TCP loss sensitivity analysis

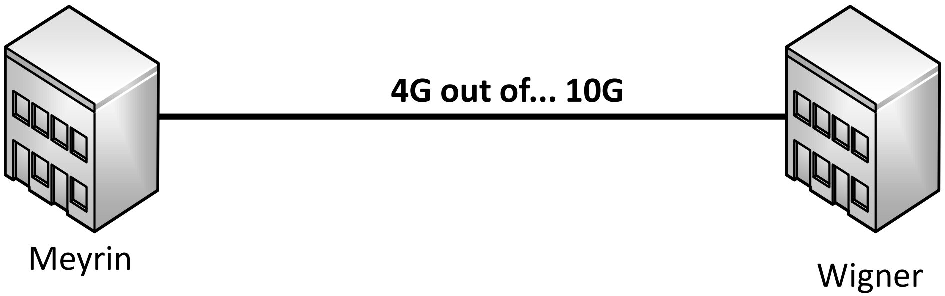
ADAM KRAJEWSKI, IT-CS-CE, CERN

The original presentation has been modified for the 2nd ATCF 2016

EDOARDO MARTELI, CERN

Original problem

IT-DB was backuping some data to Wigner CC. High transfer rate was required in order to avoid service degradation.



But the transfer rate was:

- Initial: ~**1G**
- After some tweaking: ~**4G**

The problem lies within TCP internals!

TCP algorithms (1)

The default TCP algorithm (**reno**) behaves poorly.

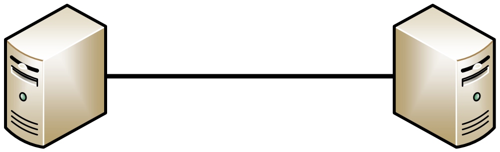
Other congestion control algorithms have been developed such as:

- BIC
- CUBIC
- Scalable
- Compund TCP

Each of them behaves differently and we want to see how they perform in our case...

Testbed

Two servers connected back-to-back.



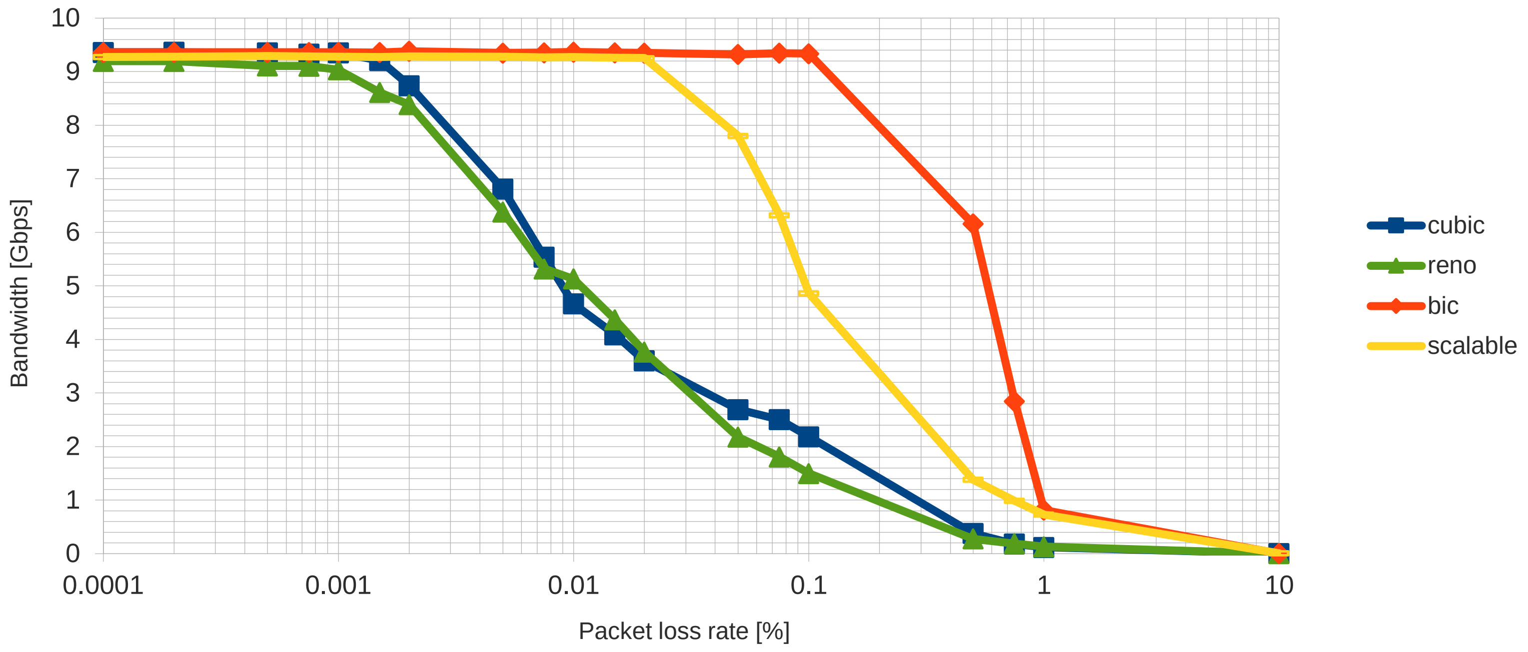
How the congestion control algorithm performance changes with **packet loss** and **delay** ?

Tested with **iperf3**.

