FIXING BUFFERBLOAT IN WIFI

STATUS AND NEXT STEPS

Toke Høiland-Jørgensen, Karlstad University
toke.hoiland-jorgensen@kau.se

Netdev 2.2, Seoul, South Korea

Nov 10th, 2017

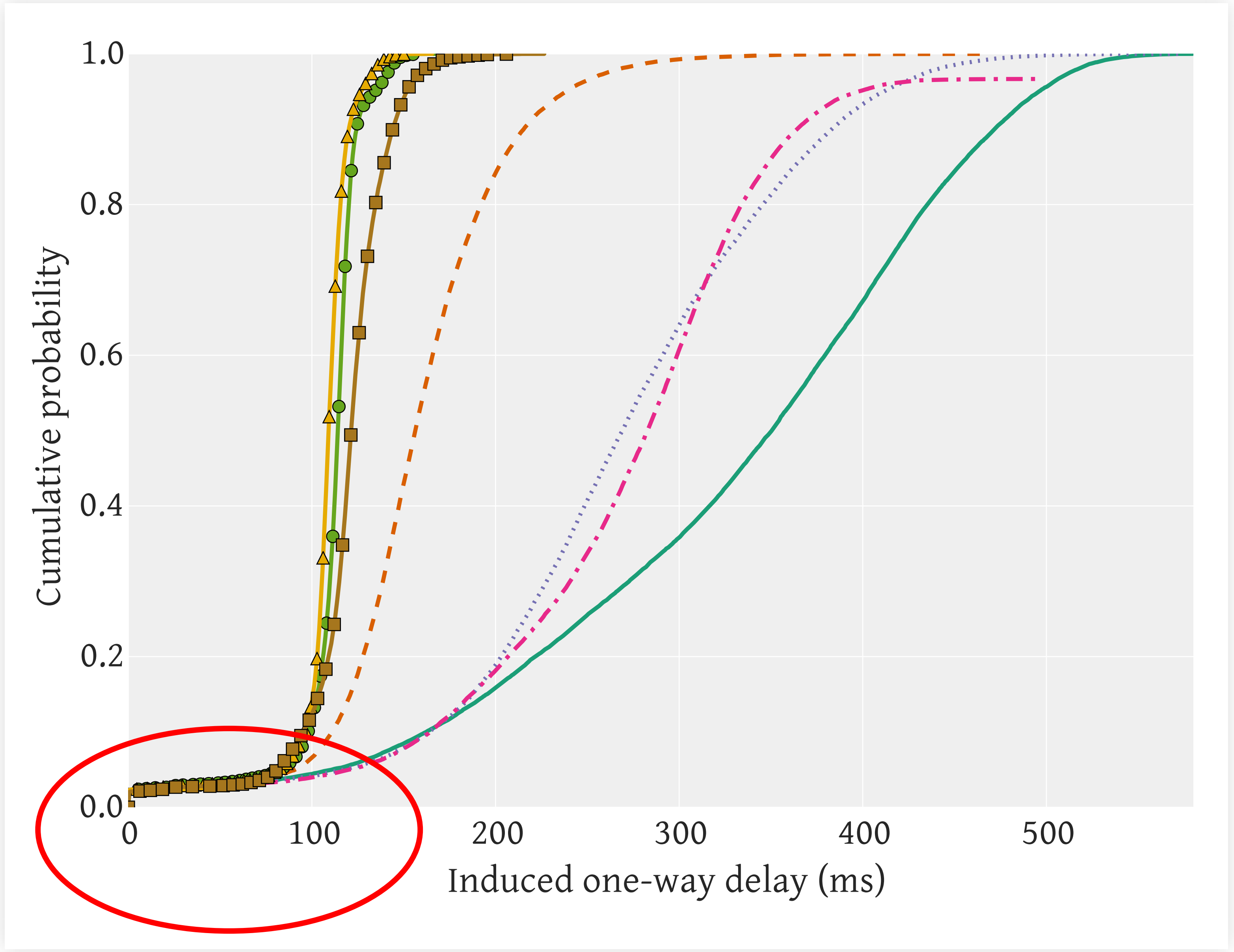
COMPUTER SCIENCE
DATAVETENSKAP

OUTLINE

- The problem
- 802.11 MAC protocol constraints
- The new mac80211 queueing structure and airtime scheduler
- Next steps - feedback wanted!
  - More latency reductions
  - Airtime policies
  - QoS handling
  - Configurability
- Summary

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

BUFFERBLOAT



Best qdisc still had 100ms+ of bloat on WiFi.

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

AIRTIME (UN)FAIRNESS

Effective transmission time $T(i)$ and rate $R(i)$ (for station $i \in I$):

$$T(i) = \begin{cases} \frac{1}{|I|} & \text{with fairness} \\ \frac{T_{data}(i)}{\sum_{j \in I} T_{data}(j)} & \text{otherwise} \end{cases}$$
$$R(i) = T(i)R_0(i)$$

Where $R_0(i) = \frac{L_i}{T_{data}(i)+T_{oh}}$ is the effective rate of a station transmitting without collisions.

Network throughput is determined by the **slowest** station.

