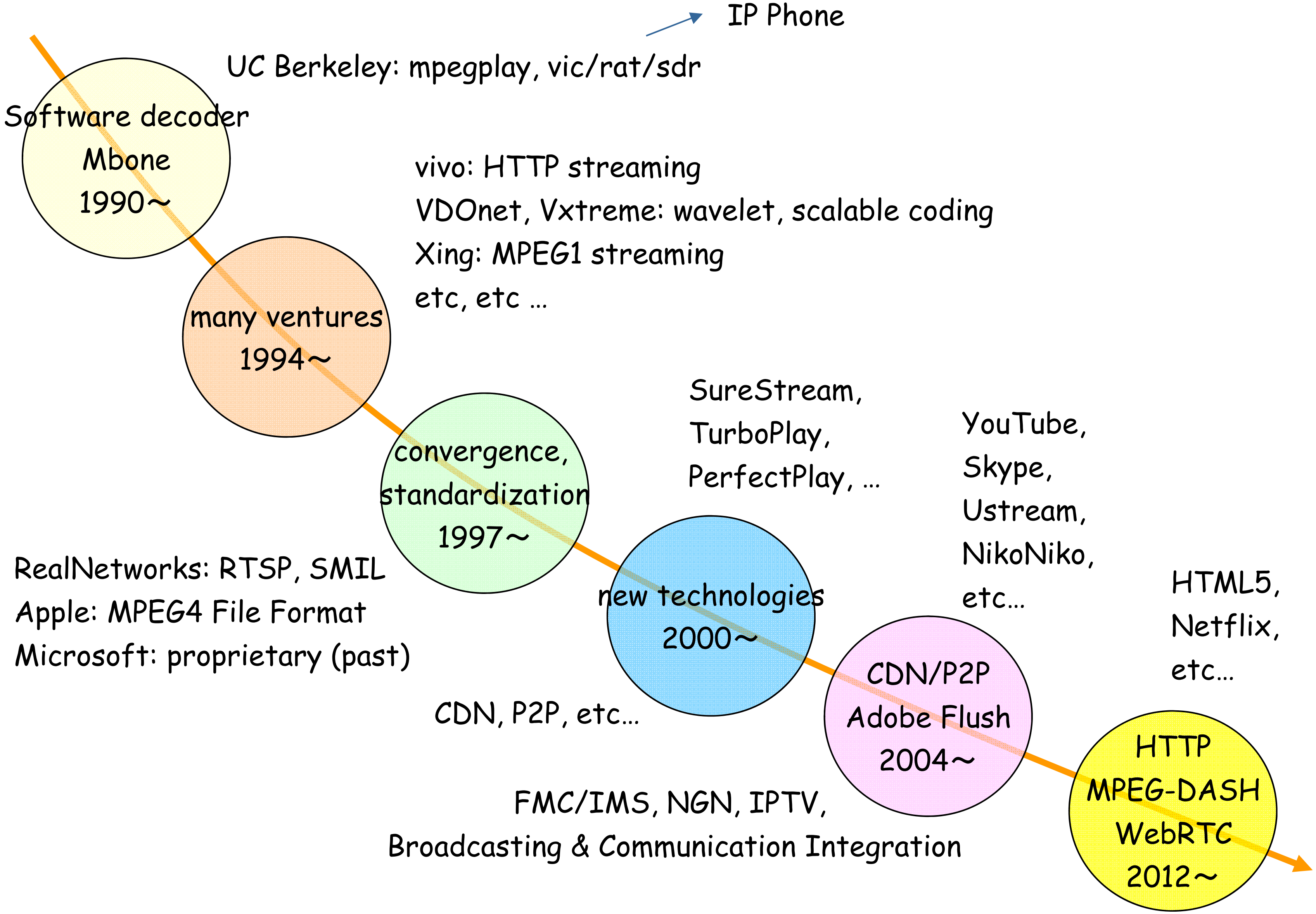
Streaming & TCP Variants

情報理工学専攻　甲藤二郎

E-Mail: katto@waseda.jp

Background

History of Streaming



(Old) Protocol Stack for Video Streaming

VoIP, IPTV and streaming shares almost common protocol stack

application (L7)	video (H.264 etc...)	audio	SDP	layout (HTML. SMIL)
adaptation	RTP / RTCP		RTSP, SIP, SAP*	HTTP
transport (L4)	UDP / TCP / DCCP		TCP / UDP / SCTP	
network (L3)	IP (IPv4, IPv6, IP-multicast)			
datalink & physical (L2 & L1)	actual networks (802.3 (ethernet), 802.11 (WiFi), etc)			

* SAP: delivered by IP-multicast for program advertisement

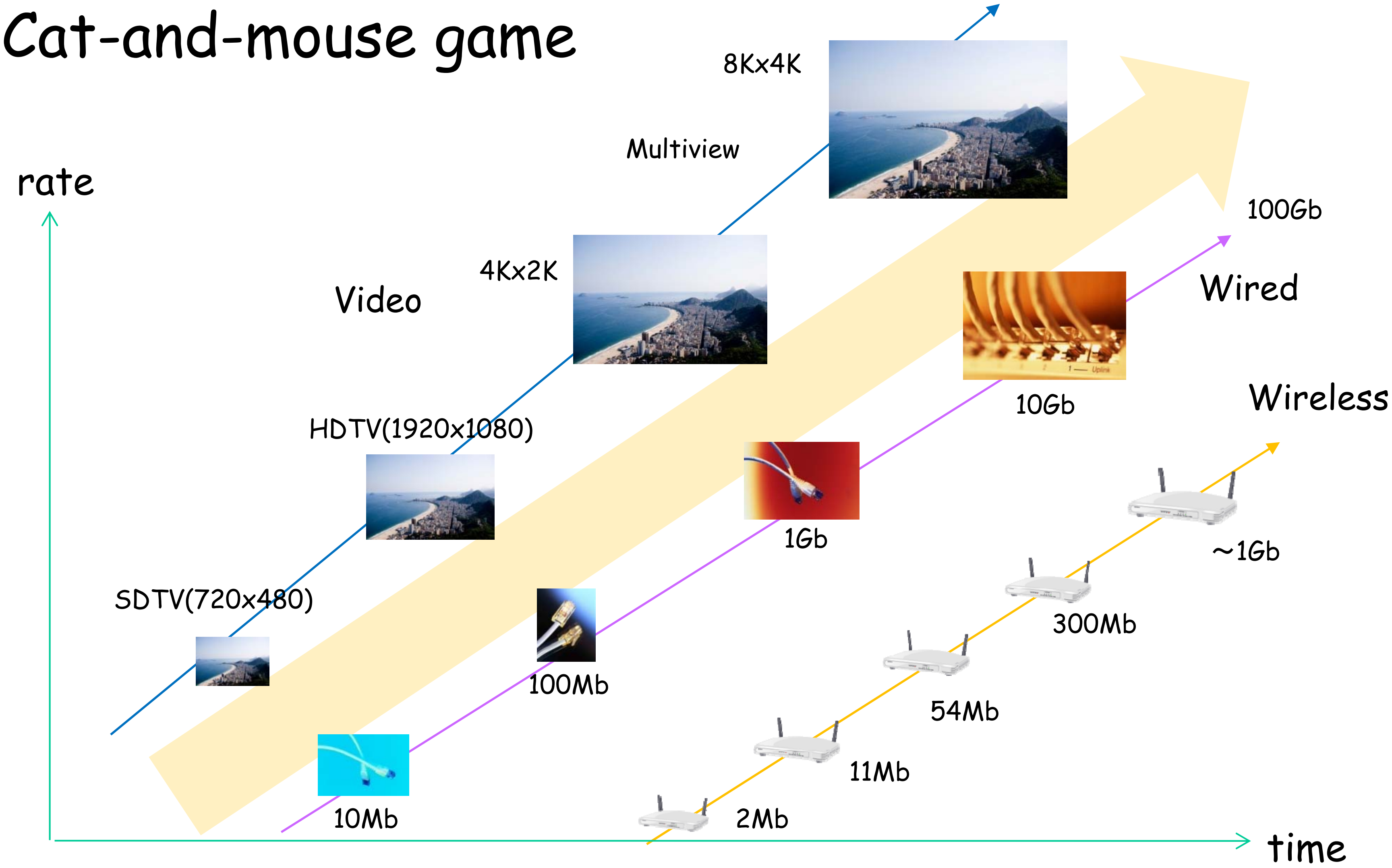
Protocol Stack for HTTP Video Streaming

VoIP, IPTV and streaming shares almost common protocol stack

application (L7)	video (H.264 etc...)	audio	MPD (MPEG-DASH)	layout (HTML)
adaptation	HTTP			
transport (L4)	TCP			
network (L3)	IP (IPv4, IPv6)			
datalink & physical (L2 & L1)	actual networks (802.3 (ethernet), 802.11 (WiFi), etc)			

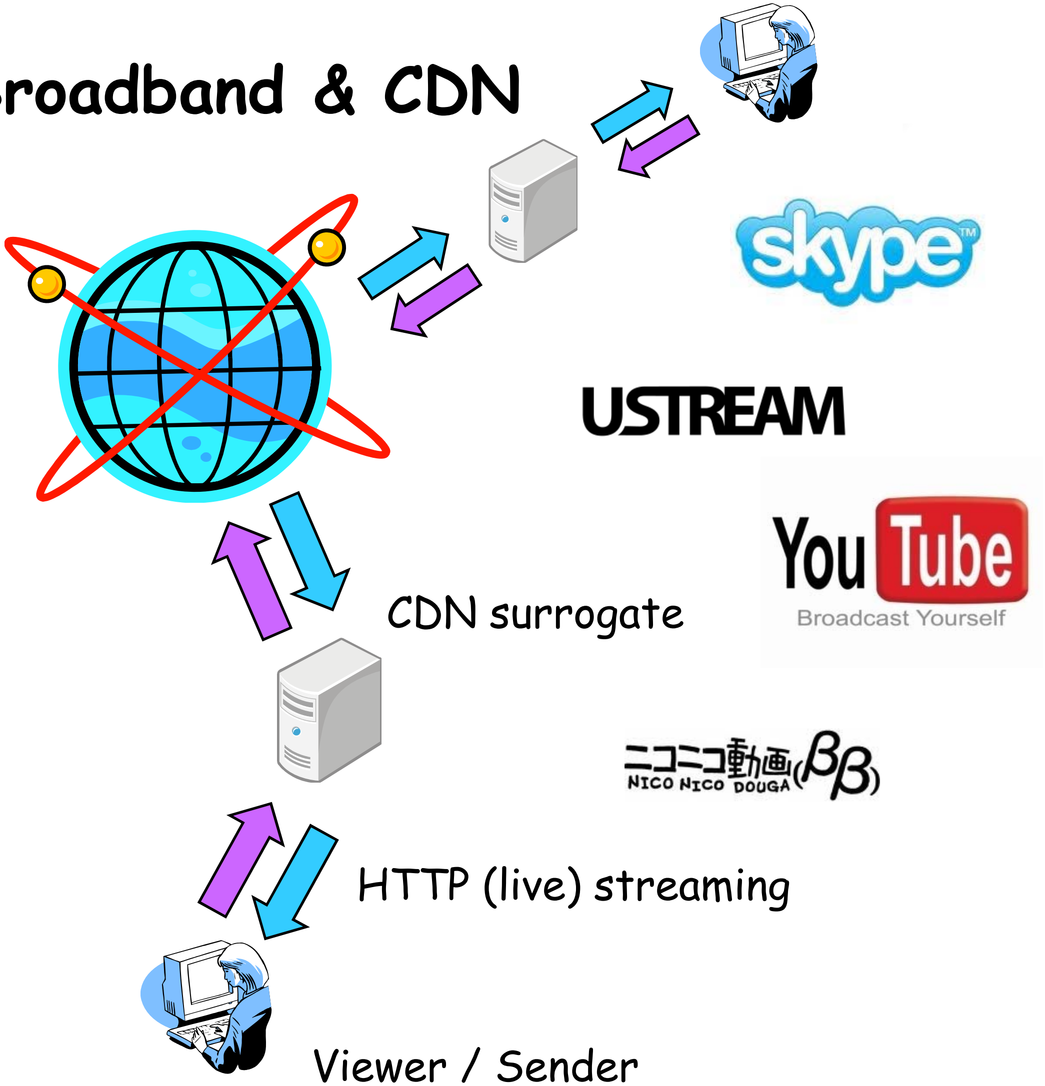
Networks and Multimedia

- Cat-and-mouse game



Wired Networks

Broadband & CDN



RTP/UDP & RTSP & TFRC

→ HTTP/TCP streaming

- Broadband
- CDN (Akamai, Lime Networks)
- Firewall (port 80)
- …

One-way (on-demand / live)

Bi-directional (interactive)

Wireless Networks

Wireless

Wireless specific problems

- Wireless LAN: IEEE 802.11
- Cellular: 3G, LTE, 4G
- WiMAX: IEEE 802.16
- Home Networks: DLNA
- (Satellite)
- ...