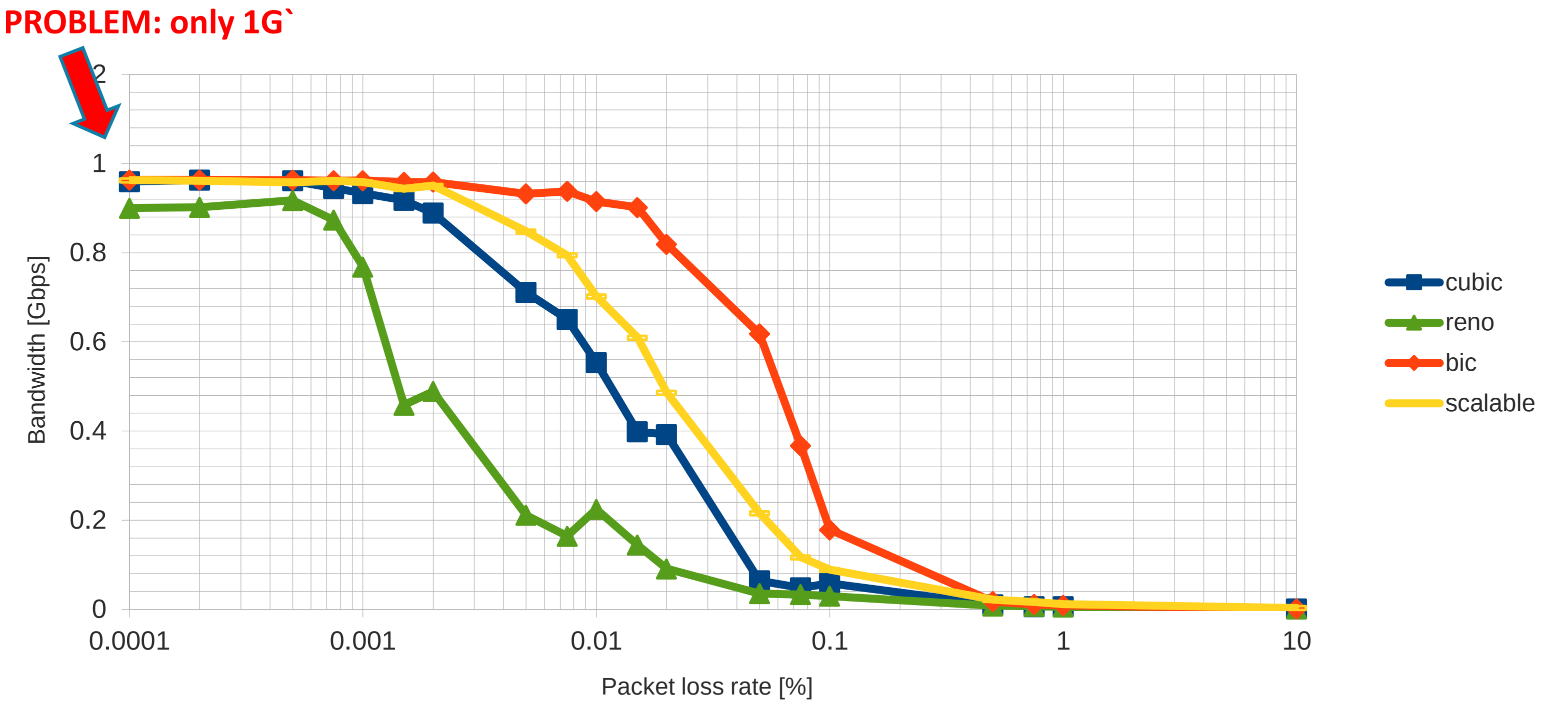
Packet loss and delay emulated using NetEm in Linux:

- `tc qdisc add dev eth2 root netem` **`loss 0.1 delay 12.5ms`**

0 ms delay, default settings



25ms delay, default settings



Growing the window (1)

We need...

$$W_{25} = B * RTT = 10Gbps * 25ms \approx 31.2\ MB$$

Default settings:

- net.core.rmem_max = 124K
- net.core.wmem_max = 124K
- net.ipv4.tcp_rmem = 4K 87K 4M
- net.ipv4.tcp_wmem = 4K 16K 4M
- net.core.netdev_max_backlog = 1K

Window can't grow bigger than maximum TCP send and receive buffers!

$$B_{max} = \frac{W_{max}}{RTT} = \frac{4MB}{25ms} \approx 1.28\ Gbps$$

WARNING: The **real OS** numbers are in pure bytes count so less readable.

Growing the window (2)

We tune **TCP settings** on **both server and client**.

Tuned settings:

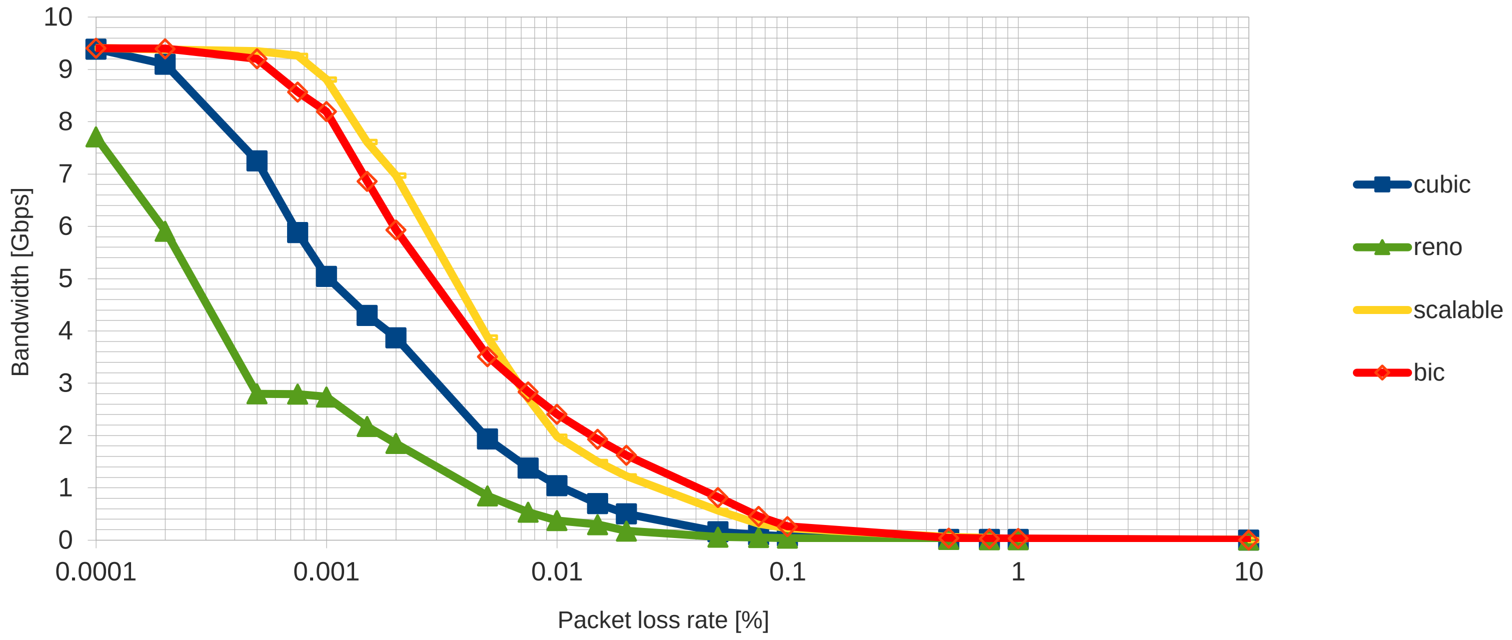
- net.core.rmem_max = 67M
- net.core.wmem_max = 67M
- net.ipv4.tcp_rmem = 4K 87K 67M
- net.ipv4.tcp_wmem = 4K 65K 67M
- net.core.netdev_max_backlog = 30K

