TCP-Fusion: A Hybrid Congestion Control Algorithm for High-speed Networks

Kazumi KANEKO Tomoki FUJKAWA

Zhou SU Jiro KATTO

Graduate School of Science and Engineering, Waseda University

February 7 2007 at PFLDnet 2007

E-mail: kaneko@katto.comm.waseda.ac.jp

Outlines

- Backgrounds
- Existing Protocols
  - Loss-based Protocol
  - Delay-based Protocol
  - Loss-based protocol using delay metric
- The design of TCP-Fusion
- Implementation Experiments
- Simulation Experiments
- Conclusions

Backgrounds

- Throughput deterioration of TCP-Reno in high-speed networks with large Bandwidth-Delay product
  - TCP-Reno's window control is too conservative in increase and drastic in decrease in high-speed networks

- Many TCP variants have been proposed to achieve mainly three requirements ;
  - Efficiency
    - Improving throughput in high-speed networks
    - Robustness to random packet losses
  - Friendliness
    - Coordination with TCP-Reno
  - Fairness
    - Among coexisting different RTT flows
    - Stability of throughput/Congestion window
    - Convergence

Existing Protocols

- Loss-based Protocol
  - High-speed TCP(HSTCP) [S. Floyd RFC3649, 2003], BIC-TCP [L. Xu INFOCOM 2004]
    - Quickly increase and slowly decrease than TCP-Reno
  - TCP-Westwood [C. Casetti Mobicom 2001]
    - Adaptive decrease mechanism via the bandwidth/rate estimation
  - CUBIC [I. Rhee PFLDnet 2005]
    - Based on elapsed time since the last congestion event
- Delay-based Protocol
  - FAST TCP [C. Jin INFOCOM 2004]
    - Uses RTTs variations as a network congestion estimator

When these advanced protocols and TCP-Reno are competing, either throughput deteriorates to less than the fair share

Existing Protocols (cont'd)

- Combined loss-based and delay-based protocols
  - Gentle High-speed TCP [K. Tokuda, HSN 2003] /TCP-Africa [R. King, INFOCOM 2005]
    - Switch increase modes adaptively according to the delay information
  - TCP-Adaptive Reno (ARENO) [H. Shimonishi, PFLDnet 2006]
    - Rely on congestion level measurement
    - Adjusts increase/decrease parameters based on congestion level measurement
  - Compound TCP (CTCP) [K. Tan, INFOCOM 2006]
    - Implemented Windows Vista
    - Combined Loss-based and Delay-based windows
    - Similar efficiency as HSTCP

A new congestion control with efficiency and friendliness to TCP-Reno is essential

The design of TCP-Fusion

- TCP-Fusion belongs to ***Combined loss-based and delay-based approach***
- Maintain two congestion window sizes
  - *Loss-based window*
    - Like TCP-Reno
    - Increased by 1*MSS*/*RTT*
    - Halved upon a packet loss
  - *Delay-based window*
    - Based on delay information like TCP-Vegas
    - Adjust according to the left unused capacity
    - Set parameters to be a scalable manner for high-speed networks
    - Adaptive decrease upon a packet loss like TCPW-RE
- Adopt *max* (*Loss-based window*, *Delay-based window*)

Congestion window decrease

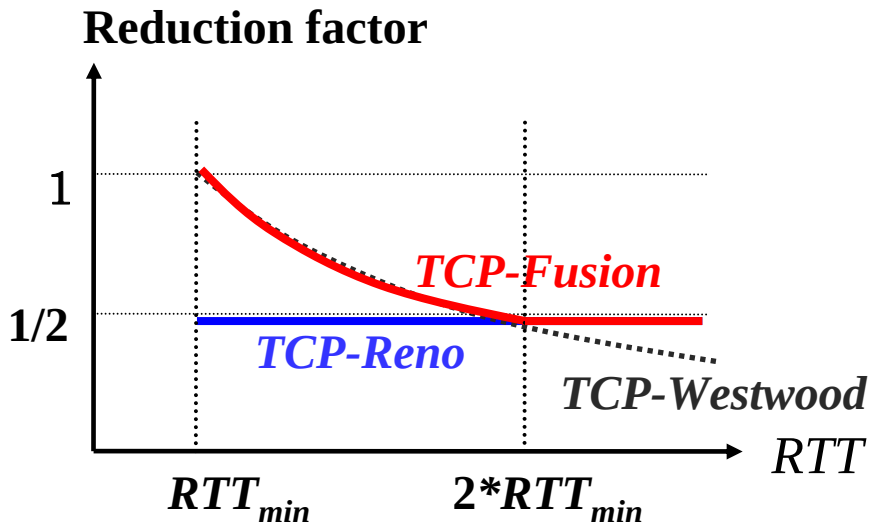
- The decrease parameter of TCPW-RE
  - Expressed as RTT_{min}/RTT
    - To clear the packets in the bottleneck queue
    - To keep the sending rate

⇒ It is friendly to Reno only if the buffer capacity is set to the BDP

- Set the threshold to the half
  - Not to work negatively than TCP-Reno

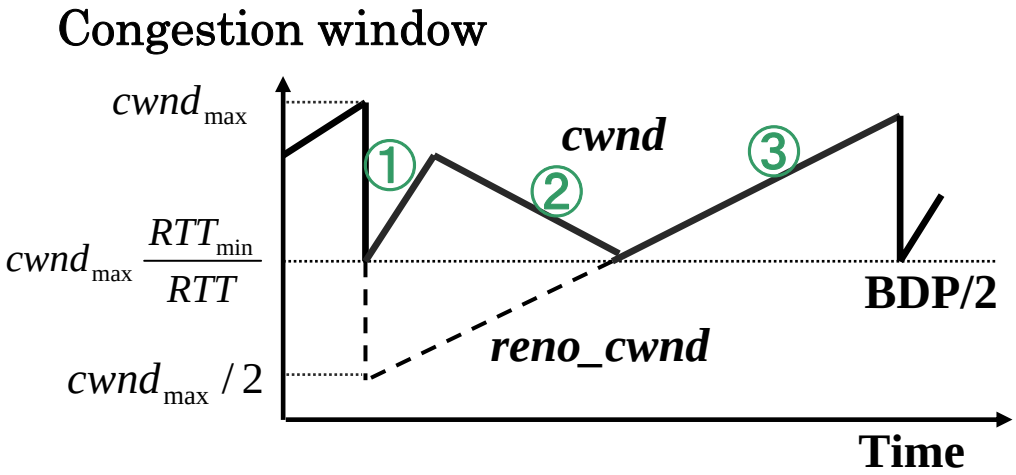
Decrease parameter

max{ RTT_{min} / RTT, 0.5}



(*Queuing delay=RTTmin*)
->BDP buffer size

Congestion window increase



# of packets in bottleneck queue

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

① $cwnd = cwnd + W_{inc} / cwnd$ if $diff < \alpha$

- Recognized as underutilized
- Increased rapidly to fill the link capacity

② $cwnd = cwnd + (-diff + \alpha) / cwnd$ if $diff > 3*\alpha$

- Recognized as utilized and early congestion
- Decreased to the value that has at least α in the bottleneck link queue

③ $cwnd = reno\_cwnd$ if $cwnd < reno\_cwnd$

- Ensure TCP-Reno's performance

Setting parameters

- Set α according to two requirements;
  - Small value in proportion to the link bandwidth ***B*** to fit in *diff* resolution
  - Inverse proportional value to the number of coexisting TCP-Fusion flow ***N***

•Assumption of minimum buffer size

$$G = \frac{B * D_{\min}}{packet\_size * 8}$$

– Corresponds to the queuing delay D_{min}

– **Proportional value to the link bandwidth**

•Set α

$$\alpha = \frac{G}{N} = \frac{(B/N) * D_{\min}}{packet\_size * 8} \approx \boxed{\frac{RE * D_{\min}}{packet\_size * 8}}$$