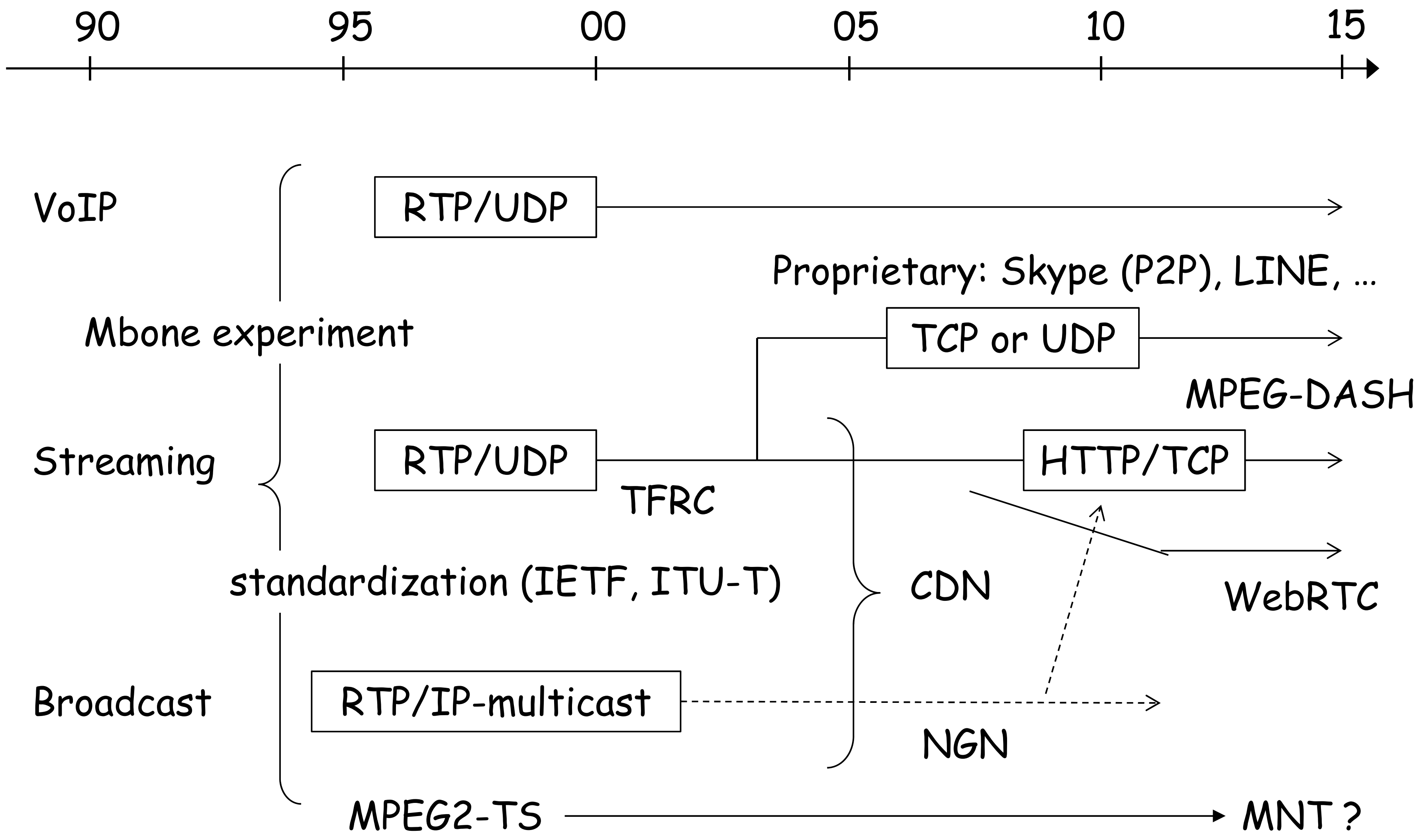
- **Wireless issues**

random errors, collisions, interference, delay increase

- **Multi-hop issues**

severe interference, lower throughput and higher delay

Protocol Transition

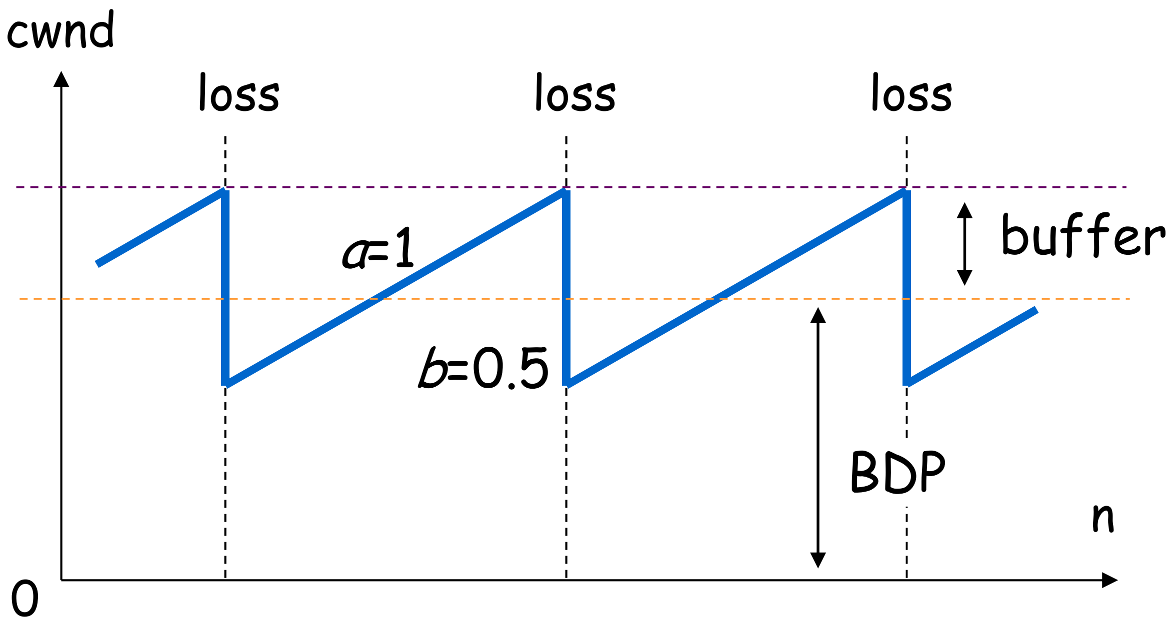


Overview

- TCP
- TFRC
- CDN & P2P
- MPEG-DASH
- WebRTC

TCP Variants

TCP-Reno (loss-based)

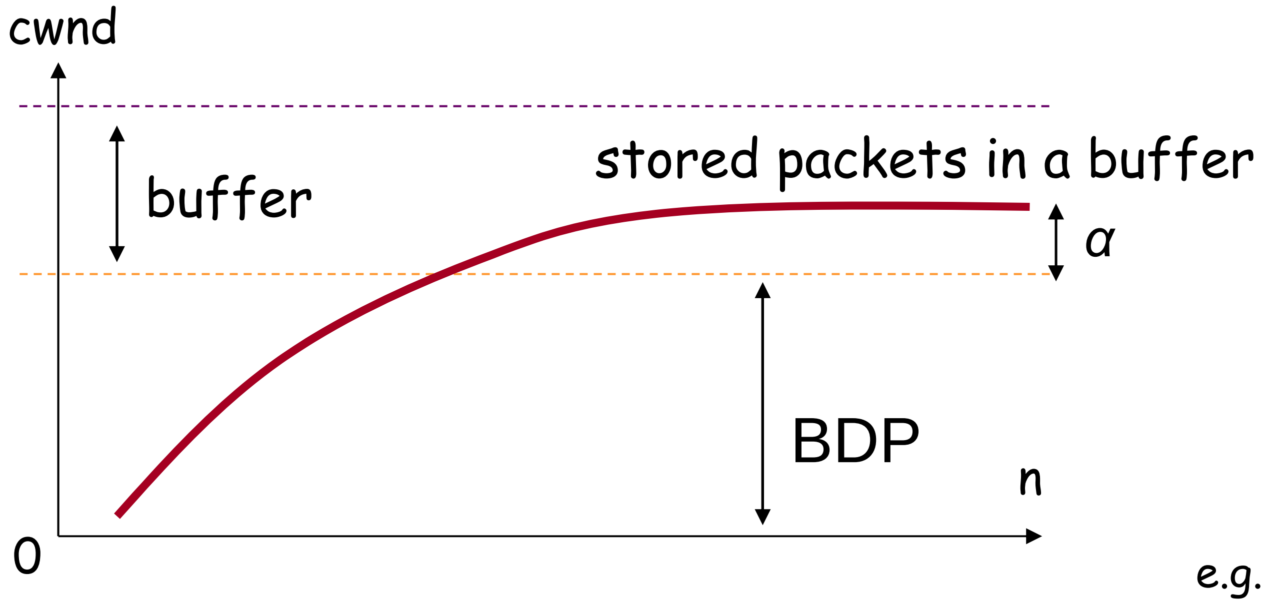


increase: cwnd = cwnd + 1/cwnd

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 ten years ago

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

- TCP-Reno (NewReno, SACK) problem
  - Reno expels Vegas (unfriendliness)

TCP Variants in the 21th century

- **Loss-driven (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
  - H-TCP (PFLDnet 2004)
  - TCP-Westwood (ACM MOBICOM 2001)
- **Delay-driven (RTT Observation)**
  - TCP-Vegas (IEEE JSAC, Oct 1995)
  - FAST-TCP (INFOCOM 2004)
- **Hybrid**
  - Gentle High-Speed TCP (PfHSN 2003)
  - TCP-Africa (IEEE INFOCOM 2005)
  - **Compound TCP** (PFLDnet 2006) ... Windows
  - Adaptive Reno (PFLDnet 2006)
  - YeAH-TCP (PFLDnet 2007)
  - TCP-Fusion (PFLDnet 2007)

Loss-based TCPs

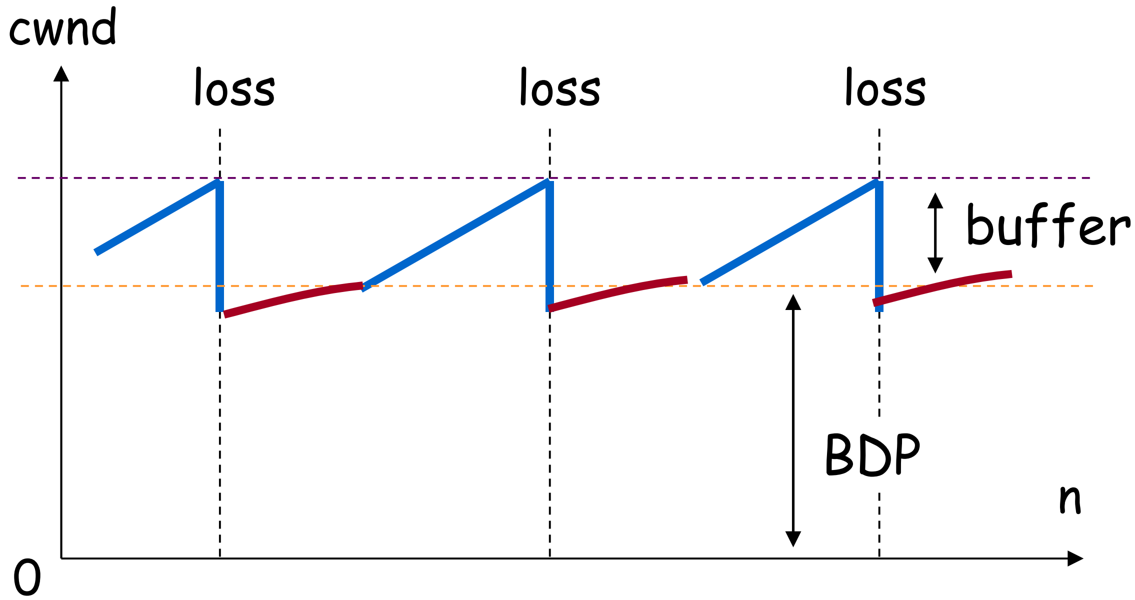
	Variants	Increase / Update (a)	Decrease (b)
	TCP-Reno	1	0.5
aggressive	HighSpeed TCP (HS-TCP)	$a(w) = \frac{2w^2 \cdot b(w) \cdot p(w)}{2 - b(w)}$ e.g. 70 (10Gbps, 100ms)	$b(w) = \frac{\log(w) - \log(W_{low})}{\log(W_{high}) - \log(W_{low})}(b_{high} - 0.5) + 0.5$ e.g. 0.1 (10Gbps, 100ms)
aggressive	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

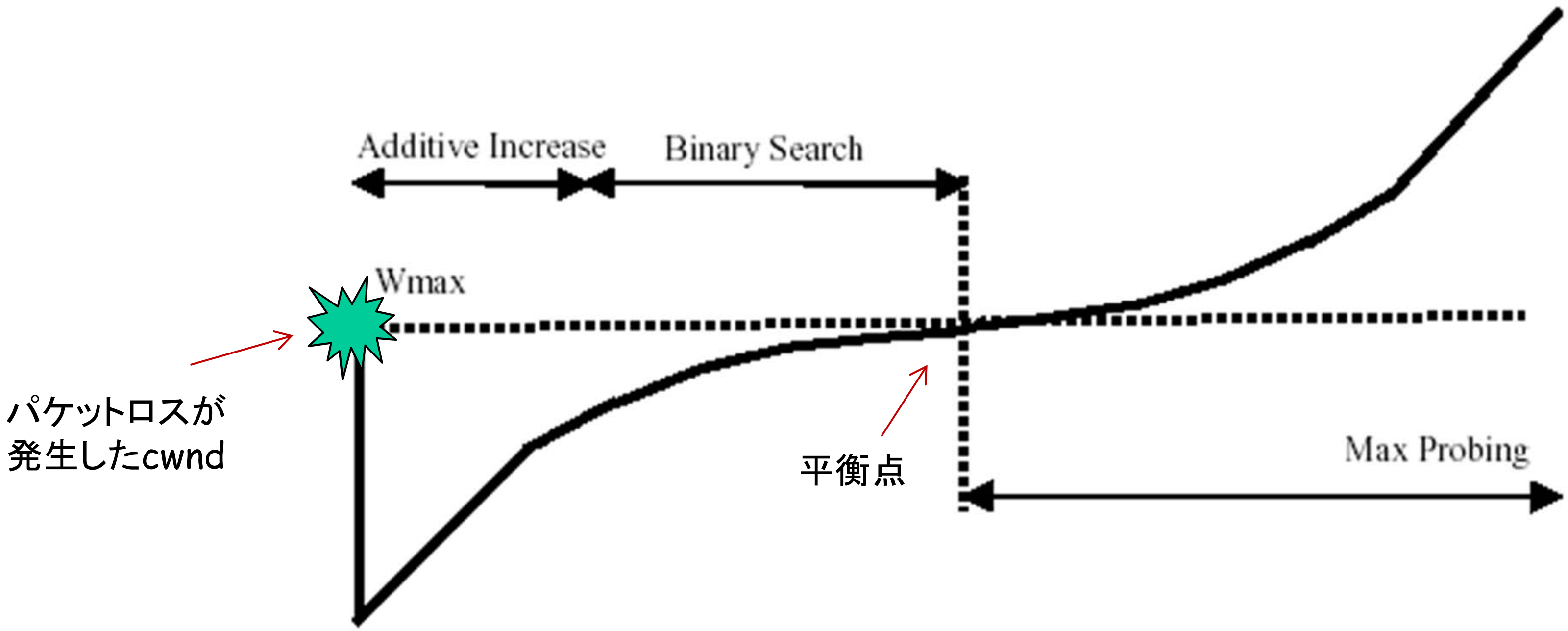
Hybrid TCPs

	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

BIC-TCP (1)

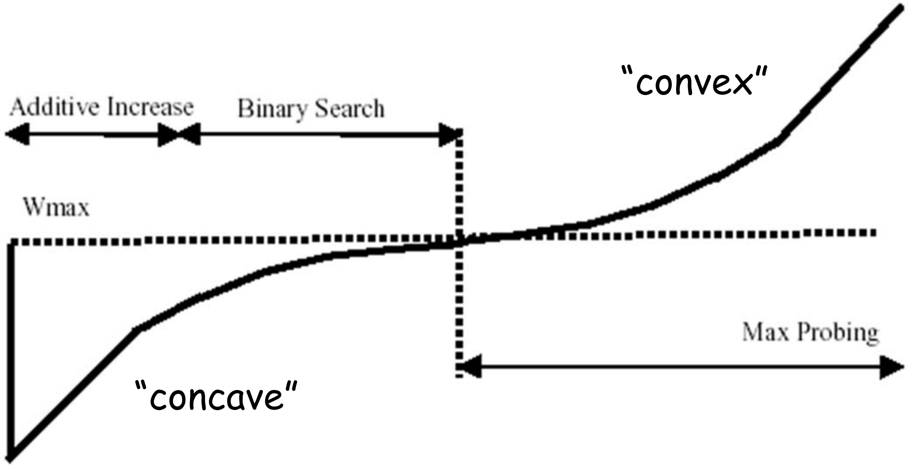
- Binary Increase Congestion Control



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

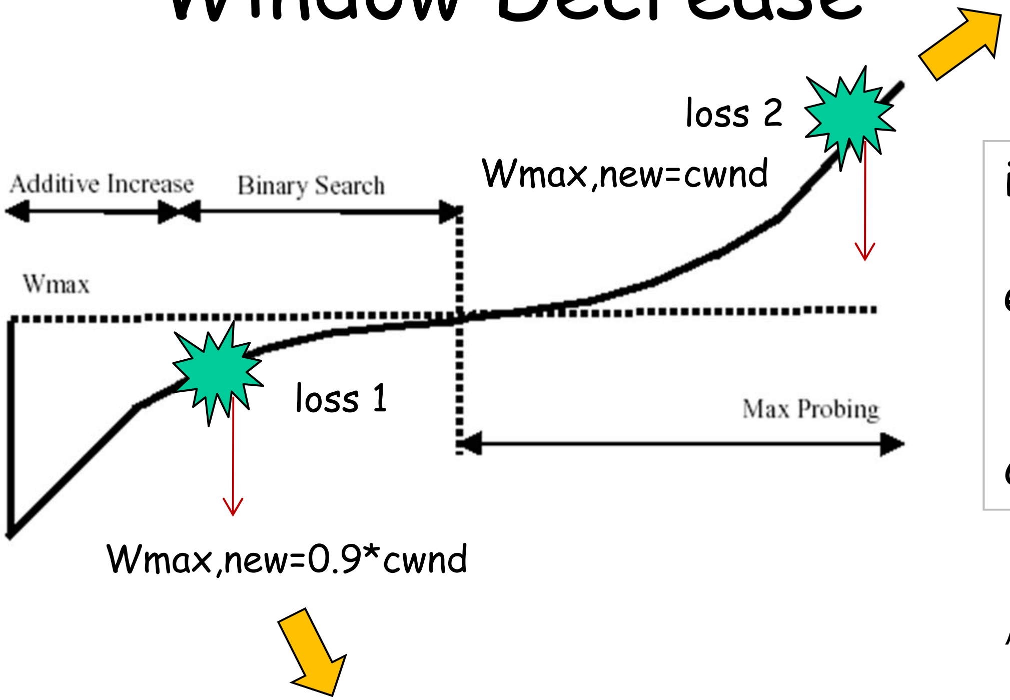
Smax: maximum increase rate (e.g. 32)

