画像情報特論 (2,3,4)
Advanced Image Information (2,3,4)

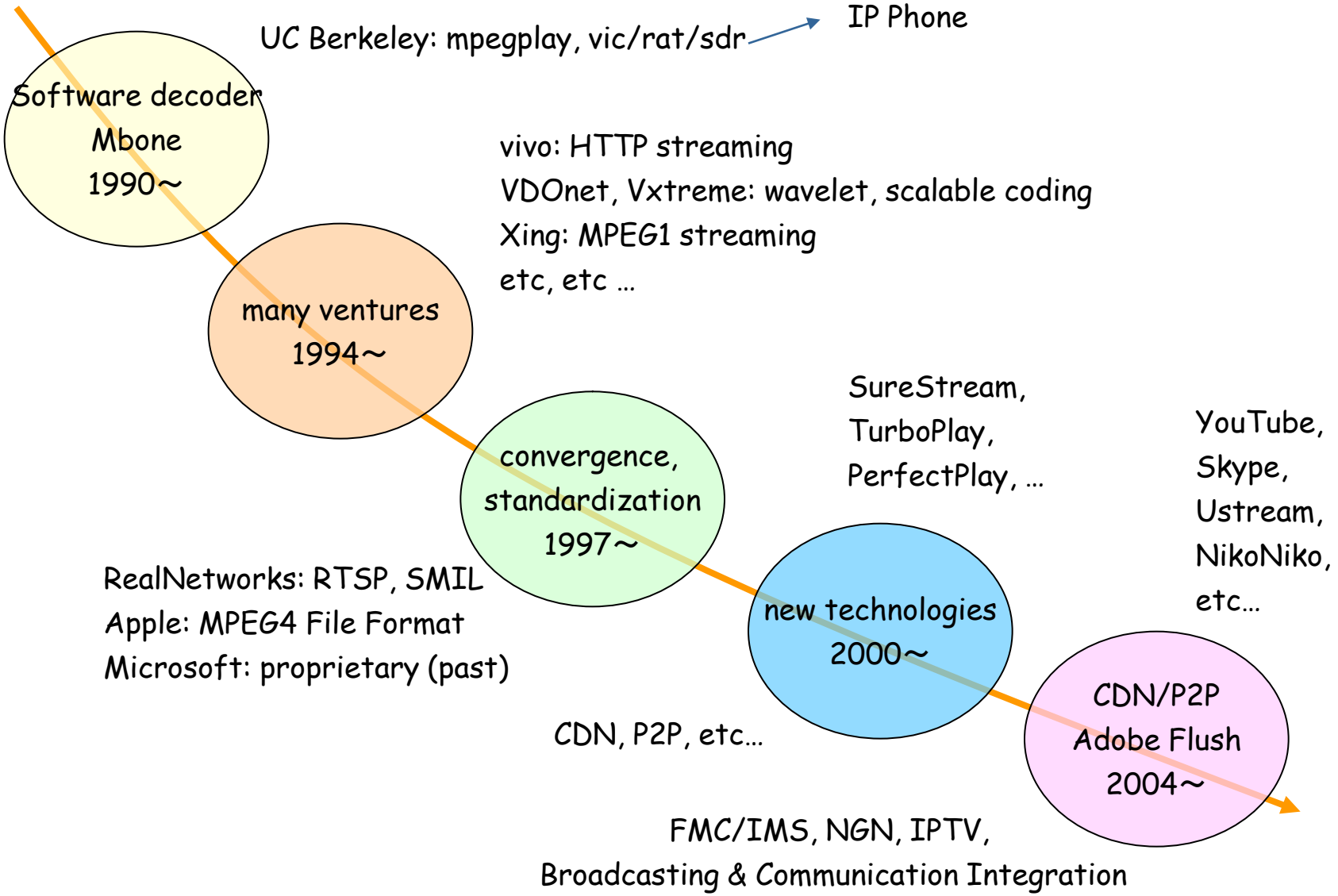
Hybrid TCP / TFRC / Wireless

情報理工学専攻　甲藤二郎

E-Mail: katto@waseda.jp

Background

History of Streaming



Protocol Stack for Streaming

- Network Architecture

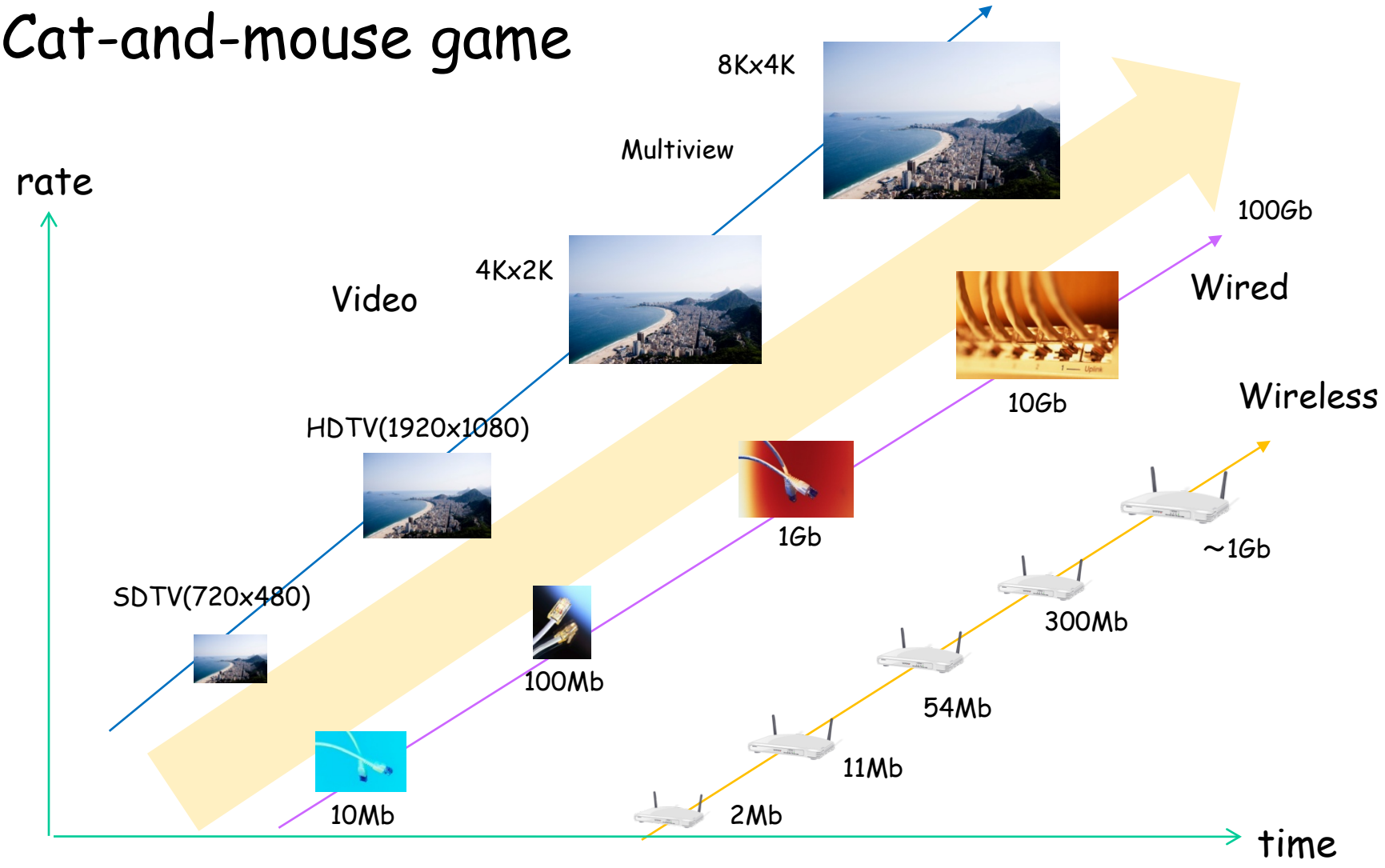
VoIP, IPTV and streaming shares almost common protocol stack

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)			

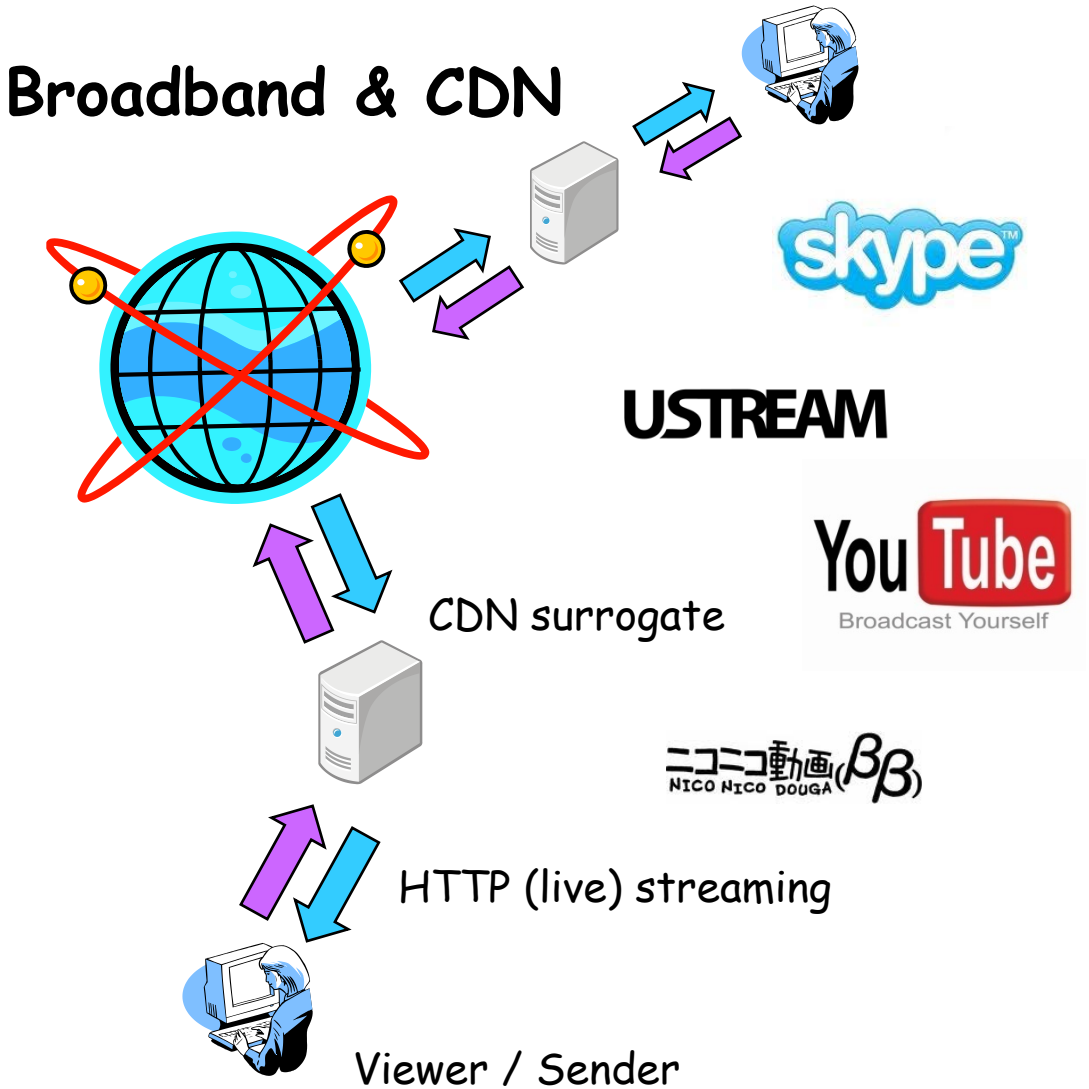
* SAP: delivered by IP-multicast for program advertisement

Networks and Multimedia

- Cat-and-mouse game



Wired Networks



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 specific problems

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

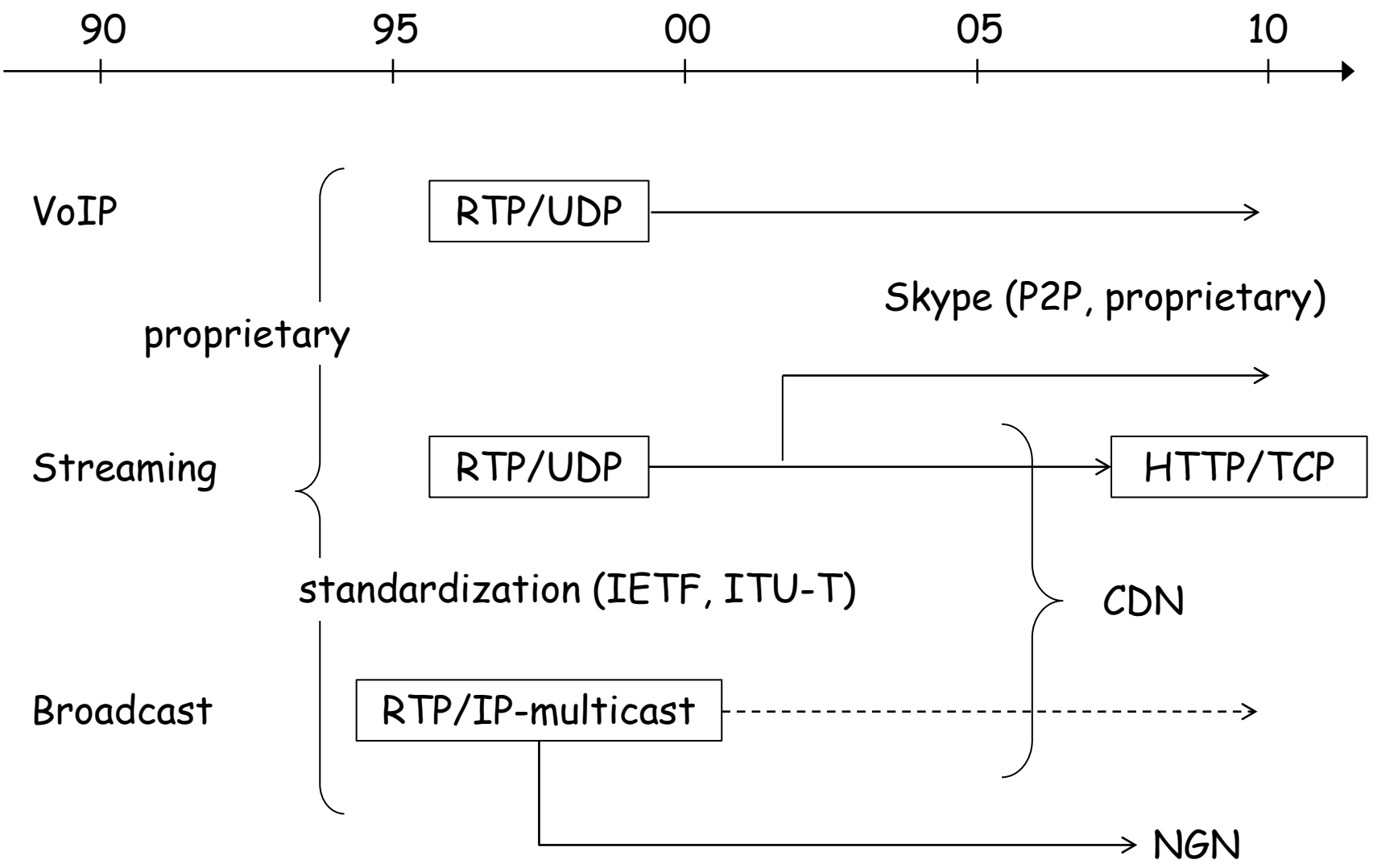
- **Wireless issues**

random errors, collisions, interference, delay increase

- **Multi-hop issues**

severe interference, lower throughput and higher delay

Protocol Transition



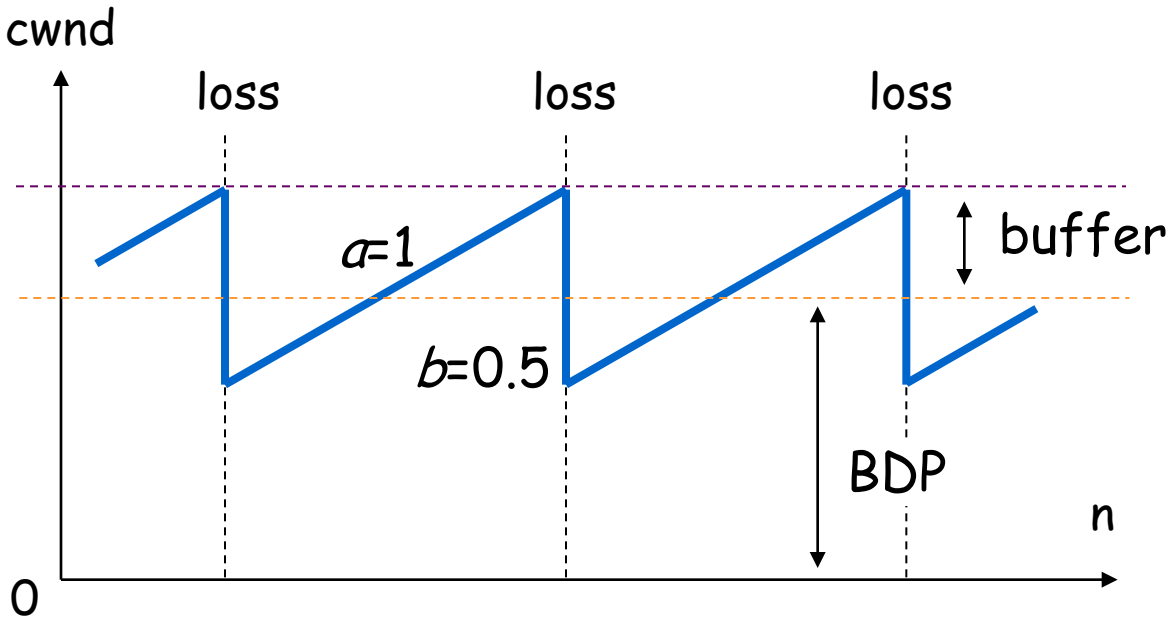
Overview

- Multimedia Delivery Techniques over IP Networks

- Broadband Wired Network
  - Transport techniques with efficiency, friendliness and low-delay properties
  - TCP and TFRC
- Wireless Network
  - Transport techniques towards low-delay transmission over wireless networks
  - Mobility
  - Multi-hop
  - Underwater sensor networks
- Network Simulators and Emulators

TCP Variants

TCP-Reno (loss-based)

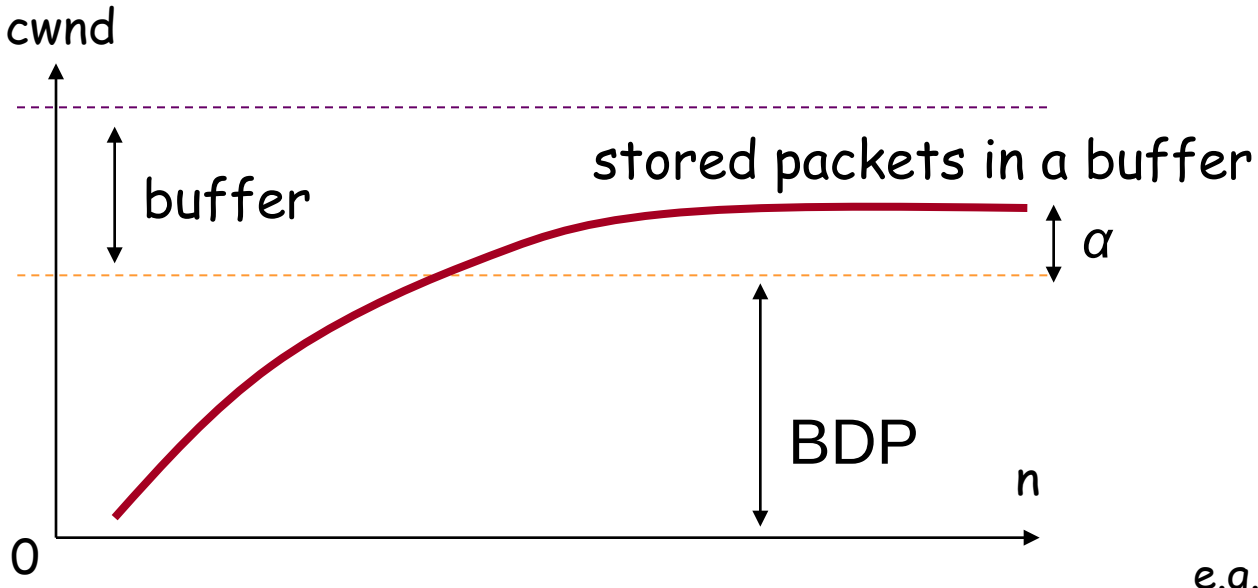


increase: cwnd = cwnd + 1/cwnd

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)

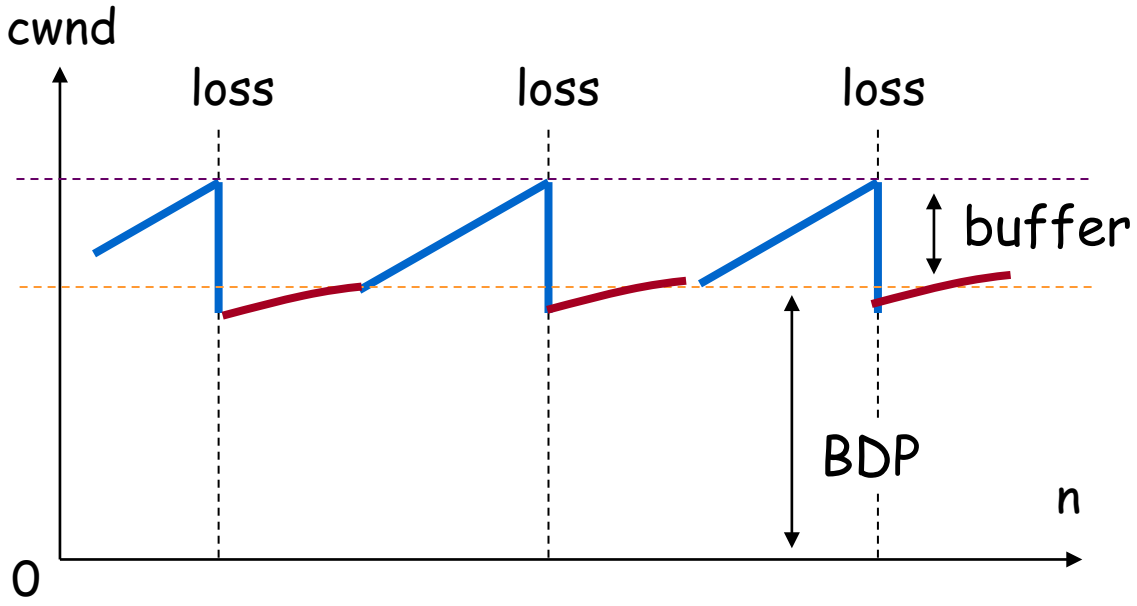
Loss-based TCPs

	Variants	Increase / Update (a)	Decrease (b)
aggressive	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	$w \leftarrow \begin{cases} w+1 & (\textit{no congestion}) \\ w & (\textit{stable}) \\ w-1 & (\textit{early congestion}) \end{cases}$	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

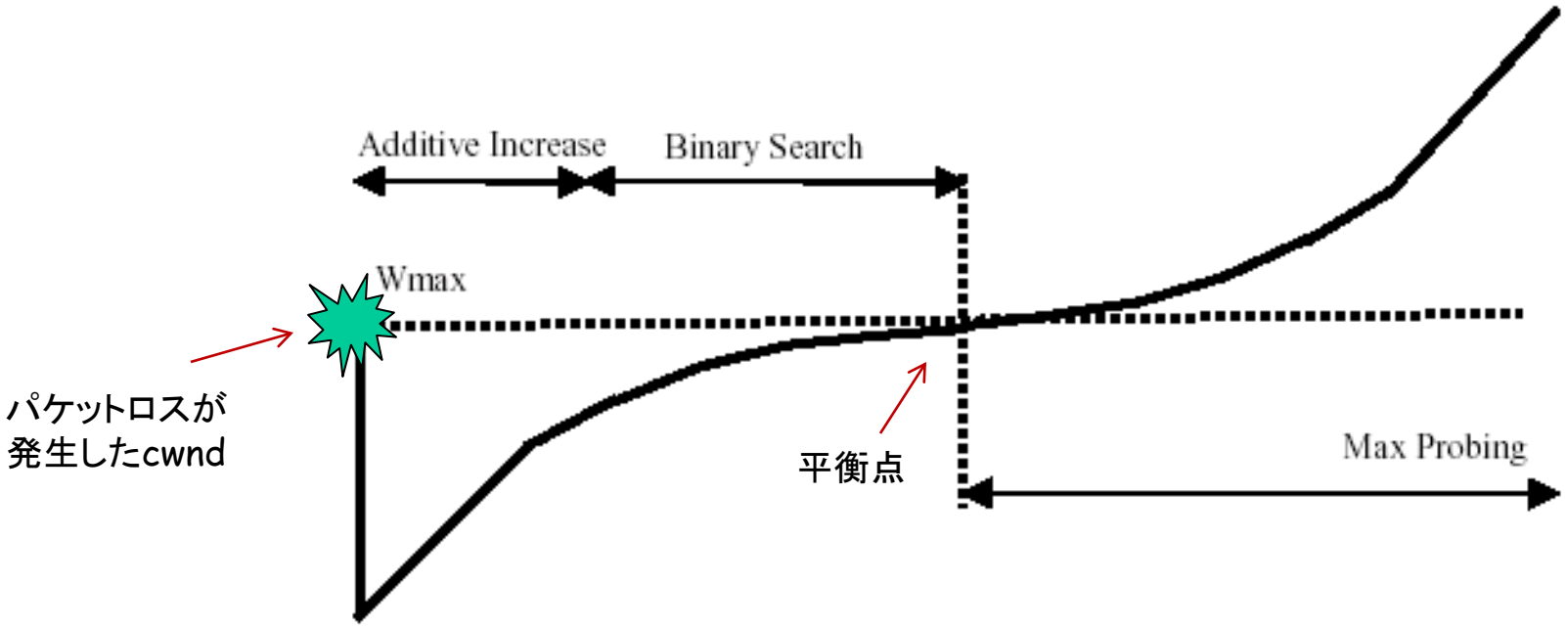
Hybrid TCPs

	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	$\begin{cases} 1 & (\textit{non congestion loss}) \\ 0.5 & (\textit{congestion loss}) \end{cases}$
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)

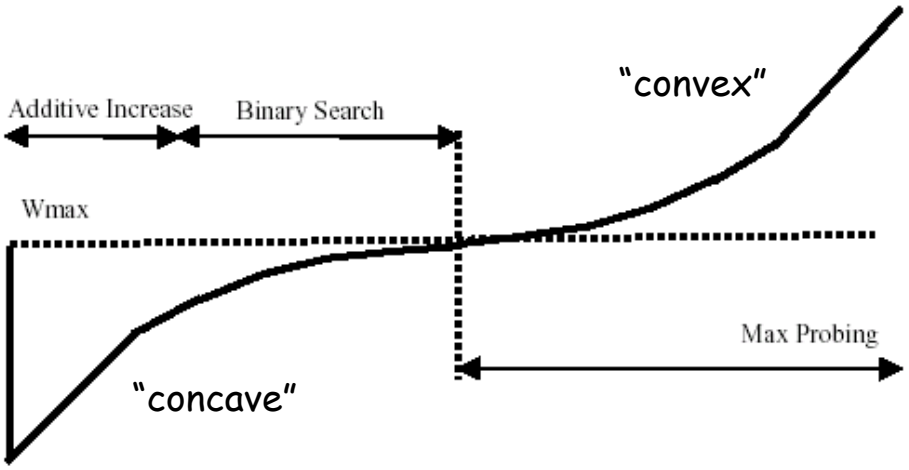
- Binary Increase Congestion Control



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

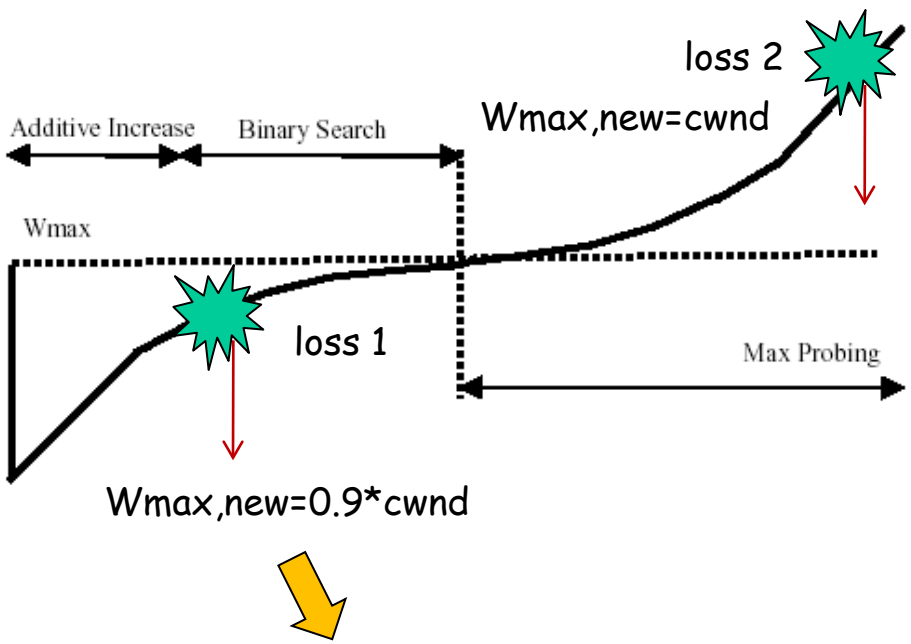
Smax: maximum increase rate (e.g. 32)

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);
```

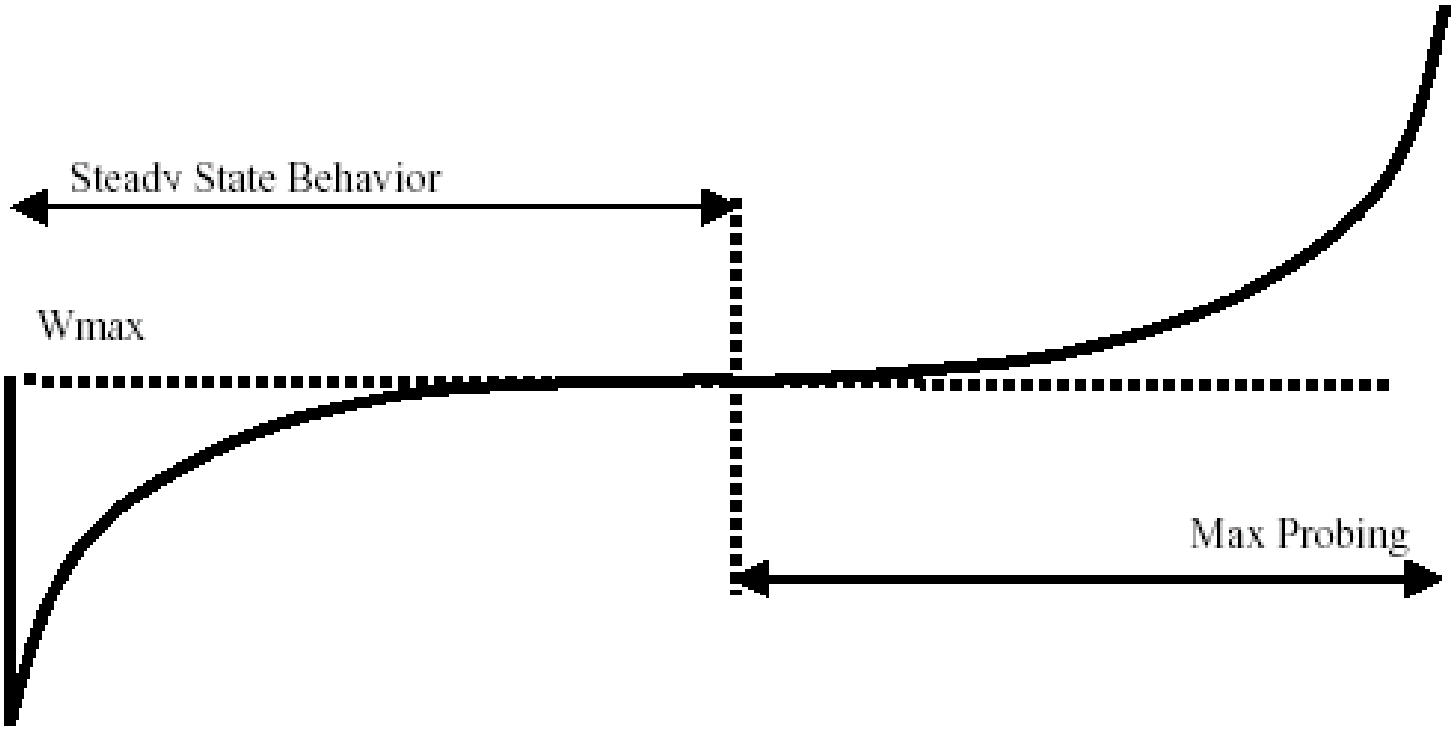
β: decrease rate (e.g. 0.2)

*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)

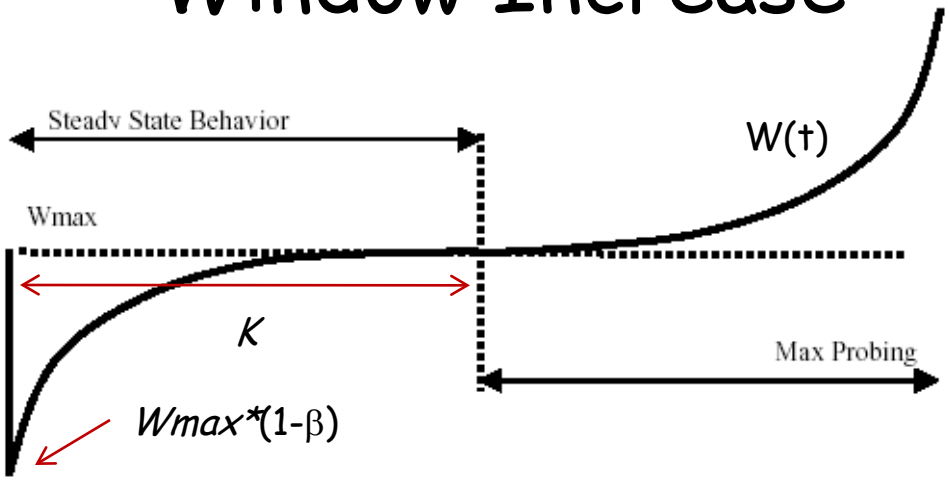
- Cubic approximation of BIC-TCP



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

CUBIC-TCP (2)

- Window Increase



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

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

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

equivalent to Reno