– **Inverse proportion to *N***

- W_{inc} is upper bounded by G

$$W_{inc} = \boxed{B/12Mbps}$$

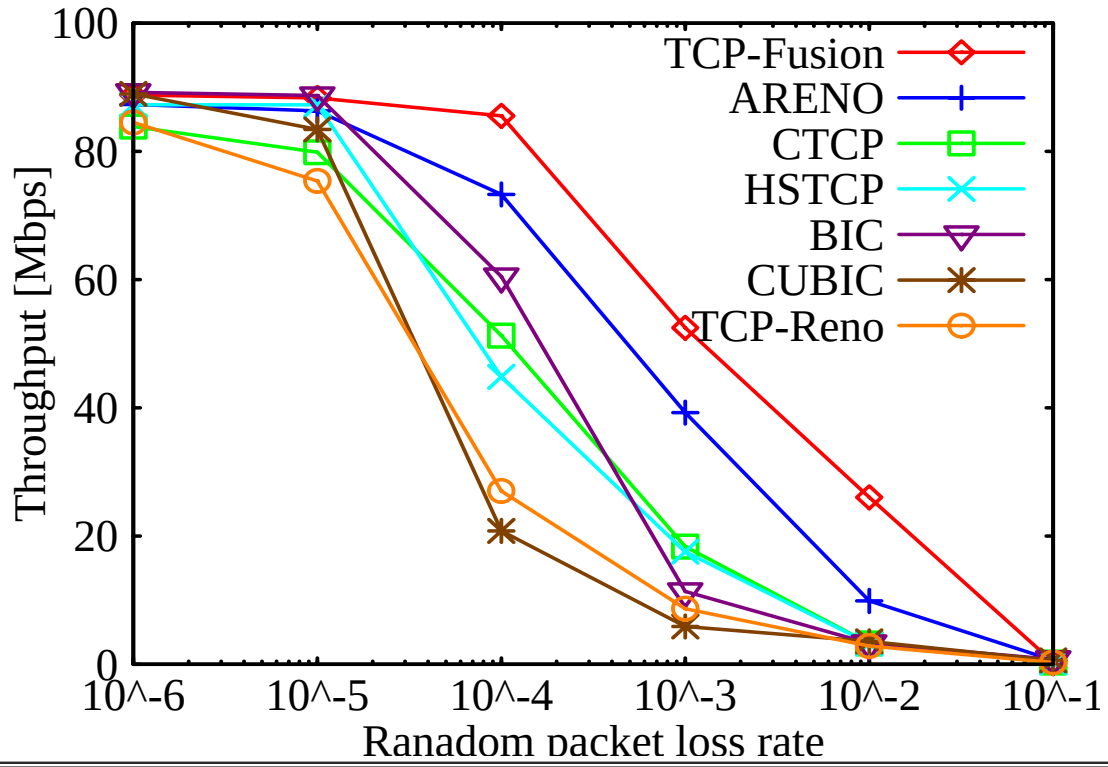
– Equal to G where D_{min}=1ms and 1500B packet size are assumed

Implementation Experiments

- Linux Implementation (Kernel 2.6.15)
- We newly implemented TCP-Fusion, ARENO and CTCP
  - Targets
    - TCP-Reno
    - High-speed TCP (HSTCP)
    - BIC-TCP (BIC)
    - CUBIC
    - TCP-Adaptive Reno (ARENO)
    - Compound TCP (CTCP)
    - TCP-Fusion
  - TCP send/receive buffer: 10MB
- Simple dumb-bell topology

Robustness to random packet loss

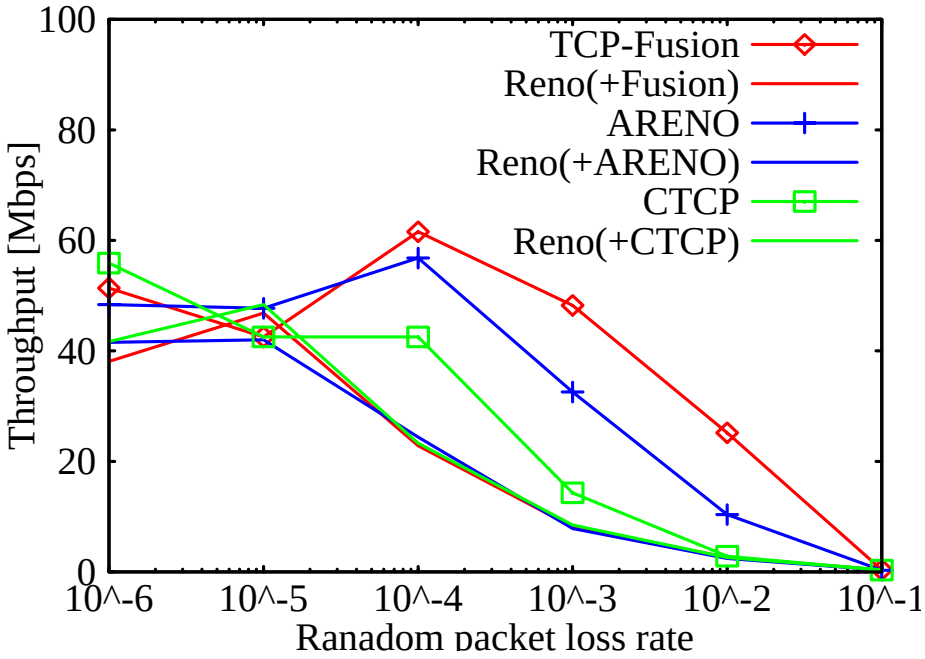
- A single flow throughput
  - Link bandwidth : 100Mbps, Buffer : BDP, RTT: 40ms
  - Packet loss rate: 10^{-6} - 10^{-1}

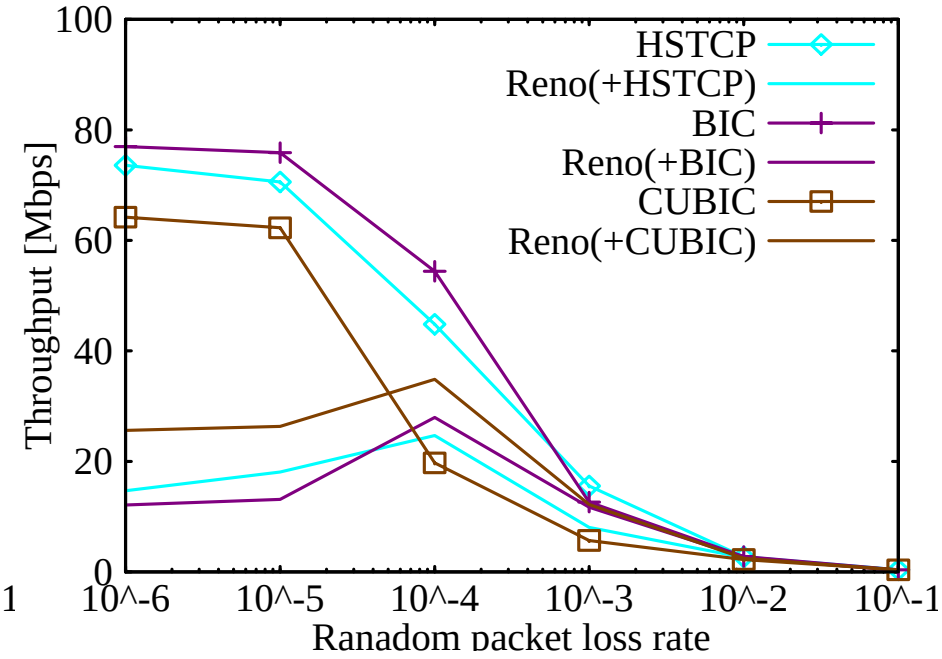


TCP-Fusion is most efficient and robust to random packet losses

Friendliness in lossy link

- Throughputs of coexisting TCP-Reno and TCP variant flows
  - Link bandwidth : 100Mbps, Buffer : BDP, RTT: 40ms
  - Packet loss rate: 10^{-6} - 10^{-1}





When the loss rate is small, TCP variant should share fairly with Reno, otherwise, use up the left unused capacity