Now it's fine

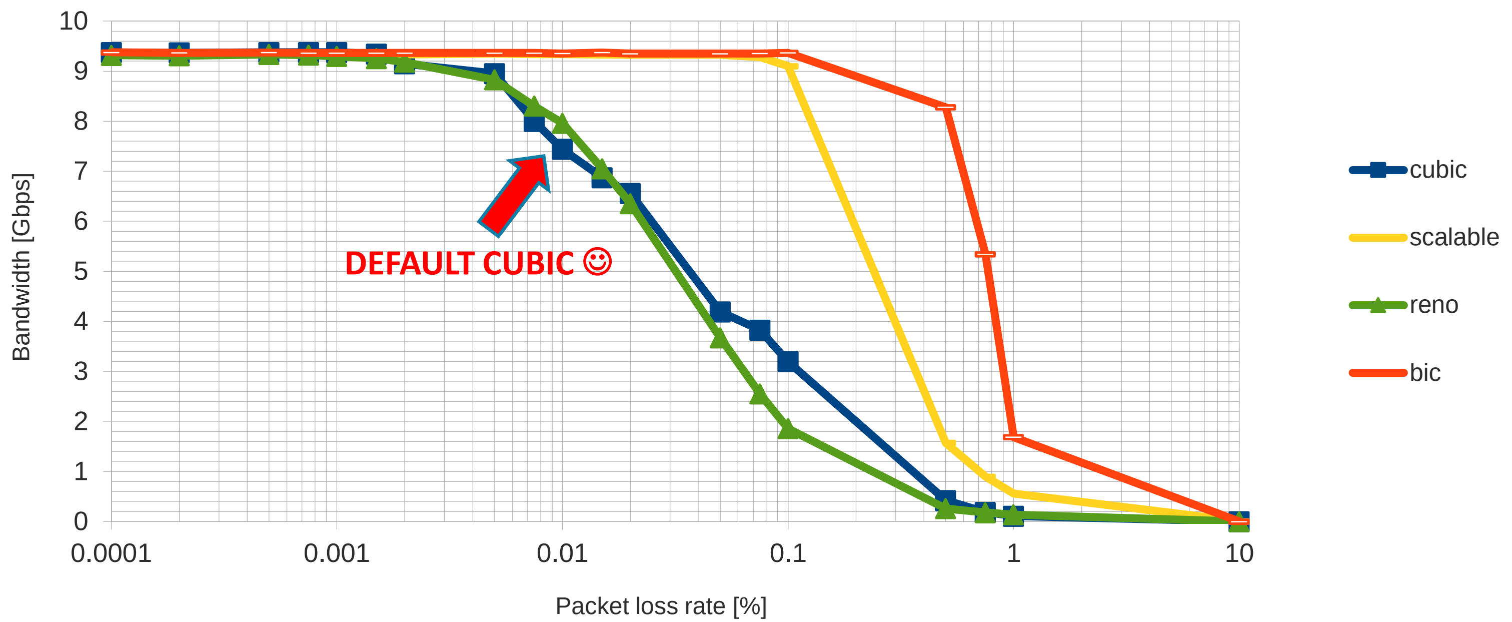
WARNING: TCP buffer size doesn't translate directly to window size because TCP uses some portion of its buffers to allocate operational data structures**. So the effective window size will be smaller than maximum buffer size set.**

