Advanced Computer Networks

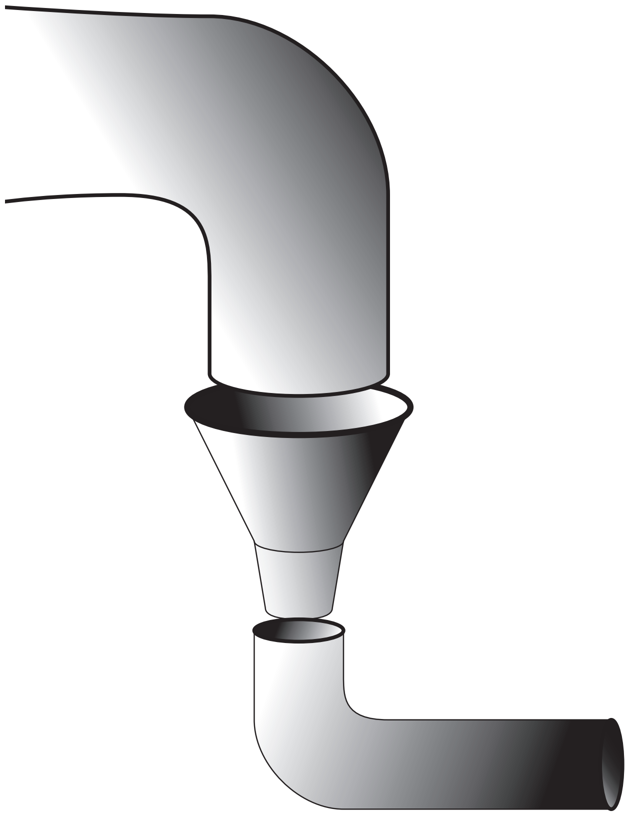
Active Queue Management

Prof. Andrzej Duda
duda@imag.fr

http://duda.imag.fr

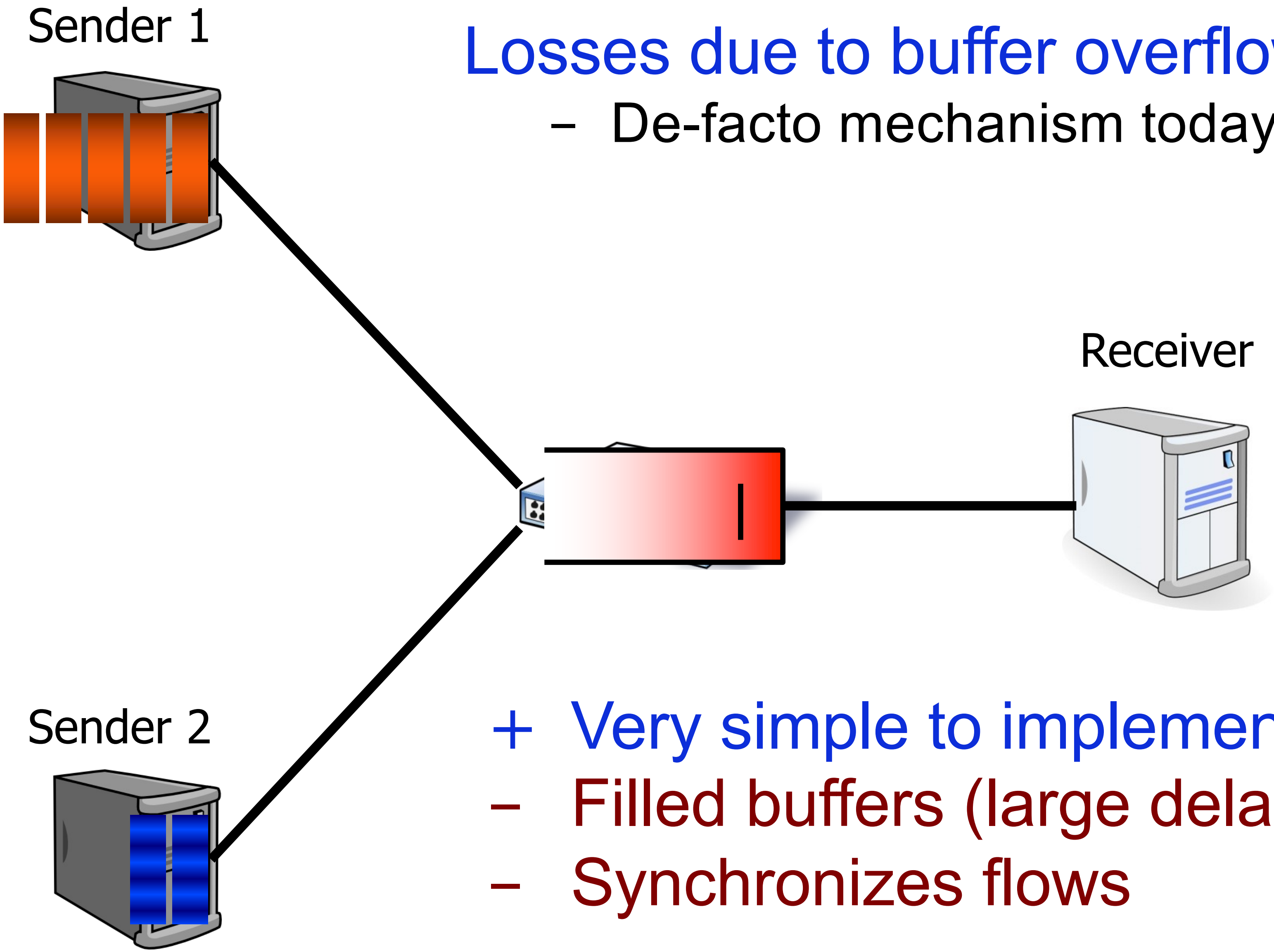
Contents

- AQM
  - Bufferbloat
  - RED
  - CoDel
  - FQ-CoDel



Active Queue Management (AQM)

Drop-tail queues



Losses due to buffer overflow

– De-facto mechanism today

\+ Very simple to implement

– Filled buffers (large delay)

– Synchronizes flows

How big should router buffers be?

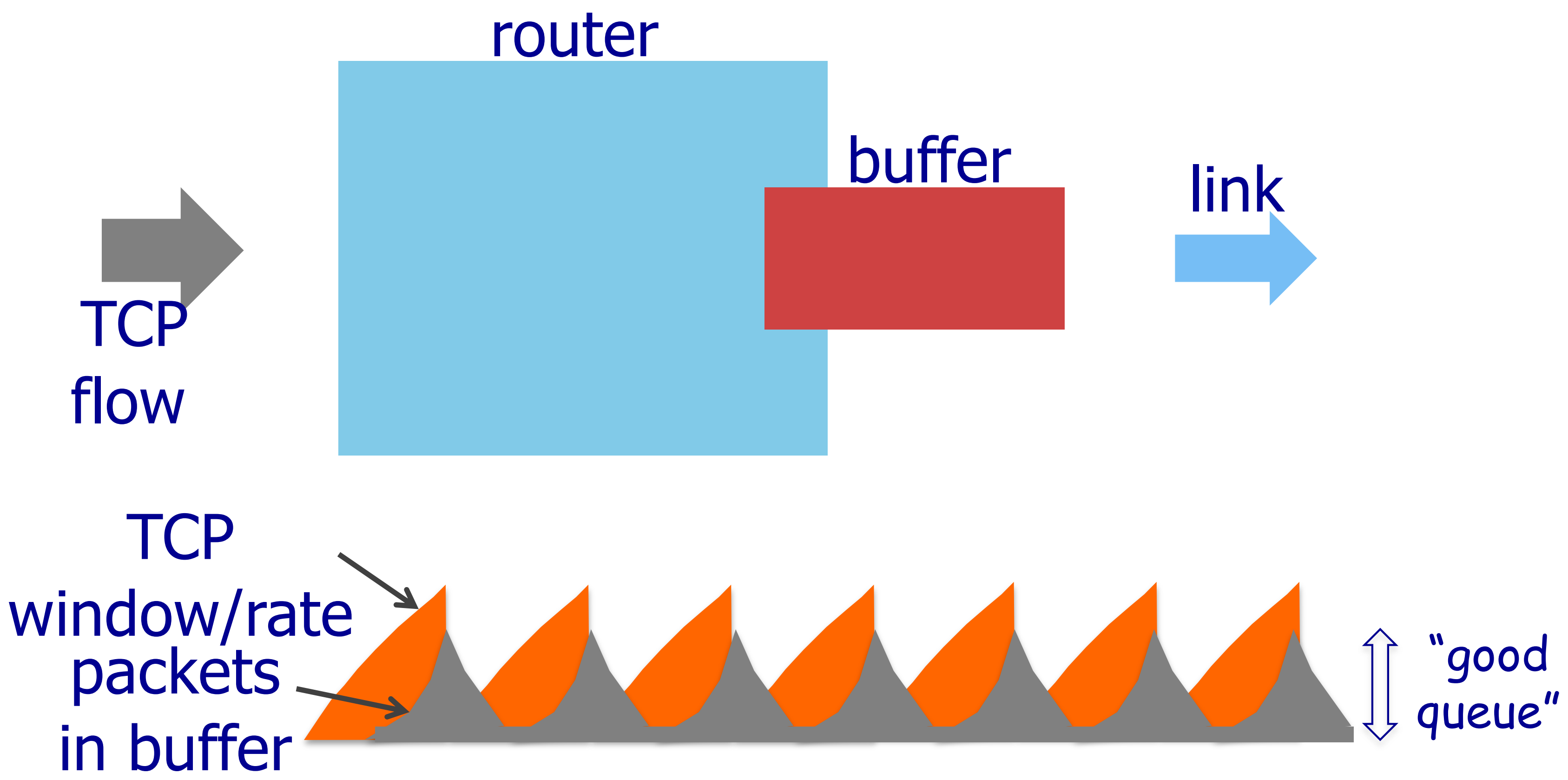
- Classic buffer-sizing rule: $B = C * RT_{prop}$
  - BDP buffer
  - Single TCP flow halving its window still gets a throughput of 100% link rate
- Q: should buffers be BDP-sized?
- Significant implications:
  - Massive pkt buffers (e.g., 40 Gbit/s with 200ms RT_{prop}): high cost
  - Massive pkt delays: bufferbloat

“Bufferbloat”

- Problem: too large buffers, notably in home routers, get filled by TCP leading to long packet latency