Friendliness between TCP variants

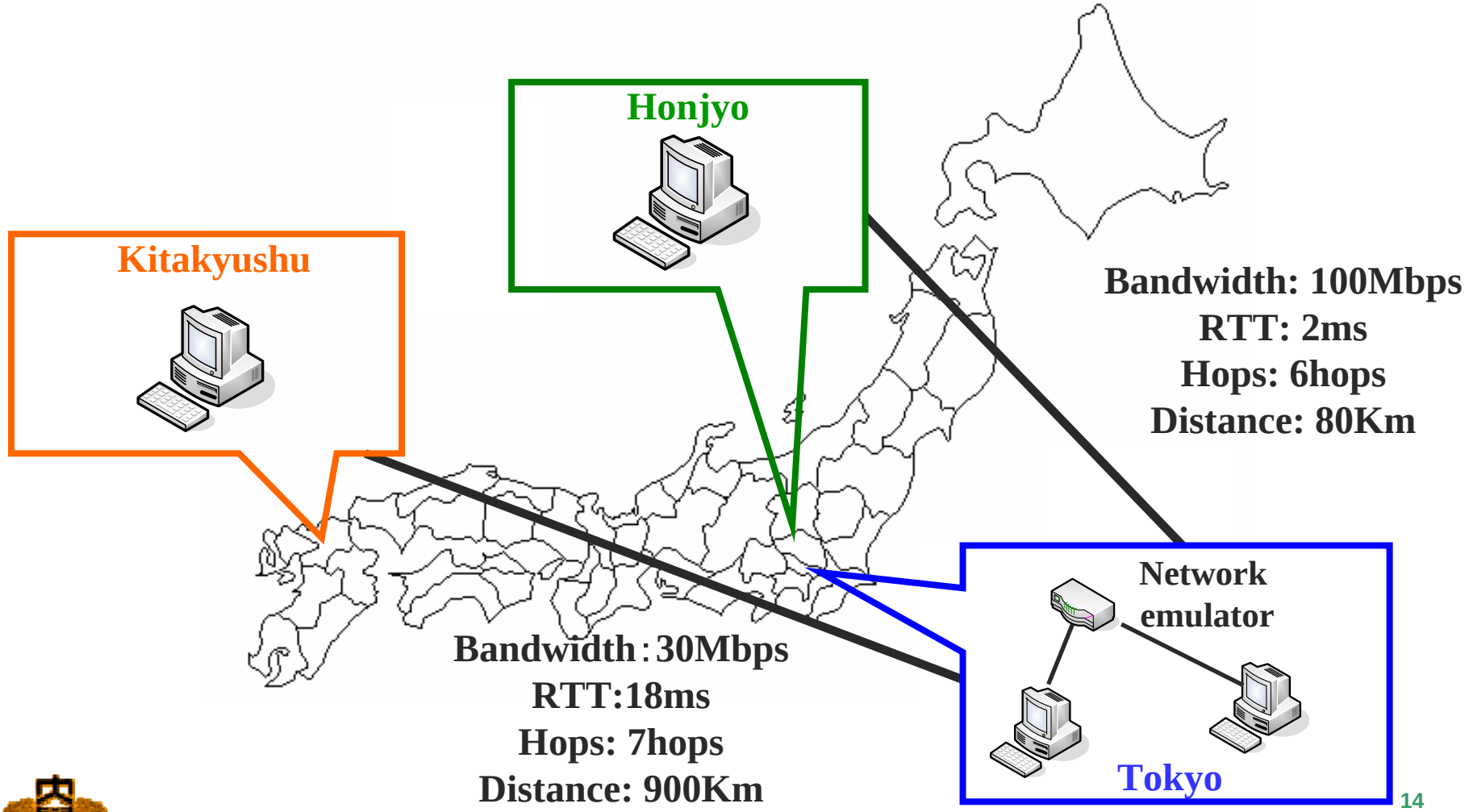
- Throughputs of coexisting two different TCP variant flows
  - RTT : 80ms Buffer : BDP/2
  - 0.0001 % random packet loss rate

	Reno	Fusion	ARENO	CTCP	HSTCP	BIC	CUBIC
Reno	44.89	40.24	42.35	38.58	10.32	10.44	18.58
Fusion	50.02	43.61	45.96	42.95	21.05	11.39	18.65
ARENO	40.52	44.01	43.24	37.96	16.47	12.19	19.36
CTCP	53.89	46.53	50.53	44.91	20.82	13.79	15.03
HSTCP	78.22	61.41	68.15	65.81	43.21	29.88	29.65
BIC	81.23	78.99	78.04	76.21	59.72	44.81	45.39
CUBIC	71.71	70.33	68.44	73.99	59.57	43.91	44.25

TCP variants with high friendliness have almost the same bandwidth

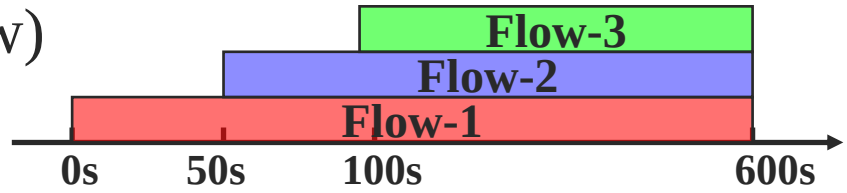
Actual network experiments

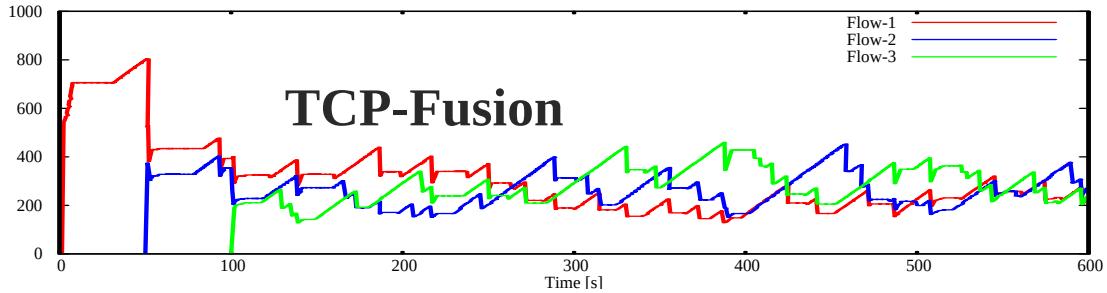
- Three end points and two actual network lines



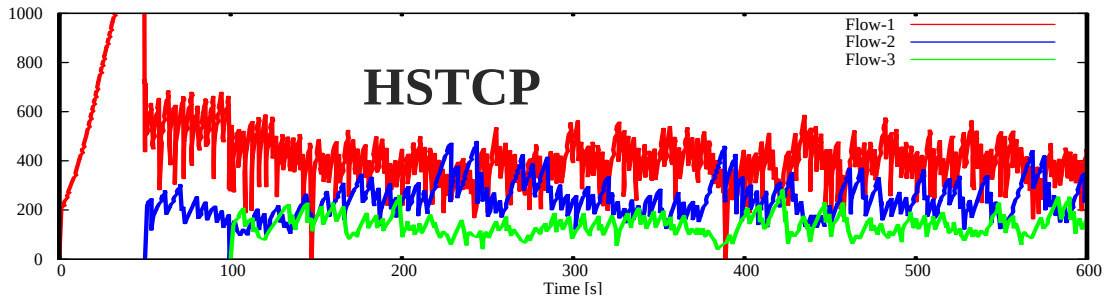
Fairness among identical flows

- Stability and convergence (3flow)
  - Additional RTT : 80ms
  - Honjyo -> Tokyo

