Global Synchronization Protection for Packet Queues

draft-lauten-aqm-gsp-00

Wolfram Lautenschlaeger

Wolfram.Lautenschlaeger@alcatel-lucent.com

IETF 89, AQM WG

Why avoid Global Synchronization?

- Today:
  - Bandwidth * Delay product rule for buffer dimensioning at all network levels
  - For 100% capacity utilization by TCP traffic

- Goal: Reduce queue size from BDP per link towards BDP per flow
  - With *N* bottleneck sharing TCP flows

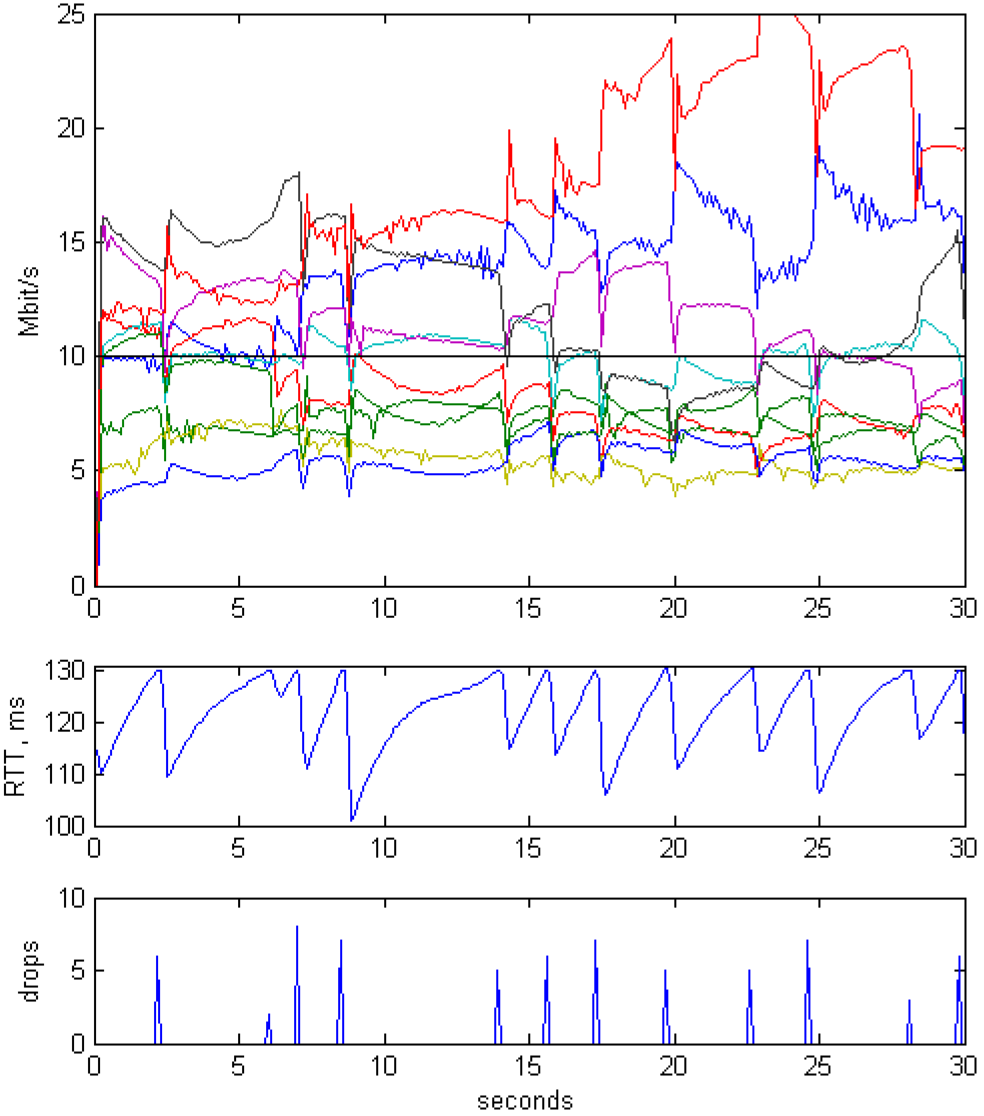
$$BDP \Rightarrow \frac{BDP}{\sqrt{N}} \Rightarrow \frac{BDP}{N}$$

(realistic) (ideal)

- Not a Goal: Reduce queue size for a particular single flow (*N*=1)