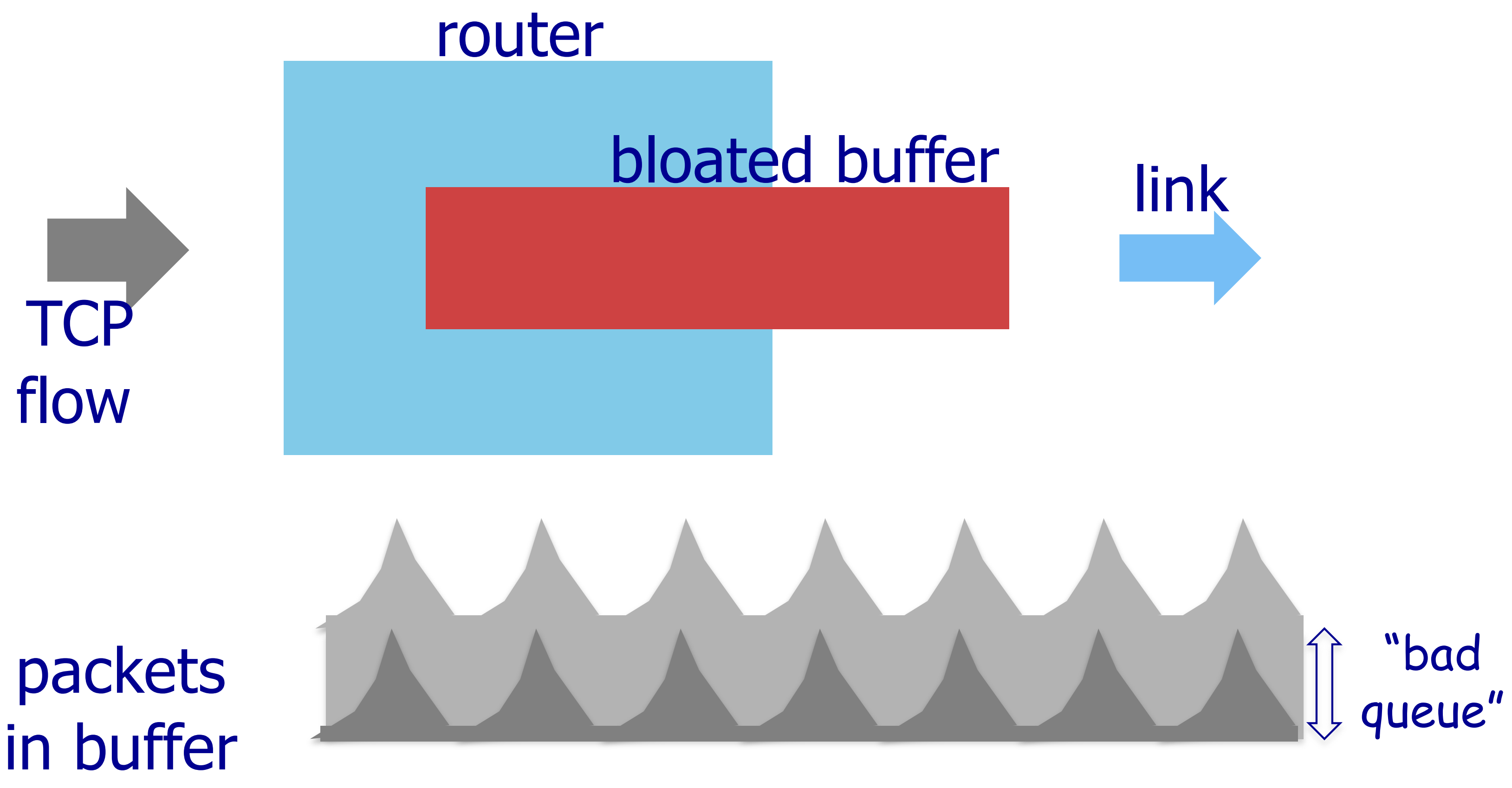
Bufferbloat

- How TCP should use the buffer



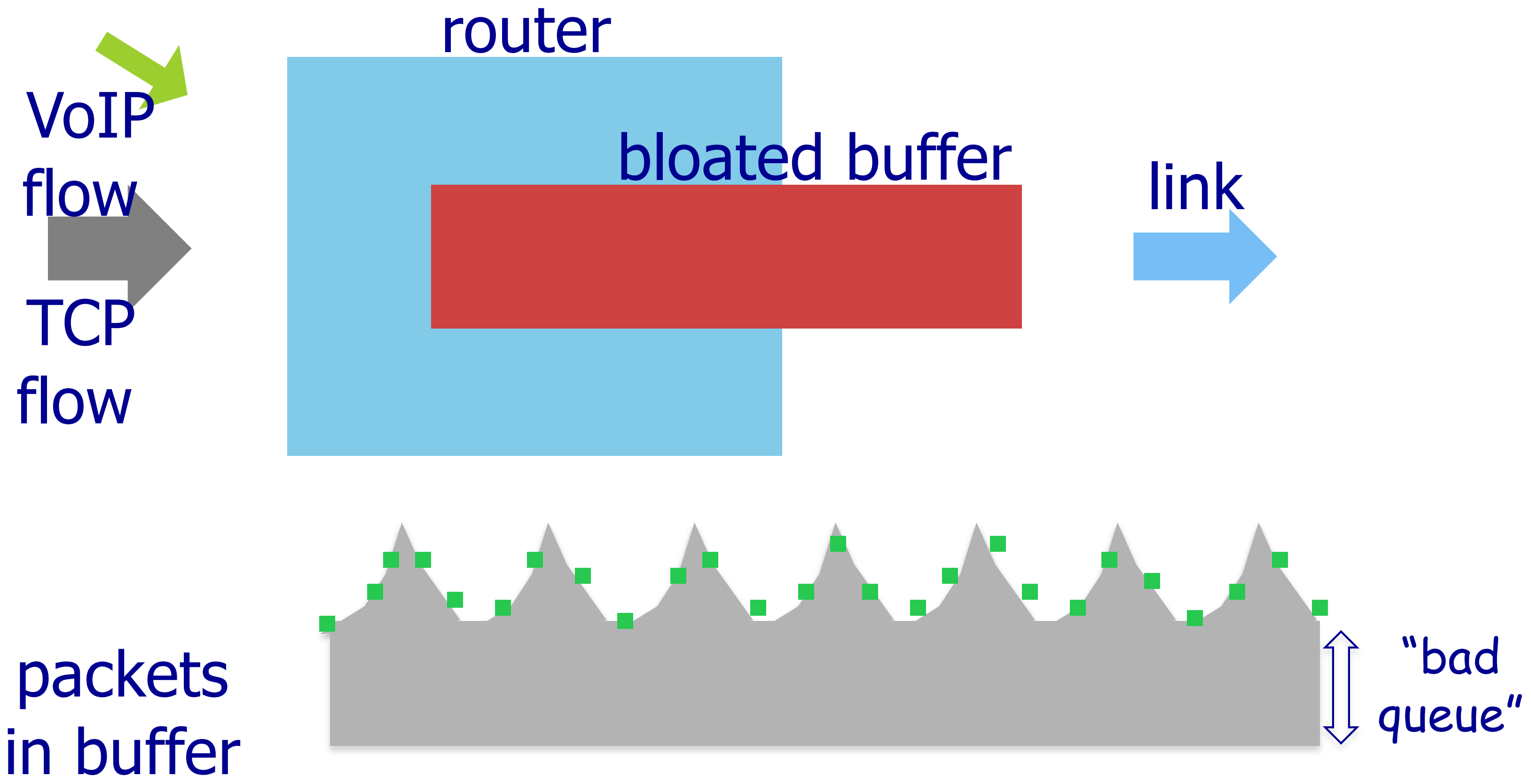
Bufferbloat

- Impact of a bloated buffer: longer delays, same throughput



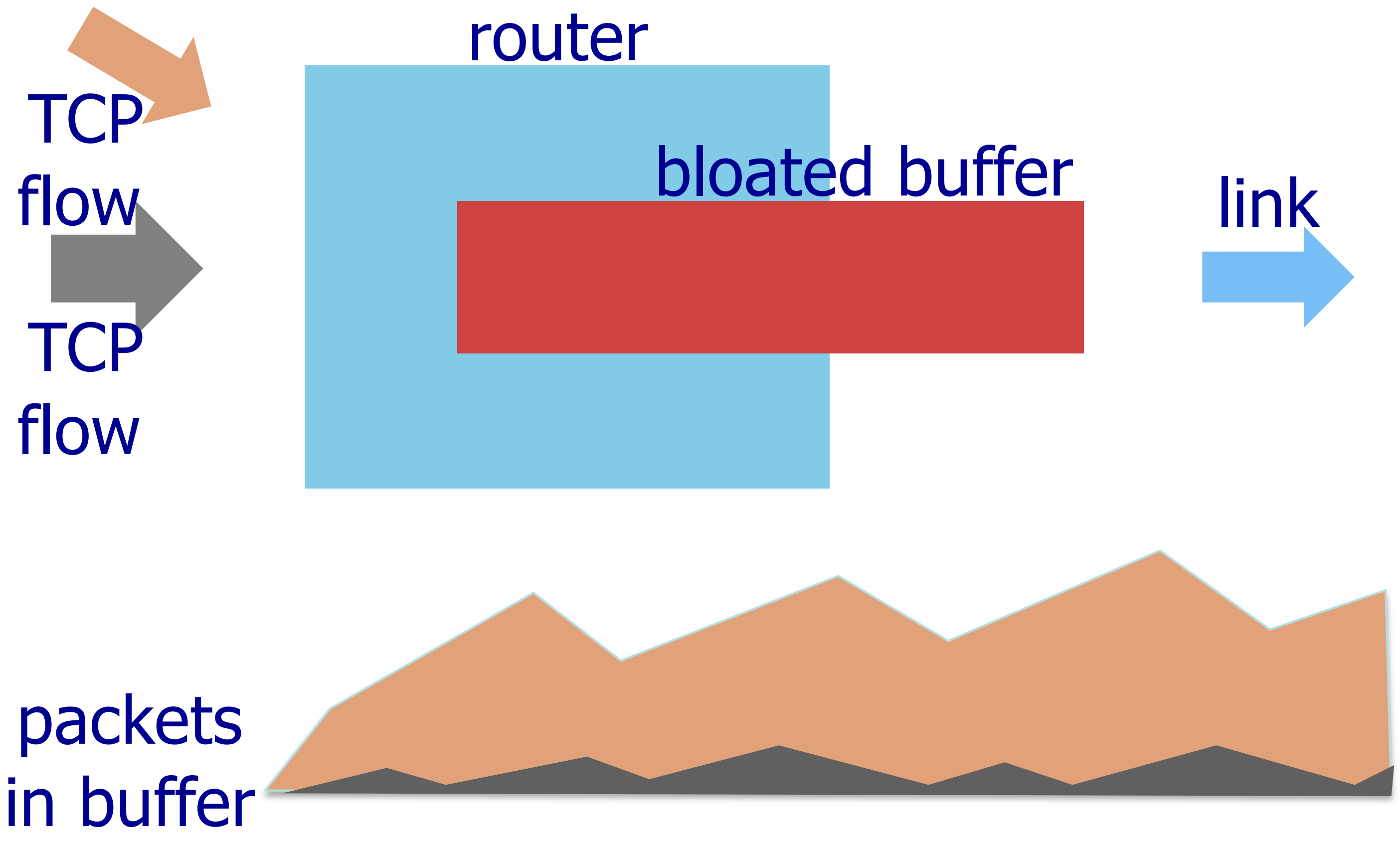
Bufferbloat

- impact of a bloated buffer: high latency for real time flows



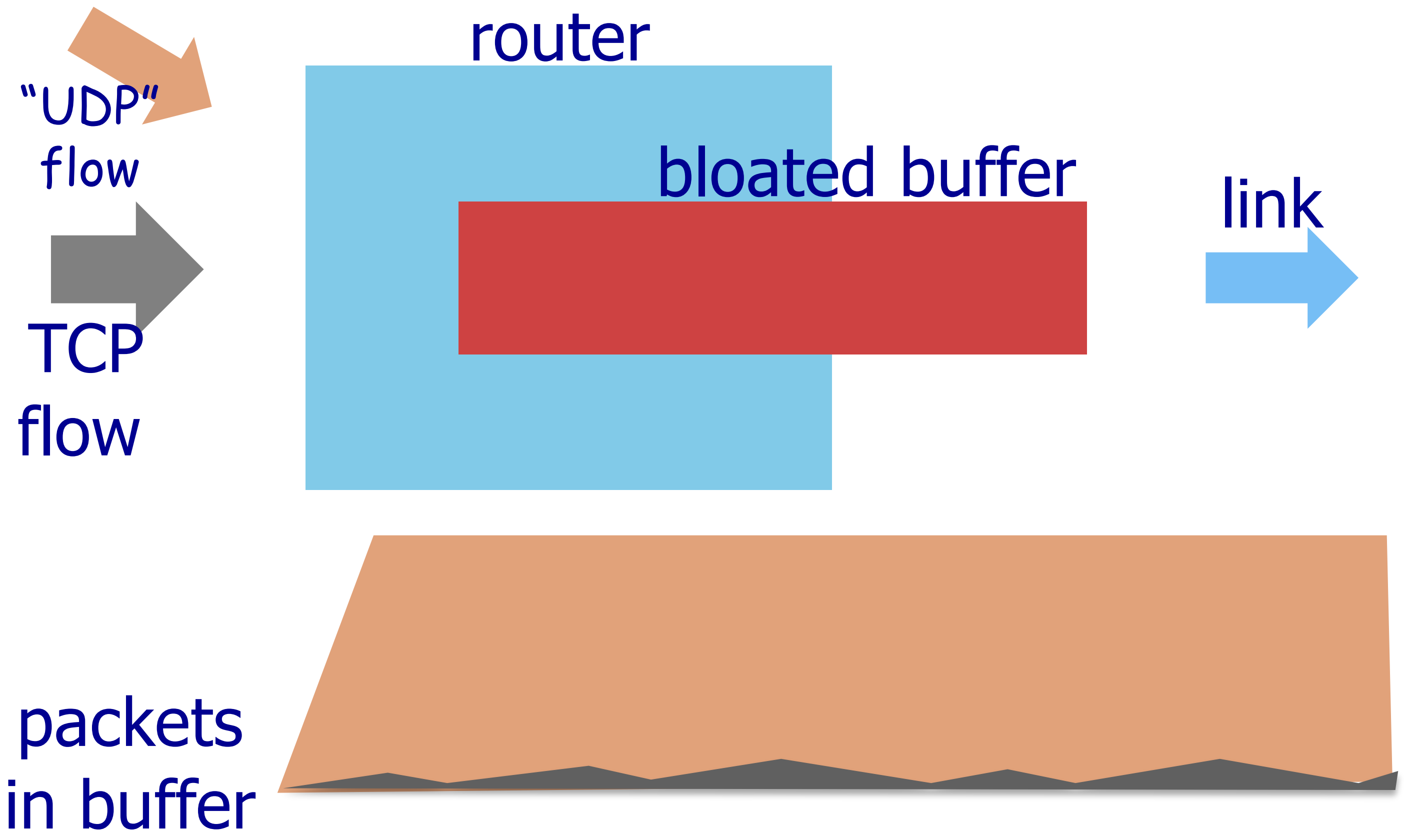
Bufferbloat

- impact of drop tail: unfair bandwidth sharing

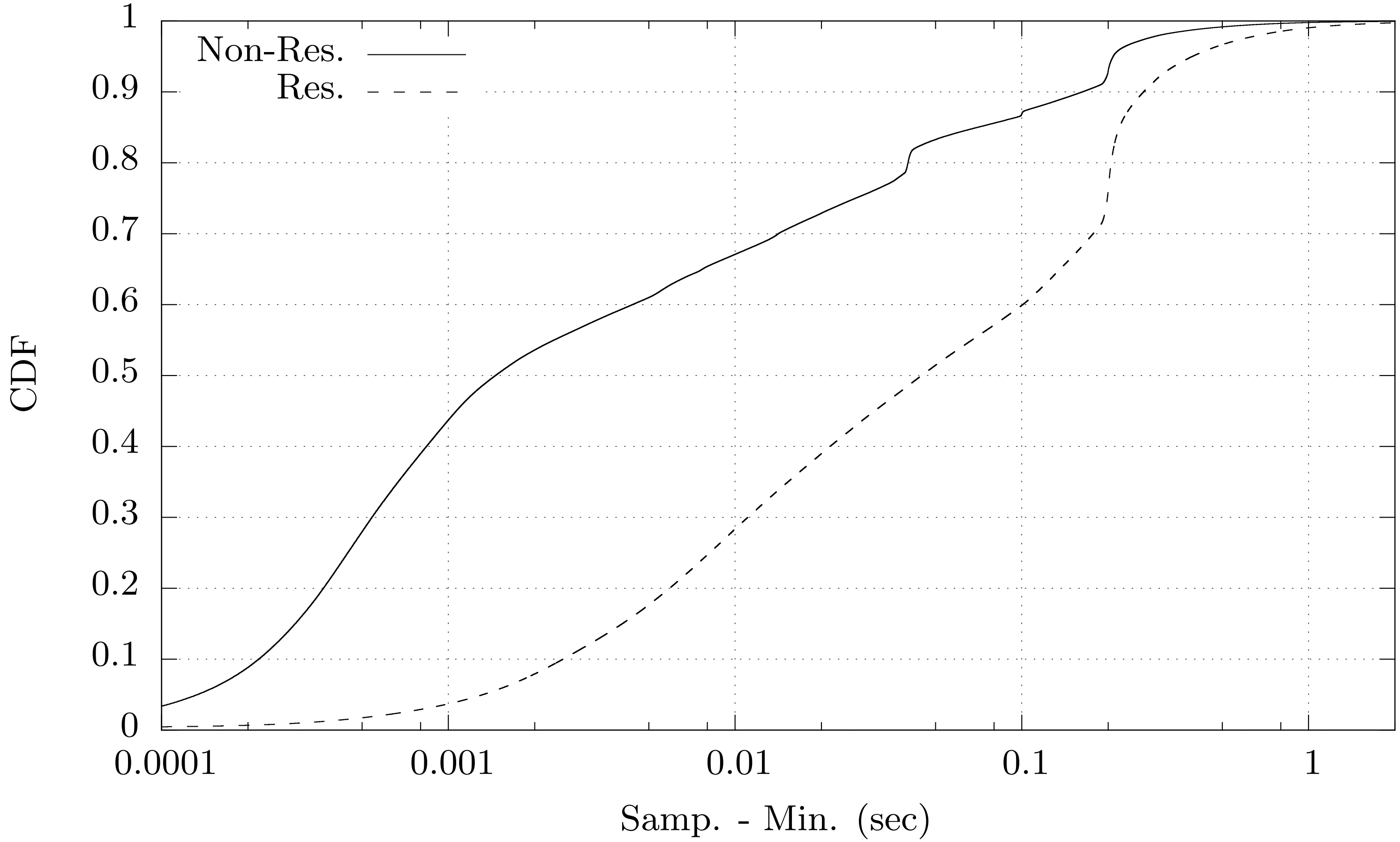


Bufferbloat

- impact of drop tail: unfair bandwidth sharing



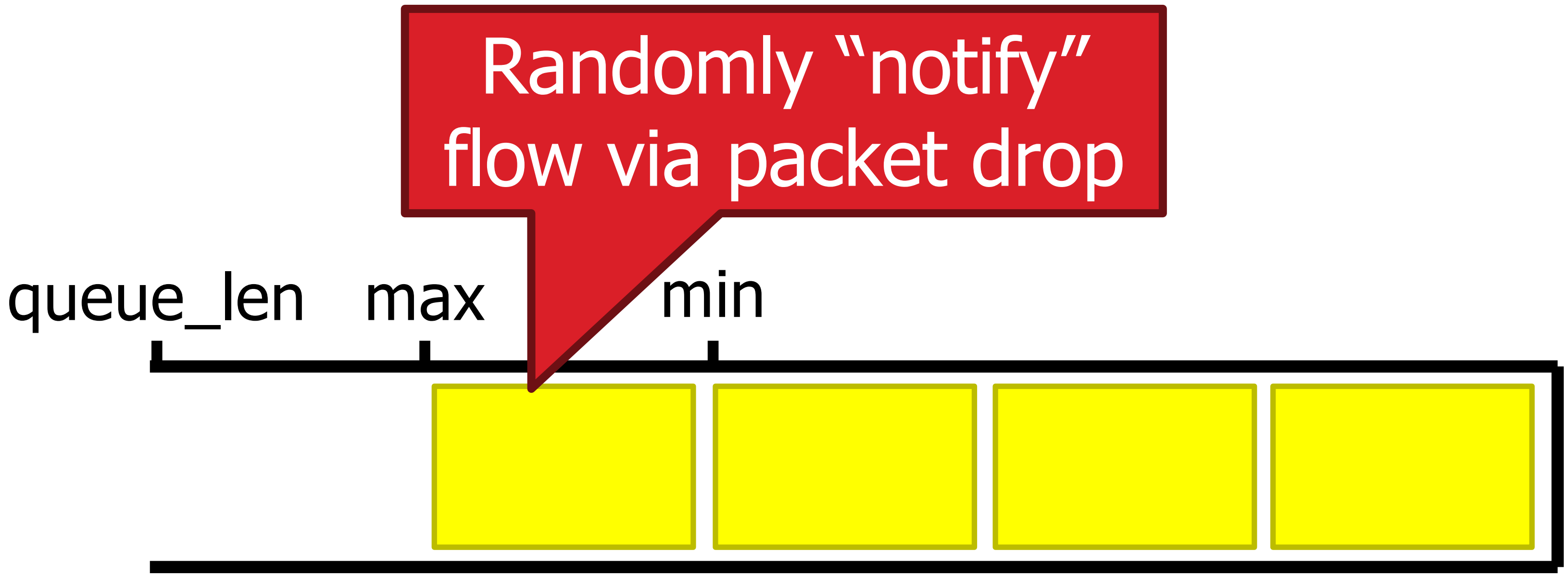
RTT measurements



- Difference between each sample and the minimum RTT for the given host pair
- The median increase in RTT is just over 1 msec for non residential peers and roughly 45 msec for residential peers

Active Queue Management (AQM)

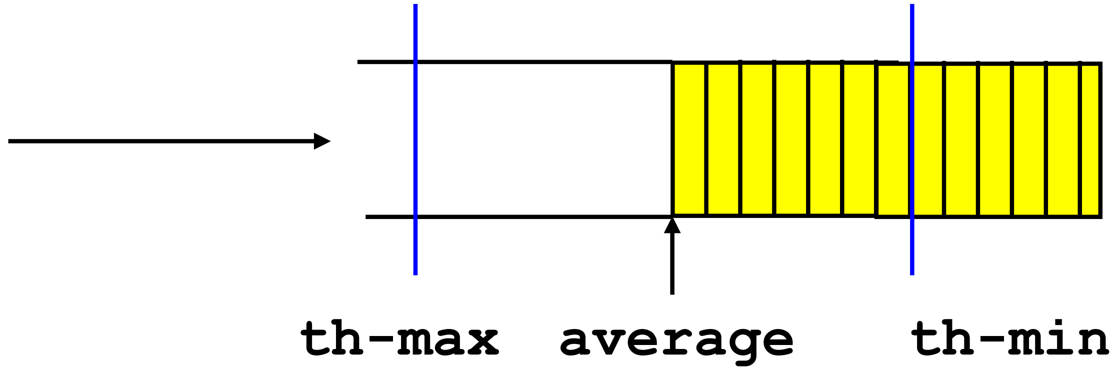
- Detect “incipient” (early) congestion in the router
- Try to keep average queue size in “good” range
- Randomly choose flows to notify about congestion
  - e.g. RED: packet drops are implicit notifications



Random Early Detection

- Family of techniques used to detect congestion and notify sources
  - when a queue is saturated, packets are dropped
  - losses interpreted as congestion signals → decrease rate
- Idea
  - act before congestion and reduce the rate of sources
  - threshold for starting to drop packets
- Losses are inefficient
  - result in retransmissions, dropped packets should be retransmitted - enter Slow Start
- Synchronization of TCP sources
  - several packets dropped
  - several sources detect congestion and enter slow start at the same instant

RED



- Estimation of the average queue length
  - **average ← q × measure + (1 - q) × average**
- If **average ≤ th-min**
  - accept the packet
- If **th-min < average < th-max**
  - drop with probability **p**
- If **th-max ≤ average**
  - drop the packet