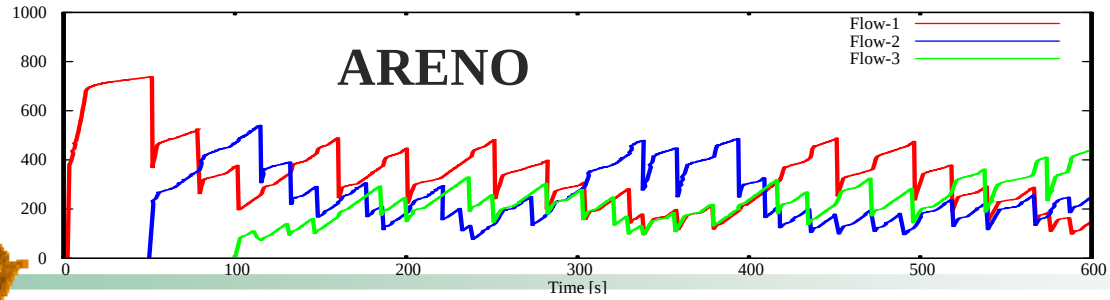




Flow-1 : 36.62Mbps
Flow-2 : 31.15Mbps
Flow-3 : 33.45Mbps



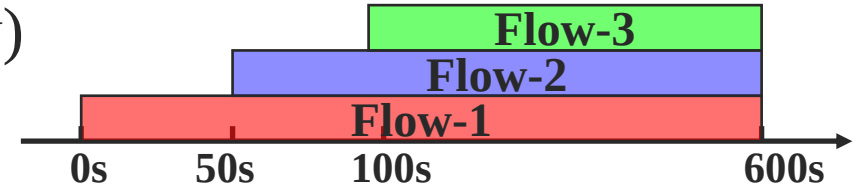
Flow-1 : 51.07Mbps
Flow-2 : 28.69Mbps
Flow-3 : 16.54Mbps

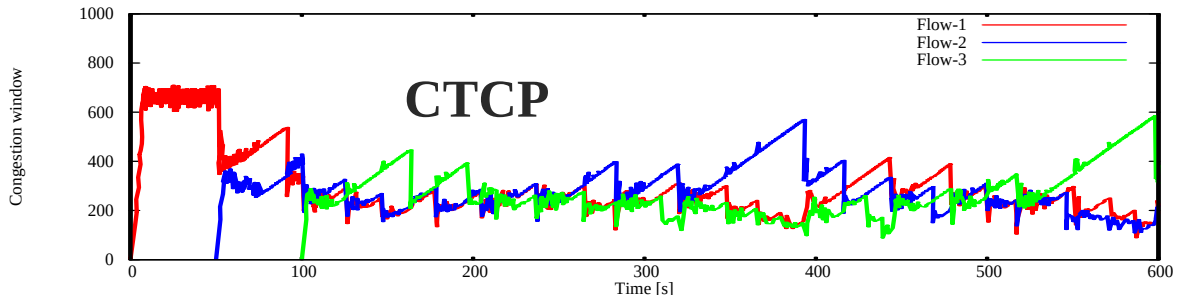


Flow-1 : 40.78Mbps
Flow-2 : 30.96Mbps
Flow-3 : 27.34Mbps

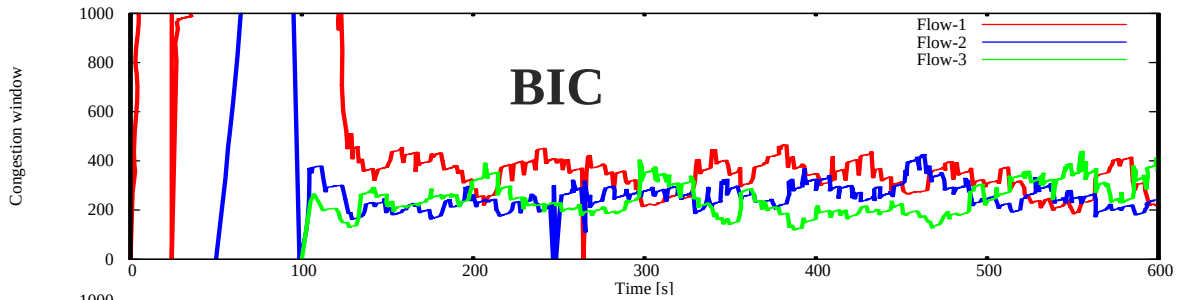
Fairness among identical flows (cont'd)

- Stability and convergence (3flow)
  - Additional RTT : 80ms
  - Honjyo -> Tokyo

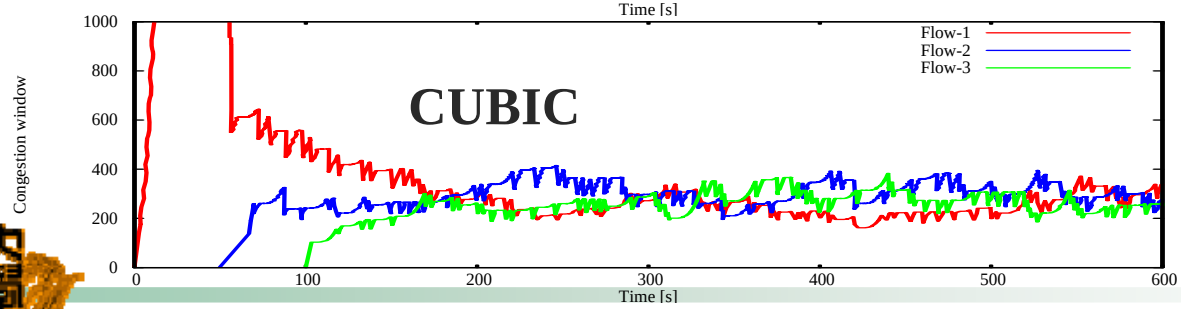




Flow-1 : 35.56Mbps
Flow-2 : 33.32Mbps
Flow-3 : 31.55Mbps



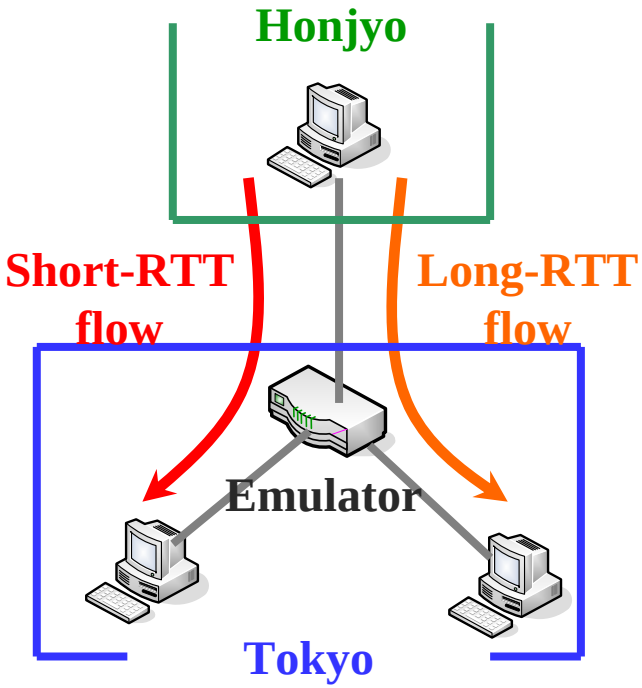
Flow-1 : 45.81Mbps
Flow-2 : 27.09Mbps
Flow-3 : 26.50Mbps