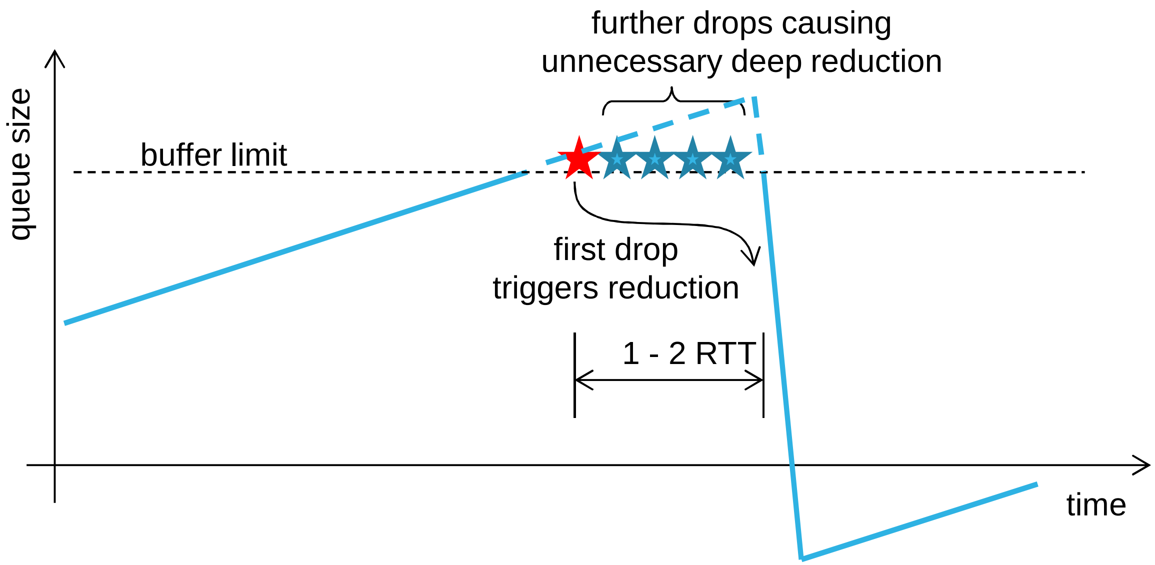
The Effect of Global Synchronization

- 10 TCP flows crossing a
- 100Mbit/s bottleneck
- RTT_0 = 100ms
- tail drop queue 375kB (30ms)
  - Linux kernel 3.5
  - CUBIC, SACK, delayed ACK
  - 10G Ethernet network



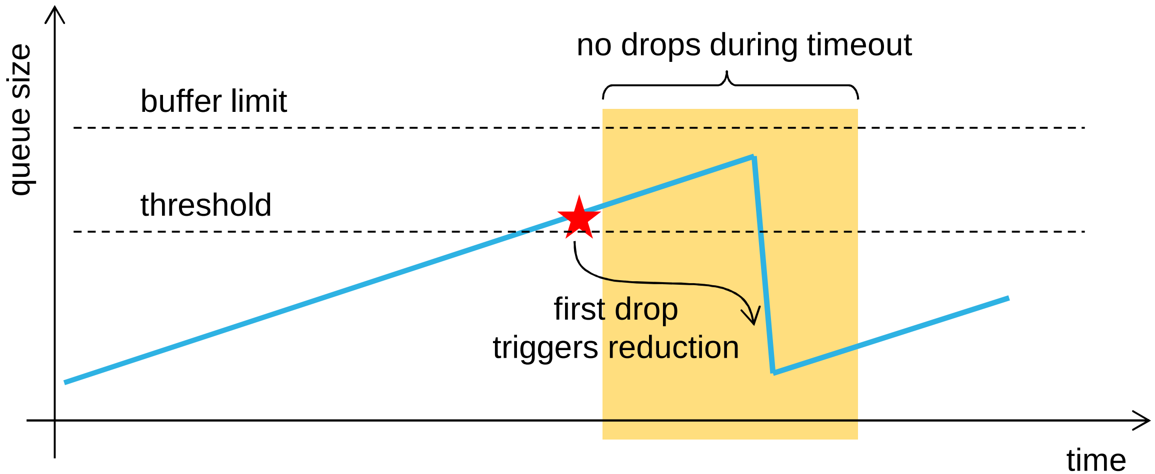
- ➢ Simultaneous packet drops affecting ≈half of the flows
- ➢ Simultaneous CWND reductions

Why? How to avoid?