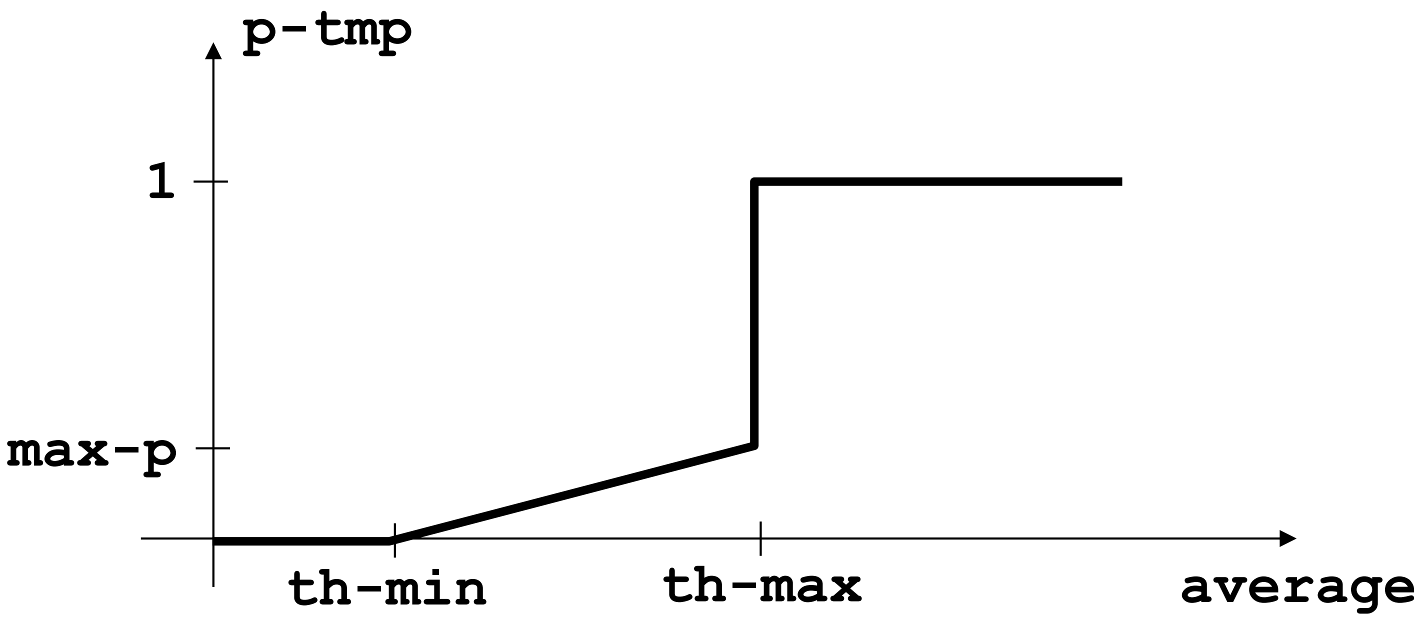
RED Characteristics

- Tends to keep the queue reasonably short
  - low delay
- Suitable for TCP
  - single loss recovered by Fast Retransmit
- Probability **p** of choosing a given flow is proportional to the rate of the flow
  - more packets of that flow, higher probability of choosing one of its packet

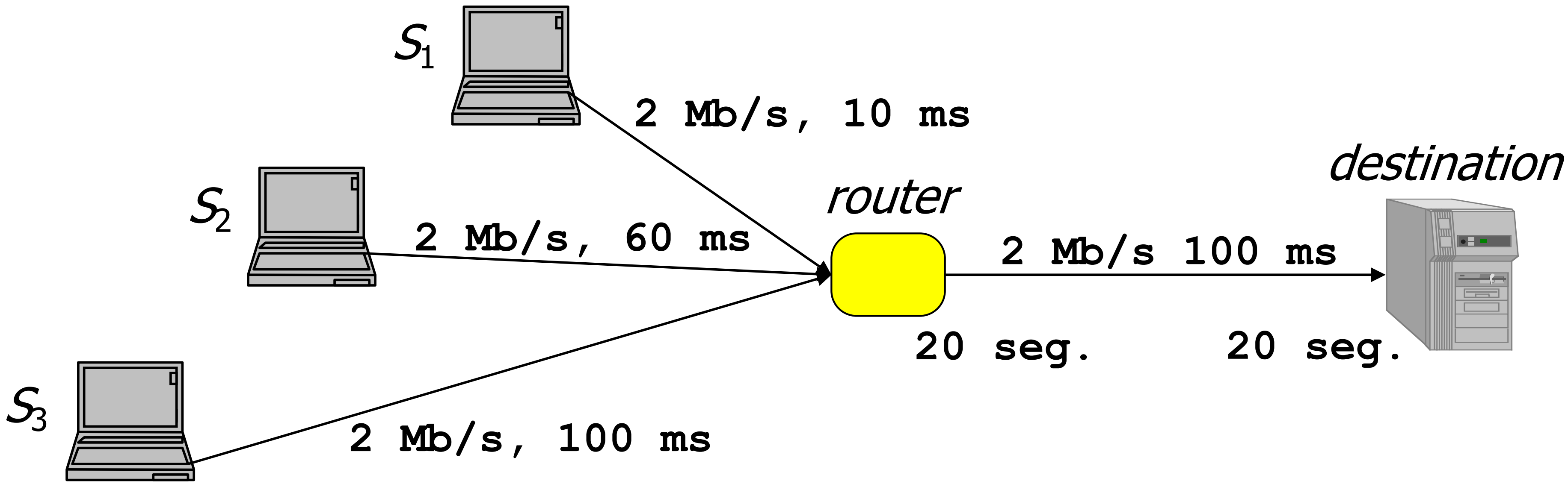
RED Characteristics

- Dynamic probability **`p`**
  - **`p-tmp = max-p × (average - th-min)/ (th-max - th-min)`**
  - **`max-p`**: maximal drop probablility when the queue attains **`th-max`** threshold
  - **`p = p-tmp/(1 - nb-packets × p-tmp)`**
  - **`nb-packets`**: how many packets have been accepted since the last drop
  - **`p`** increases slowly with **`nb-packets`**
  - drops are spaced in time
- Recommended values
  - **`max-p`** = 0.02
  - if **`average`** in the middle of two thresholds, 1 drop in 50

Drop probability

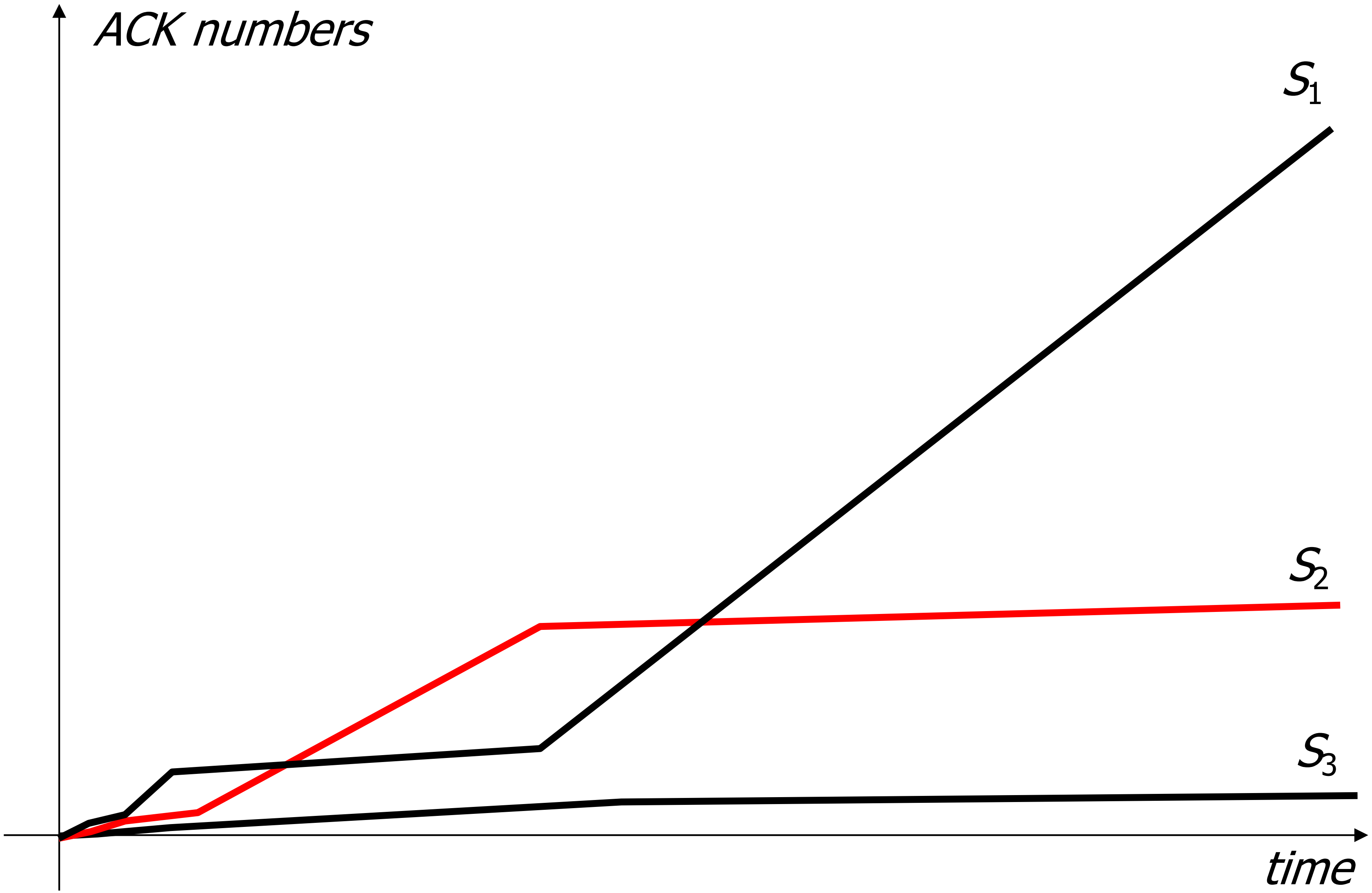


Example network for RED

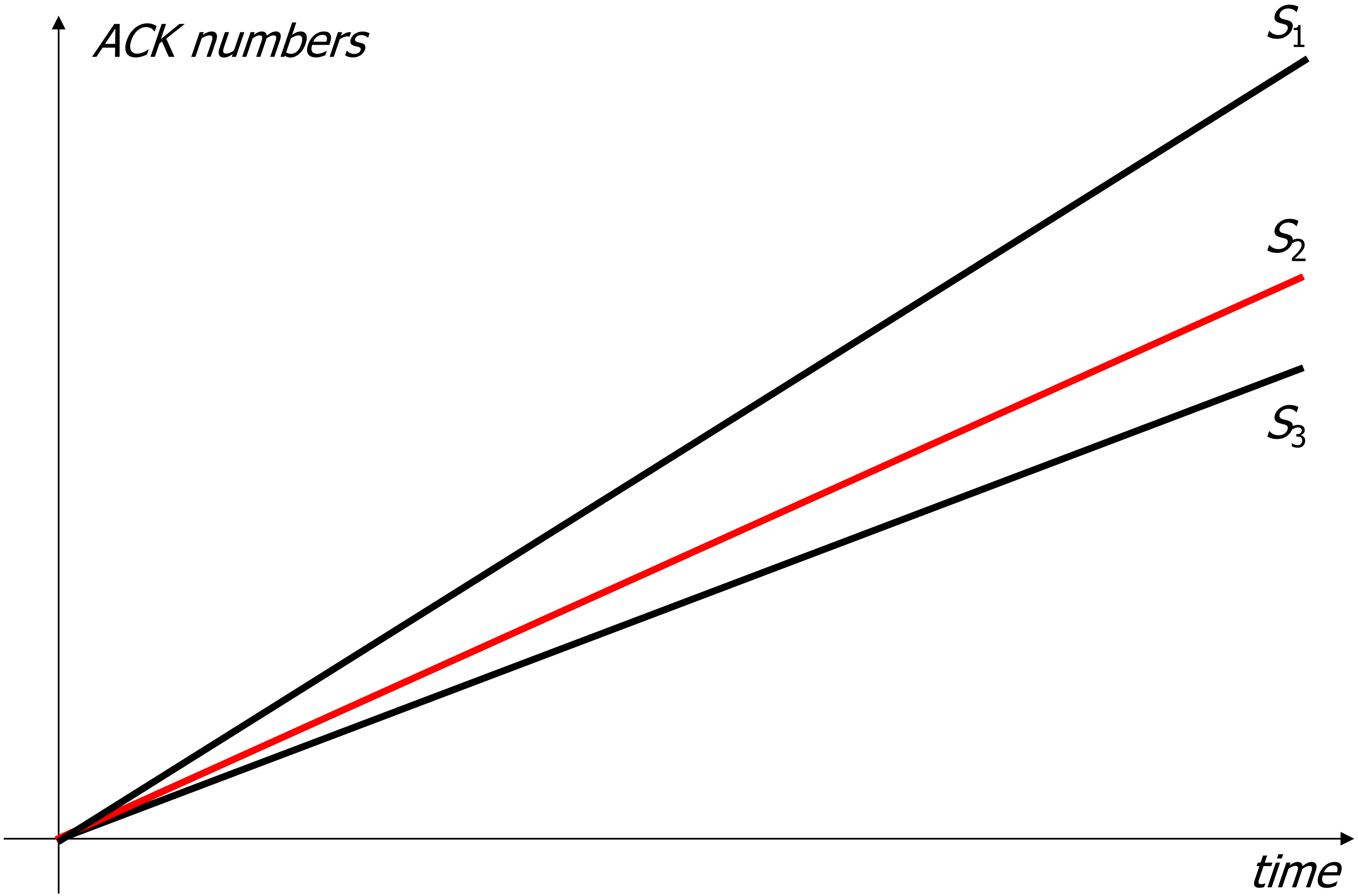


- Example network with three TCP sources
  - different link delays
  - limited queues on the link (20 packets)

