<Slides download>
https://www.pf.is.s.u-tokyo.ac.jp/classes

Advanced Operating Systems

#10

Hiroyuki Chishiro
Project Lecturer

Department of Computer Science
Graduate School of Information Science and Technology
The University of Tokyo

Course Plan

- Multi-core Resource Management
- Many-core Resource Management
- GPU Resource Management
- Virtual Machines
- Distributed File Systems
- High-performance Networking
- Memory Management
- Network on a Chip
- Embedded Real-time OS
- Device Drivers
- Linux Kernel

Schedule

1. 2018.9.28 Introduction + Linux Kernel (Kato)
2. 2018.10.5 Linux Kernel (Chishiro)
3. 2018.10.12 Linux Kernel (Kato)
4. 2018.10.19 Linux Kernel (Kato)
5. 2018.10.26 Linux Kernel (Kato)
6. 2018.11.2 Advanced Research (Chishiro)
7. 2018.11.9 Advanced Research (Chishiro)
8. 2018.11.16 (No Class)
9. 2018.11.23 (Holiday)
10. 2018.11.30 Advanced Research (Chishiro)
11. 2018.12.7 Advanced Research (Kato)
12. 2018.12.14 Advanced Research (Kato)
13. 2018.12.21 Linux Kernel
14. 2019.1.11 (No Class)
15. 2019.1.18 10:25-12:10 Linux Kernel
16. 2019.1.25 (No Class)

High-Performance Networking

Introducing TCP Congestion Control Algorithms

/* The Cases for BIC, CUBIC, and BBR */

Acknowledgement

Prof. Injong Rhee, “Congestion Control on High-Speed Networks”, NCSU

Prof. Injong Rhee, “CUBIC : A New TCP-Friendly High-Speed TCP Variant”, NCSU

Mr. Neal Cardwell et al., “BBR Congestion Control”, Google

Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

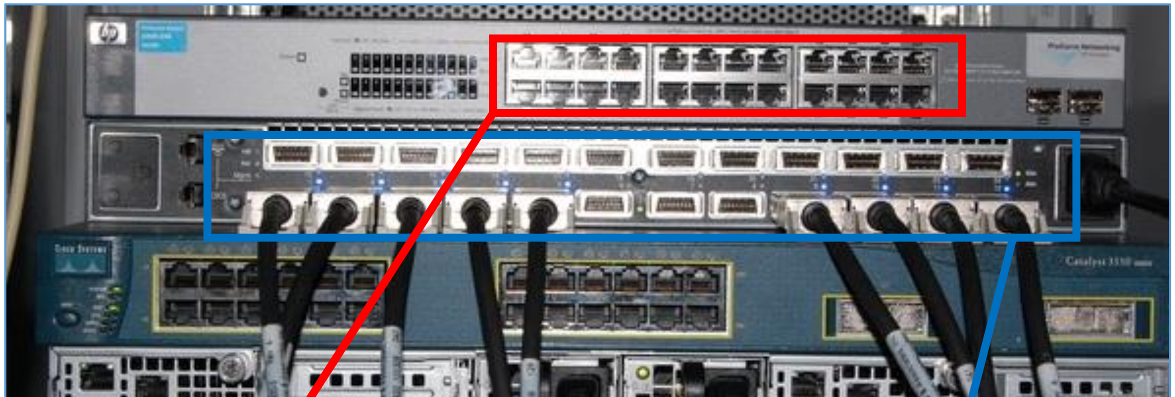
Why High-Performance Networking?

- High-Performance Computing (HPC) Servers
  - Data Centers
  - Supercomputers (E.g.: T2K, TSUBAME)
- HPC applications are:
  - Weather Forecast, Video Conference, Gene Analysis, Search Engine, Artificial Intelligence

T2K: https://www.ccs.tsukuba.ac.jp/

TSUBAME: http://www.gsic.titech.ac.jp/en/tsubame

Representative Network Interfaces for High-Performance Networking



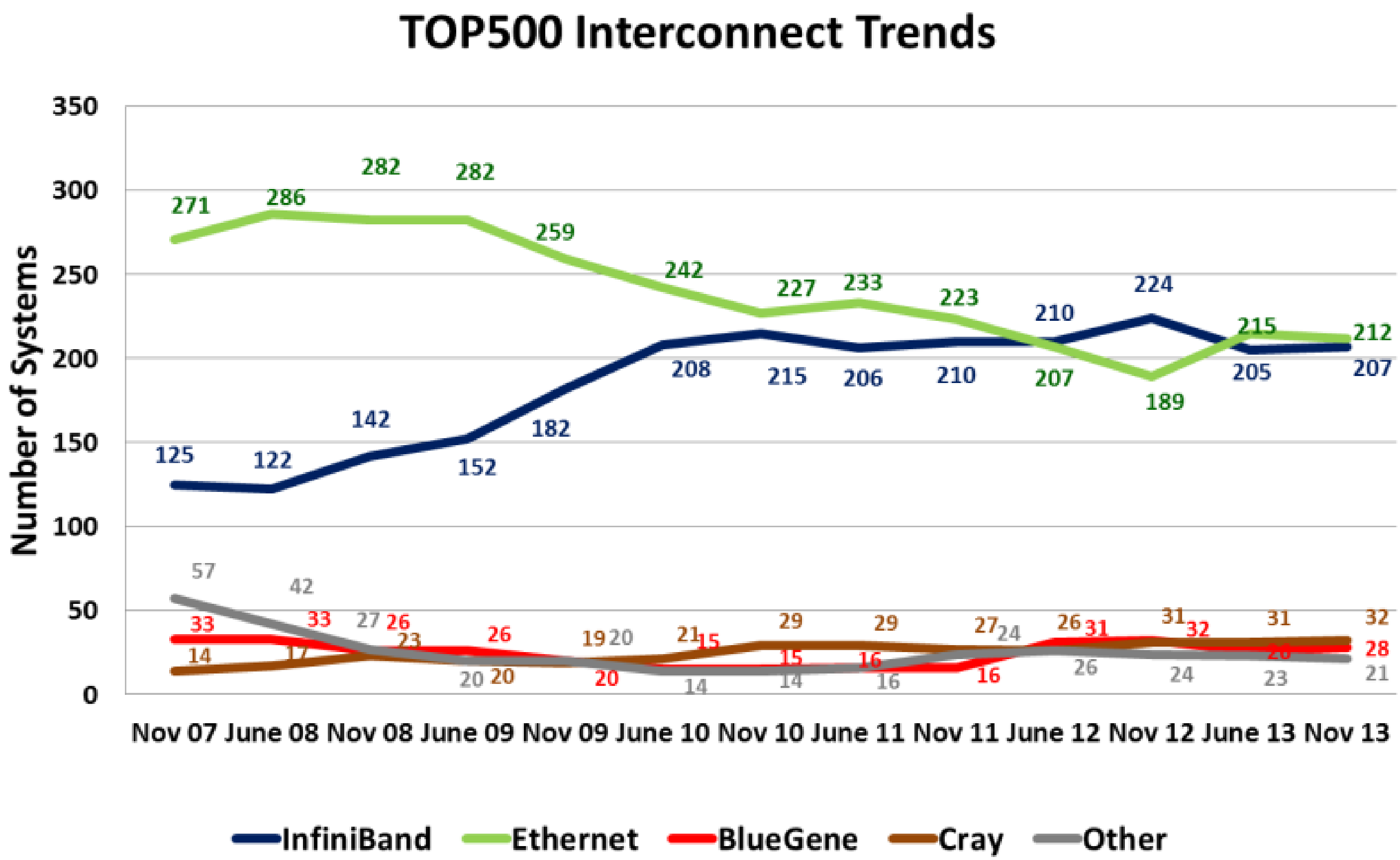
Network Switch

Ethernet

InfiniBand

Network Switch Image From http://edu.escience-lab.org/index.php/EthiopiaCluster/Cable
Ethernet Image from https://ja.wikipedia.org/wiki/%E3%82%A4%E3%83%BC%E3%82%B5%E3%83%8D%E3%83%83%E3%83%88
InfiniBand Image from http://www.lucidti.com/wp-content/uploads/2012/10/Infiniband-4X-Pull-Latch-R.png

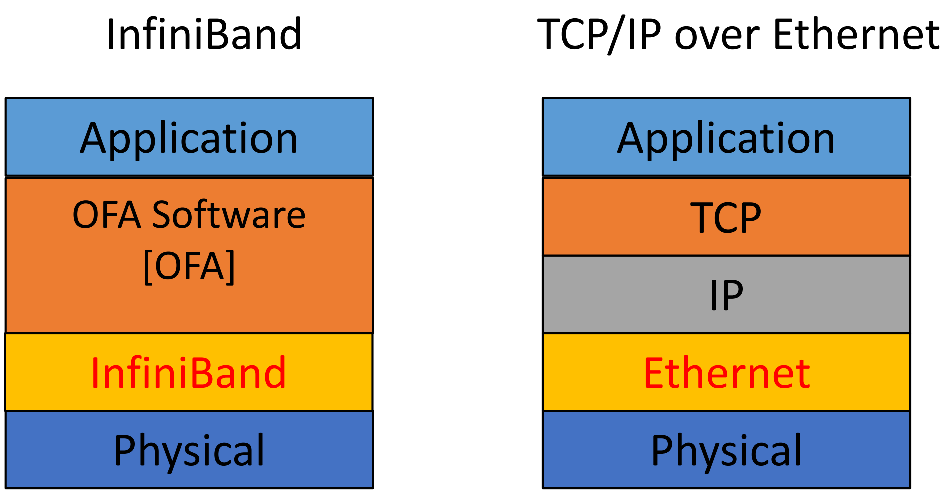
Network Usage in Supercomputers



TSUBAME and T2K use InfiniBand.
Some private companies use Ethernet.
Almost all operating systems in Supercomputers are Linux.

Image from http://www.gigalight.com/solutions/&i=48&comContentId=48.html

Protocol Stack in InfiniBand and Ethernet



InfiniBand does not use TCP/IP but uses low-level connection by OFA software.

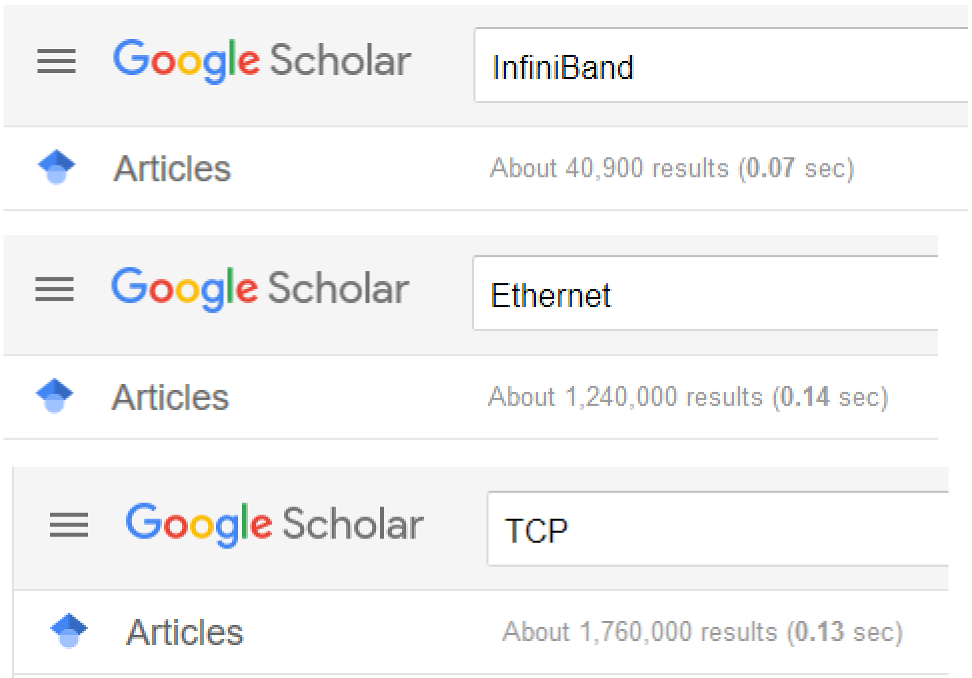
[OFA] OpenFabrics Alliance (OFA). https://openfabrics.org/

InfiniBand and Ethernet in Linux Kernel

- InfiniBand
  - drivers/infiniband
- Ethernet
  - drivers/net/ethernet

- TCP/IP
  - net/ipv4
  - net/ipv6

Search by Google Scholar (2018/11/30)



Ethernet is majority in research and industry.
This class focuses on TCP.

Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

Control of TCP

- Retransmission Control
  - resends a packet if it is lost.
  - NOTE: TCP is connection-oriented protocol.
- Flow Control
  - manages the rate of data transmission between two nodes.
  - prevents the sender from overwhelming the receiver.
- Congestion Control
  - avoids the collapse of packets.
  - is a hot topic in research.

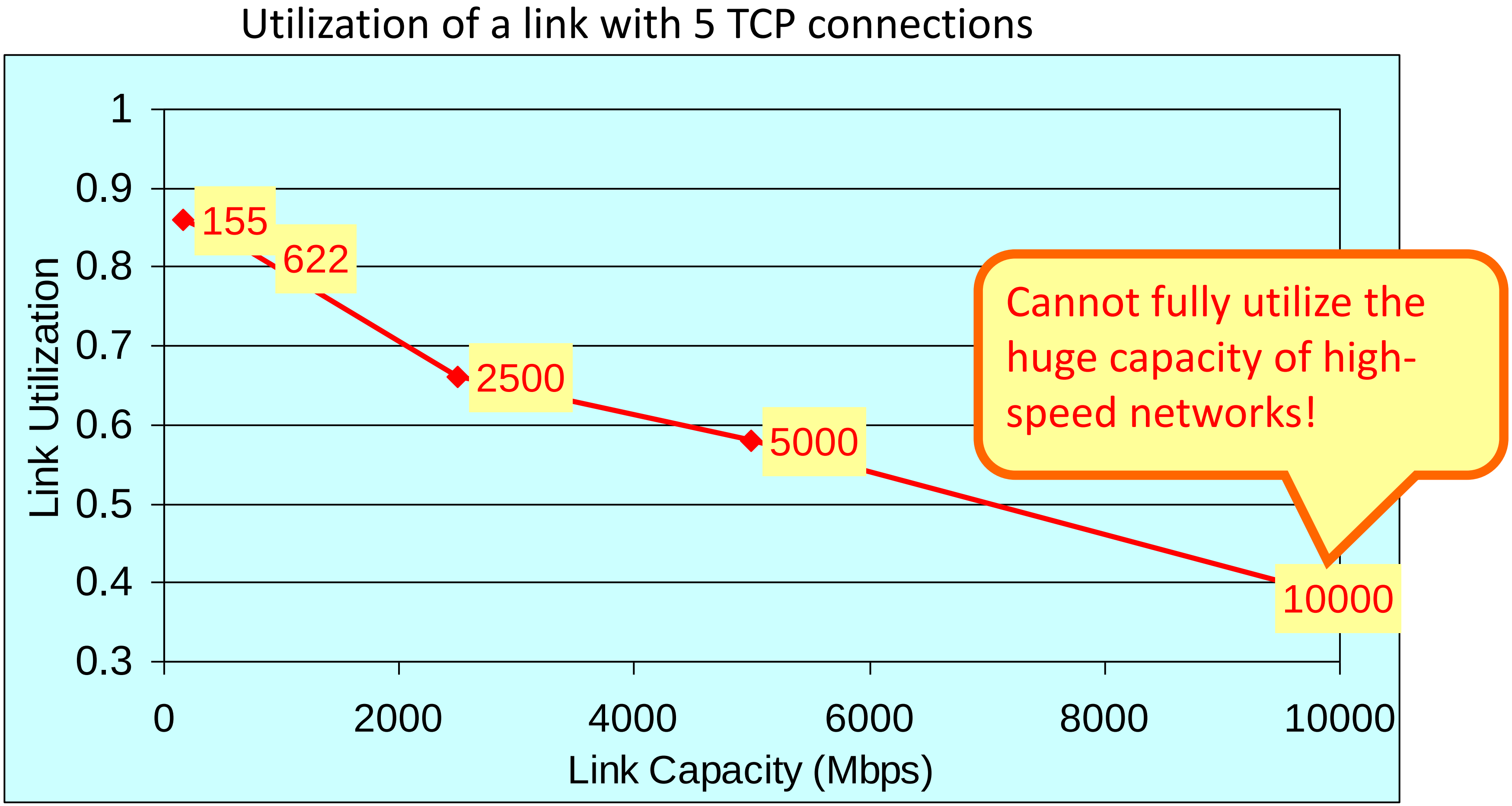
Why Control of TCP?

- Limitation of Network Capacity
  - Queueing Delay
  - Packet Loss
- These issues result in decreasing throughput.

Control Parameters

- Round Trip Time (RTT)
  - = S + R
  - S: The length of time for a packet to be sent
  - R: The length of time for an acknowledgment (ACK) of the packet to be received
- Congestion Window Size (cwnd)
  - is size to be sent.
  - is maintained by sender.
- TCP Window Size
  - is size to be received.
  - is maintained by receiver.