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);
```

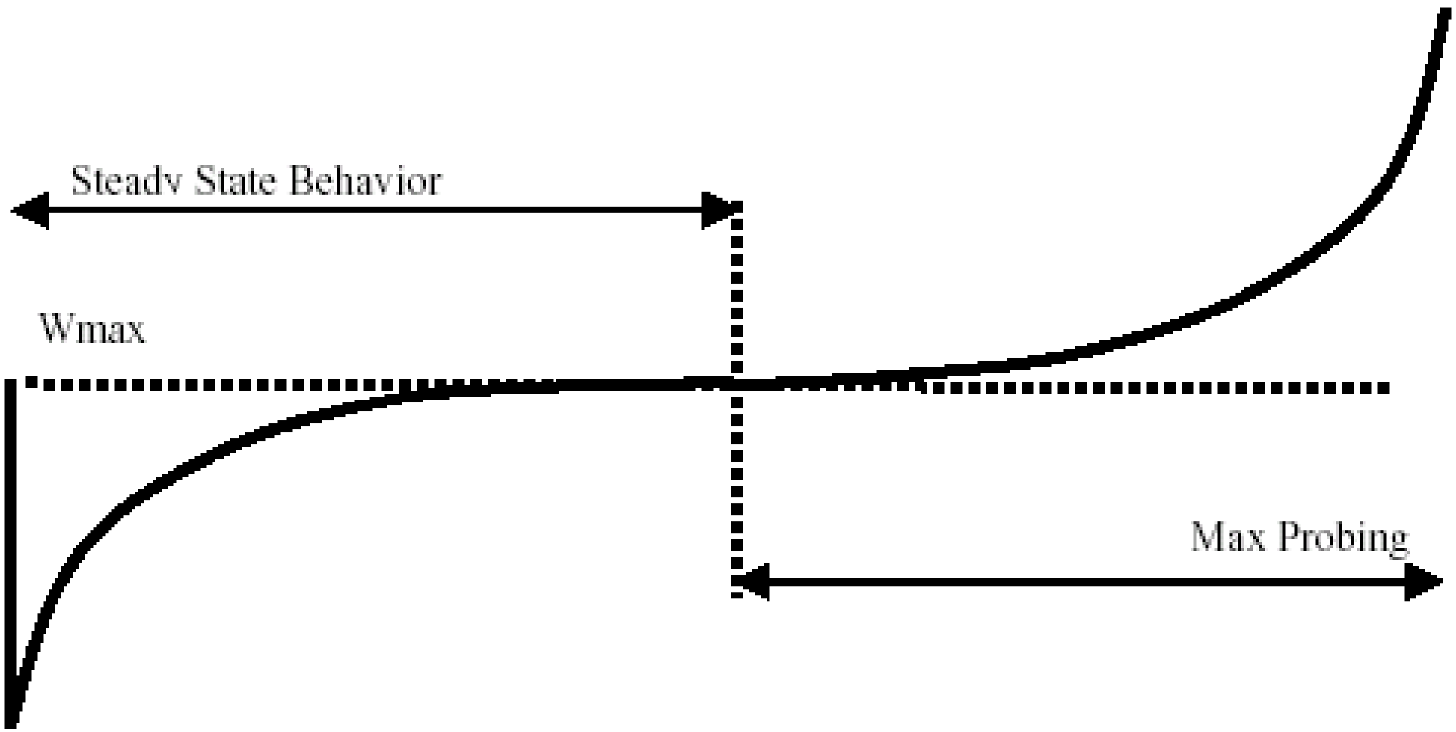
β: decrease rate (e.g. 0.2)

*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)

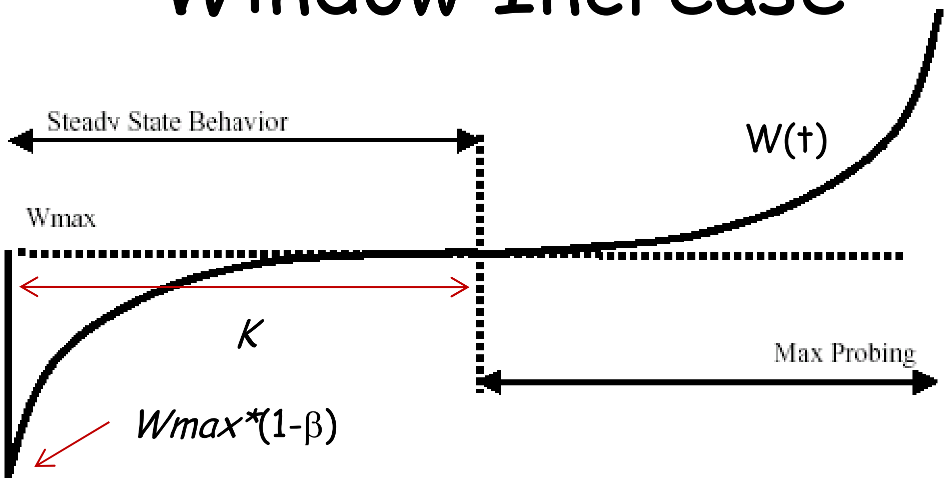
- Cubic approximation of BIC-TCP



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

CUBIC-TCP (2)

- Window Increase



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

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

equivalent to Reno

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

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

cwnd = cwnd + Winc / cwnd;

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

※ window decrease is the same as BIC

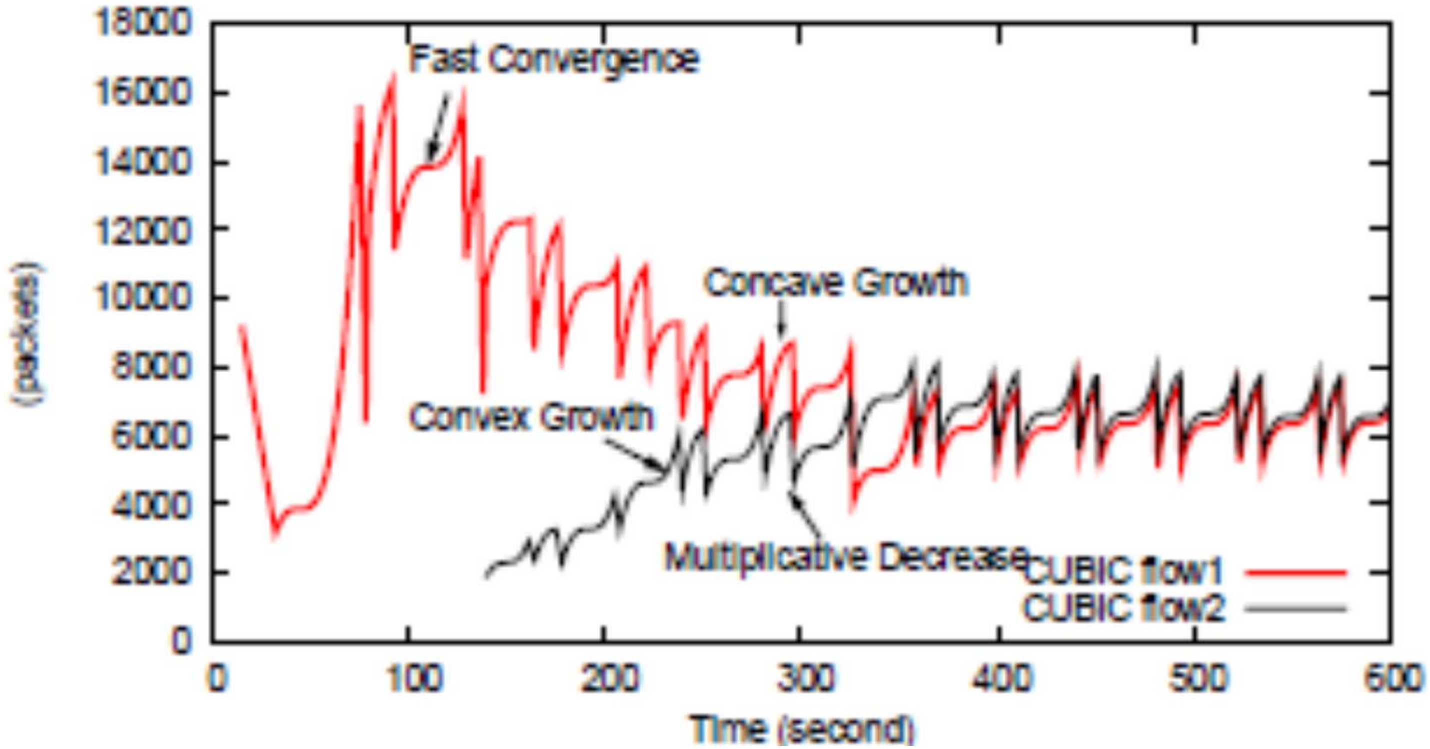
β: decrease rate (e.g. 0.2)

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)

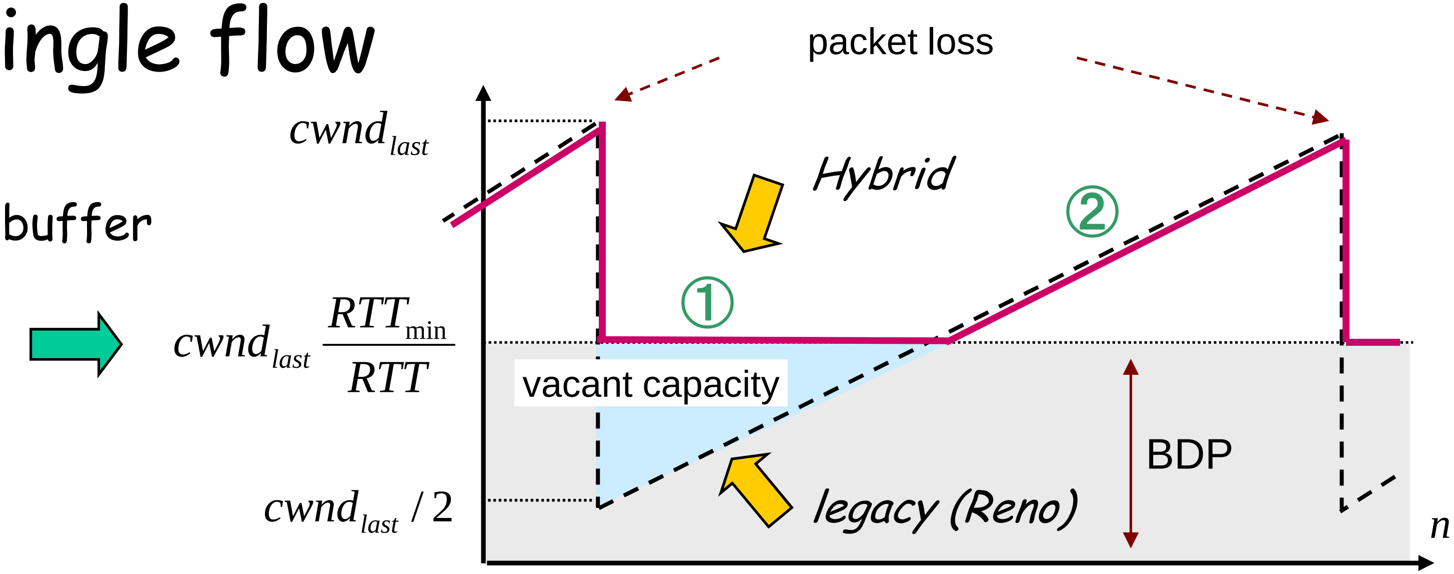
K.Munir et al: "Linux beats Windows! or the Worrying Evolution of TCP...", PFLDNet 2007.

Hybrid TCPs

Hybrid TCP (1)

- single flow

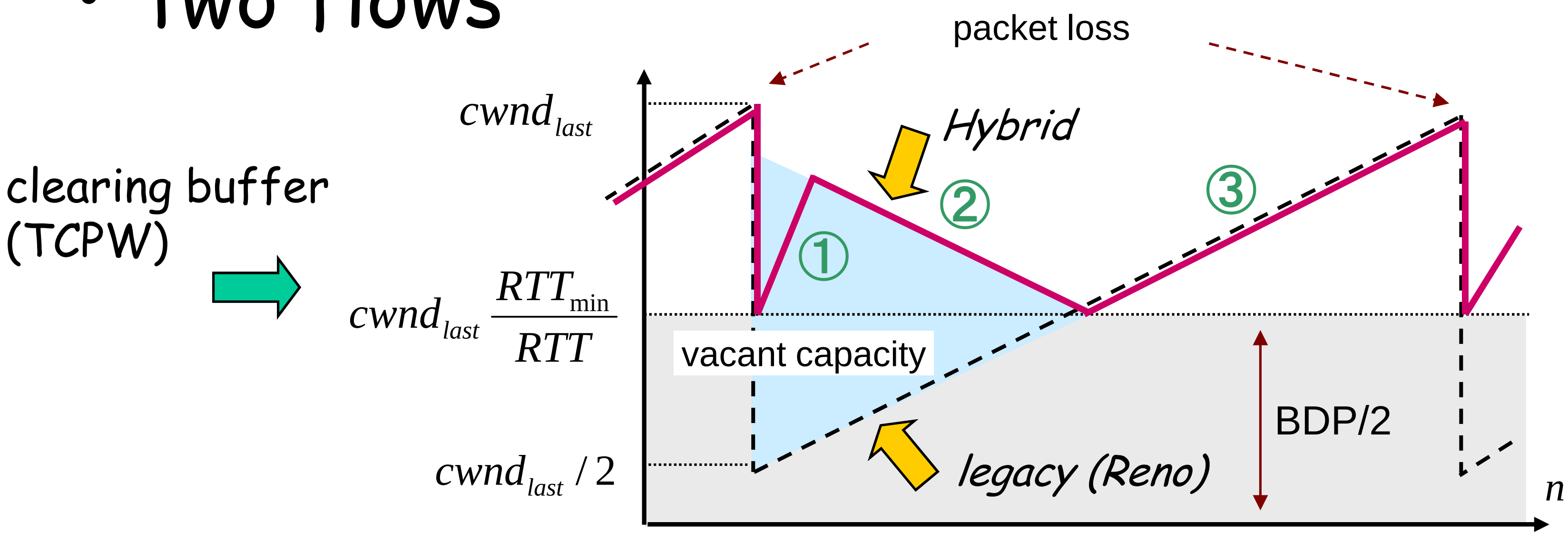
clearing buffer
(TCPW)



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):

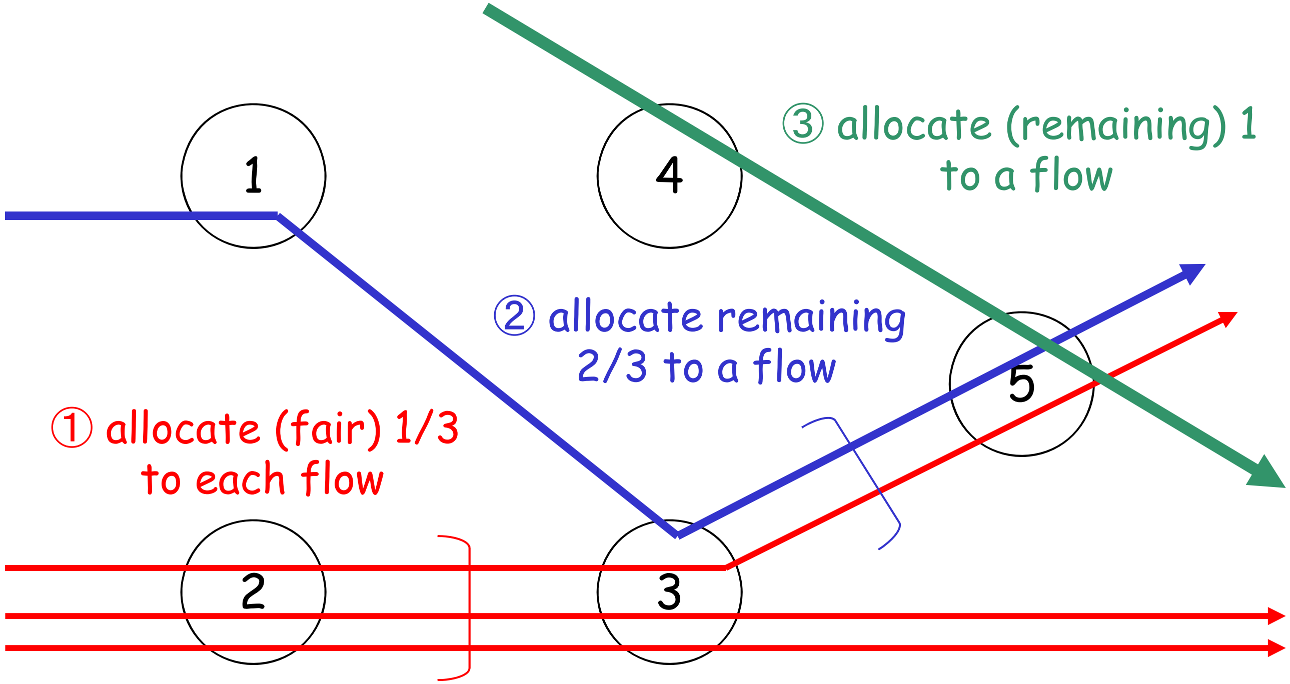
① fast cwnd increase (delay … "efficiency")

② mild cwnd decrease (delay … congestion avoidance)

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

Min-Max Fair (ideal case)

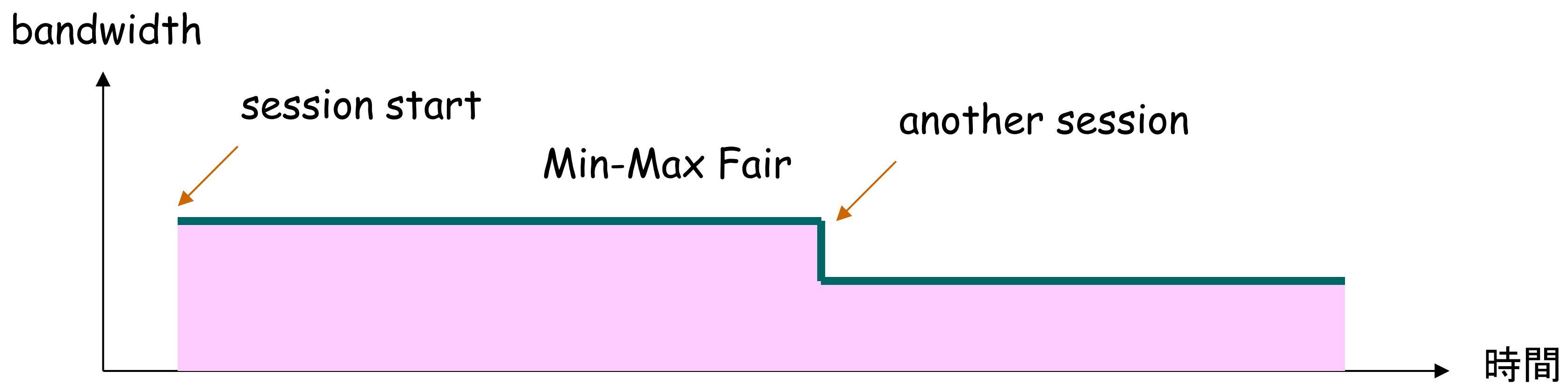
- Min-Max-Fair: allocate “maximum bandwidth” to a user who has “minimum bandwidth”



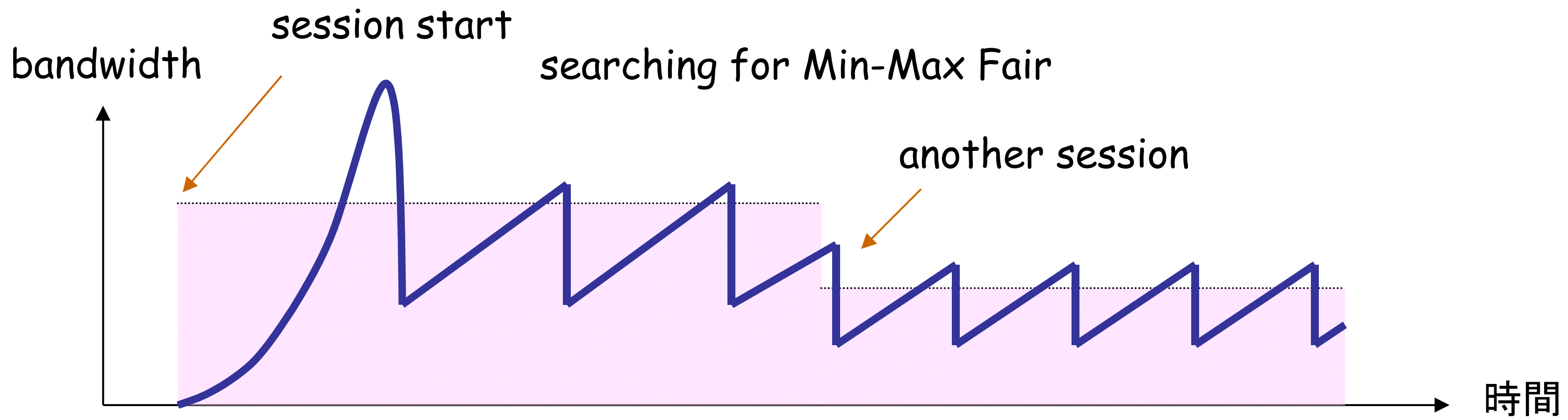
D.Bertsekas and R.Gallager: “Data Networks,” Prentice Hall.

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)

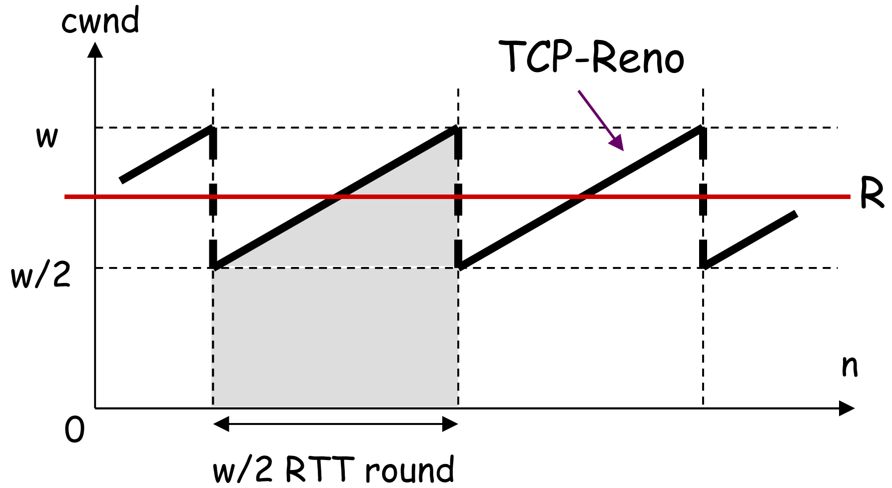
- parameter definition
  - w : cwnd when a packet loss is detected
  - W : cwnd which just fills a pipe ~ BDP
  - p : packet loss rate

- 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

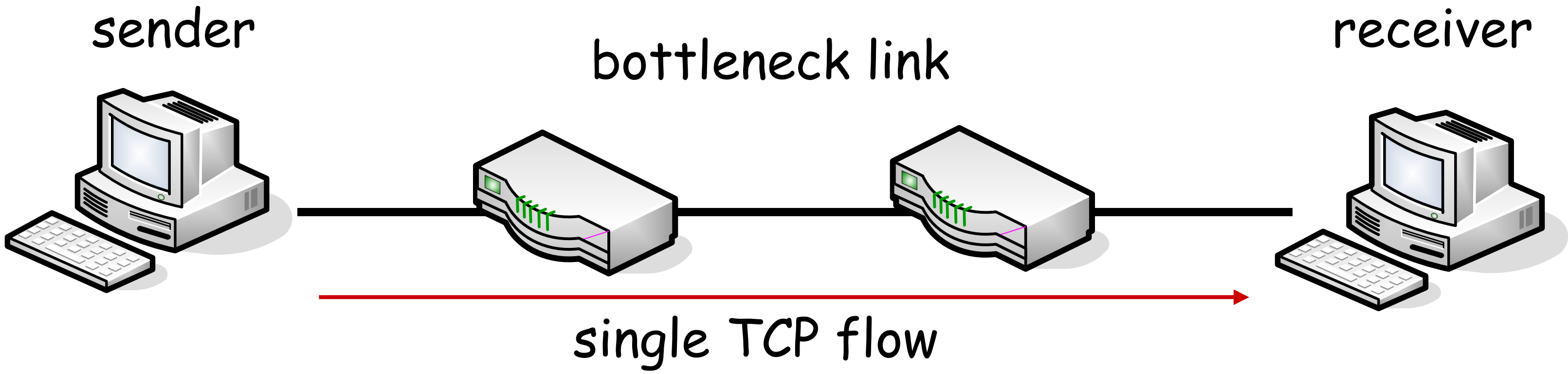
= area of a trapezoid

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

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