```
/* 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

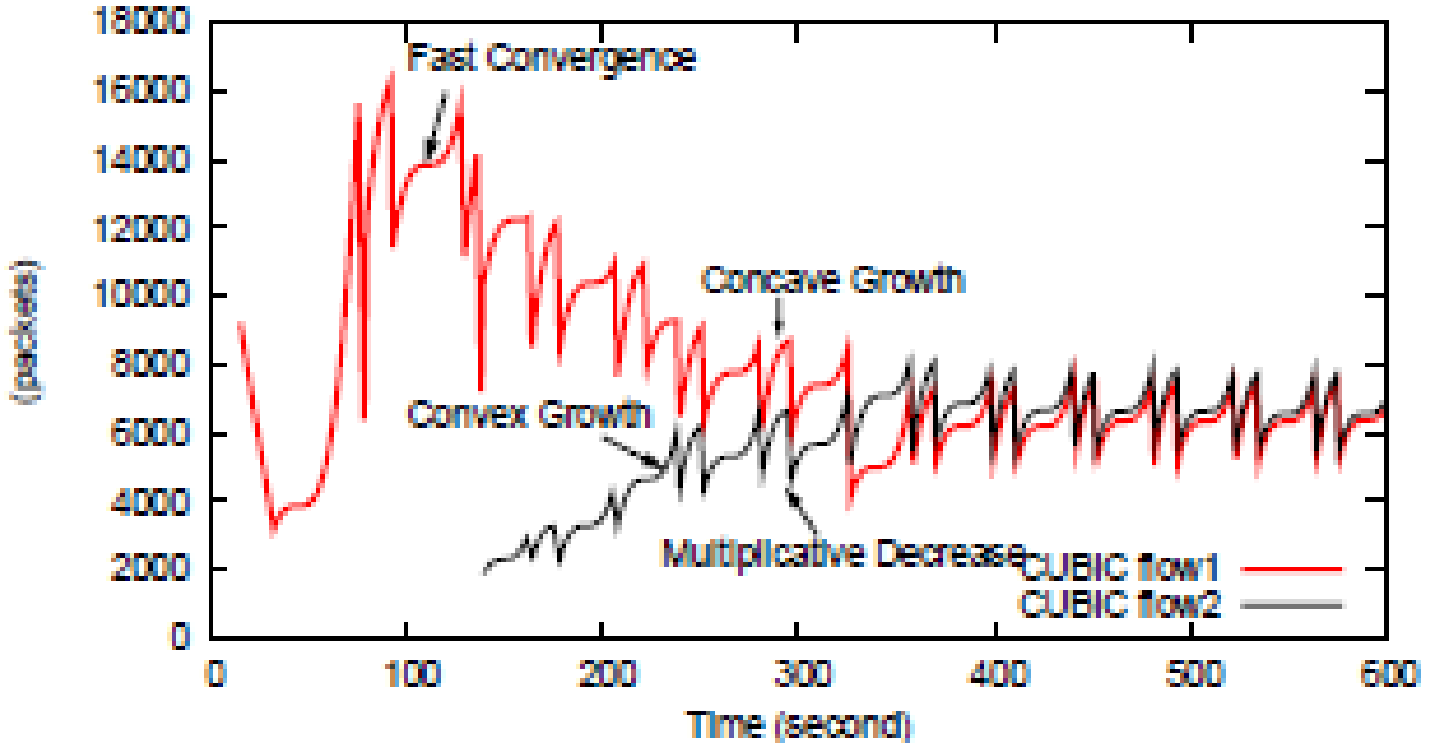
β: decrease rate (e.g. 0.2)

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)

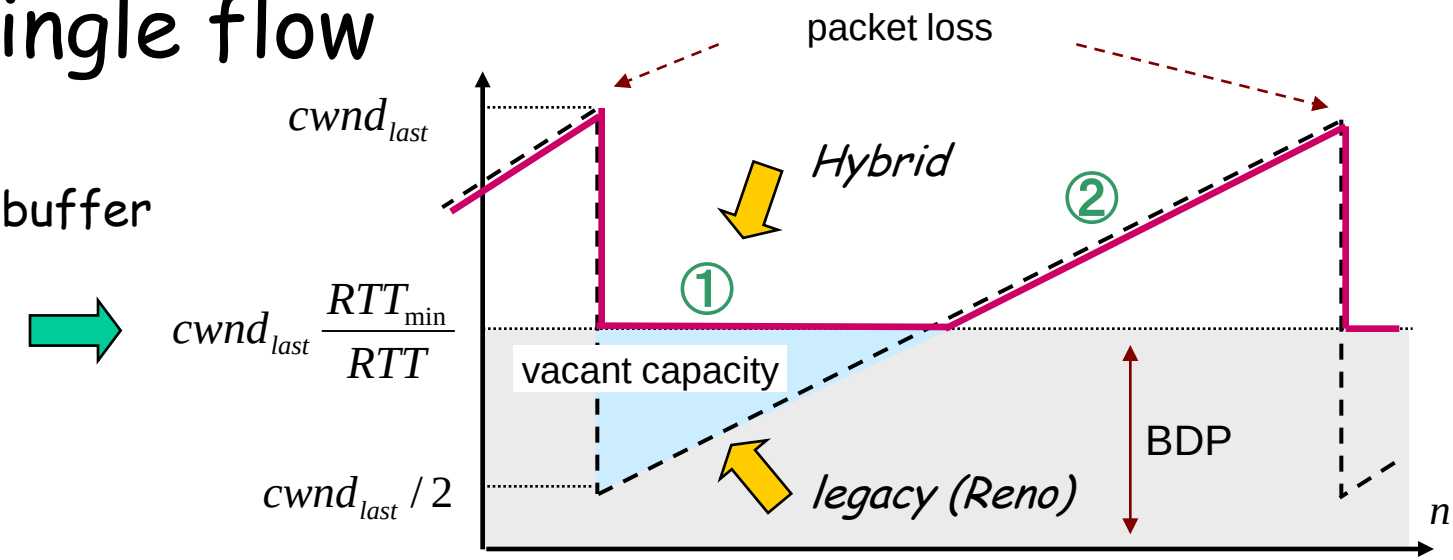
K.Munir et al: "Linux beats Windows! or the Worrying Evolution of TCP...", PFLDNet 2007.

Hybrid TCPs (and its Performance Analysis)

Hybrid TCP (1)

- single flow

clearing buffer (TCPW)



adaptive switching of two modes (loss & delay):

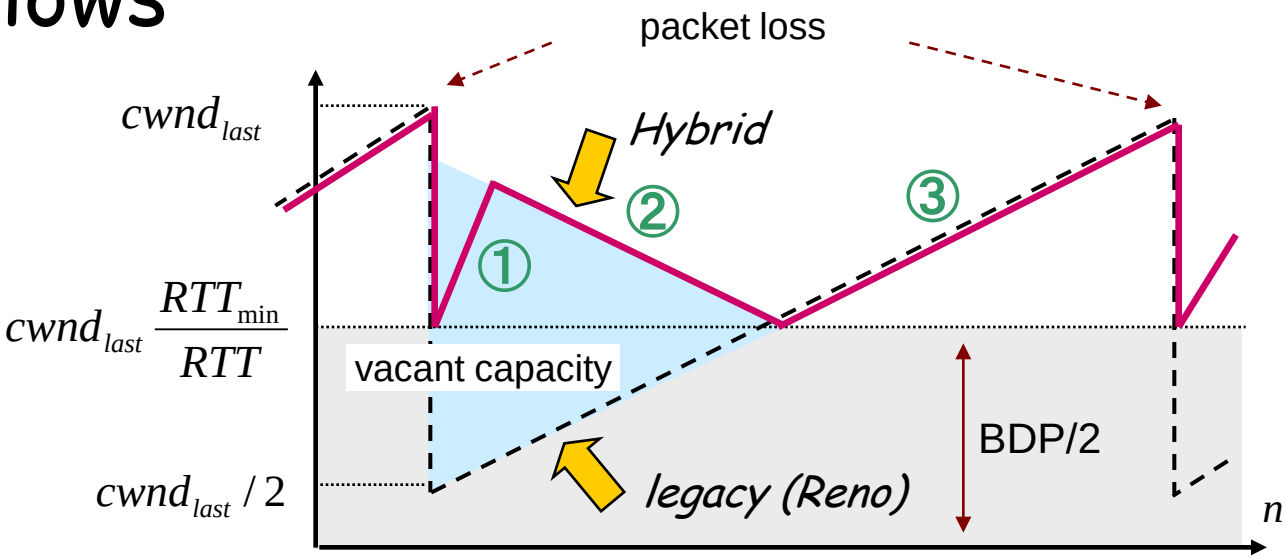
① constant rate until RTT increases (delay mode) : "efficiency" and "low delay"

② performs as Reno when RTT increases (loss mode) : "friendliness"

Hybrid TCP (2)

- two flows

clearing buffer (TCPW)



adaptive switching of two modes (loss & delay):

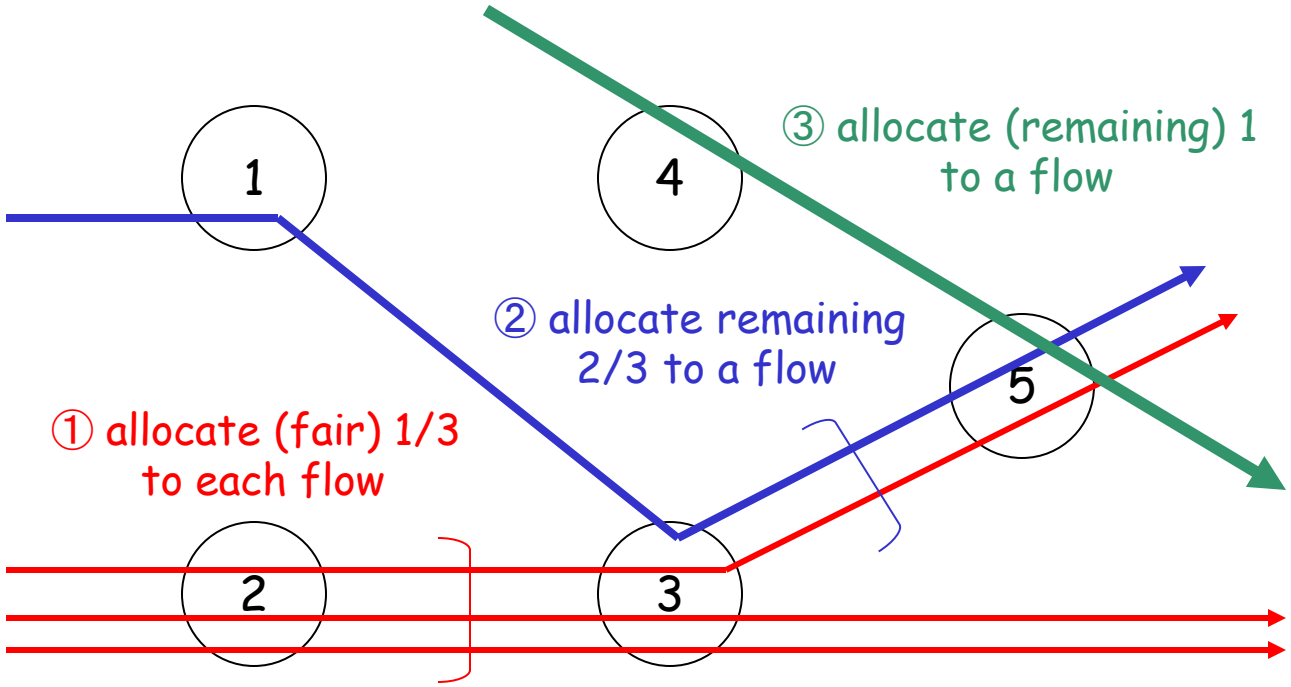
① fast cwnd increase (delay ... "efficiency")

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

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

Min-Max Fair (ideal case)

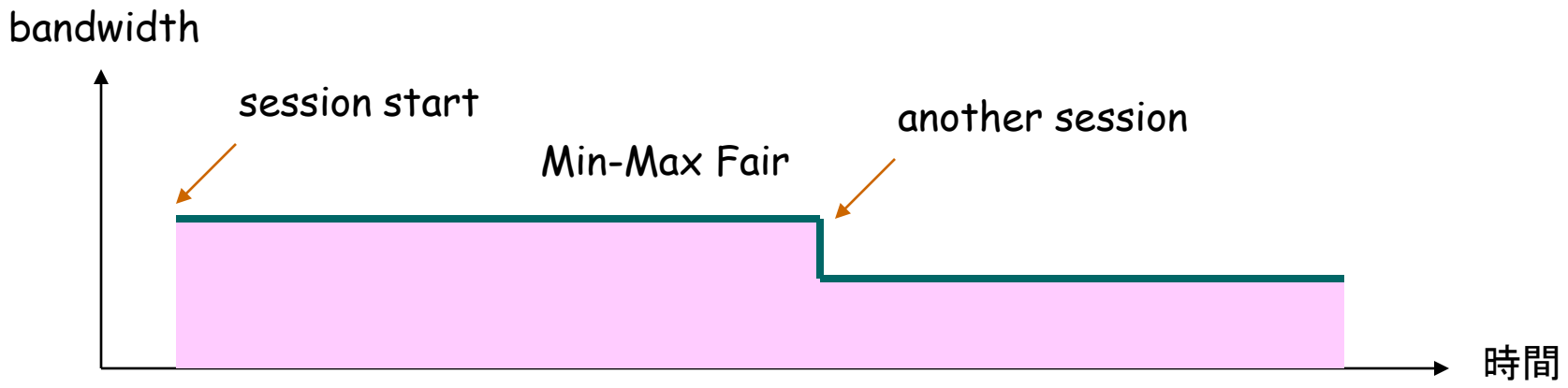
- Min-Max-Fair: allocate "maximum bandwidth" to a user who has "minimum bandwidth"



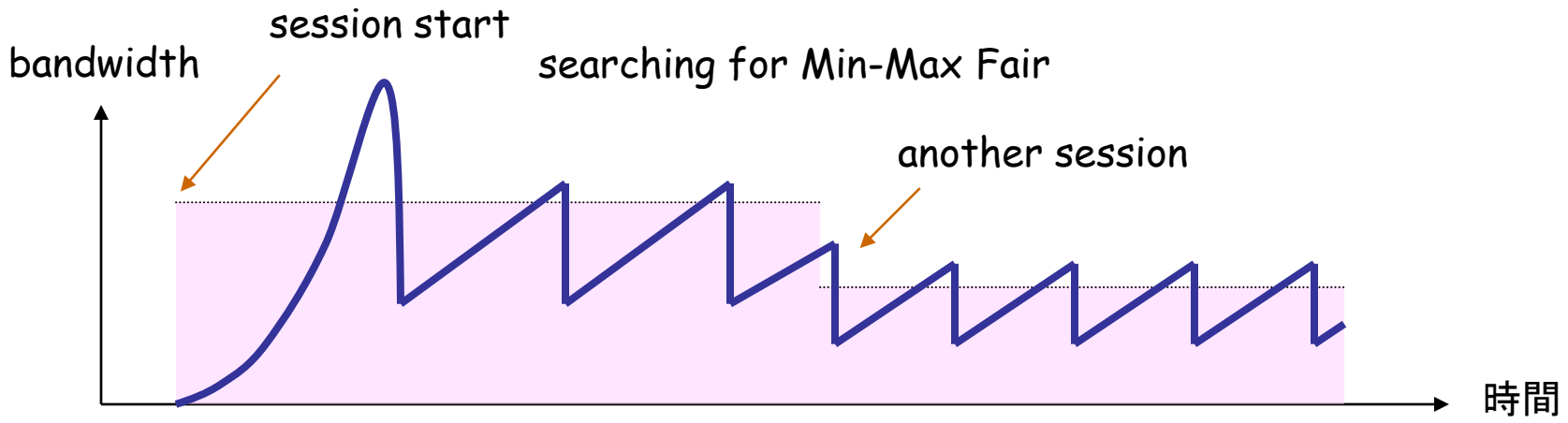
D.Bertsekas and R.Gallager: "Data Networks," Prentice Hall.

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)

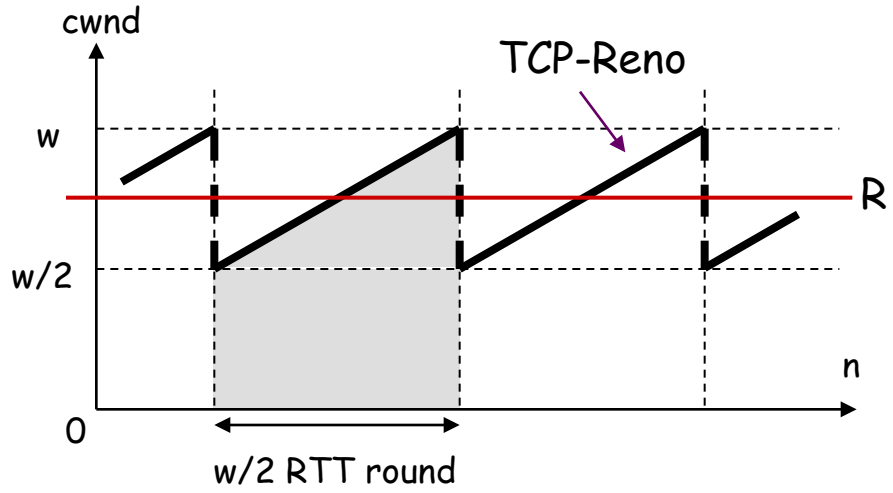
- parameter definition
  - w : cwnd when a packet loss is detected
  - W : cwnd which just fills a pipe ~ BDP
  - p : packet loss rate

- 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

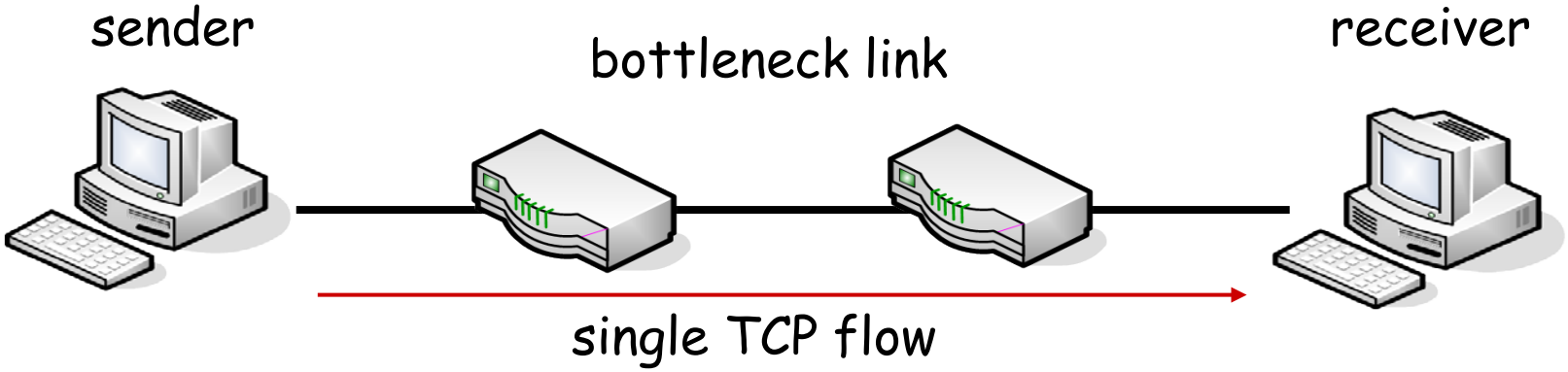
= area of a trapezoid

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

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

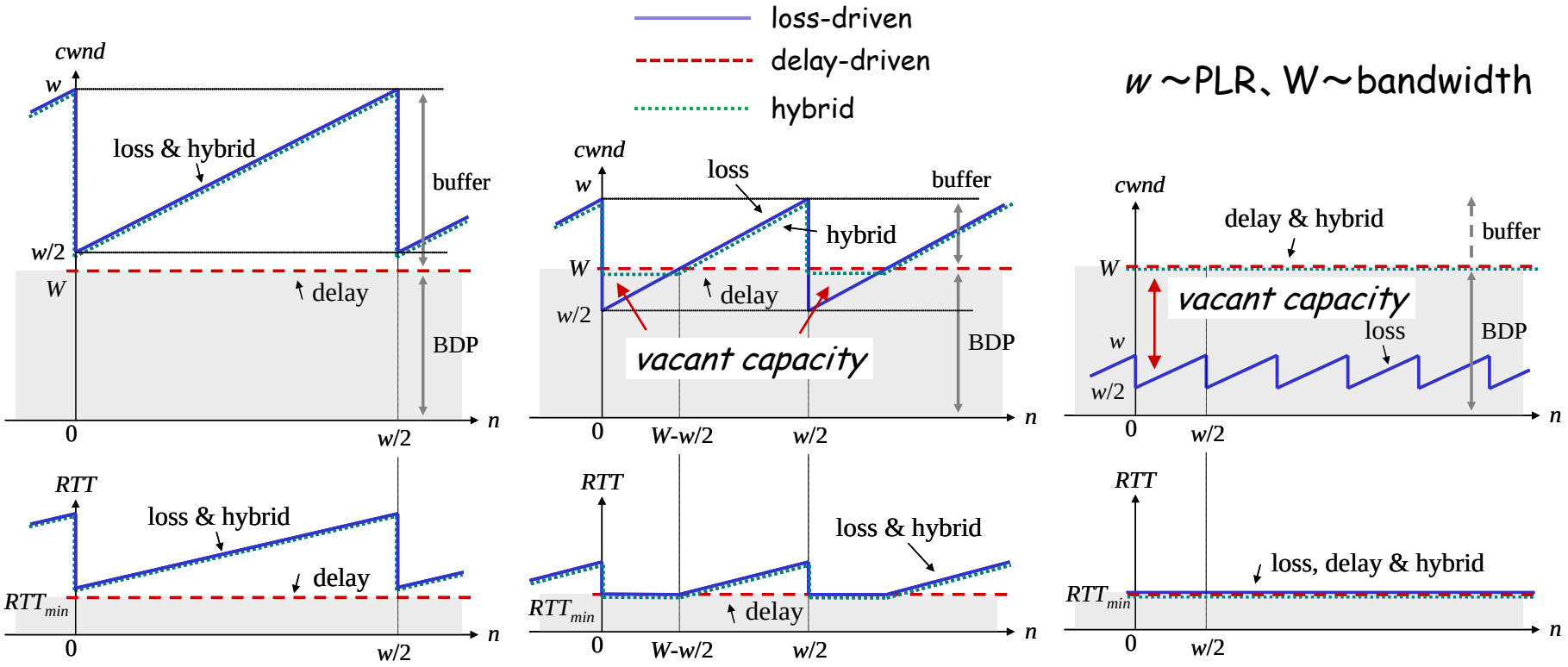
TCP behavior model (4)

- 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)

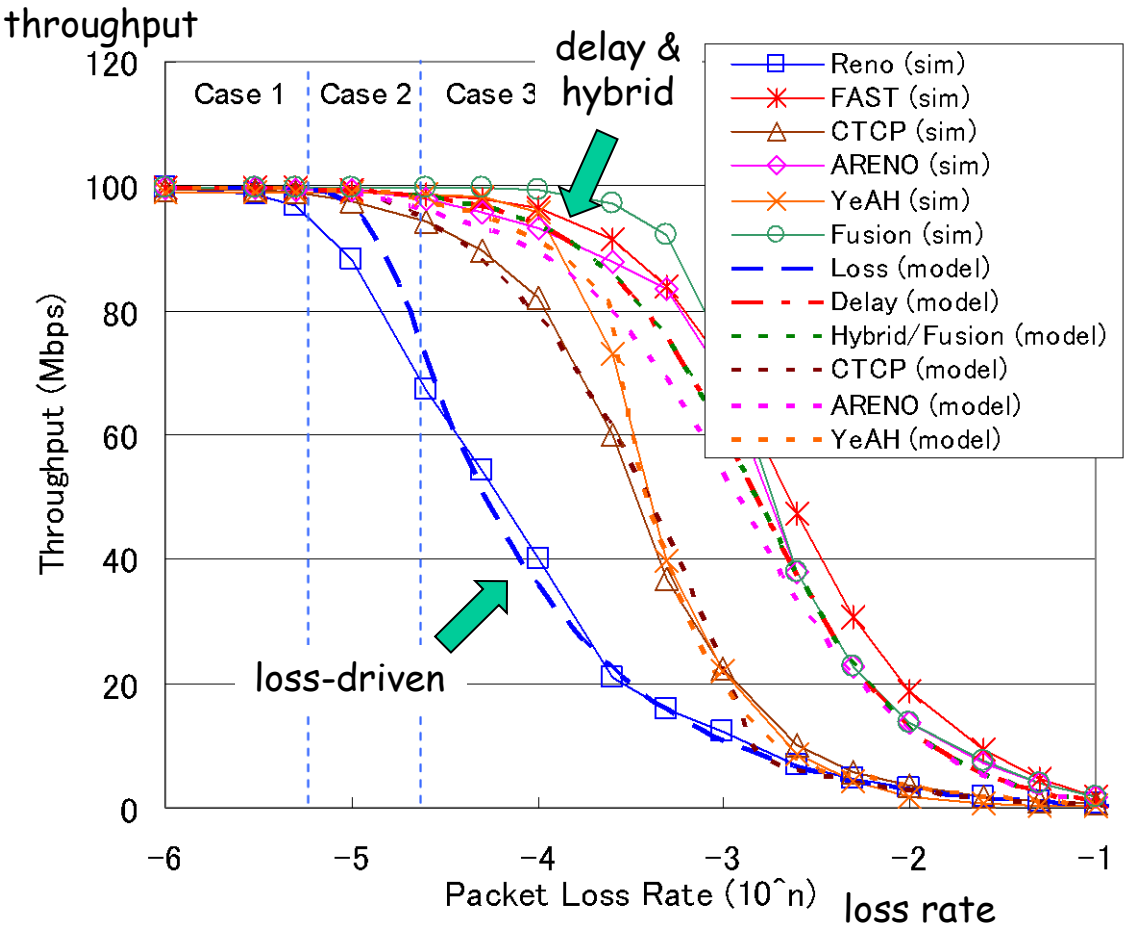
- abstraction of actual hybrids

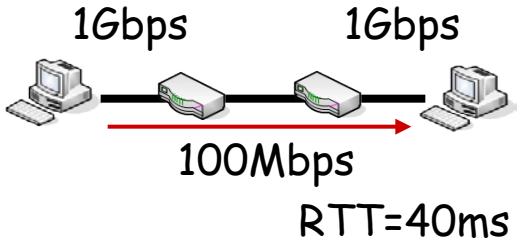
Hybrids	Window increase	Window decrease
Compound TCP	$0.125*cwnd^{0.75}$	1/2
ARENO	B/10Mbps	1/2～1
YeAH-TCP	Scalable TCP (1.01)	1/2, RTT_{min}/RTT, 7/8
TCP-Fusion	$B*D_{min}/(N*PS)$	RTT_{min}/RTT

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

TCP behavior model (8)

- evaluation by models and simulations





buffer size = BDP (constant)

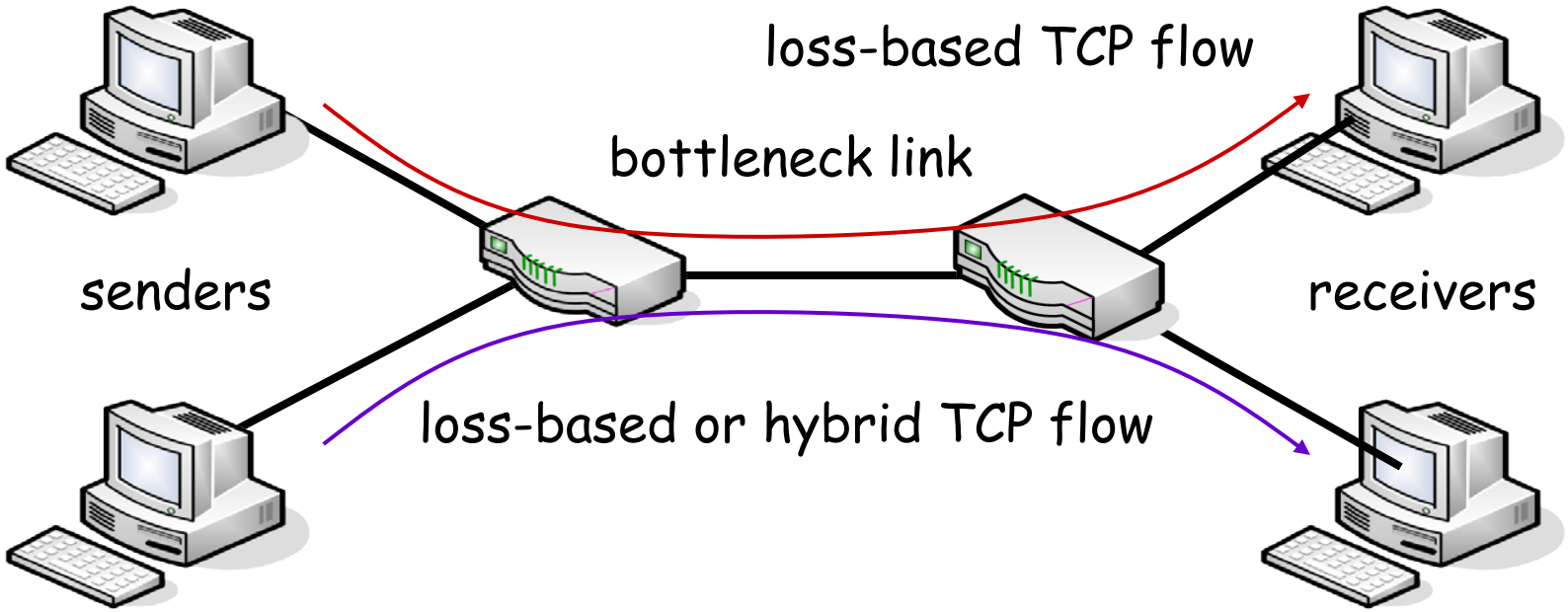
Packet loss rate : variable

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

degradation of Compound & YeAH is due to fixed window decrease

TCP behavior model (9)

- two flows (competing)

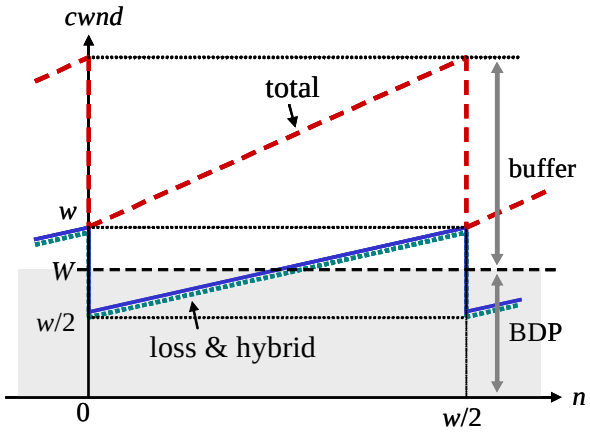


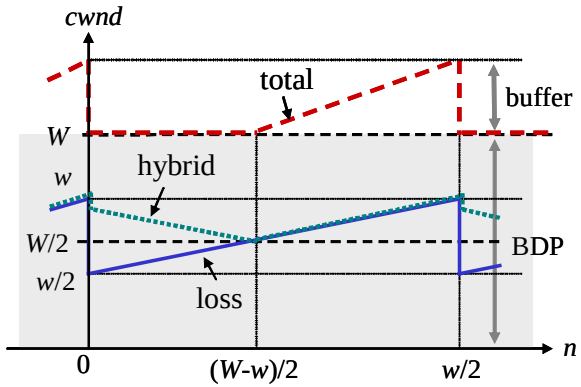
TCP behavior model (10)