TCP Performance

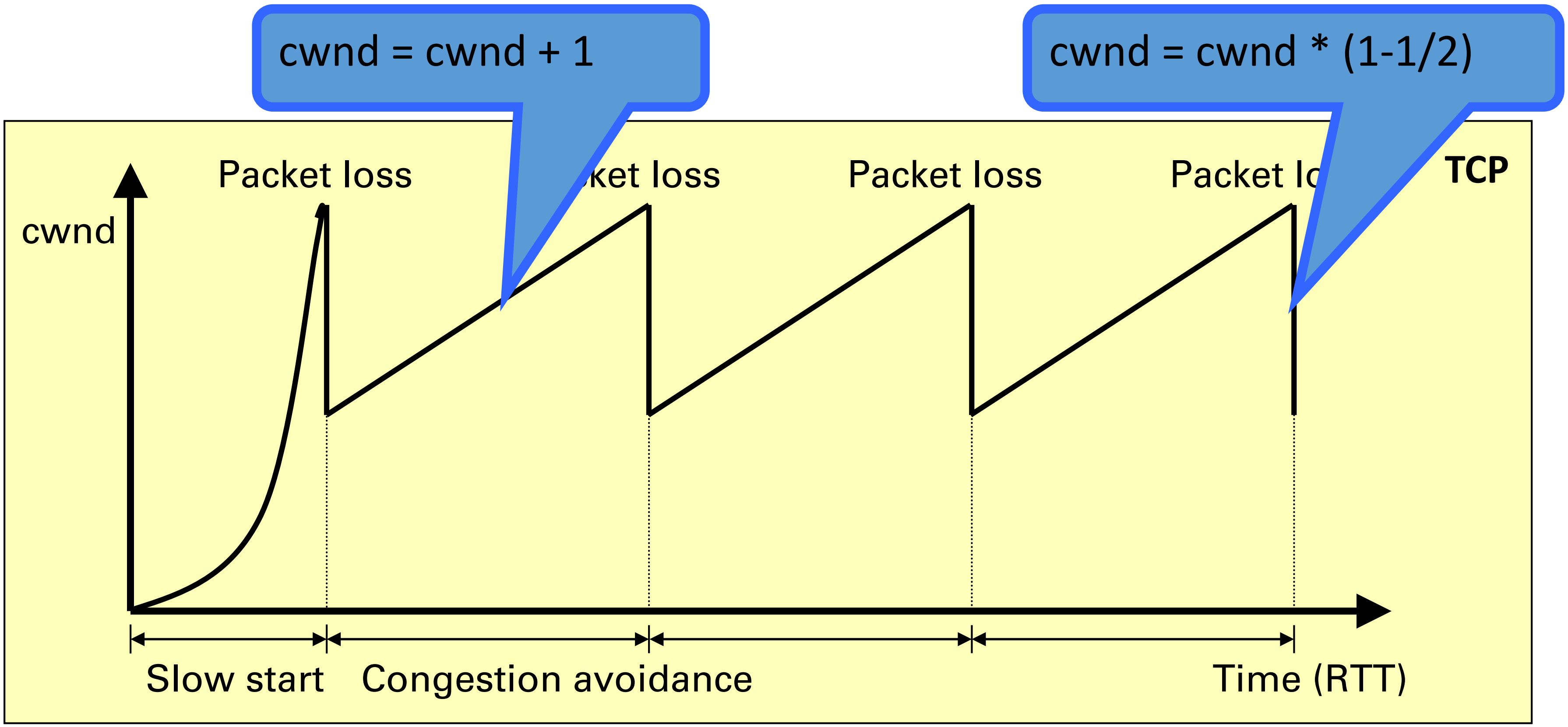


NS-2 Simulation (100 sec)

- Link Capacity = 155Mbps, 622Mbps, 2.5Gbps, 5Gbps, 10Gbps,
- Drop-Tail Routers, 0.1 Bandwidth Delay Product (BDP) Buffer
- 5 TCP Connections, 100ms RTT, 1000-Byte Packet Size

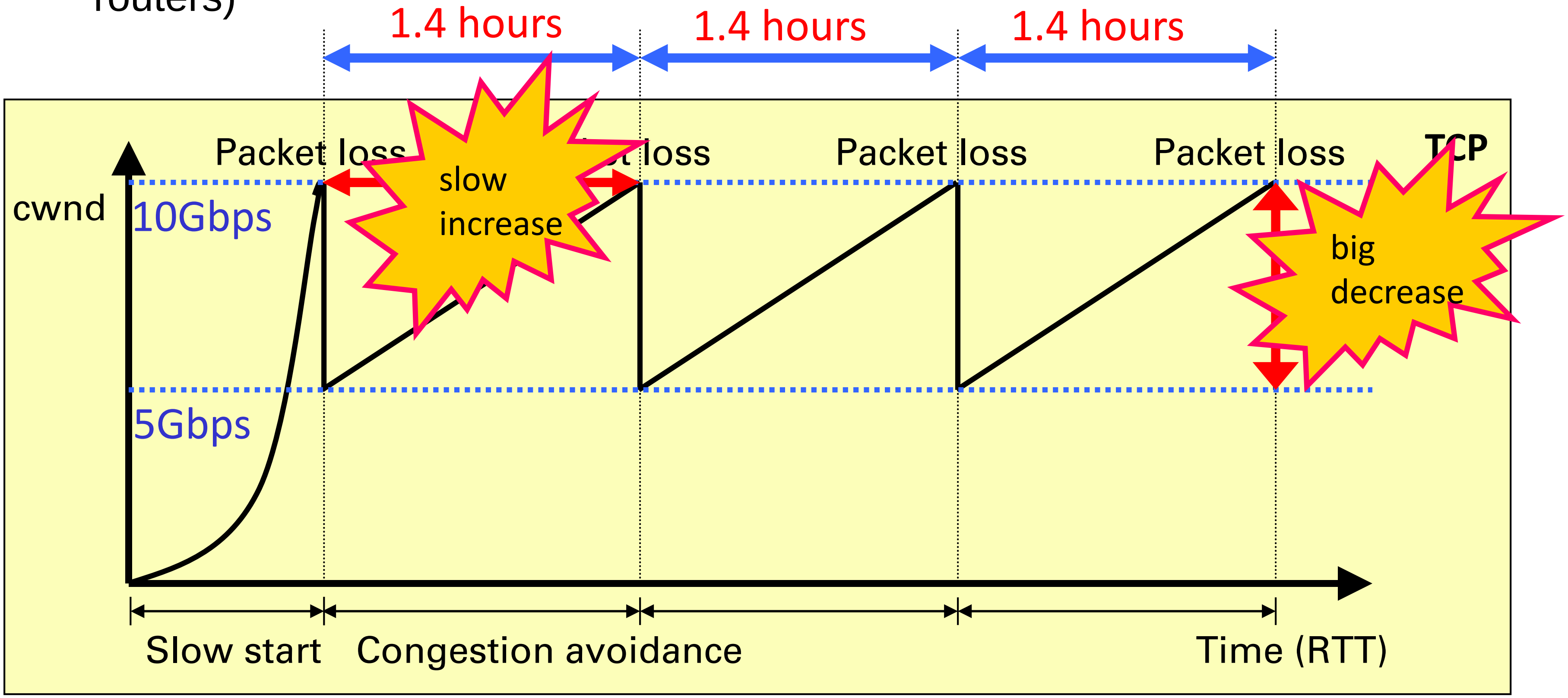
TCP Congestion Control

- The instantaneous throughput of TCP is controlled by a variable *cwnd*,
- TCP transmits approximately a *cwnd* number of packets per RTT.



TCP over High-Speed Networks

- A TCP connection with 1250-Byte packet size and 100ms RTT is running over a 10Gbps link (assuming no other connections, and no buffers at routers)



Response Function of TCP [Padhye 98]

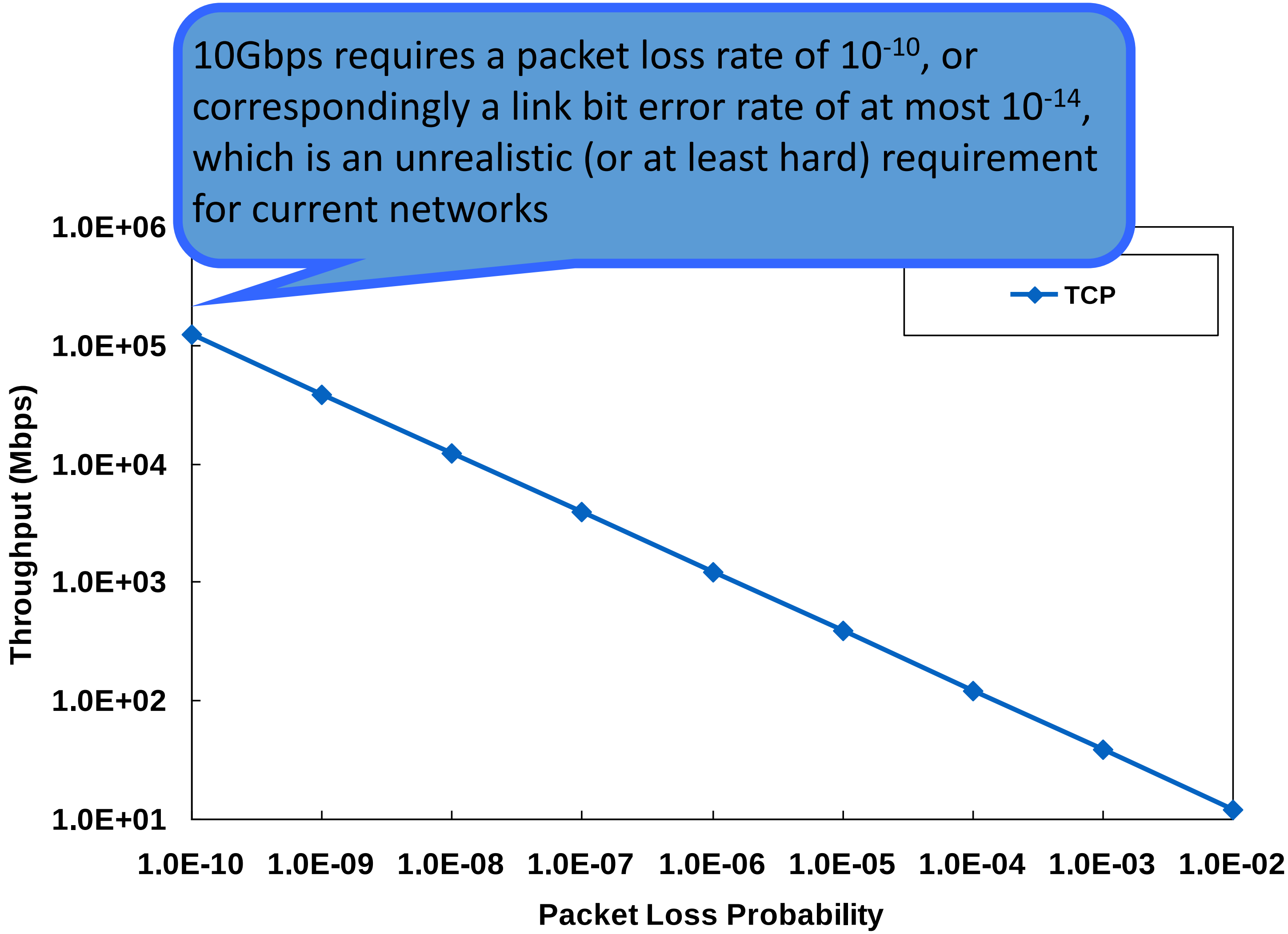
- Response function of TCP is the average throughput of a TCP connection in terms of the packet loss probability, the packet size, and the round-trip time.
- Response function of TCP is:

$$R = \frac{MSS}{RTT}\frac{1.2}{p^{0.5}}$$

  - R: Average Throughput
  - MSS: Packet Size
  - RTT: Round-Trip Time
  - P: Packet Loss Probability

[Padhye 98] Jitendra Padhye, Victor Firoiu, Don Towsley, and Jim Kurose Modeling TCP Throughput: a Simple Model and its Empirical Validation. Proceedings of SIGCOMM 98, pp. 303-314, August 1998.

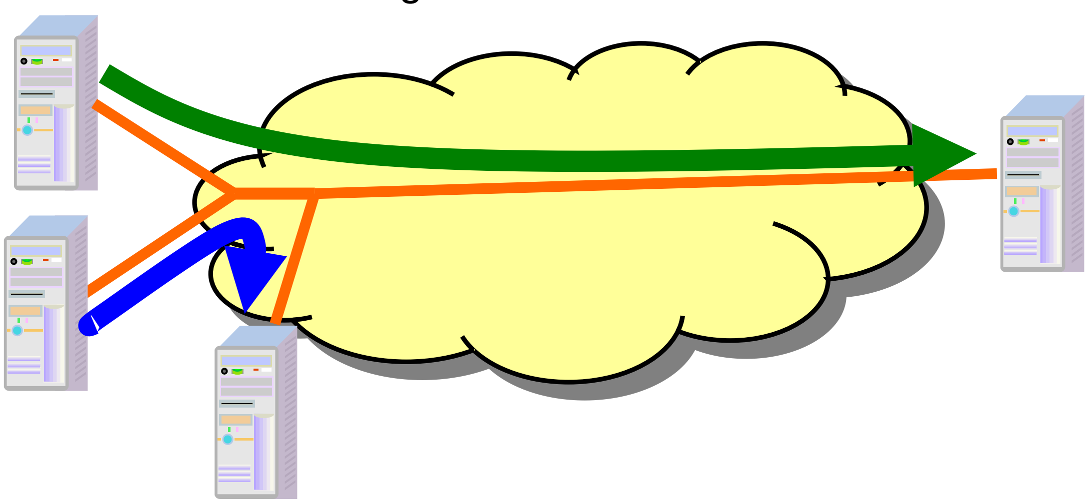
Response Function of TCP



Assuming 1250-Byte packet size, and 100ms RTT

Requirements for TCP Congestion Control

- Bandwidth Scalability
  - The ability to achieve 10Gbps with a reasonable packet loss probability
- TCP Friendliness
  - The ability to share bandwidth with TCP connections on low-speed networks
- RTT Fairness
  - Different connections may have quite different round-trip times, and a good protocol should allocate bandwidth fairly among those connections
  - RTT fairness index = throughout ratio of two flows with different RTTs



RTT Fairness on Low-Speed Networks

- For a protocol with the following response function, where *c* and *d* are protocol-related constants.

$$R = \frac{MSS}{RTT}\frac{c}{p^{d}}$$

- The RTT Fairness Index (or the throughput ratio of two flows) on low-speed networks is

$$\left(\frac{RTT_2}{RTT_1}\right)$$

- On low-speed networks, different protocols have the same RTT fairness.

RTT Fairness on High-Speed Networks

- For a protocol with the following response function, where *c* and *d* are protocol-related constants.

$$R = \frac{MSS}{RTT}\frac{c}{p^d}$$

- The RTT Fairness Index (or the throughput ratio of two flows) on high-speed networks is

$$\left(\frac{RTT_2}{RTT_1}\right)^{\frac{1}{1-d}}$$

- On high-speed networks, the RTT fairness of a protocol depends on the exponent *d* in the response function.

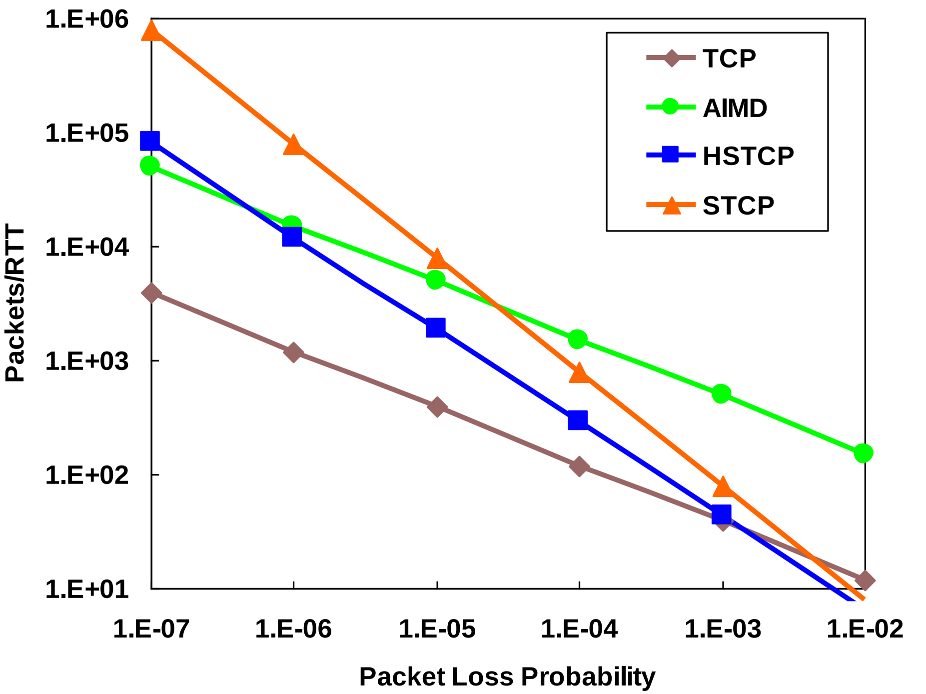
Slope Determines the RTT Fairness

- If the response function is: then the RTT Fairness is:

$$R(p) = \frac{MSS}{RTT}\frac{c}{p^d}$$

$$\left(\frac{RTT_2}{RTT_1}\right)^{\frac{1}{1-d}}$$

- The figure is shown in log-log scale, so exponent *d* in the response function is the slope of the line in the figure.
- The slope of the line determines the RTT fairness of a protocol on high-speed networks.



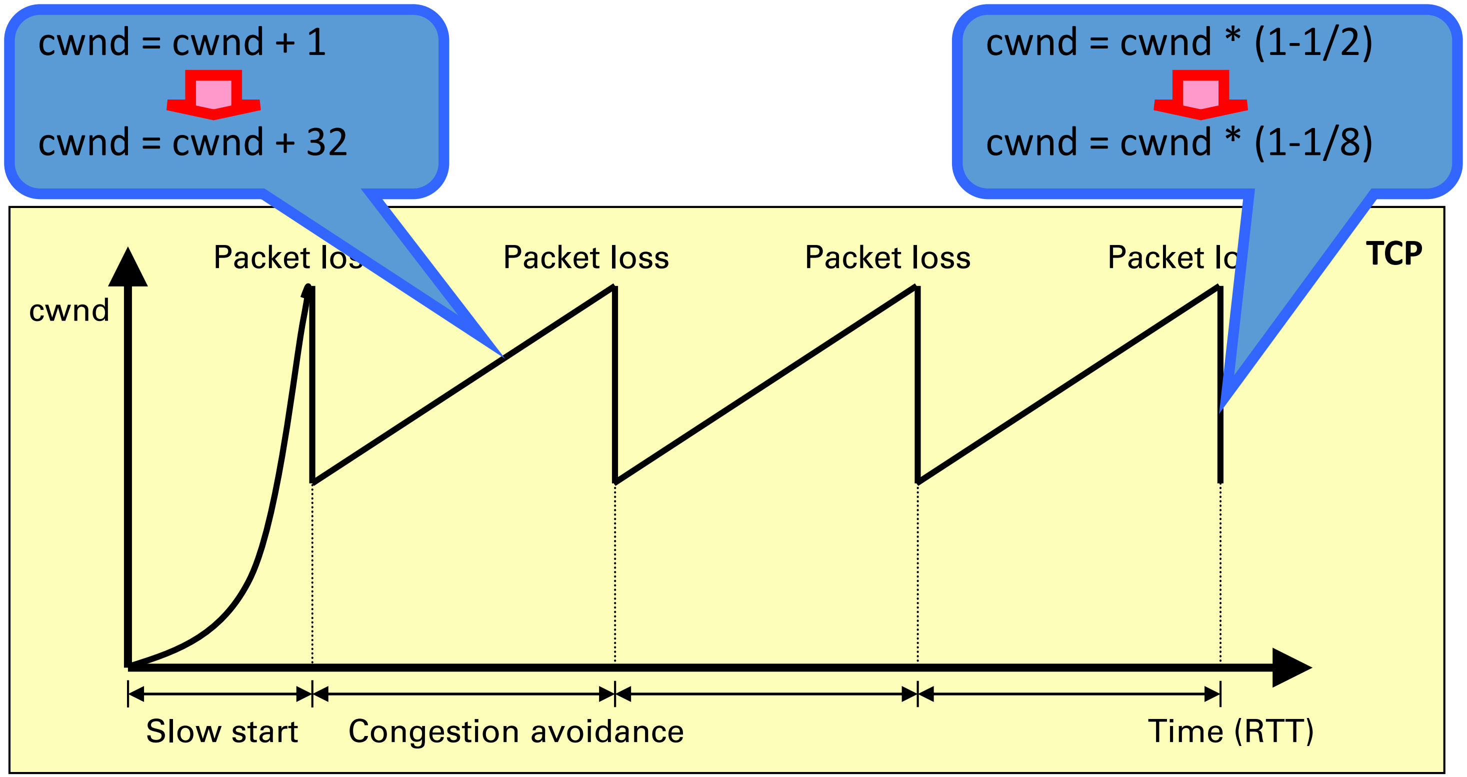
Congestion Control Algorithms

- Additive Increase Multiplicative Decrease (AIMD) [Chiu 89]
- Scalable TCP (STCP) [Kekky 03]
- High Speed TCP (HSTCP) [Floyd 03]

[Chiu 89] Dah-Ming Chiu and Raj Jain. Analysis of Increase and Decrease Algorithms for Congestion Avoidance in Computer Networks. Computer Networks and ISDN Systems, Vol. 17, No. 1, pp. 1-14, June 1989.
[Kekky 03] Tom Kelly. Scalable TCP: Improving Performance in Highspeed Wide Area Networks. ACM SIGCOMM Computer Communication Review, Vol. 33, No. 2, pp. 83-91, April 2003.
[Floyd 03] Sally Floyd. HighSpeed TCP for Large Congestion Windows. https://buildbot.tools.ietf.org/html/rfc3649

Additive Increase Multiplicative Decrease (AIMD)