COMPUTER SCIENCE
DATAVETENSKAP

802.11 MAC PROTOCOL CONSTRAINTS

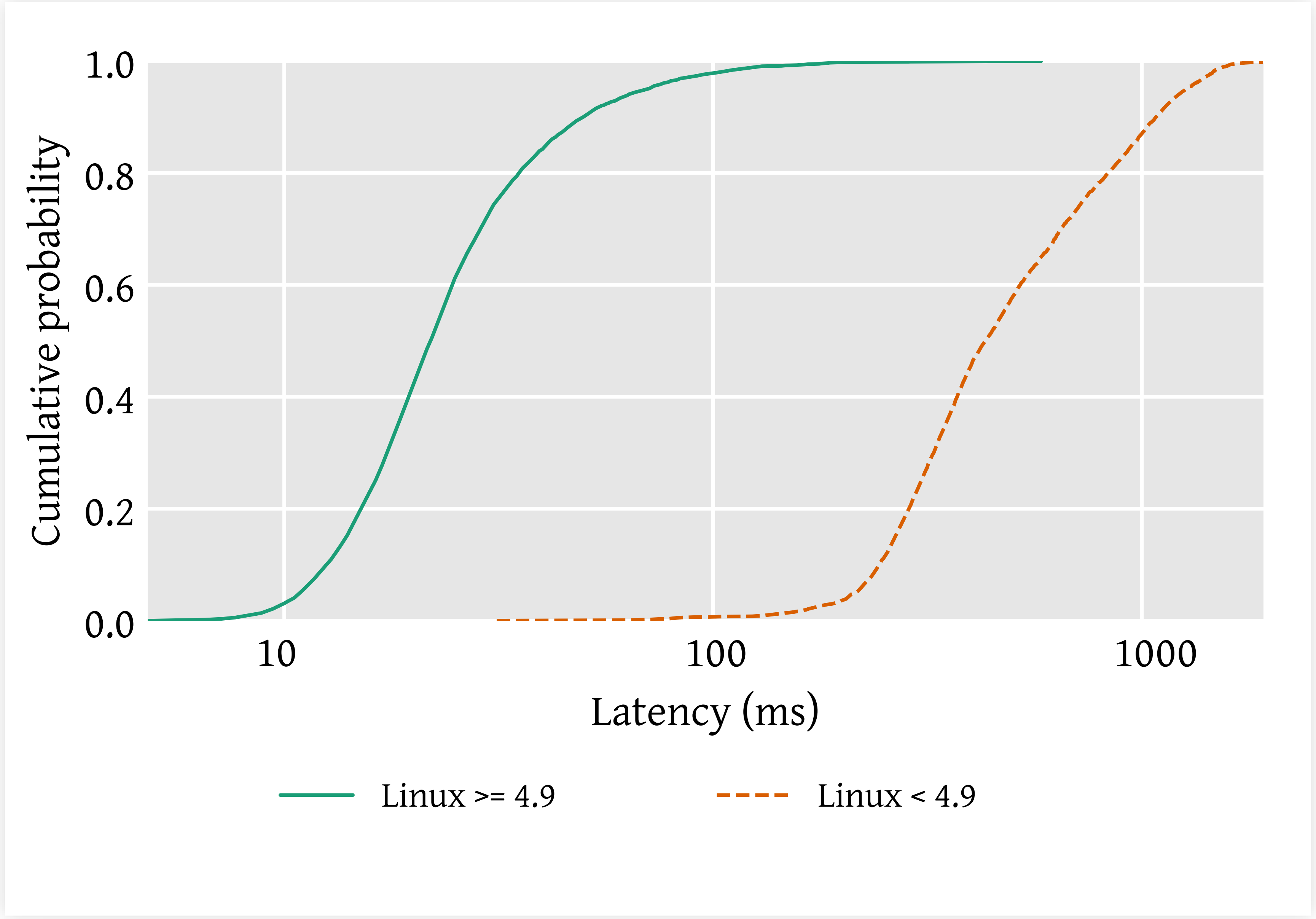
1. We must handle aggregation per Traffic ID (TID)
2. We must handle re-injection of packets for retransmission
3. We must be able to keep the hardware busy
4. We must support low-power access points (down to 32MB of RAM)
5. We cannot modify clients

Also, some operations are sensible to reordering (crypto, seqno)

1 & 2 means we can't use a qdisc

WHAT IS ALREADY IN MAINLINE?

- WiFi bufferbloat reduced by an order of magnitude
- Almost perfect airtime fairness
- Support in ath9k and ath10k (partial)



SO HOW DID WE DO IT?

We **increased** the amount of queueing in the WiFi stack by a factor of 16.

Queue smarter, not harder!

SERIOUSLY, HOW DOES IT WORK?

We designed a **queueing structure** and an **airtime fairness scheduler**.