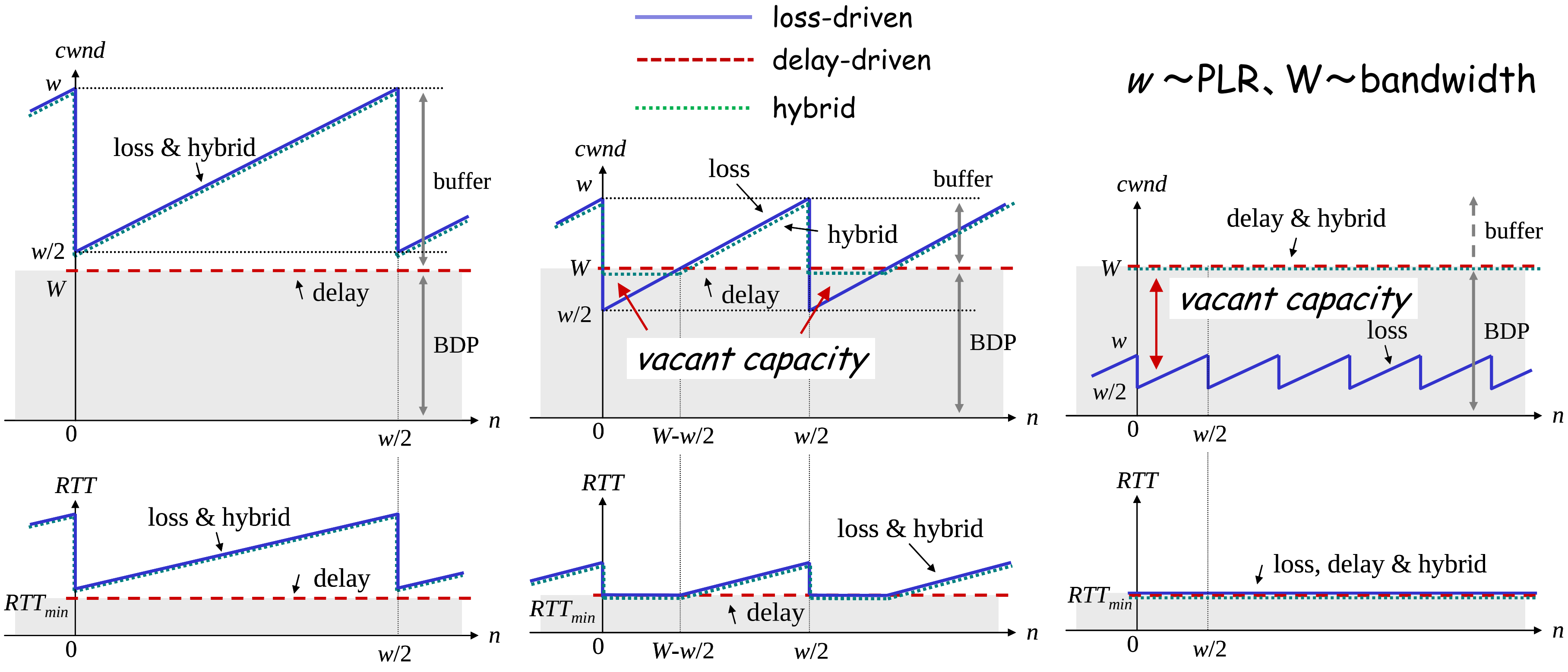
TCP behavior model (4)

- single flow



TCP behavior model (5)

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



(i) $W < w/2$	(ii) $w/2 < W < w$	(iii) $w < W$
large buffer, small PLR (always loss-mode)	small buffer, medium PLR (delay ↔ loss)	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)

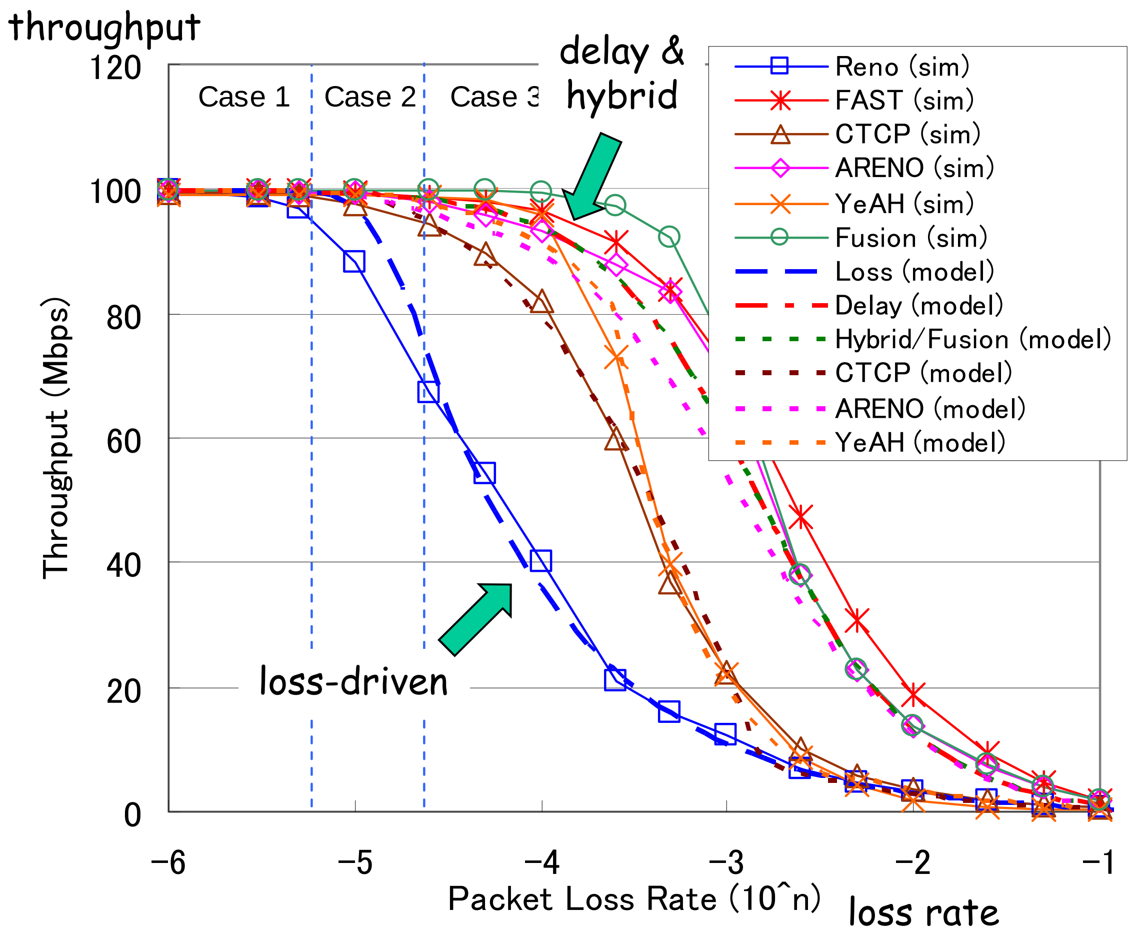
- abstraction of actual hybrids

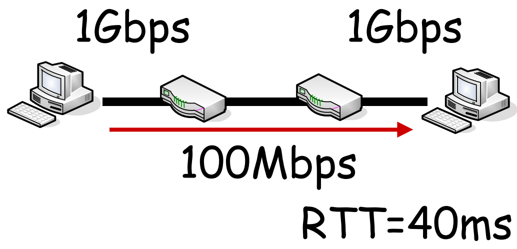
Hybrids	Window increase	Window decrease
Compound TCP	$0.125*cwnd^{0.75}$	1/2
ARENO	B/10Mbps	1/2～1
YeAH-TCP	Scalable TCP (1.01)	1/2, RTT_{min}/RTT, 7/8
TCP-Fusion	$B*D_{min}/(N*PS)$	RTT_{min}/RTT

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

TCP behavior model (8)

- evaluation by models and simulations





buffer size = BDP (constant)

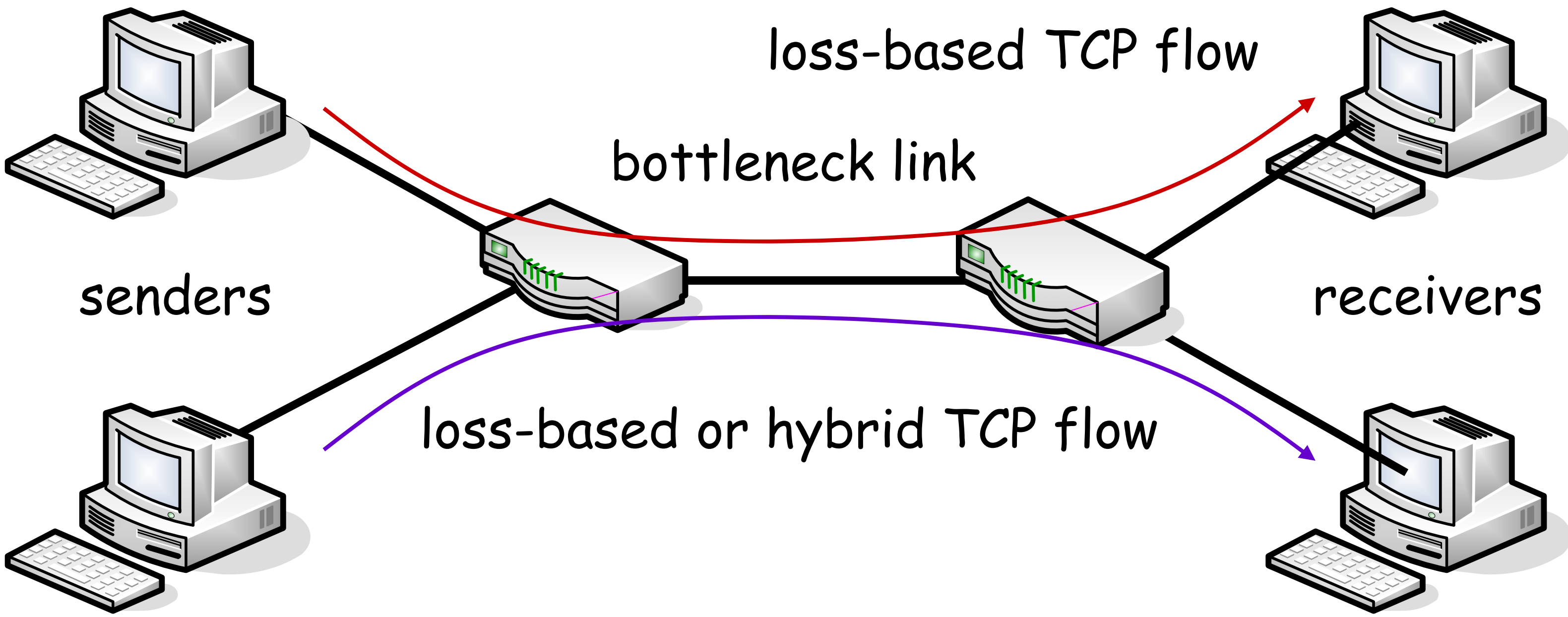
Packet loss rate : variable

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

degradation of Compound & YeAH is due to fixed window decrease

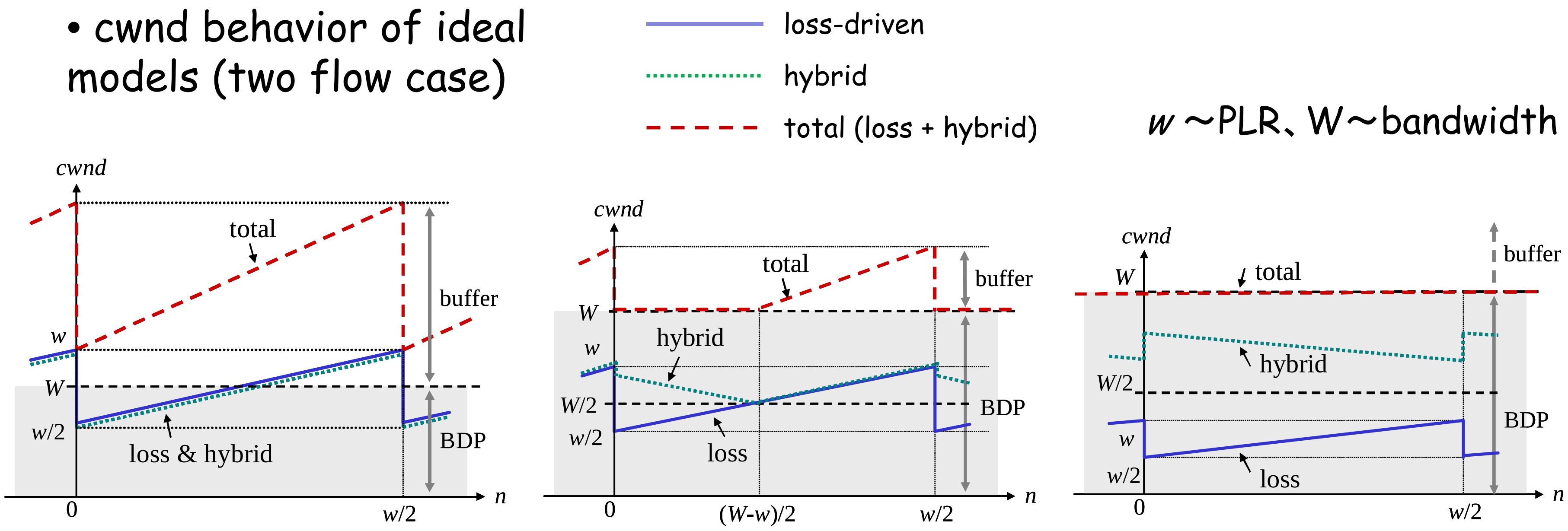
TCP behavior model (9)

- two flows (competing)



TCP behavior model (10)

- cwnd behavior of ideal models (two flow case)



(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)

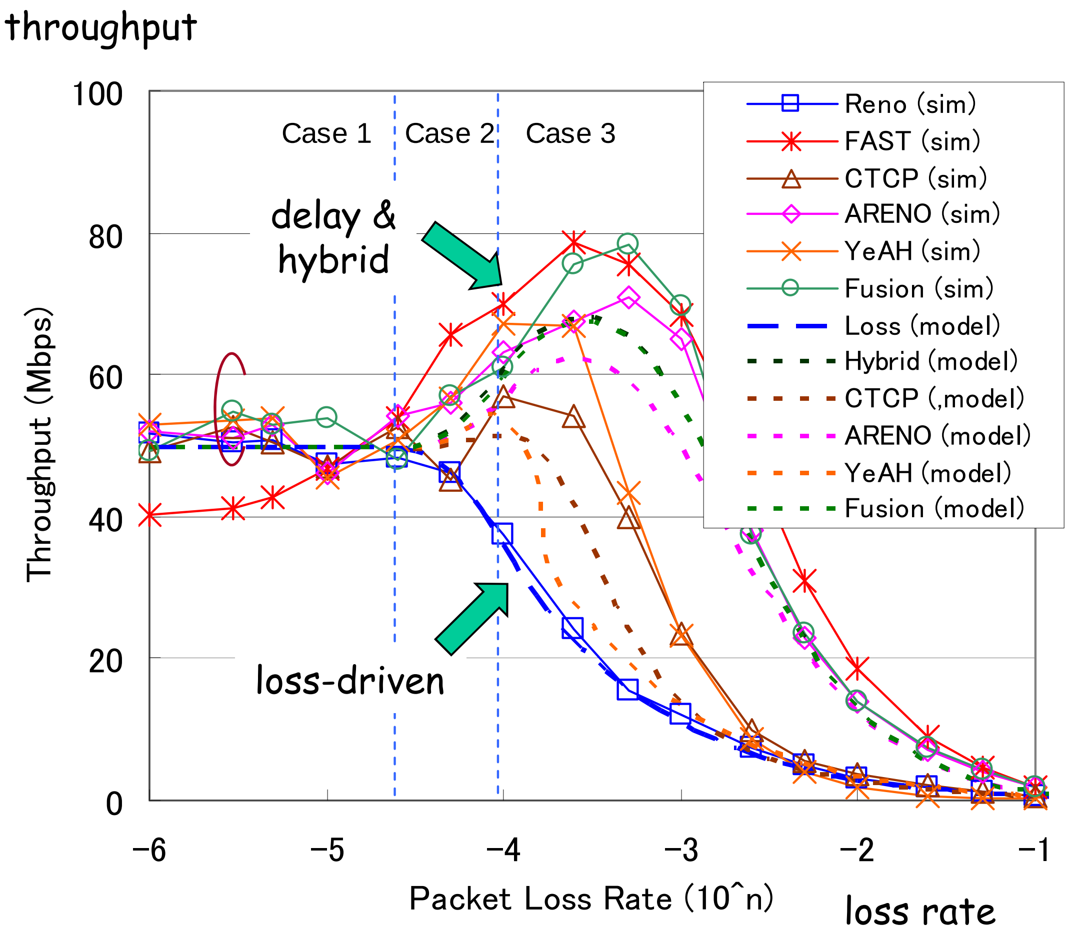
- formulation

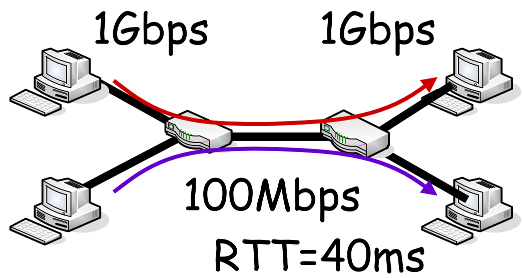
TCP	CA round	(i) $W < w$	(ii) $w \le W < 2w$	(iii) $2w \le W$
Loss	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$
Hybrid	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2 + \frac{1}{4}(W-w)^2$	$\frac{1}{2}w \cdot W - \frac{3}{8}w^2$
(common)	elapsed time	$\frac{1}{2}w \cdot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

PS: Packet size, B: Link bandwidth

TCP behavior model (12)

- evaluation by models and simulations





buffer size = BDP (constant)

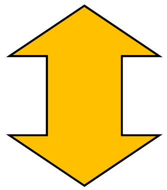
Packet loss rate : variable

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

TCP over Wireless

Wireless issues

- error control (L1)