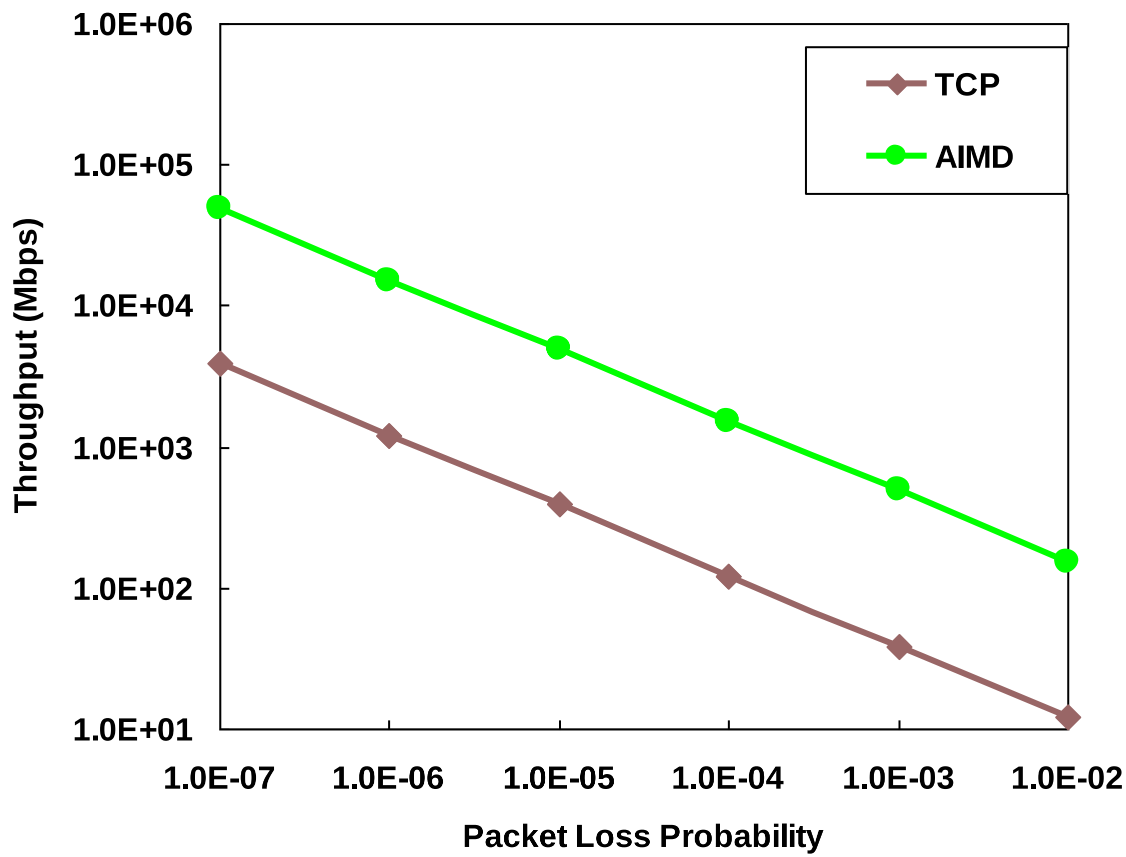
- AIMD increases *cwnd* by a larger number by 32, instead of 1 per RTT.
- After a packet loss, AIMD decreases *cwnd* by 1/8, instead of 1/2



Response Function of AIMD

- TCP: $R = \frac{MSS}{RTT}\frac{1.2}{p^{0.5}}$
- AIMD: $R = \frac{MSS}{RTT}\frac{15.5}{p^{0.5}}$

The throughput of AIMD is always about 13 times larger than that of TCP



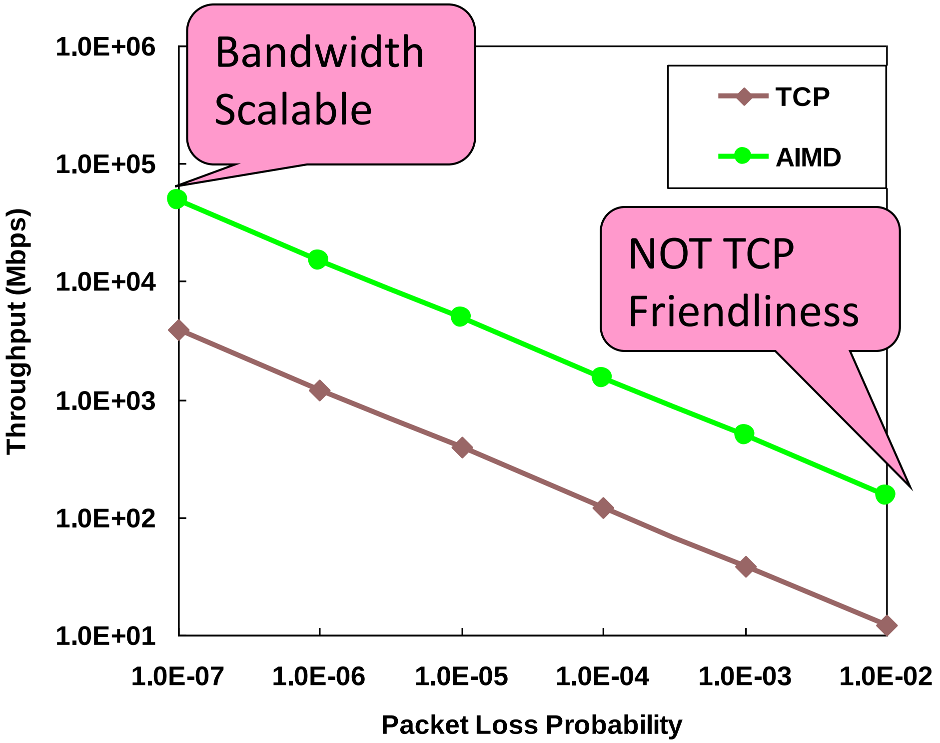
Properties of AIMD

- Bandwidth Scalability

  The ability to achieve 10Gbps with a reasonable packet loss probability

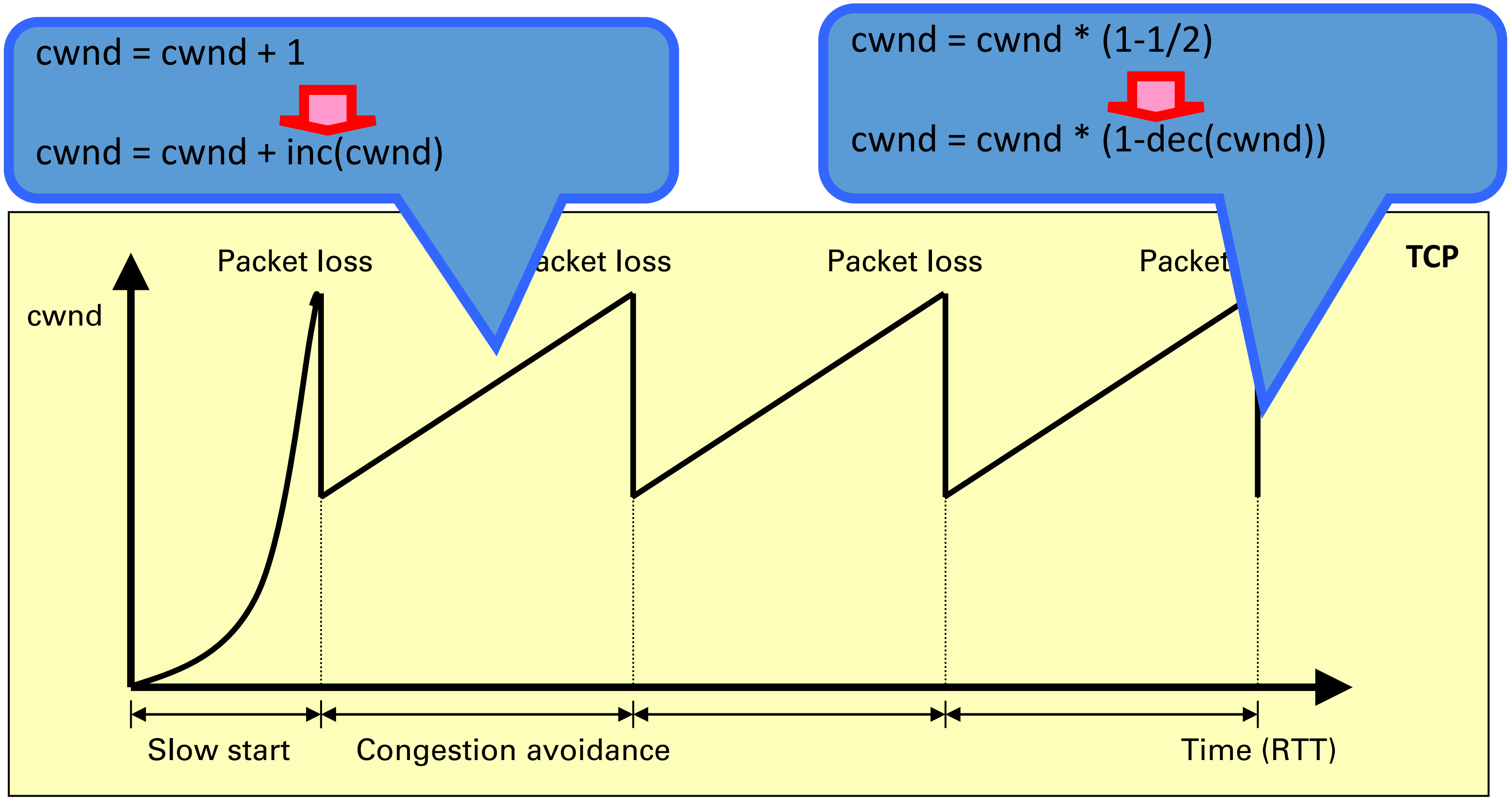
- TCP Friendliness

  The ability to share bandwidth with TCP connections on low-speed networks



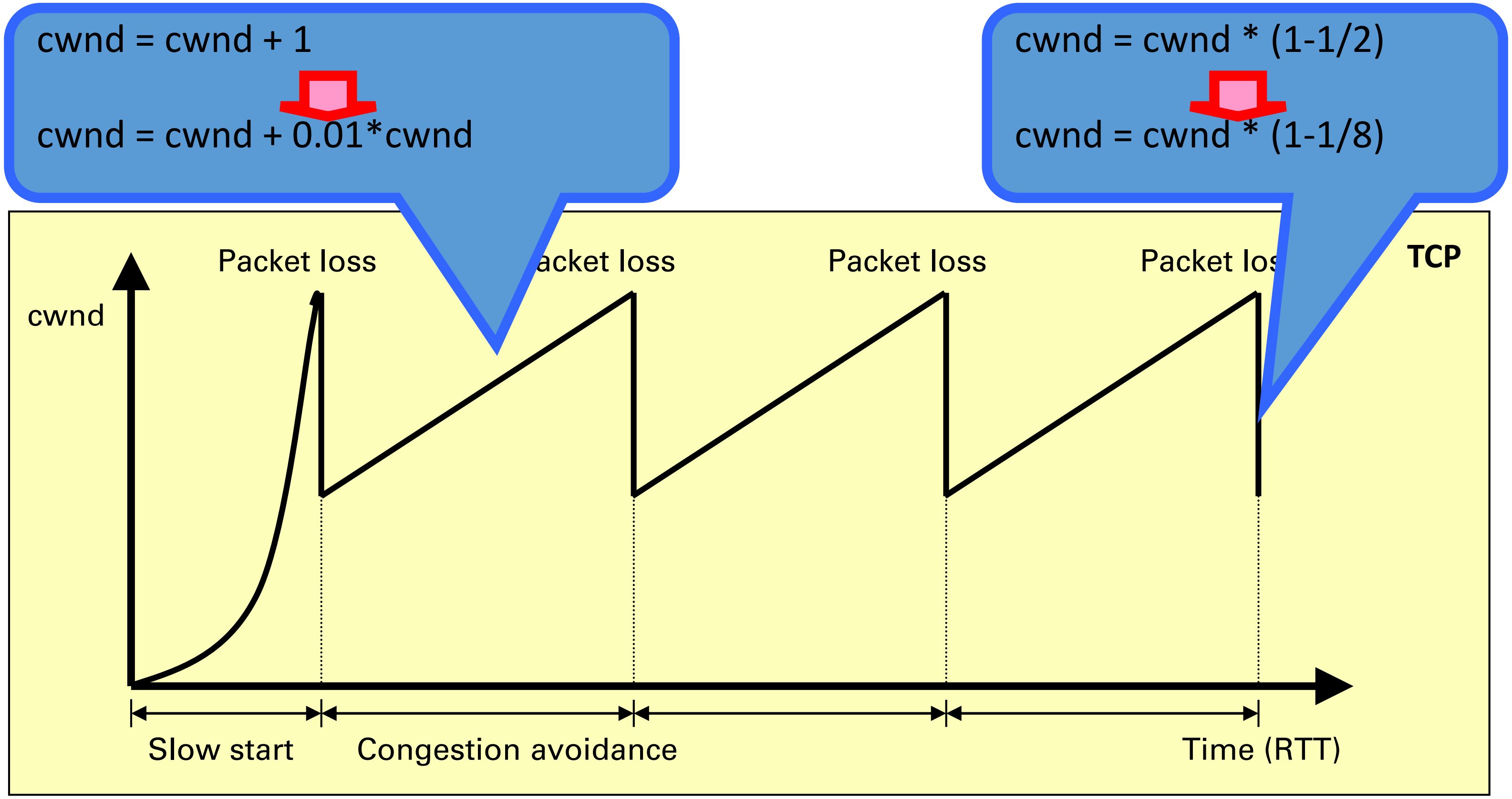
High Speed TCP (HSTCP)

- HSTCP adaptively increases *cwnd*, and adaptively decreases *cwnd*.
- The larger the *cwnd*, the larger the increment, and the smaller the decrement.



Scalable TCP (STCP)

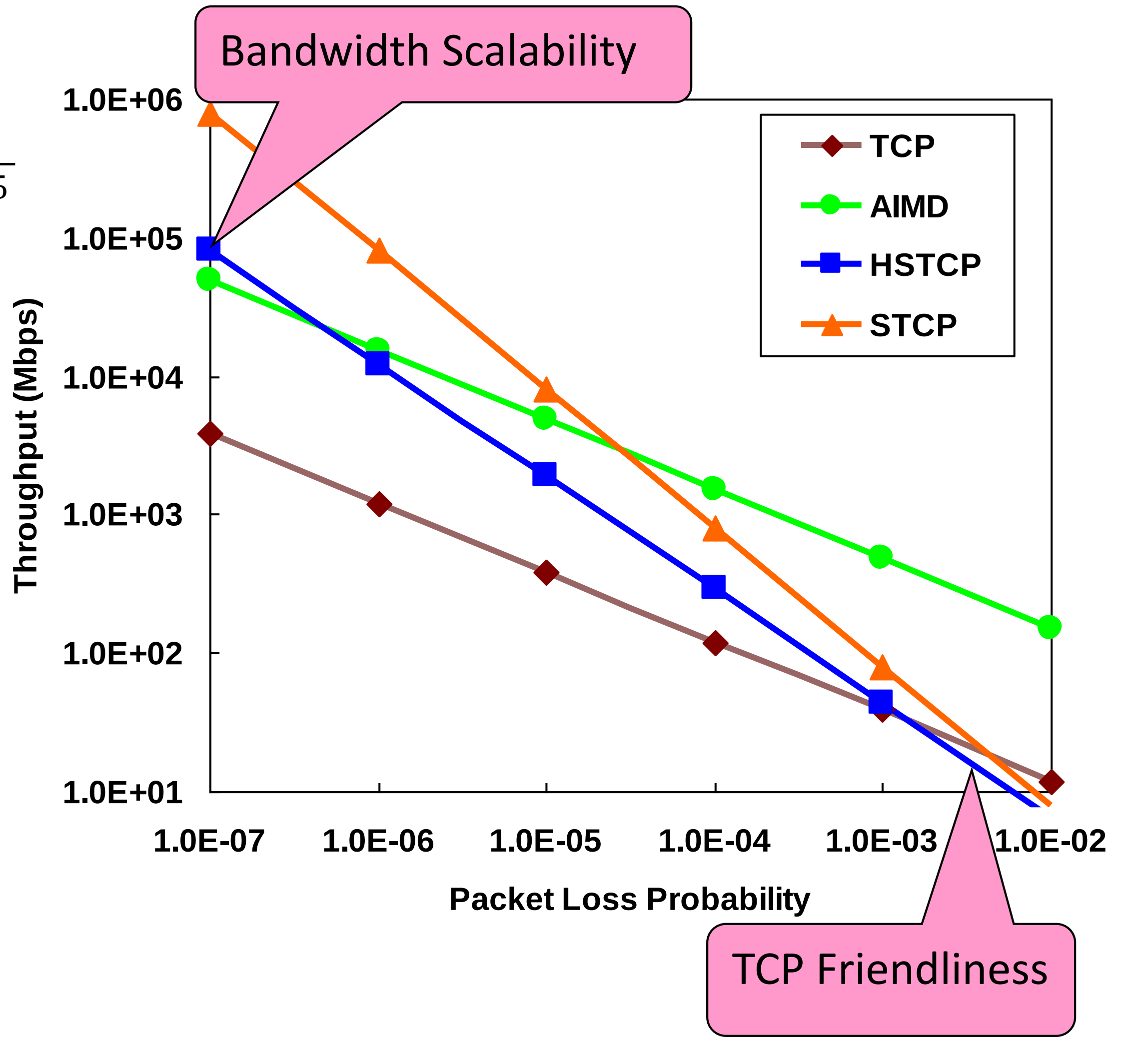
- STCP adaptively increases *cwnd*, and decreases *cwnd* by 1/8.



Response Functions of HSTCP and STCP

- HSTCP: $R = \dfrac{MSS}{RTT}\dfrac{0.12}{p^{0.835}}$
- STCP: $R = \dfrac{MSS}{RTT}\dfrac{0.08}{p}$

HSTCP and STCP have both bandwidth scalability and TCP friendliness



Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

Congestion Control Algorithms in Windows and MacOS X

- Compound TCP (CTCP) [Kun 06]
  - Windows XP (x64), Vista, 2008, 7, 8, 8.1, and 10
- Data Center TCP (DCTCP) [Alizadehzy 10]
  - Windows 2012, 10
- CUBIC [Ha 08]
  - (Recent) Windows 10, 2016 [CUBIC]
  - MacOS X
- TCP NewReno [NewReno]
  - MacOS X

[Kun 06] Kun Tan, Jingmin Song, Qian Zhang, and Murari Sridharan. A Compound TCP Approach for High-speed and Long Distance Networks. Proceedings of the 25th IEEE International Conference on Computer Communications, pp. 1-12, April 2006.

[Alizadehzy 10] Mohammad Alizadehzy, Albert Greenbergy, David A. Maltzy, Jitendra Padhyey, Parveen Pately, Balaji Prabhakarz, Sudipta Senguptay, and Murari Sridharan. Data Center TCP (DCTCP). Proceedings of the SIGCOMM 2010, pp. 63-74, August 2010.

[Ha 08] Sangtae Ha, Injong Rhee, and Lisong Xu. CUBIC: A New TCP Friendly HighSpeed TCP Variant. ACM SIGOPS Operating Systems Review - Research and developments in the Linux kernel archive Vol. 42 No. 5, pp. 64-74, July 2008.