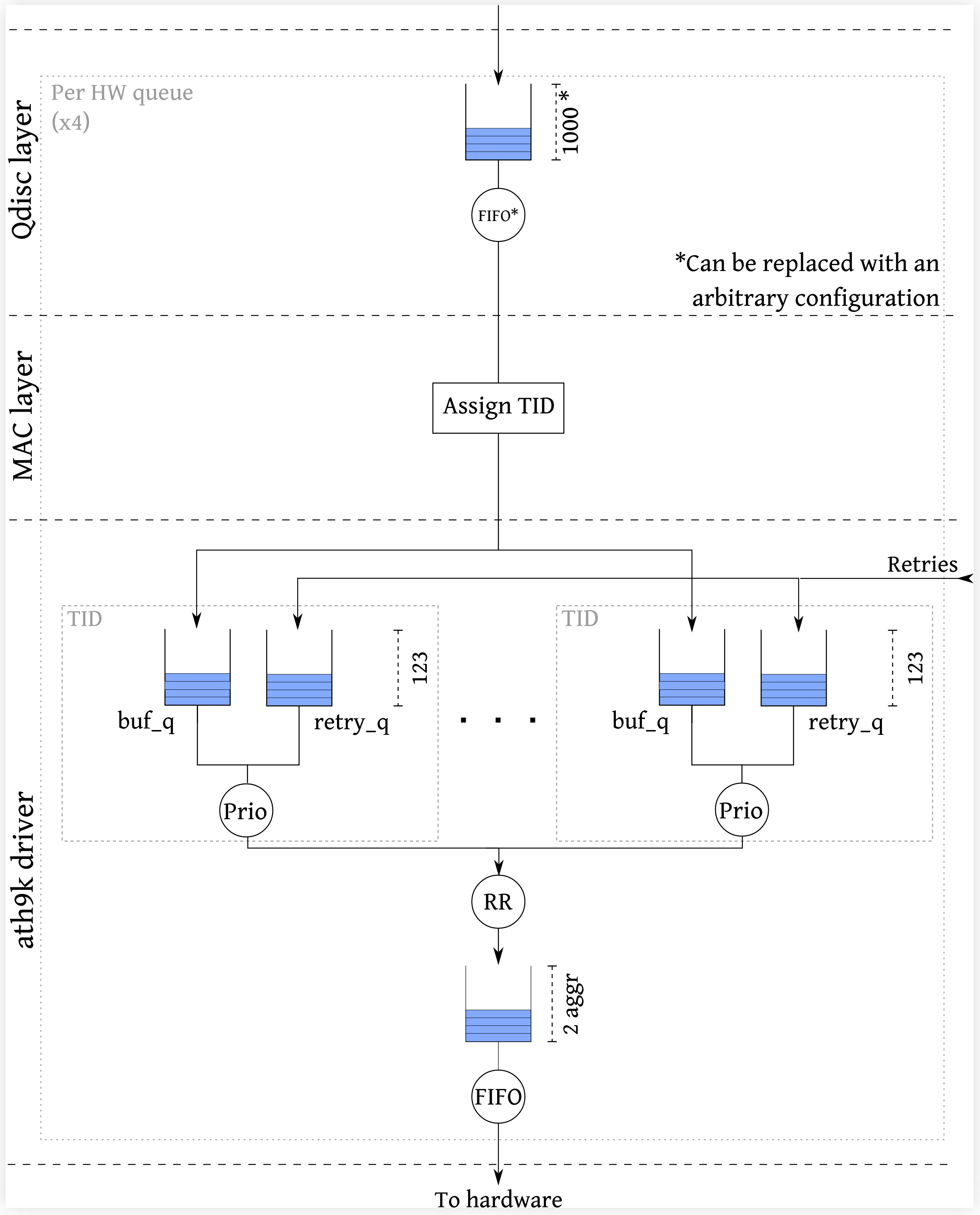
QUEUEING STRUCTURE

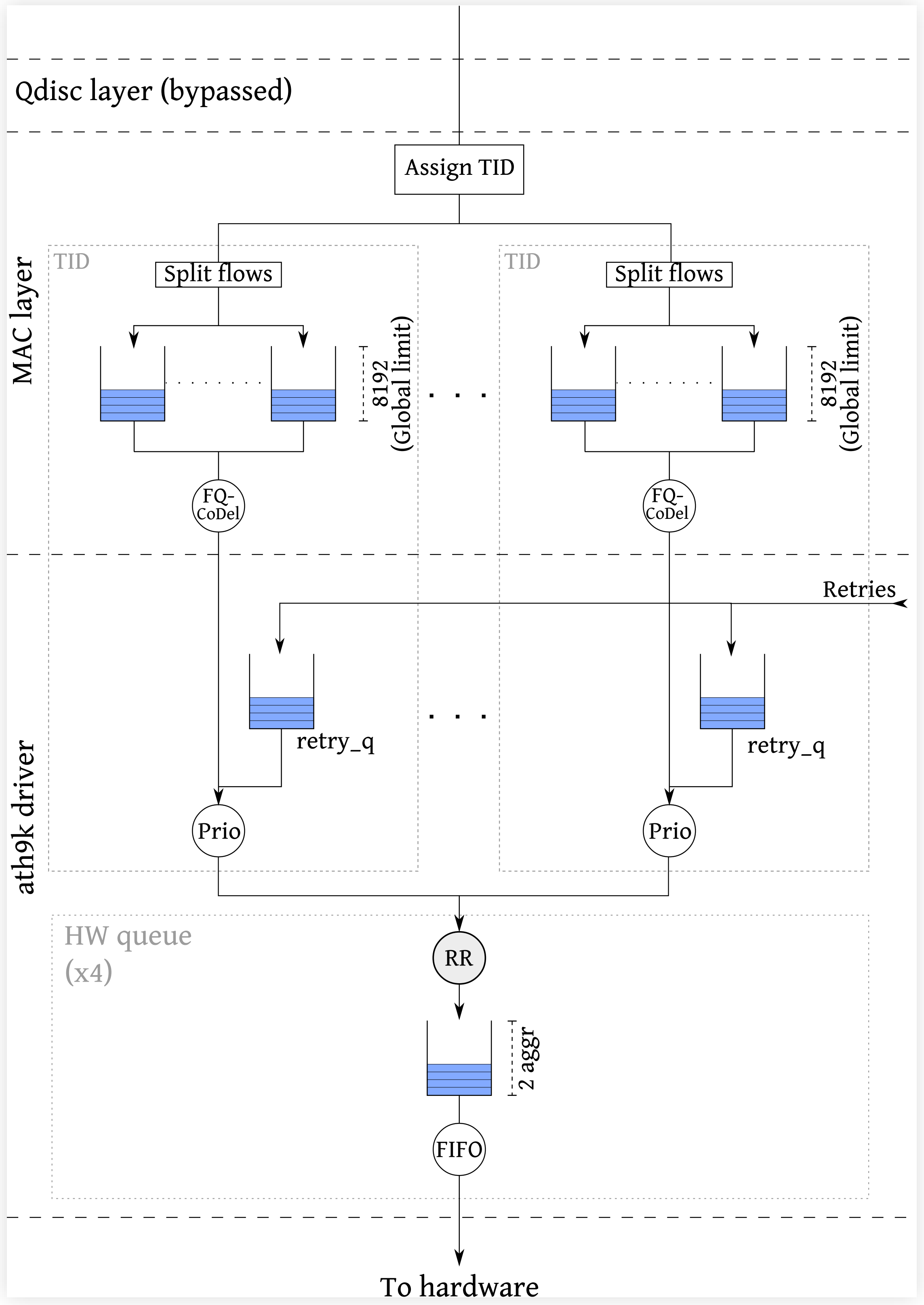
- Per-flow queueing based on FQ-CoDel
- Shared pool of queues to avoid memory explosion
- Supports per-TID dequeueing and scheduling

AIRTIME SCHEDULER

- Measure actual airtime usage of each station
- Run a DRR-based scheduler to even them out
- Optimise for sparse stations

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>



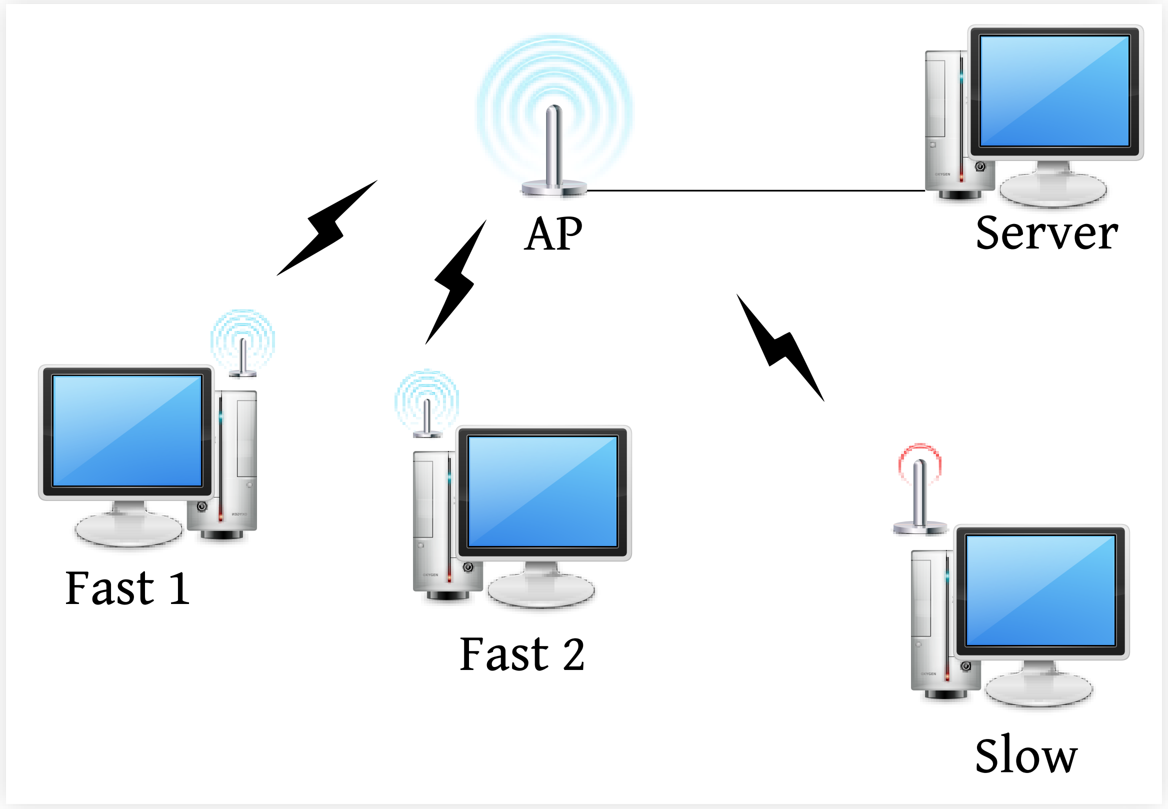


WiFi queueing structure (ath9k) before and after the redesign.