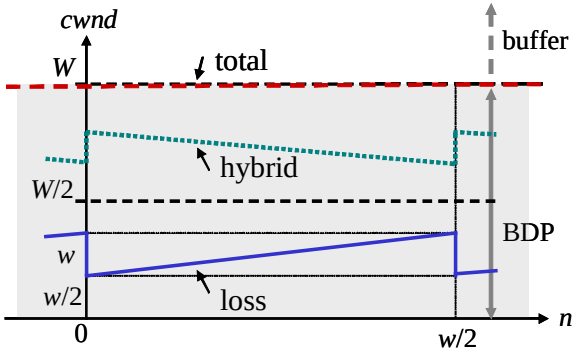
- cwnd behavior of ideal models (two flow case)

loss-driven
hybrid
total (loss + hybrid)

w ~PLR、W~bandwidth







(i) *W* < *w* (low PLR)

always buffered
(loss mode)

large buffer, small PLR

(ii) *w* < *W* < 2**w* (medium PLR)

vacant → buffered
(delay → loss)

small buffer, medium PLR

(iii) 2**w* < *W* (high PLR)

always vacant
(delay mode)

large PLR, always vacant

TCP behavior model (11)

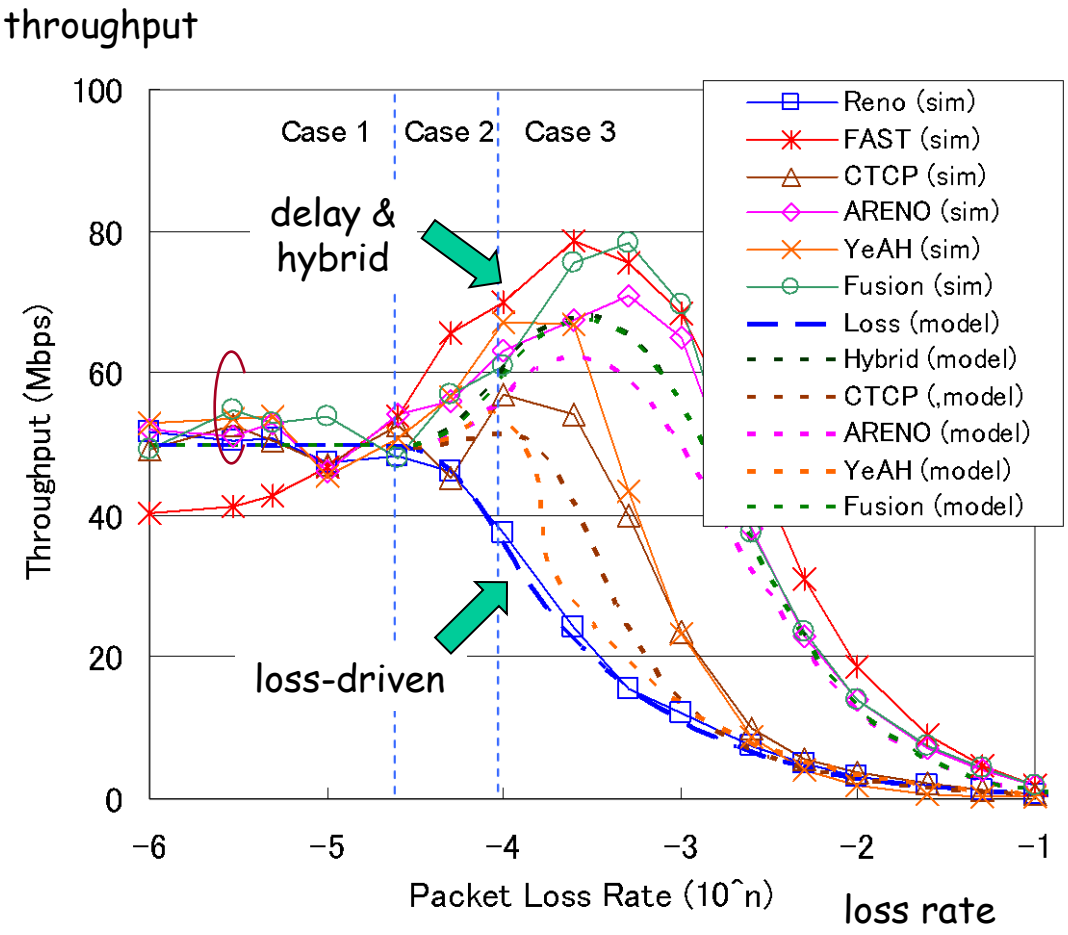
- formulation

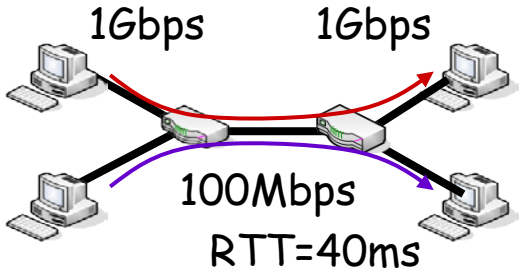
TCP	CA round	(i) $W < w$	(ii) $w \le W < 2w$	(iii) $2w \le W$
Loss	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$	$\frac{3}{8}w^2$
Hybrid	transmitted packets	$\frac{3}{8}w^2$	$\frac{3}{8}w^2 + \frac{1}{4}(W-w)^2$	$\frac{1}{2}w \cdot W - \frac{3}{8}w^2$
(common)	elapsed time	$\frac{1}{2}w \cdot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

PS: Packet size, B: Link bandwidth

TCP behavior model (12)

- evaluation by models and simulations





buffer size = BDP (constant)

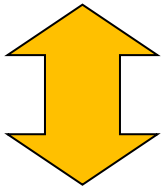
Packet loss rate : variable

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

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

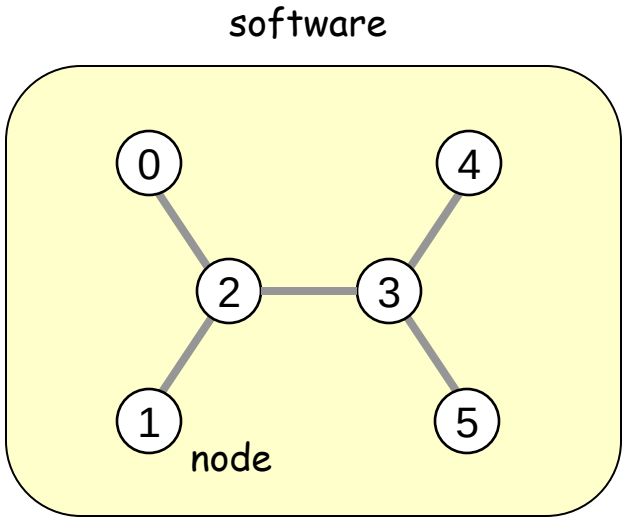
Network Simulation & Emulation

Networking Research

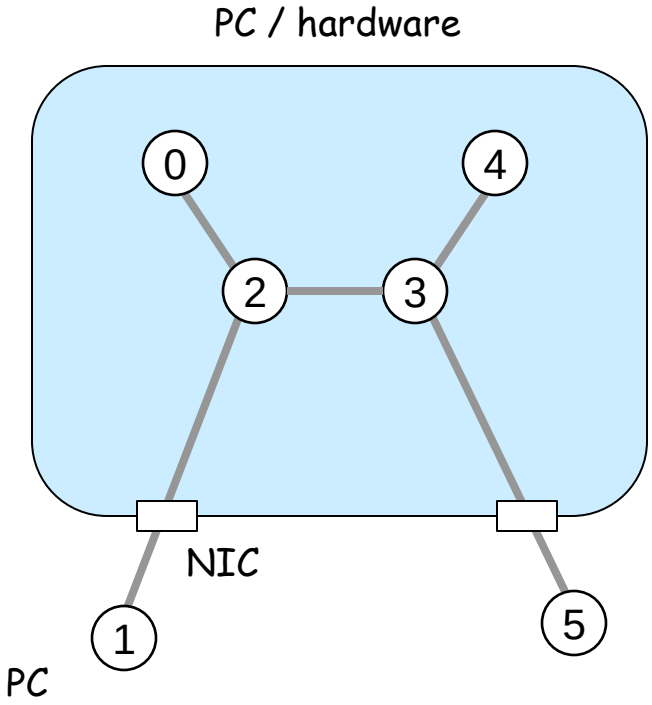
- Algorithm
- Theory (model)
- Simulation
- Emulation
- Implementation

Simulator & Emulator (1)

- simulation



- emulation



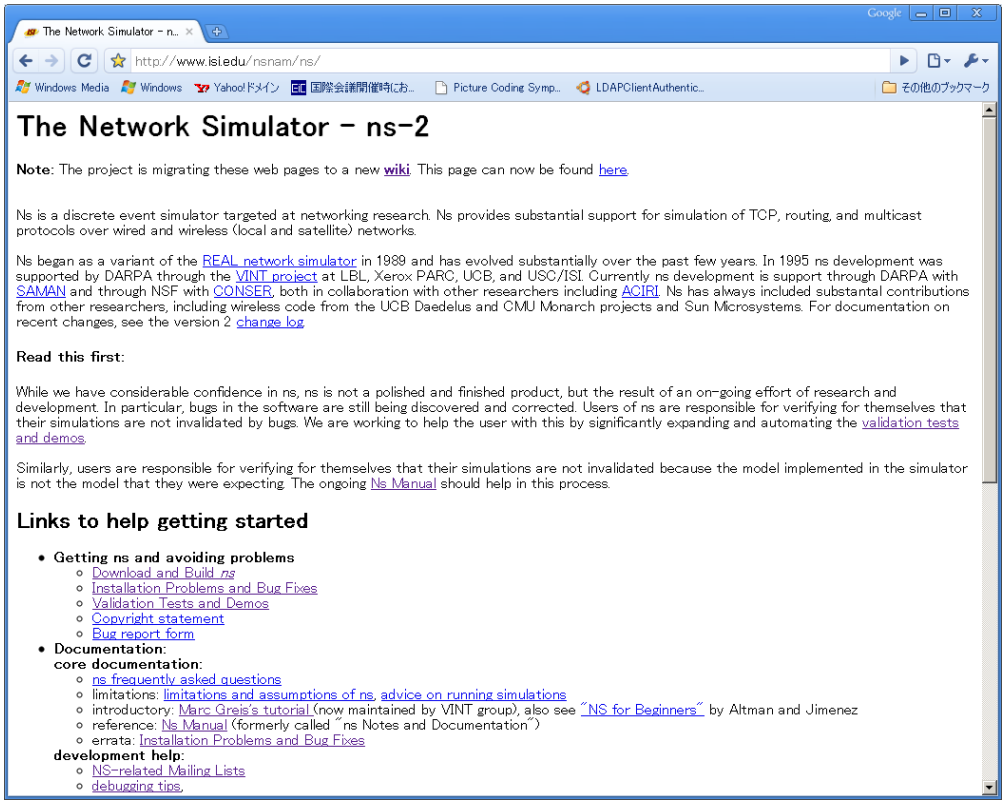
Simulator & Emulator (2)

simulator	emulator	URL
ns-2 (ns)	(nse)	http://www.isi.edu/nsnam/ns/
ns-3	nsc	http://www.nsnam.org/
OPNET		http://www.opnet.com/
Qualnet, GloMoSim	EXata	http://www.scalable-networks.com/
Scenargie		http://www.spacetime-eng.com/
	PacketStorm	http://www.packetstorm.com/
	Cloud	http://www.shunra.com/ve-cloud.php

ns-2

Ns-2 (1)

- http://www.isi.edu/nsnam/ns/



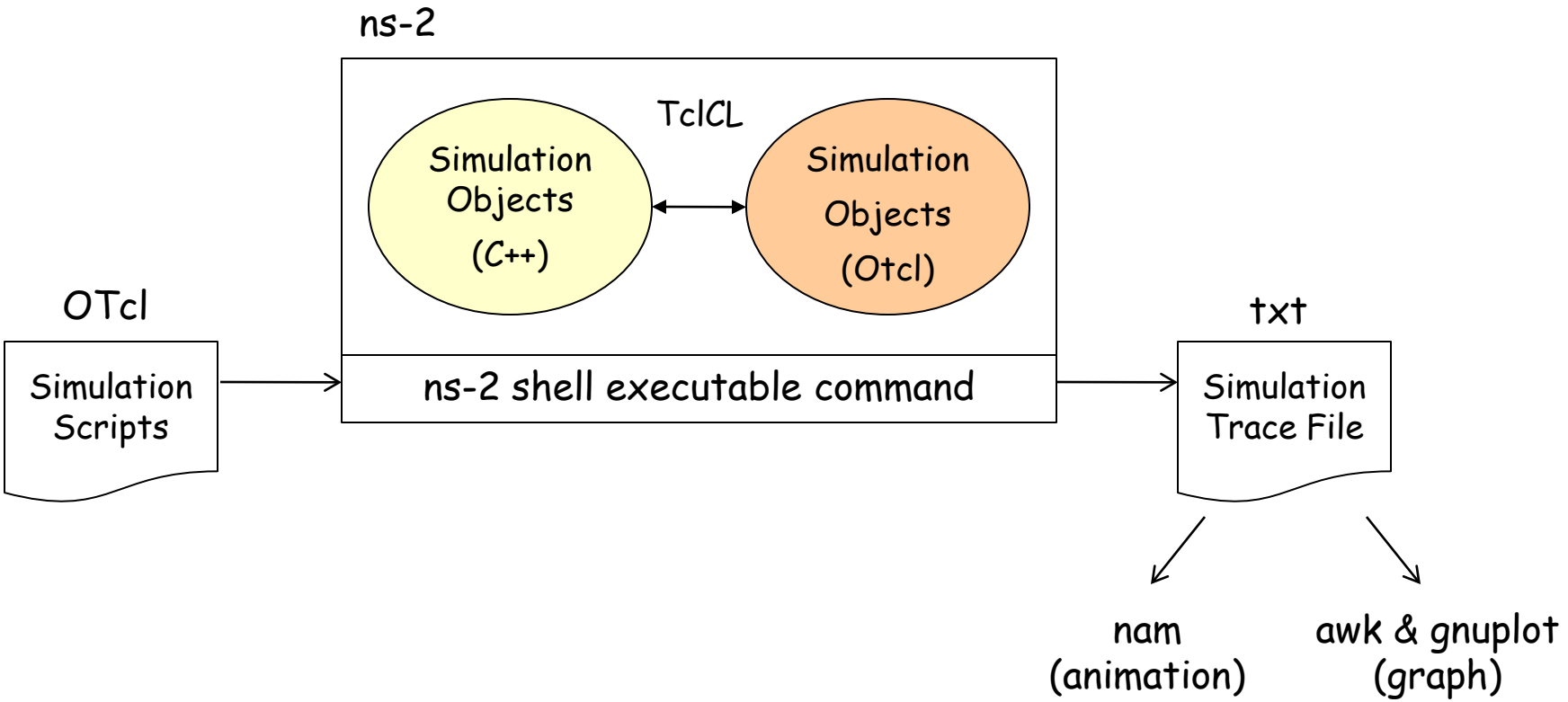
Ns-2 (2)

- download
  - 2.29 and later: http://sourceforge.net/projects/nsnam/
  - before 2.28: http://www.isi.edu/nsnam/dist/

Download “allinone”, expand、configure、and make
(Tcl/Tk, Otcl, TclCL, ns, nam)

Ns-2 (3)

- ns-2 Architecture



Ns-2 (4)

- Simulation scripts (*.tcl)

```
# initialization
# Simulator object
set ns [ new Simulator ]
# network topology
# definition of agents and apps
# procedure definition (e.g. finish)
proc finish () …
# event definition
$ns at 1.0 "$ftp start"
# simulation start
$ns run
```

```
set ns [new Simulator]
set f [open out.tr w]
$ns trace-all $f

set n0 [$ns node]
set n1 [$ns node]
$ns duplex-link $n0 $n1 100Mb 1ms DropTail

set udp0 [new Agent/UDP]
$ns attach-agent $n0 $udp0
set cbr0 [new Application/Traffic/CBR]
$cbr0 attach-agent $udp0
…
```

Ns-2 (5)

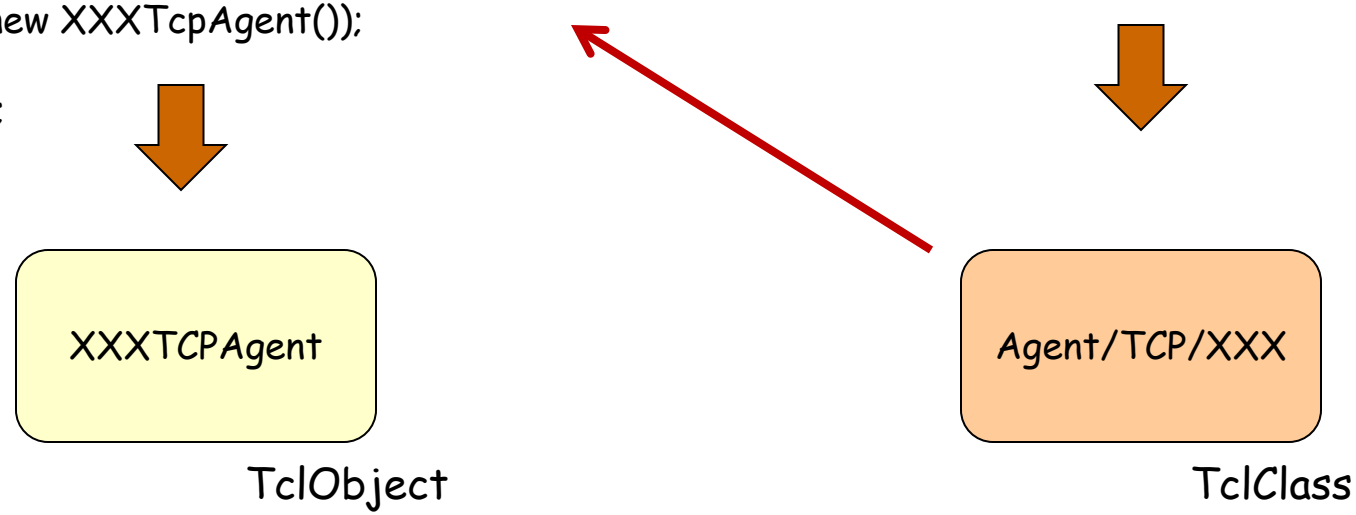
- Simulation Objects (C++/OTcl)

C++

```
static class XXXTcpClass : public TclClass {
public:
   XXXTcpClass() : TclClass("Agent/TCP/XXX") {}
   TclObject* create(int, const char*const*) {
      return (new XXXTcpAgent());
   }
} class_XXX;
```

OTcl

```
set tcp [new Agent/TCP/XXX]
```



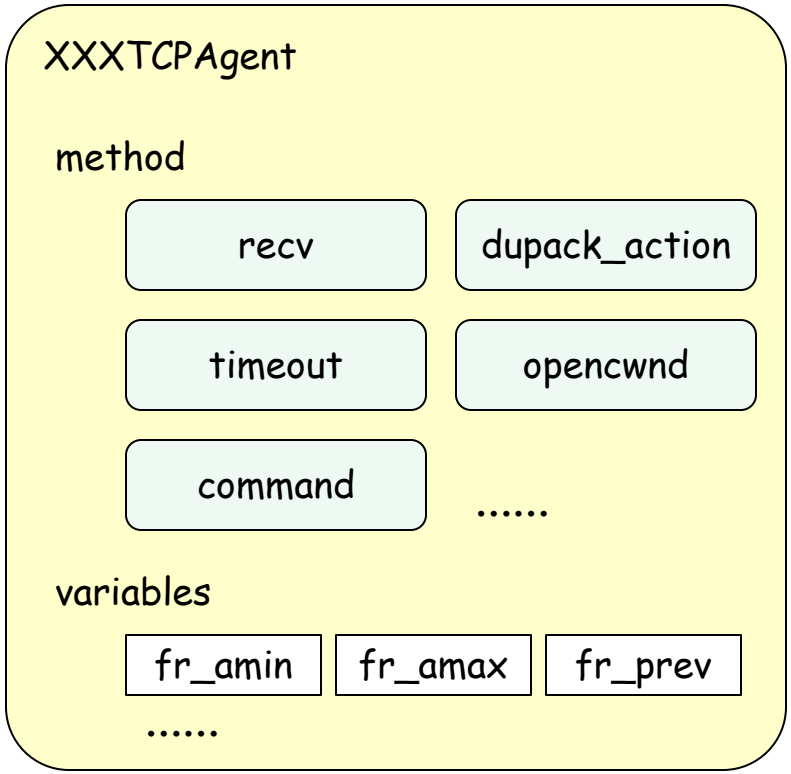
Ns-2 (6)

- Simulation Objects (C++/OTcl)

C++

```
class XXXTcpAgent : public TcpAgent {
public:
   XXXTcpAgent();
   virtual void recv(Packet *pkt, Handler*);
   virtual void dupack_action();
   virtual void timeout (int tno);
   virtual void opencwnd();
   …
protected:
   int command(int argc, const char*const* argv);

   double fr_amin_;
   double fr_amax_;
   double fr_prev_;
}
```



Ns-2 (7)

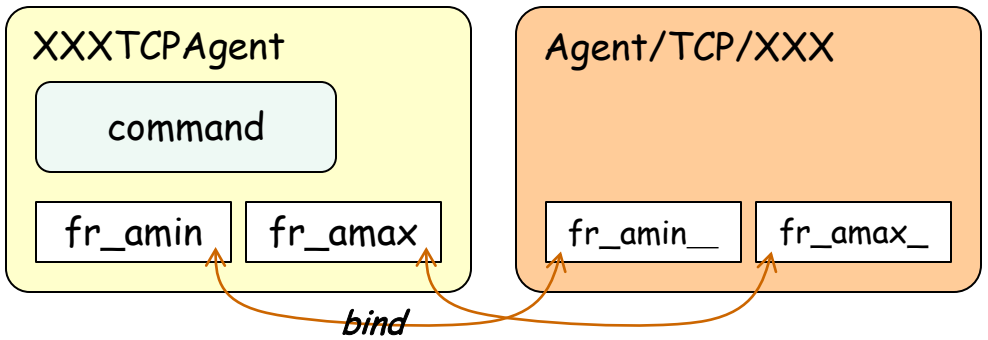
- Simulation Objects (C++/OTcl)

C++

```
XXXTcpAgent::XXXTcpAgent()
{
  bind("fr_amin_", &fr_amin_);
  bind("fr_amax_", &fr_amax_);
  …
}
XXXTCPAgent::command(int argc, const char*const* argv)
{
  if  (argc == 3) {
    if ( strcmp(argv[1], "target") == 0 ) {
      …
    }
  }
  return (NsObject::command(argc,argv));
}
```

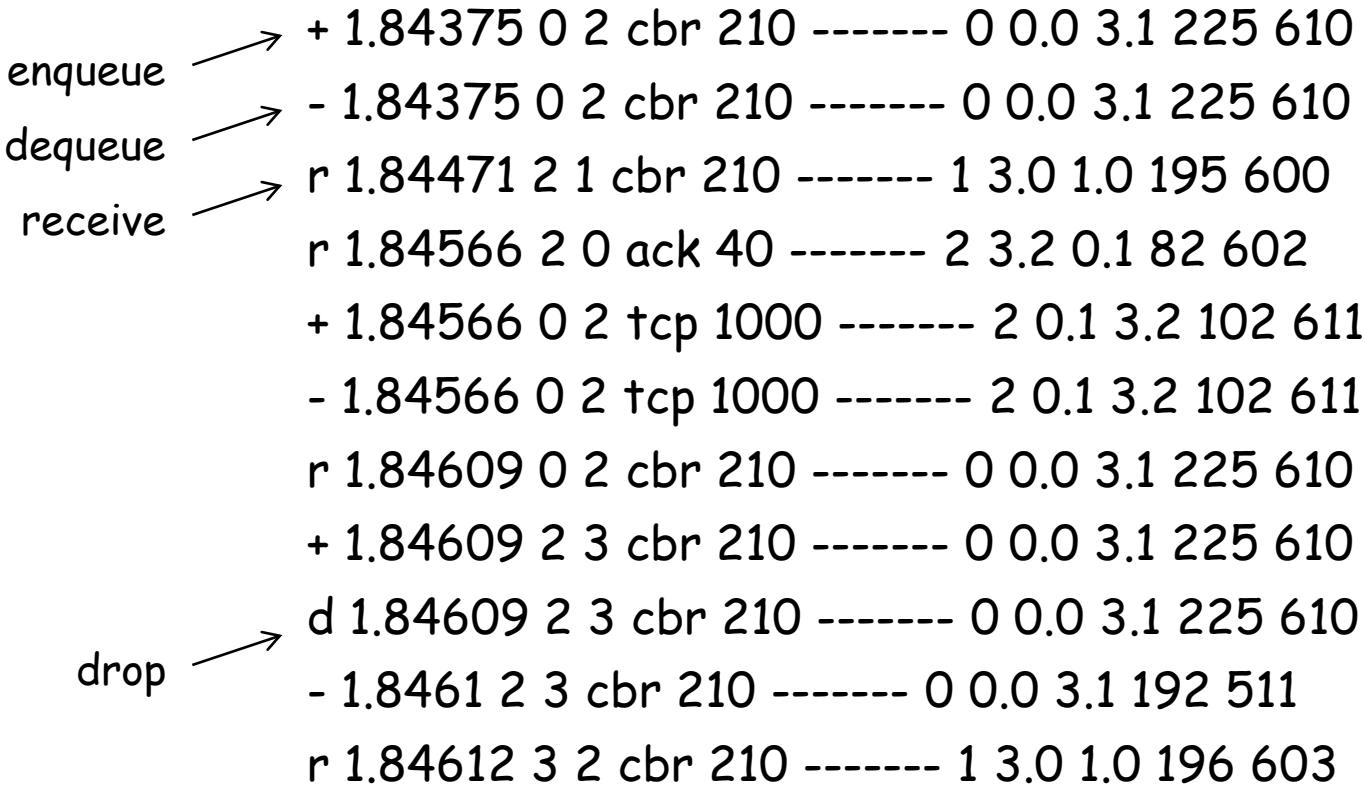
OTcl

```
set tcp [new Agent/TCP/XXX]
$tcp set fid_ 1
$tcp set fr_amin_ 0.2
$tcp set fr_amax_ 0.8
$tcp target [new Agent/Null]
```



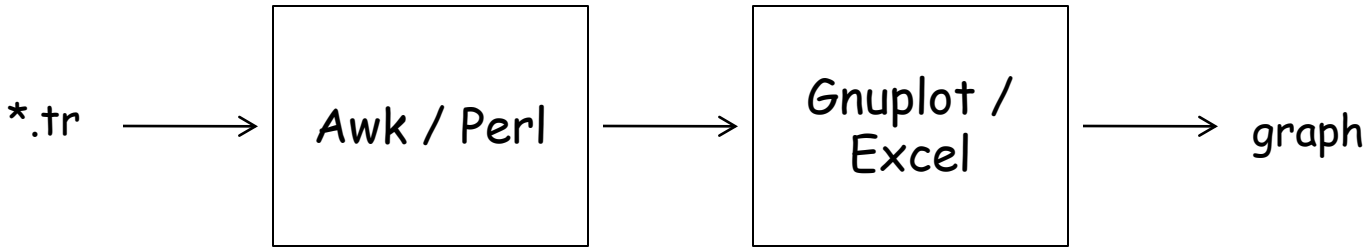
Ns-2 (8)

- Trace File (*.tr)



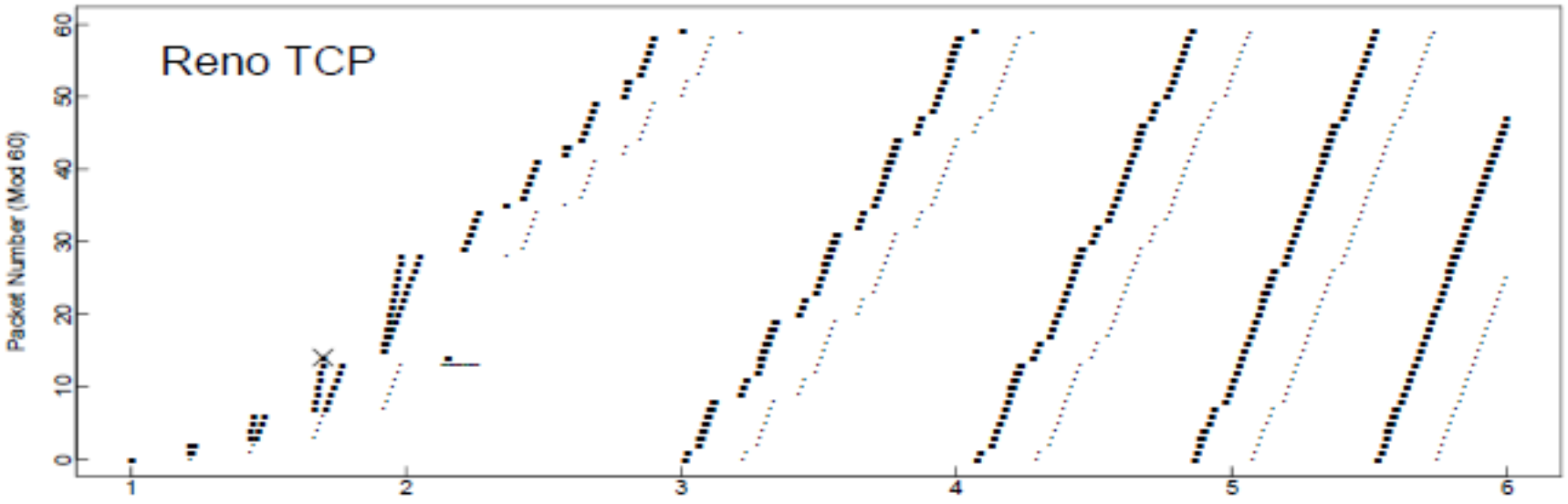
Ns-2 (9)

- Awk / Perl (script)
  - applied to trace files
  - generate graph files, e.g. for GnuPlot



Ns-2 (10)

- example



IEICE Network System Technical Group's Archive

- http://www.ieice.org/~ns/jpn/archives.html
  - 2009/8: ns-2 (summer school)
  - 2009/8: OPNET (summer school)
  - 2009/12: Qualnet (tutorial)

Googling "ns-2 tutorial" gives many sites

※ "ns-3" sites are increasing

ns-2 TCP-Linux

http://netlab.caltech.edu/projects/ns2tcplinux/

ns-2 TCP-Linux (1)

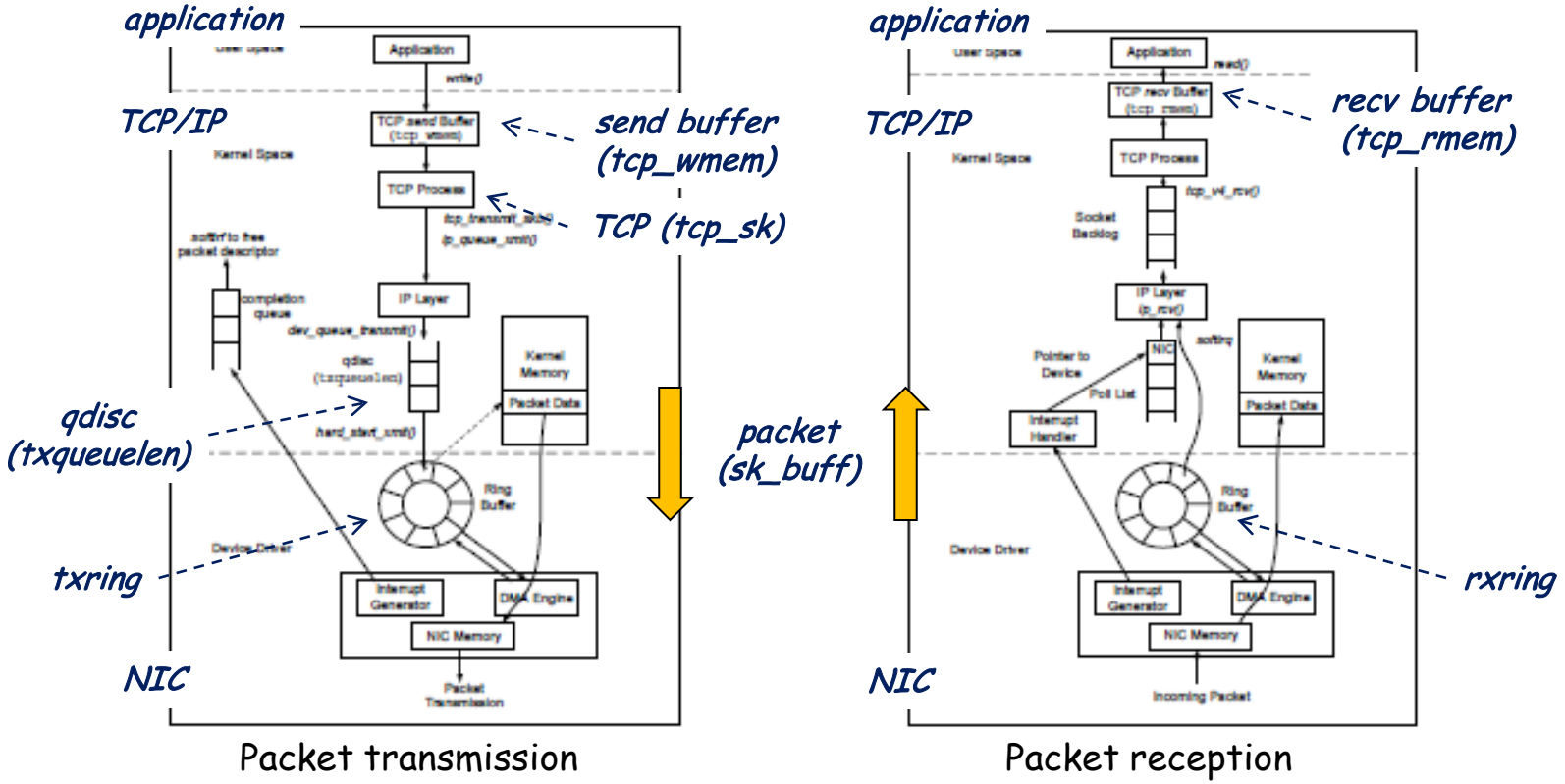
- ns-2 simulation using TCP implementation code in Linux kernel
  - bridge between implementations (Linux kernel) and simulations (ns-2)
    - fill a gap between implementation and simulation
  - verification of implementation codes

ns-2 TCP-Linux (2)