[CUBIC] https://blogs.technet.microsoft.com/networking/2017/07/13/core-network-stack-features-in-the-creators-update-for-windows-10/

[NewReno] https://tools.ietf.org/html/rfc6582

TCP Configuration in Windows10

選択Windows PowerShell

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

PS C:¥Windows¥System32¥WindowsPowerShell¥v1.0> get-nettcpsetting internetcustom


SettingName                     : InternetCustom
MinRto(ms)                      : 300
InitialCongestionWindow(MSS)    : 10
CongestionProvider              : CUBIC
CwndRestart                     : False
DelayedAckTimeout(ms)           : 40
DelayedAckFrequency             : 2
MemoryPressureProtection        : Disabled
AutoTuningLevelLocal            : Restricted
AutoTuningLevelGroupPolicy      : NotConfigured
AutoTuningLevelEffective        : Local
EcnCapability                   : Disabled
Timestamps                      : Disabled
InitialRto(ms)                  : 3000
ScalingHeuristics               : Disabled
DynamicPortRangeStartPort       : 1025
DynamicPortRangeNumberOfPorts   : 64510
AutomaticUseCustom              : Disabled
NonSackRttResiliency            : Disabled
ForceWS                         : Enabled
MaxSynRetransmissions           : 2
AutoReusePortRangeStartPort     : 0
AutoReusePortRangeNumberOfPorts : 0
```

TCP Configuration in MacOS X (High Sierra)

$ sysctl -a

ターミナル — less — 80×24

```
net.inet.tcp.max_persist_timeout: 0
net.inet.tcp.always_keepalive: 0
net.inet.tcp.timer_fastmode_idlemax: 10
net.inet.tcp.broken_peer_syn_rexmit_thres: 10
net.inet.tcp.tcp_timer_advanced: 9
net.inet.tcp.tcp_resched_timerlist: 217
net.inet.tcp.pmtud_blackhole_detection: 1
net.inet.tcp.pmtud_blackhole_mss: 1200
net.inet.tcp.cc_debug: 0
net.inet.tcp.newreno_sockets: 0
net.inet.tcp.background_sockets: 11
net.inet.tcp.cubic_sockets: 45
net.inet.tcp.use_newreno: 0
net.inet.tcp.cubic_tcp_friendliness: 0
net.inet.tcp.cubic_fast_convergence: 0
net.inet.tcp.cubic_use_minrtt: 0
net.inet.tcp.lro_sz: 8
net.inet.tcp.lro_time: 10
net.inet.tcp.bg_target_qdelay: 100
net.inet.tcp.bg_allowed_increase: 8
net.inet.tcp.bg_tether_shift: 1
net.inet.tcp.bg_ss_fltsz: 2
net.inet.udp.checksum: 1
:
```

Congestion Control Algorithms in Linux

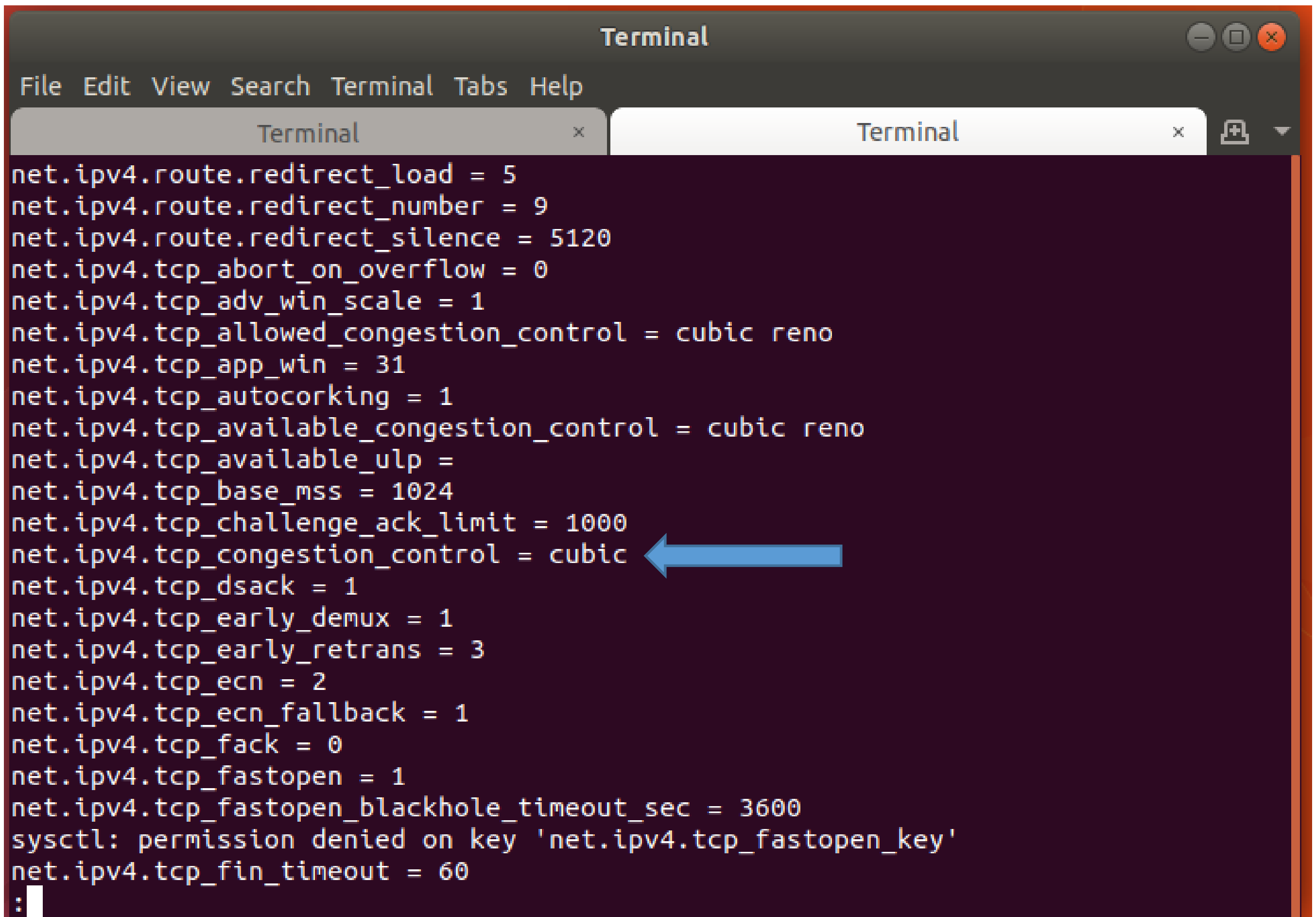
- Binary Increase Congestion Control (BIC) [Xu 04]
- CUBIC
- Westwood
- H-TCP
- High Speed TCP
- TCP-Hybla Congestion Control Algorithm
- TCP Vegas
- TCP NV
- Scalable TCP
- TCP Low Priority
- TCP Veno
- YeAH TCP
- TCP Illinois
- Data Center TCP (DCTCP)
- CAIA Delay-Gradient (CDG)
- Bottleneck Bandwidth and RTT (BBR) [Cardwell 17]

[Xu 04] Lisong Xu, K. Harfoush, and Injong Rhee. Binary Increase Congestion Control (BIC) for Fast Long-Distance Networks. In Proceedings of the Twenty-third Annual Joint Conference of the IEEE Computer and Communications Societies, Vol. 4, pp. 1-11, March 2004.

[Cardwell 17] Neal Cardwell, Yuchung Cheng, C. Stephen Gunn, Soheil Hassas Yeganeh, Van Jacobson. BBR: Congestion-Based Congestion Control. ACM Queue, Vol. 14, No. 5, December 2016.

TCP Configuration in Linux 4.13

$ sysctl -a



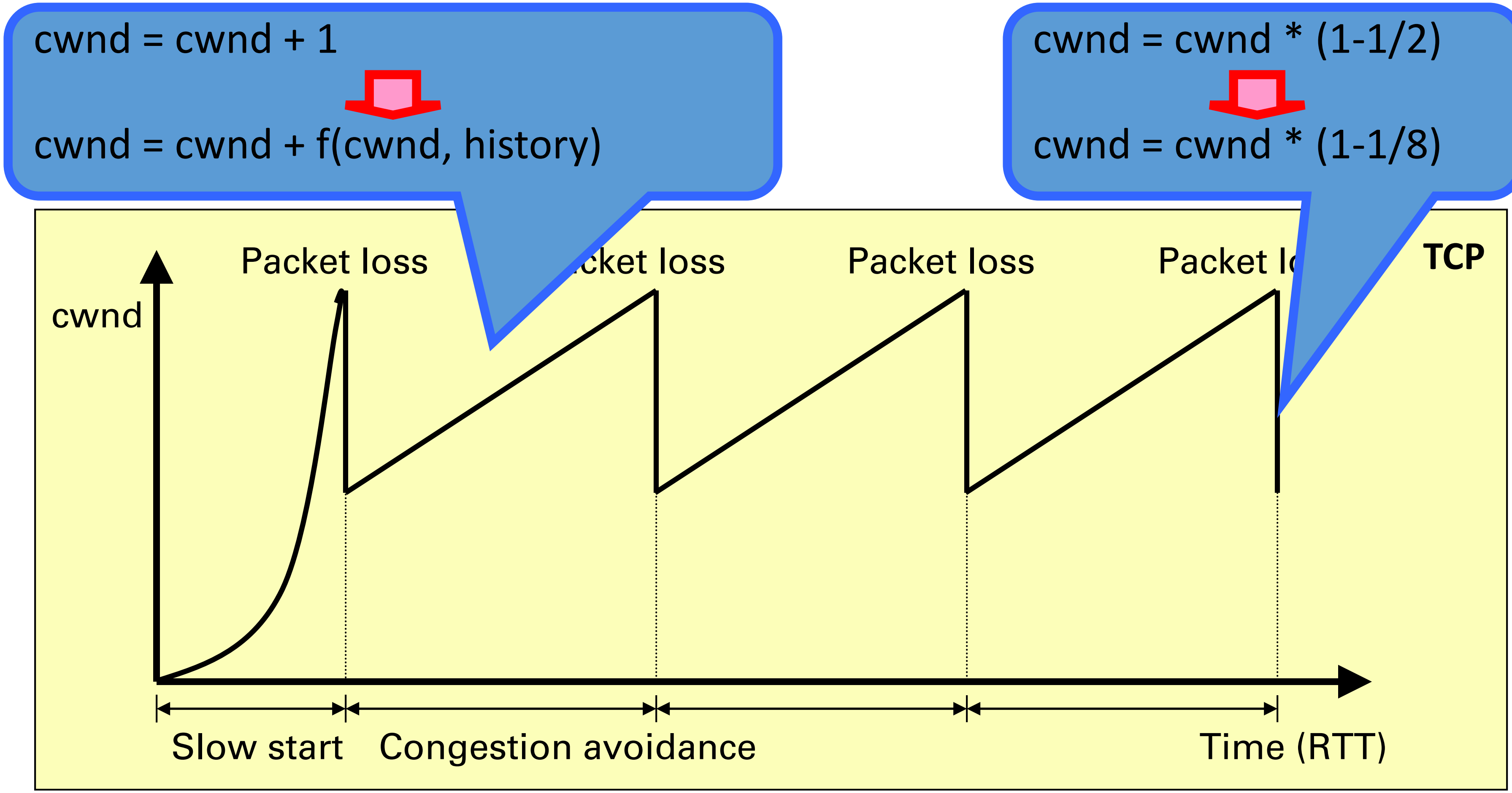
Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

Binary Increase Congestion control (BIC)

- satisfies:
  - Round Trip Time (RTT) Fairness
  - TCP Friendliness
  - Bandwidth Scalability
- has two windows size control policies.
  - Additive Increase
  - Binary Search Increase
- has been mainlined since Linux 2.6.6.
  - net/ipv4/tcp_bic.c

BIC adaptively increase cwnd, and decrease cwnd by 1/8



A Search Problem

- We consider the increase part of congestion avoidance as a search problem, in which a connection looks for the available bandwidth by comparing its current throughput with the available bandwidth, and adjusting cwnd accordingly.

- Q: How to compare R with A?

  R = current throughput

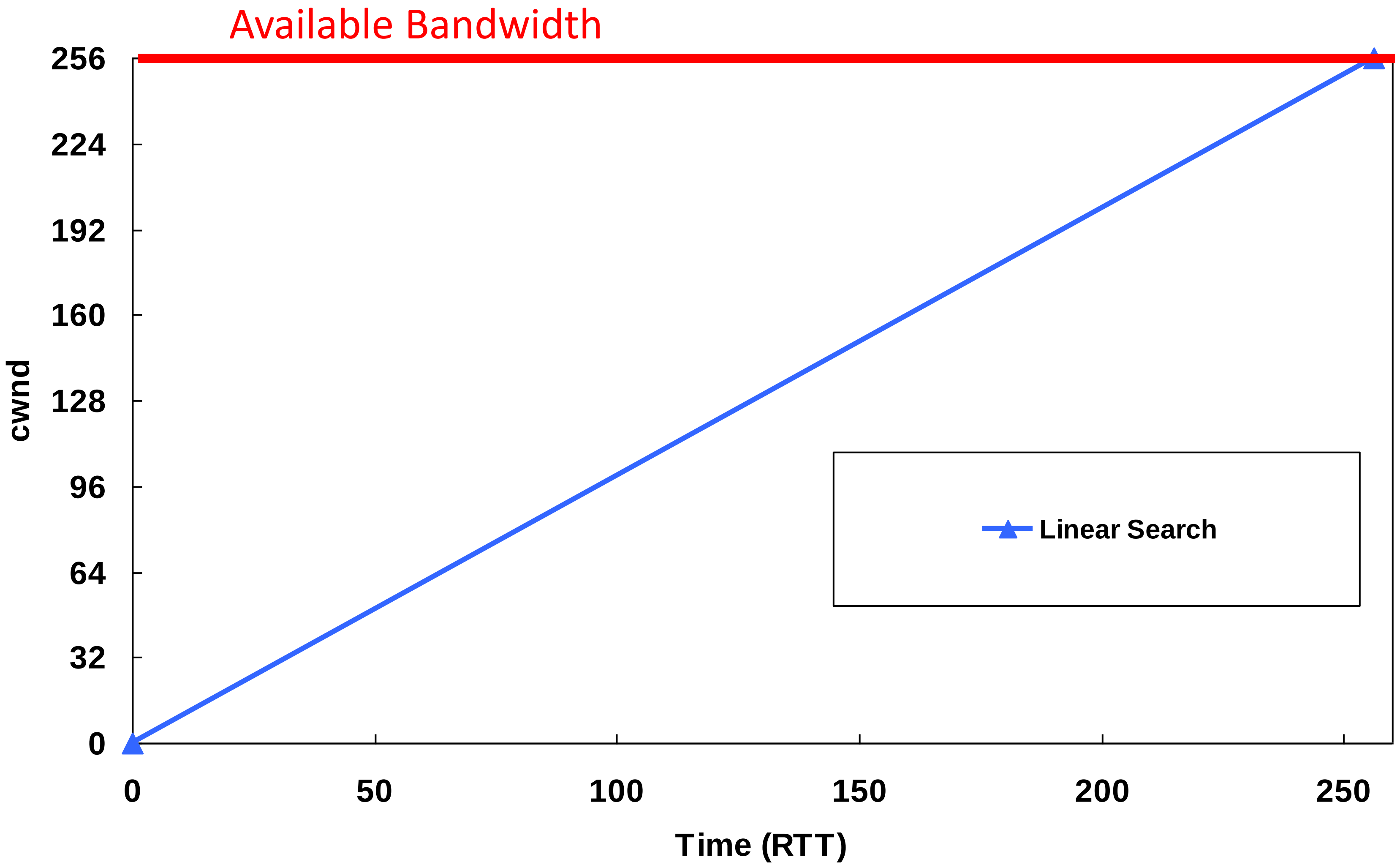
  = cwnd/RTT

  A = available bandwidth
- A: Check for packet losses
  - No packet loss: R <= A
  - Packet losses : R > A

- How does TCP find the available bandwidth?
- Linear search