EVALUATION

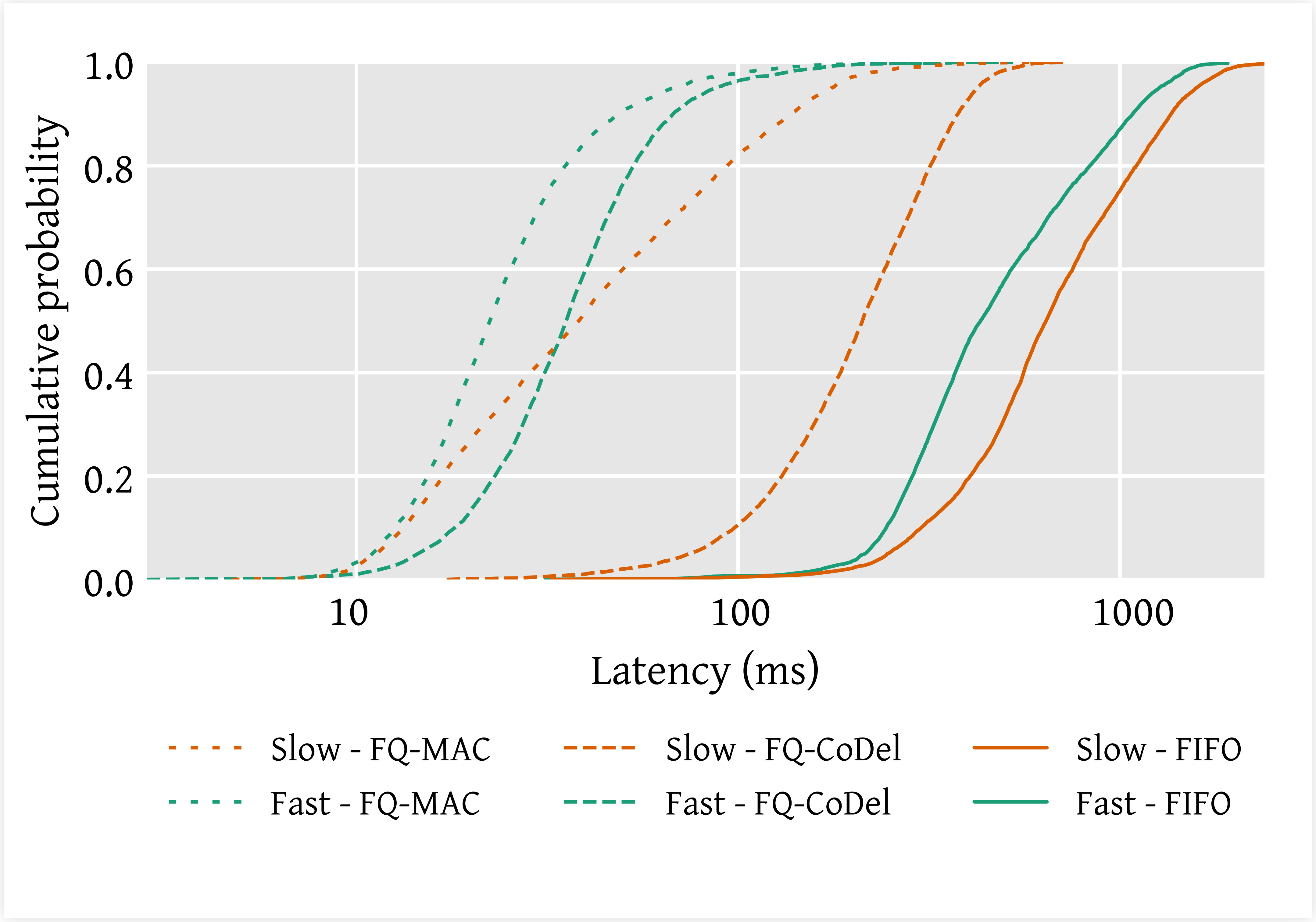
Four scenarios:

- **FIFO**: Default before modifications
- **FQ-CoDel**: FQ-CoDel qdisc on WiFi interface
- **FQ-MAC**: Our restructured MAC layer queues
- **Airtime fairness**: FQ-MAC + airtime fairness scheduler



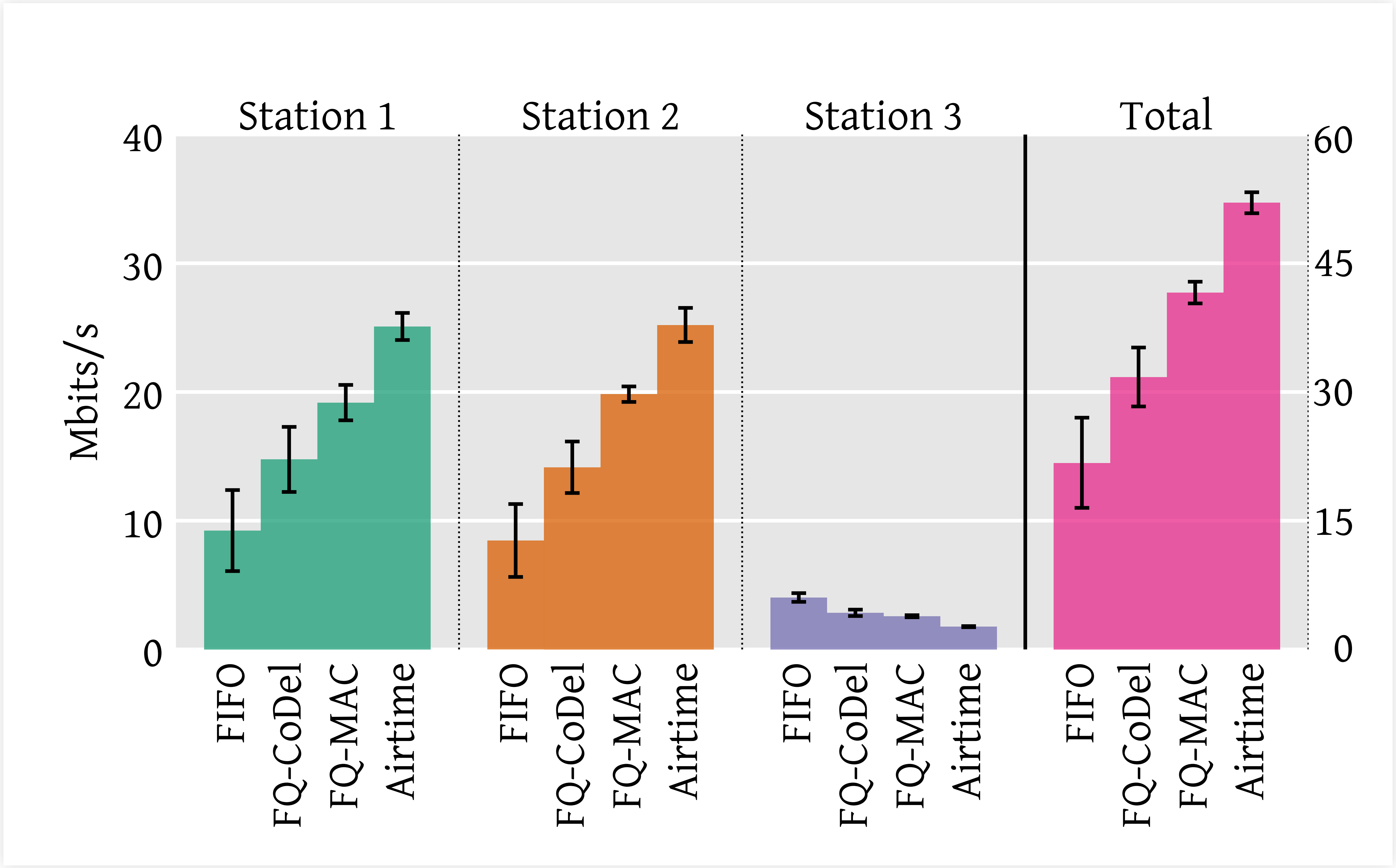
Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