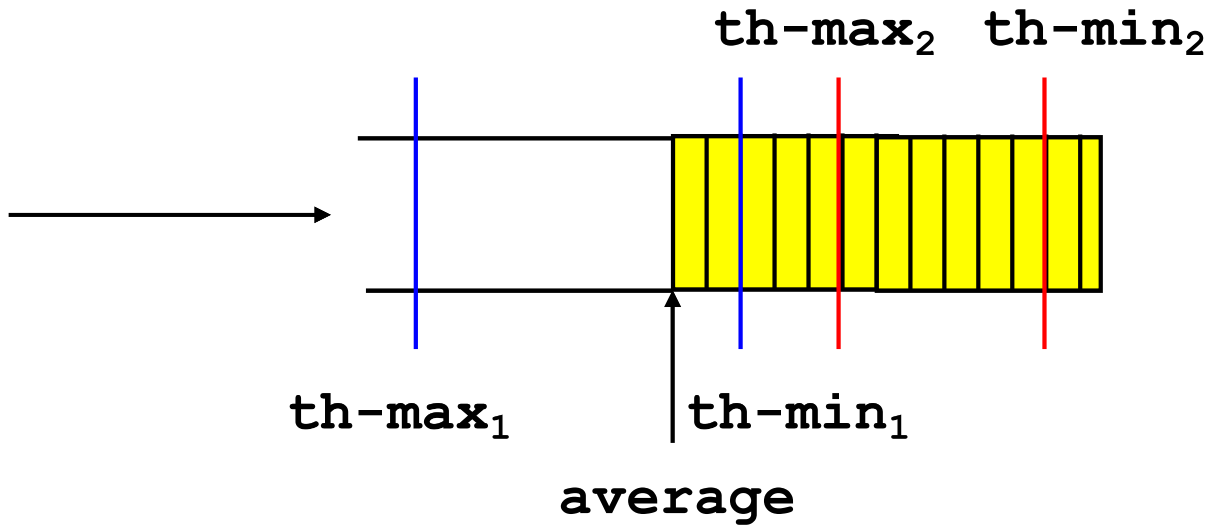
Throughput in time



Throughput in time with RED

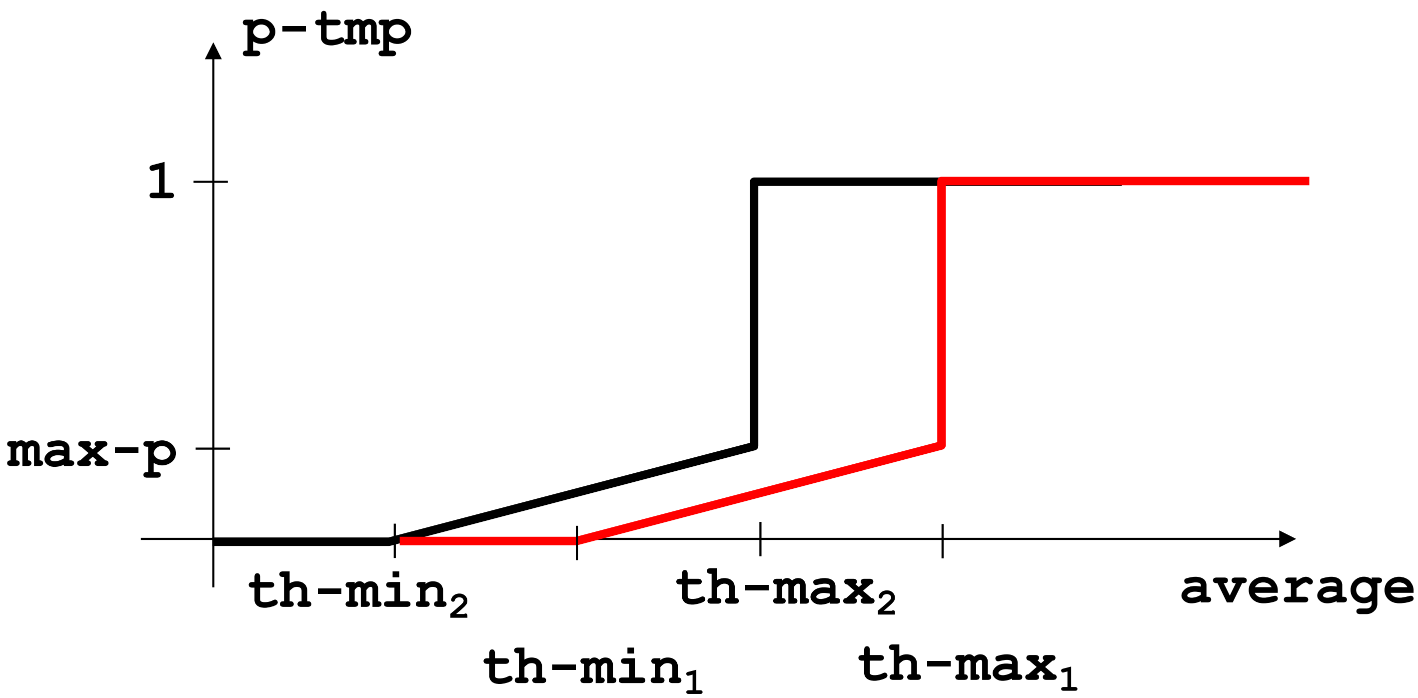


WRED



- Weighted RED
- Different thresholds for different classes
  - higher priority class - higher thresholds
    - lower drop probability
  - lower priority class - lower thresholds
    - greater drop probability
- Method for service differentiation

Different drop probabilities

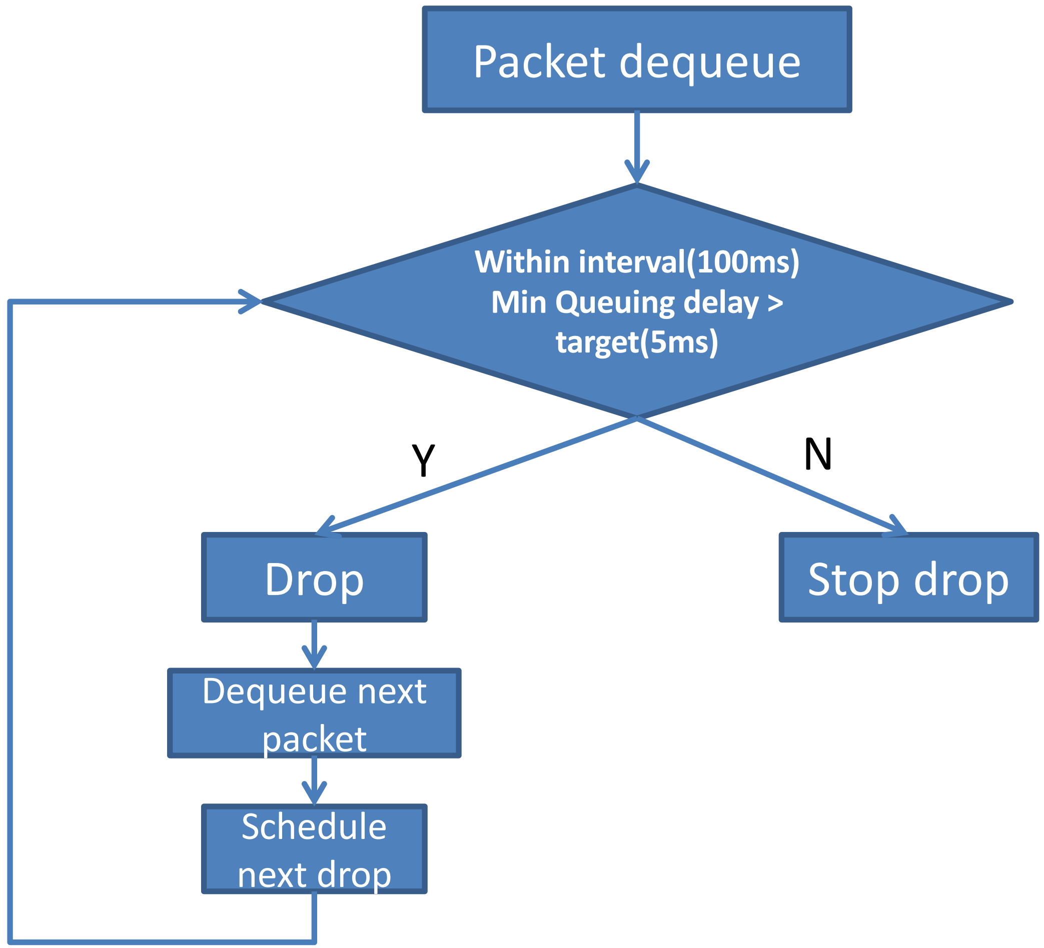


CoDel - Controlled Delay Management

CoDel

- Keep a single-state variable of how long the minimum delay has been above or below the TARGET value for standing queue delay
- Rather than measuring queue *size* in bytes or packets, we used the packet-sojourn time through the queue
  - Need to add timestamp of packet arrival time to the packet in the queue
- Standing queue of TARGET delay is OK
- No drop of packets if fewer than one MTU in the buffer
- CoDel identifies the persistent delay by tracking the (local) minimum queue delay packet experience
- To ensure that the minimum value does not become stale, it has to have been experienced within the most recent INTERVAL (time on the order of a worst-case RTT of connections through the bottleneck)