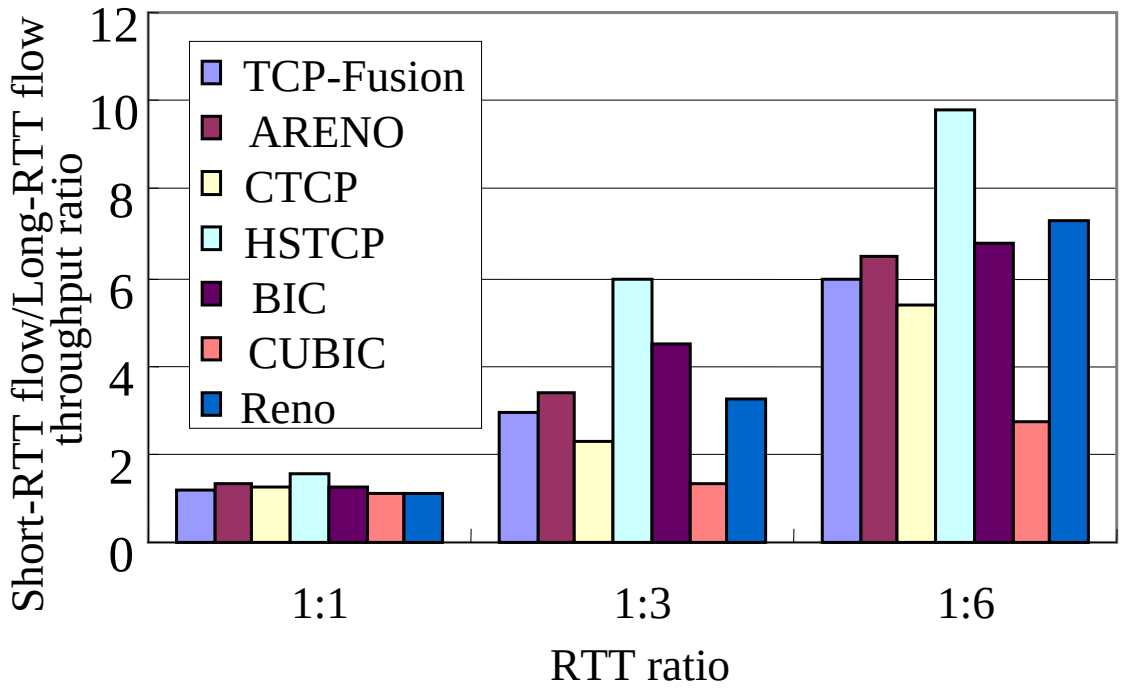


Flow-1 : 38.37Mbps
Flow-2 : 32.86Mbps
Flow-3 : 28.93Mbps

Fairness with different RTT flows

- Throughput ratio of the short RTT flow to the long RTT flow
  - Short-RTT flow : 20ms
  - Long-RTT flow: 20ms, 60ms or 120ms

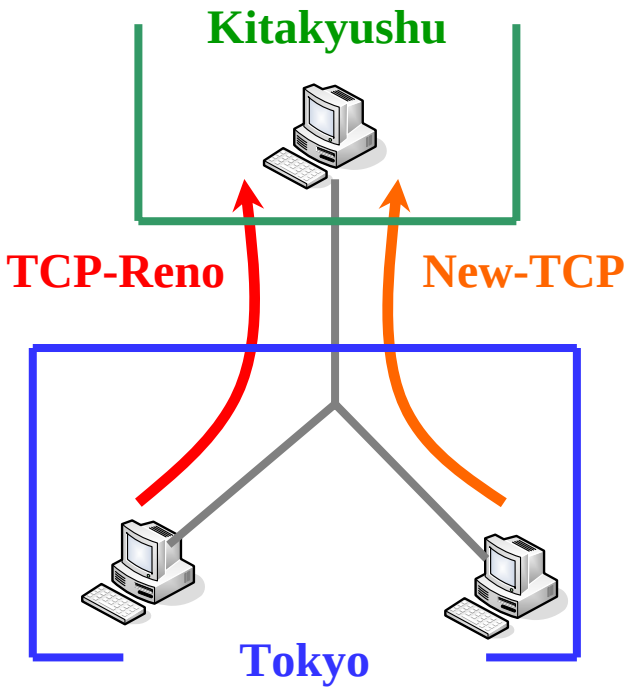


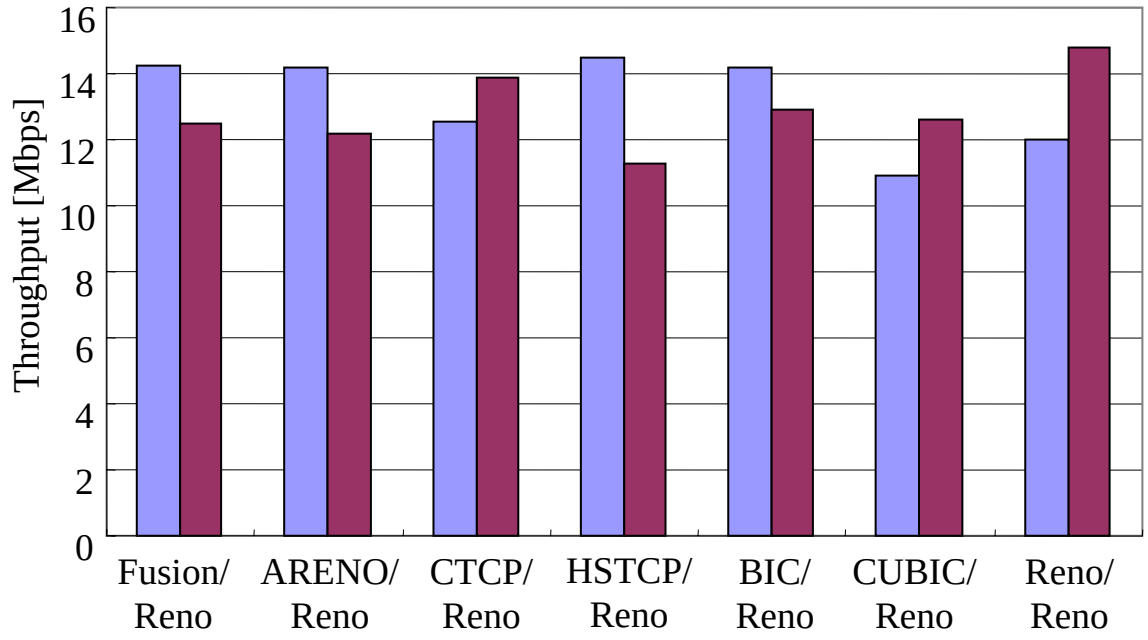


CUBIC performs most fairly <-> HSTCP
TCP-Fusion, ARENO, CTCP and TCP-Reno show the same ratio

Friendliness in small BDP network

- Throughputs of coexisting Reno and TCP variant flows
  - BDP is 50pkts (Bandwidth=30Mbps, RTT=18ms)





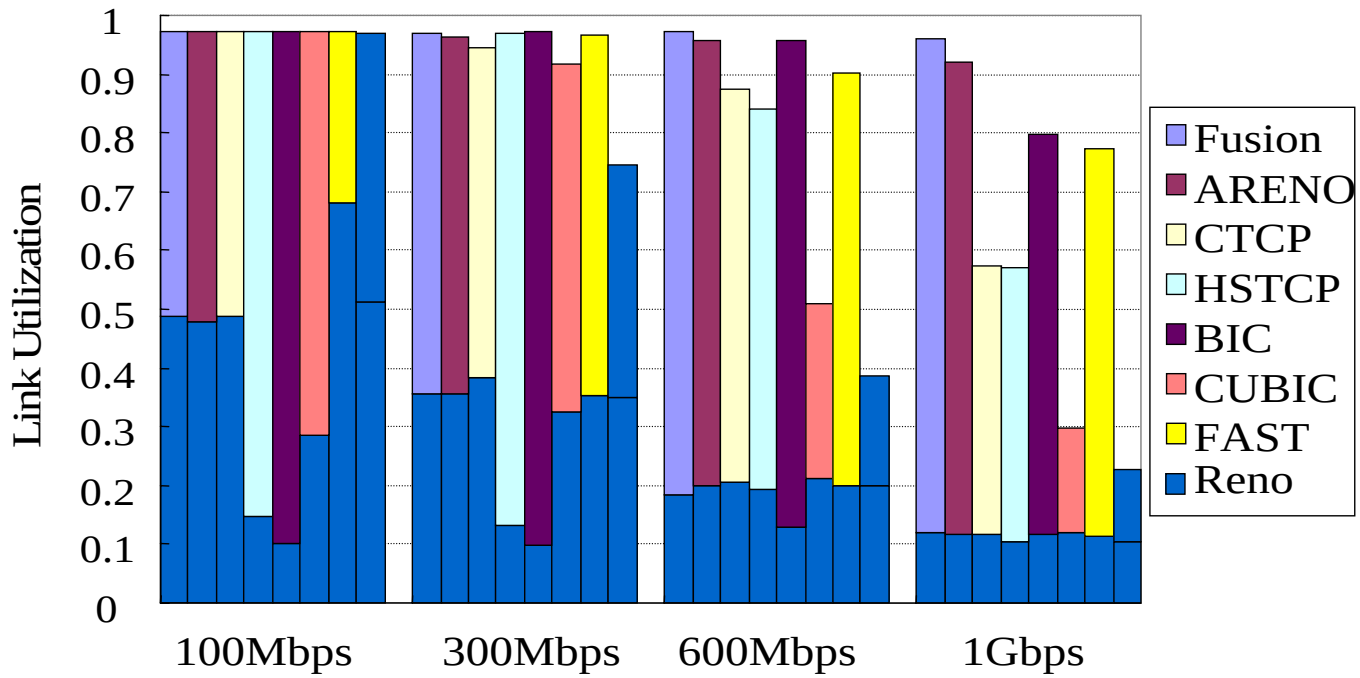
All TCP variants share fairly with TCP-Reno