Helpful link: https://fasterdata.es.net/host-tuning/linux/

25 ms delay, tuned settings

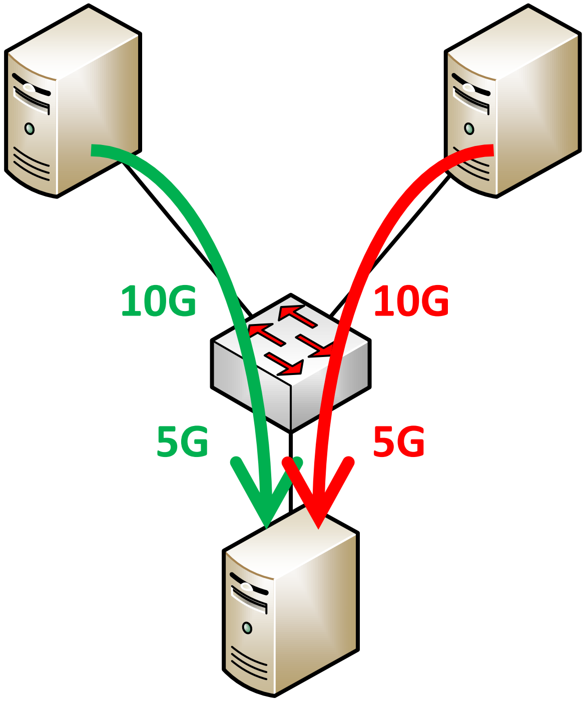


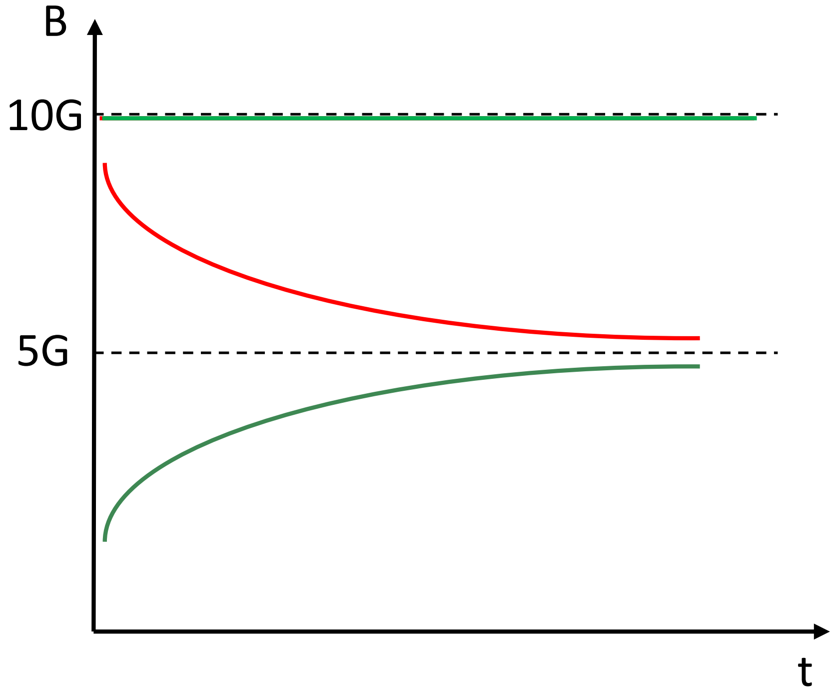
0 ms delay, tuned settings



Bandwith fairness

Goal: Each TCP flow should get a **fair share of available bandwidth**





Bandwith competition

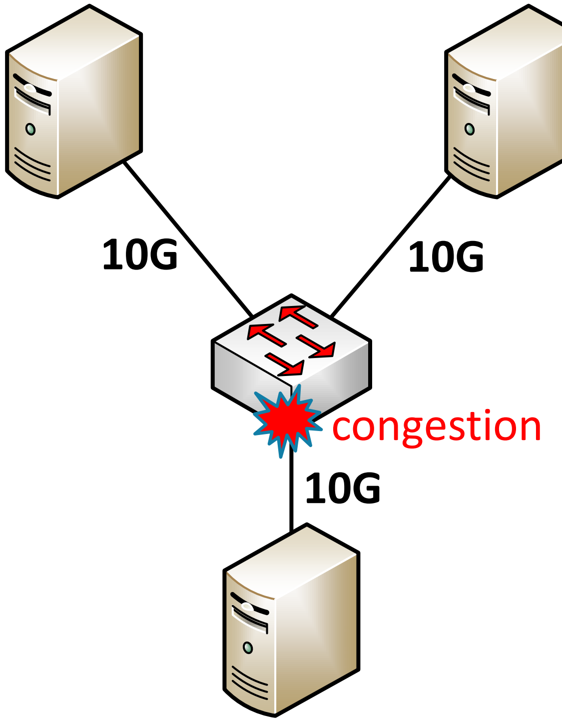
How do congestion control algorithms compete?

Tests were done for:

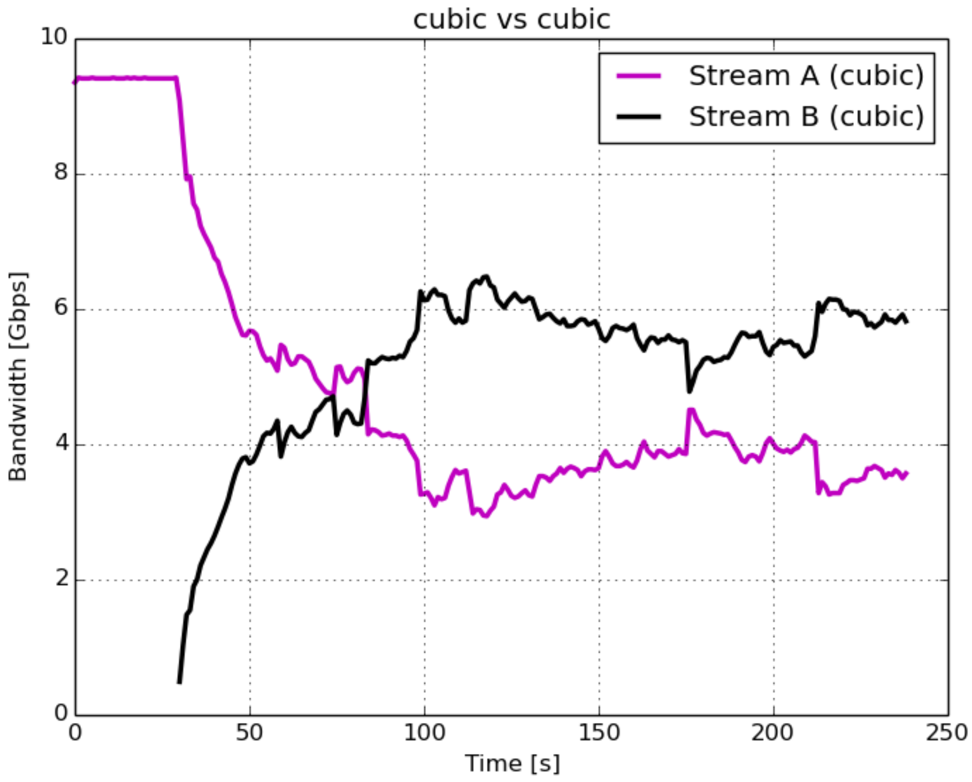
- Reno, BIC, CUBIC, Scalable
- 0ms and 25ms delay

Two cases:

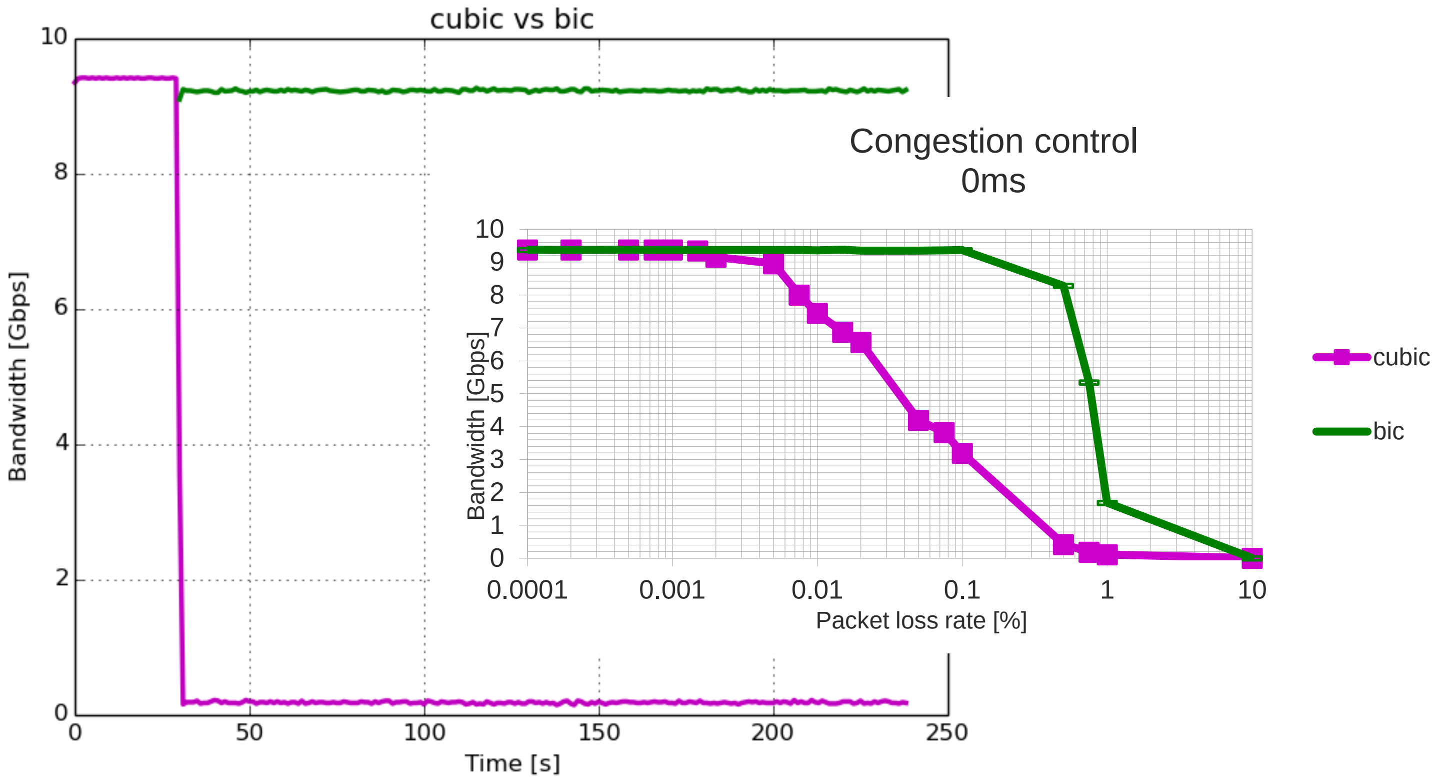
- 2 flows starting at the same time
- 1 flow delayed by 30s



