Peeking at the bottleneck:
bufferbloat prevention congestion control

Alejandro Popovsky <apopov@palermo.edu>
Universidad de Palermo

IETF 95 – ICCRG - April 2016 - Buenos Aires

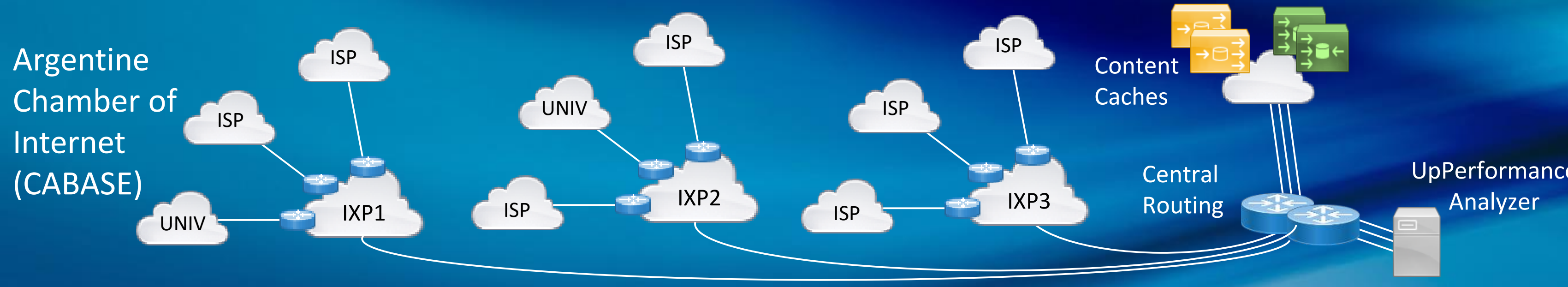
Goals:

1. Bufferbloat mitigation

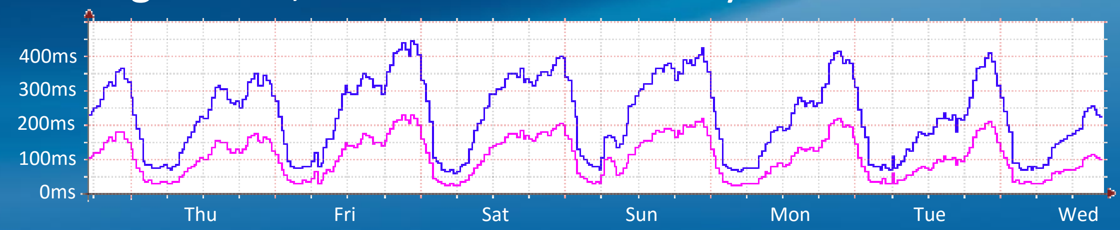
2. Available capacity (fair) sharing

Universidad de Palermo

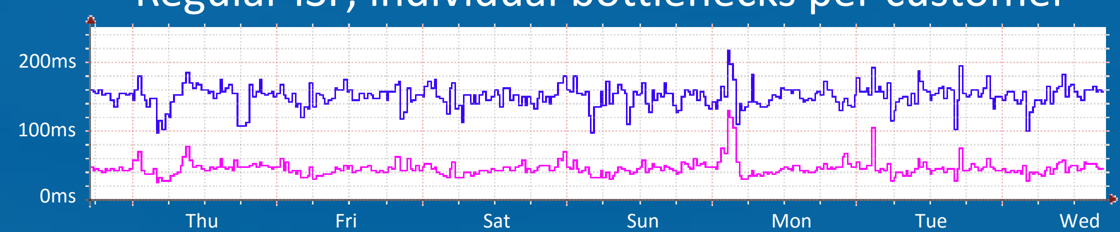
Bufferbloat in Regular ISPs:



Regular ISP, bottlenecks shared by several customers



Regular ISP, individual bottlenecks per customer

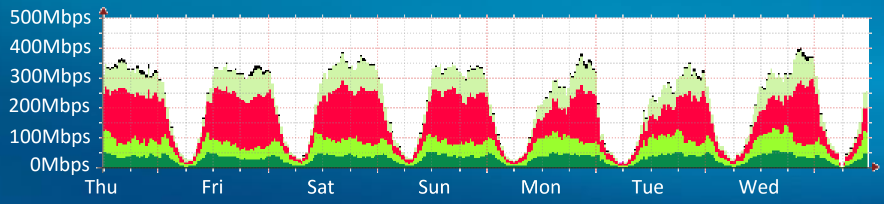


Average TCP minimum RTT

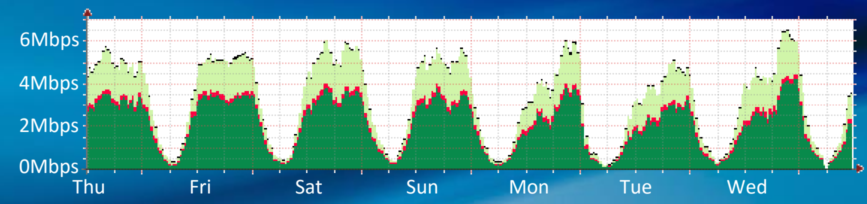
Average TCP RTT

Traffic profiles for regular ISPs:

Google Cache to Regular ISP traffic

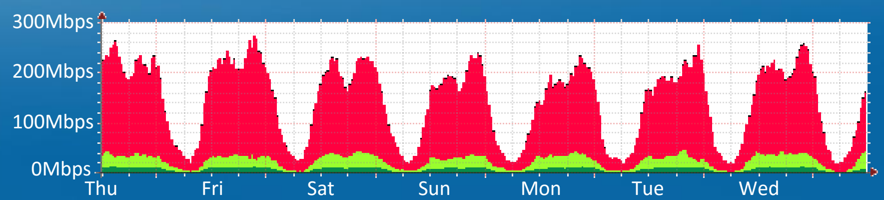


Downstream

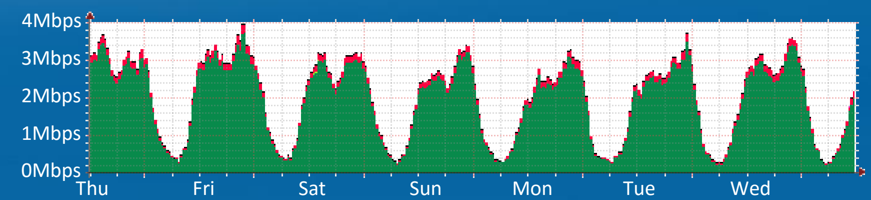


Upstream

Akamai Cache to Regular ISP traffic



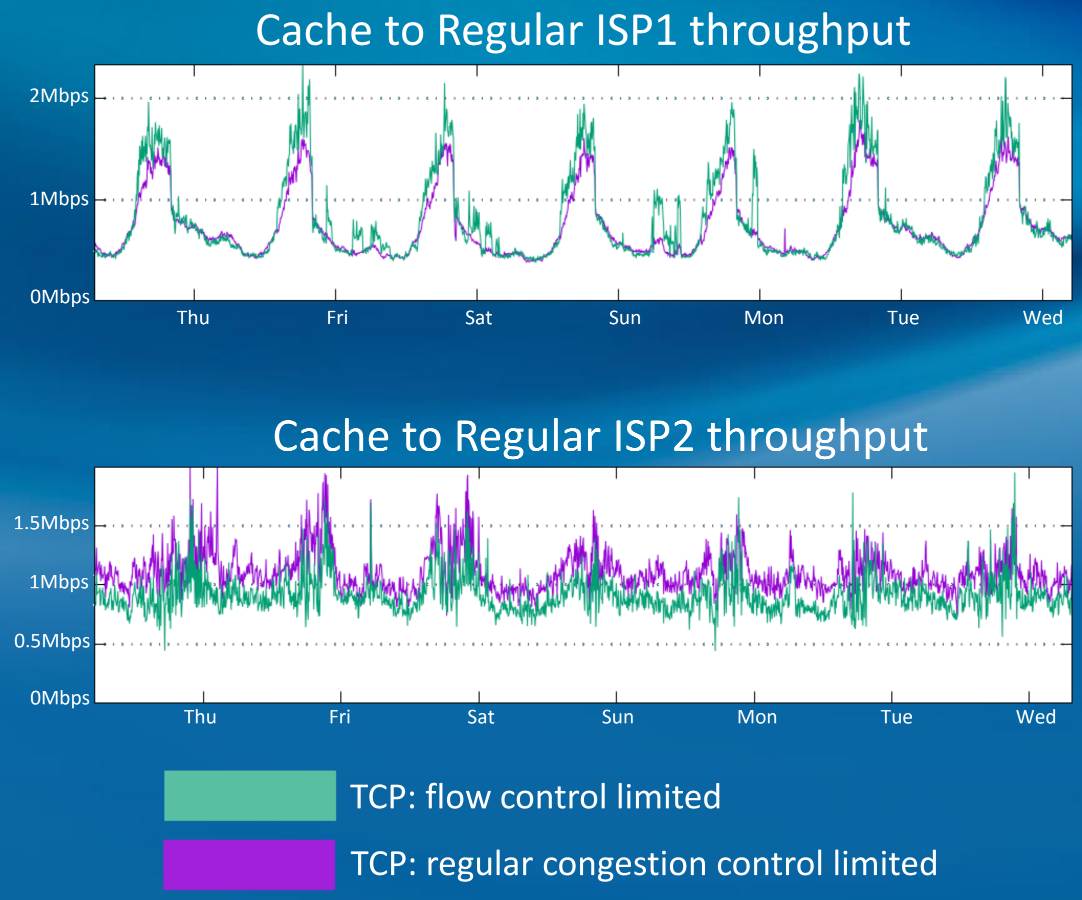
Downstream



Upstream

- TCP: congestion control limited
- ??? ← TCP: flow control limited (mostly no rwin scaling)
- TCP: data generation rate limited
- UDP: inelastic, or user space congestion control

Congestion Control and Flow control for Regular ISPs



Congestion control and flow control are currently getting similar throughput !!

Opportunity

- Content servers and ISPs not currently fighting bufferbloat caused by TCP
  - Content servers: not using sender side congestion control
  - Local ISPs: not using AQM in rate limiting devices.
- End users could still fight it using recevier side congestion control

Bottleneck feedback

Goal: estimate the

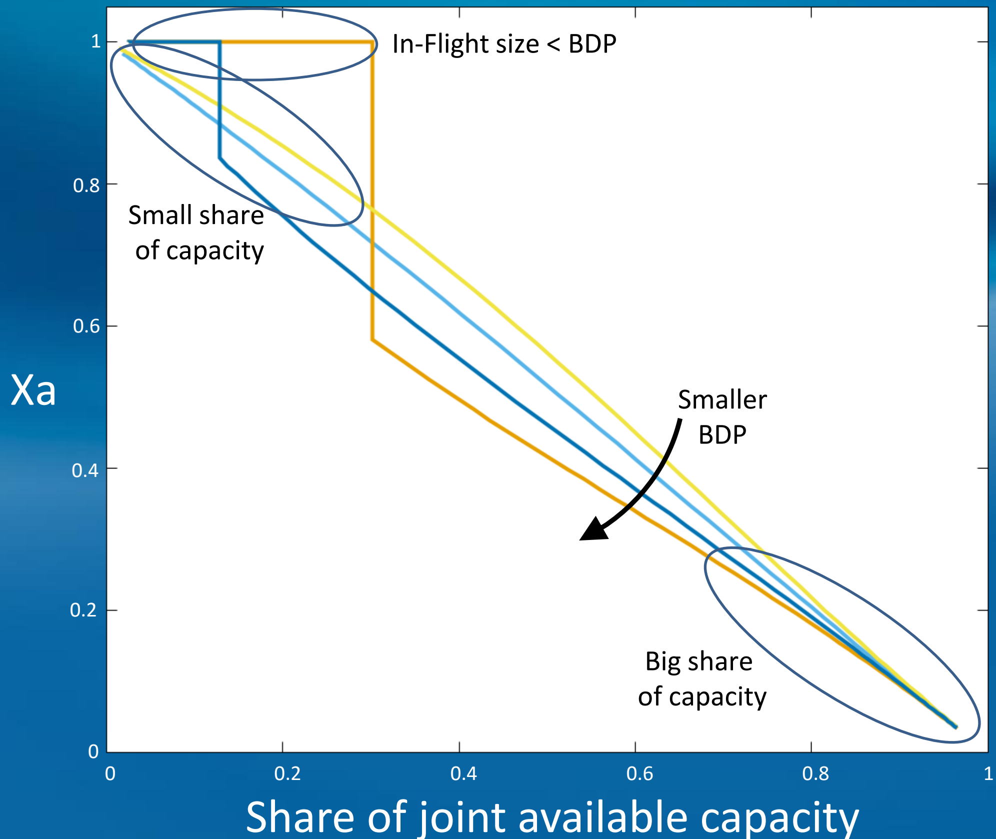
share of the joint available capacity

Proposed variable:

Proportional rate (Ra) response to In-flight size (Ca) variations

$$Xa = \left(\frac{\Delta Ra}{\Delta Ca}\right)\left(\frac{Ca}{Ra}\right)$$

Estimating Bottleneck share with Xa



Exclusive user of bottleneck:

In-Flight size<BDP => Xa=1

In-Flight size>BDP => Xa=0

Shared bottleneck:

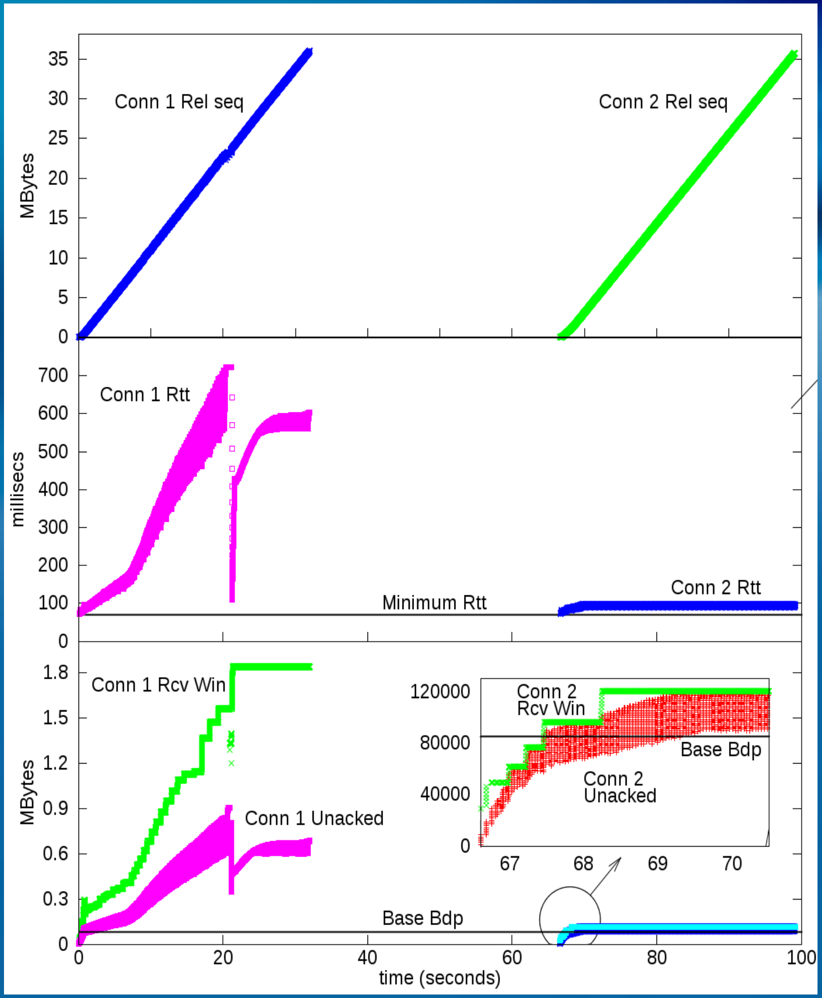
Xa ≈ (1 – share of capacity)

Current Algorithm

- Grow receive window only on Xa above threshold.
- Decrease receive window when detecting other connections leaving bottleneck
- Consider other connections induced noise in Xa measurement
- If possible prevent bufferbloat, else revert to regular behavior
- Aim for fair sharing, avoiding starvation