After first drop:

- CWNDs continue to grow
- Queue continues to grow
- Reduction not earlier than after one RTT
- ≈ N/2 excess packets
  - N – number of flows
  - Mathematically approved for Reno, del.Ack



Basic Algorithm:

- set threshold below buffer limit
- drop if threshold is violated
- prevent further drops for at least timeout ≥ 1…2 RTT
- letting the queue grow beyond threshold until reduction occurs

Limitations of basic algorithm

Overload

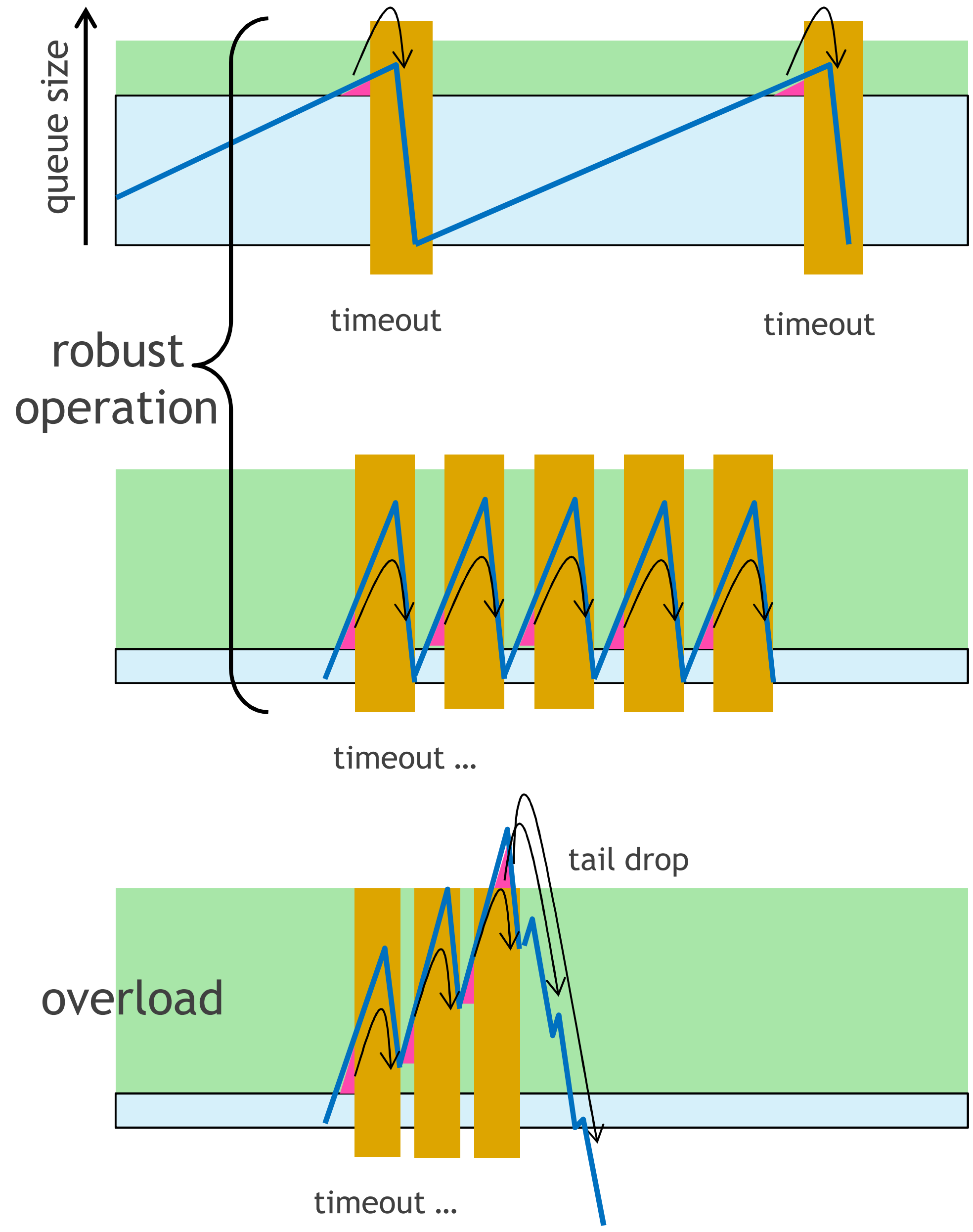
- If, during the timeout, the queue increases larger than subsequent reduction

Why?