LATENCY (TCP)



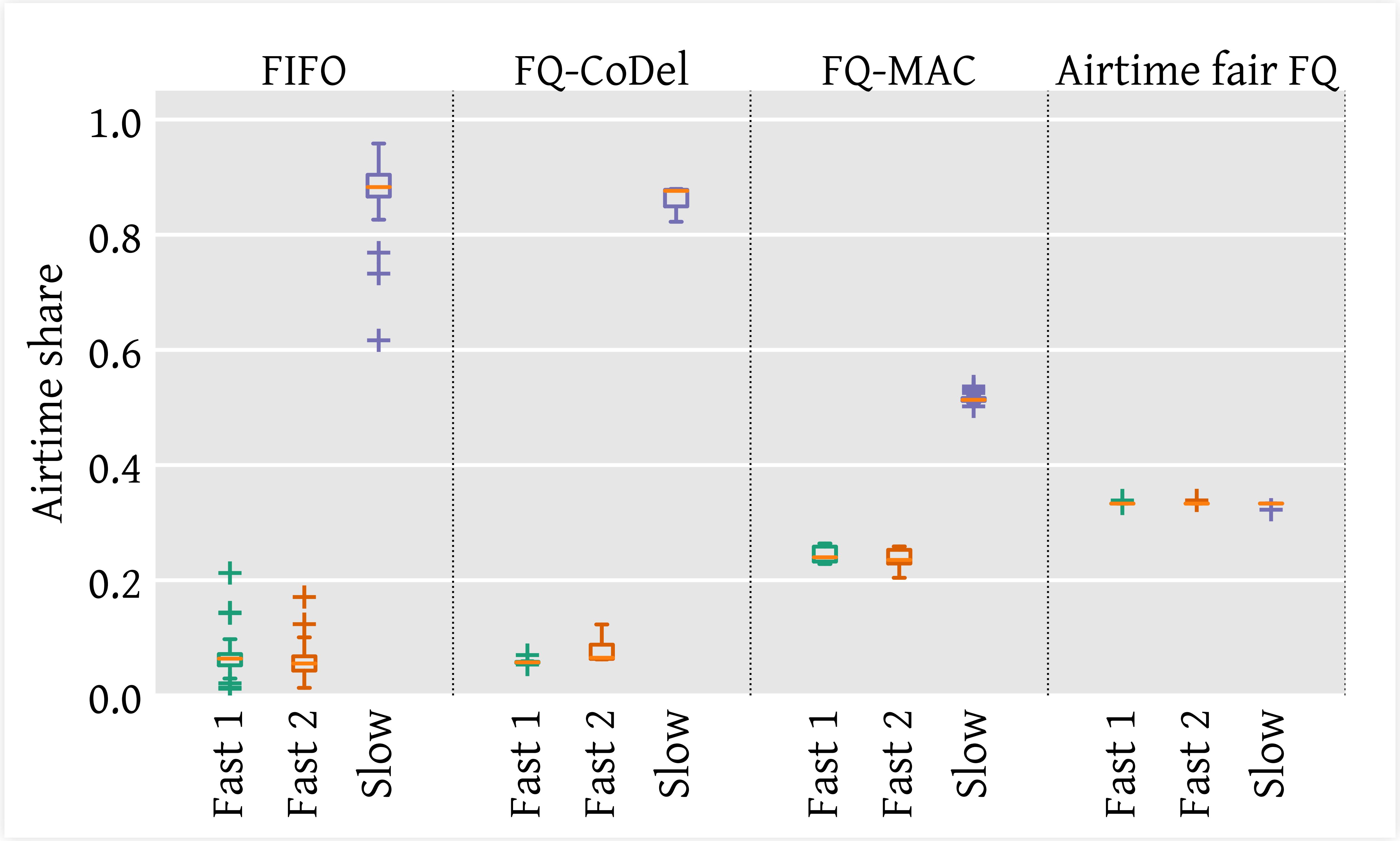
COMPUTER SCIENCE
DATAVETENSKAP

THROUGHPUT (TCP)



Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

AIRTIME (UDP)


FIFO
FQ-CoDel
FQ-MAC
Airtime fair FQ
Airtime share
1.0
0.8
0.6
0.4
0.2
0.0
Fast 1
Fast 2
Slow

NEXT STEPS

- More latency reductions
- Airtime policies
- QoS handling
- Configurability

Feedback wanted!

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

MORE LATENCY REDUCTIONS

- Minimise hardware buffering
  - When aggregation is in firmware: BQL-like mechanism
  - Otherwise: Precise aggregate scheduling
  - Interrupt at TX start?
- Time-based max retransmission counter
  - Maybe include queueing time?
- Dynamic aggregate sizing
  - Many stations active → Smaller aggregates
  - Throughput/latency tradeoff; what's the right one?