```
while (no packet loss){
    cwnd++;
}
```

Linear Search



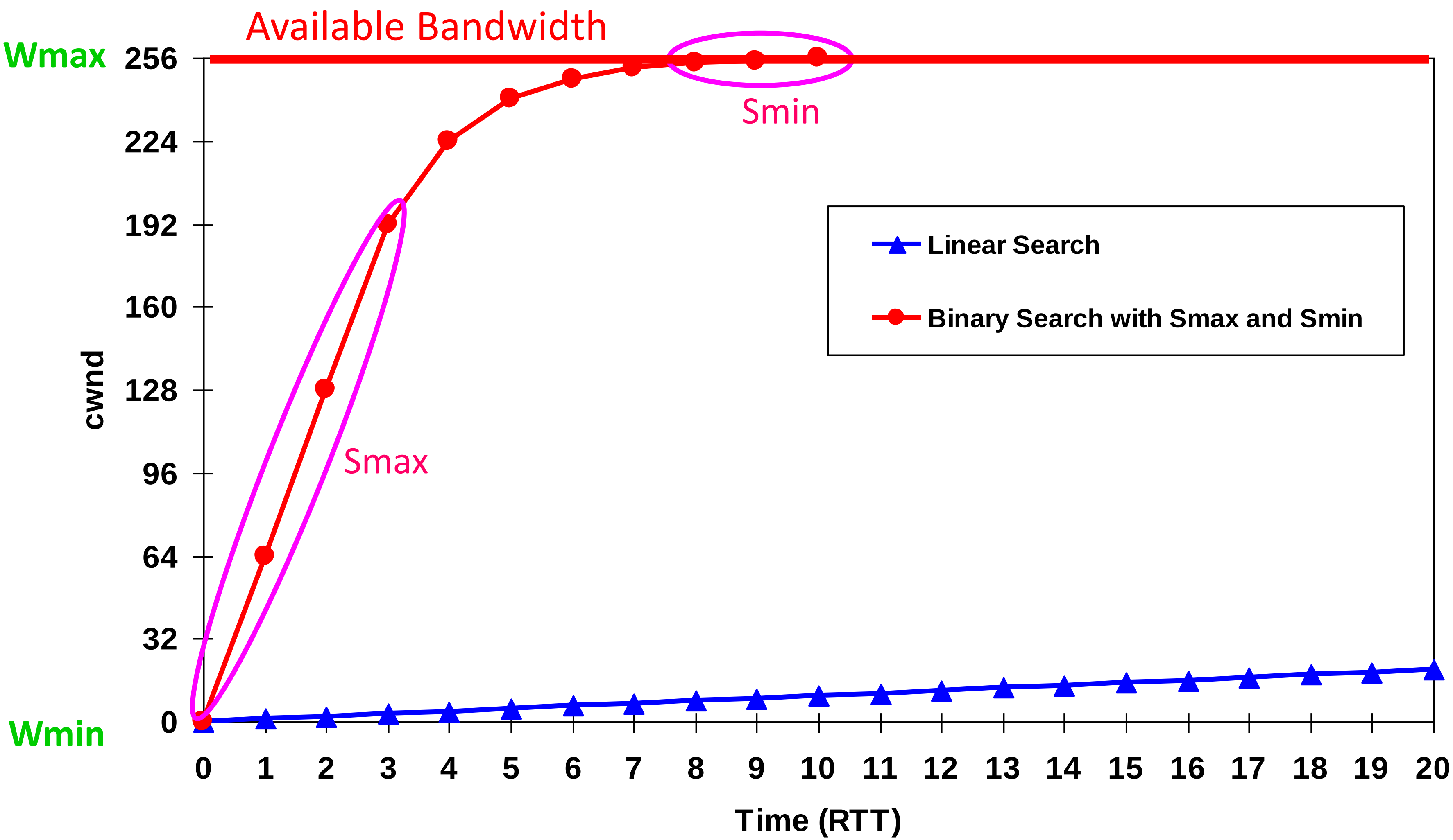
BIC: Binary Search with Smax and Smin

- BIC - Binary search

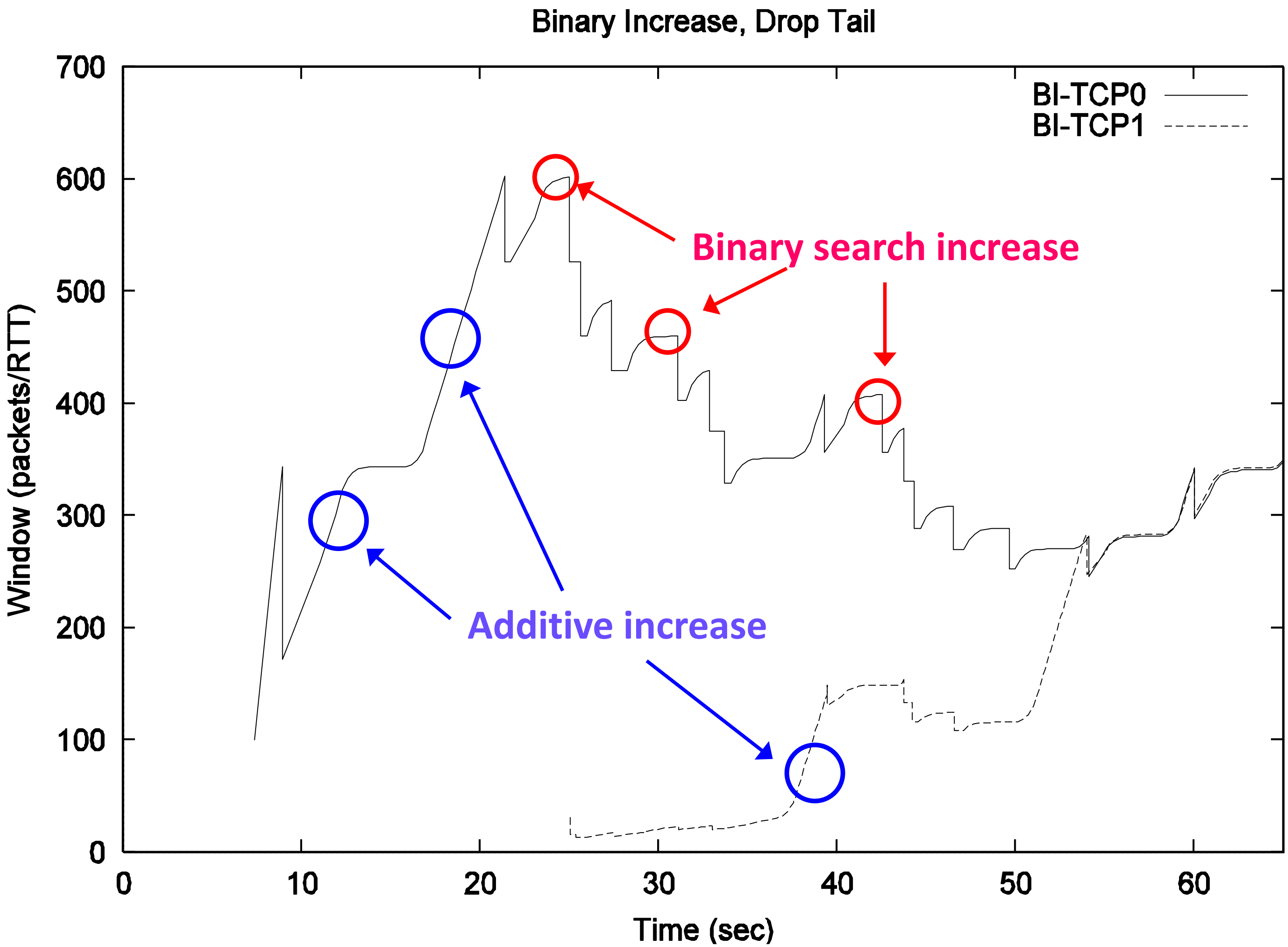
```
while (Wmin <= Wmax){
    inc = (Wmin+Wmax)/2 - cwnd;
    if (inc > Smax)
            inc = Smax;
    else if (inc < Smin)
            inc = Smin;
    cwnd = cwnd + inc;
    if (no packet losses)
            Wmin = cwnd;
    else
            break;
}
```

- Wmax: Max Window
- Wmin: Min Window
- Smax: Max Increment
- Smin: Min Increment

Binary Search with Smax and Smin



Binary Increase Congestion Control (BIC)

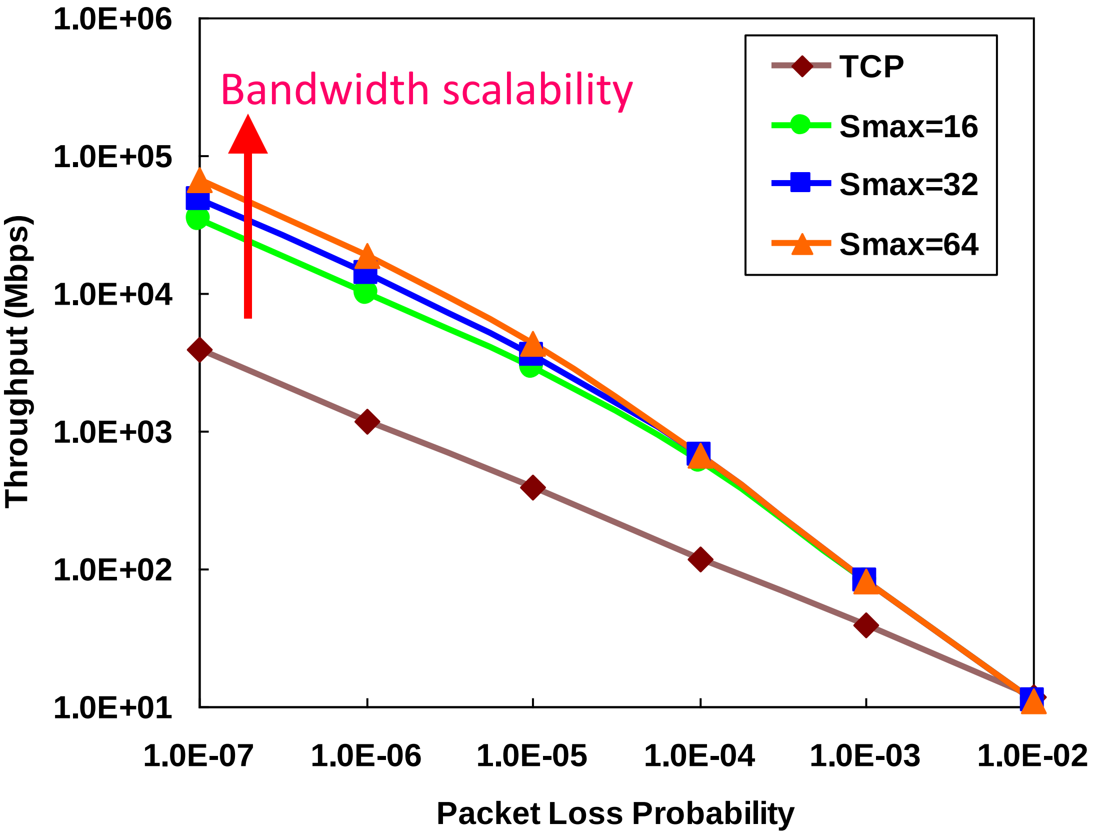


Setting Smax

- Response Function of BIC on high-speed networks

$$R = \frac{MSS}{RTT} \frac{2.7\sqrt{S_{\max}}}{p^{0.5}}$$

- Bandwidth scalability of BIC depends only on Smax
- RTT Fairness of BIC on high-speed networks is the same as that of AIMD

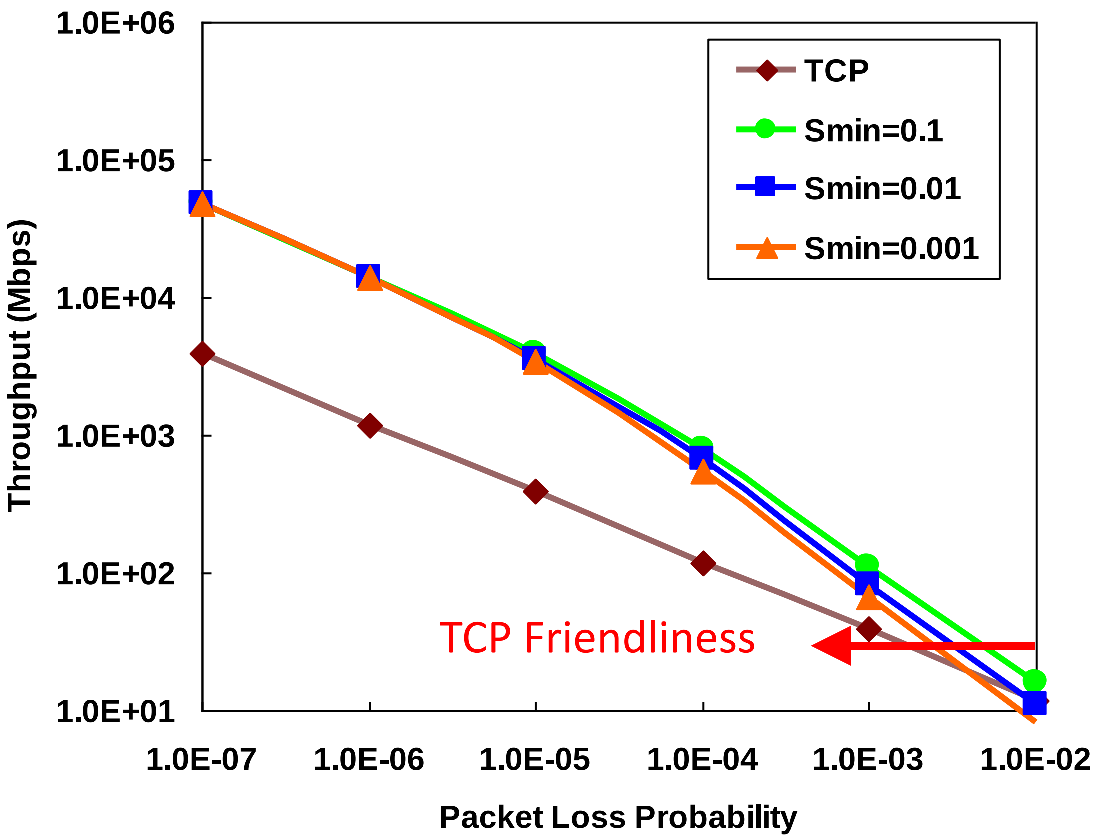


Setting Smin

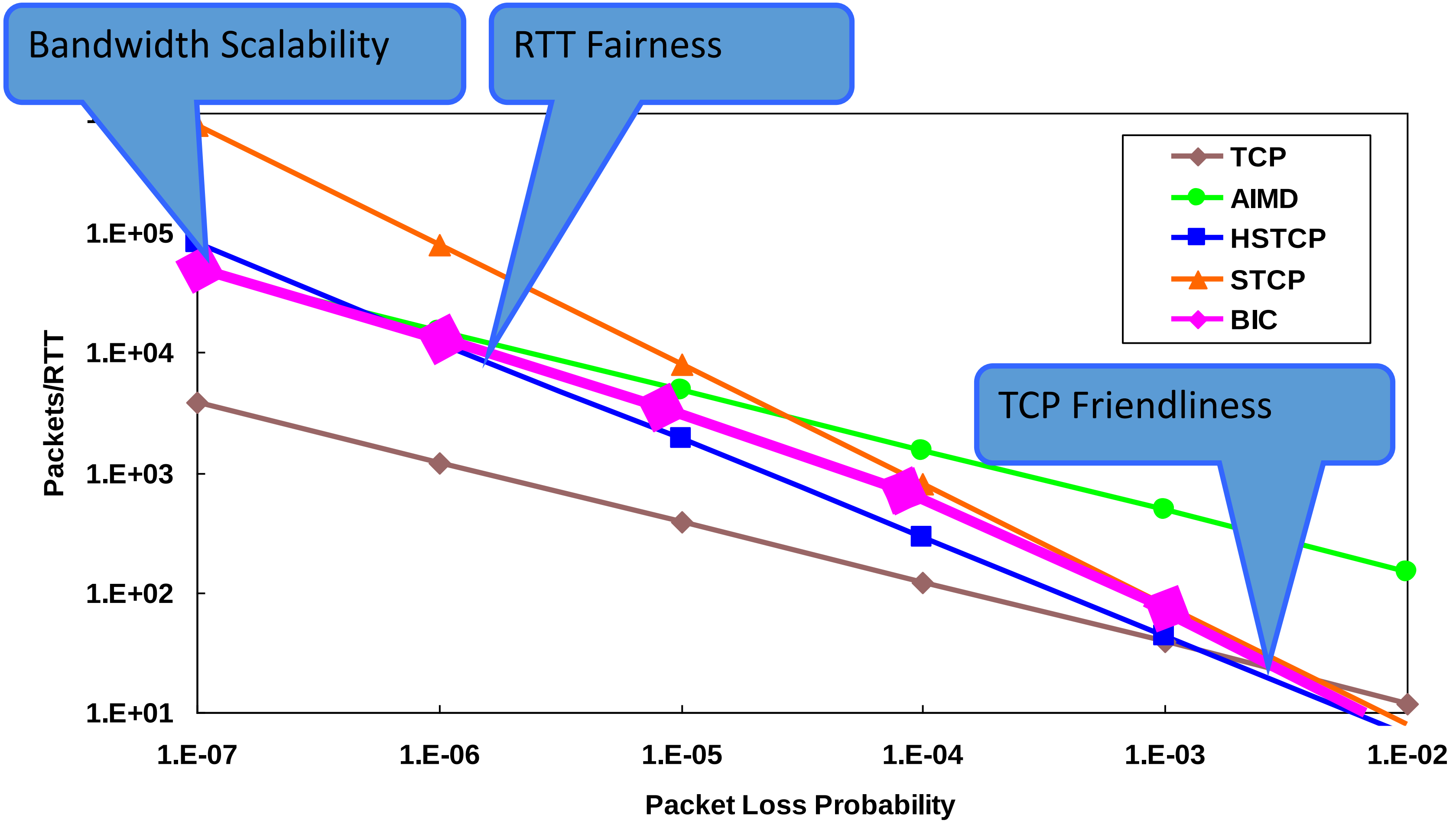
- Response Function of BIC on low-speed networks

$$R = \frac{MSS}{RTT} f\left(p, S_{\min}\right)$$

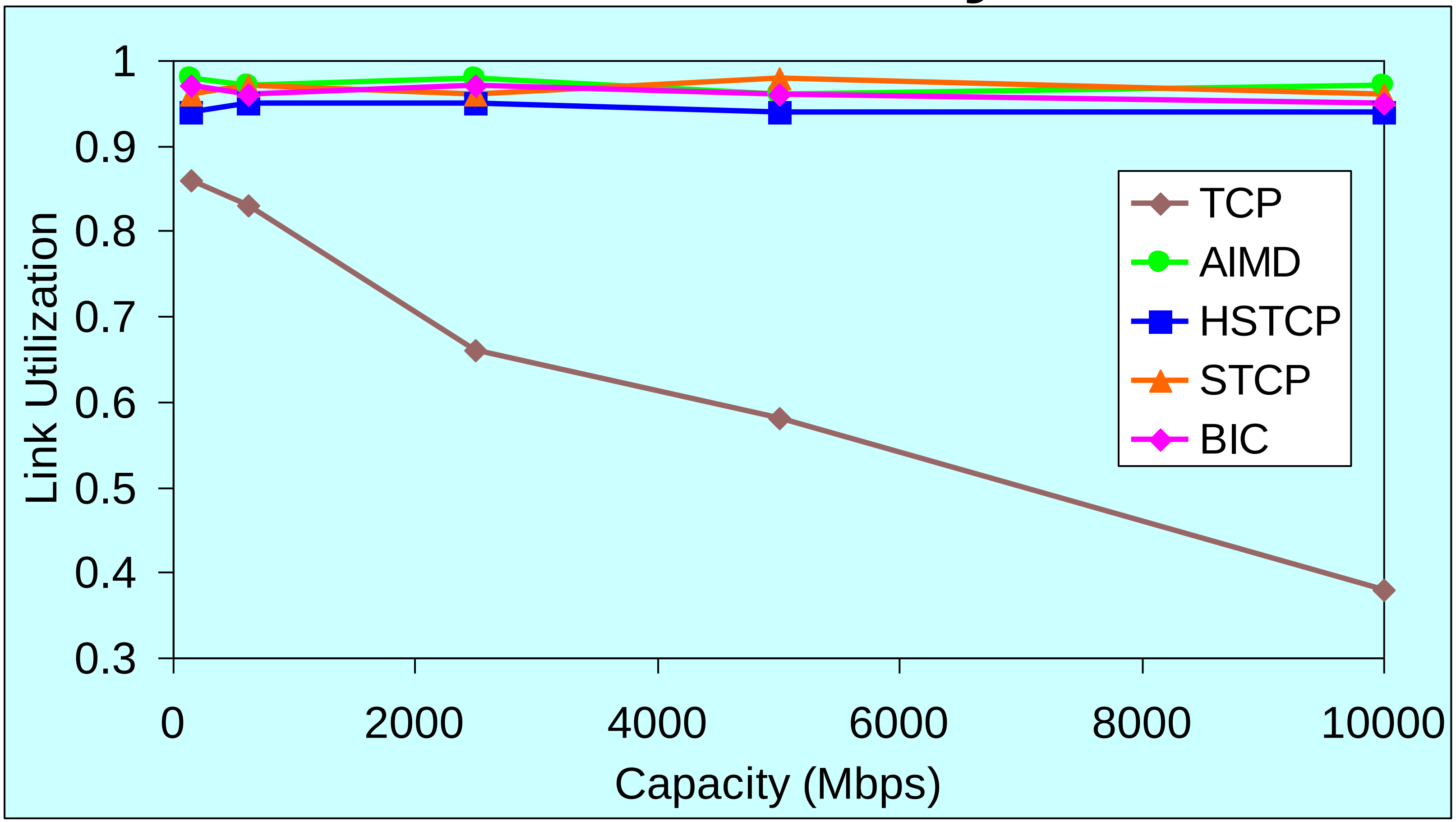
- TCP friendliness of BIC depends only on Smin



Response Functions



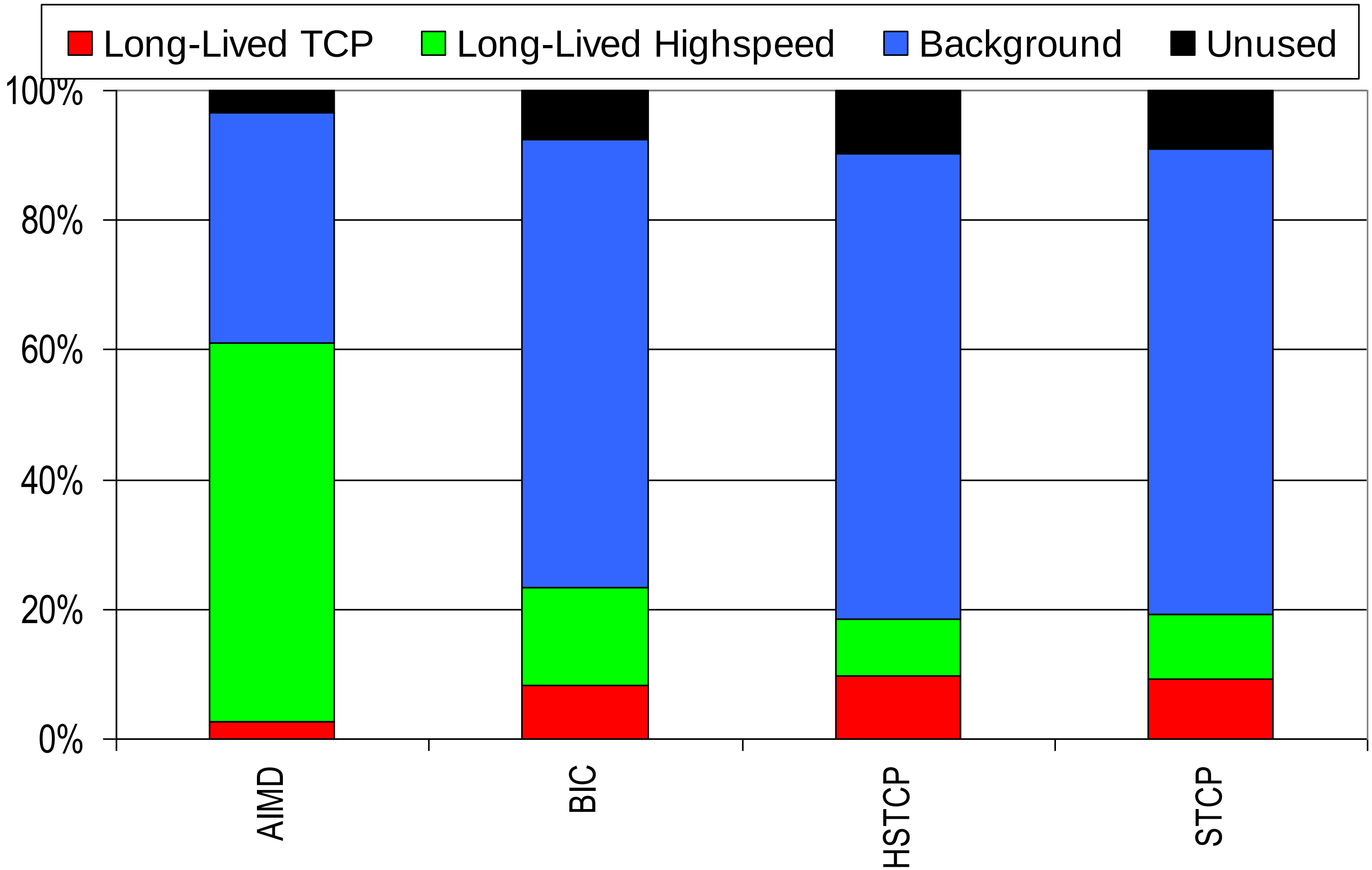
Bandwidth Scalability



NS-2 Simulation (100 sec)

- Link Capacity = 155Mbps, 622Mbps, 2.5Gbps, 5Gbps, 10Gbps,
- Drop-Tail Routers, 0.1BDP Buffer
- 5 Connections, 100ms RTT, 1000-Byte Packet Size

TCP Friendliness:
Balanced Link Utilizations are Better



Simulation setup: 20Mbps, BDP Buffer, Drop Tail, Reverse Traffic, 2 TCP flows, 2 high-speed flows, and some background traffic

RTT Fairness: Values close to 1 are Better

- Throughput ratio of two flows with different RTTs on a 2.5Gbps link

Inverse RTT Ratio	1	3	6
BIC	1	12	38
AIMD	1	6	22
HSTCP	1	29	107
STCP	1	127	389

Simulation setup: BDP Buffer, Drop Tail, Reverse Traffic, Forward Background Traffic (short-lived TCP, Web Traffic)

Summary of BIC

	AIMD	HSTCP	STCP	BIC
Scalability	√	√	√	√
TCP Friendliness	×	√	√	√
RTT Fairness	√	×	×	√

Outline

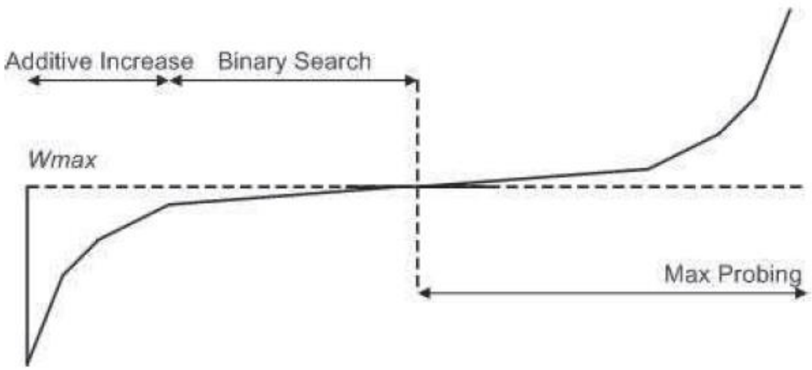
- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

CUBIC

- is an enhanced version of BIC.
  - simplifies the BIC window control using a cubic function.
  - improves its TCP friendliness & RTT fairness
  - Window growth becomes independent on RTT
    - RTT fairness and also TCP friendliness – under low delays.
- has been mainlined since Linux 2.6.13.
  - net/ipv4/tcp_cubic.c
- $ cat /proc/sys/net/ipv4/tcp_congestion_control
  - cubic

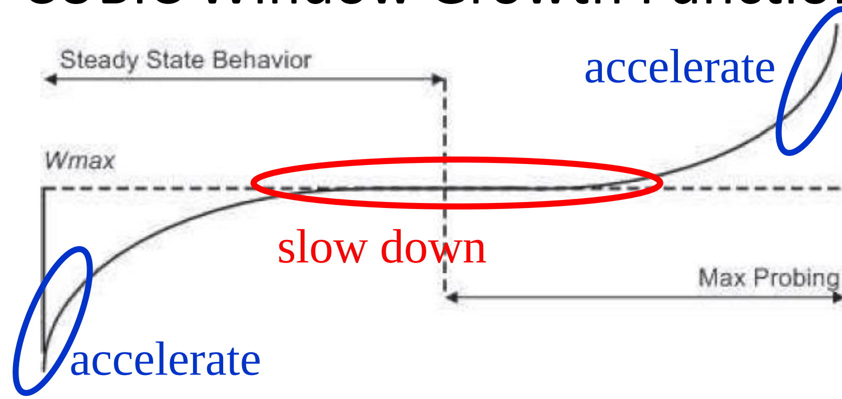
BIC and CUBIC Functions

BIC Window Growth Function



- Max probing uses a window growth function exactly symmetric to those used in additive increase and binary search (which is logarithmic; its reciprocal will be exponential) and then additive increase.
- BIC (also HSTCP & STCP) window growth function can be still aggressive for TCP especially under short RTTs or low speed networks.

CUBIC Window Growth Function



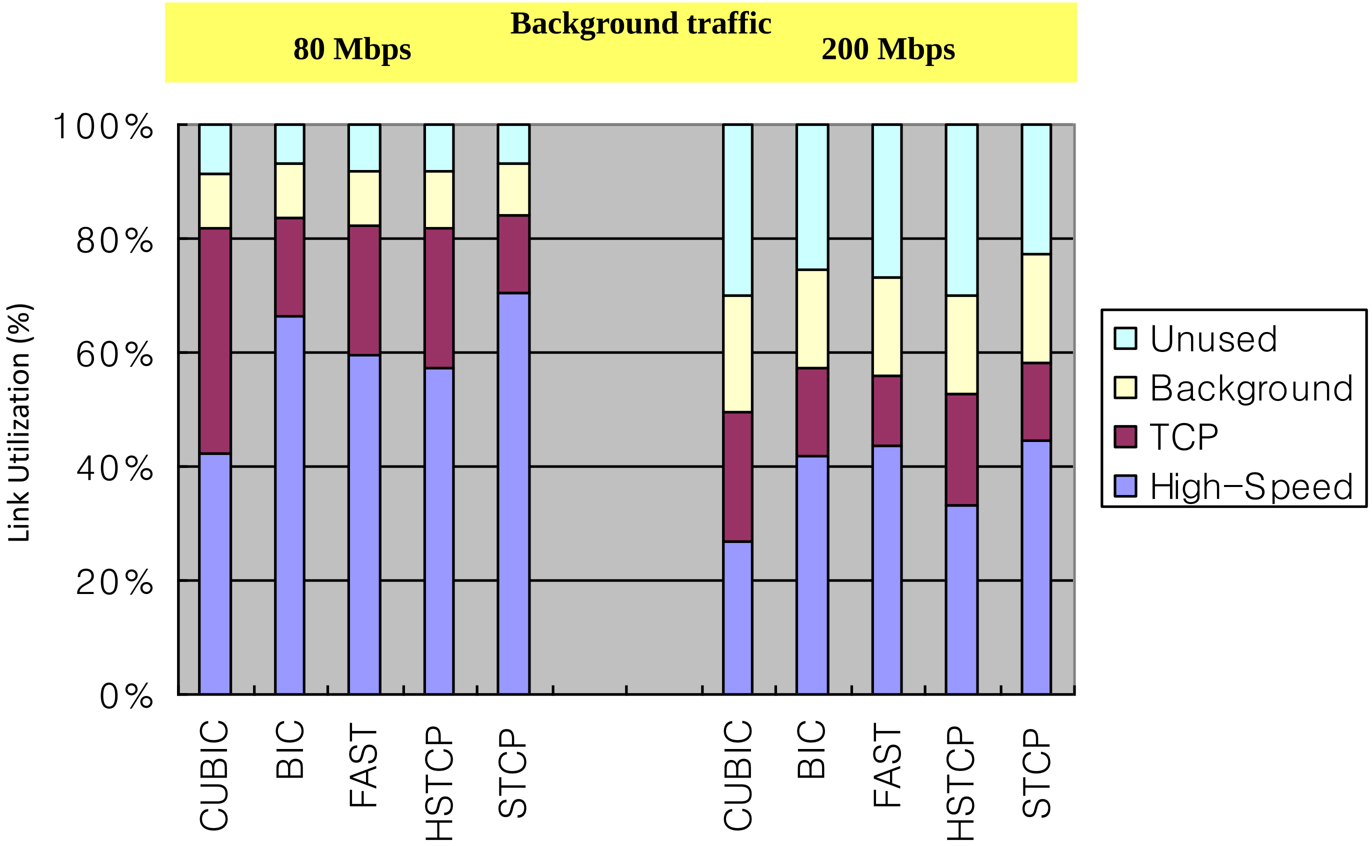
$$W_{cubic} = C(t-K)^3 + W_{\max}$$

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

where ***C*** is a scaling factor, ***t*** is the elapsed time from the last window reduction, and ***β*** is a constant multiplication decrease factor

TCP Friendliness (1/4)
Balanced Link Utilizations are Better

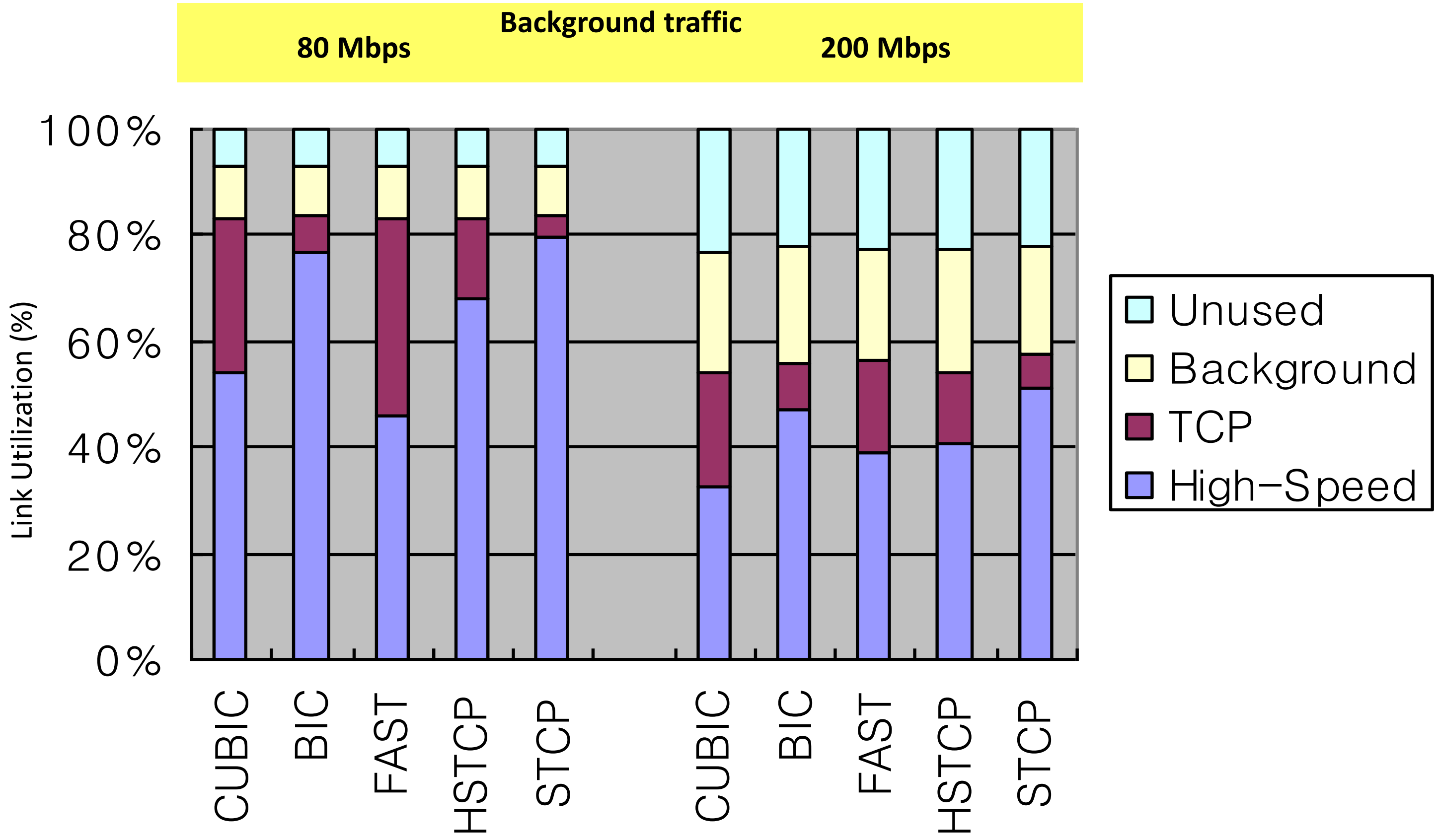
- Dummynet Testbed : RTT 5ms & 800 Mbps, 100% router buffer of the BDP
  with 80 ~ 200 Mbps background traffic



TCP Friendliness on short RTT - 5ms

TCP Friendliness (2/4)
Balanced Link Utilizations are Better

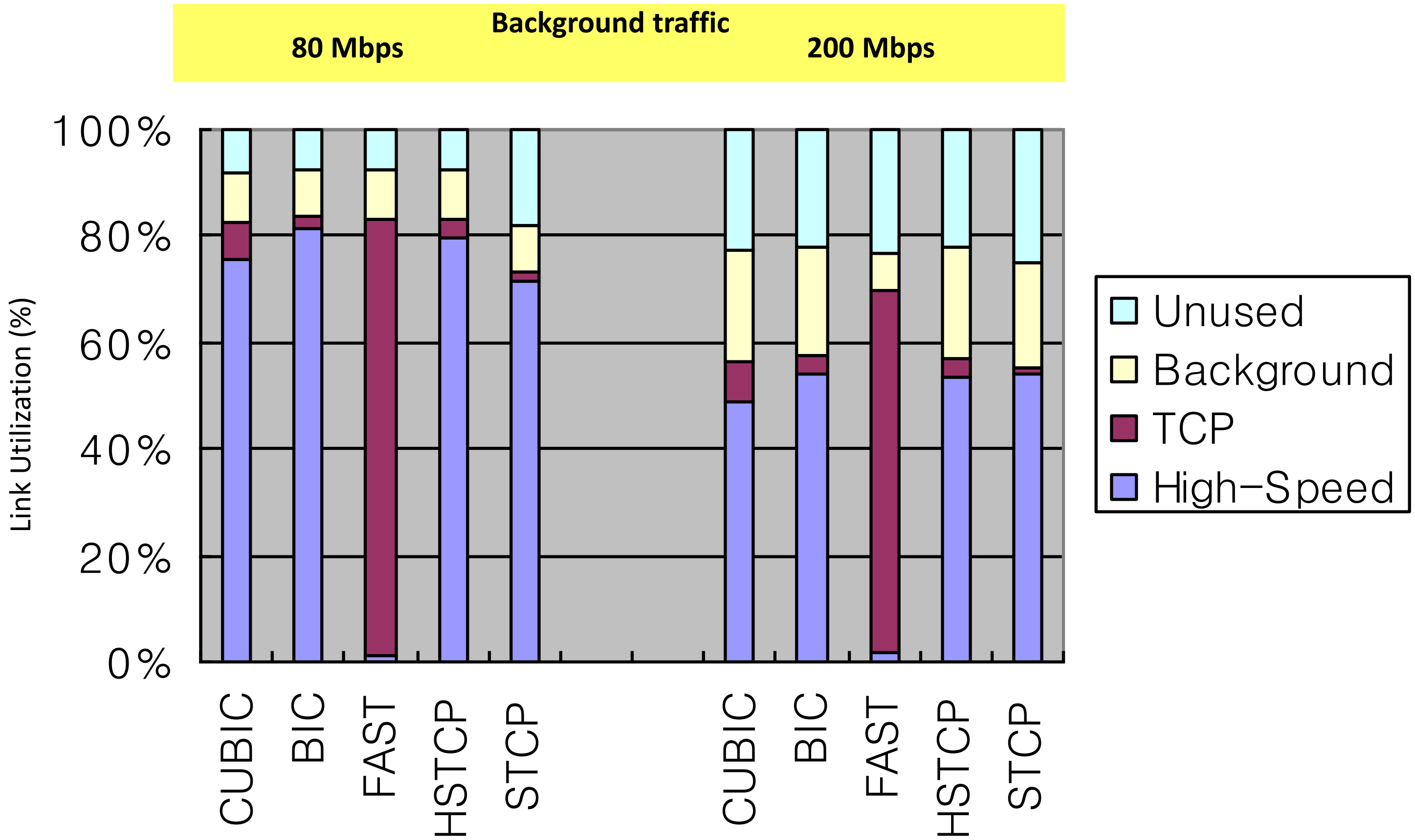
- Dummynet Testbed : RTT 10ms & 800 Mbps, 100% router buffer of the BDP with 80 ~ 200 Mbps background traffic



TCP Friendliness on short RTT - 10ms

TCP Friendliness (3/4)
Balanced Link Utilizations are Better

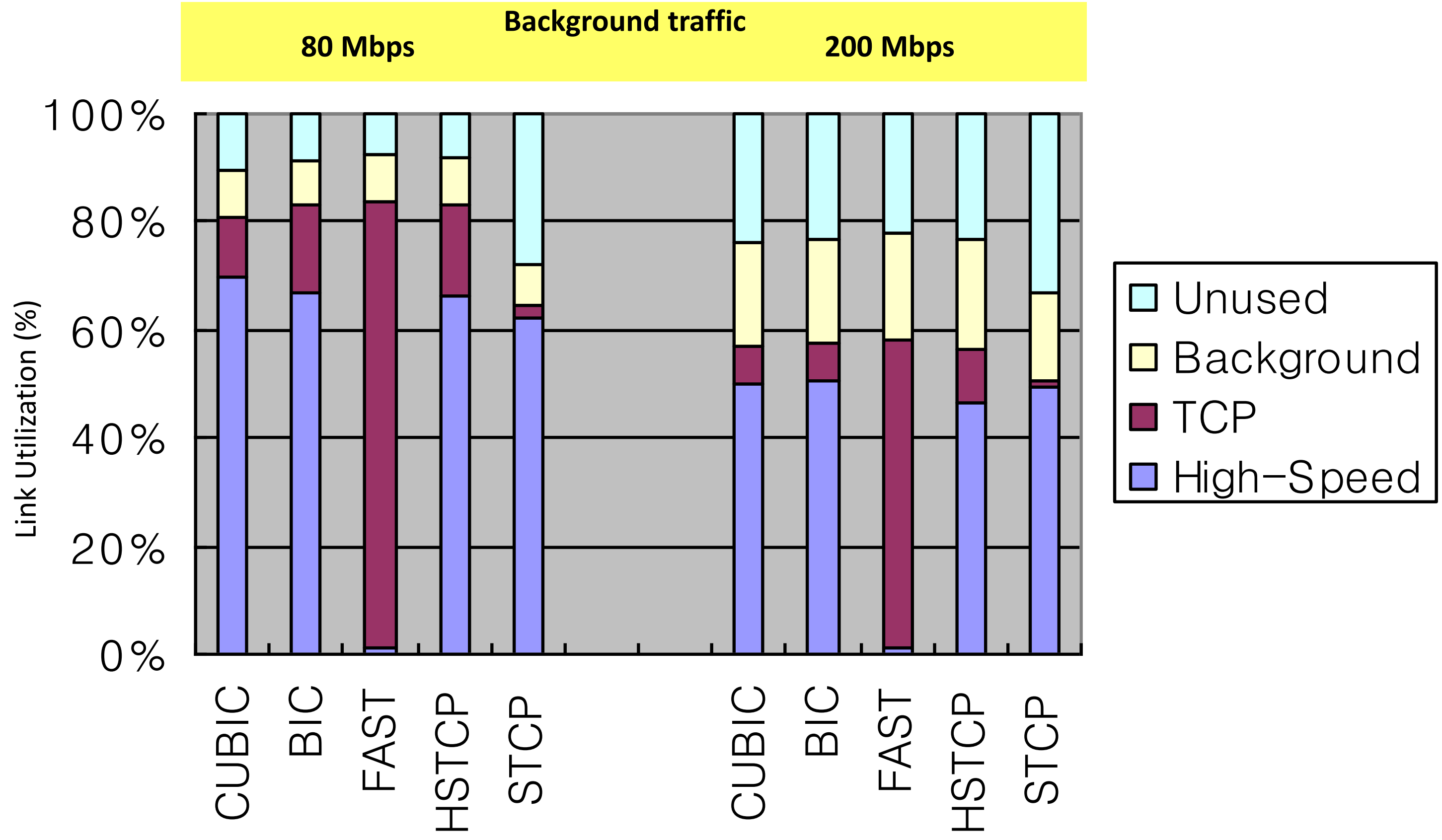
- Dummynet Testbed : RTT 100ms & 800 Mbps, 100% router buffer of the BDP with 80 ~ 200 Mbps background traffic



TCP Friendliness on long RTT - 100ms

TCP Friendliness (4/4)
Balanced Link Utilizations are Better

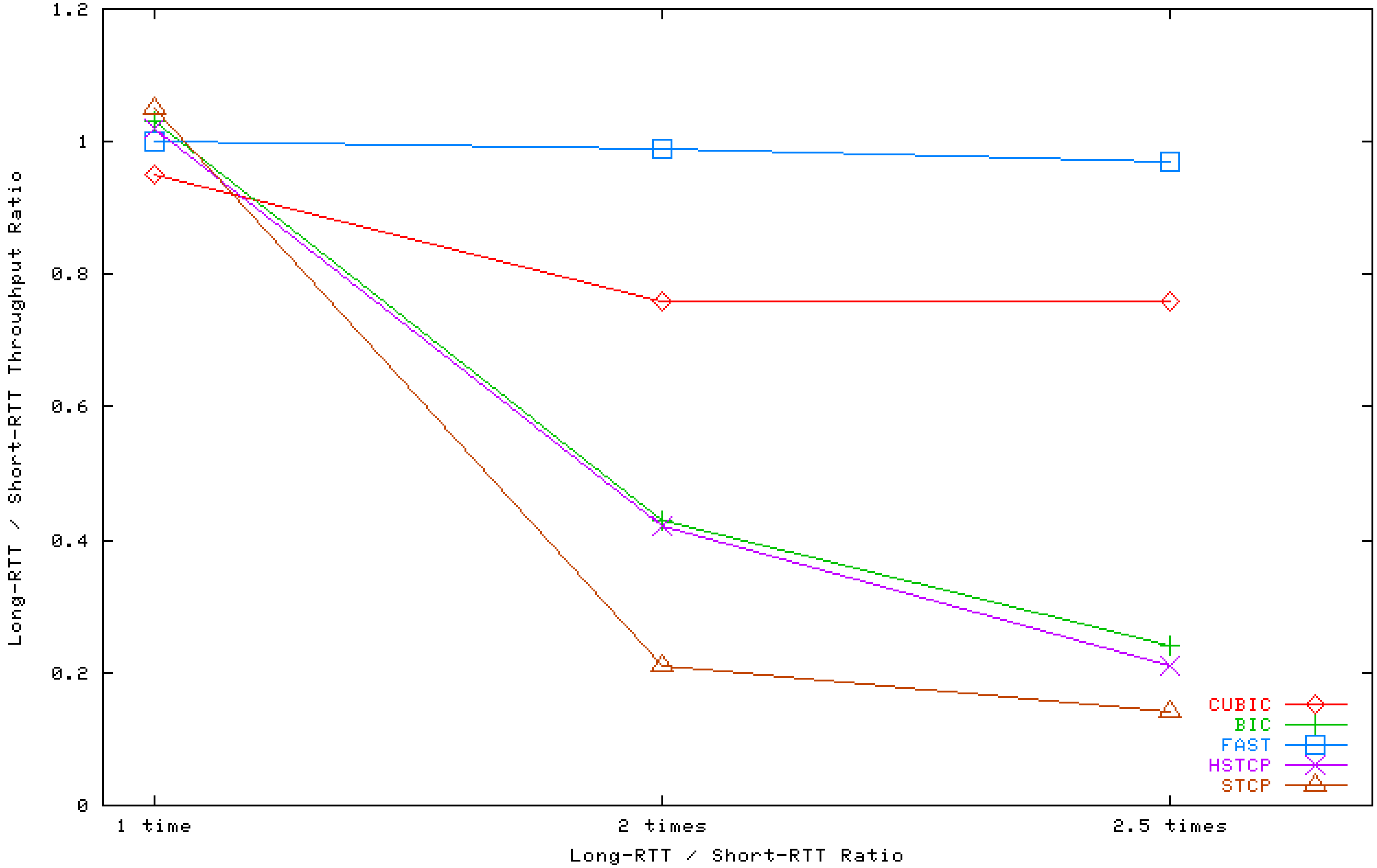
- Dummynet Testbed : RTT 200ms & 800 Mbps, 100% router buffer of the BDP with 80 ~ 200 Mbps background traffic