  - BER (bit error rate), PER (packet error rate)
  - error model: AWGN, Two-States Markov
- access control (L2)
  - CSMA/CA (MACA, MACAW):
    - interference, collision
    - hidden terminal, exposed terminal
    - grey zone: receive range & carrier sense range
  - TDMA, FDMA, CDMA
- ad-hoc routing (L3)
  - DSDV, DSR, AODV, OLSR, TORA, AOMDV, ...
- transport protocol (L4)
  - Wireless TCP/TFRC, multi-hop TCP/TFRC
- mobility management (L3 / L7)
  - mobility model: Random Waypoint, Random Trip
  - handover
- energy consumption (all layers)
  - energy model

Discussion

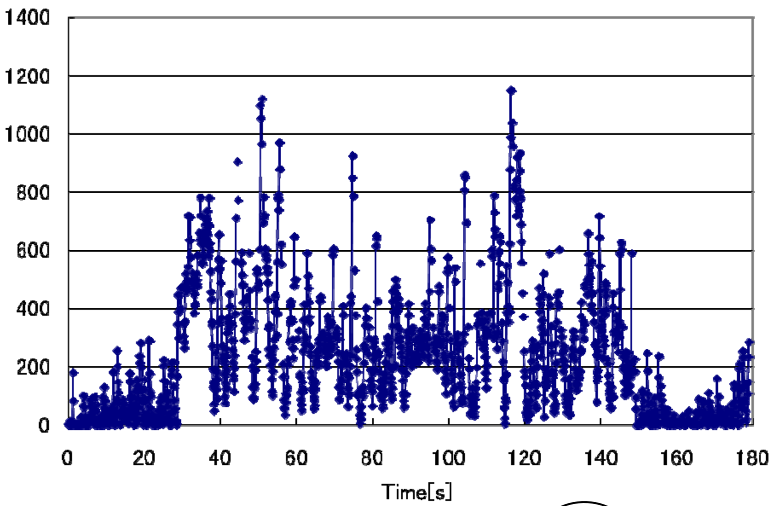
- Wireless LAN
  - CSMA/CA, half-duplex, interferences, random errors, ...
    - cannot send packets when the sender wants to
    - packets are continuously stored into a transmission buffer of the sender
  - NIC buffer size is very large
    - Hybrid TCP always operates in the loss mode only
  - Unfairness between upload and download
    - D.Leith: WiOpt 2005

TCP over Wireless Networks
(1) TCP Differentiation

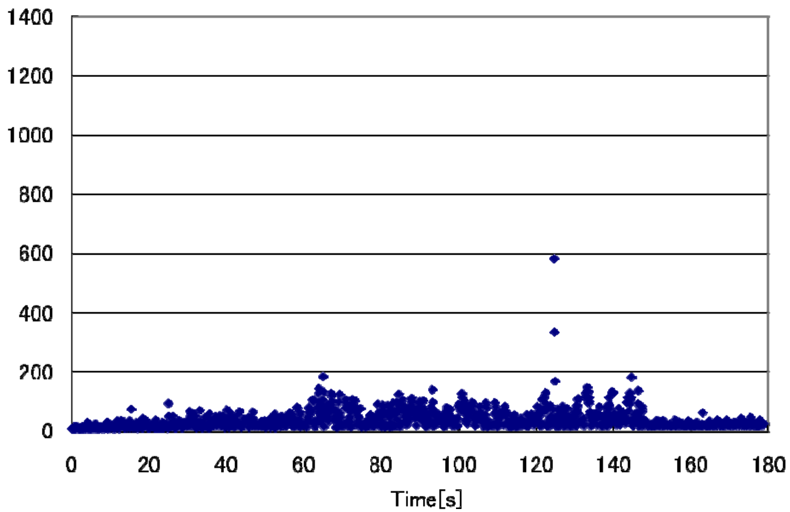
WiFi Example

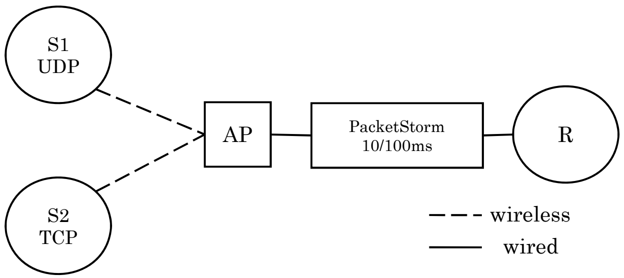
- RTT instability and unfairness between upload and download

RTT upload, wireless to wired



RTT download, wired to wireless



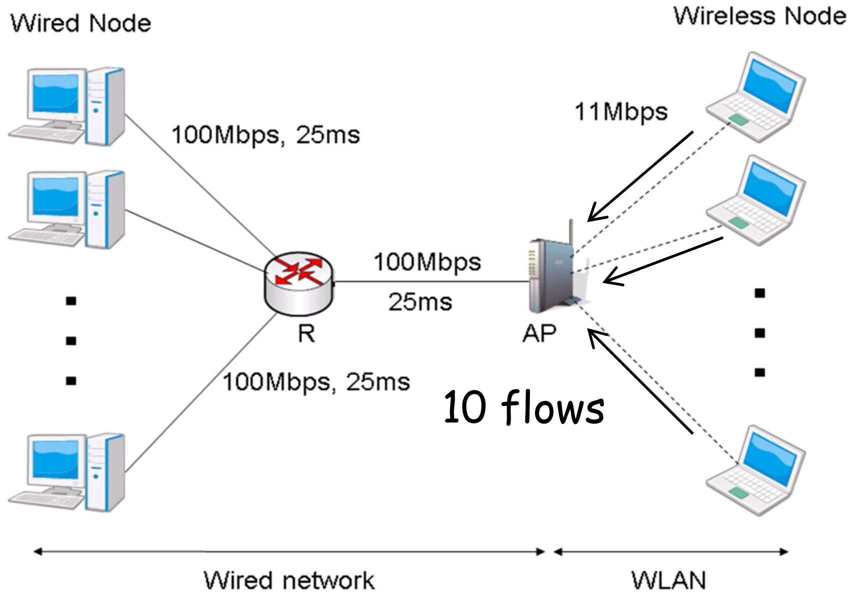


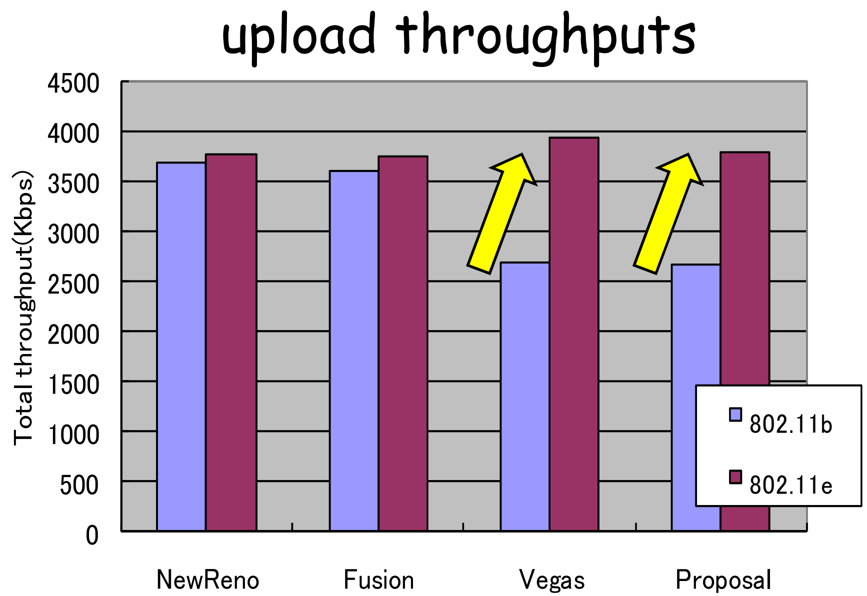
K.Kanai et al: “Performance evaluations of adaptive rate control mechanisms ...”, IEICE Tech. Report, 2011.

Wireless LAN (1)

※ ns-2 simulation

- TCPs and throughputs





Apply IEEE 802.11e to alleviate the unfairness problem between upload and download

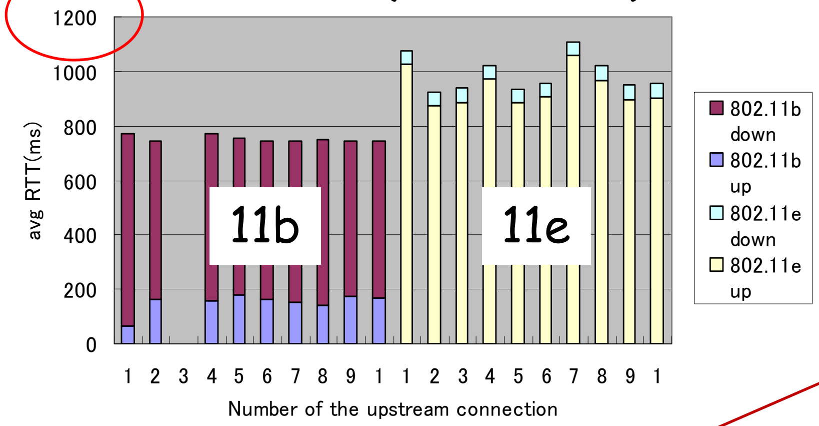
TCP-Reno: loss based
TCP-Fusion: hybrid
TCP-Vegas: delay based
Proposal: Vegas extension

K.Sonoda et al: "TCP Differentiation using Version Identification…", IEEE CCNC 2013.

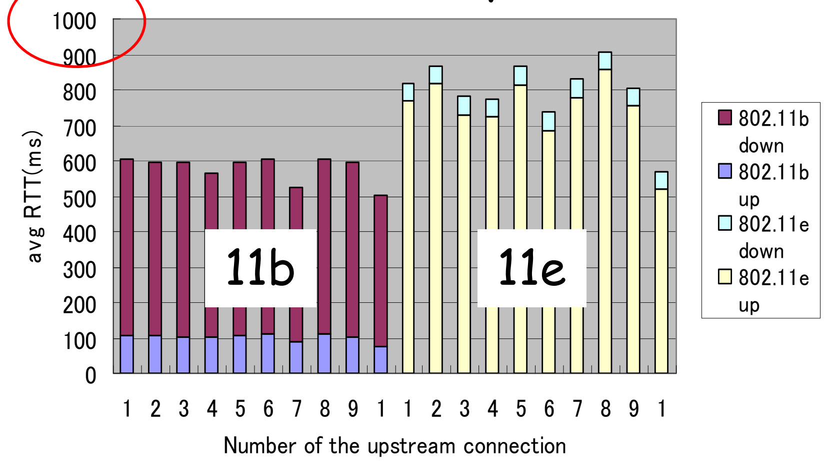
Wireless LAN (2)

- TCPs and delays

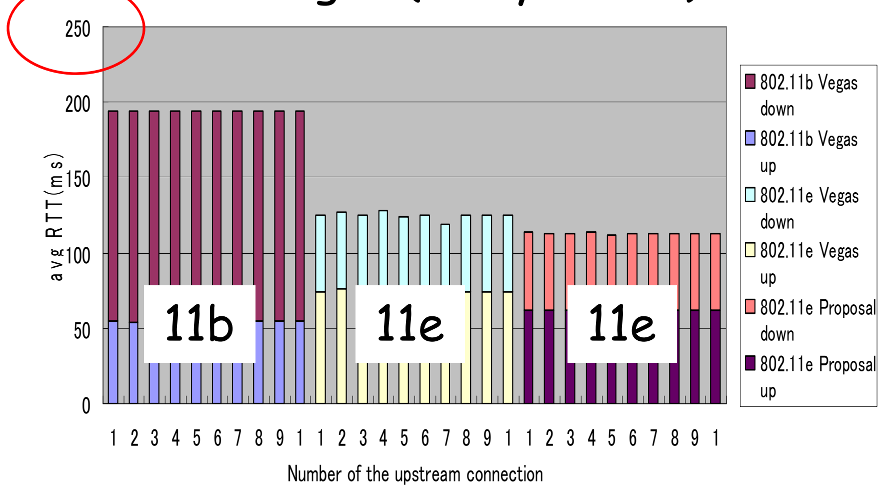
TCP-Reno (loss based)



TCP-Fusion (hybrid)



TCP-Vegas (delay based)



Reno, Fusion: though unfairness was alleviated, delay increases (esp. upload)