AIRTIME POLICIES

Strict fairness generally desirable, with a few exceptions. Such as:

- When the wireless music player is in the next room
  - A bit more airtime can make it work
- When a group of clients should be limited
  - E.g., limiting a guest network

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

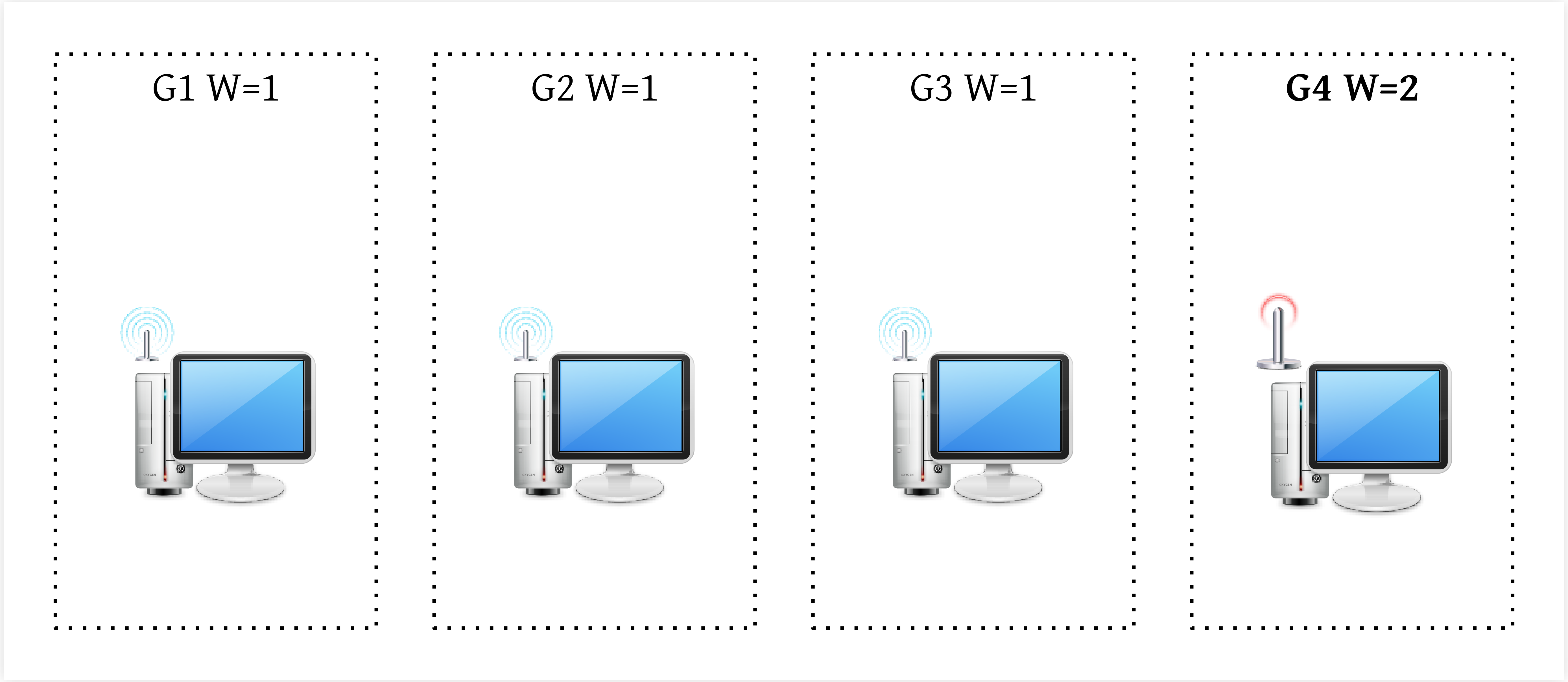
AIRTIME POLICIES - MECHAMISM

Idea: Implement scheduling groups

- Stations can be assigned to groups (by userspace)
- Airtime is divided between groups, then between stations within
- Groups could also be recursive (station is always its own group)
- Groups can be weighted

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

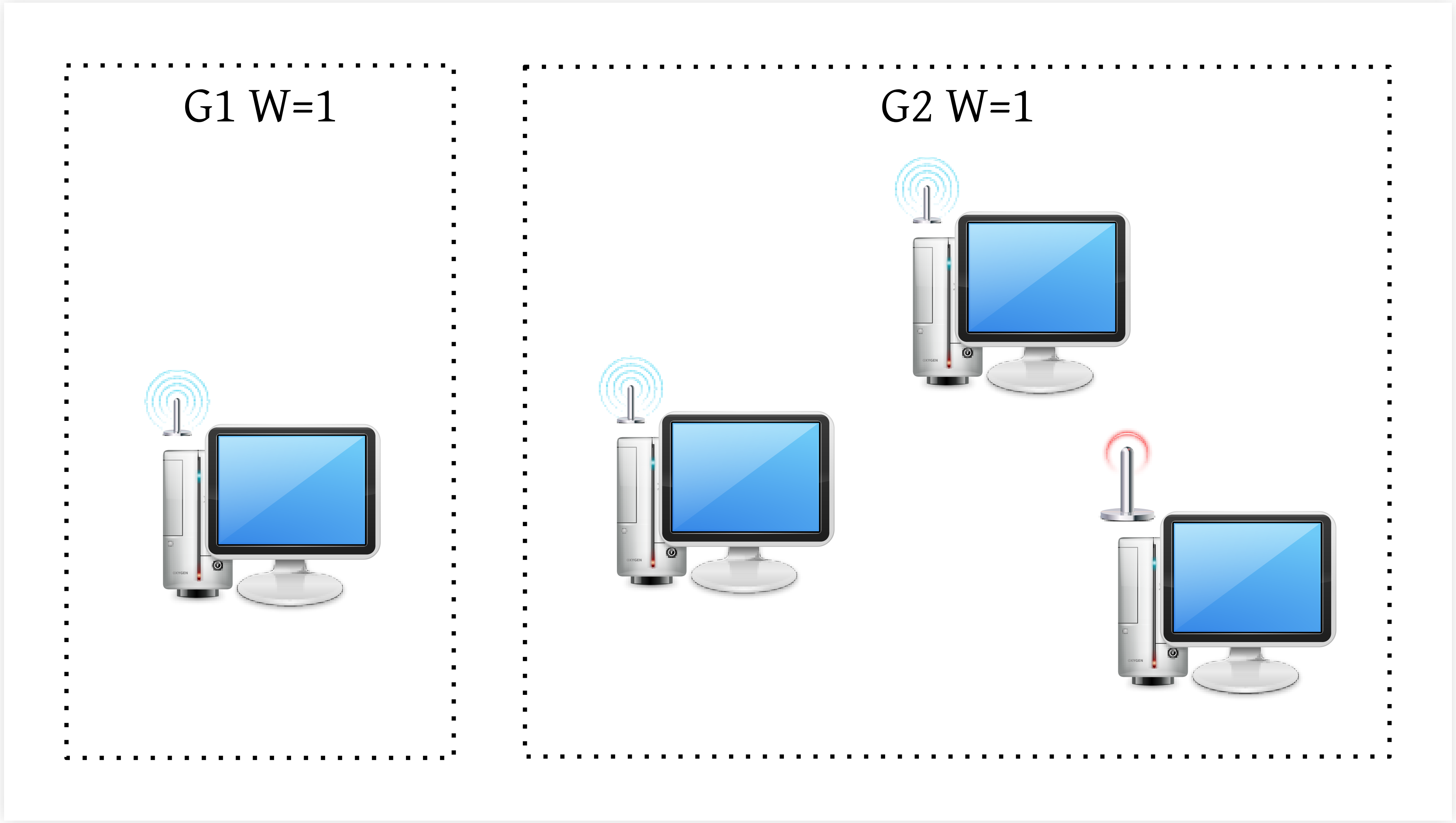
EX: PRIORITISE SLOW STATION



The slow station gets twice its fair airtime share

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

EX: LIMIT GUEST NETWORK



The guest network (G2) gets only half the airtime

But what if the guest network is G1?