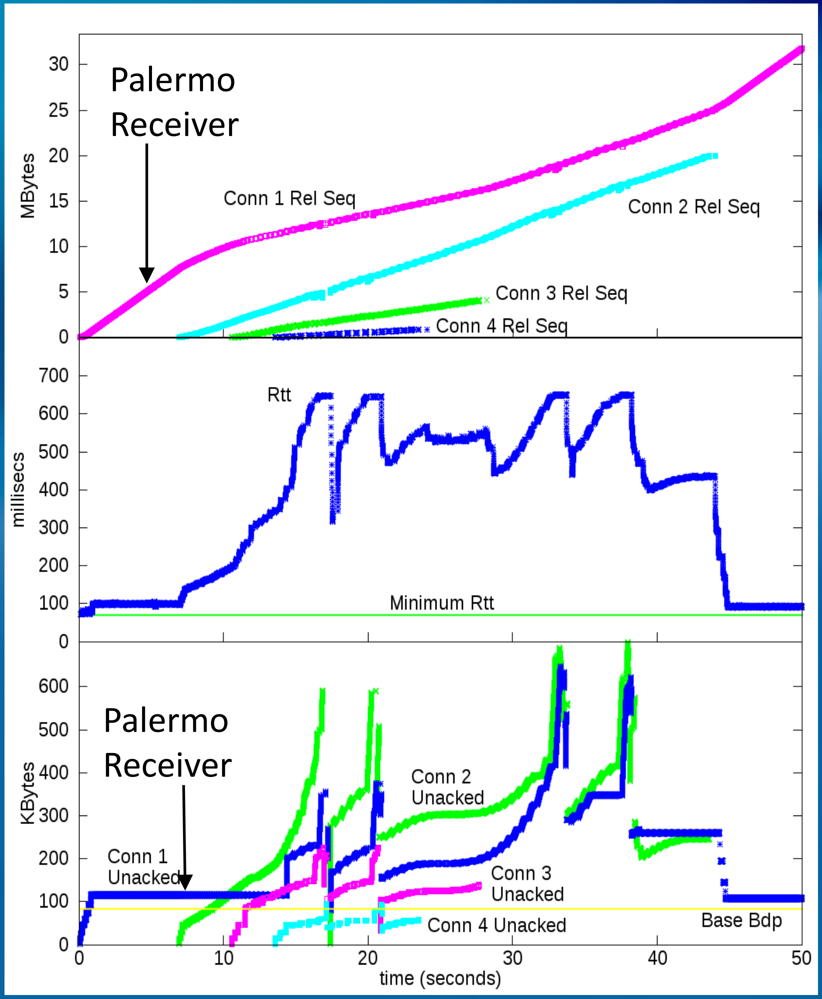
Congestion Control comparison

Cubic Sender
Regular DRS receiver

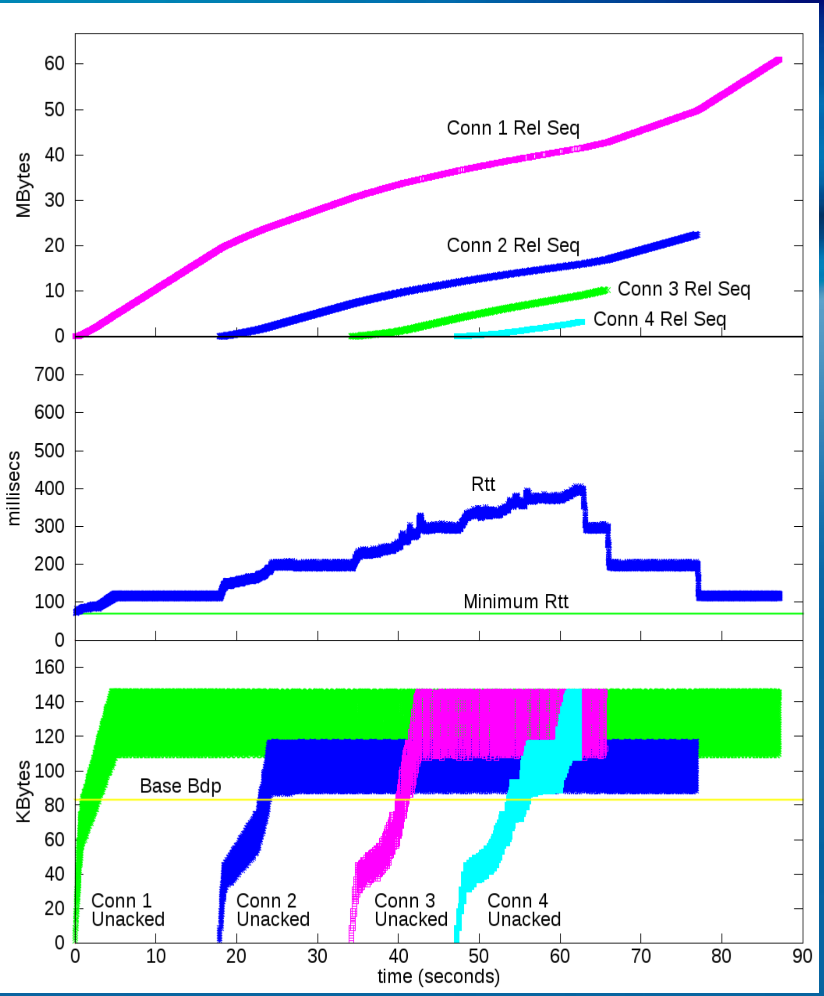


Cubic Sender