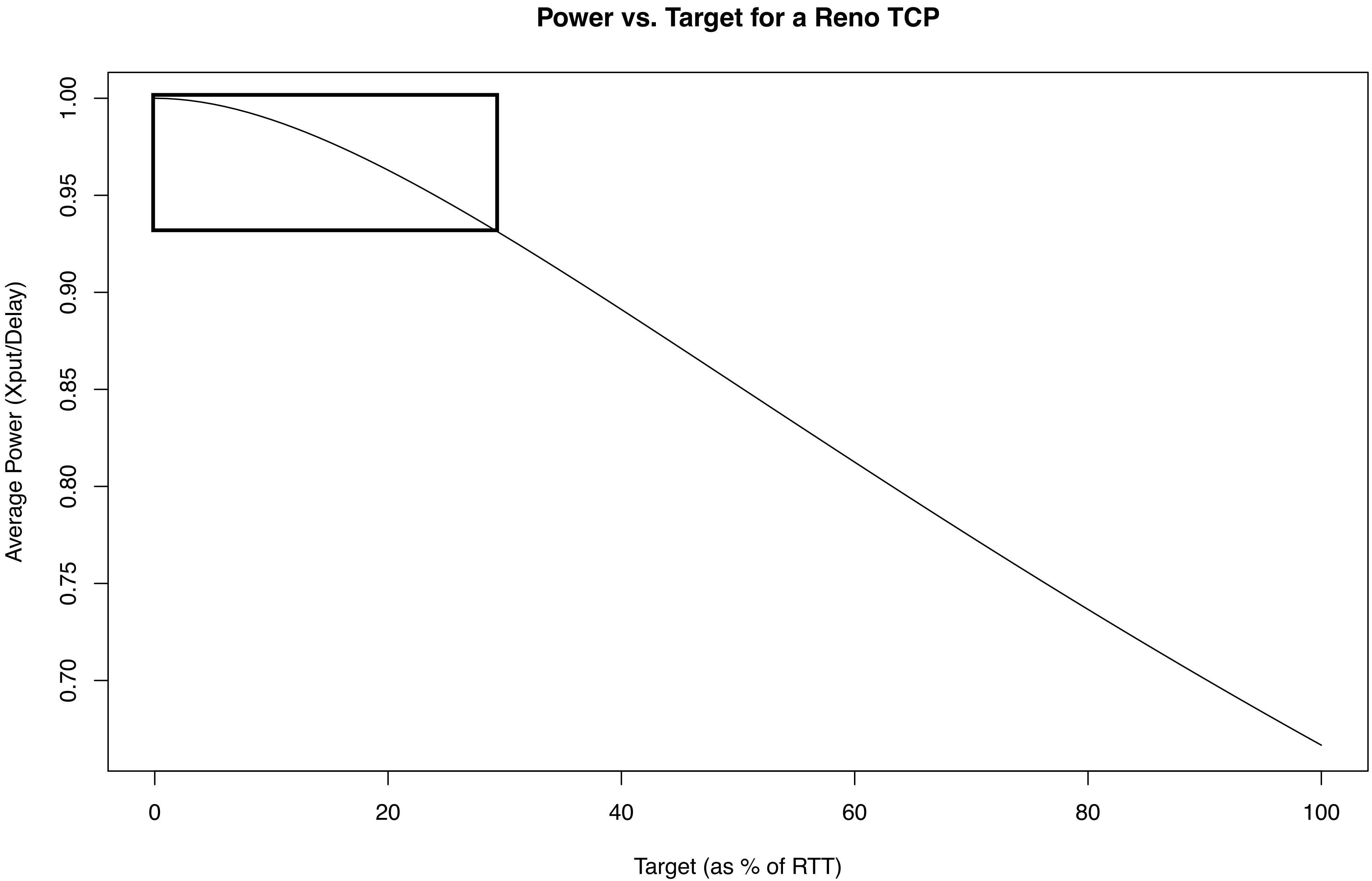
CoDel



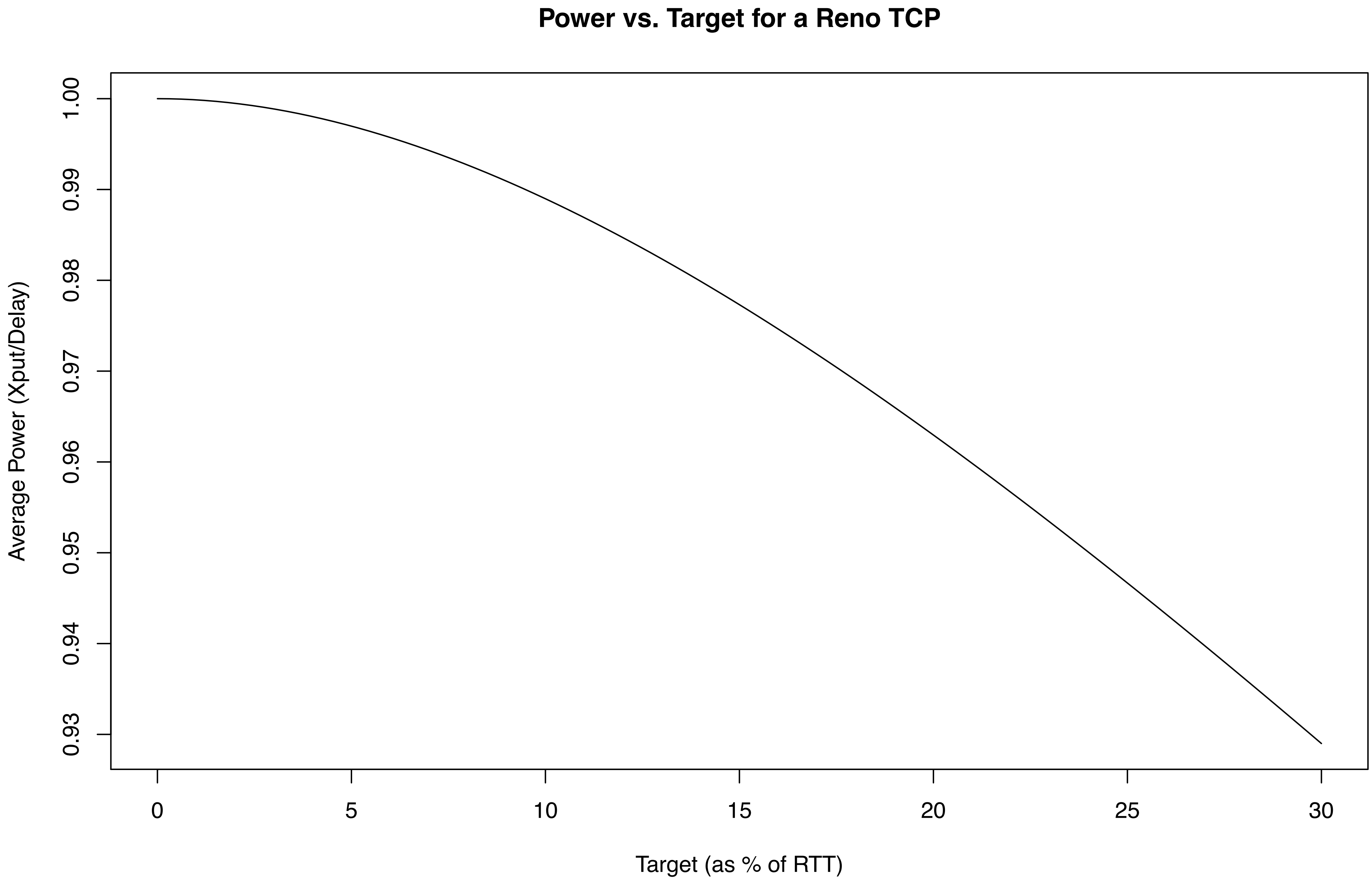
Packet dequeue
Within interval(100ms)
Min Queuing delay >
target(5ms)
Y
N
Drop
Stop drop
Dequeue next packet
Schedule next drop

CoDel

- TARGET = 5 ms (optimizes power)
- INTERVAL = 100 ms (normal Internet usage)
- Dequeue packet
  - track whether the sojourn time is above or below TARGET and, if above, if it has remained above continuously for at least INTERVAL
  - If the sojourn time has been above TARGET for INTERVAL, enter DROPPING STATE - minimum packet sojourn time is greater than TARGET
- If in DROPPING STATE
  - drop first packet: ++count,
  - fix next time for the next drop:
    - t + INTERVAL / sqrt(count) // next drop time is decreased in inverse proportion to the square root of the number of drops since the DROPPING STATE



Power vs. Target for a Reno TCP
Average Power (Xput/Delay)
1.00
0.95
0.90
0.85
0.80
0.75
0.70
0
20
40
60
80
100
Target (as % of RTT)



Power vs. Target for a Reno TCP
Average Power (Xput/Delay)
1.00
0.99
0.98
0.97
0.96
0.95
0.94
0.93
0
5
10
15
20
25
30
Target (as % of RTT)

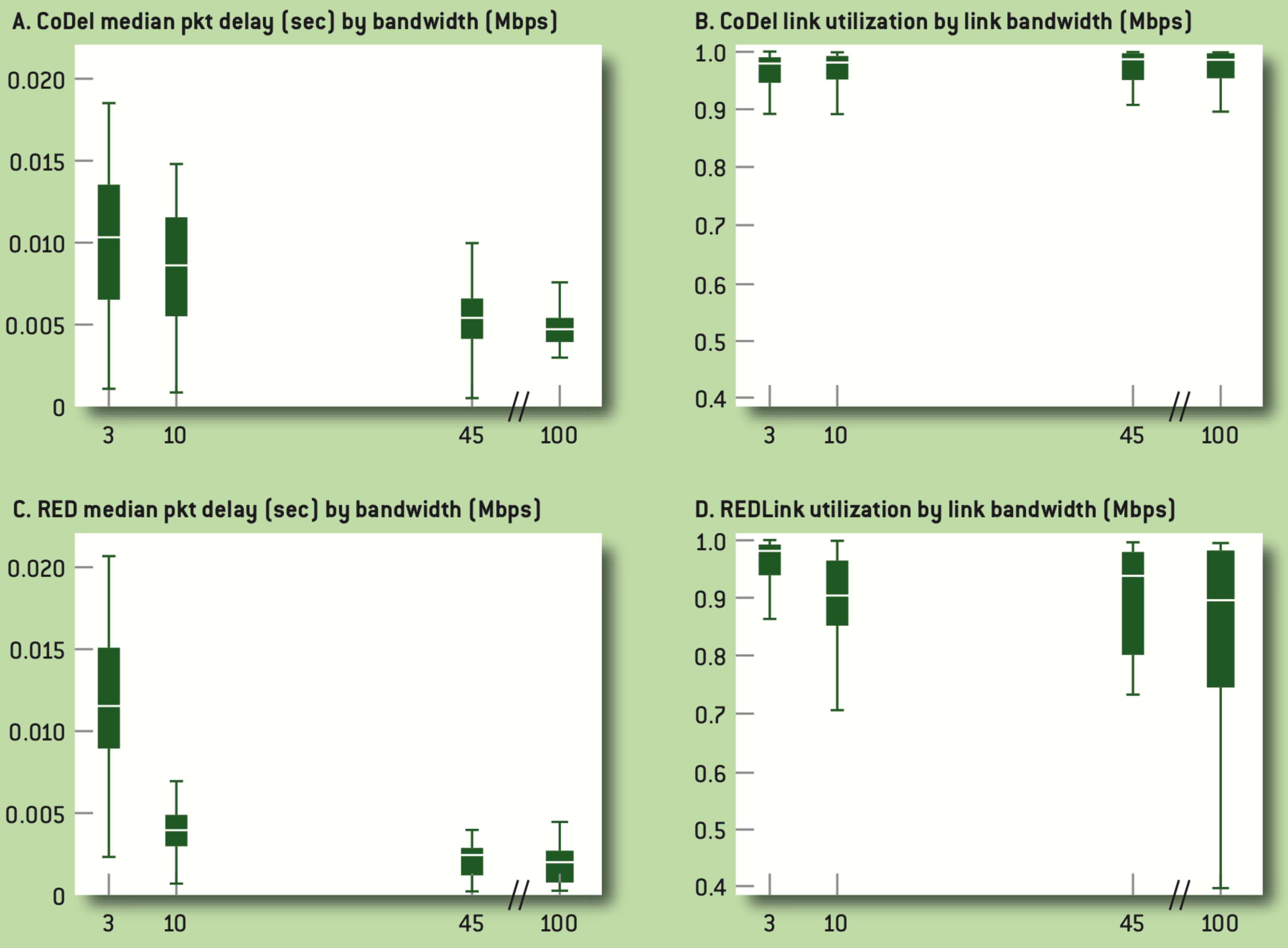
CoDel

- Setpoint target of 5% of nominal RTT (5 ms for 100 ms RTT) yields substantial utilization improvement for small added delay