AIRTIME POLICIES - SUMMARY

- Grouping mechanism pretty expressive, but some limitations
- Alternative: Just allow userspace to divide airtime - How? BPF?
- Maybe need to move scheduling and accounting out of the fast path.
- Prereq: Move airtime scheduler to mac80211 (patch series)

Comments?

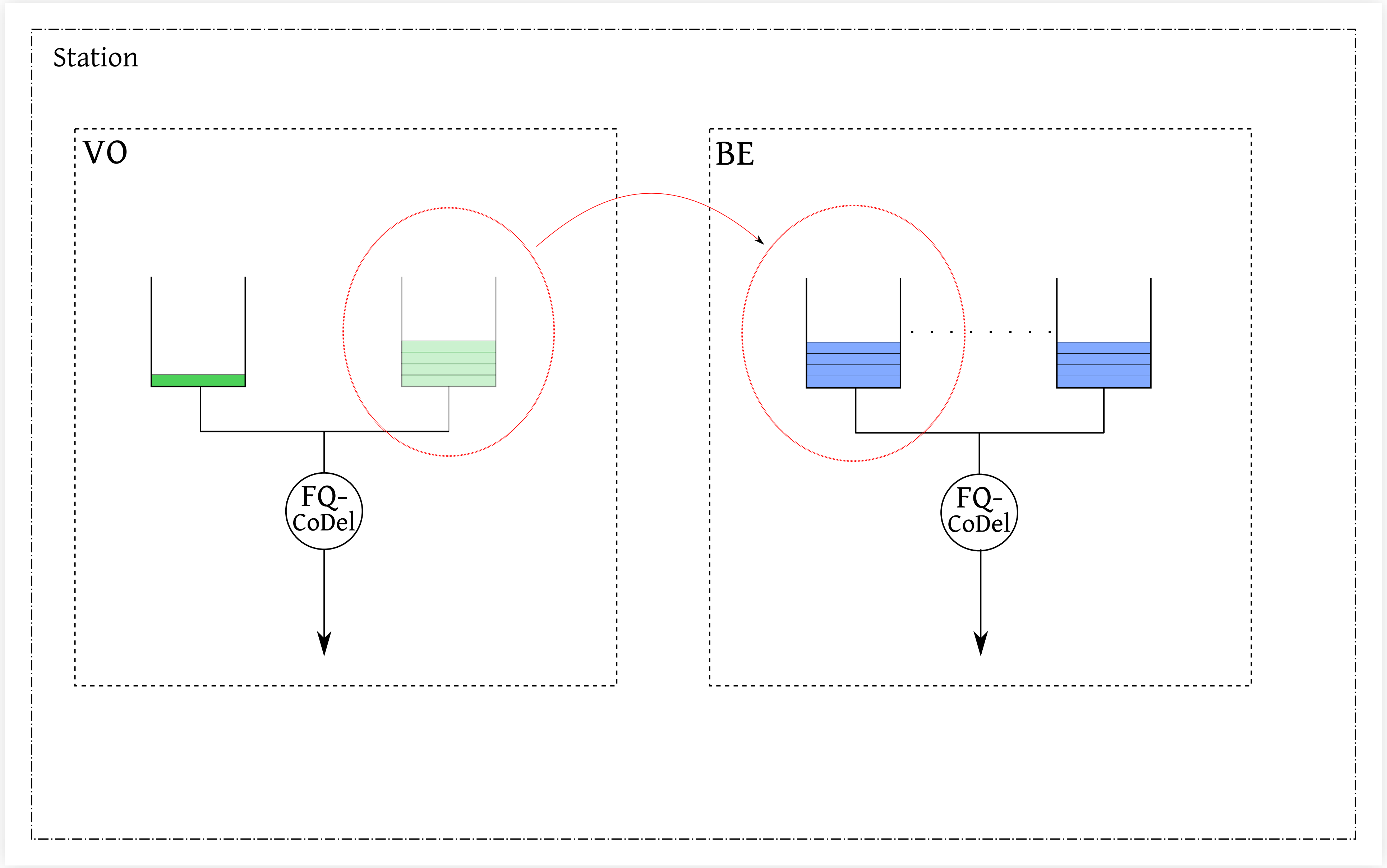
Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

QOS HANDLING

Some potential issues with current QoS handling:

- No admission control - potential for lockout
- Strict priority can be inefficient
- Interactions with airtime fairness