- TCPs implemented in Linux kernel (2.6.16-3)
  - TCP-Reno, TCP-Vegas, HighSpeed-TCP, Scalable-TCP, BIC-TCP, CUBIC-TCP (default), TCP-Westwood, H-TCP, TCP-Hybla
- TCPs to be implemented
  - TCP-Veno, TCP-LowPriority, Compound-TCP (Windows)

ns-2 TCP-Linux (3)

- TCP Implementation in Linux



http://www.ece.virginia.edu/cheetah/documents/papers/TCPlinux.pdf

ns-2 TCP-Linux (4)

- Variables in tcp_sk

Name	Meaning	Equivalent in NS-2 *TCPAgent*
snd_ ssthresh	slow-start threshold	ssthresh_
snd_ cwnd	integer part of the congestion window	trunc(cwnd_)
snd_ cwnd_cnt	fraction of congestion window	trunc(cwnd_^2) %trunc(cwnd_)
icsk_ca_ priv	a 512-bit array to hold per-flow state for a congestion control algorithm	n/a
icsk_ca_ ops	a pointer to the congestion control algorithm interface	n/a

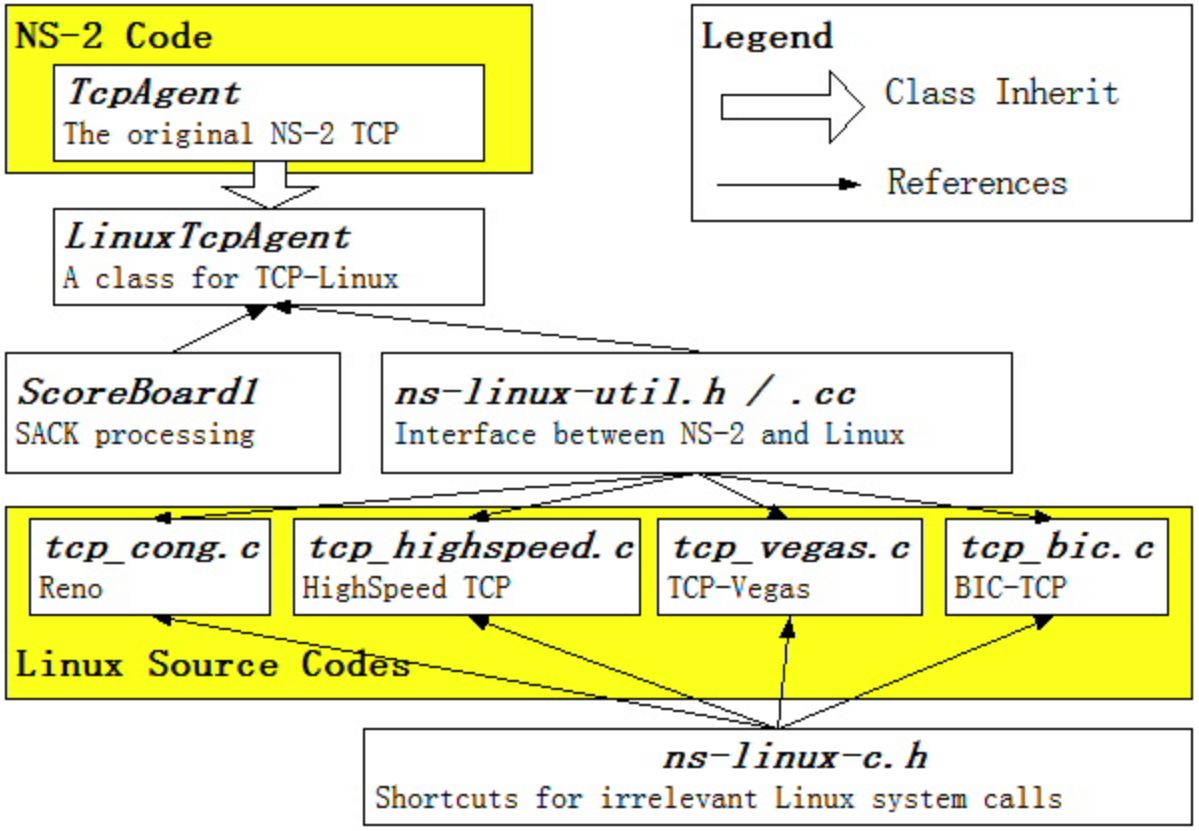
Congestion control modules:

cong_avoid: slow start & congestion avoidance
ssthresh: loss event handling
min_cwnd: fast retransmission

http://netlab.caltech.edu/projects/ns2tcplinux/

ns-2 TCP-Linux (5)

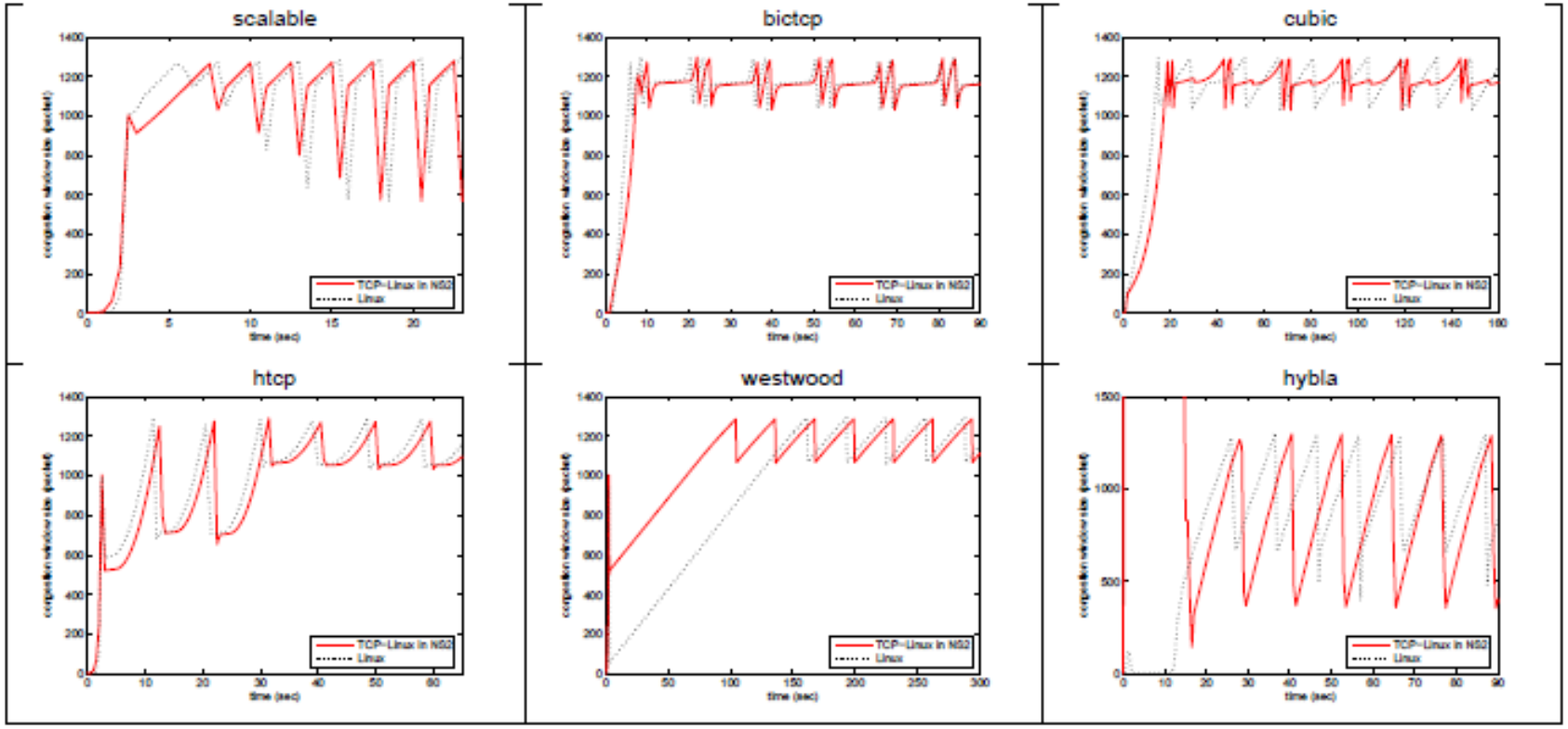
- Code structure



http://netlab.caltech.edu/projects/ns2tcplinux/

ns-2 TCP-Linux (6)

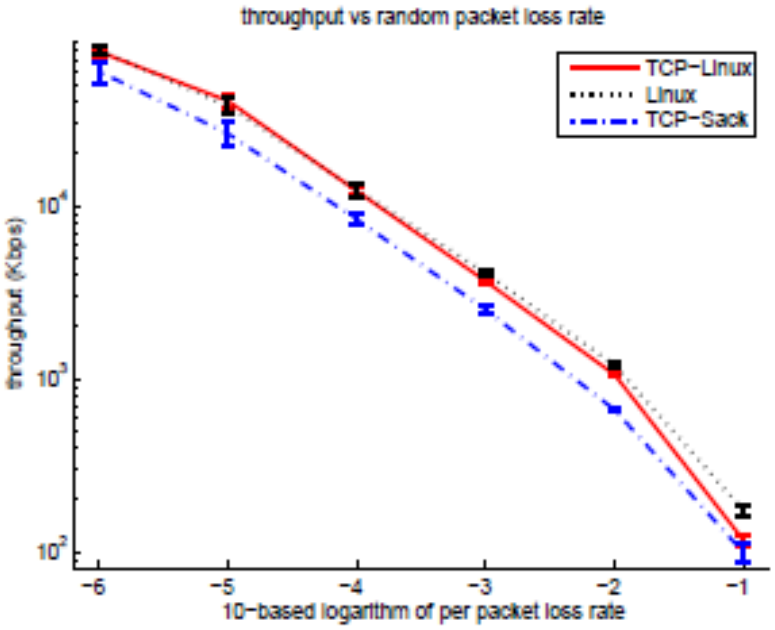
- Simulation (1) ns-2 & Linux



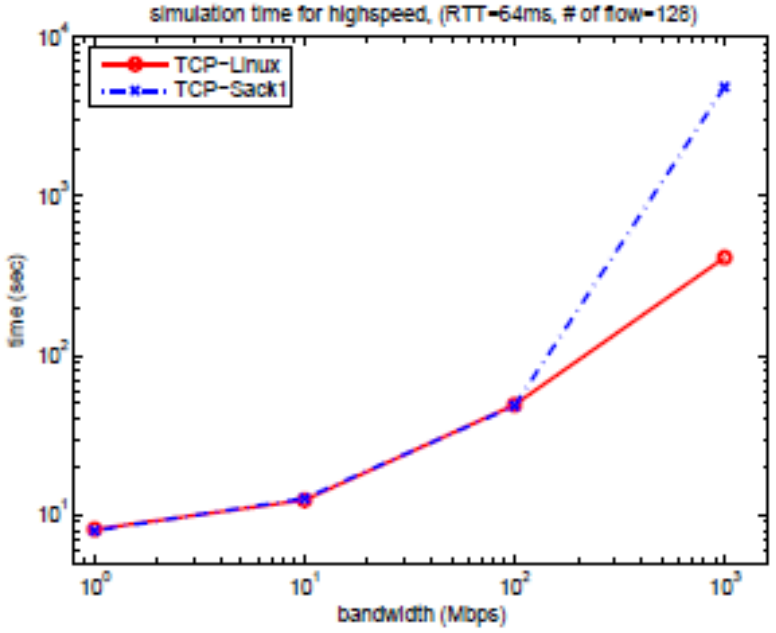
http://netlab.caltech.edu/projects/ns2tcplinux/

ns-2 TCP-Linux (7)

- Simulation (2) accuracy & speed



Accuracy



Speed

http://netlab.caltech.edu/projects/ns2tcplinux/

ns-3

ns-3 software overview

- ns-3 is written in C++, with bindings available for Python
  - simulation programs are C++ executables or Python programs
  - Python is often a glue language, in practice

- ns-3 is a GNU GPLv2-licensed project
- ns-3 lacks an integrated development / visualization environment (IDE)
- ns-3 is not backwards-compatible with ns-2

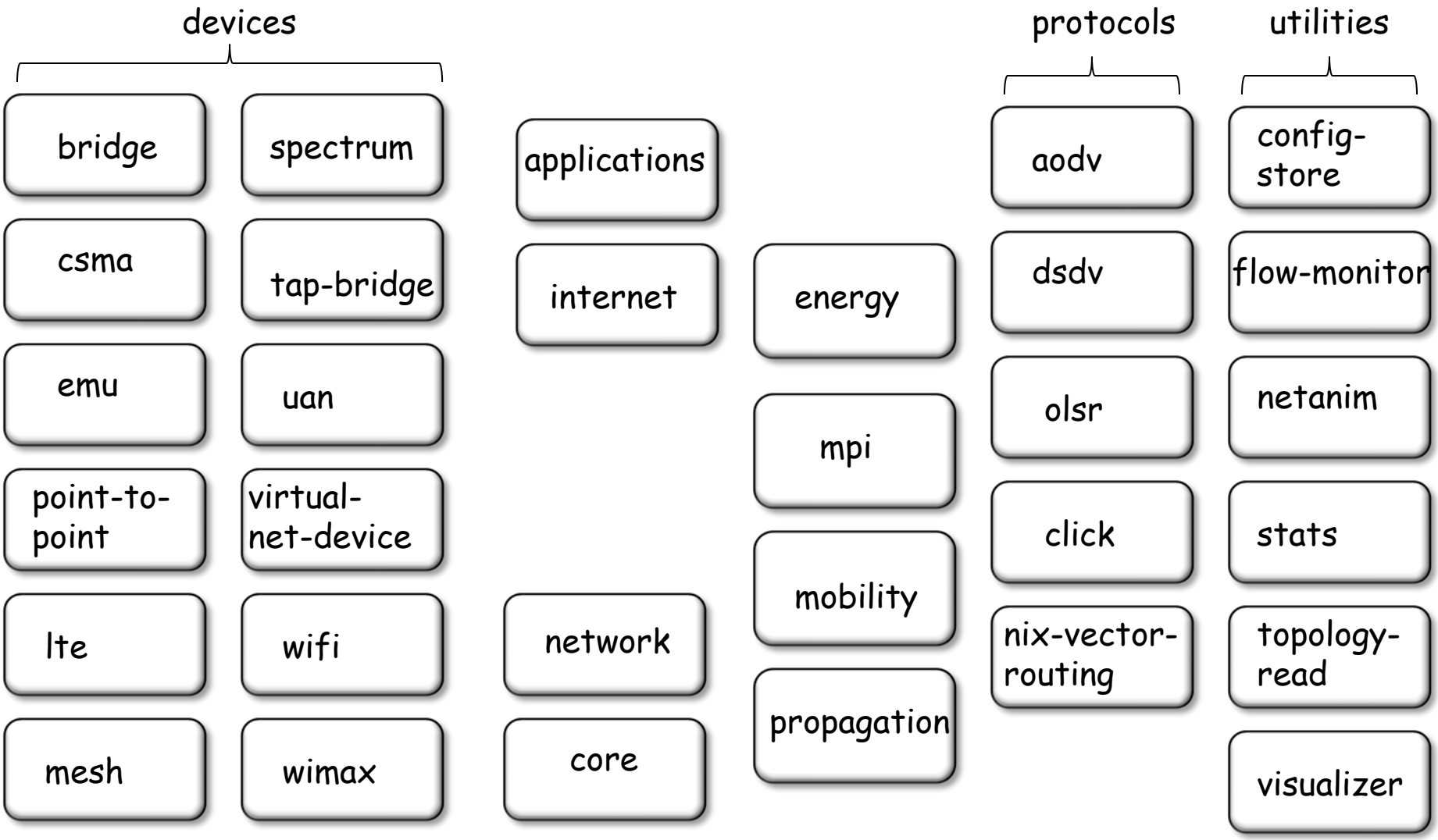
http://www.nsnam.org/docs/ns-3-overview.ppt

ns-3 development process

ns-3 is run as an open source project backed by research funding

- GPLv2 licensing
- open mailing lists
- uses standard tools (Mercurial, Bugzilla, Mediawiki, GNU/Linux development)
- ~20 maintainers worldwide

Available modules (ns-3.11 May 2011)

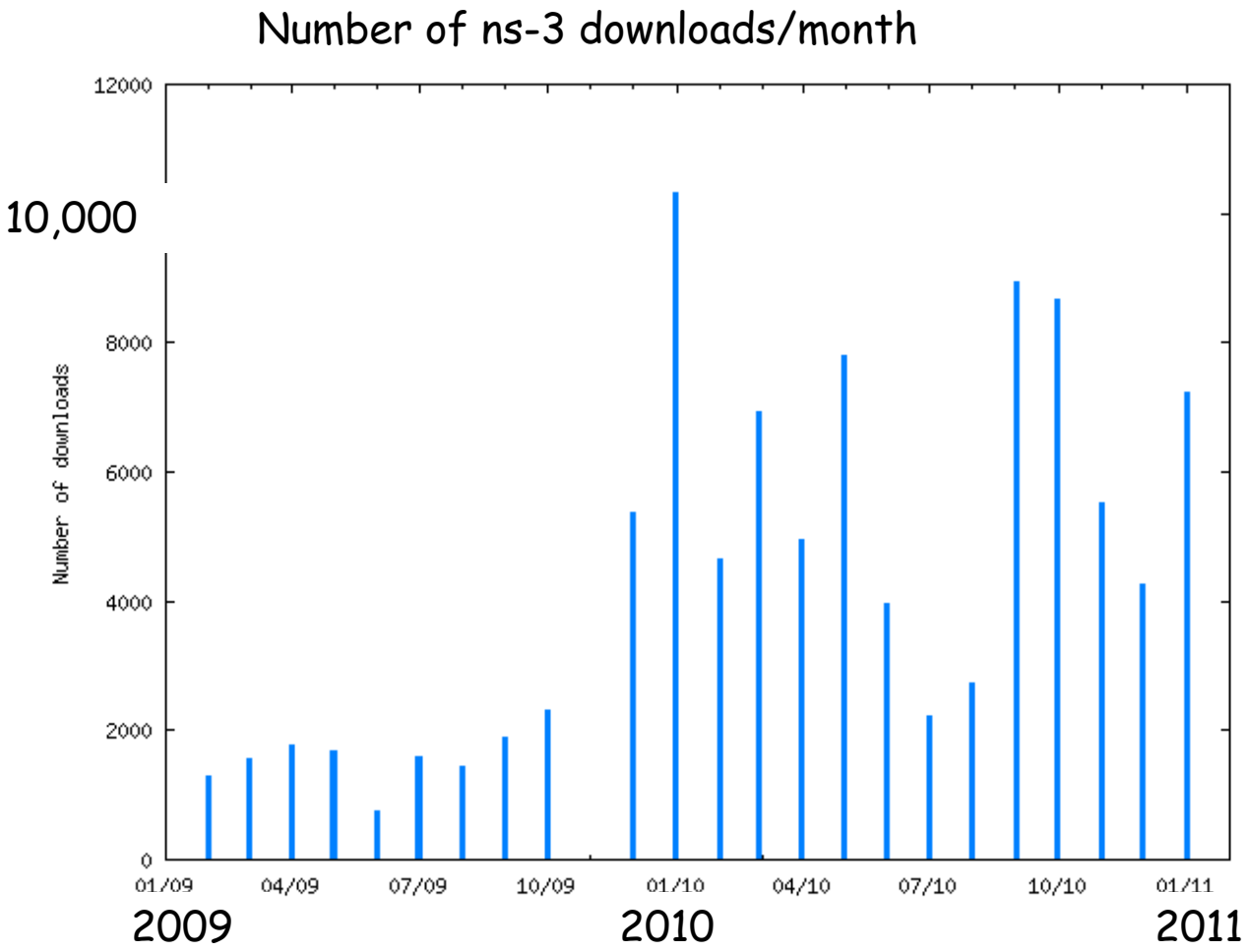


http://www.nsnam.org/docs/ns-3-overview.ppt

Analytics

Mailing list subscriptions:
- ns-3-users: 963
- ns-developers: 1176

Downloads:
- 6000/month in 2010



http://www.nsnam.org/docs/ns-3-overview.ppt

summary of ns-3 features

- modular, documented core
- C++ programs or (optionally) Python scripting
- alignment with real systems (sockets, device driver interfaces)

- emphasis on software integration
- virtualization and testbed integration are a priority (emulation modes)
- well-documented attribute system
- updated models

http://www.nsnam.org/docs/ns-3-overview.ppt

Resources

Web site:

http://www.nsnam.org

Mailing list:

http://mailman.isi.edu/mailman/listinfo/ns-developers

IRC: #ns-3 at freenode.net

Tutorial:

http://www.nsnam.org/docs/tutorial/tutorial.html

Code server:

http://code.nsnam.org

Wiki:

http://www.nsnam.org/wiki/index.php/Main_Page

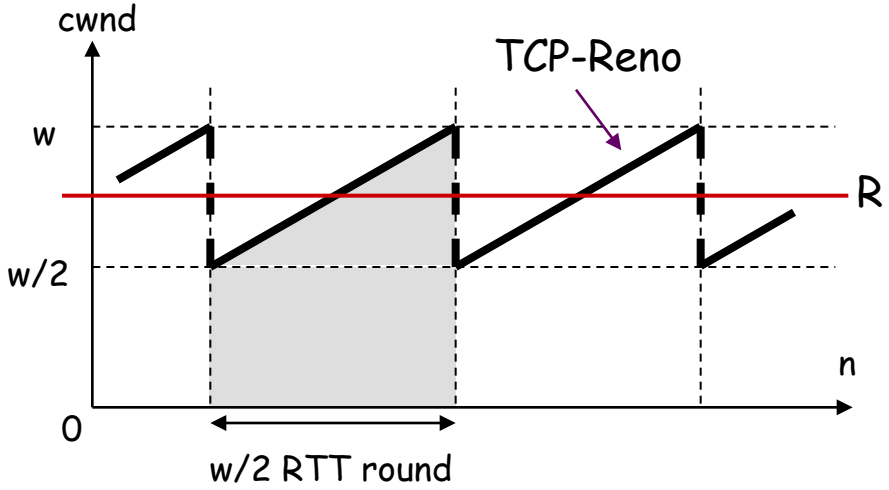
"Architecture, design and source code comparison of ns-2 and ns-3 network simulators", 2010 Spring Simulation Multi-conference

http://portal.acm.org/citation.cfm?id=1878651

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

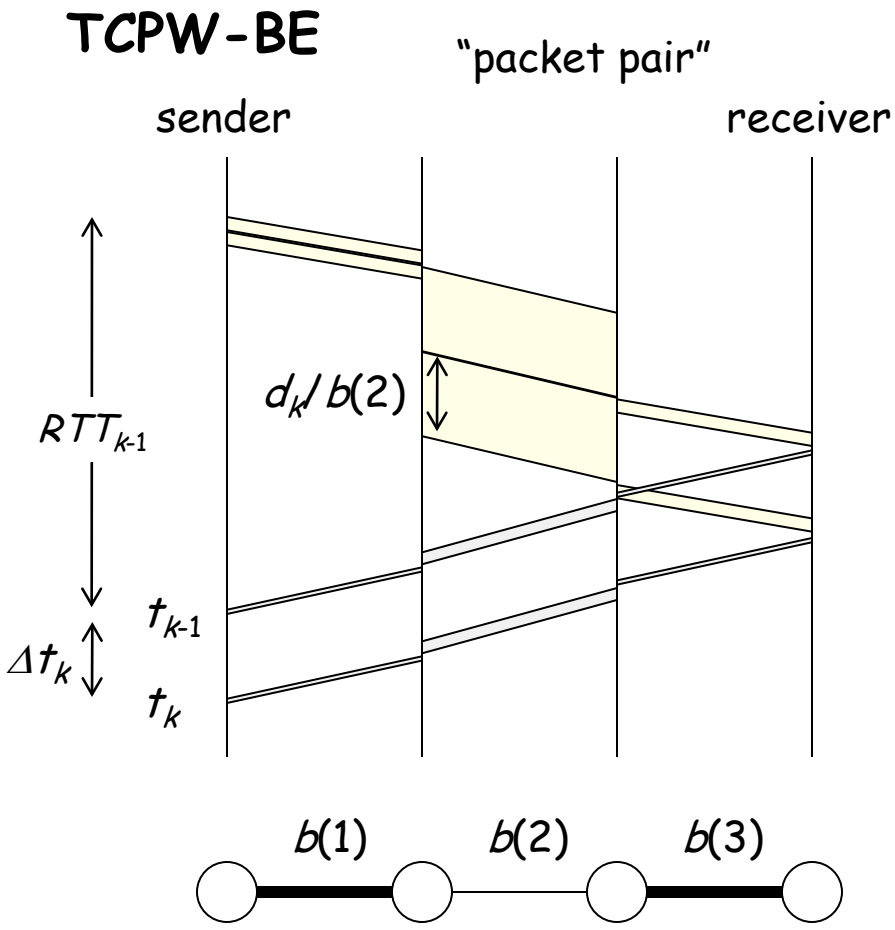
$$ssthresh = cwnd / 2$$

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

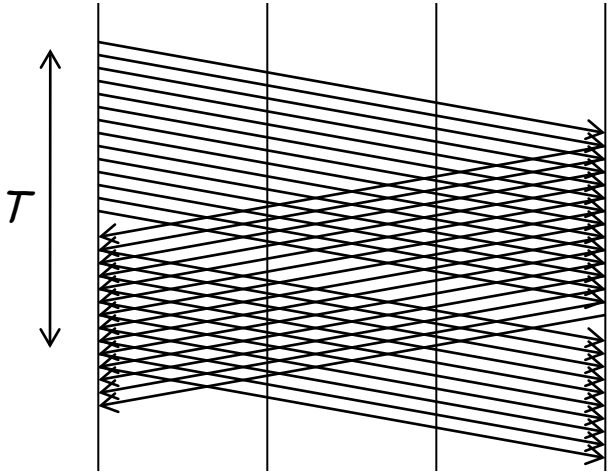
C.Casetti et al: "TCP Westwood: Bandwidth Estimation for Enhanced Transport over Wireless Links", ACM MOBICOM 2001.

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= $\Sigma \Delta t_k$

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

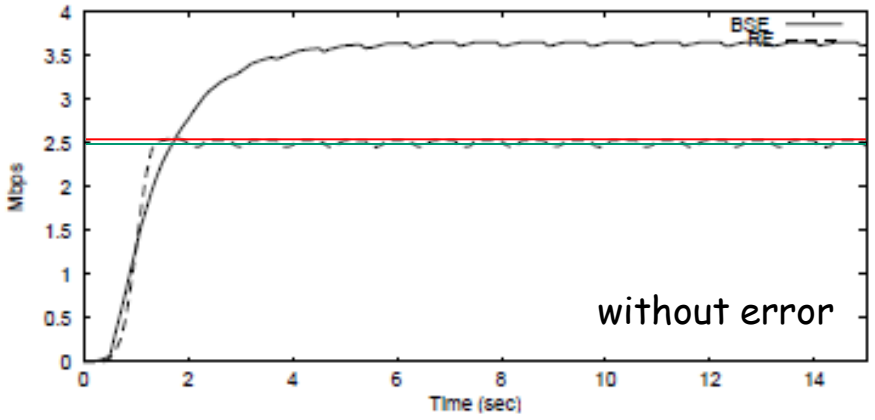
moving average: $\hat{RE}_k \rightarrow FSE$

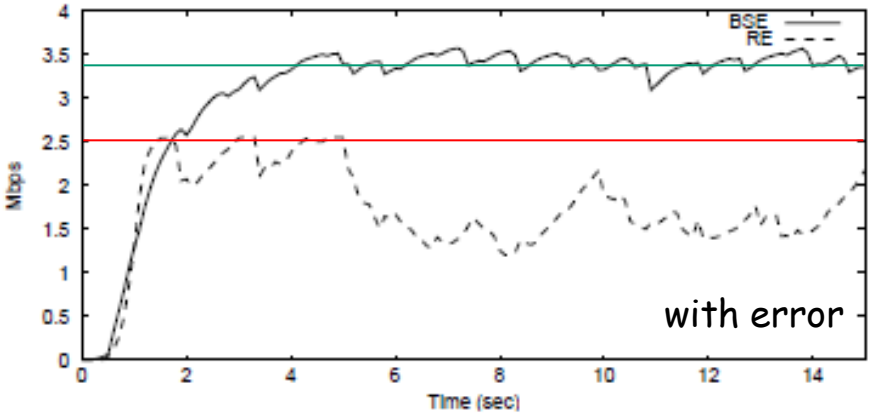
$T = n \cdot RTT$ (e.g. n=4)

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



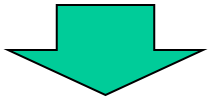


- BSE tends to overestimate (due to burstiness)
- RE tends to underestimate when losses occur

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.

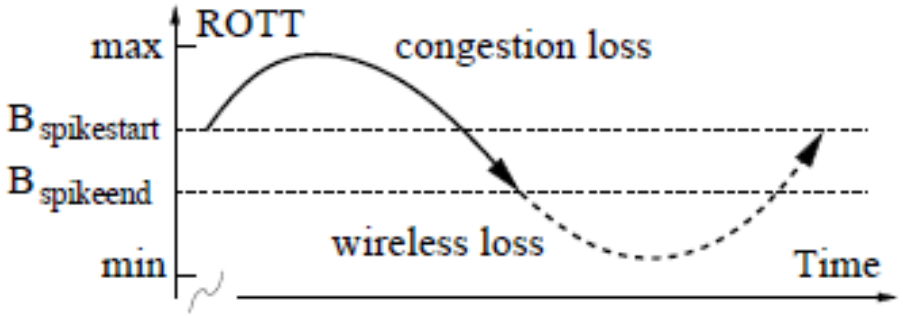
Disadvantage of TFRC

- 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

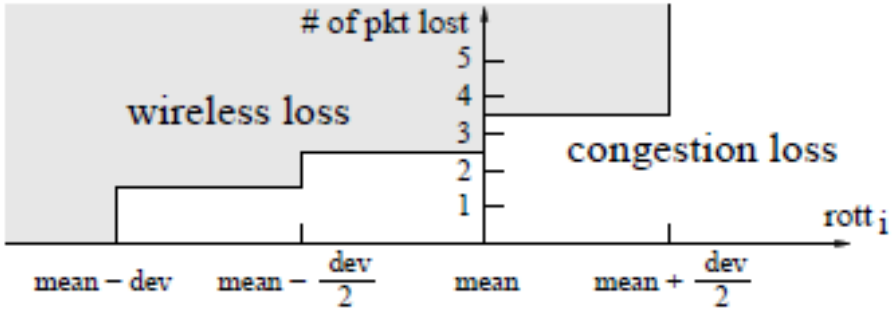
LDA (1)

"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.

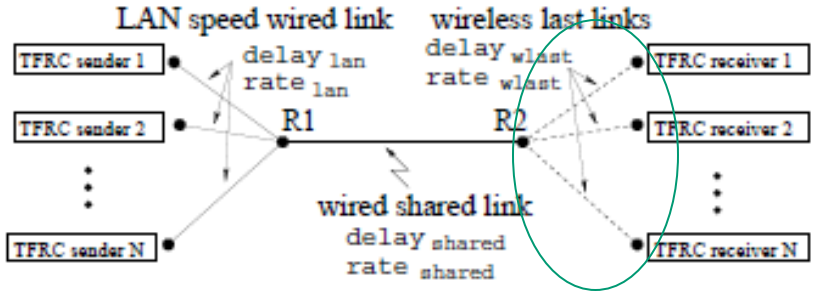
LDA (2)

"TFRC Wireless"

- Simulation results

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

Wireless last hop

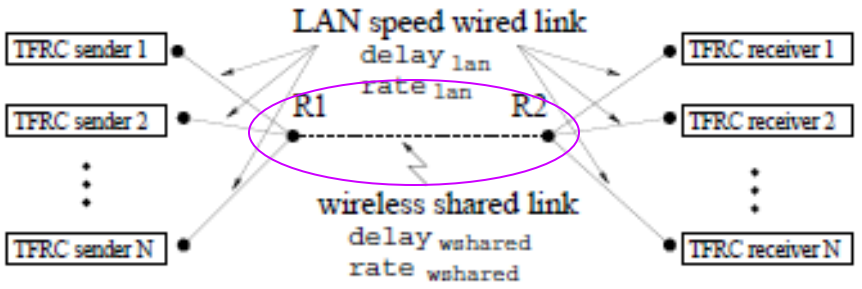


PERFORMANCE FOR WIRELESS LAST HOP, 1 FLOW

	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.