Vegas & Proposal: unfairness and delay are decreased (compare vertical axis)

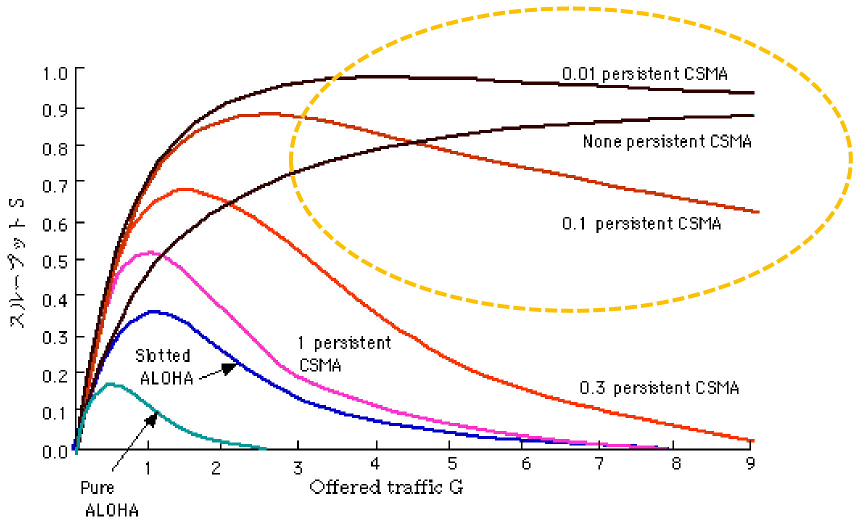
→ Hybrid TCP works in loss mode only

K.Sonoda et al: "TCP Differentiation using Version Identification…", IEEE CCNC 2013.

Wireless LAN (3)

- Common to wired
  - Delay based TCP design is effective if we require low delay transmission (but, it is expelled by loss based flows)
- Differences to wired
  - Hybrid does not operate in "hybrid" (delay mode) due to huge transmission buffer
  - Too many packet insertion causes huge delay due to multiple access mechanism (i.e. CSMA)

Critical throughput-delay tradeoff due to CSMA/CA



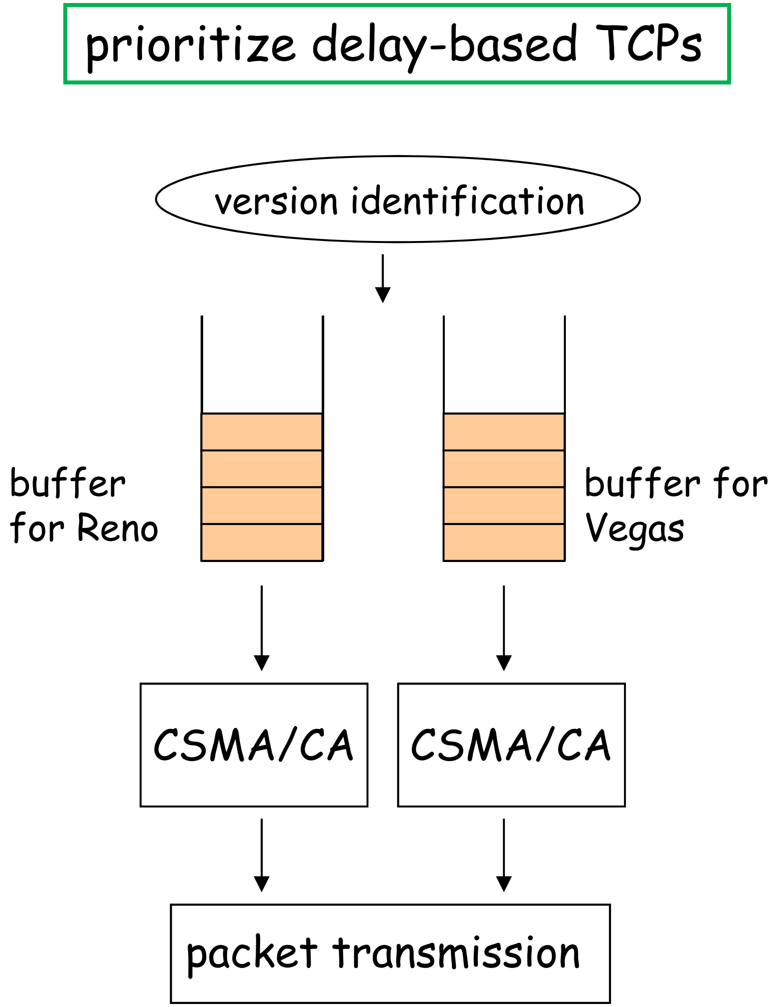
TCP Version Differentiation (1)

TCP version identification and differentiation

1. Access points identify TCP versions using RTT/cwnd estimation

2. Access points separate different TCP versions into different buffers

3. Prioritize delay based TCP flows by tuning CSMA/CA parameters of IEEE 802.11e

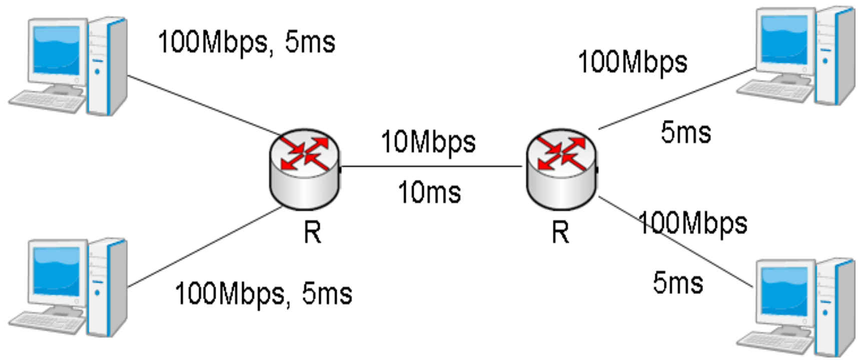


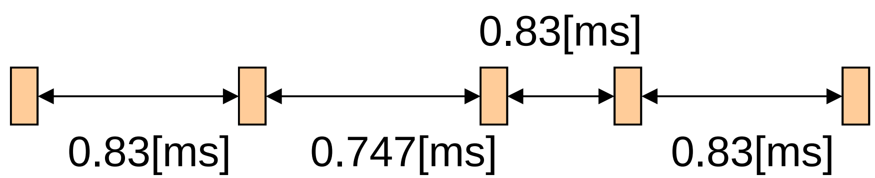
K.Sonoda et al: "TCP Differentiation using Version Identification…", IEEE CCNC 2013.

TCP Version Differentiation (2)

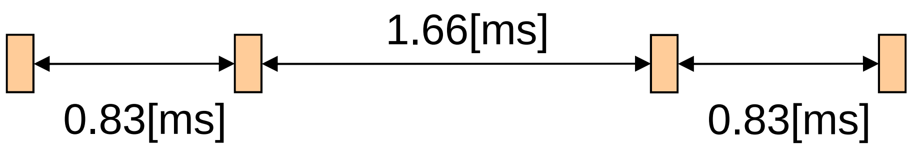
- RTT estimation for delay based flow
  - When cwnd increases by one, two consecutive packets are transmitted
  - When cwnd decreases by one, no packets are transmitted for the last ACK
- cwnd estimation
  - Access points let the number of arrived packets per RTT be "cwnd"

TCP behavior estimation at AP





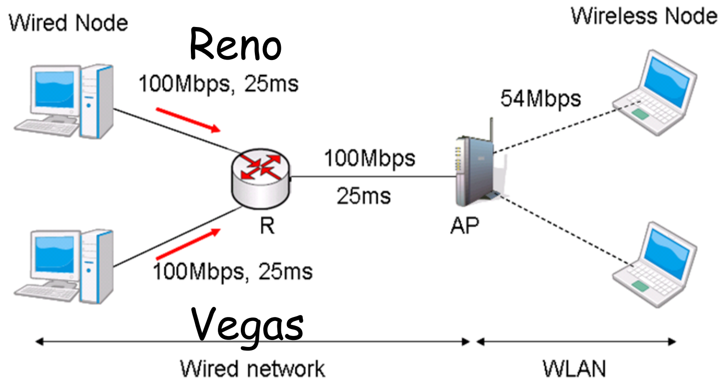
when cwnd increases by 1



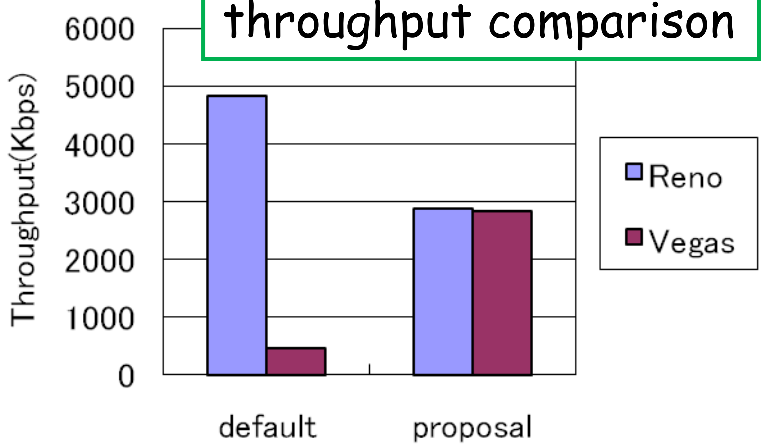
when cwnd decreases by 1

K.Sonoda et al: "TCP Differentiation using Version Identification…", IEEE CCNC 2013.

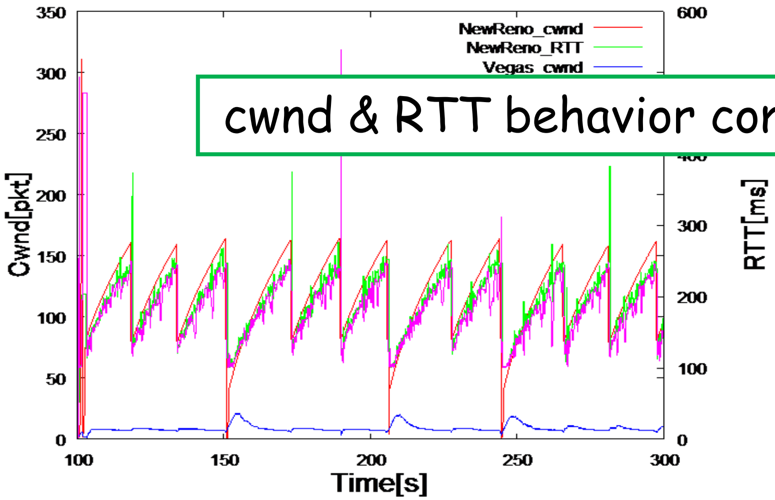
TCP Version Differentiation (3)



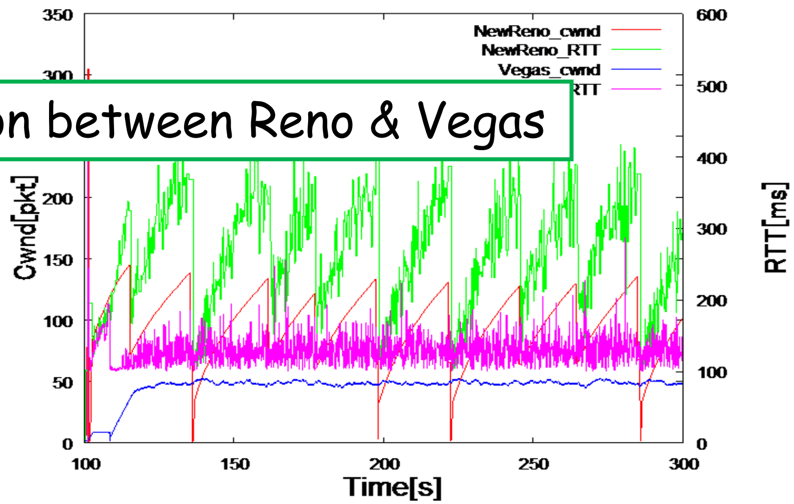
throughput comparison



cwnd & RTT behavior comparison between Reno & Vegas



without differentiation