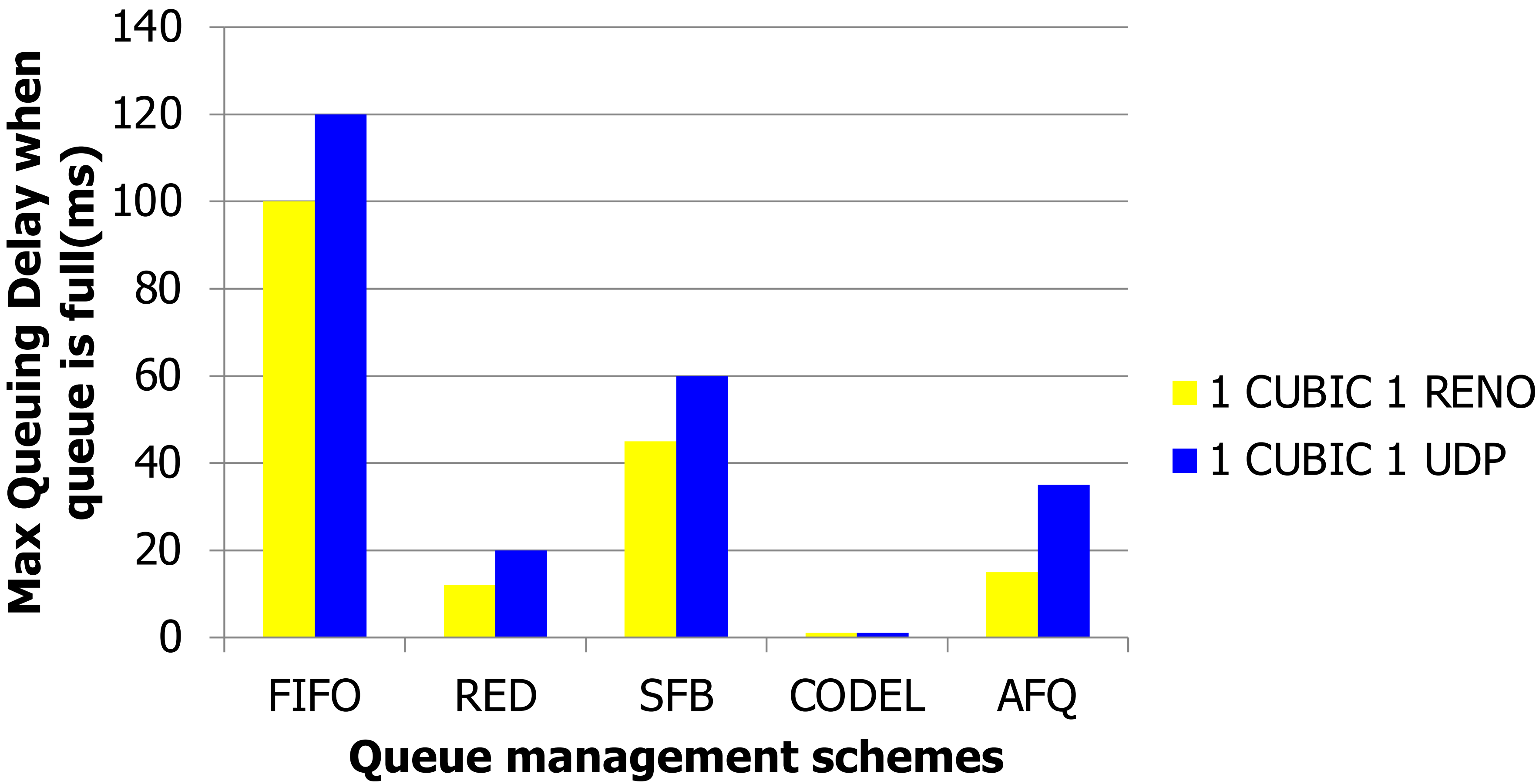
CoDel - Controlled Delay Management

- AQM that has the following characteristics:
- parameterless—it has no knobs for operators, users, or implementers to adjust.
- treats good queue and bad queue differently—that is, it keeps the delays low while permitting bursts of traffic.
- controls delay, while insensitive to round-trip delays, link rates, and traffic loads.
- adapts to dynamically changing link rates with no negative impact on utilization.
- simple and efficient—it can easily span the spectrum from low-end, Linux-based access points and home routers up to high-end commercial router silicon.

CoDel and RED Performance Variation with Link Bandwidth

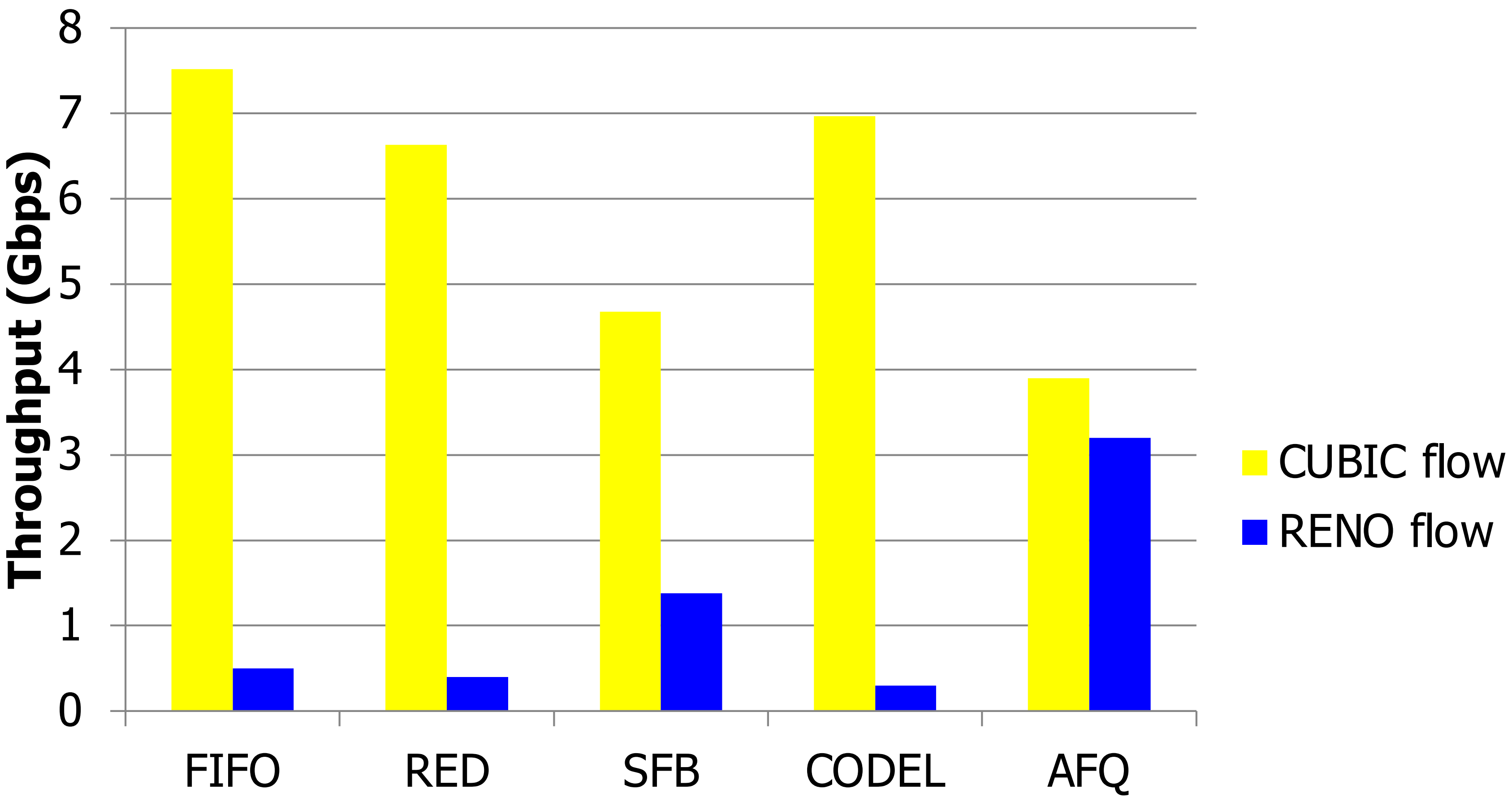


Delay



CODEL works great for control delay!

Throughput and Fairness



Queue Management Schemes

1 CUBIC flow and 1 RENO flow competing in the bottleneck

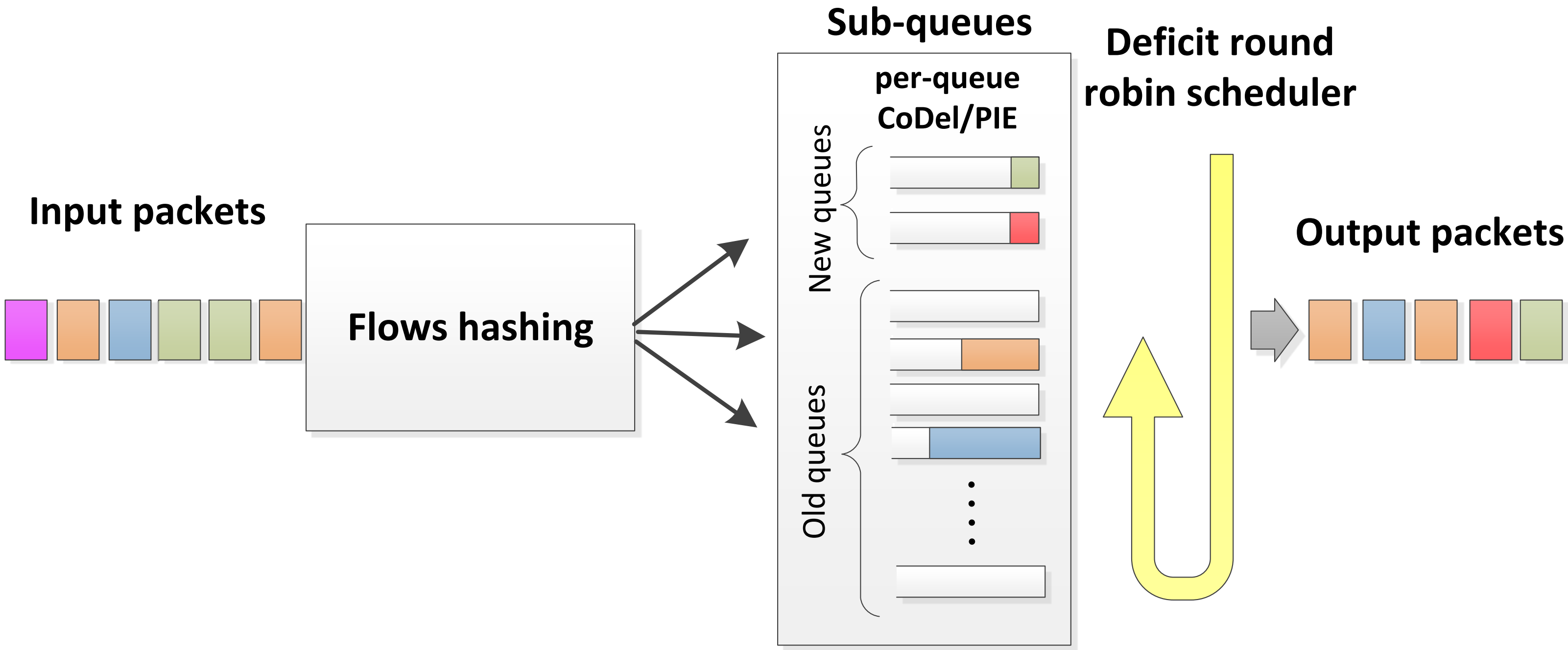
Unfairness for heterogeneous TCPs!

AFQ approximately solves fairness problem

Flow Queue CoDel Packet Scheduler - FQ-CoDel

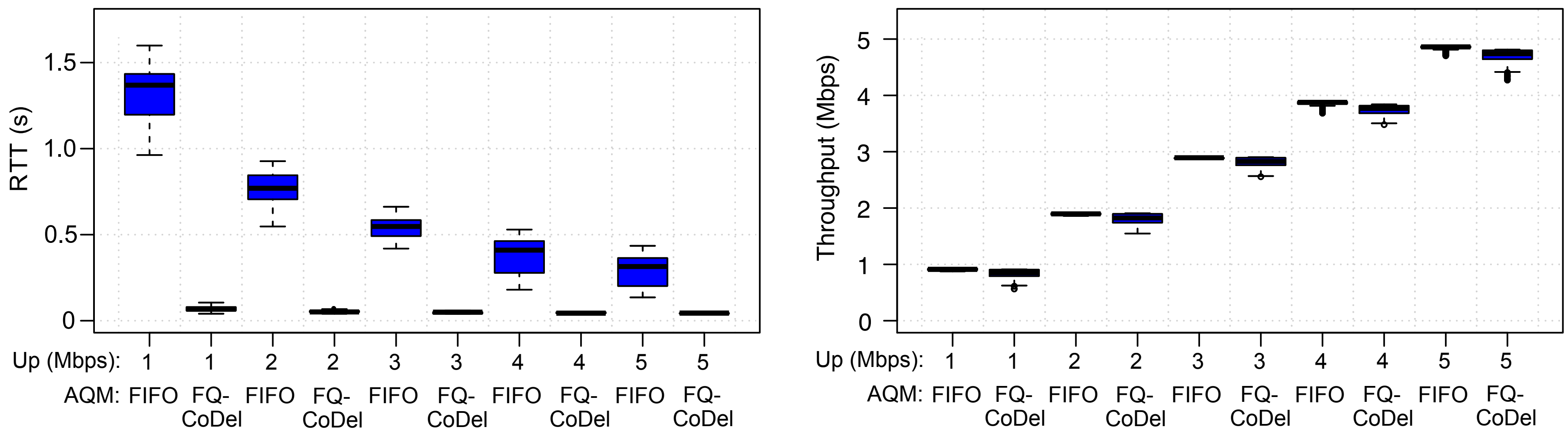
- Combine a packet scheduler (DRR) with Co-Del (AQM)
- Optimization for sparse flows similar to Shortest Queue First (SQF)
  - "flow queueing" rather than "fair queueing", as flows that build a queue are treated differently from flows that do not
- FQ-CoDel stochastically classifies incoming packets into different queues by hashing 5-tuple (not exactly Flow Queueing)
  - each queue managed by the CoDel AQM
  - packet ordering within a queue is preserved - FIFO
- Round-robin mechanism distinguishes between
  - "new" queues (which don't build up a standing queue) and
  - "old" queues (which have queued enough data to be active for more than one iteration of the round-robin scheduler).