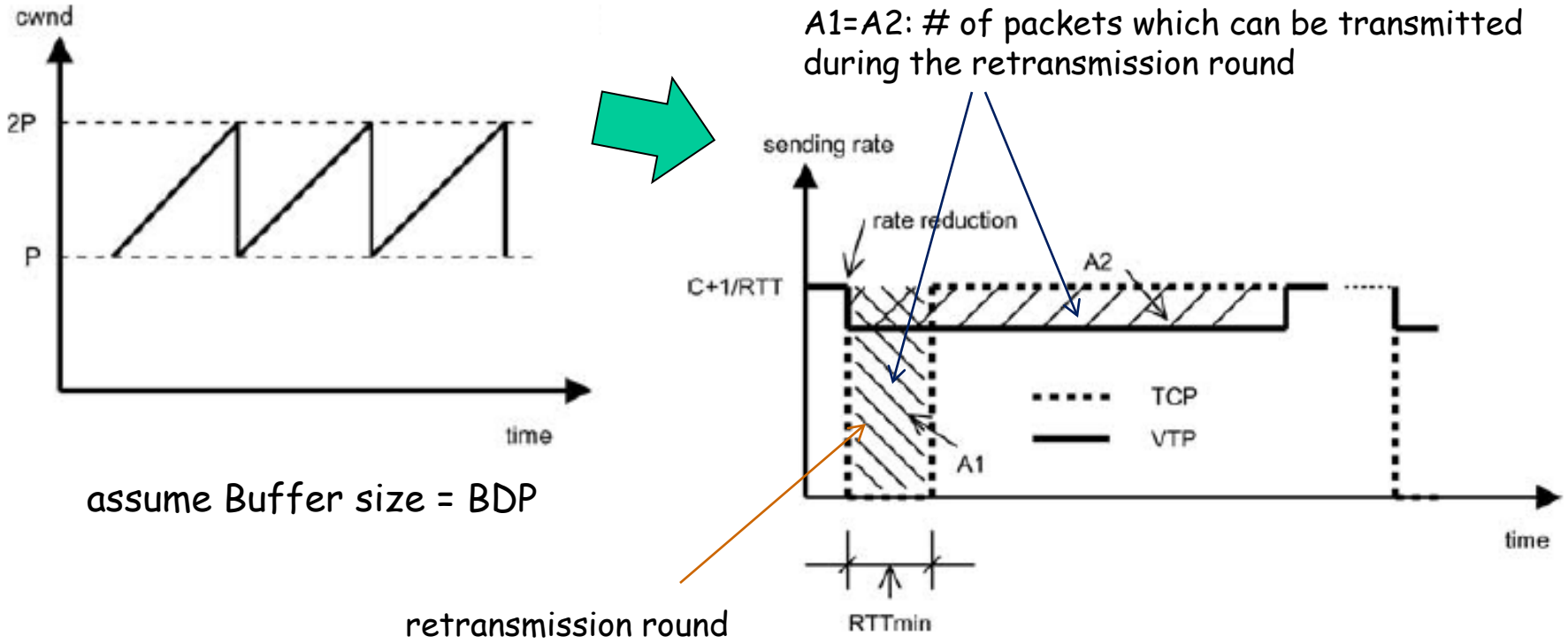
VTP (1)

- 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

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

$$ewnd_k = R_k \times RTT_k$$

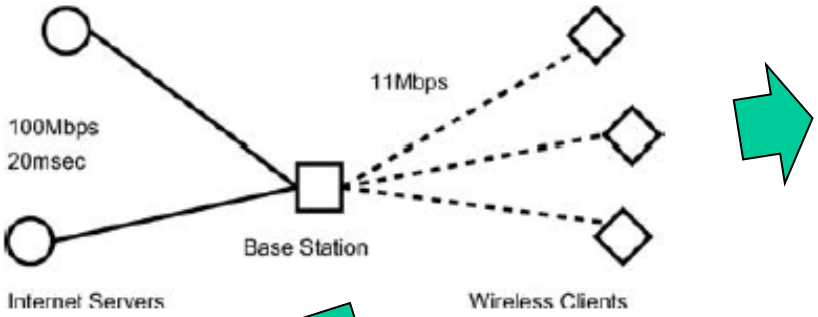
if no RTT increase, $R_{k+1} = R_k + \frac{1}{RTT_k}$

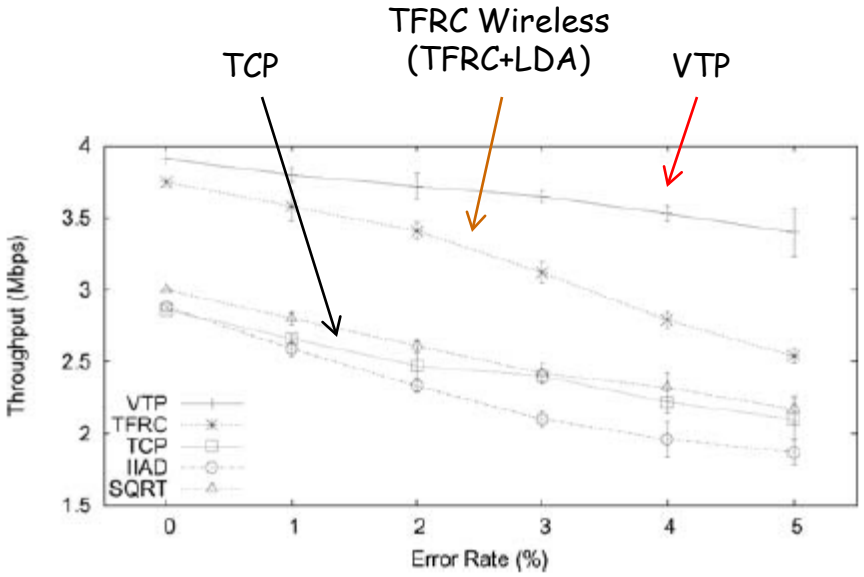
$$R_{k+1} = \frac{ewnd_{k+1}}{RTT_{k+1}} = \frac{ewnd_{k+1}}{RTT_k + \Delta RTT_k} = \frac{R_k \times RTT_k + 1}{RTT_k + (RTT_k - RTT_{k-1})}$$

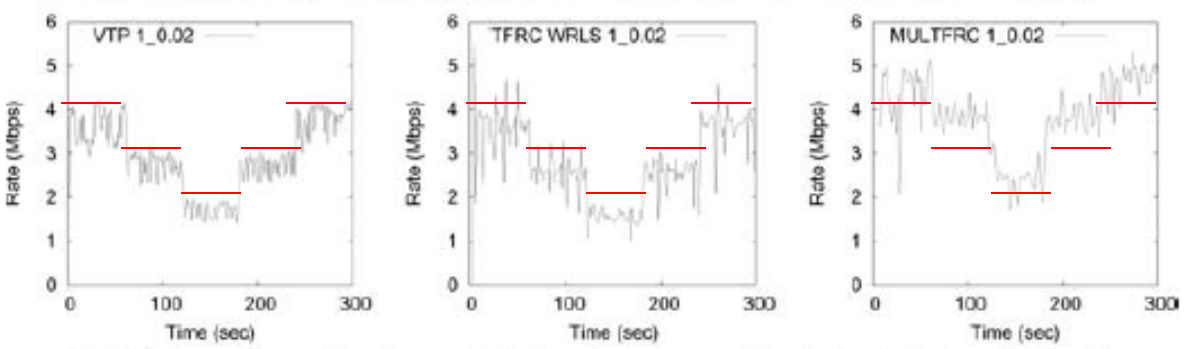
G.Yang et al: "Smooth and efficient real-time video transport in the presence of wireless errors", ACM Trans. MCCAP, 2006.

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

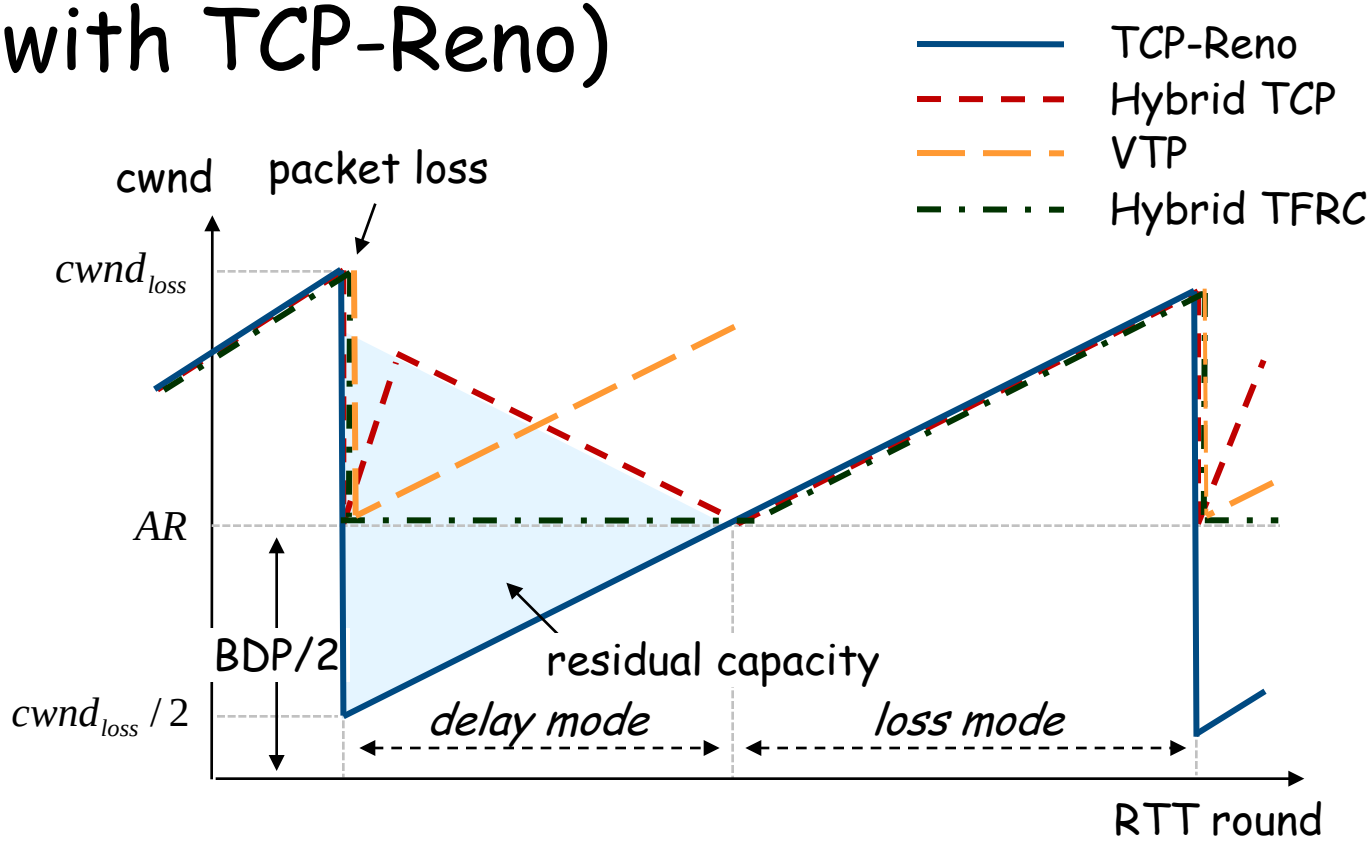
Hybrid TFRC (1)

- 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)

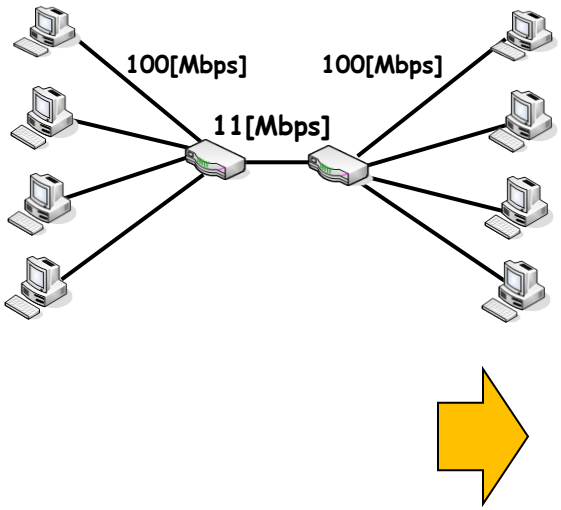
- Hybrid TCP behavior (when competing with TCP-Reno)

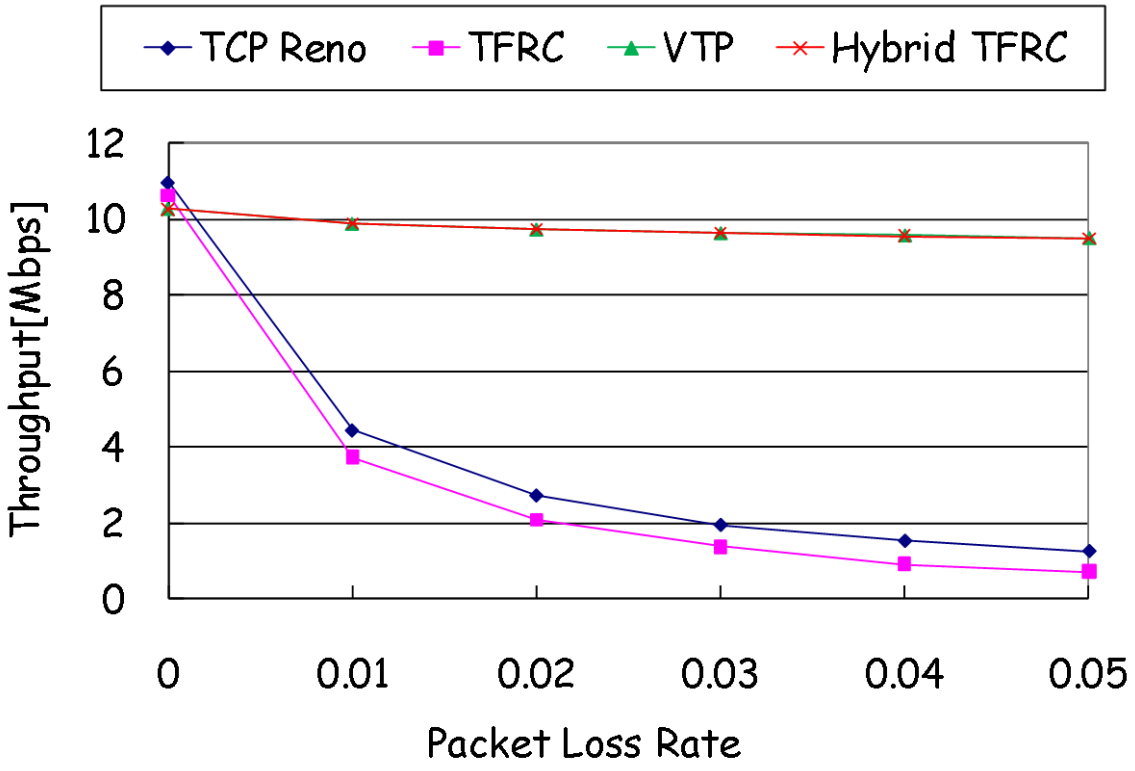


Hybrid TFRC (3)

- simulation result (1)

Buffer size = BDP

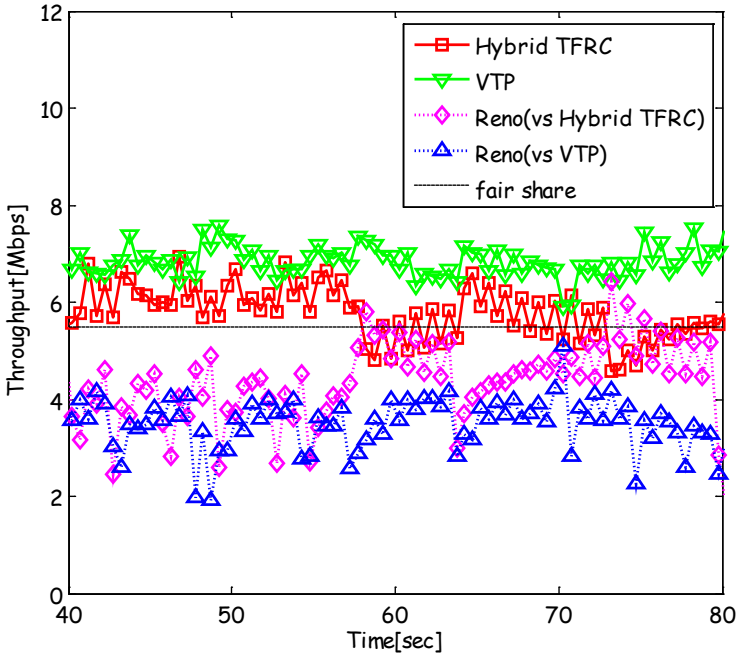




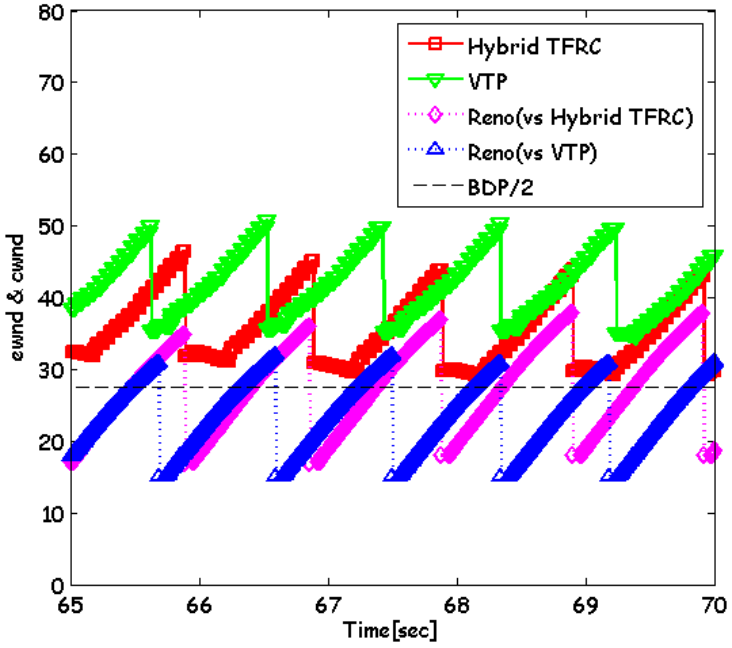
Hybrid TFRC (4)

- 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

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

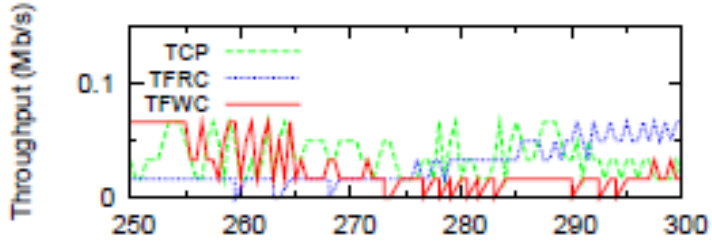
  - stable because RTT is eliminated

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

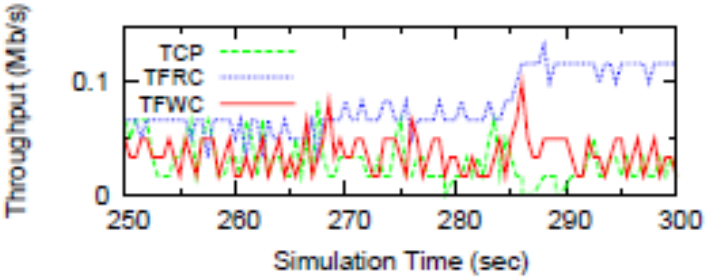
TCP-Friendly Window-based Congestion Control (3)

- Hybrid Window & Rate
  - too small cwnd causes timeout

  - operates in TFRC when cwnd is too small



(a) operated by window-based only



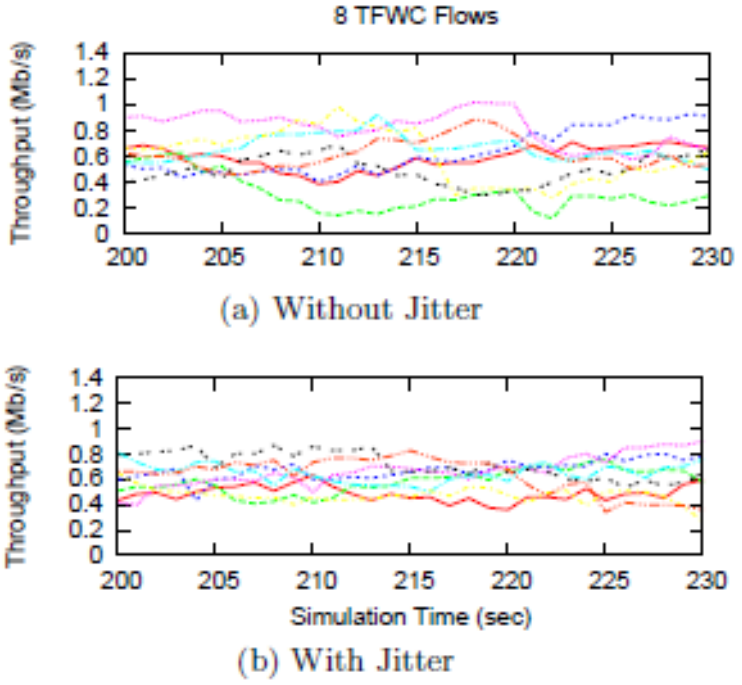
(b) operated by window/rate-based

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

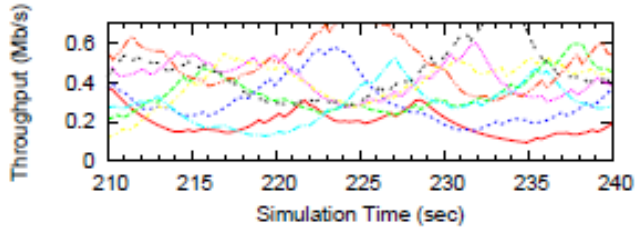
- Analogy: add small "random number" to cwnd



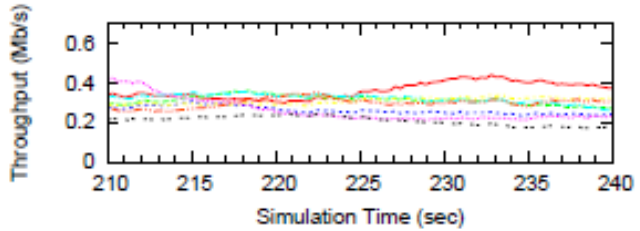
S.H.Choi et al: "Designing TCP-Friendly Window-based Congestion Control …", PFLDNet 2009.

TCP-Friendly Window-based Congestion Control (5)

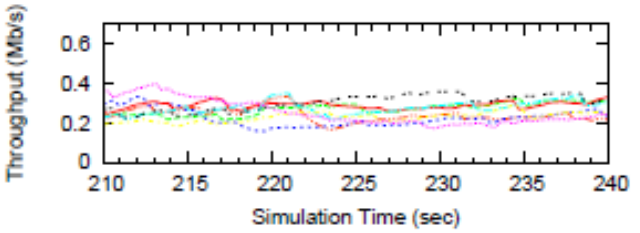
- Smoothness



(a) TCP's Abrupt Sending Rate Change

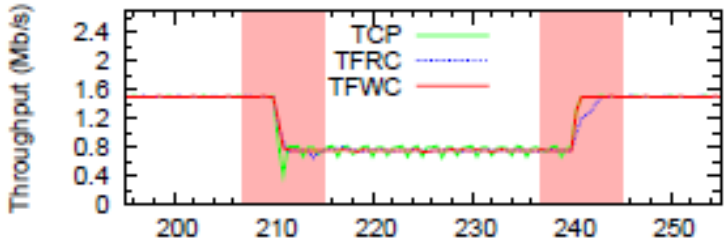


(b) TFRC Smoothness

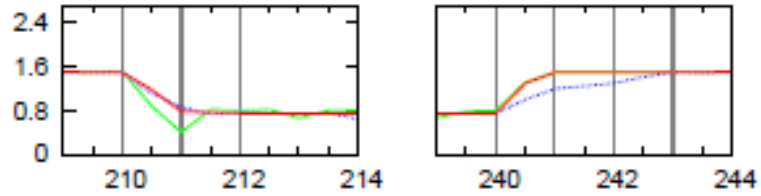


(c) TFWC Smoothness

- 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.

TCP over Wireless Networks

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 (L3 / L7)
  - mobility model: Random Waypoint, Random Trip
  - handover
- energy comsumption (all layers)
  - energy model

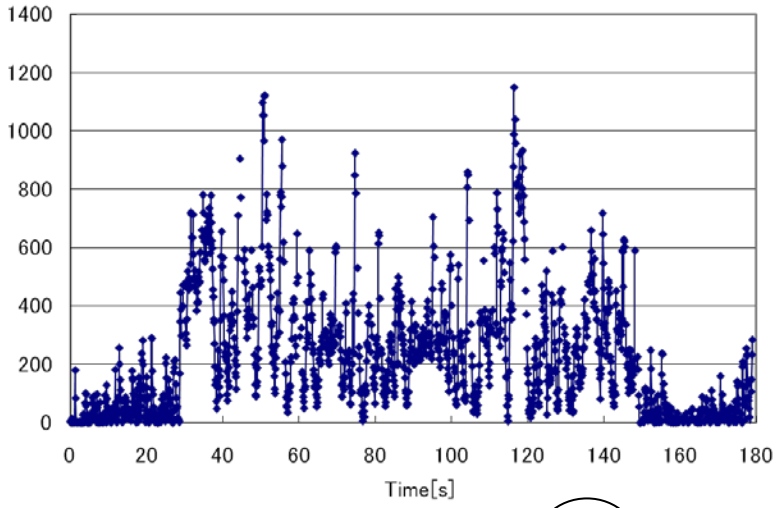
Discussion

- 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

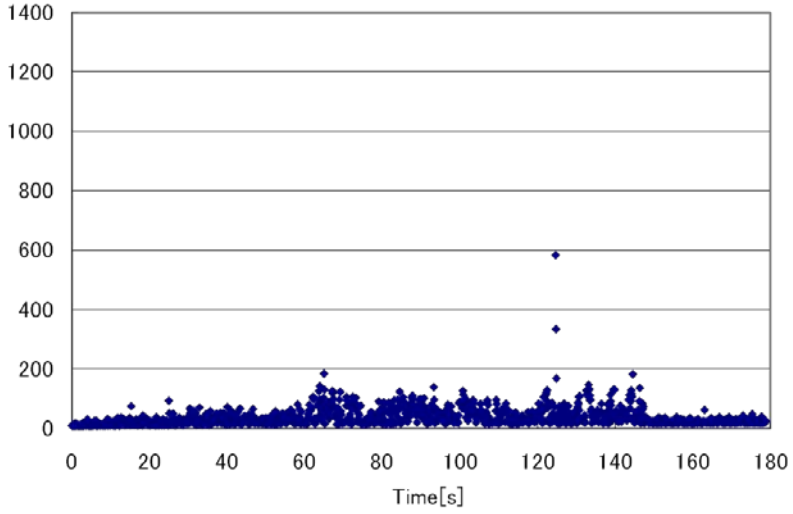
WiFi Example

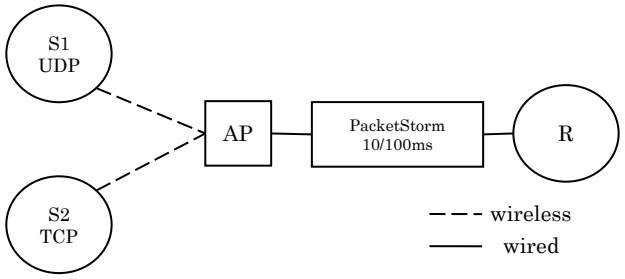
- RTT instability and unfairness between upload and download

RTT upload, wireless to wired



RTT download, wired to wireless



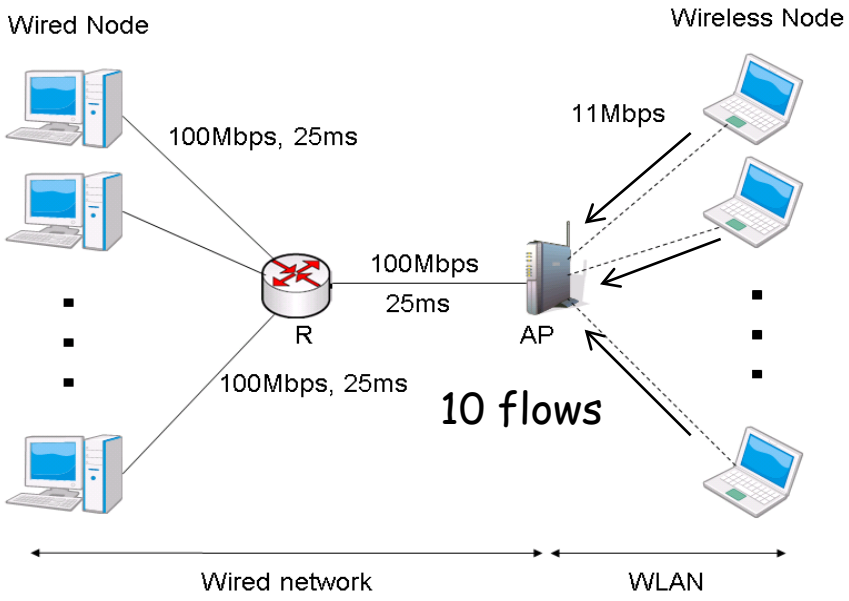


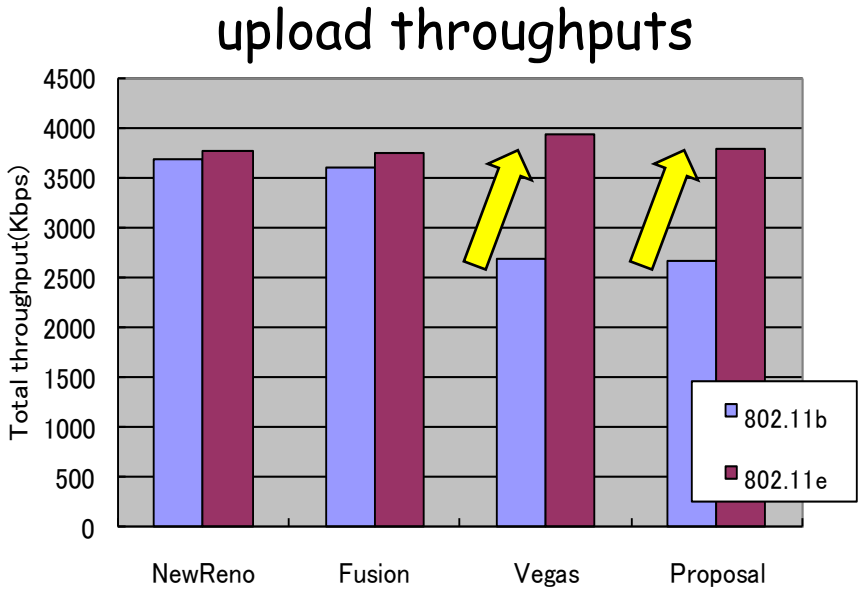
K.Kanai et al: "Performance evaluations of adaptive rate control mechanisms ...", IEICE Tech. Report, 2011.

Wireless LAN (1)

※ ns-2 simulation

- TCPs and throughputs





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

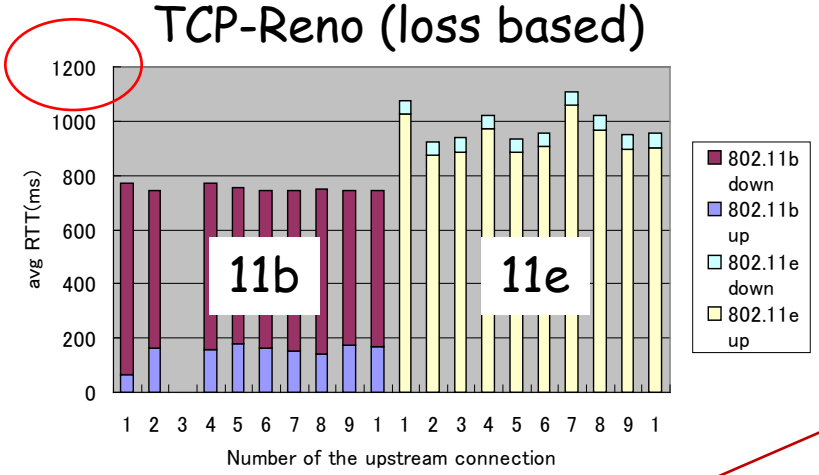
TCP-Reno: loss based
TCP-Fusion: hybrid
TCP-Vegas: delay based
Proposal: Vegas extension

K.Sonoda et al: "Performance Improvement of TCP-Vegas ...", IEICE Tech. Report, 2010.