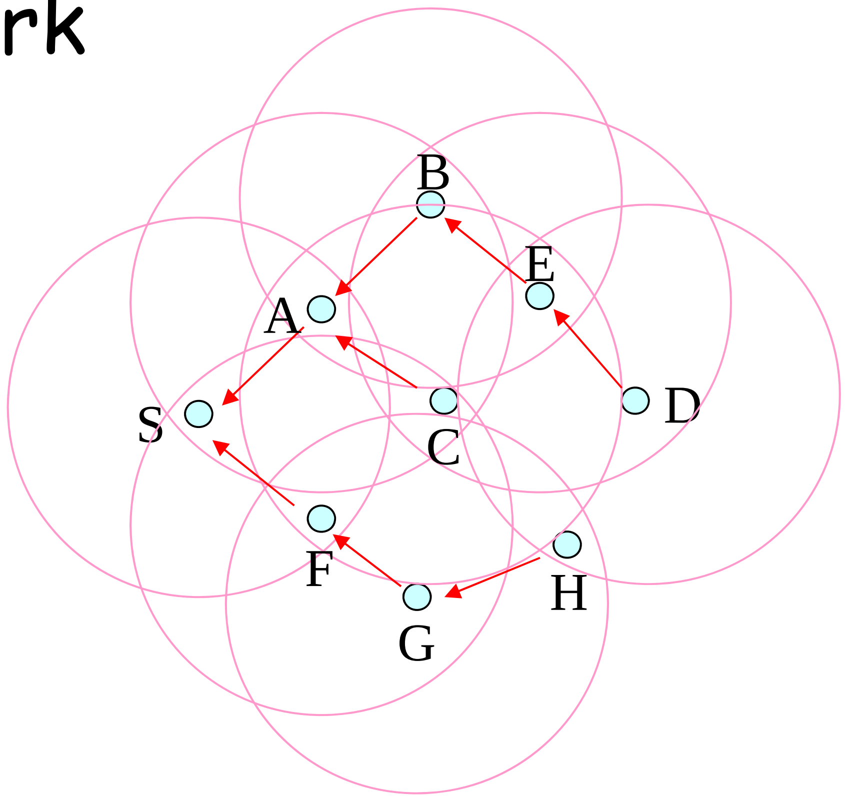
with differentiation

K.Sonoda et al: “TCP Differentiation using Version Identification…”, IEEE CCNC 2013.

TCP over Wireless Networks
(2) Multi-hop

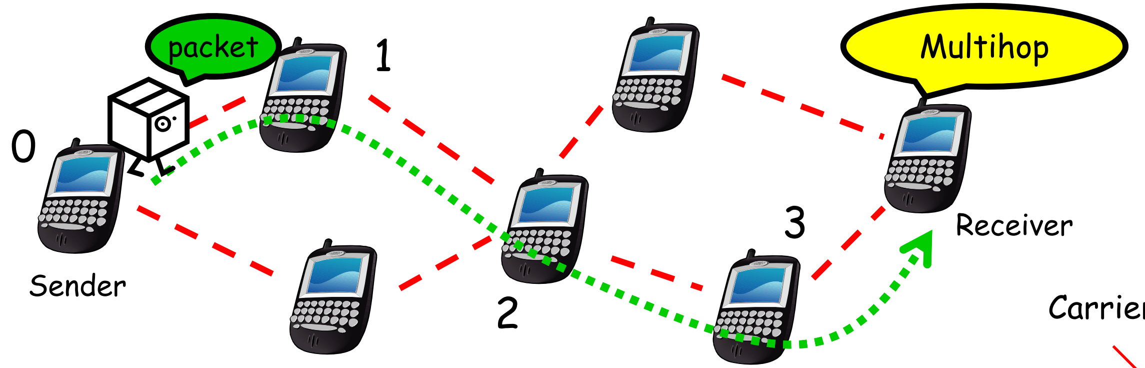
Wireless Multihop Networks

- ad-hoc network
- sensor network

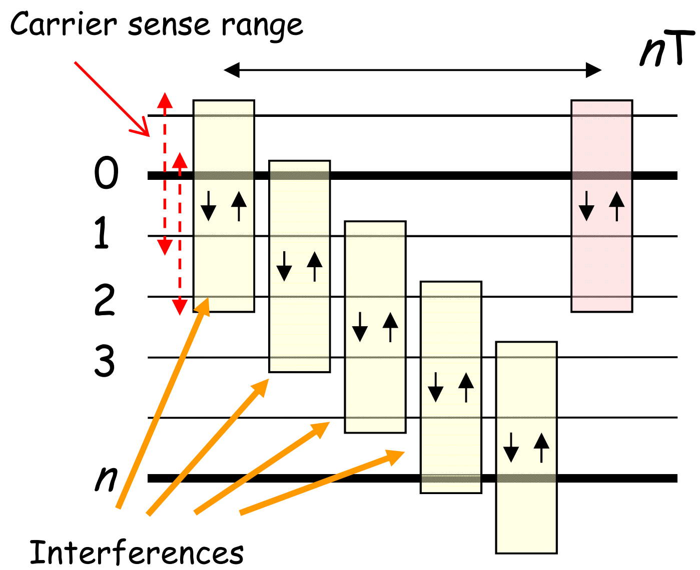


Wireless Multihop Networks (1)

- Single Radio Multi-hop Transmission



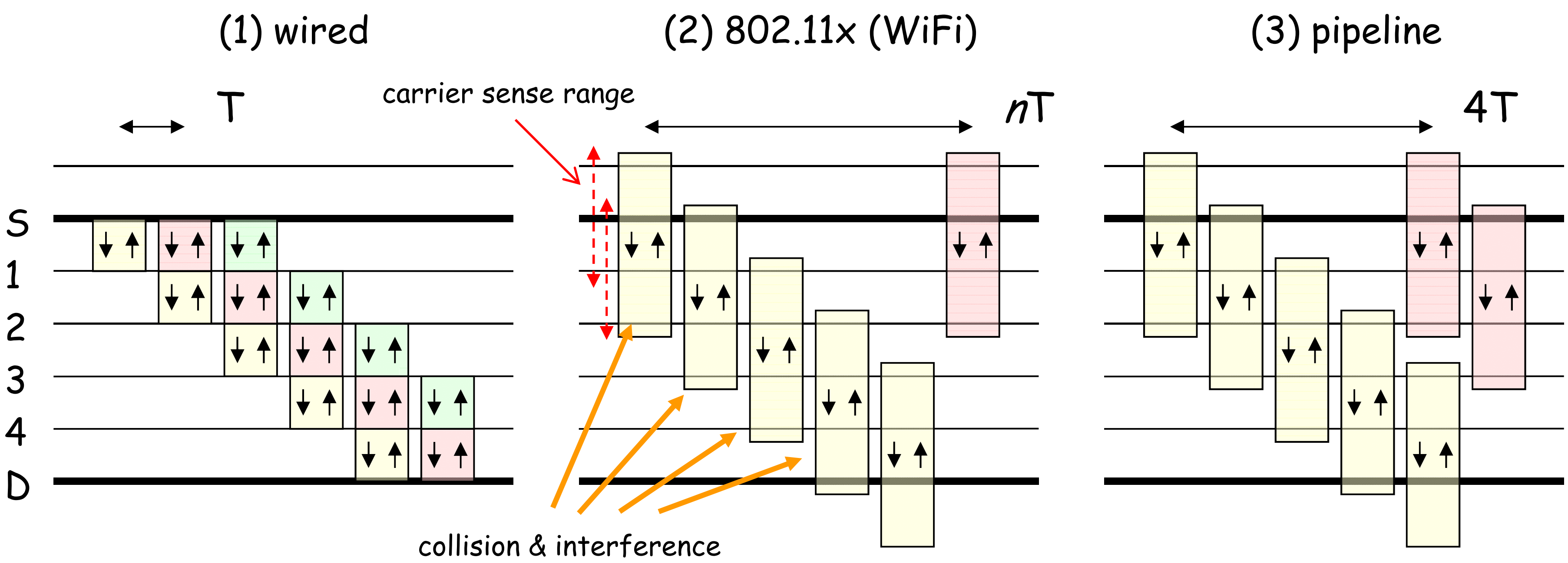
Decrease of link utilization due to radio interferences



Link utilization ratio can be at most 1/4 (or 1/*n* without pipelining, where *n* = # of hops) (J.Li et al.: *ACM Mobicom 2001*)

Small packet buffering at the intermediate nodes (Z.Hu et al: IEEE INFOCOM 2003)

Multihop Capacity (1)



channel efficiency = 1

channel efficiency = $1/n$
(n : hop count)

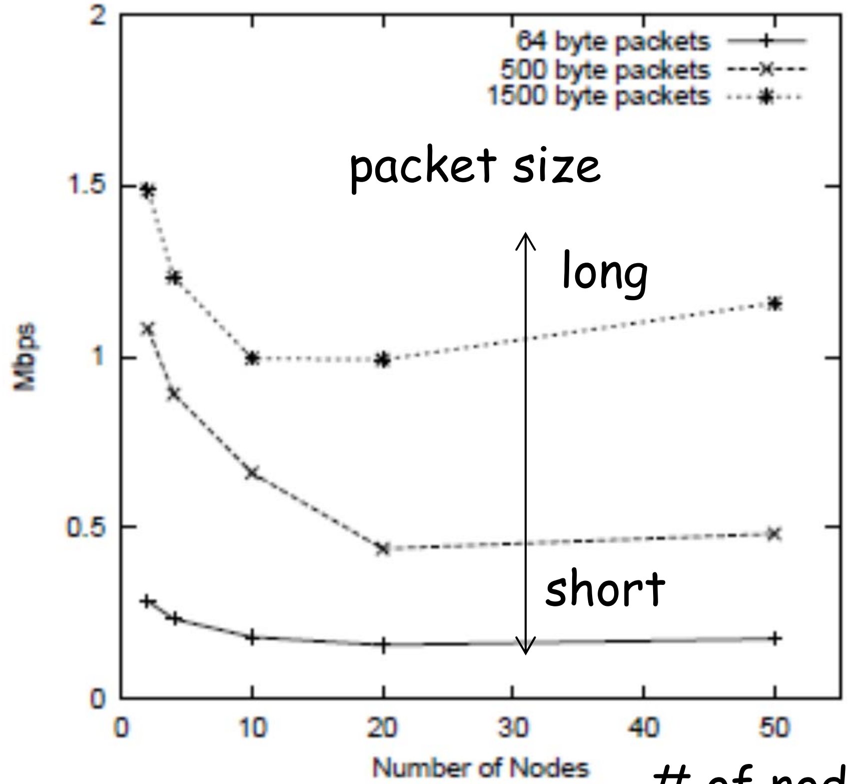
channel efficiency = 1/4
avoid collision & interference

J.Li et al: "Capacity of Wireless Ad Hoc Networks", ACM Mobicom 2001.

Multihop Capacity (2)

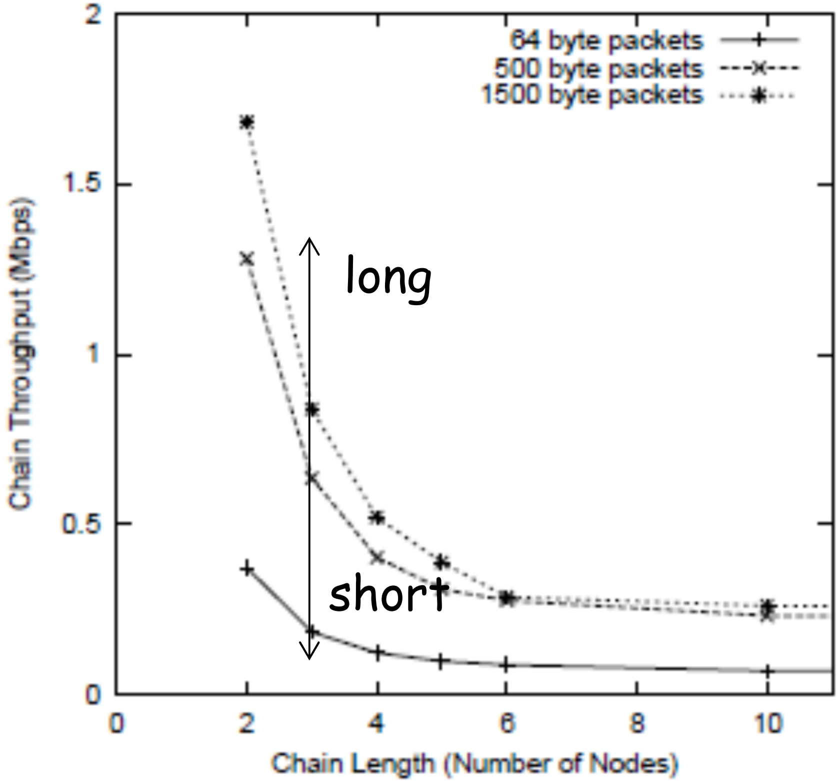
throughput

2Mb/s link



# of nodes

Cell (2D)



# of hops

Chain (1D)

J.Li et al: “Capacity of Wireless Ad Hoc Networks”, ACM Mobicom 2001.

Link RED & Adaptive Pacing

- Wireless capacity is limited by # of hops (1/4 is the theoretical maximum channel efficiency for chain topology)

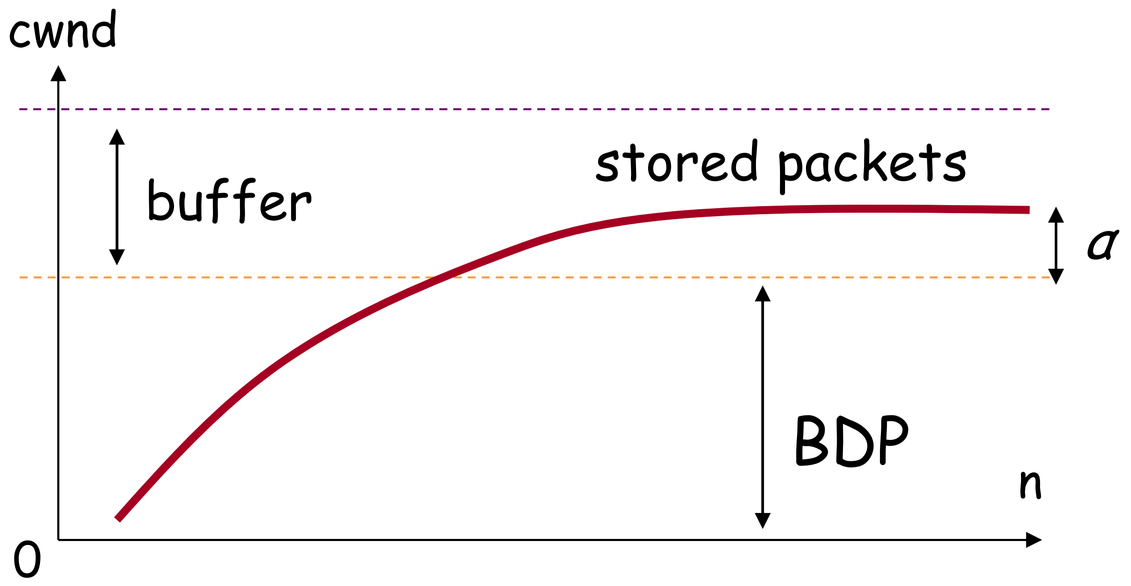
- Distributed Link RED: drops packets randomly at the link level when link load increases (analogous to random early detection)
- Adaptive Pacing: controls packet transmission scheduling in order to approach 1/4 (spatial channel reuse)

	TCP NewReno	LRED+
flow 1	244 Kbps	166 Kbps
flow 2	0 Kbps	153 Kbps
Aggregate	244 Kbps	319 Kbps
Fairness	0.5	0.9983

	TCP NewReno w/standard LL	TCP NewReno w/LL+LRED+PACING
flow 1	532 Kbps	85512 Kbps
flow 2	126229 Kbps	90459 Kbps
flow 3	115554 Kbps	70334 Kbps
flow 4	1608 Kbps	47946 Kbps
Aggregate	242923	294251
Fairness	0.51	0.95

Z.Hu et al: "The Impact of Multihop Wireless Channel on TCP Throughput and Loss", IEEE INFOCOM 2003.