Simulation Experiments

- Network simulator (ns-2)
- Add FAST to targets of implementation experiments
  - Targets
    - TCP-Reno
    - High-speed TCP (HSTCP)
    - BIC-TCP (BIC)
    - CUBIC
    - TCP-Fusion
    - TCP-Adaptive Reno (ARENO)
    - Compound TCP (CTCP)
    - FAST

Efficiency and Friendliness in high-speed network

- Link utilization ratio of coexisting TCP-Reno and TCP variant flows
  - Simple dumb-bell topology
  - Bandwidth: 100Mbps~1Gbps, loss rate: 10^{-5}, RTT: 44ms, buffer: BDP



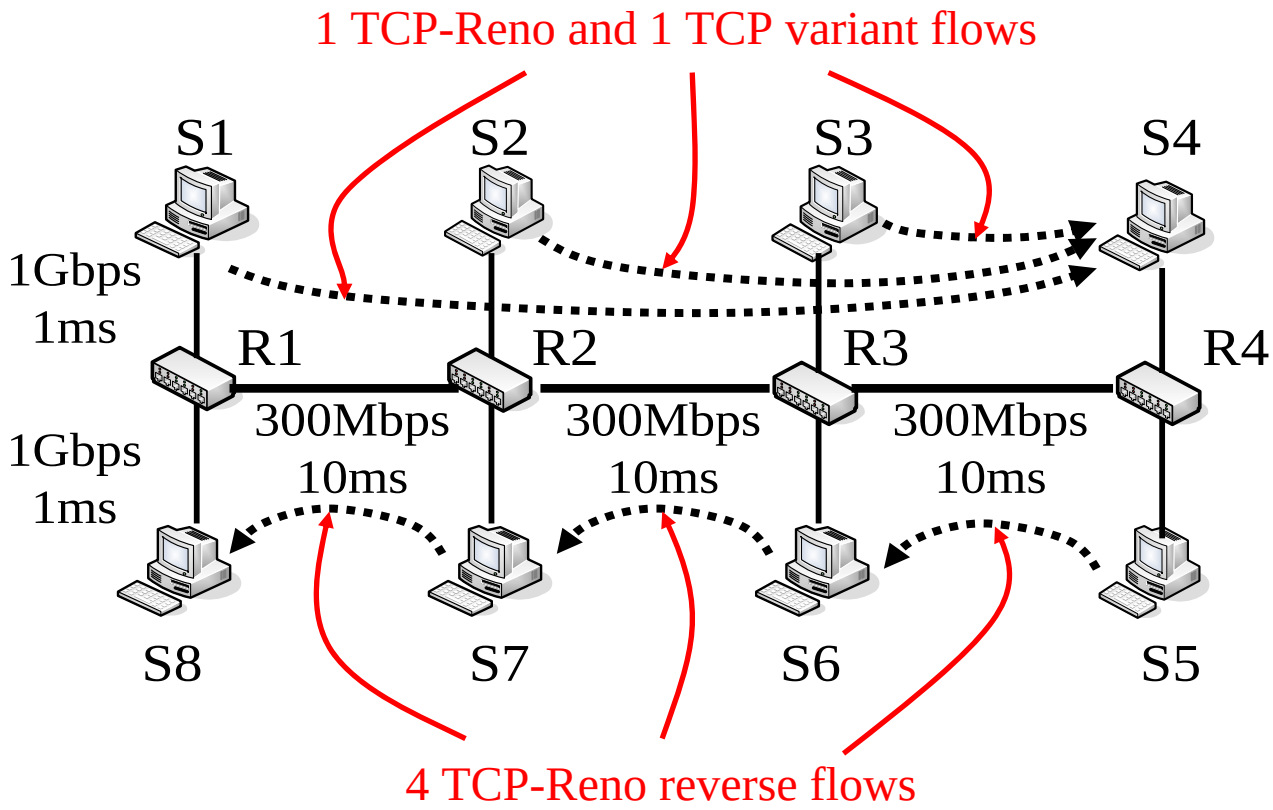
TCP-Fusion is most efficient maintaining the friendliness to TCP-Reno

Conclusions

- TCP-Fusion
- Integration of TCP-Reno, TCP-Vegas and TCPW
- Good balance between high efficiency and friendliness without damaging fairness issues
- Future work
  - More investigation to optimally choose parameters
  - Ongoing implementation experiments in realistic environments

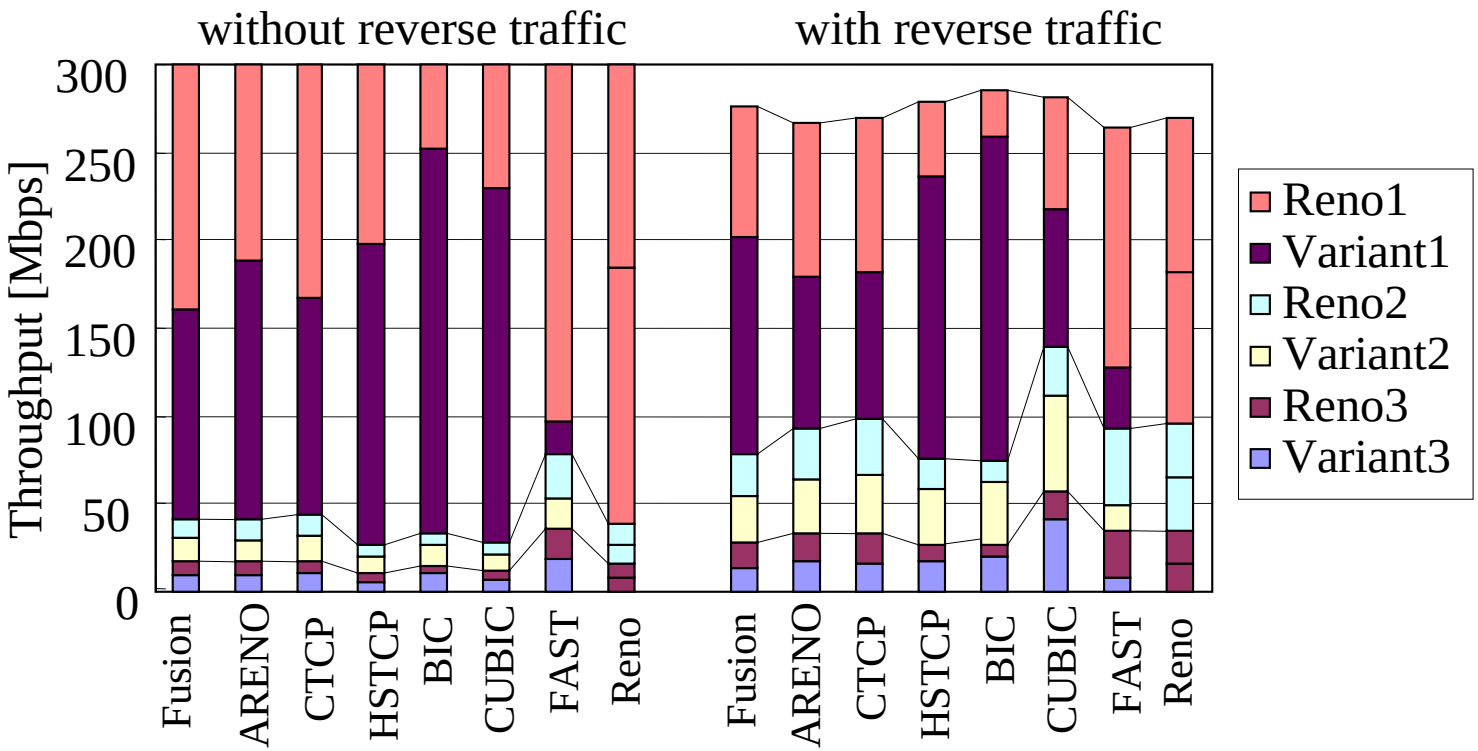
Fairness in complex links

- Parking lot topology
  - 6flows (3 TCP variant and 3 Reno) in forward direction
  - Each 4 TCP-Reno flows in ACK return path