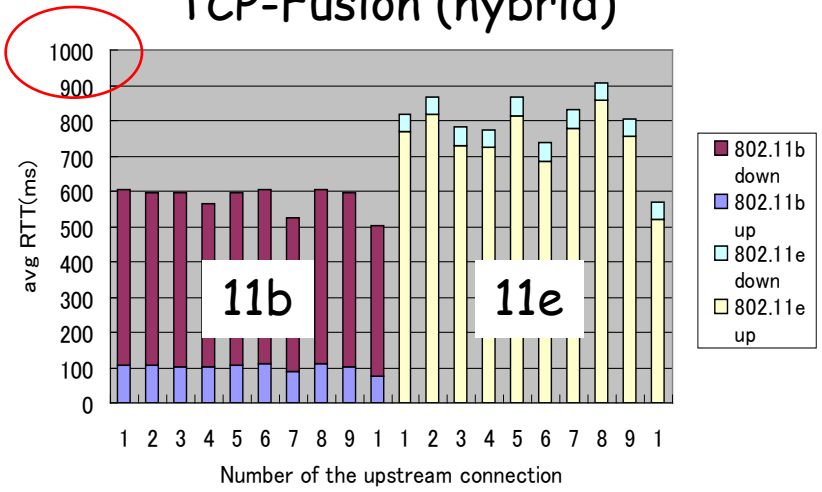
Wireless LAN (2)

- TCPs and delays

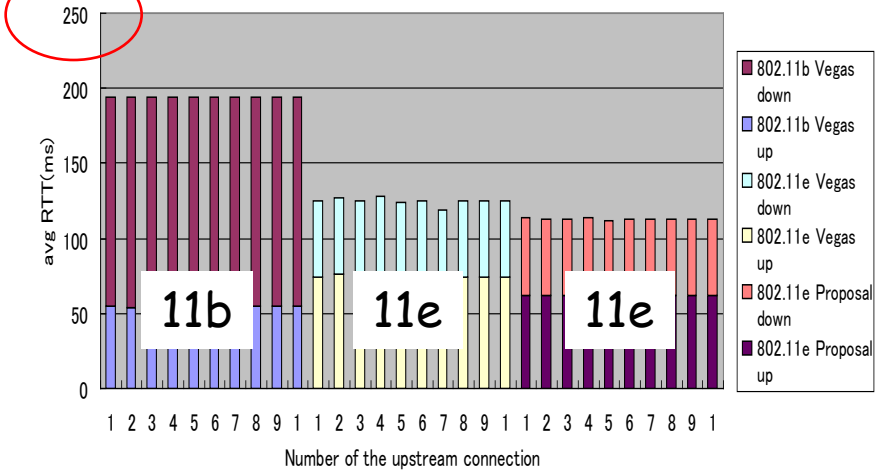
TCP-Reno (loss based)



TCP-Fusion (hybrid)



TCP-Vegas & Proposal (delay based)



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

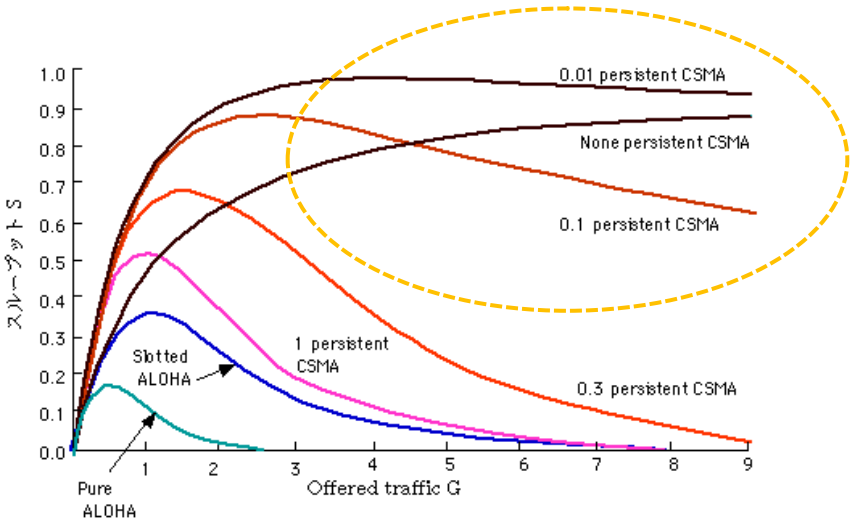
→ Hybrid TCP works in loss mode only

K.Sonoda et al: "Performance Improvement of TCP-Vegas ...", IEICE Tech. Report, 2010.

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



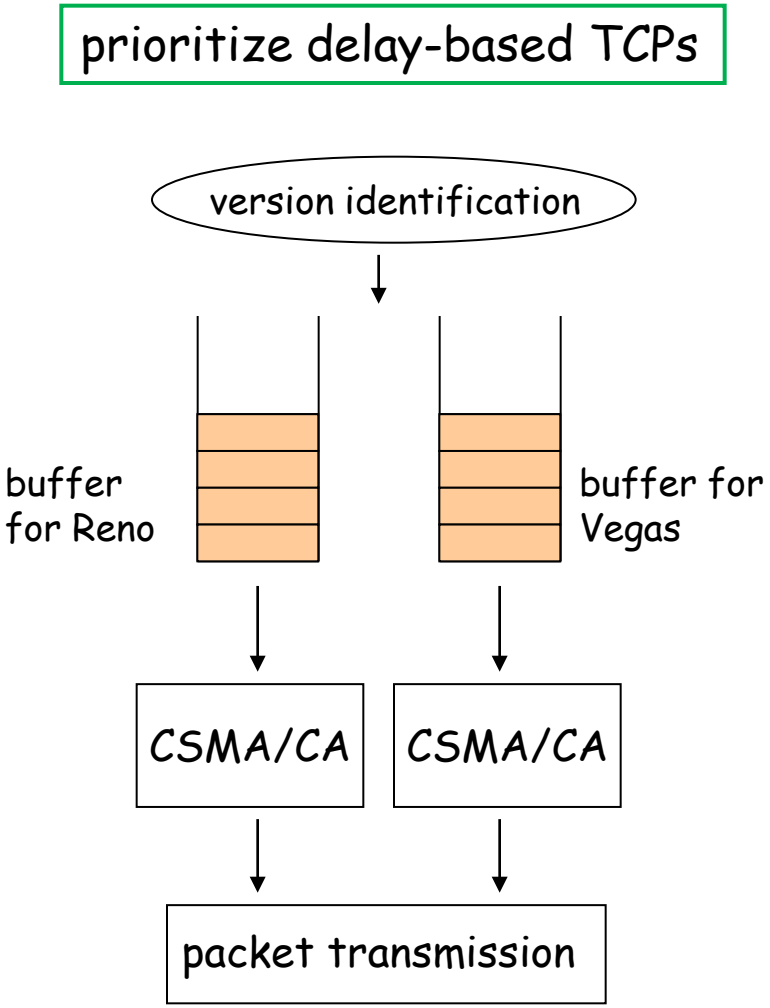
TCP Version Differentiation (1)

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

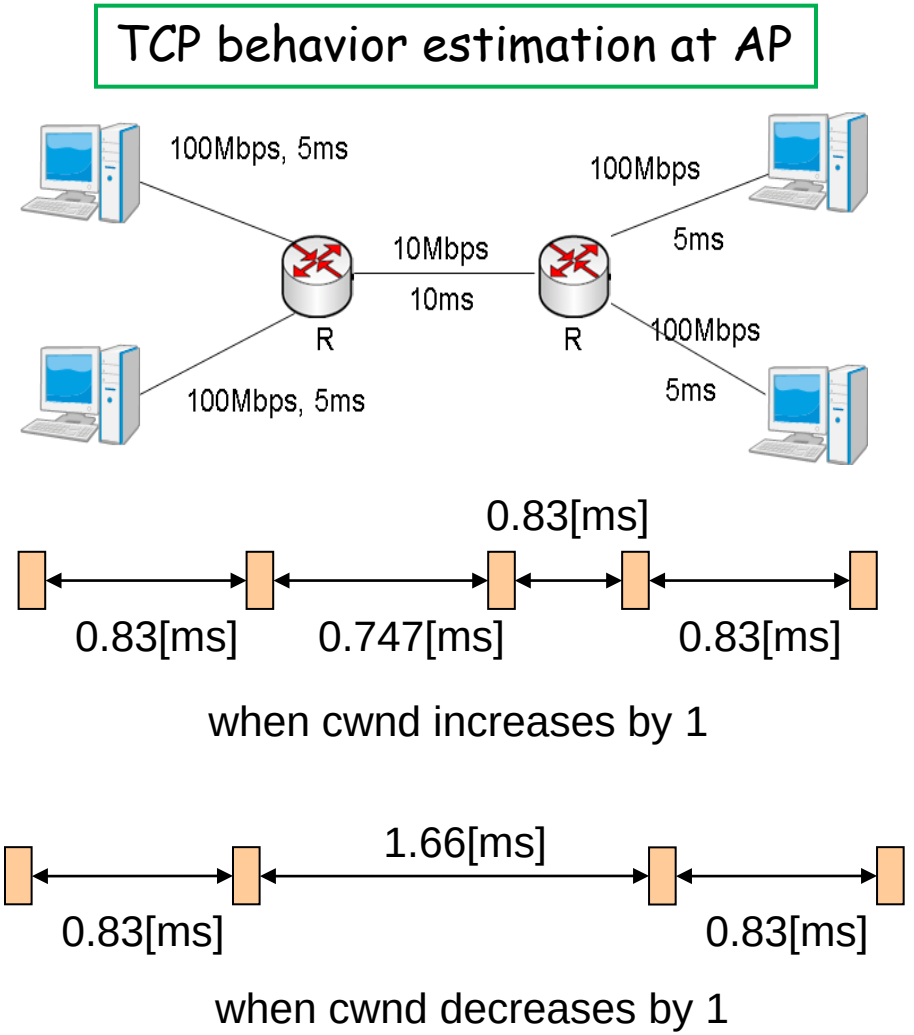
3. Prioritize delay based TCP flows by tuning CSMA/CA parameters of IEEE 802.11e



K.Sonoda et al: "A Method of TCP Version Identification ...", IEICE Tech. Report, 2011.

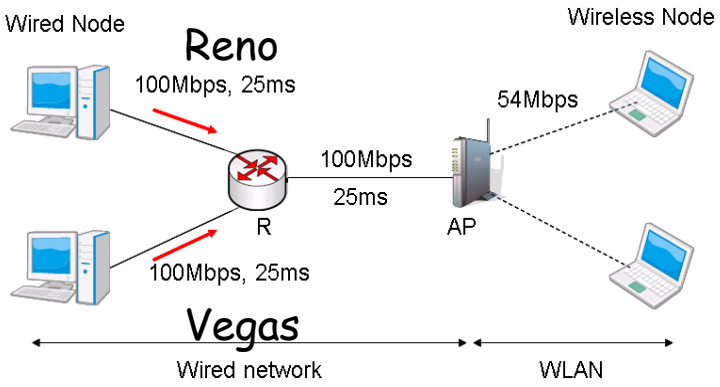
TCP Version Differentiation (2)

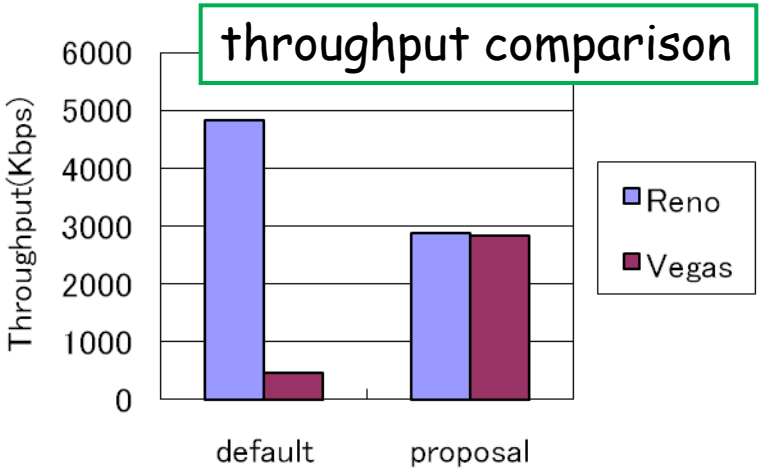
- RTT estimation for delay based flow
  - When cwnd increases by one, two consecutive packets are transmitted
  - When cwnd decreases by one, no packets are transmitted for the last ACK
- cwnd estimation
  - Access points let the number of arrived packets per RTT be "cwnd"



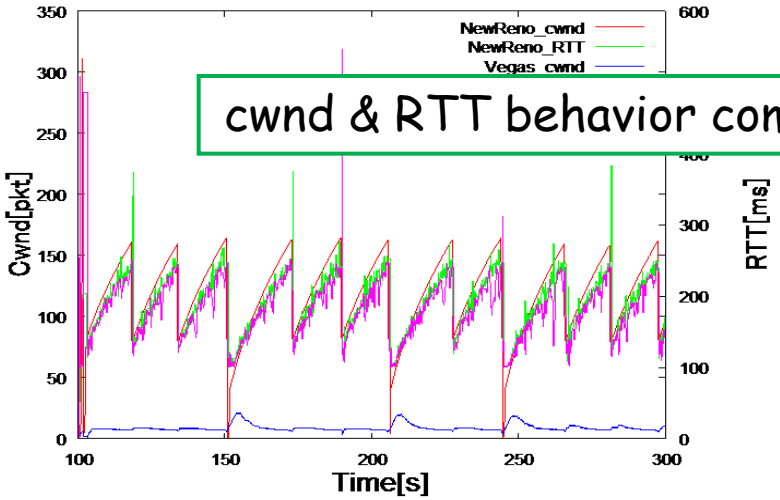
K.Sonoda et al: "A Method of TCP Version Identification ...", IEICE Tech. Report, 2011.

TCP Version Differentiation (3)

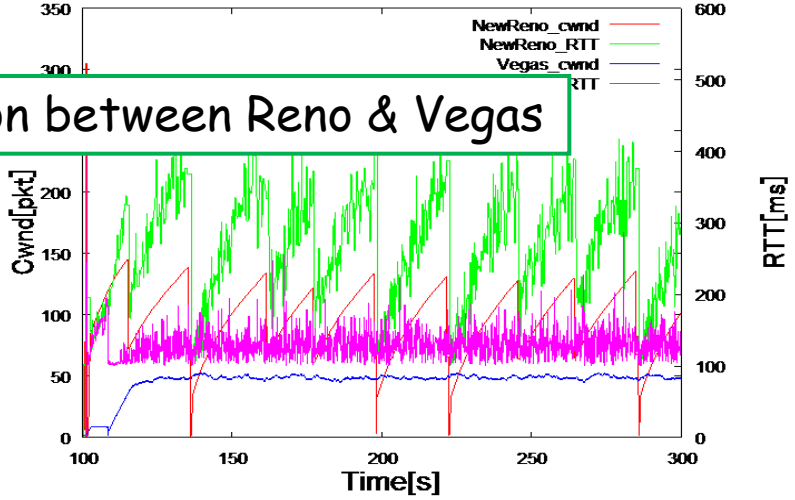




cwnd & RTT behavior comparison between Reno & Vegas



without differentiation

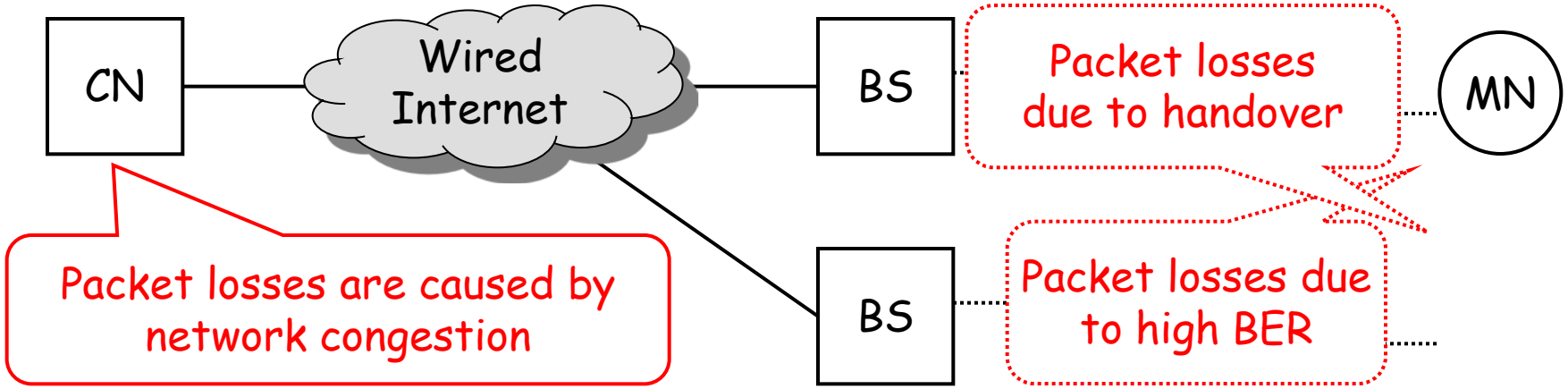


with differentiation

K.Sonoda et al: "A Method of TCP Version Identification ...", IEICE Tech. Report, 2011.

TCP over Wireless "Mobile" Networks

TCP over Mobile Networks



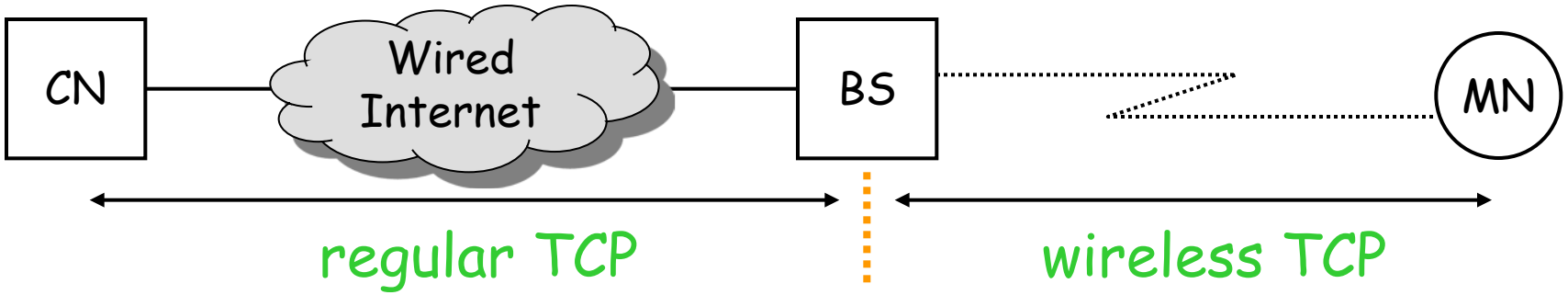
Congestion Control

Throughput Degradation

Related Work

- TCP Extensions for Mobile Networks
  - Split Connection: e.g. Indirect TCP
  - Proxy: e.g. Snoop TCP
  - End-to-End: e.g. Triple ACKs and Freeze TCP

(1) Split Connection



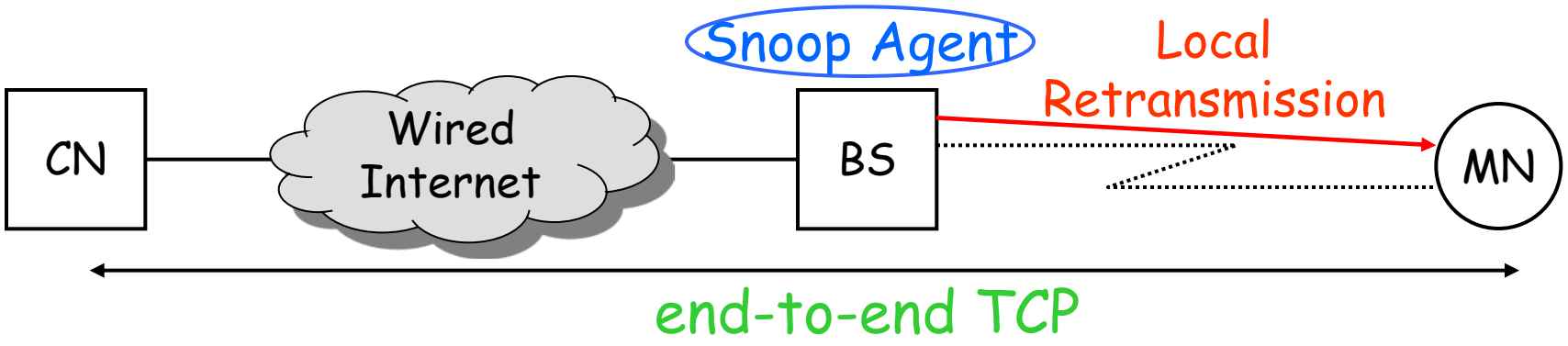
Transmission errors over wireless links are not propagated to wired networks

Forces heavy load on a base station

Breaks end-to-end semantics

Cannot handle encrypted IP payload

(2) Proxy



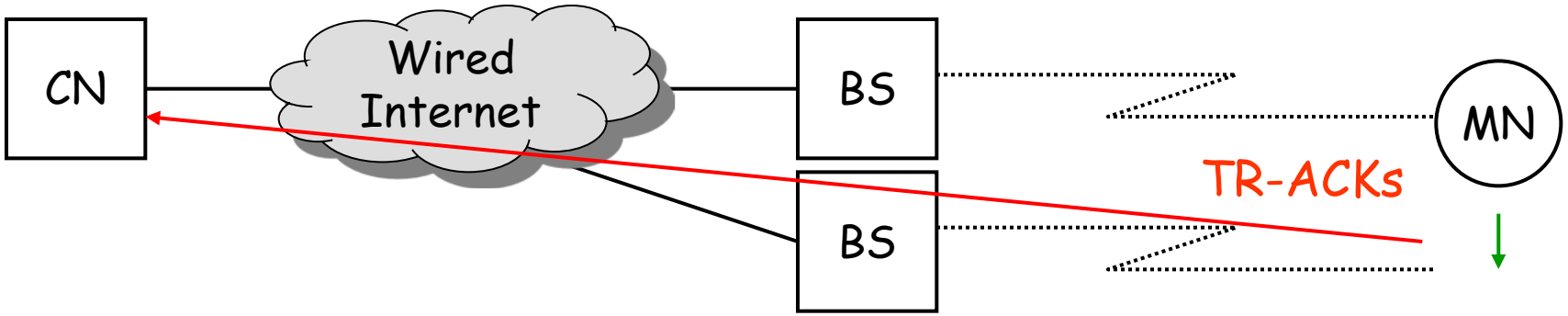
Local retransmission over wireless links avoids initiation of unnecessary congestion avoidance

Forces heavy load on a base station

Transfer of per-flow state information to a new base station causes huge handover delay

Cannot handle encrypted IP payload

(3-1) Triple ACKs



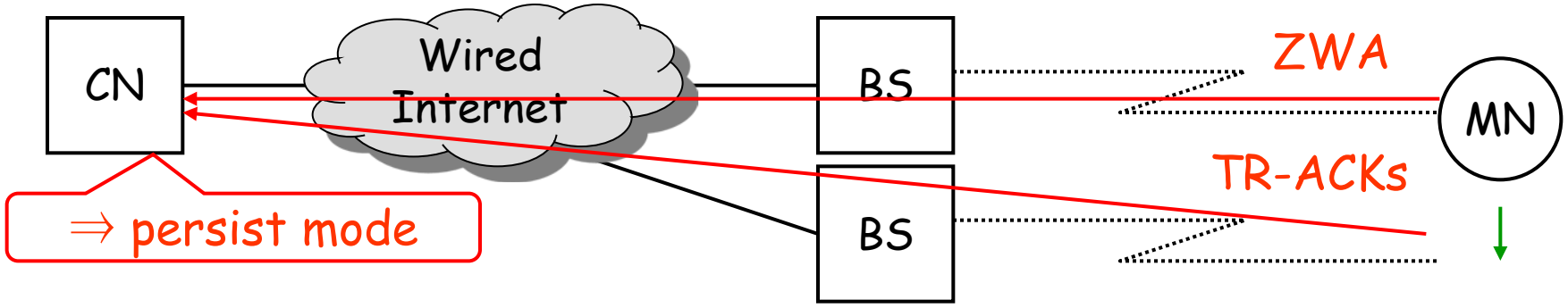
Does not need any base station support

Only TCP of mobile nodes should be modified (trivial)

Normal TCP retransmission process (e.g. fast recovery) might be too conservative

Sometimes suffers from TCP's retransmit timeout (RTO)

(3-2) Freeze TCP



Does not need any base station support

Only TCP of mobile nodes should be modified

A mobile node has to predict a link break precisely before the actual break happens

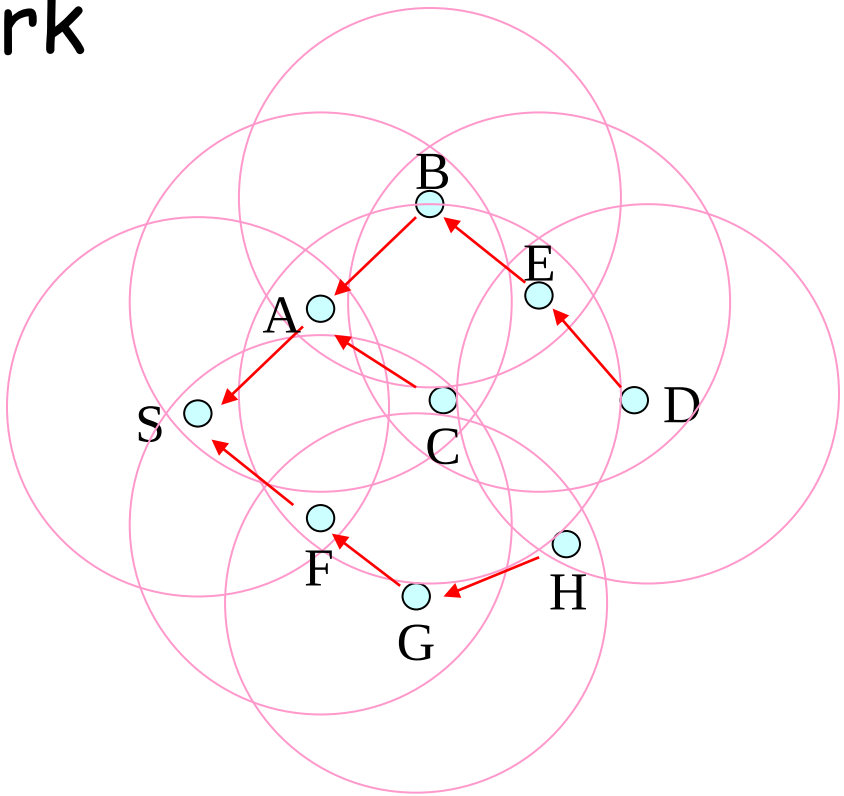
TCP performance strongly depends on transmission timing of ZWA (ideal "warning period" is equal to RTT)

ZWA: Zero Window Advertisement

TCP over Wireless "Multihop" Networks

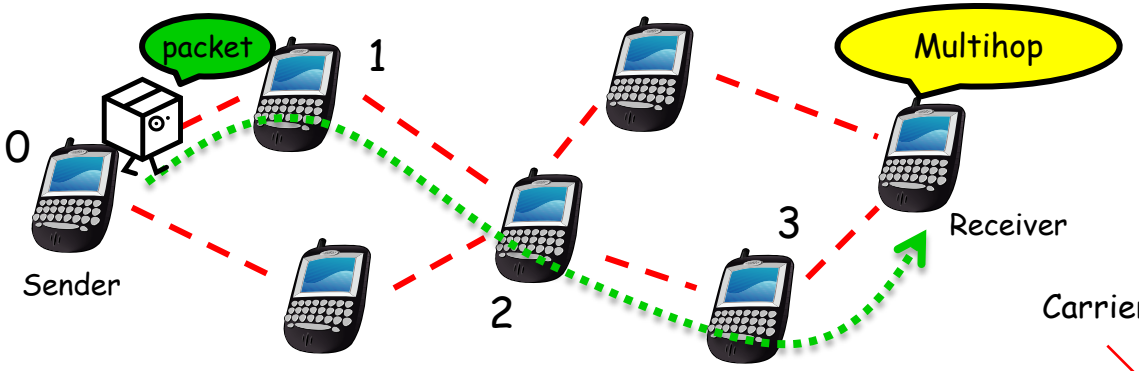
Wireless Multihop Networks

- ad-hoc network
- sensor network

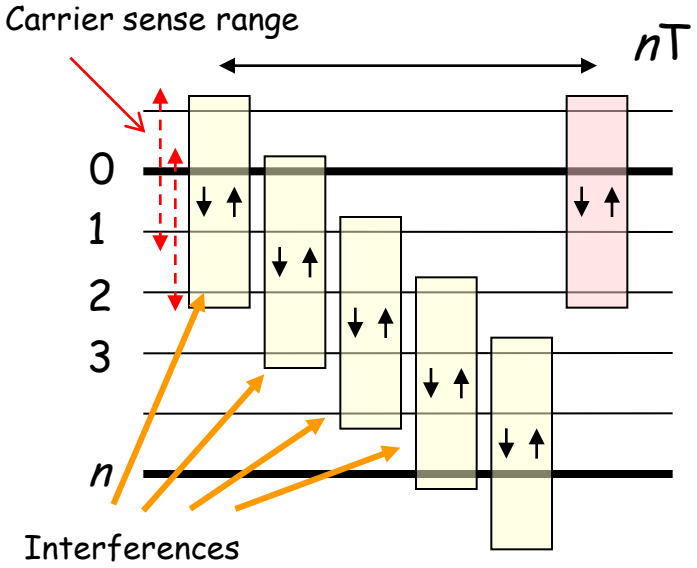


Wireless Multihop Networks (1)

- Single Radio Multi-hop Transmission



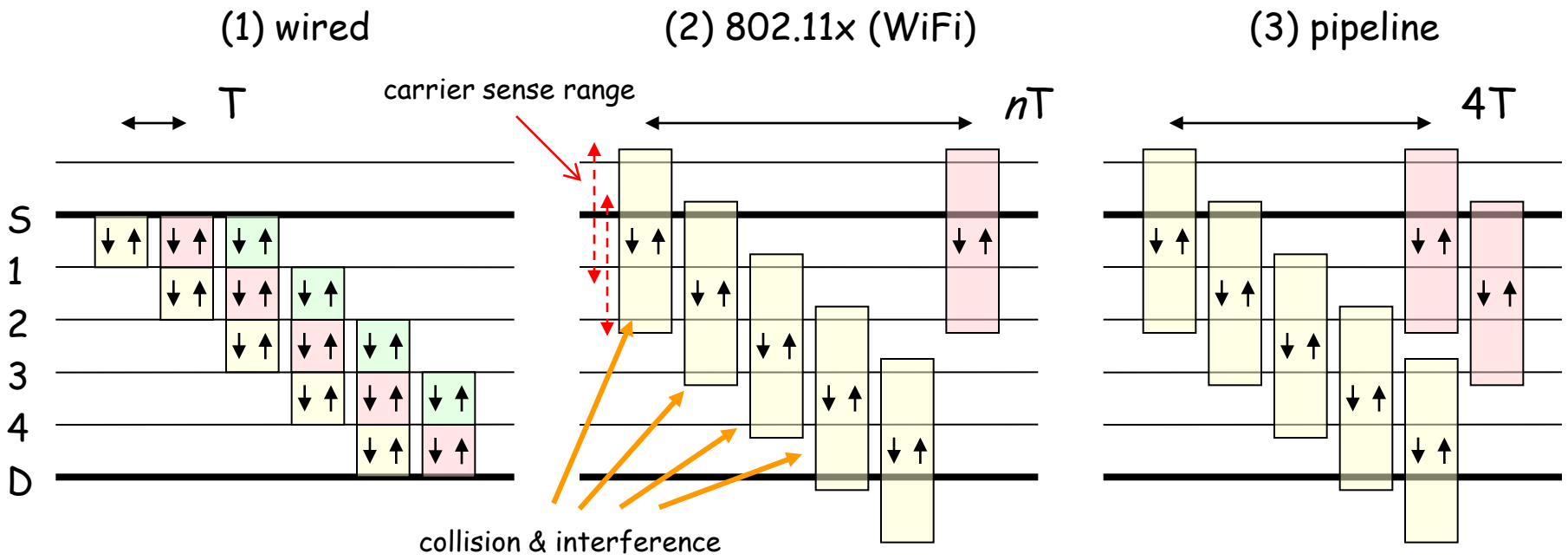
Decrease of link utilization due to radio interferences