- large N – too many flows
- small RTT – dominating RTT is smaller than expected
- aggressive TCP flavor

Solution:

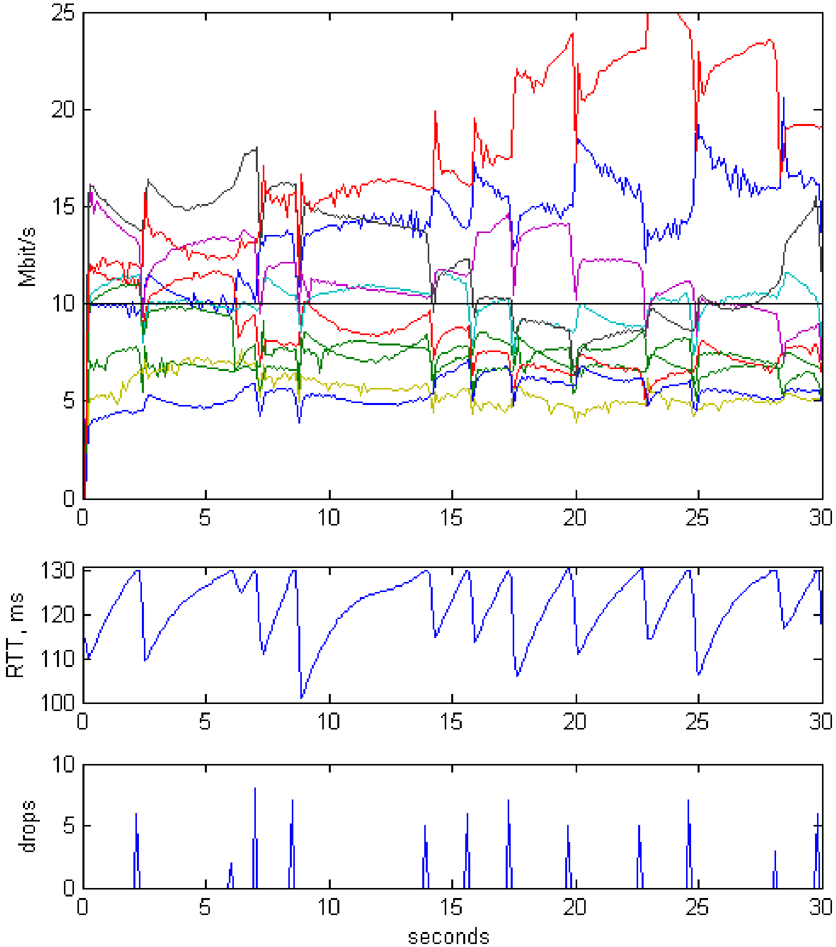
- adaptive timeout reduction
- criterion: cumulative time above threshold exceeds cumulative time below