FQ-CoDel



Input packets
Flows hashing
Sub-queues
per-queue
CoDel/PIE
New queues
Old queues
Deficit round
robin scheduler
Output packets

Performance of FQ-CoDel



(a) RTT: Very low over FQ-CoDel, very high over FIFO

(b) Throughput: Slightly lower under FQ-CoDel compared to FIFO

Figure 2: A single upstream TCP flow through a FIFO or FQ-CoDel bottleneck, $RTT_{base} = 40ms$, link speeds 1-5Mbps

Performance of Reno - FIFO

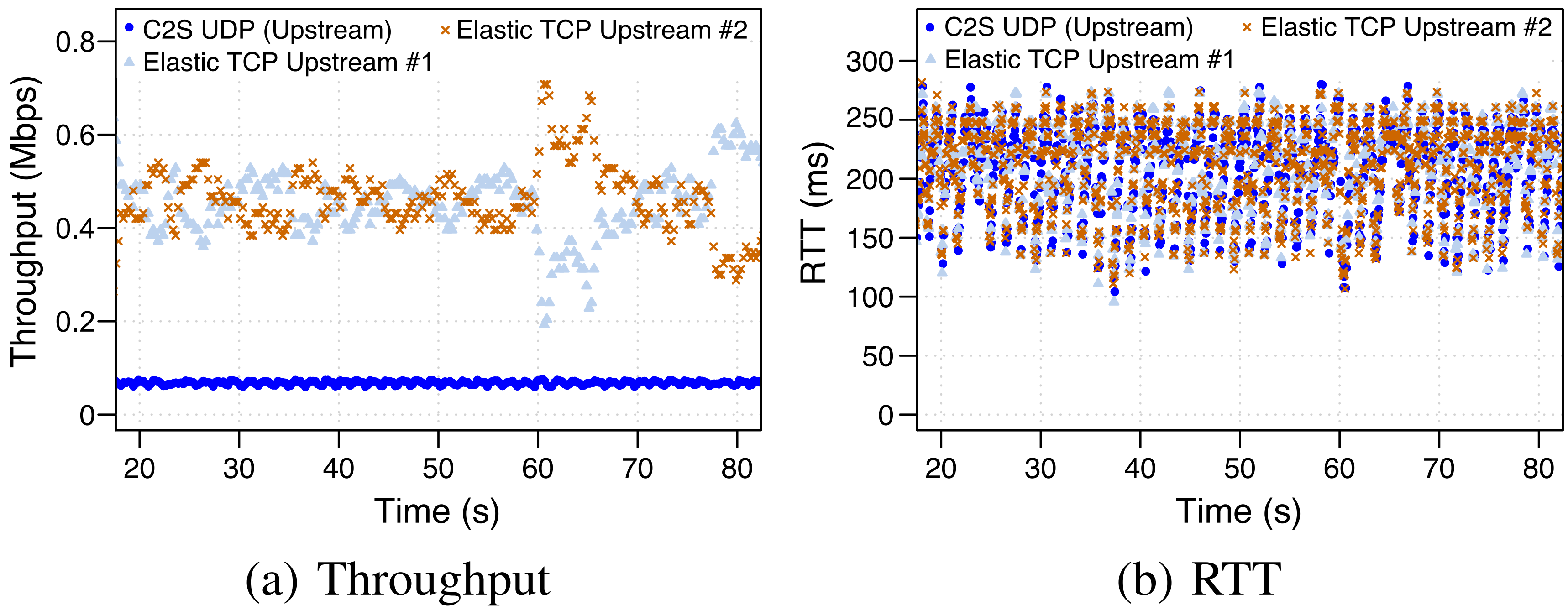
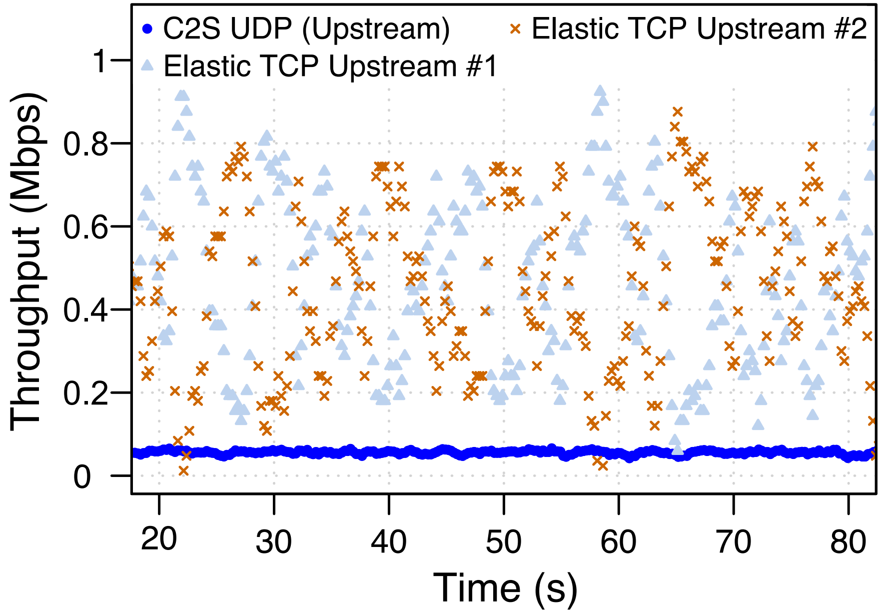


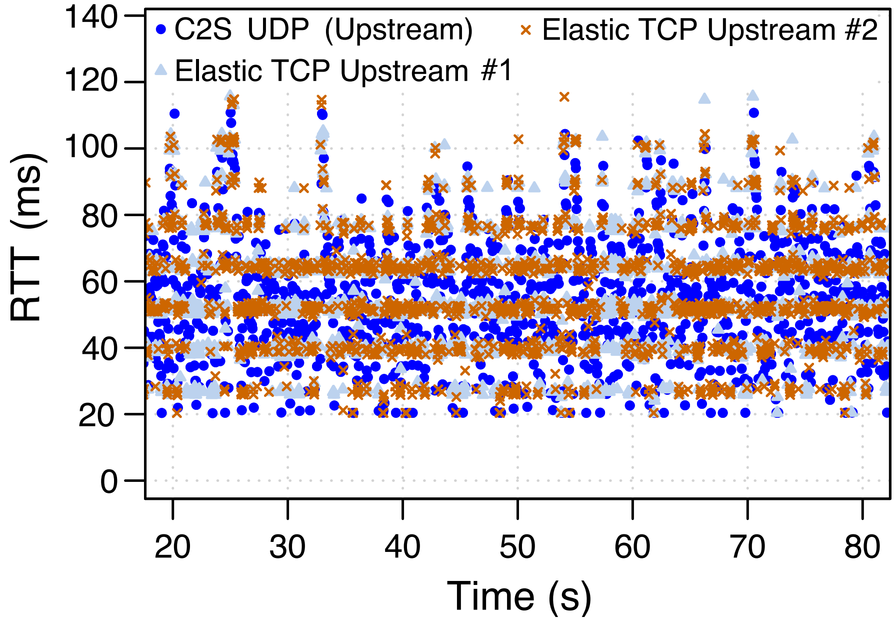
Figure 4: Upstream congestion of a 15/1Mbps link using FIFO bottleneck, $RTT_{base} = 20ms$

- 2 Reno sources, 1 UDP gaming stream, FIFO

Performance of Reno - CoDel



(a) Throughput

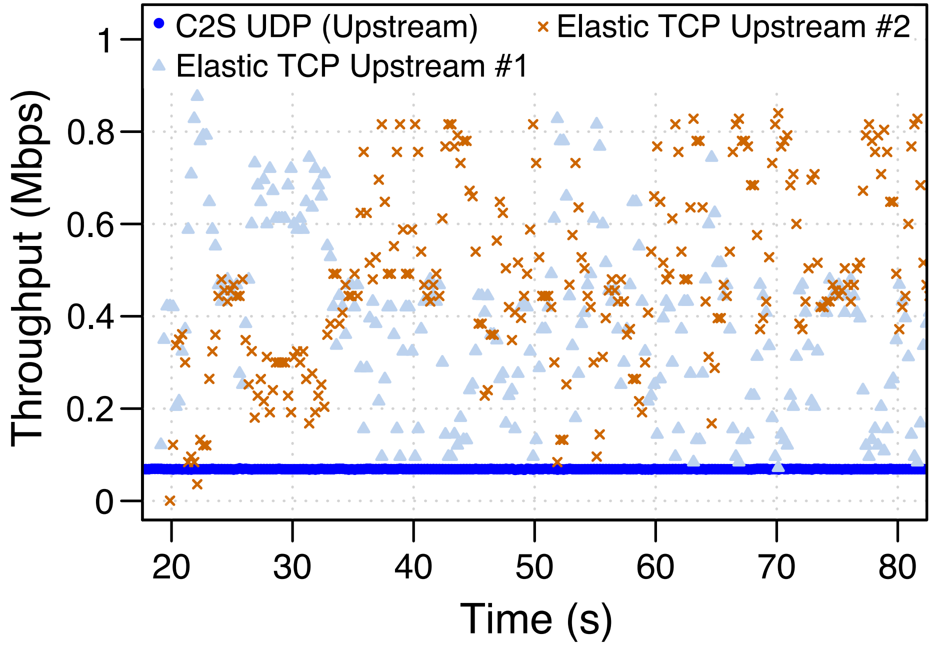


(b) RTT

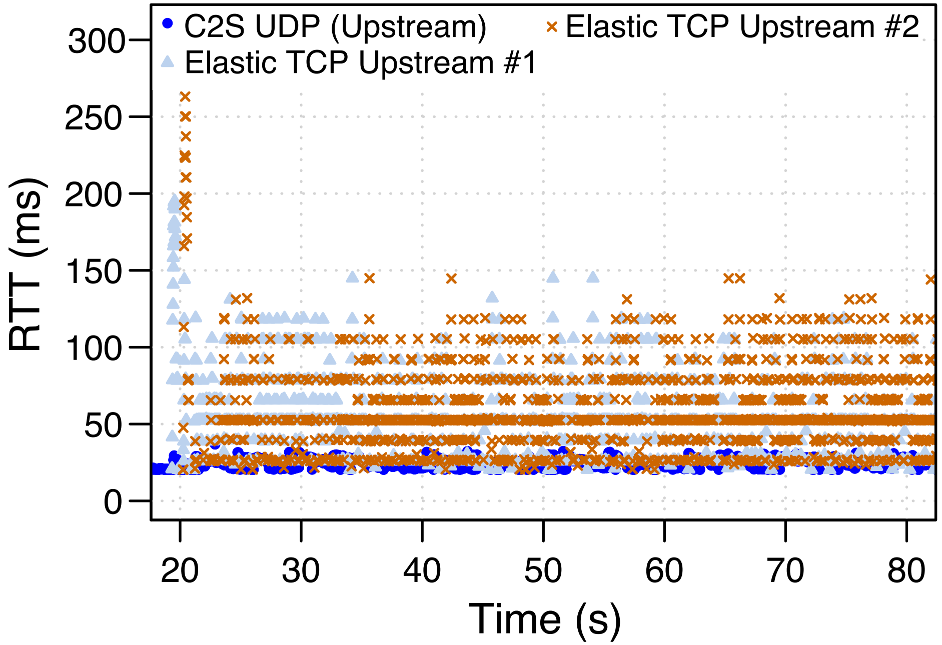
Figure 5: CoDel during upstream congestion, 15/1Mbps, $RTT_{base} = 20ms$

- 2 Reno sources, 1 UDP gaming stream, CoDel

Performance of Reno - FQ-CoDel



(a) Throughput, $RTT_{base} = 20ms$



(b) RTT, $RTT_{base} = 20ms$

- 2 Reno sources, 1 UDP gaming stream, FQ-CoDel