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

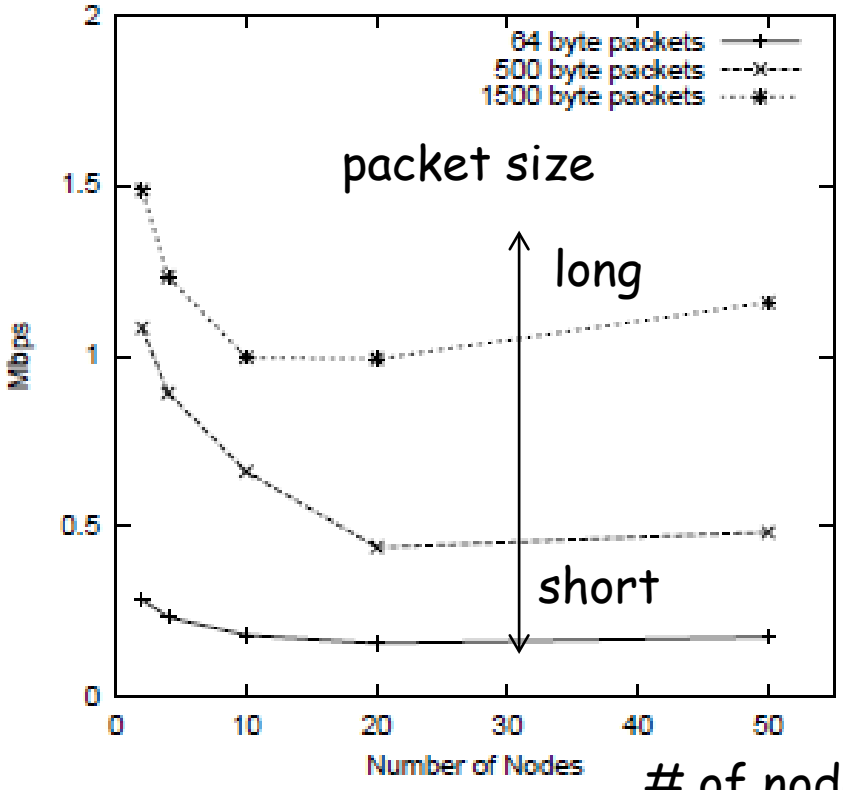
avoid collision & interference

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

Multihop Capacity (2)

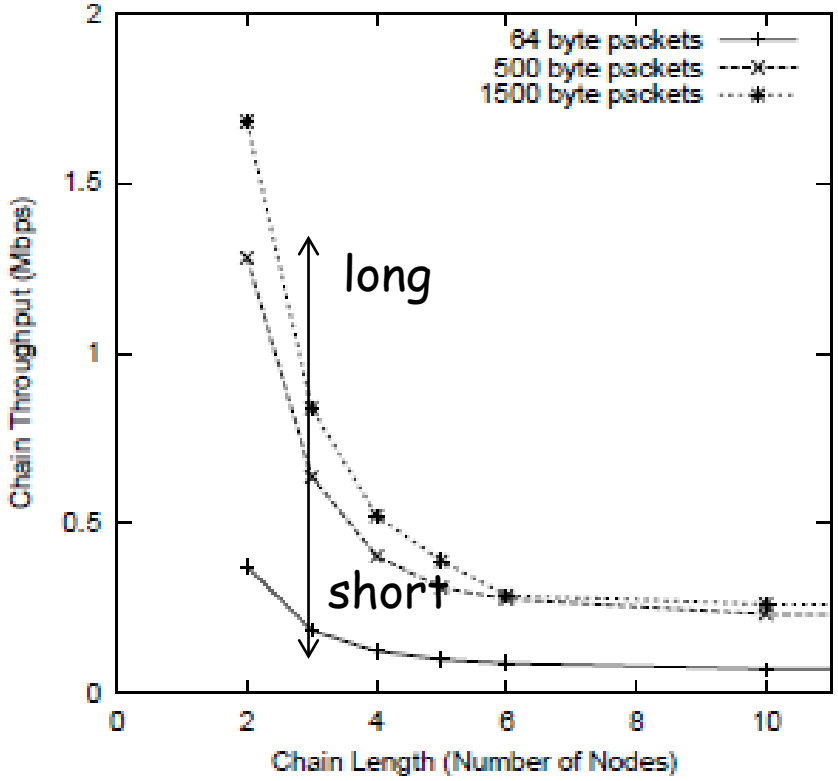
throughput

2Mb/s link



# of nodes

Cell (2D)



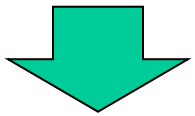
# of hops

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)



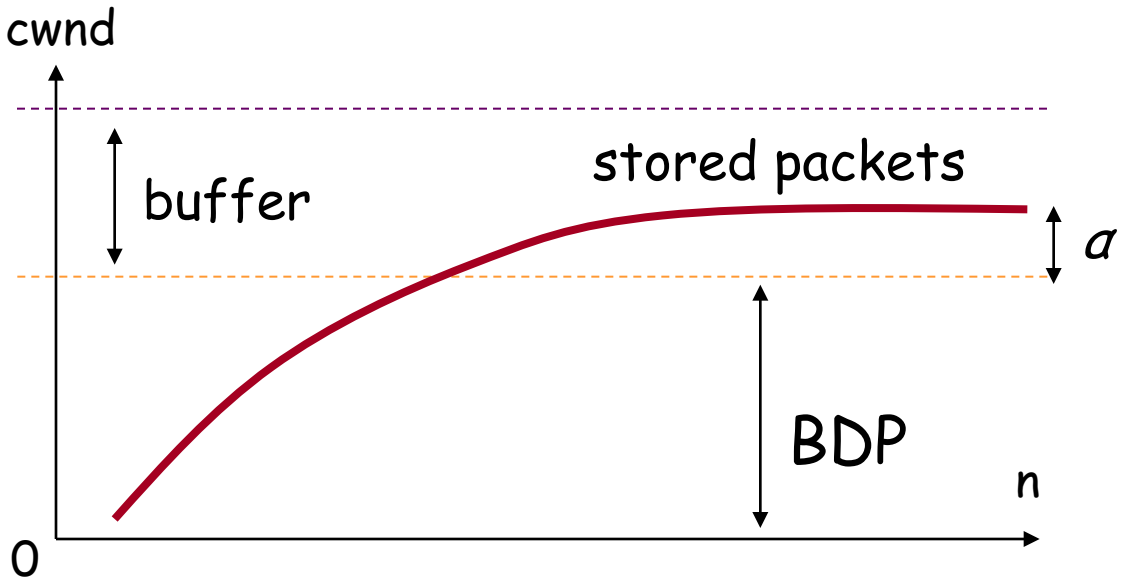
- Distributed Link RED: drops packets randomly at the link level when link load increases (analogous to random early detection)
- Adaptive Pacing: controls packet transmission scheduling in order to approach 1/4 (spatial channel reuse)

	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

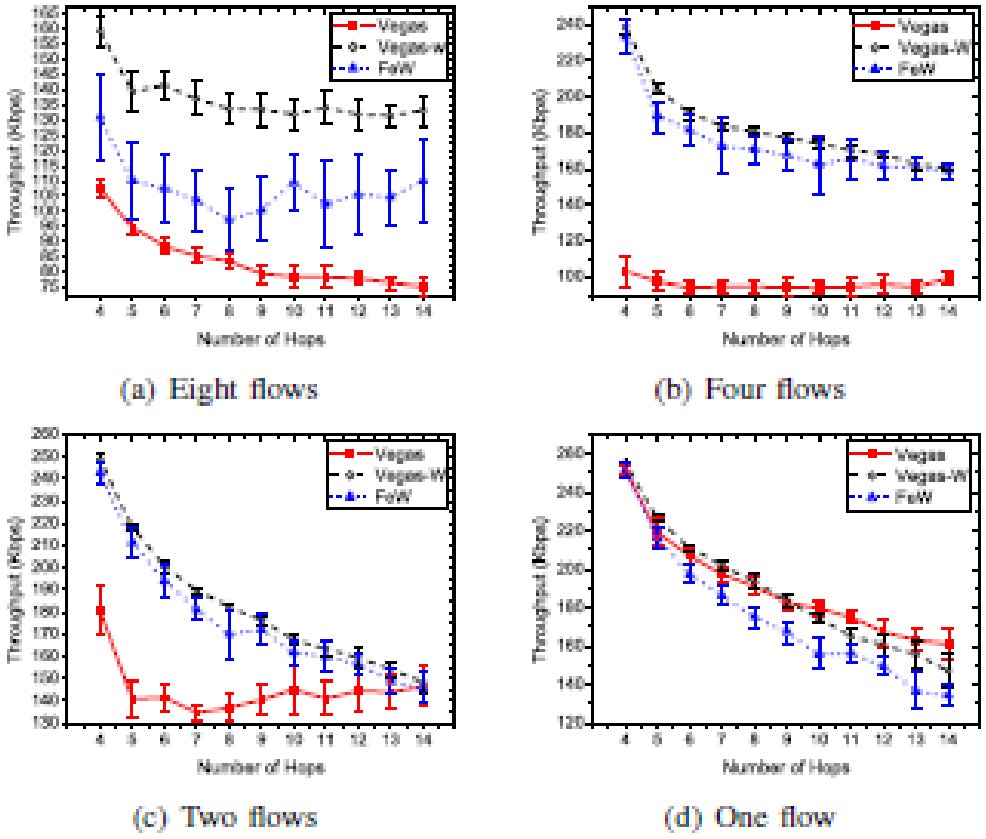
$(\alpha \le \Delta \le \beta \text{ or } \Delta \le \alpha \ \& \ n_{CA} \le N_{CA})$

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)



(a) Eight flows (b) Four flows

(c) Two flows (d) One flow

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.

Slower Vegas (1)

for multihop & USN

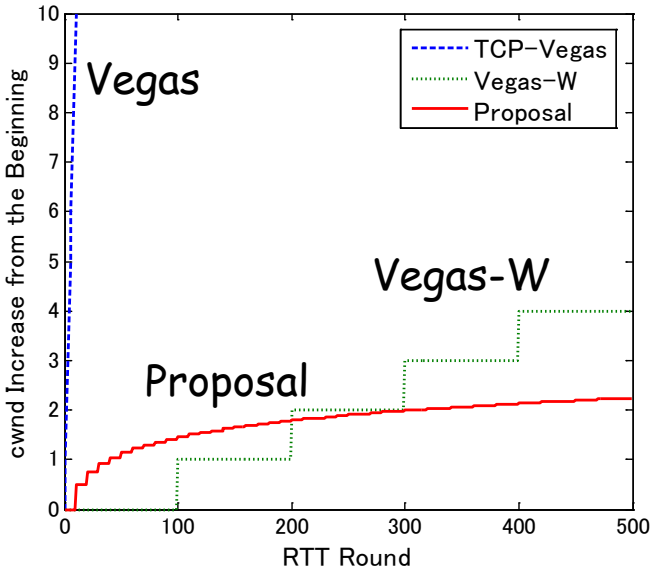
- Our proposal [IEICE, 2009]
  - **Exponential decrease of window increase**

$$cwnd = \begin{cases} cwnd + 1/(cwnd \times 2 \times count) & (\Delta < \alpha \;\&\&\; succ > N) \\ cwnd & (\alpha \le \Delta < \beta \text{ or } \Delta < \alpha \;\&\&\; succ \le N) \\ cwnd - 1/cwnd & (\Delta \ge \beta) \end{cases}$$

succ: # of states consecutively entering into $\Delta < \alpha \;\&\&\; succ \le N$

count: suppression parameter to be incremented

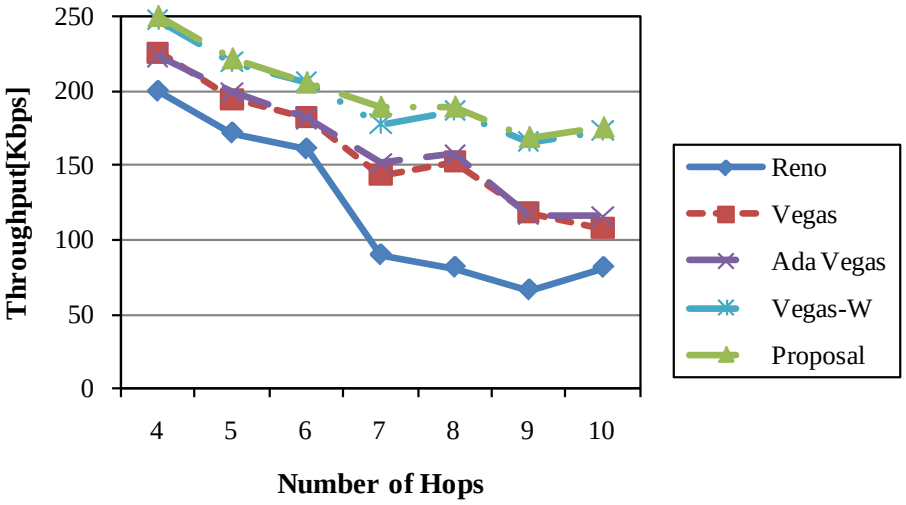
N: succ maximum (e.g. 10)



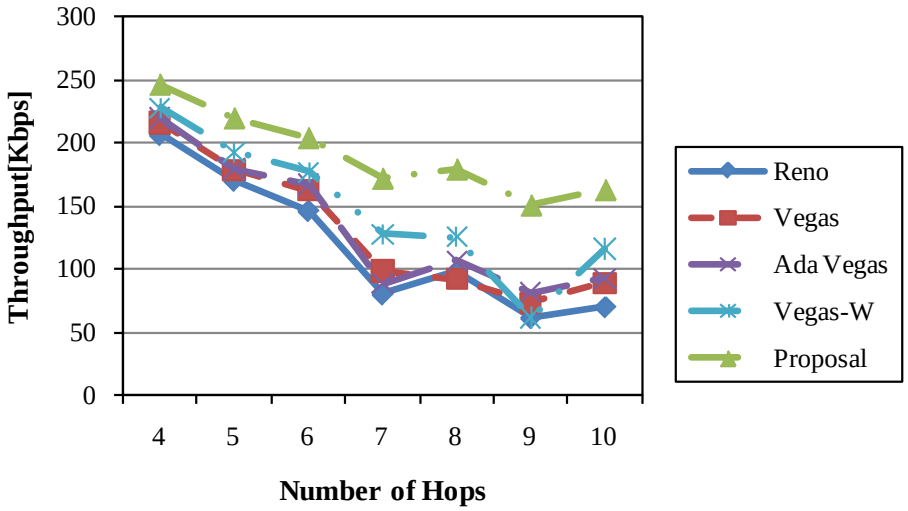
N.Iikubo et al: IEICE Tech. Report, 2009.

Slower Vegas (2)

※ ns-2 simulations



two flows



four flows

effective in slow link capacity,
but might be heuristic

802.11, 2Mbps

N TCP Flow

0 1 2 R

Sender Receiver

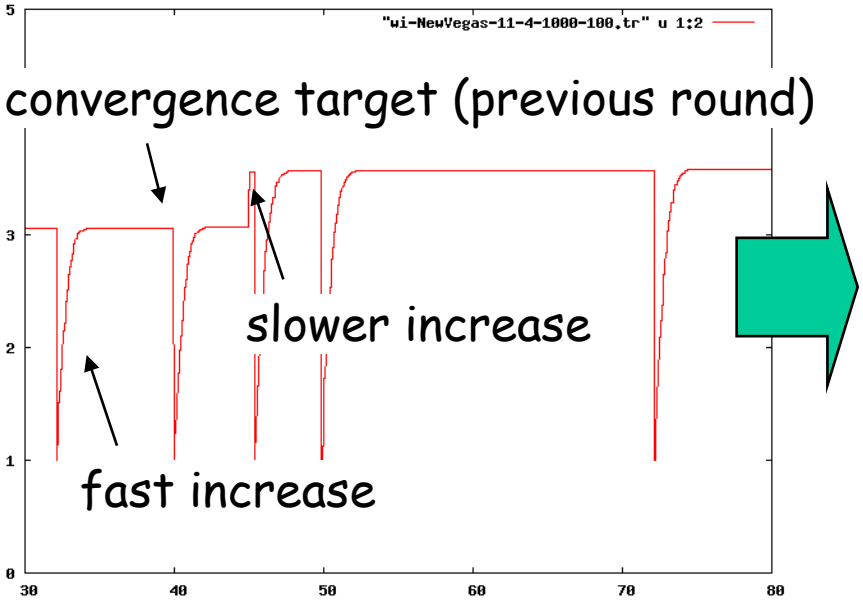
200m

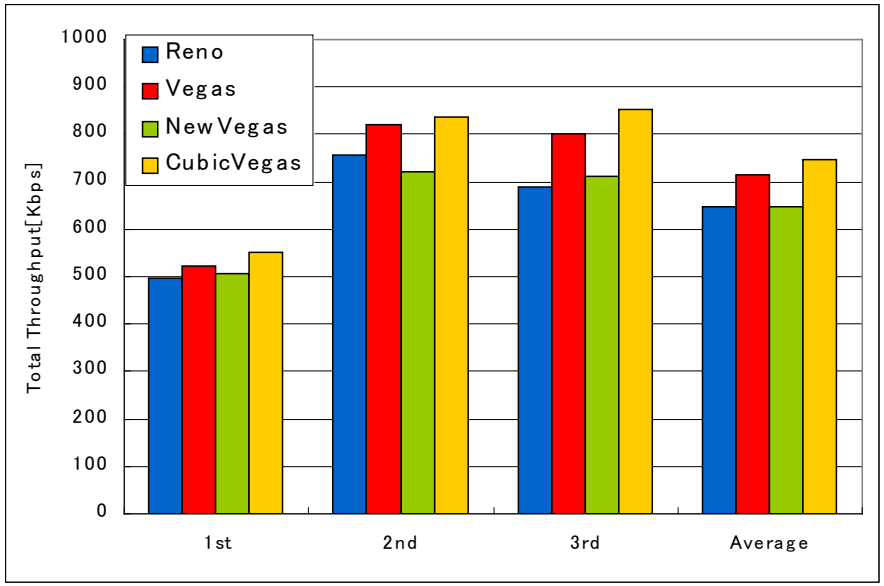
N.Iikubo et al: IEICE Tech. Report, 2009.

Slower Vegas (3)

for multihop & USN

- Our proposal [IEICE, 2011]
  - **CUBIC-Vegas (use cwnd history)**





N.Iikubo et al: IEICE Tech. Report, 2009.

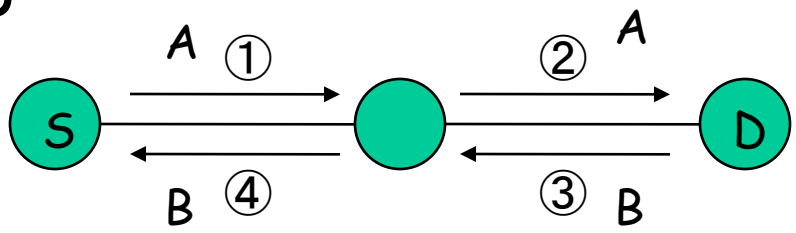
Summary of Wireless Multihop Networks

- 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

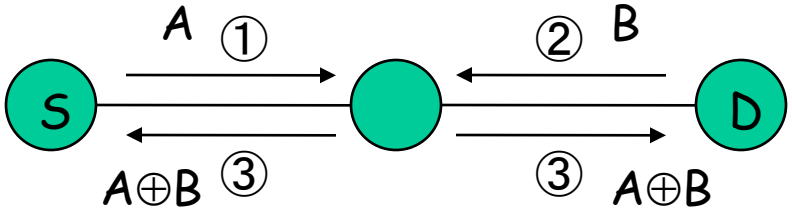
- Multihop



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

4 (channel efficiency1/4)

- Network Coding (in Wireless)



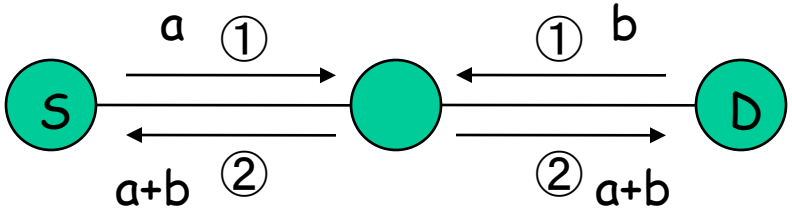
⊕: XOR

B=A⊕(A⊕B)

A=B⊕(A⊕B)

3 (channel efficiency 1/3)

- Physical-Layer Network Coding



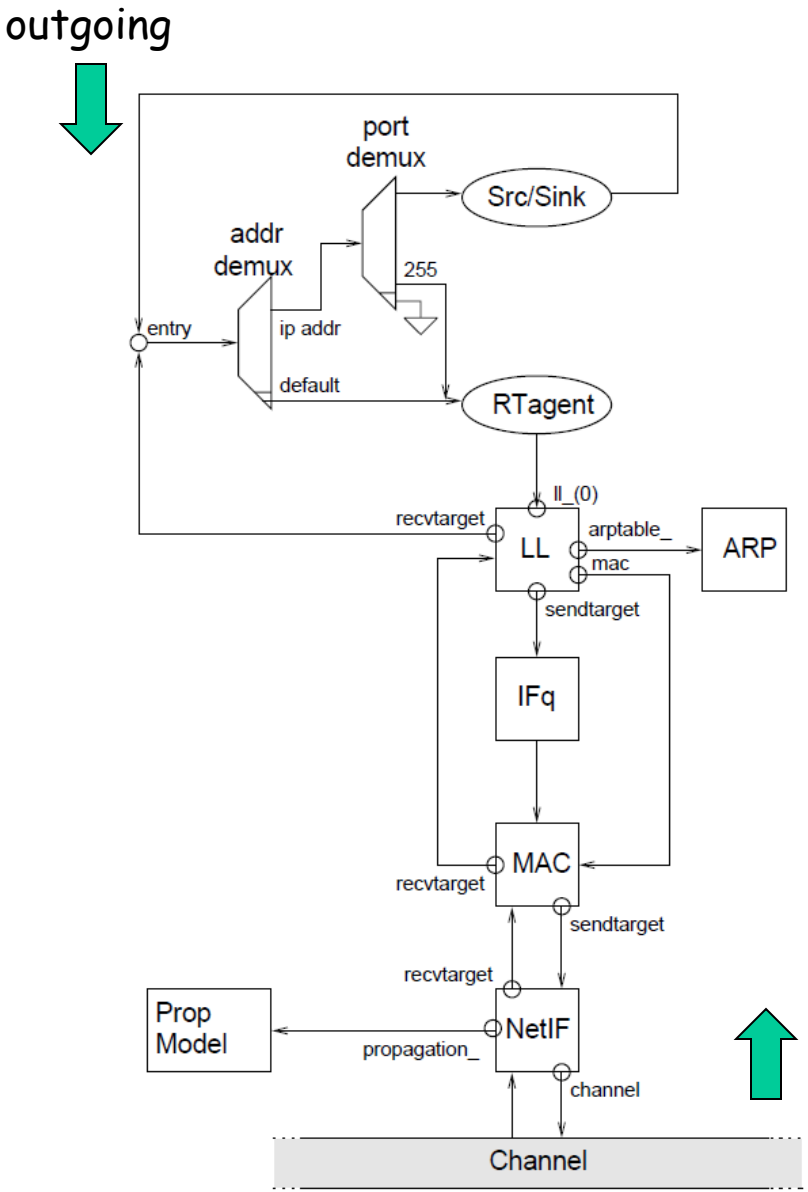
2 (channel efficiency 1/2)

synchronization is the key point

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

ns-2 wireless model

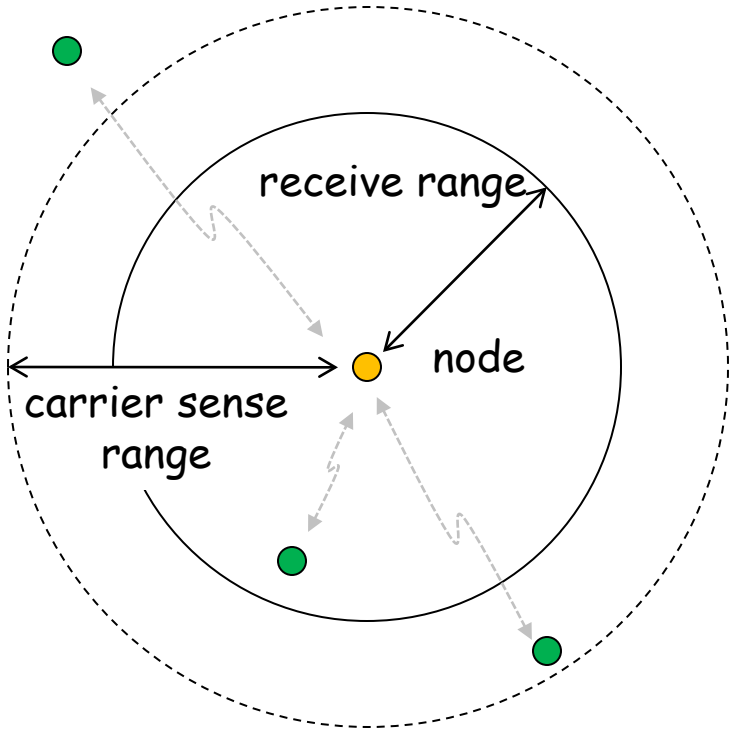
Mobile Node



- RTagent: routing protocol
- LL: link layer
- ARP: ARP table
- IFq: interface queue
- MAC: media access control layer
- NetIF: network interface
- Prop Model: radio propagation
- Wireless Channel
- Mobility Models

CMU Monarch Project: "The CMU Monarch Project's Wireless and Mobility Extensions to ns", Aug.1999.

Radio Propagation (1)



- definition
  - carrier sense range: a node can detect signals
  - receive range: a node can receive packets
  - physical carrier sense: direct signal sensing
  - virtual carrier sense: indirect carrier sensing via RTS/CTS messages

J.Broch et al: "A Performance Comarison of Multi-Hop Ad Hoc Network Routing Protocols", ACM MOBICOM 1998.

Radio Propagation (2)

- parameters
  - Pr: receiving power(function of a distance between nodes)
  - Pr.prev: receiving power of the preceding packets
  - CSThresh: power threshold for carrier sensing
  - RXThresh: power threshold for packet reception
  - CPThresh: power difference which can avoid packet collisions

```
// Network interface
if ( Pr < CSThresh )
  discard as "noise"
elseif ( Pr < RXThresh )
  mark as "error" packet
else
  receive a new packet, goto MAC
  // MAC layer
  if ( state is not "idle" )
    if ( Pr.prev > Pr + CPThresh )
      "capture", drop the new packet
    else
      "collision", drop both packets
  else // ( i.e. state is "idle" )
    receive the new packet
```

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 18: Radio Propagation Models.

Radio Propagation (3)

- Free space model (Friis formula, direct)

$$P_r(d) = \frac{P_t G_t G_r \lambda^2}{(4\pi)^2 d^2 L} \approx \frac{P_t \lambda^2}{(4\pi)^2 d^2}$$

square of distance

d : distance
Pr(d): receiving power
Pt: transmission power
λ: wavelength

- Two-way ground reflection model (direct + reflection)

biquadrate of distance

$$P_r(d) = \frac{P_t G_t G_r h_t^2 h_r^2}{d^4 L} \approx \frac{P_t h_t^2 h_r^2}{d^4}$$

ht : height of transmission antenna
hr: height of receiving antenna

- near ~ Friis、far ~ Two-ray

cross-over distance:

$$d_c = (4\pi h_t h_r) / \lambda$$

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 18: Radio Propagation Models.

Radio Propagation (4)

- example

tcl/ex/wireless-test.tcl

```
set opt(chan)           Channel/WirelessChannel
set opt(prop)           Propagation/TwoRayGround
set opt(netif)          Phy/WirelessPhy
set opt(ant)            Antenna/OmniAntenna
...
Antenna/OmniAntenna set X_ 0
Antenna/OmniAntenna set Y_ 0
Antenna/OmniAntenna set Z_ 1.5          // height of antenna
Antenna/OmniAntenna set Gt_ 1.0         // transmission gain
Antenna/OmniAntenna set Gr_ 1.0         // reception gain

Phy/WirelessPhy set CPThresh_ 10.0          // capture threshold
Phy/WirelessPhy set CSThresh_ 1.559e-11     // carrier sense threshold (550m)
Phy/WirelessPhy set RXThresh_ 3.652e-10     // packet reception threshold (250m)
Phy/WirelessPhy set Rb_ 2*1e6               // bit rate
Phy/WirelessPhy set Pt_ 0.2818              // transmission power
Phy/WirelessPhy set freq_ 914e+6            // frequency (⇔ wavelength)
Phy/WirelessPhy set L_ 1.0                  // system loss
```

indep-util/propagation/threshold.cc

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 18: Radio Propagation Models.

Radio Propagation (5)

ns-2.34

- Channel:
  - WirelessChannel in mac/channel.cc
- Propagation:
  - FreeSpace in mobile/propagation.cc
  - TwoRayGround in mobile/tworayground.cc
- Antenna:
  - OmniAntenna in mobile/omni-antenna.cc
- Phy/WirelessPhy (NetIF):
  - WirelessPhy in mac/wireless-phy.cc
  - WirelessPhyExt in mac/wireless-PhyExt.cc (with mac/mac-802_11Ext.cc)
  - 802_15_4 (ZigBee) in wpan/p802_15_4phy.cc

Routing protocols

- Ad hoc network routing protocols:
  - DSDV (Destination-Sequenced Distance Vector)
  - AODV (Ad hoc On-demand Distance Vector)
  - DSR (Dynamic Source Routing)
  - TORA (Temporally-Ordered Routing Algorithm)
  - AOMDV (Ad hoc On-demand Multipath Distance Vector)

- Data centric routing (for sensor networks):
  - Directed Diffusion

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 16: Mobile Networking in ns.