Experiments

10 flows in 100Mbit/s

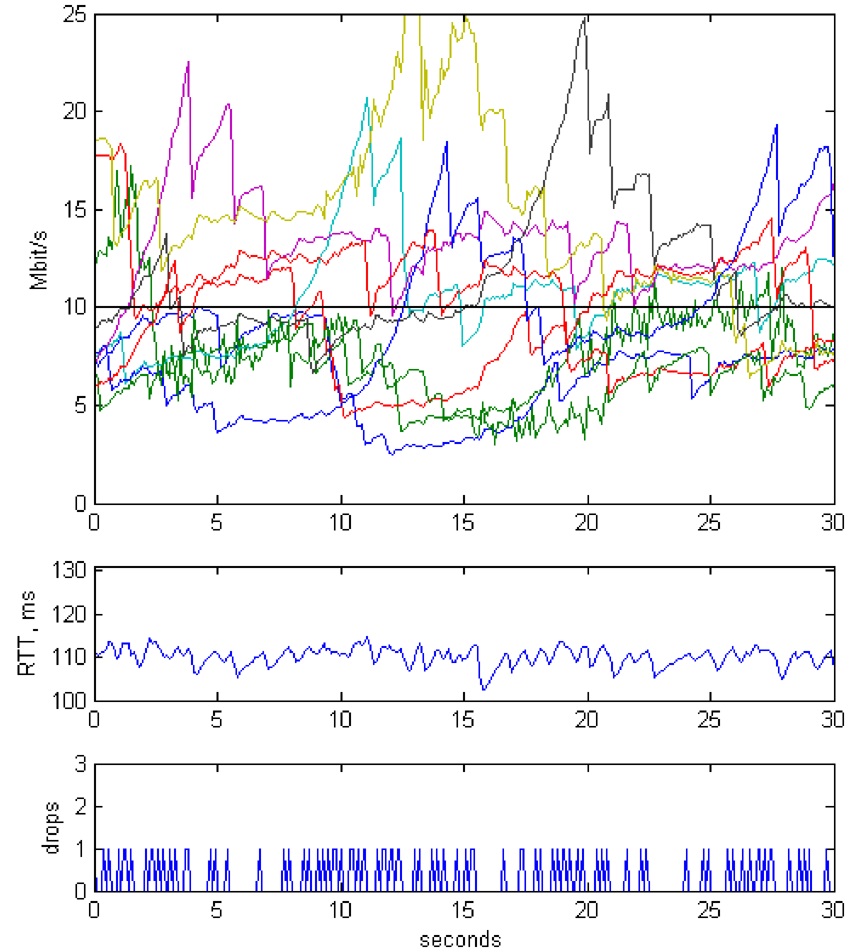
tail drop



drop bursts

(total 67 drops in 30 sec)

GSP



distributed single drops

(total 82 drops in 30 sec)