TCP-Vegas (revisited)



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: $$cwnd = cwnd * 0.75$$

Vegas-W (1)

for wireless multihop

- Vegas-W [Ding, C&C 2008]
  - **Slower window increase than TCP-Vegas**

$$cwnd = \begin{cases} cwnd + 1/cwnd & (\Delta < \alpha \ \& \ n_{CA} > N_{CA}) \\ cwnd & (\alpha \le \Delta \le \beta \text{ or } \Delta \le \alpha \ \& \ n_{CA} \le N_{CA}) \\ cwnd - 1/cwnd & (\Delta > \beta) \end{cases}$$

n_{CA}: # of consecutive states entering into

$(\alpha \le \Delta \le \beta \text{ or } \Delta \le \alpha \ \& \ n_{CA} \le N_{CA})$

N_{CA}: threshold (e.g. 100)

→ much slower than TCP-Vegas

L.Ding et al.: "Improve throughput of TCP-Vegas in multihop ad hoc networks", Computer Communications, Jun.2008.

Vegas-W (2)

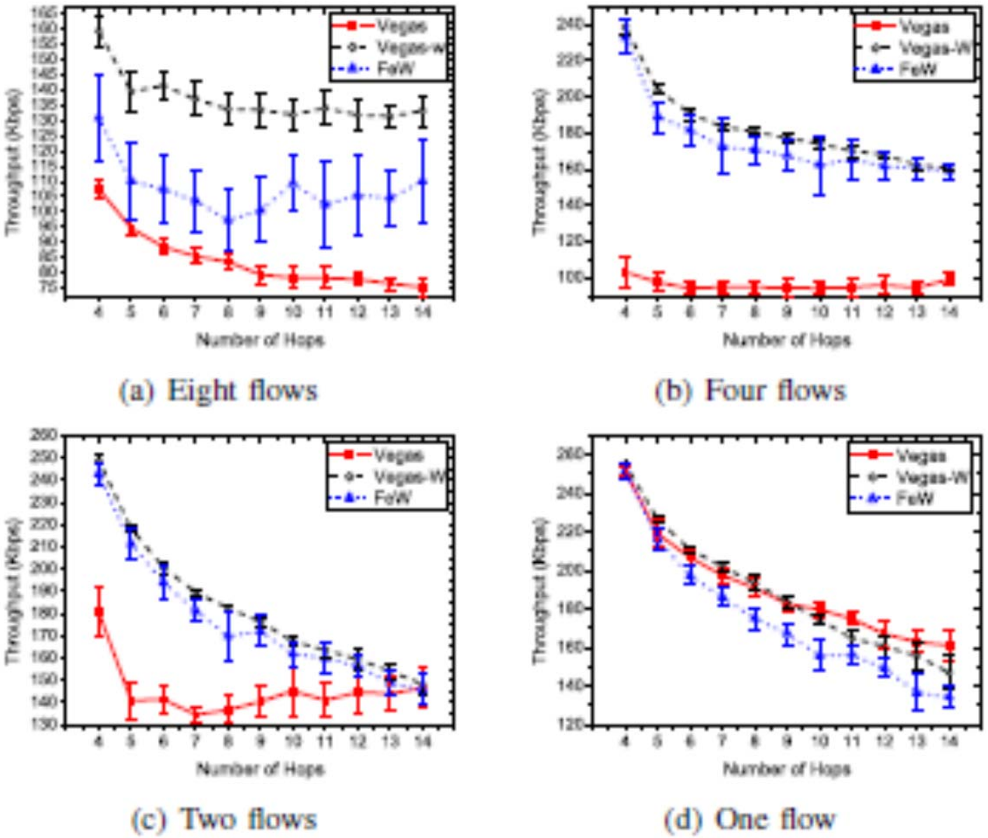


Fig. 4: Throughput comparison over chain topology with DSR and 95% confidence interval.

FeW: Fractional Window Increment (ACM Mobihoc 2005)

Vegas degraded as # of flows increases

Vegas-W improves as # of flows increases

L.Ding et al.: “Improve throughput of TCP-Vegas in multihop ad hoc networks”, Computer Communications, Jun.2008.

Summary of Wireless Multihop Networks

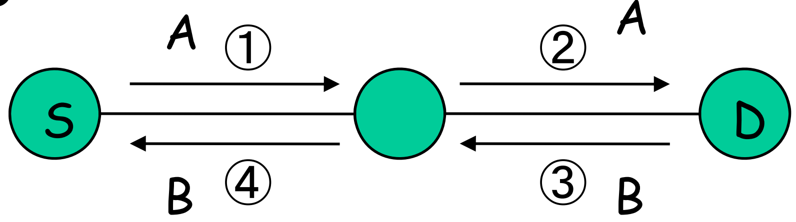
- Common to wired & wireless LAN
  - delay-based TCP is effective as long as no competing loss-based flows exist
- Gap to the wired case
  - wired case: faster window increase "immediately" fills a pipe
  - multi-hop case: slower window increase "safely" fills a pipe

A,B: symbol
a,b: signal

(ref.) Wireless Network Coding

- Multihop

# of time-slots which is necessary to transmit packets A&B between source and destination

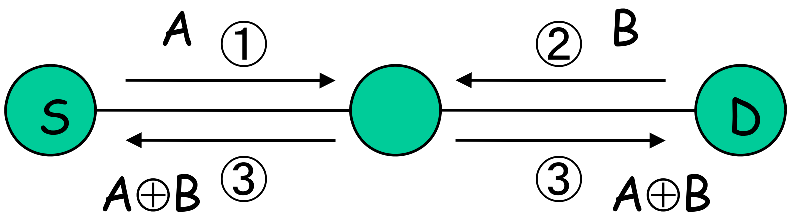


4 (channel efficiency1/4)

- Network Coding (in Wireless)

⊕: XOR

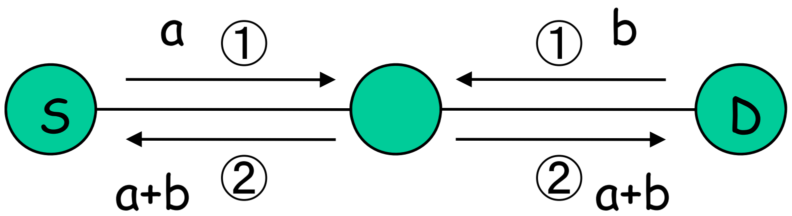
B=A⊕(A⊕B)



A=B⊕(A⊕B)

3 (channel efficiency 1/3)

- Physical-Layer Network Coding



2 (channel efficiency 1/2)

synchronization is the key point

S.Zhang: "Physical-Layer Network Coding", ACM Mobicom 2006.