Fairness in complex link (cont'd)

- Throughputs in multi-bottleneck link with/without reverse traffic
  - e.g. Reno1 represents TCP-Reno flow with one hop count



Link characteristics

- The output of *traceroute* (Honjyo -> Tokyo)

traceroute to 133.9.x.x (133.9.x.x), 30 hops max, 40 byte packets
1 133.9.x.x (133.9.x.x) 0.070 ms 0.061 ms 0.064 ms
2 133.9.x.x (133.9.x.x) 0.514 ms 0.497 ms 0.210 ms
3 133.9.x.x (133.9.x.x) 1.664 ms 1.643 ms 1.649 ms
4 133.9.x.x (133.9.x.x) 1.826 ms 1.682 ms 1.681 ms
5 133.9.x.x (133.9.x.x) 1.628 ms 1.616 ms 1.623 ms
6 133.9.x.x (133.9.x.x) 81.931 ms 81.910 ms 81.967 ms ← Additional delay : 80ms

When the absence of TCP traffic

traceroute to 133.9.x.x (133.9.x.x), 30 hops max, 40 byte packets
1 133.9.x.x (133.9.x.x) 0.100 ms 0.066 ms 0.063 ms
2 133.9.x.x (133.9.x.x) 0.348 ms 0.330 ms 0.219 ms
3 133.9.x.x (133.9.x.x) 1.668 ms 1.715 ms 1.841 ms
4 133.9.x.x (133.9.x.x) 1.779 ms 1.768 ms 1.725 ms
5 133.9.x.x (133.9.x.x) 25.882 ms 26.004 ms 26.036 ms ← Congestion point
6 133.9.x.x (133.9.x.x) 106.708 ms 106.528 ms 106.712 ms

When the presence of TCP traffic

At 5th hop, RTT grows (up to about 30ms)
-> buffer capacity is estimated about 375KB

Setting parameters

- Set α according to two requirements;
  - Small value in proportion to the link bandwidth ***B*** to fit in *diff* resolution
  - Inverse proportional value to the number of coexisting TCP-Fusion flow ***N***

•Assumption of minimum buffer size

$$G = \frac{B * D_{\min}}{packet\_size * 8}$$

– Corresponds to the queuing delay D_{min}

– **Proportional value to the link bandwidth**

•Set α

$$\alpha = \frac{G}{N} = \frac{(B/N) * D_{\min}}{packet\_size * 8} \approx \boxed{\frac{RE * D_{\min}}{packet\_size * 8}}$$

– **Inverse proportion to *N***

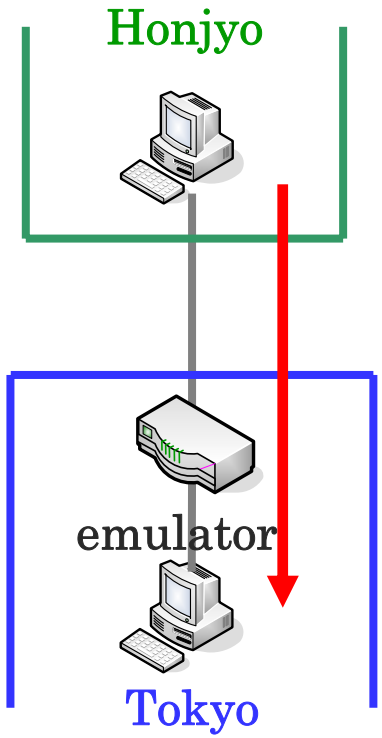
Set W_{inc} to be $W_{inc} \leqq G$

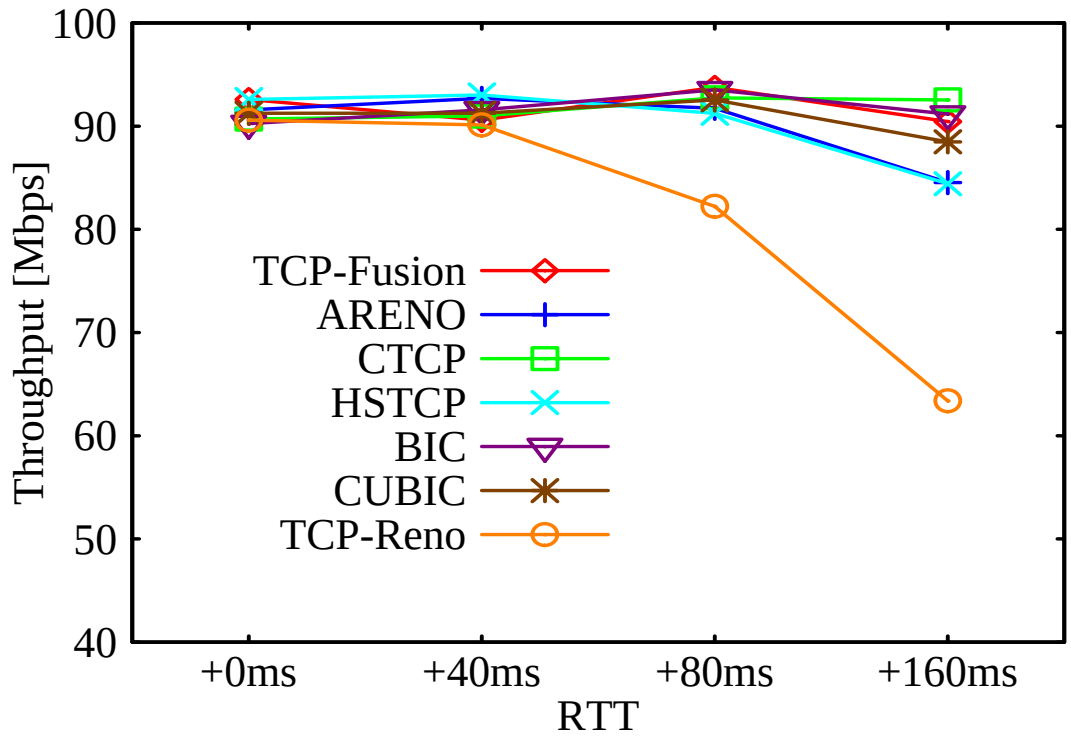
$$W_{inc} = \boxed{B/12Mbps}$$

– Equal to G where D_{min}=1ms is assumed

Efficiency in large BDP network

- A single flow Throughput
  - Additional RTT at emulator : 0ms-160ms

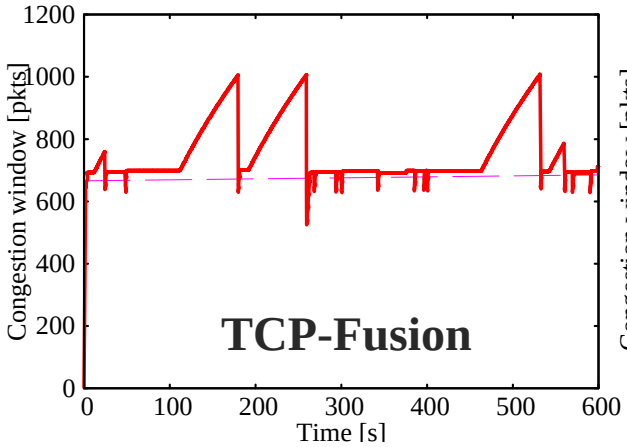


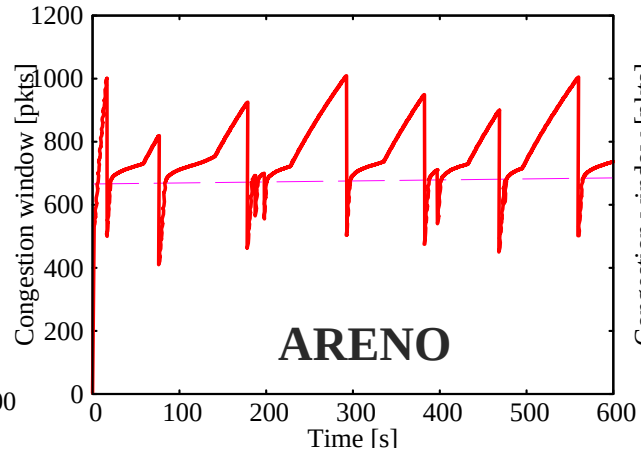


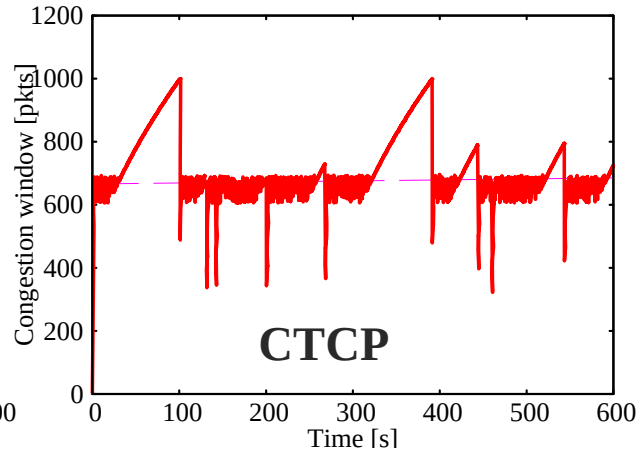
Keep the throughput in long-delay network<->degradation in Reno