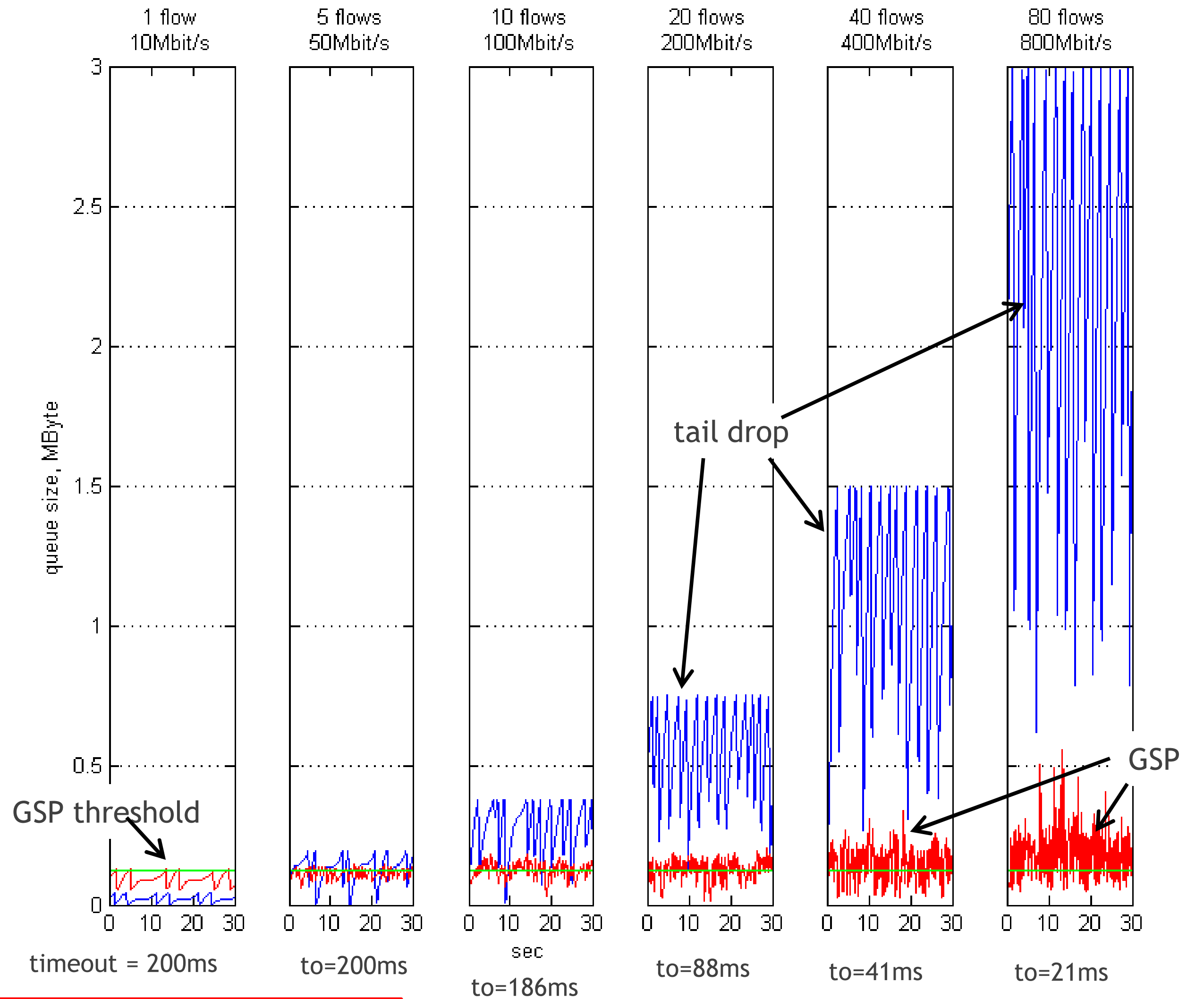
Experiments
different flow numbers

Settings
- RTT=100ms
- N flows (N=1…80)
- ≈10Mbit/s per flow
- CUBIC, delayed ACK

- tail drop at 1/3 BDP

- GSP threshold at (const) 125 kB

Results
- 100% link utilization (all)
- drop ratio 2.7 – 3.6e-4
- drop interval (per flow) ≈4sec