TCP Friendliness on long RTT - 200ms

RTT Fairness: Values close to 1 are Better

Dummynet testbed : RTT 40, 120, 240 ms & 800 Mbps, Router buffer: 50% of the BDP with 200 Mbps background traffic



Summary of CUBIC

- CUBIC and HSTCP had good TCP Friendliness especially on short RTT networks. FAST [Wei 06] needs alpha parameter tuning.
- CUBIC and FAST had good RTT Fairness under both short and long RTT paths.

[Wei 06] David X. Wei, Cheng Jin, Steven H. Low, and Sanjay Hegde. FAST TCP: Motivation, Architecture, Algorithms, Performance. IEEE/ACM Trans. on Networking. Vol. 14 No. 6, pp. 1246–1259, December 2006.

Outline

- High-Performance Networking
- Control of TCP
- Congestion Control Algorithms in OS
- BIC
- CUBIC
- BBR

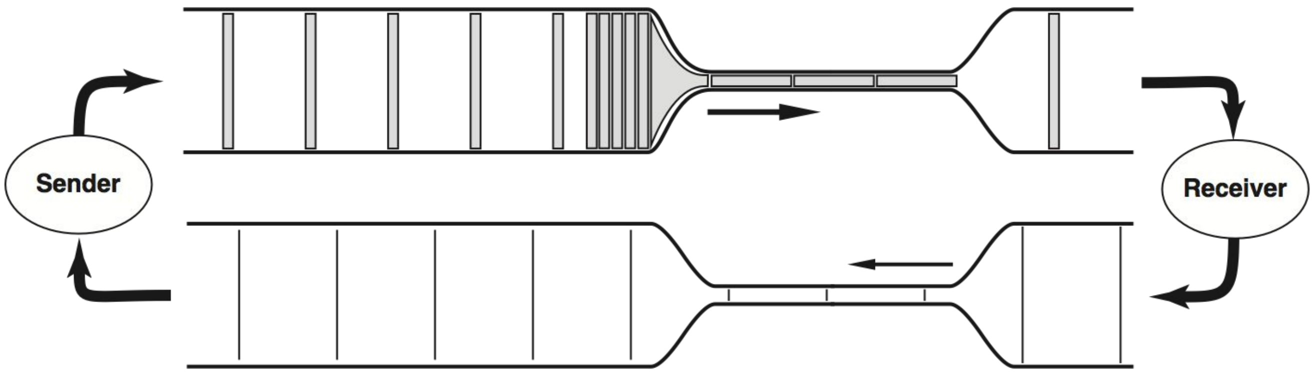
Bottleneck Bandwidth and RTT (BBR)

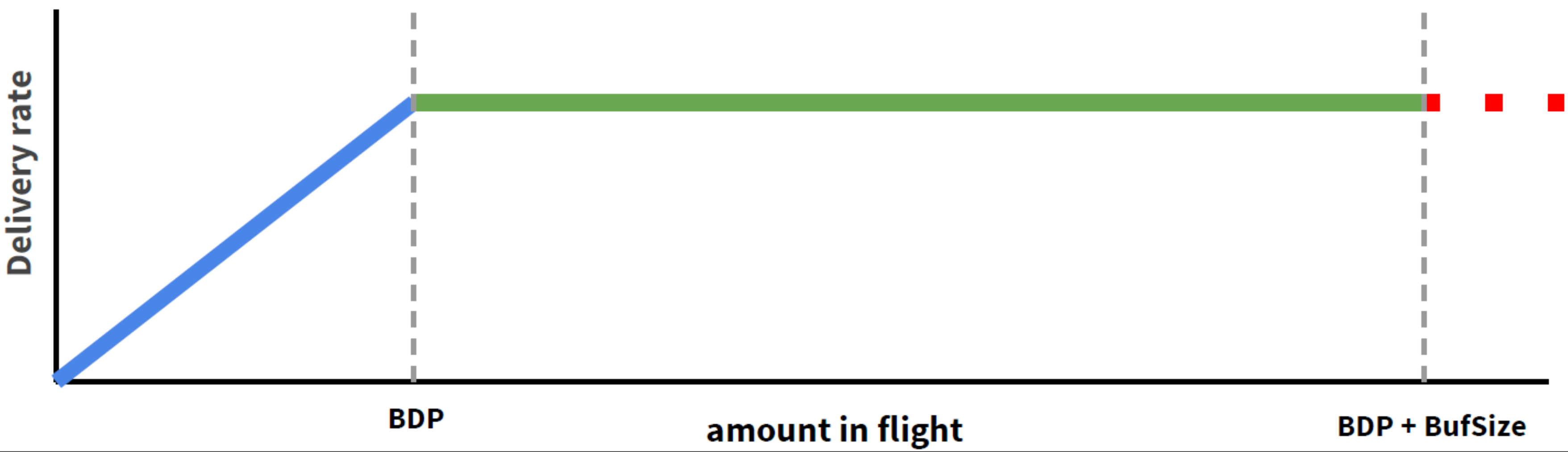
- BBR seeks high throughput with a small queue by probing BW and RTT
- Ground-up redesign of congestion control
  - Not loss-based, delay-based, ECN-based, AIMD-based
- Models the network path: probes and estimates max BW and min RTT
- Result:
  - High throughput even with shallow buffers and moderate loss rates
  - Low delay even with deep buffers ("bufferbloat")
- Used at Google: internal WAN networks; rolling out on google.com, YouTube
- Open source (Linux 4.9 TCP)
- Incrementally deployable: sender-only upgrade

Motivation of BBR

- The problem: Internet performance is often not realizing its potential:
  - Last-mile networks with seconds of latency ("bufferbloat")
  - Gigabit wide area networks need infeasible loss rates to use bandwidth
    - 10Gbit/sec, 100ms RTT needs 1/30M loss
      (if 1% loss, get .003 Gbit/sec)
- The culprit: loss-based congestion control
  - "Congestion Avoidance and Control", Jacobson & Karels, SIGCOMM 1988
  - Internet standard
- Google's unique vantage point for attacking the problem:
  - Representative traffic to every corner of the Internet
  - Control network stack on the senders

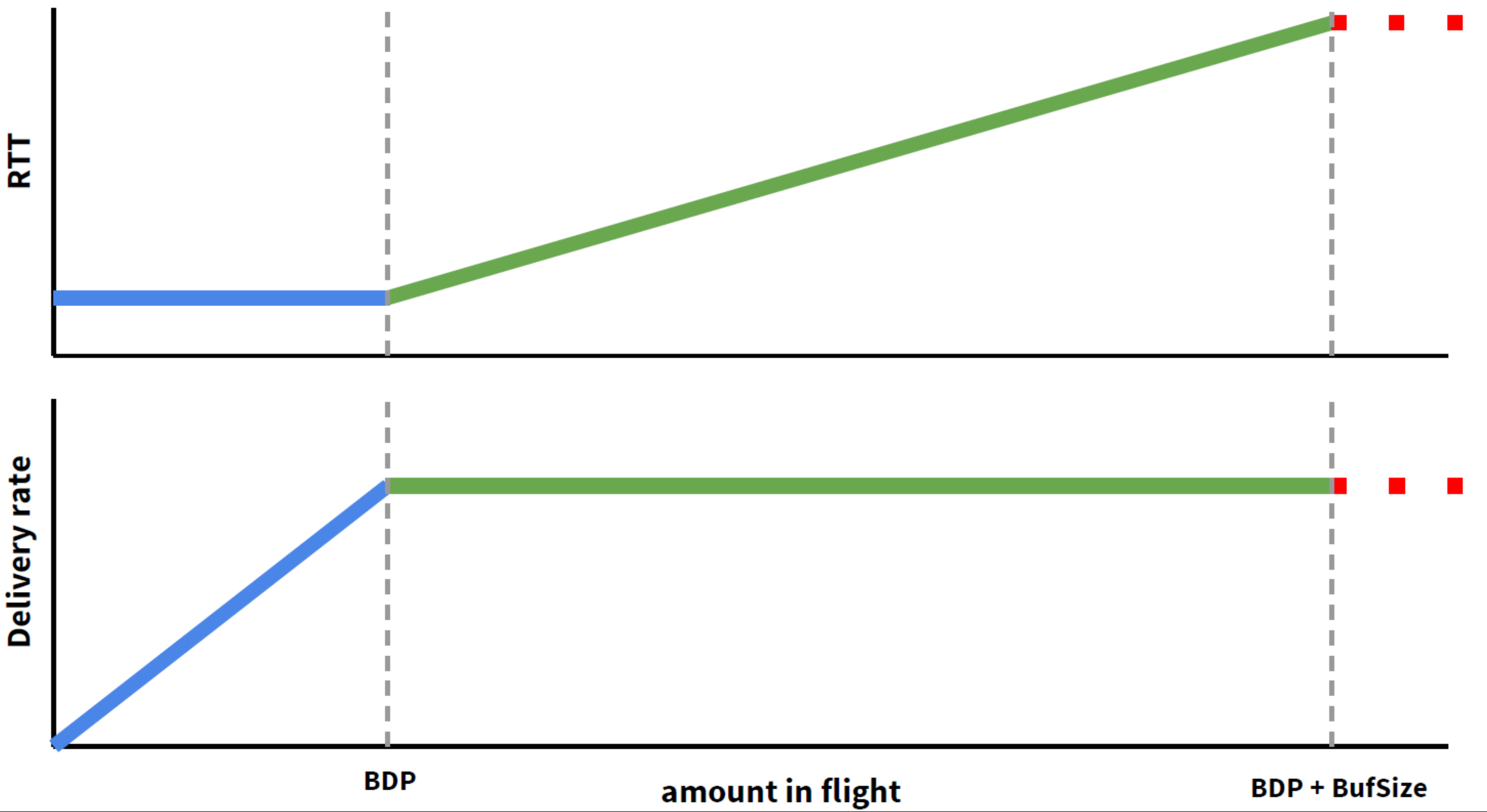
Network Congestion and Bottlenecks: Bandwidth