cubic vs cubic + offset

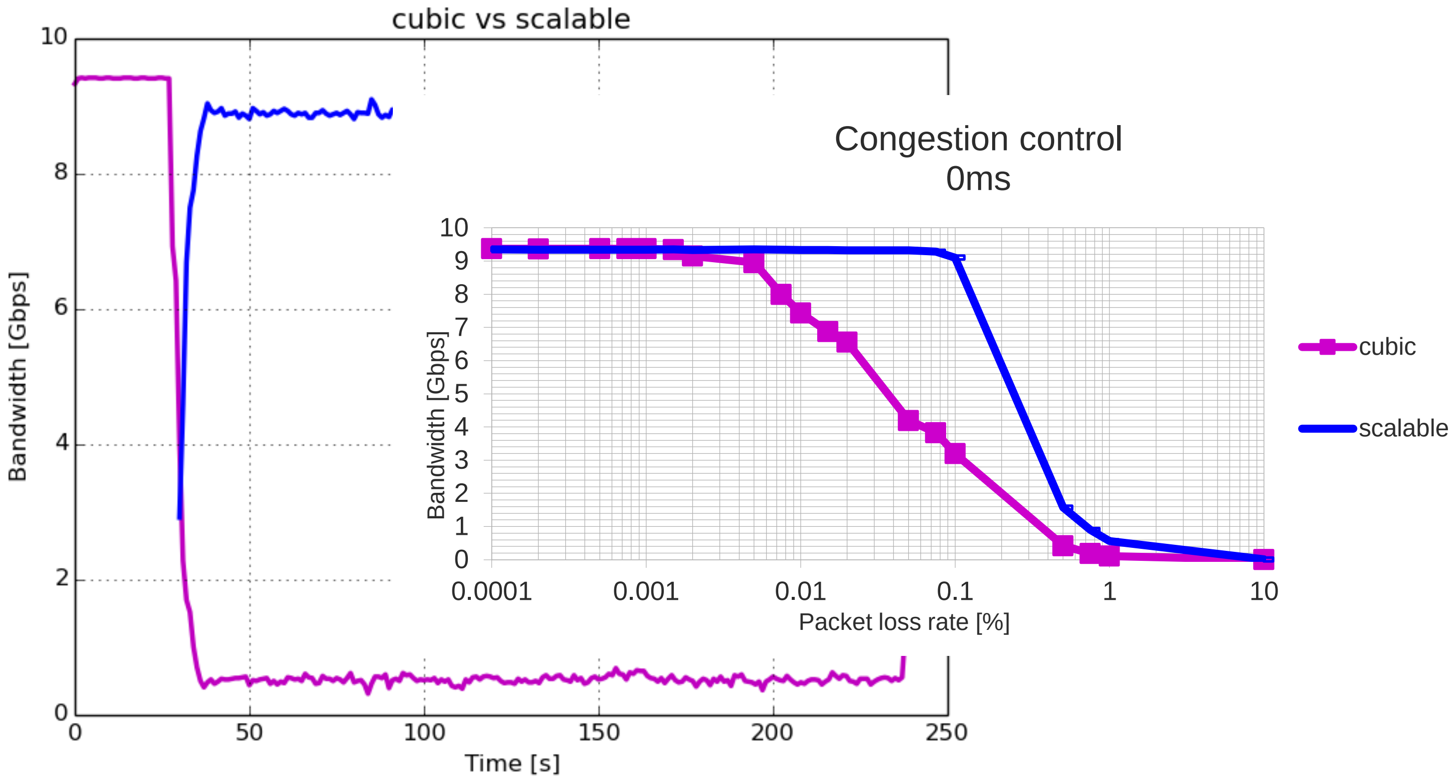


cubic vs bic

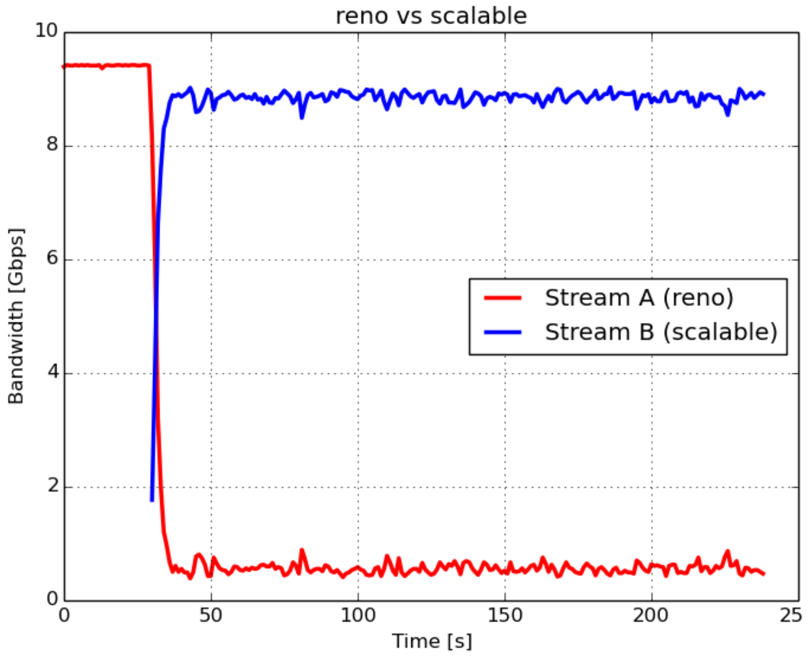


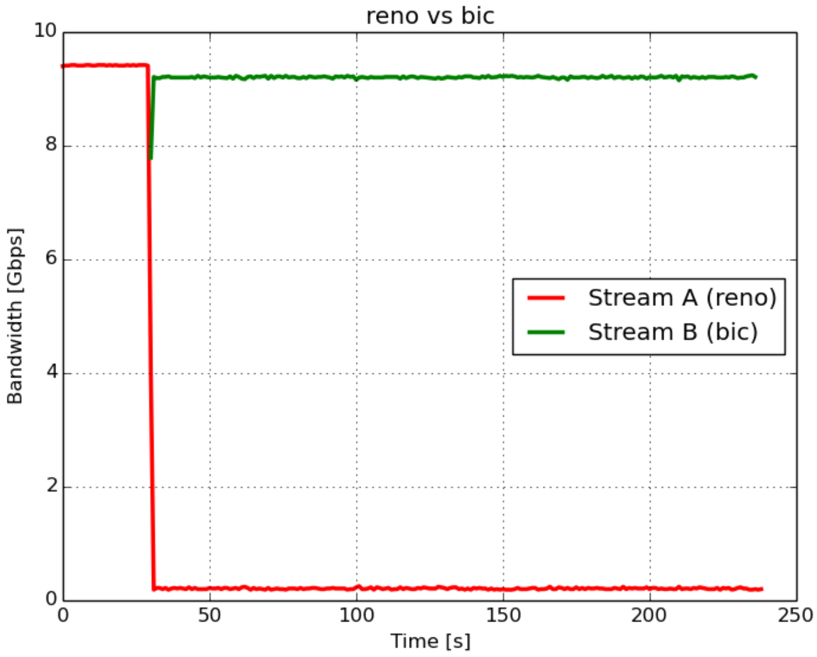
cubic vs bic
Bandwidth [Gbps]
Time [s]
0
50
100
150
200
250
0
2
4
6
8
10
Congestion control
0ms
Bandwidth [Gbps]
Packet loss rate [%]
0.0001
0.001
0.01
0.1
1
10
cubic
bic

cubic vs scalable

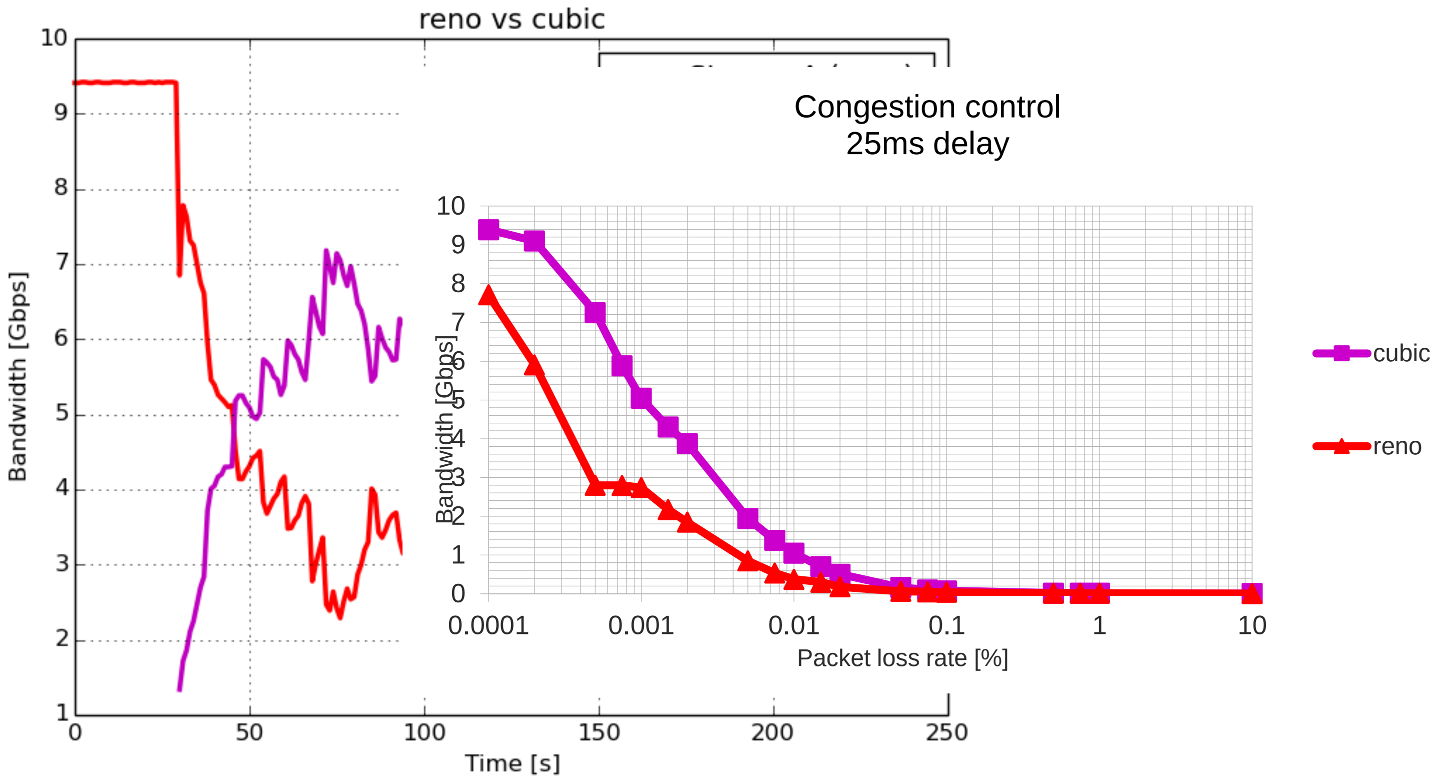


TCP (reno) friendliness





cubic vs reno



Why default cubic? (1)