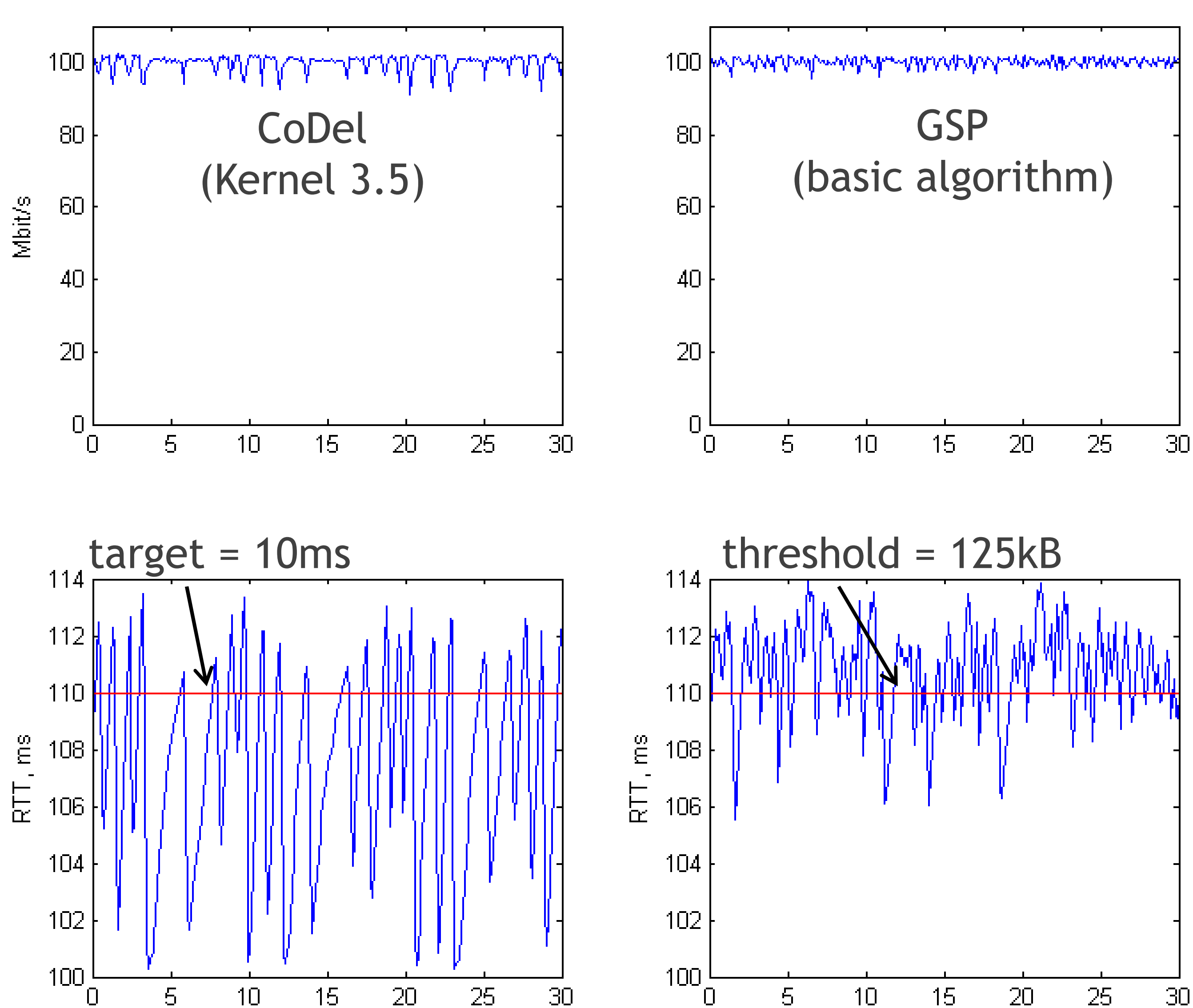


	tail drop	GSP
queue size:	2.5MB	500kB
queuing jitter:	25ms	5ms

(at 80 flows)

Comparison

10 flows
100Mbit/s
100ms RTT_0



Conclusion

Why GSP? Yet another AQM?

- Minimalistic extension to tail drop
- Deterministic drop decisions – randomness comes from traffic only
- Basic algorithm is memoryless – the queue is the memory – robust to quickly changing conditions

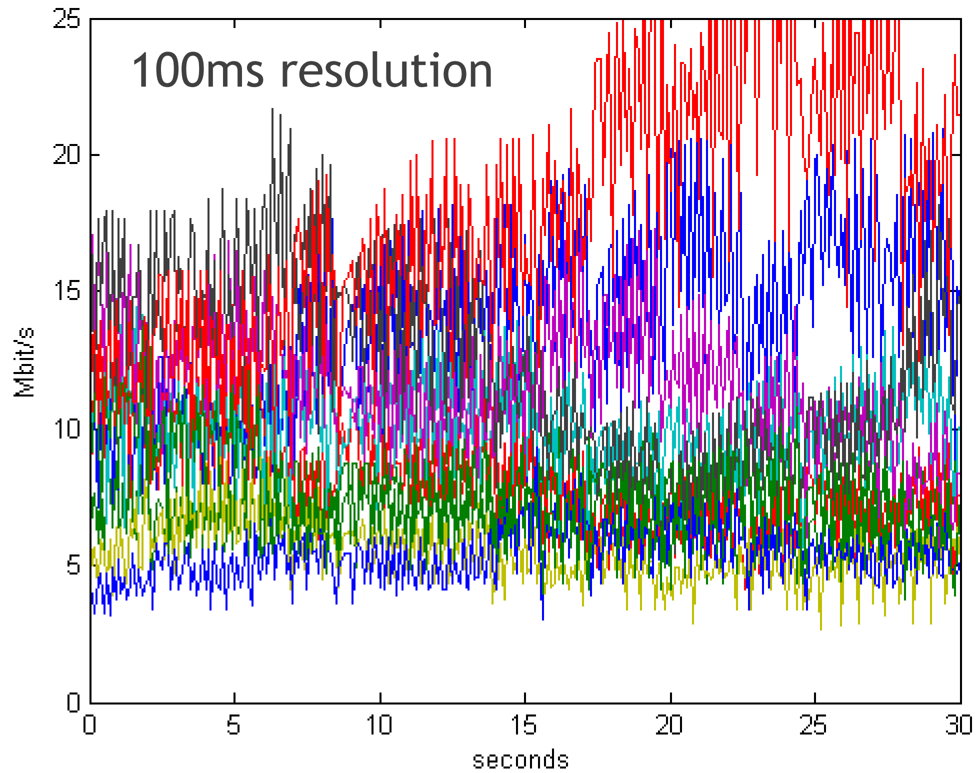
To do

- Tests at large RTT spreading
- Tests with flow renewal process instead of static flow allocation
- Refined timeout adaptation