Palermo receiver

Sharing the bottleneck with regular connections



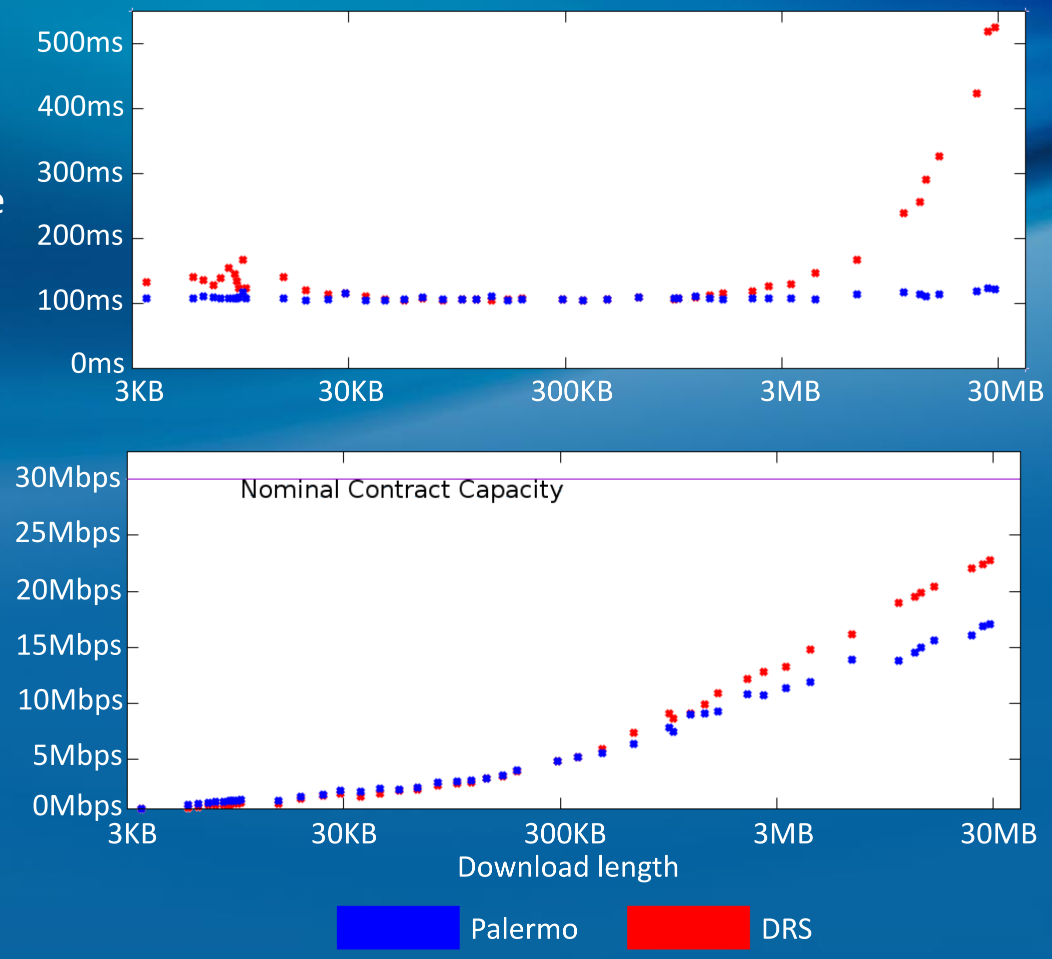
Sharing the bottleneck with well behaved connections