MAC protocols

- TDMA
  - preamble slot (signaling channel)
  - data transmission slot
  - in mac/mac-tdma.cc
- CSMA/CA
  - IEEE 802.11 DCF (Distributed Coordination Function) & extensions
  - IEEE 802.15.4 (ZigBee)
    - in wpan/p802_15_4mac.cc

802.11 MAC extensions

- 802.11 (CMU original: ad hoc mode)
  - 802_11 in mac/mac-802_11.cc

---

*: since ns-2.33

- 802.11 infrastructure mode *
  - Access point (active scanning, authentication & association, inter-AP comm., mobility support)
  - 802_11 in mac/mac-802_11.cc
    - Usage: $mac **ap** [$mac id]
- 802.11 Ext *
  - New implementation of 802.11 (support modulation & fading models)
  - 802_11Ext in mac/mac-802_11Ext.cc
- dei802.11mr *
  - Supports multiple PHY channels & SINR-based packet level error models

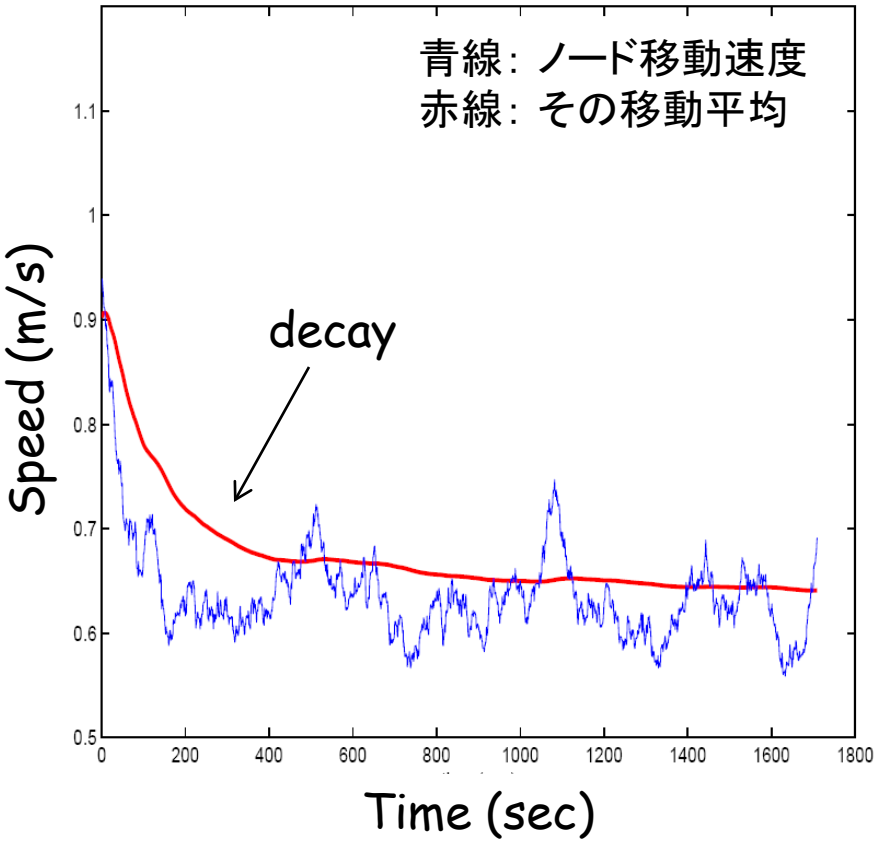
Mobility support (1)

- Random waypoint model (in ns-2)
  - Move & pause
  - Node movement generator
    - indep-utils/cmu-scen-gen/setdest/
  - Traffic generator
    - indep-utils/cmu-scen-gen/

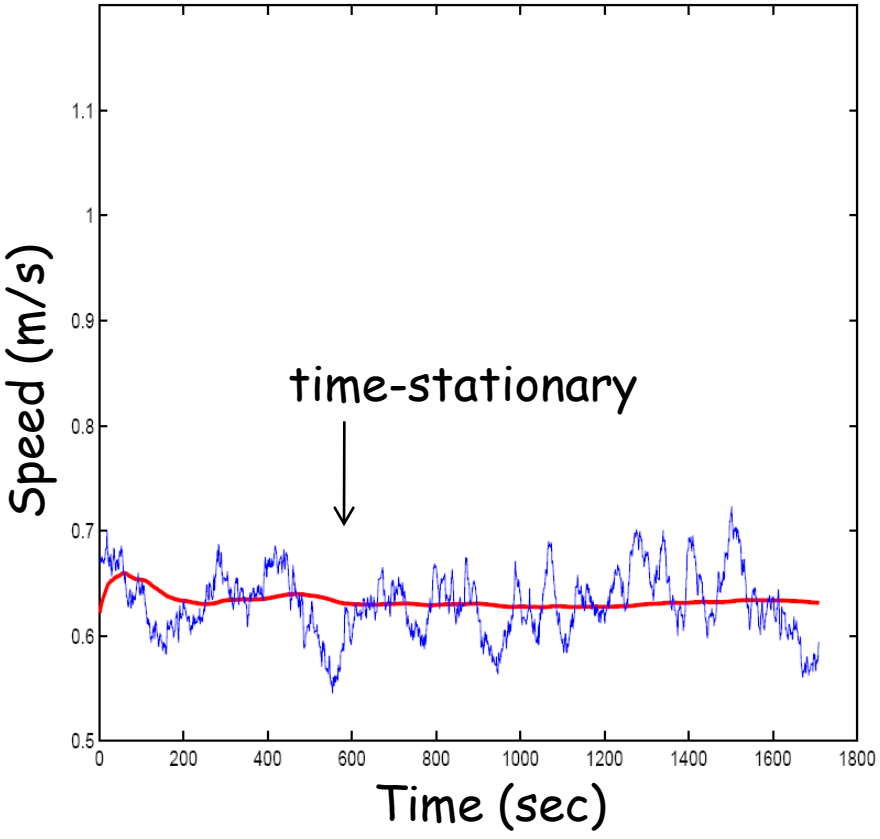
- Random trip model
  - IEEE INFOCOM 2005
  - Generalization of random waypoint
  - http://icawww1.epfl.ch/RandomTrip/

Mobility support (2)

- random waypoint



- random trip



http://icawww1.epfl.ch/RandomTrip/

Mobility support (3)

- example

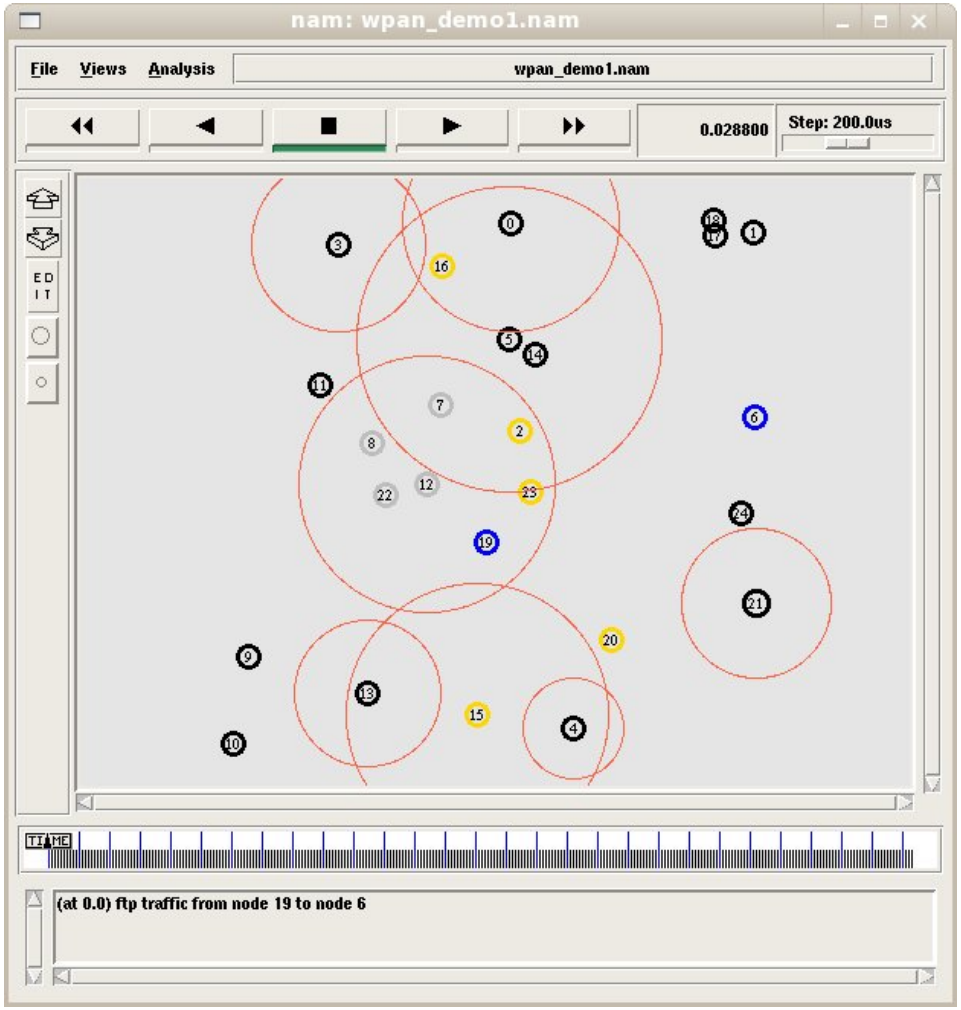
tcl/ex/wireless-test.tcl

```
set opt(mac)            Mac/802_11
set opt(ifq)            Queue/DropTail/PriQueue
set opt(ll)             LL
...
set opt(x)              670         ;# X dimension of the topography
set opt(y)              670         ;# Y dimension of the topography
set opt(cp)             "../mobility/scene/cbr-3-test"
set opt(sc)             "../mobility/scene/scen-3-test"

set opt(ifqlen)         50          ;# max packet in ifq
set opt(nn)             3           ;# number of nodes
set opt(seed)           0.0
set opt(stop)           2000.0      ;# simulation time
set opt(tr)             out-test.tr;# trace file
set opt(rp)             dsr         ;# routing protocol script
set opt(lm)             "off"       ;# log movement
...
$ns_ namtrace-all-wireless $nf $opt(x) $opt(y)
```

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 16: Mobile Networking in ns.

Mobility support (4)



tcl/ex/wpan/wpan_demo1.tcl

http://www.isi.edu/nsnam/ns/tutorial/index.html, Section 16: Mobile Networking in ns.

other resources

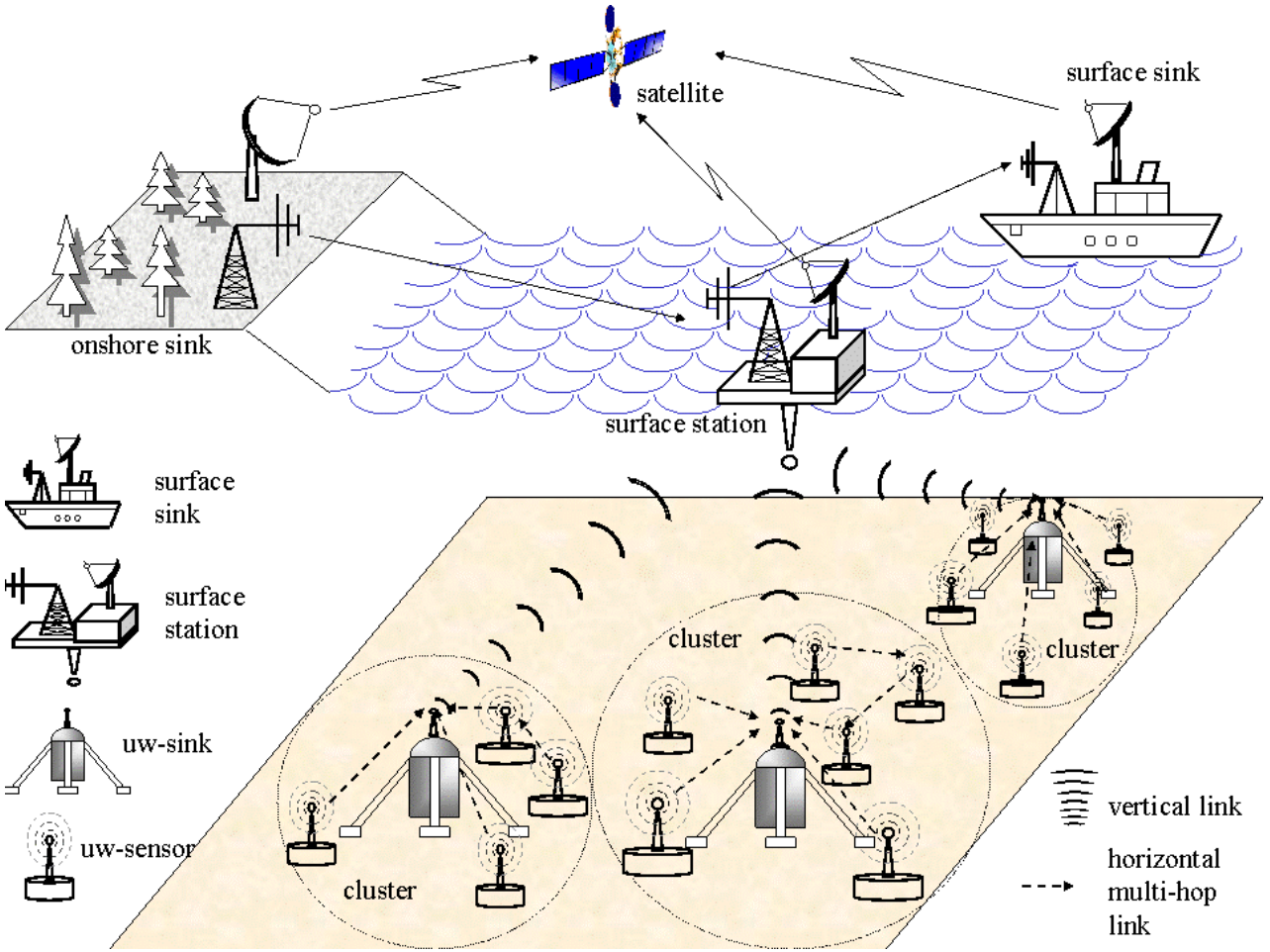
- Mobile Networking in ns
  - wired-cum-wireless (in ns-2)
  - MobileIP extension (in ns-2)
- IEEE 802.11e (Wi-Fi Multimedia)
  - http://sourceforge.net/projects/ieee80211e-ns2/
  - http://www-sop.inria.fr/planete/qni/Research.html
- IEEE 802.16 (WiMAX)
  - http://ndsl.csie.cgu.edu.tw/wimax_ns2.php
- Data aggregation (inc. Directed Diffusion)
  - LEACH: http://www.internetworkflow.com/resources/ns2leach.pdf
  - SPIN: http://www.ece.rochester.edu/~wheinzel/research.html
- Energy-aware MAC
  - SMAC (in ns-2)
  - ZMAC/DRAND: http://www4.ncsu.edu/~rhee/export/zmac/
- Underwater sensor networks
  - http://www.nbqadri.com/ns2/
  - http://ee.washington.edu/research/funlab/uan/uansim.html
- MPEG video over ns-2
  - http://hpds.ee.ncku.edu.tw/~smallko/ns2/Evalvid_in_NS2.htm

Underwater Sensor Networks

Discussion

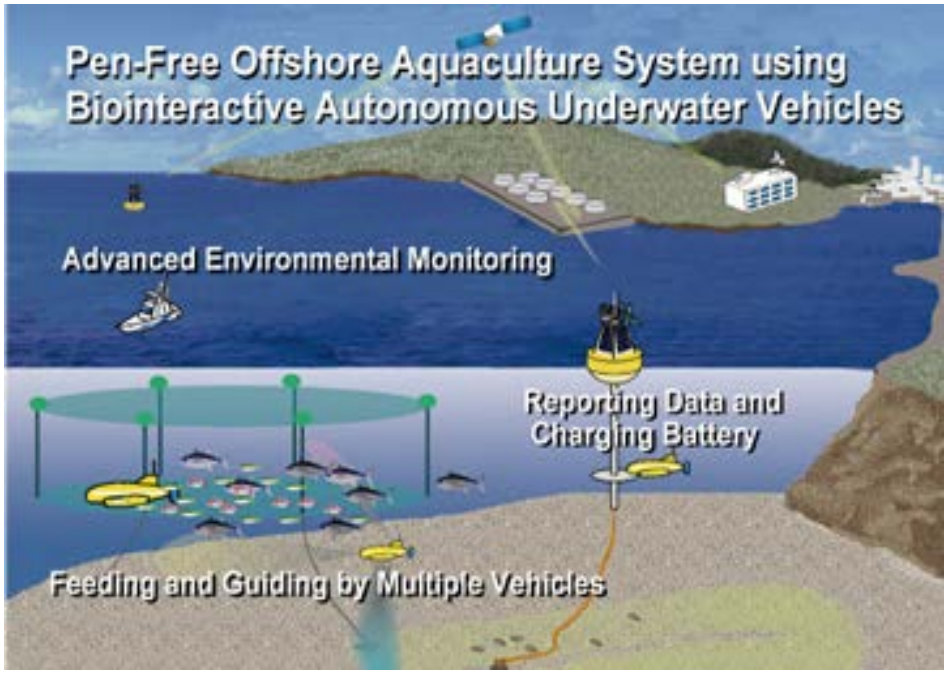
- Uniqueness of underwater sensor networks
  - use acoustic signals instead of electric wave
    - speed of light: 300 000 000 m/s
    - speed of sound (underwater): 1500m/s
  - link utilization ratio decreases as the distance increases
    - due to huge delay
  - interferences and collisions are similar

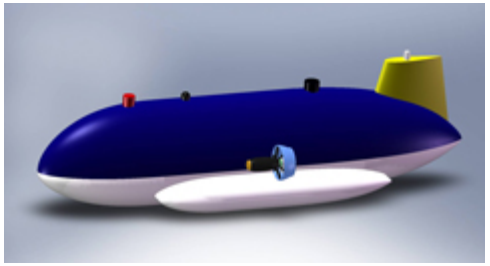
USN Architecture



http://www.ece.gatech.edu/research/labs/bwn/UWASN/

AUV (Autonomous Underwater Vehicle)





http://www.ocean.e.kaiyodai.ac.jp/jp/index.html

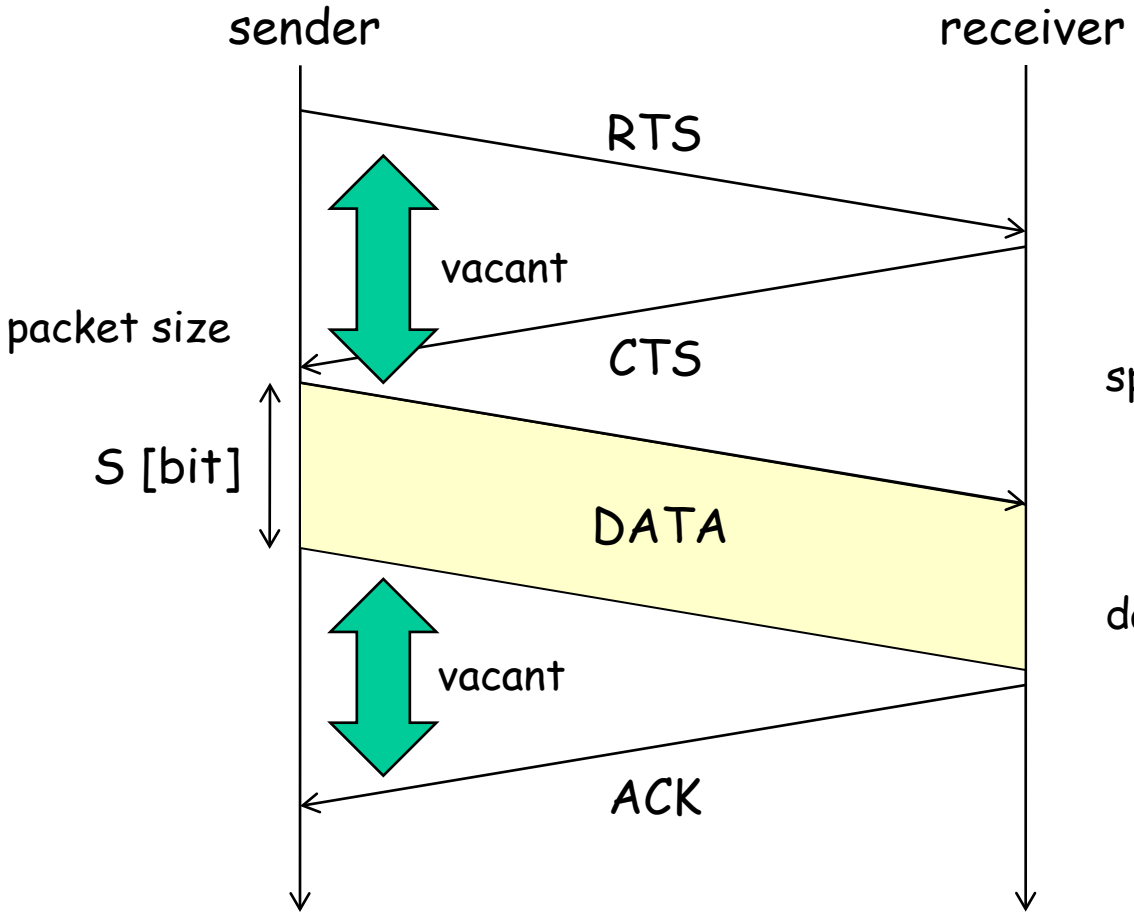
Characteristics of USN

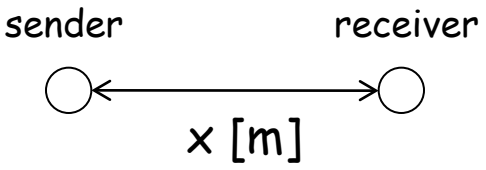
- USN uses acoustic signals
  - electric wave heavily attenuates in water
  - huge delay(1500m/s) ⇔ speed of light: $3*10^8$m/s
  - narrow bandwidth(~kHz)
  - long transmission range
  - other metrics: temperature, depth, angle, bubbles ...

Underwater Sensor Networks (1)

- Link Utilization (1)

Link utilization decrease due to slow sound speed





speed of sound $v = 1500$ [m/s]

bitrate C [bit/s]

data transmission time

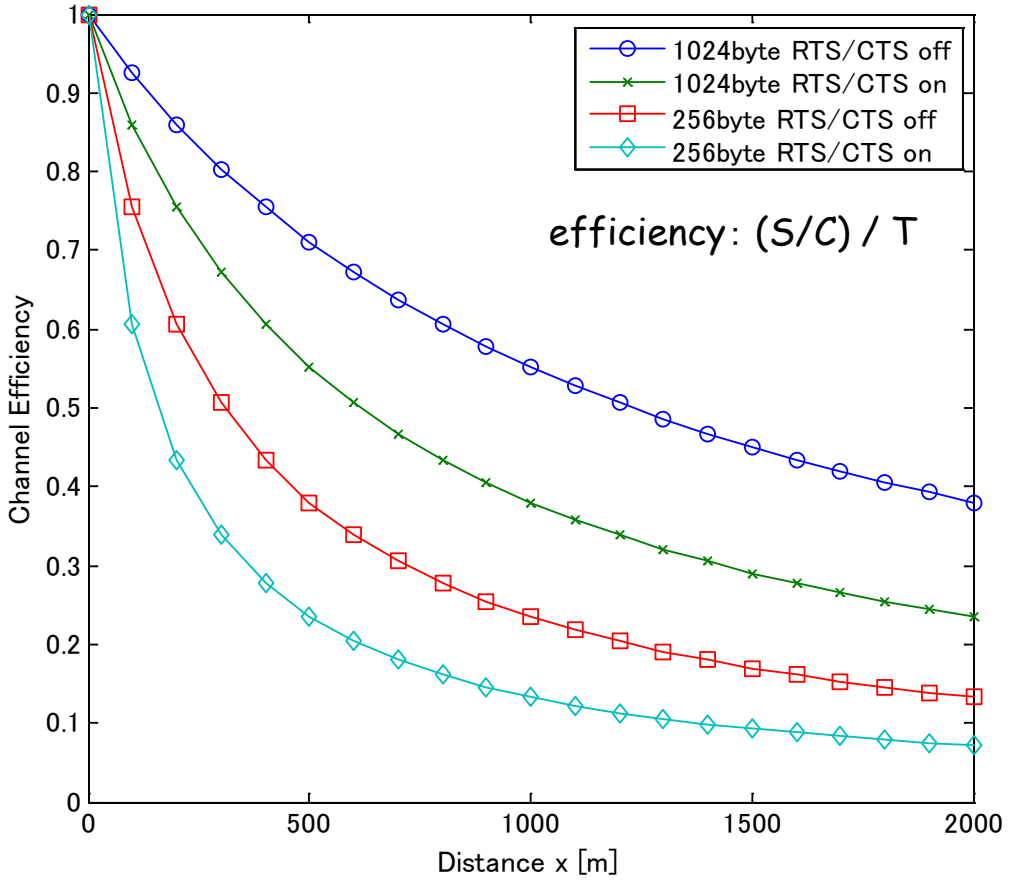
$$T = x/v * 2 + S/C + x/v * 2$$

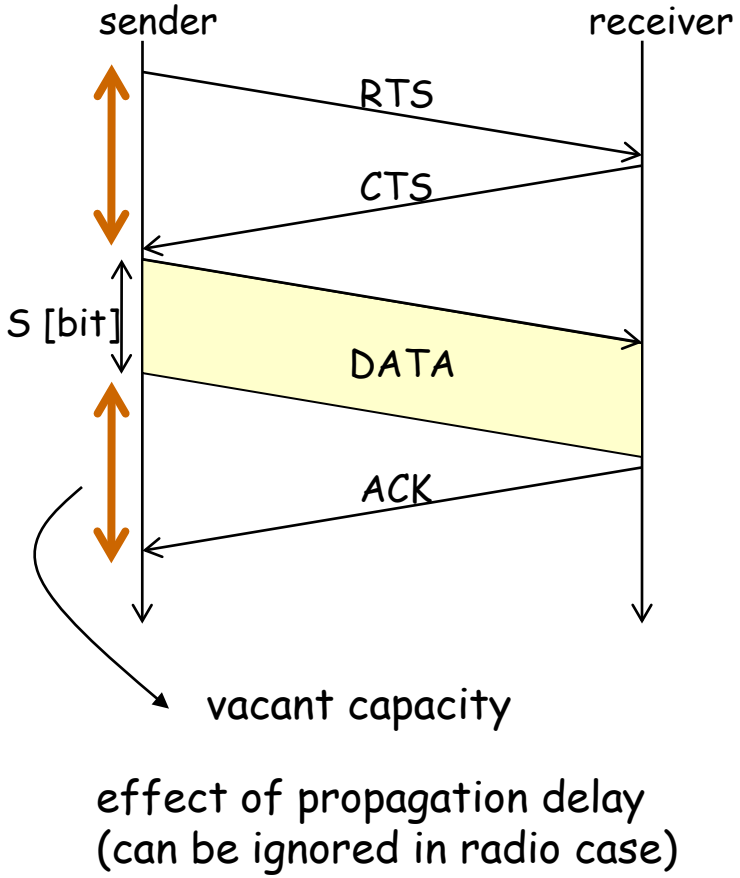
RTS/CTS DATA/ACK

Underwater Sensor Networks (2)

- Link Utilization (2)

relationship between distance and link utilization



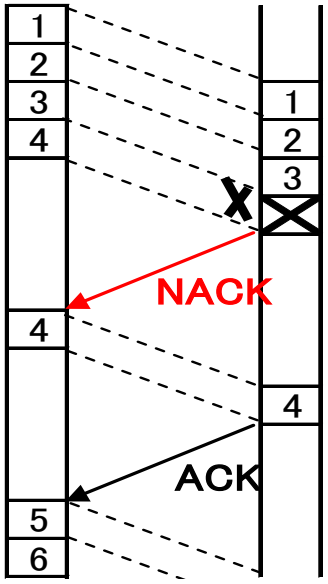


effect of propagation delay
(can be ignored in radio case)

MAC for USN (1)

Selective ARQ

J.Rice: ACM WuWNet 2007

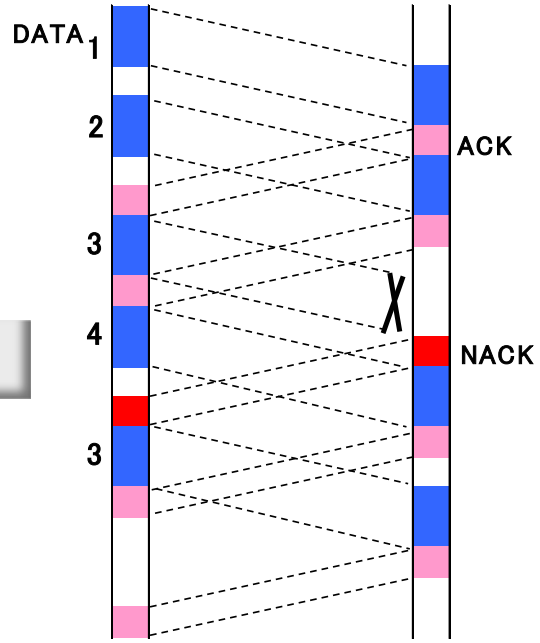


- Delayed NACK & ACK
- used in Seaweb prototype

combination

JSW ARQ

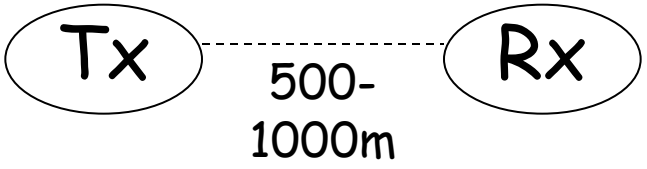
M.Gao et al., IEEE ICC 2009



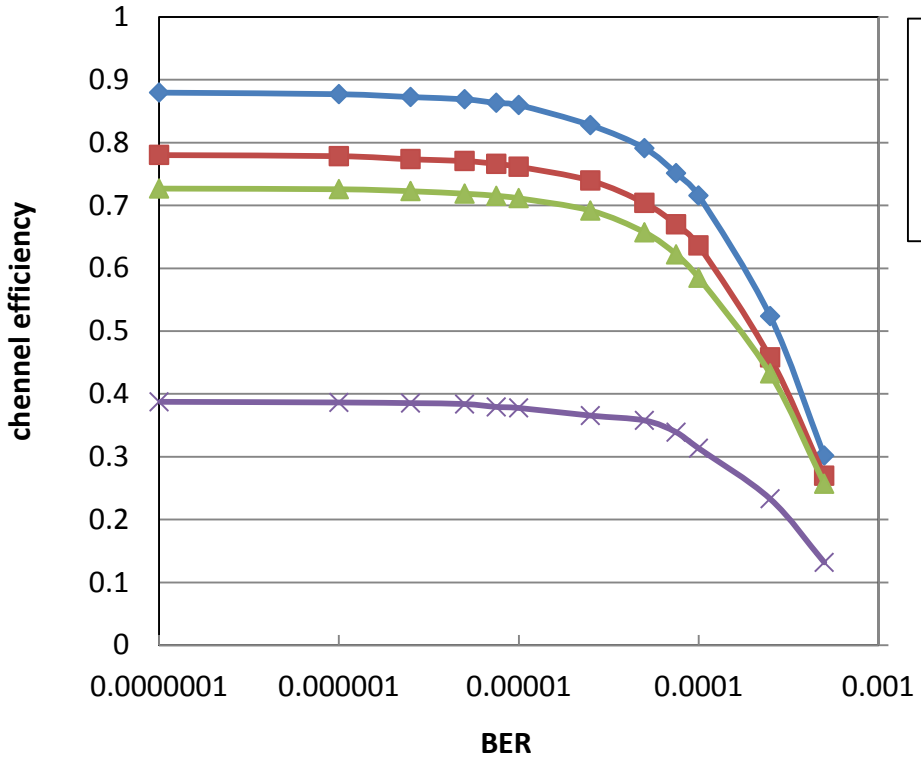
- 1 ACK for 1 DATA
- deliver multiple packets before ACK arrival
- need node synchronization

M.Yoshinaga et al: IEICE Tech. Report, 2011.

MAC for USN (2)



- 500m case



- Stop & wait (1 DATA / 1 ACK) doesn't work well due to slow sound speed
- Proposal (Selective ARQ + JSW) works the best

M.Yoshinaga et al: IEICE Tech. Report, 2011.

TCP for USN (1)

- TCP Hybla
  - TCP for satellite links having large RTT

$$W_{i+1}^{H} = \begin{cases} W_i^{H} + 2^{\rho} - 1 & (SS) \\ W_i^{H} + \rho^2 / W_i^{H} & (CA) \end{cases}$$

W^H: congestion window size

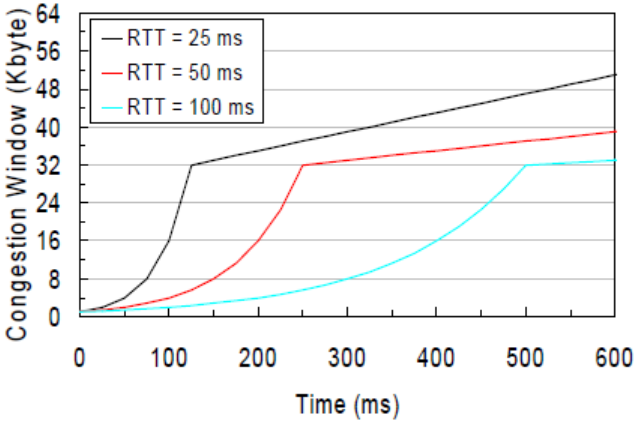
ρ: RTT/RTT_0

RTT: round trip time

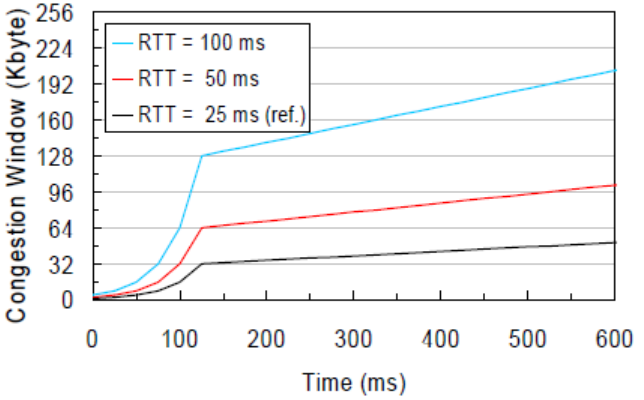
RTT_0: reference RTT (0.025[s])

SS: slow start

CA: congestion avoidance



Reno

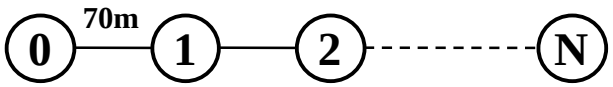


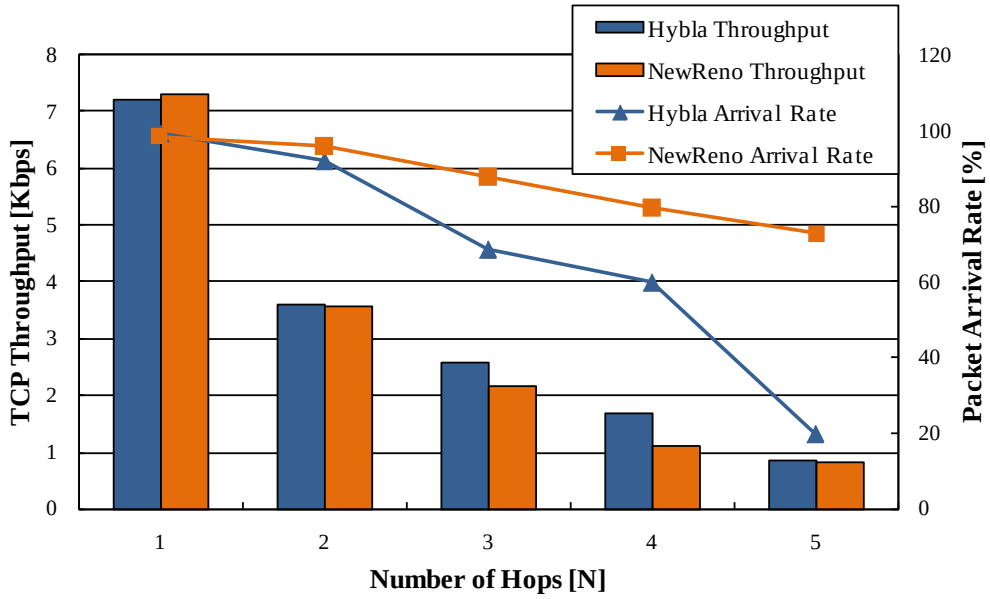
Hybla

C.Caini et al.: EAS ASMS 2003, Jul.2003.

TCP for USN (2)

- TCP Hybla usage for USN





Hybla provides better throughput than Reno, but its PLR is very high

Due to sudden RTT increase, Hybla's congestion window becomes extremely huge

This motivate us to consider lower window increase for multihop & USN (much lower packet loss & low delay)

N.Iikubo et al: IEICE Tech. Report, 2009.