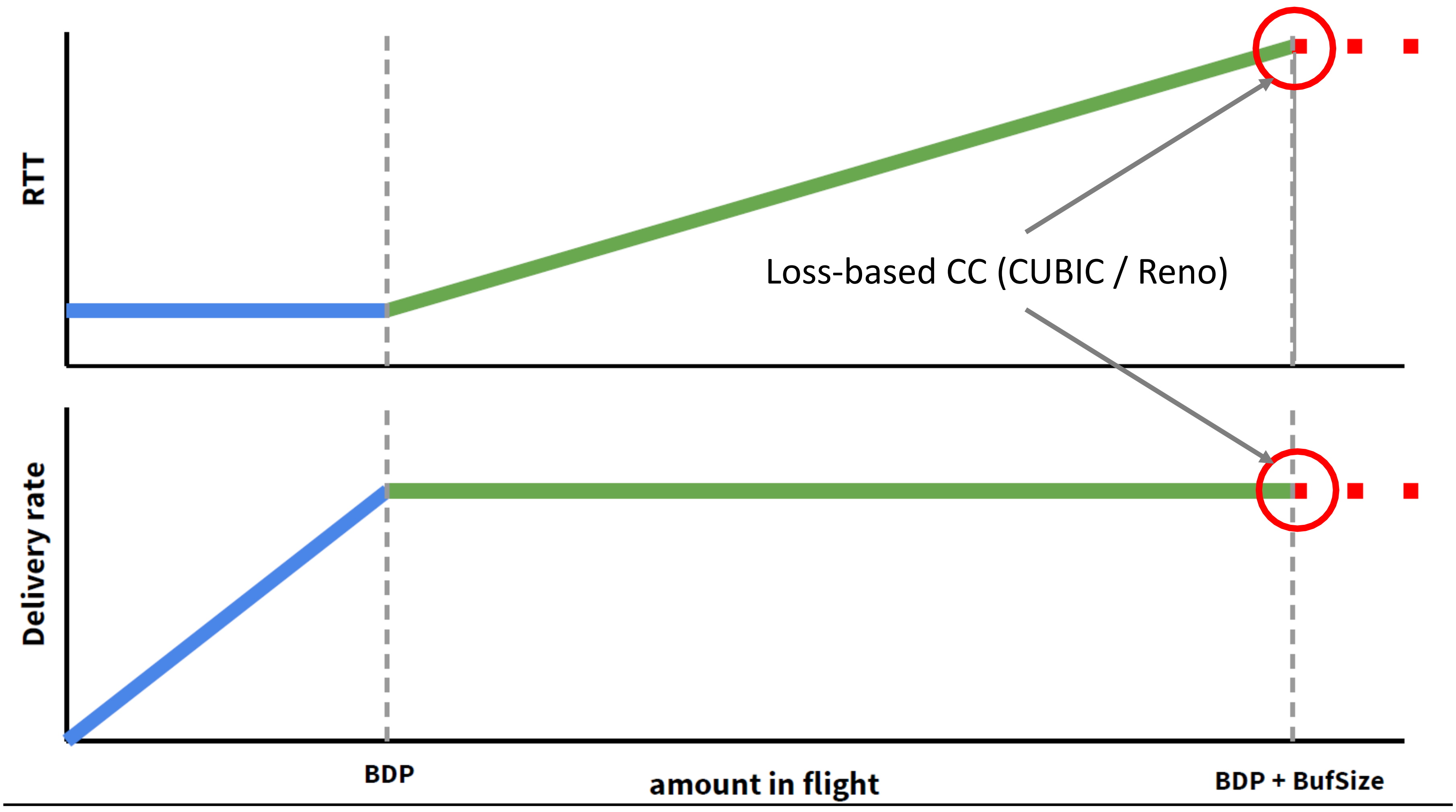


*BDP: Bandwidth Delay Product

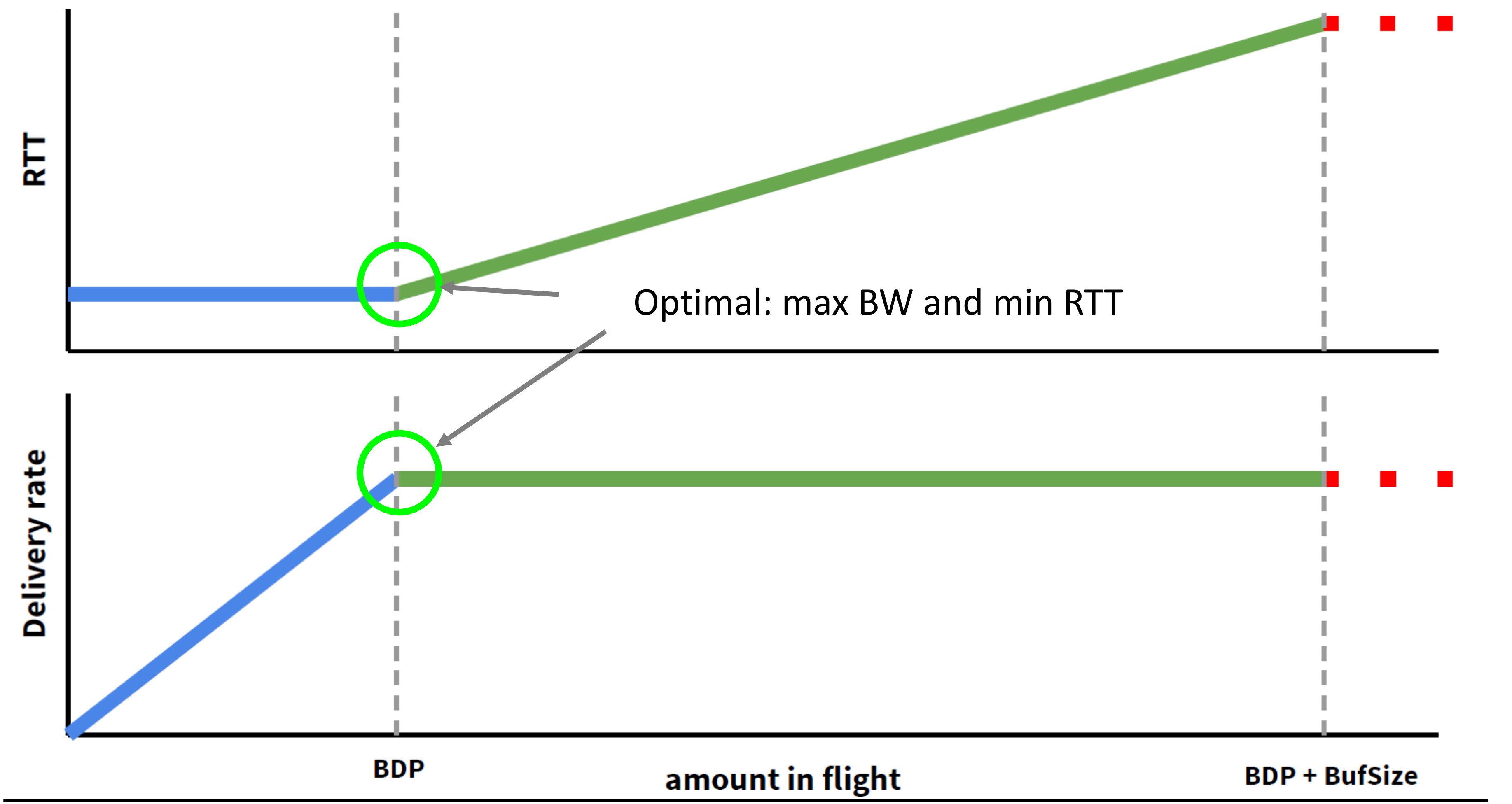
Network Congestion and Bottlenecks: Delay



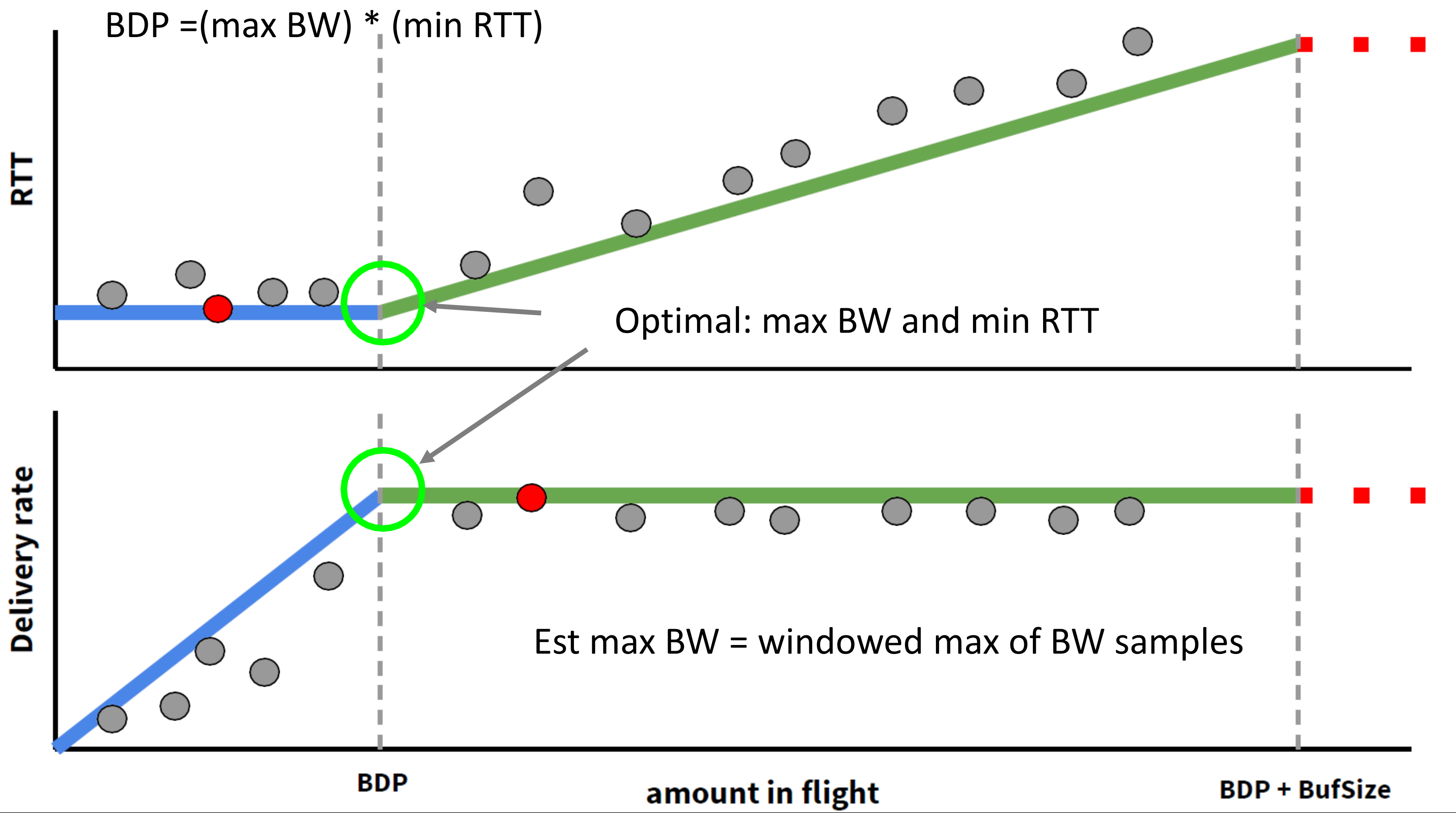
Loss-Based Congestion Control



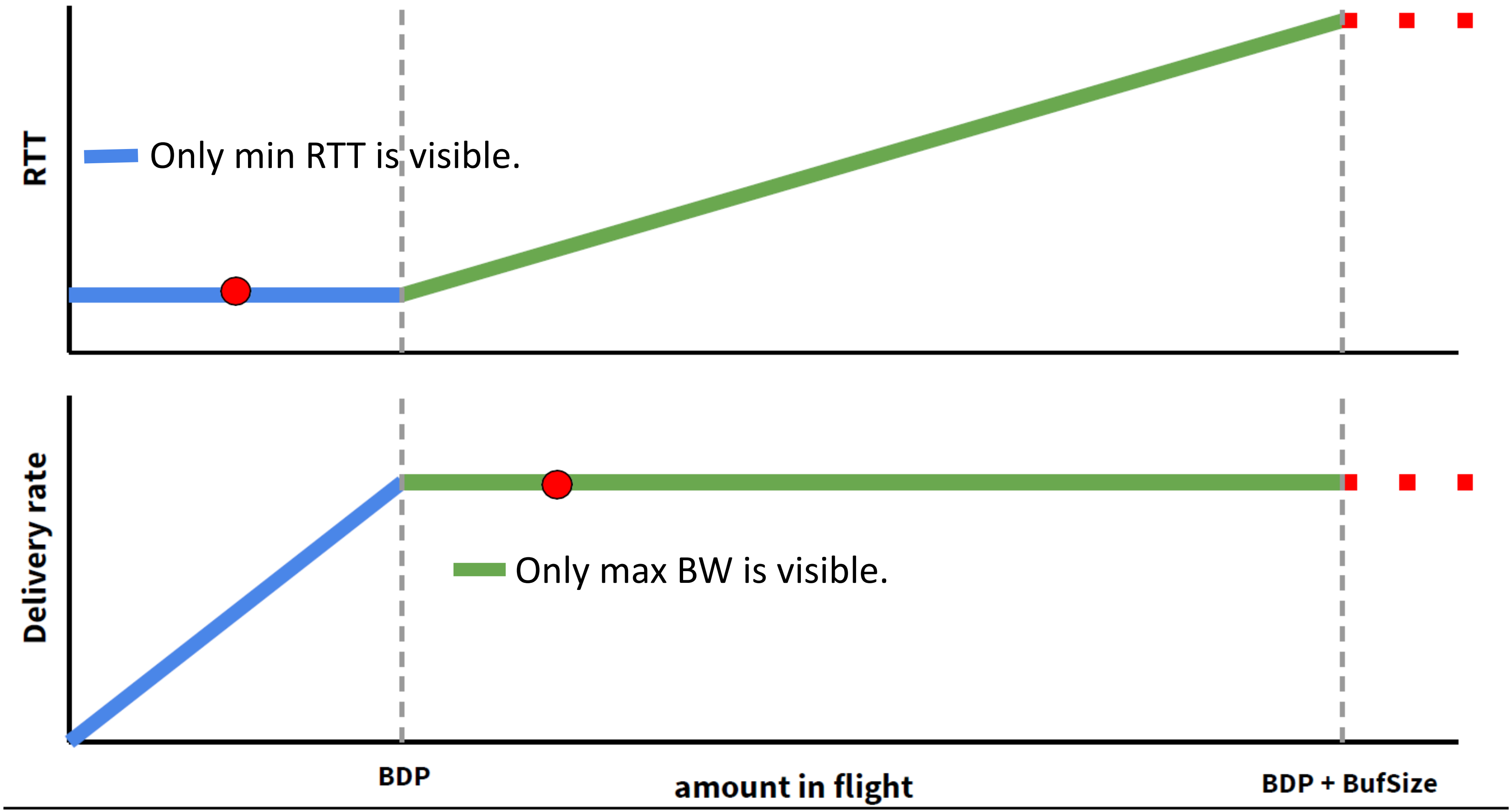
Optimal Operating Point



Estimating Optimal Point (max BW, min RTT)



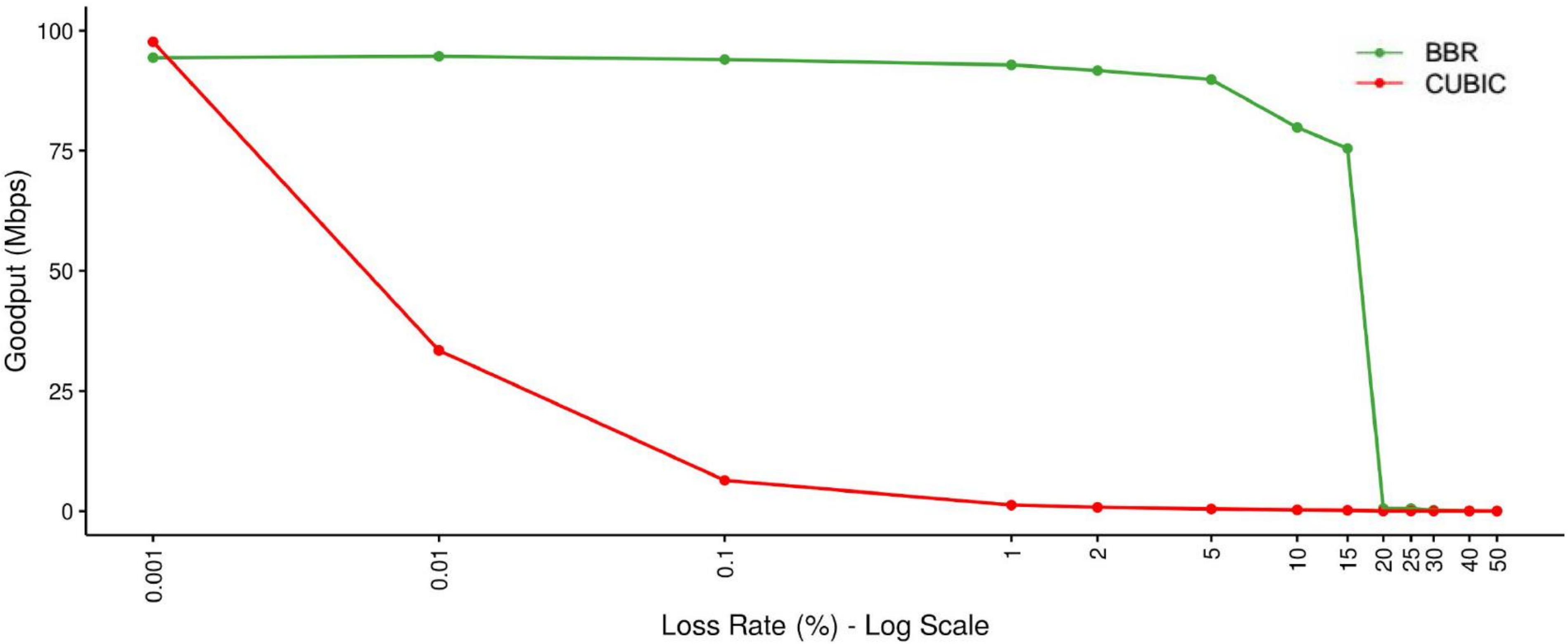
To See max BW, min RTT: Probe Both Sides of BDP



BBR: Core Design

- Model network path
  - Update estimates of max BW and min RTT on each ACK
- Control sending based on the model, to...
  - Sequentially probe max BW and min RTT, to feed the model samples
  - Pace near estimated BW, to reduce queues and loss
  - Vary pacing rate to keep inflight near BDP (for full pipe but small queue)

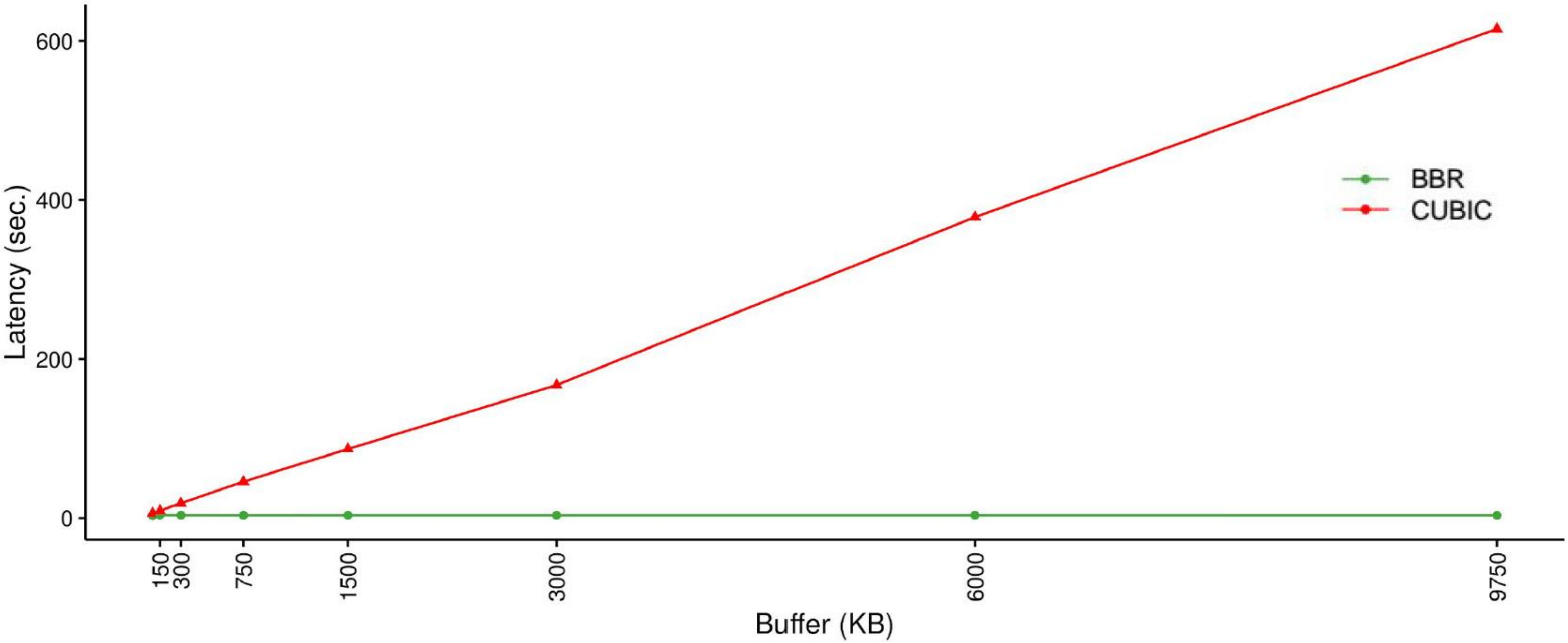
BBR: Fully Use Bandwidth Despite High Packet Loss



BBR vs CUBIC: synthetic bulk TCP test with 1 flow, bottleneck_bw 100Mbps, RTT 100ms

NOTE: Goodput is actual transmitted data.
(Throughput is whole transmitted data including packet headers and footers.)

BBR: Low Queue Delay Despite Bloated Buffers



BBR vs CUBIC: synthetic bulk TCP test with 8 flows, bottleneck_bw=128kbps, RTT=40ms

BBR: Deployment Experience

- Deployed on Google internal WANs; rolling out on google.com, YouTube
- On Google B4 WAN between datacenters (BBR used for vast majority of TCP)
  - RPCs 2-20x faster than CUBIC (8MB @ default QoS)
  - Bulk throughput up to 130x faster than CUBIC (@ default QoS)
- On Google.com
  - Faster web page downloads (particularly in developing world)
- On YouTube
  - Higher bandwidth
  - Less rebuffering
  - Lower delay: Cuts median RTT by 53% (by 80% in developing world)

BBR: Conclusion

- BBR: model-based congestion control
  - Goal: maximize bandwidth, then minimize queue
  - Result:
    - 100x bandwidth of CUBIC w/ big BDP, moderate loss (0.1% - 15%)
    - lower queuing latency with bufferbloated last mile links
- Next (WIP): reducing loss rates, improving fairness vs. loss-based CC
- For more information:
  - Reference code in Linux 4.9 TCP
  - https://lwn.net/Articles/701177/