Performance Comparison

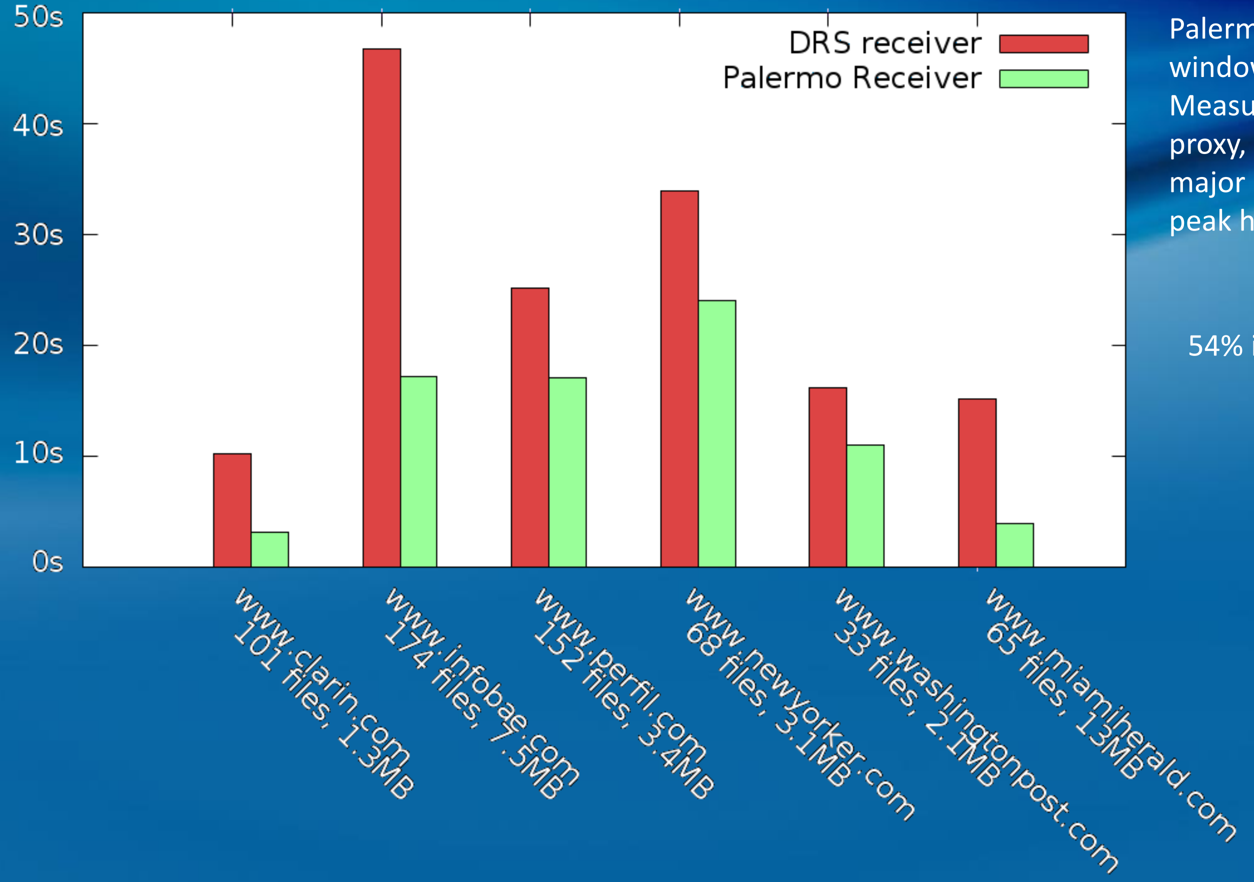
Average Round trip time

Average Throughput



Palermo versus DRS receiver window control.
Measurements at university proxy, averaging over several Centos mirrors

Performance Comparison for Transaction oriented connections



Total download time

Palermo versus DRS receiver window control. Measurements at university proxy, Downloading from major newspapers during peak hours.

54% improvement

Conclusions and Future Work

- Proposed algorithm:
  - Valid option to use at hosts and organization proxies to improve end user experience on incoming traffic.
- Next:
  - Explore robustness and variants
  - Develop sender side version
  - Upcoming publication

For more information, or Linux kernel patches with the algorithm:
apopov@palermo.edu