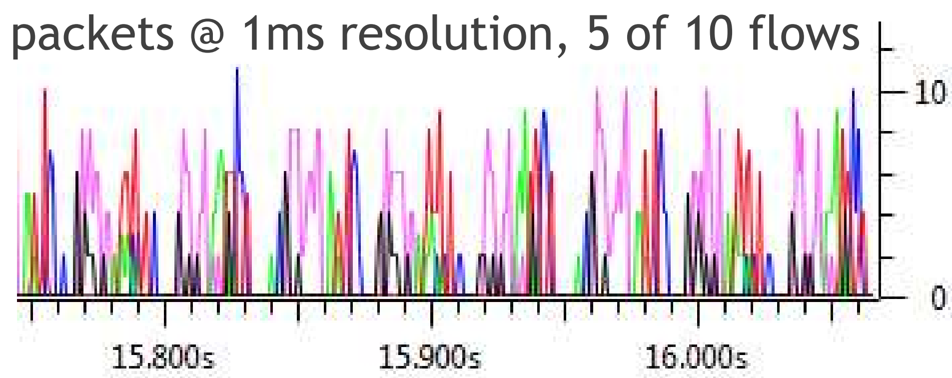
draft-lauten-aqm-gsp-00

Backup

Raw traces

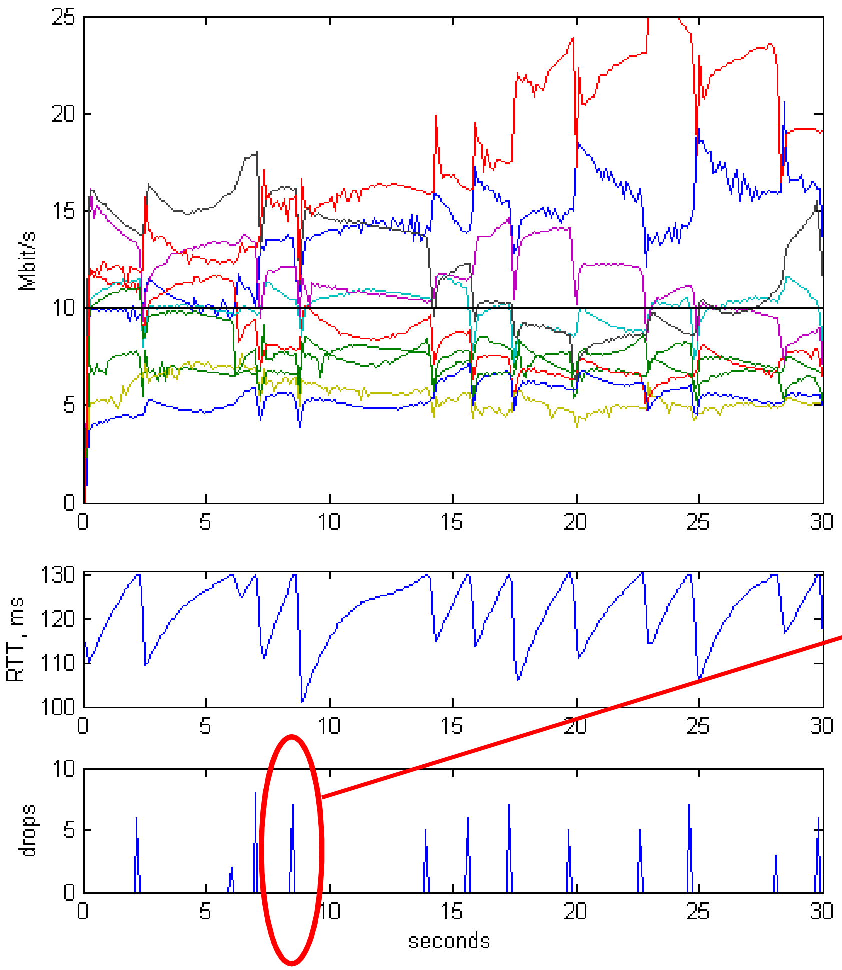


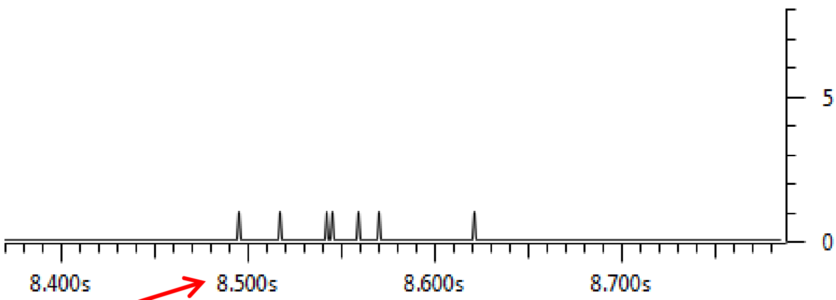
100ms resolution
Mbit/s
seconds



packets @ 1ms resolution, 5 of 10 flows
15.800s
15.900s
16.000s

Microstructure of Loss Burst





130ms
1000 packets passed
7 packets lost