Congestion window behavior

- A single TCP-Fusion, ARENO or CTCP flow
  - RTT : 80ms
  - Buffer capacity : BDP/2
  - 0.0001% random packet loss rate (at network emulator)

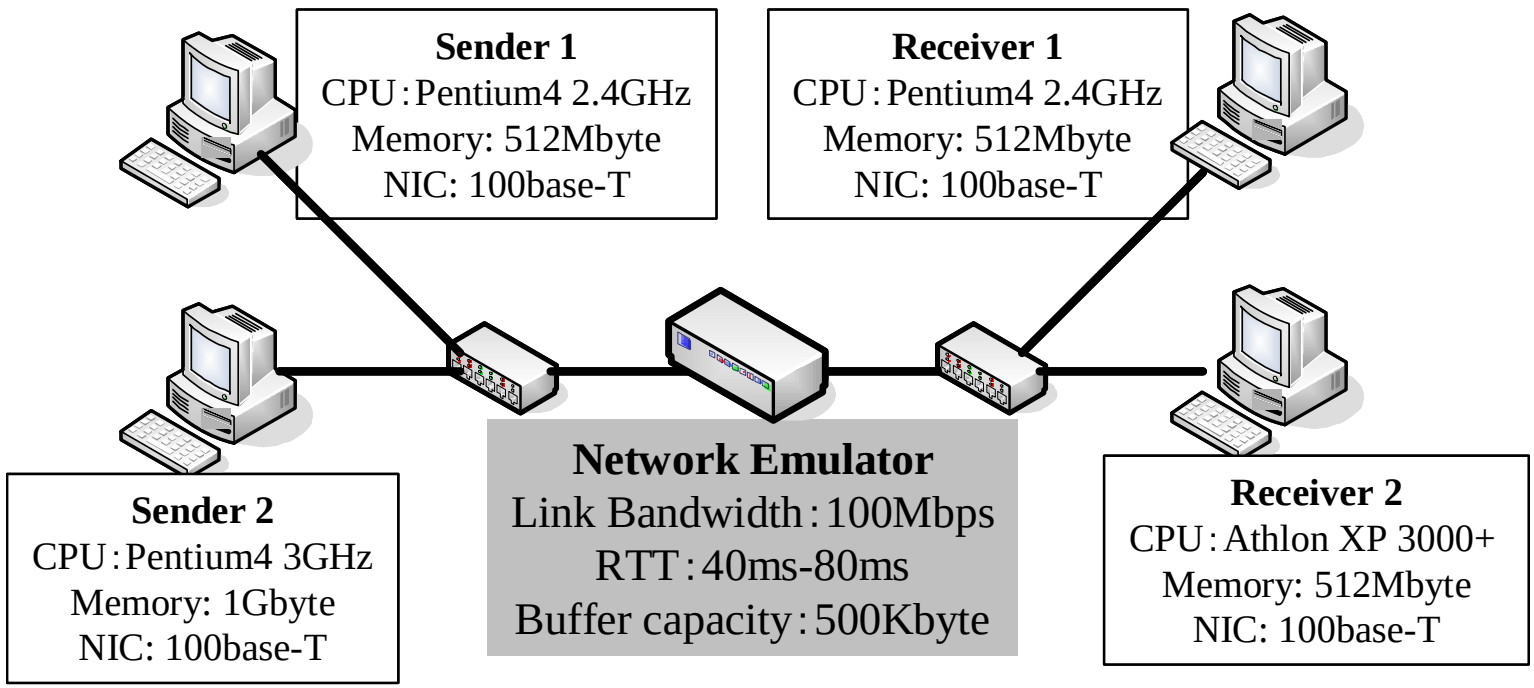






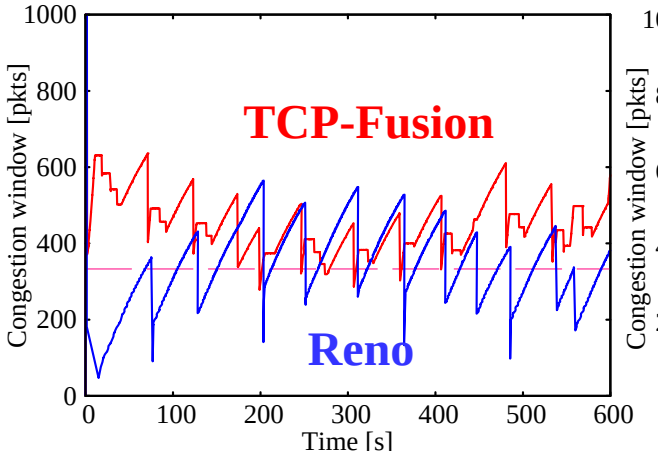
Anticipated behavior that their congestion window size attempt to keep the BDP

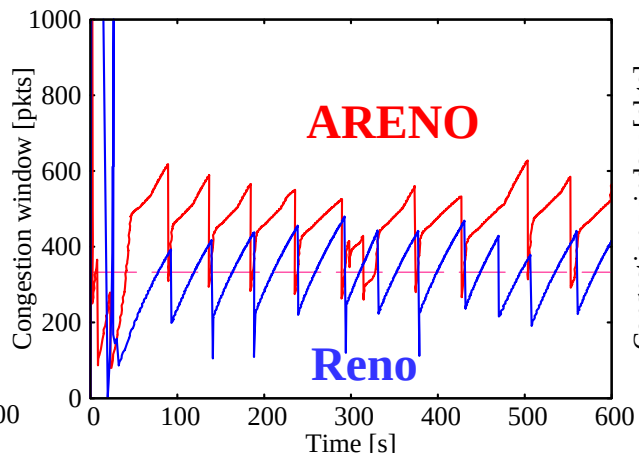
Network model in our laboratory

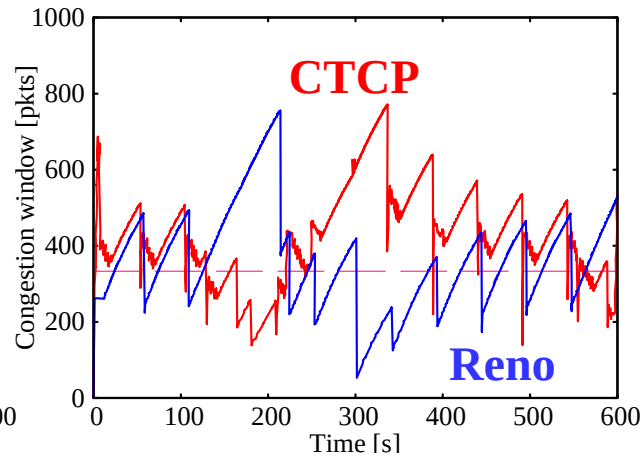


Congestion window behavior (cont'd)

- Competing TCP-Reno and TCP-Fusion/ARENO/CTCP
  - RTT : 80ms
  - Buffer capacity : BDP/2
  - 0.0001% random packet loss rate (at network emulator)







High-friendliness to TCP-Reno