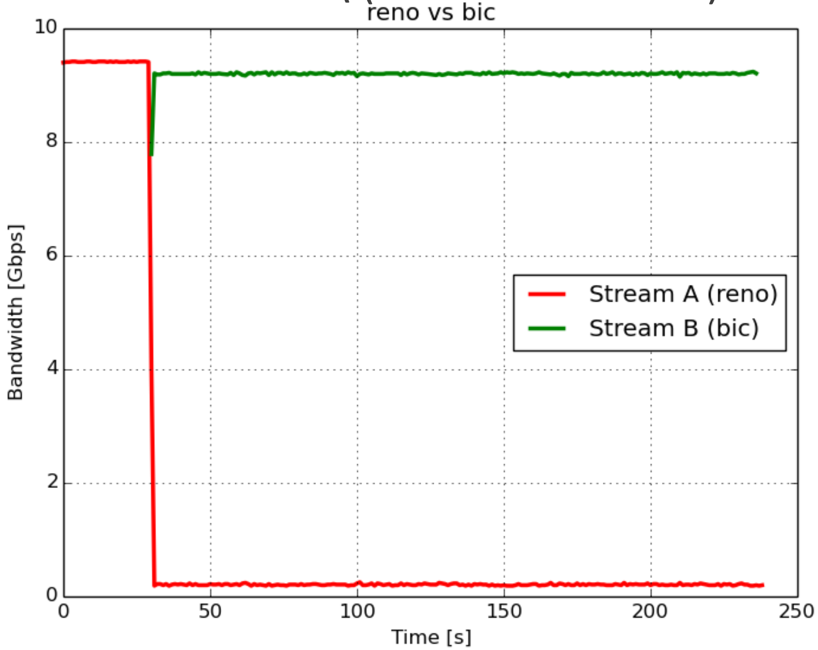
- **BIC** was the default Linux TCP congestion algorithm until kernel version 2.6.19

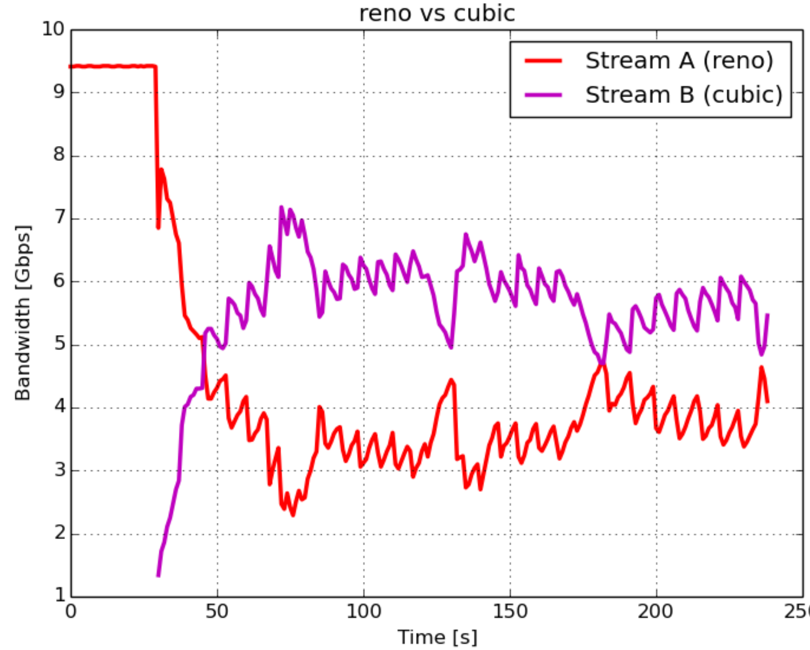
- It was changed to CUBIC in kernel 2.6.19 with commit message:
  - ”Change default congestion control used from BIC to the newer CUBIC which it the successor to BIC but has better properties over long delay links.”

- **Why?**

Why default cubic? (2)

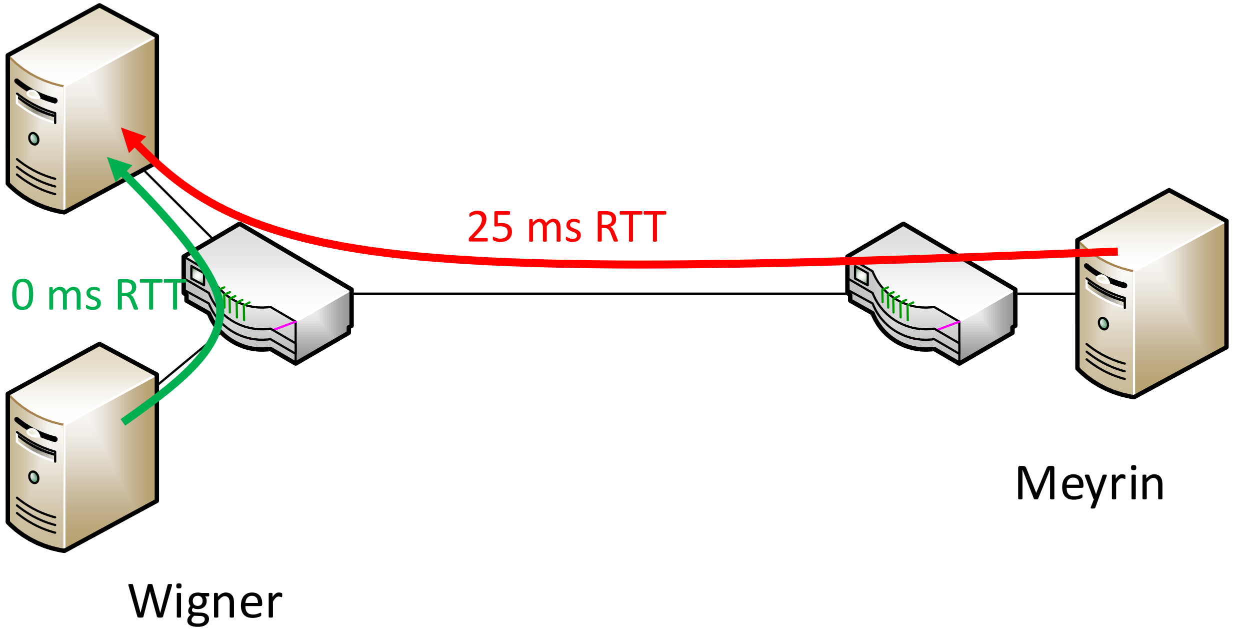
- It is **more friendly to other congestion algorithms**
  - to Reno...
  - to CTCP\ (Windows default)...





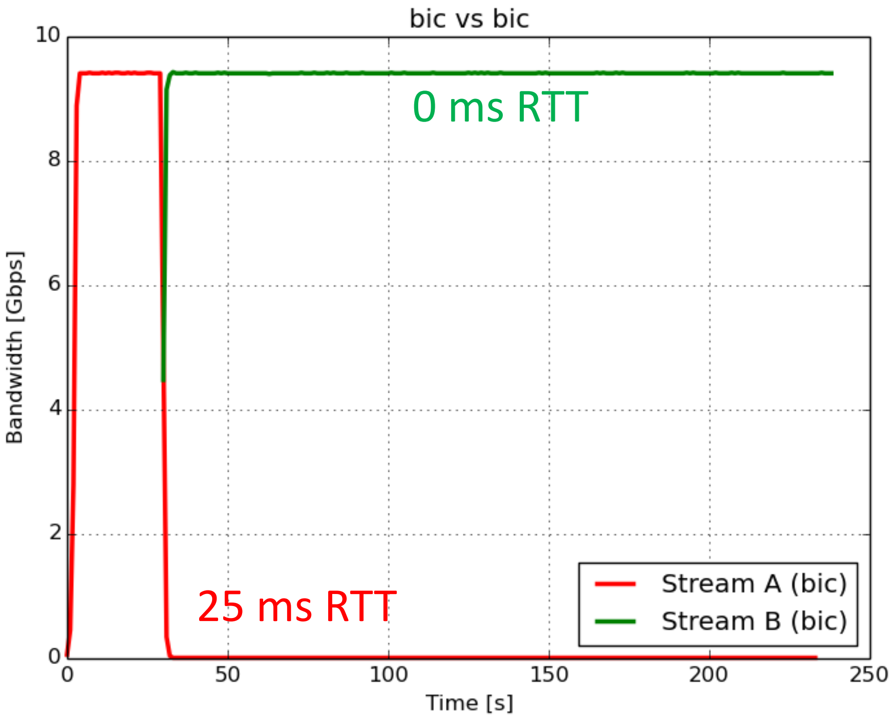
Why default cubic? (3)

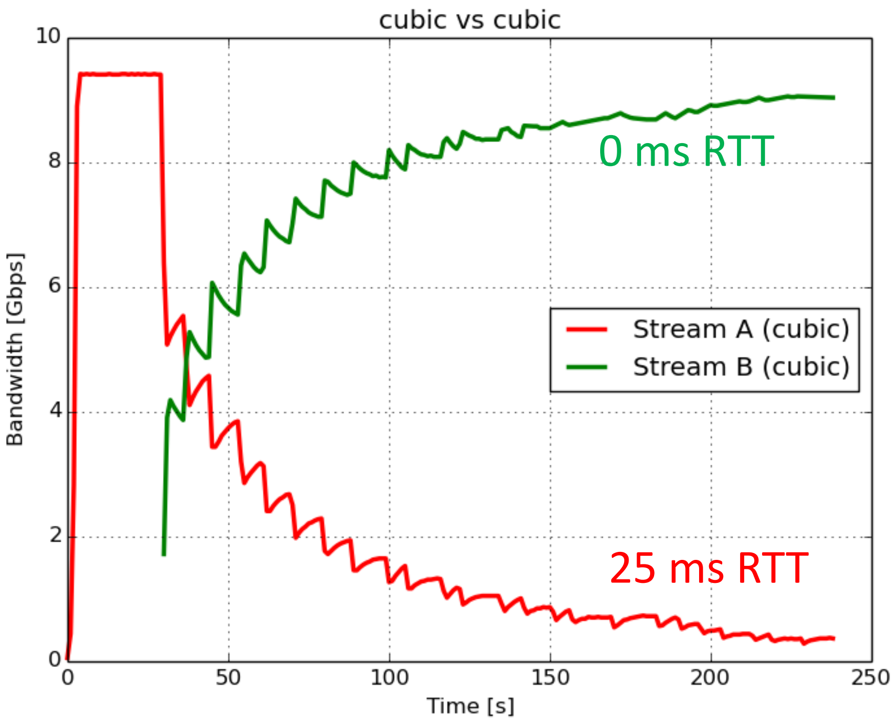
It has **better RTT fairness** properties



Why default cubic? (4)

RTT fairness test:





Setting TCP algorithms (1)

OS-wide setting (sysctl ...):

- net.ipv4.**tcp_congestion_control** = reno
- net.ipv4.**tcp_available_congestion_control** = cubic reno bic

Add new:

`# modprobe tcp_bic`

Local setting:

- setsockopt(), TCP_CONGESTION optname
- net.ipv4.**tcp_allowed_congestion_control** = cubic reno

Setting TCP algorithms (2)

OS-wide settings:

`C:\>` netsh interface tcp show global

- `Add-On Congestion Control Provider: ctcp`

Enabled by default in Windows Server 2008 and newer.

Needs to be enabled manually **on client systems:**

- **Windows 7:** C:\>netsh interface tcp set global congestionprovider=ctcp
- **Windows 8/8.1 [Powershell]:** Set-NetTCPSetting –CongestionProvider ctcp

TCP loss sensitivity analysis

ADAM KRAJEWSKI, IT-CS-CE