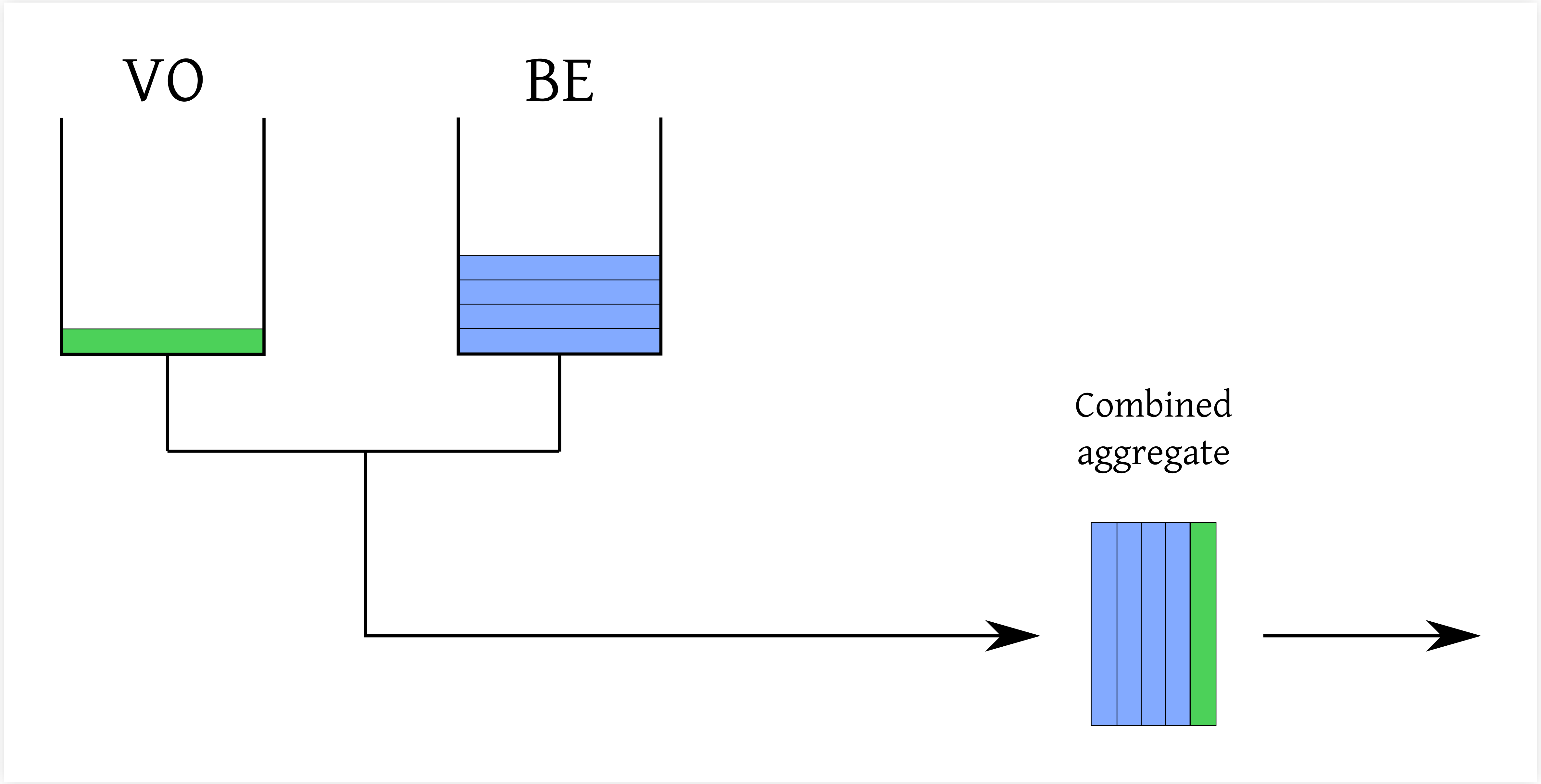
SOFT QOS ADMISSION CONTROL



Idea: Demote entire flow if it builds a queue.

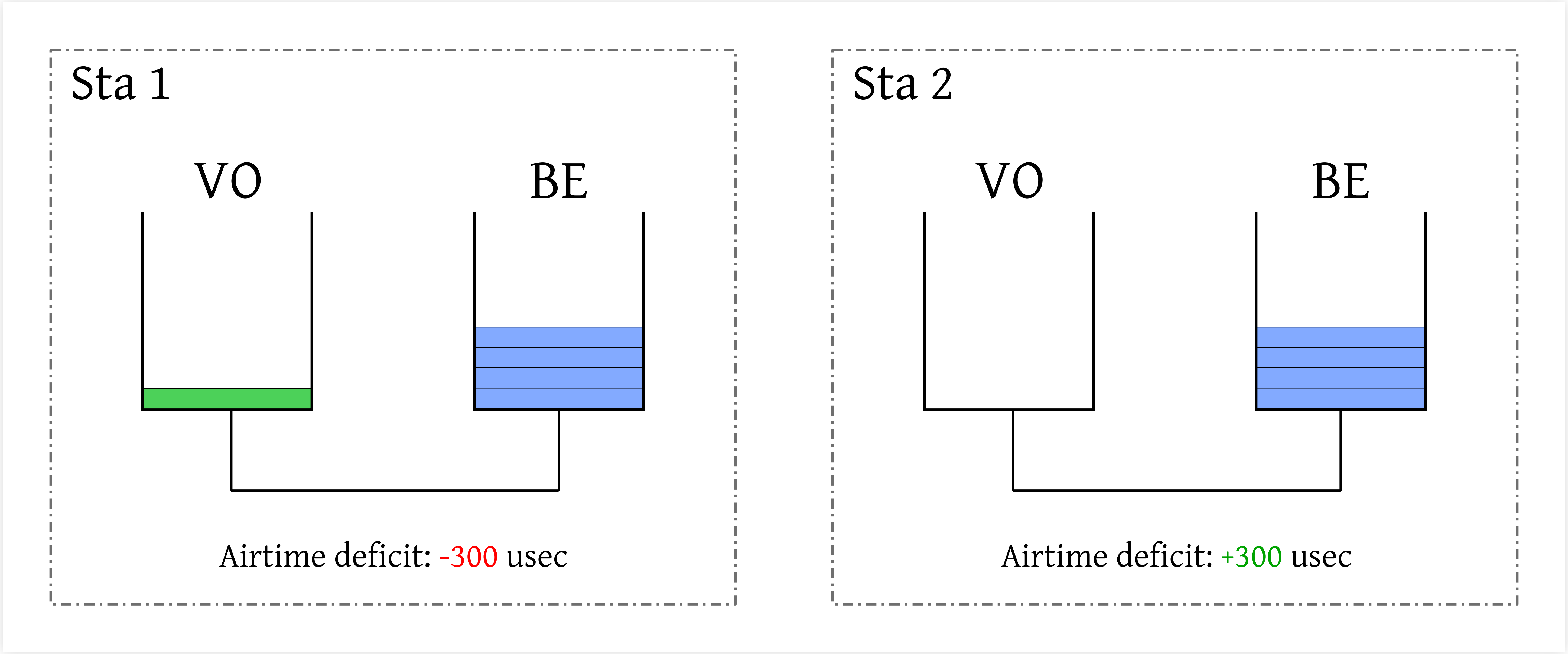
Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

COMBINING QOS LEVELS



Idea: Combine packets from different QoS levels in single aggregate.

QOS VS AIRTIME FAIRNESS



Which station goes first?

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

CONFIGURABILITY

Currently, everything is in debugfs.
How best to integrate with existing tools?

Configuration (per phy):

- FQ knobs (packet/memory limit, quantum)
- Airtime flags (count airtime on TX/RX)

Statistics (per sta):

- FQ per-tid stats (drops/marks/bytes etc)
- FQ multicast stats (per vif)
- Airtime stats (TX/RX usecs, deficit)

SUMMARY

We have:

- Reduced WiFi bufferbloat by an order of magnitude
- Achieved almost perfect airtime fairness in most cases

Going forward:

- More latency reductions
- Airtime policies
- QoS handling
- Configurability

Original paper in USENIX ATC '17:

Ending the Anomaly: Achieving Low Latency and Airtime Fairness in WiFi

with Michał Kazior, Dave Täht, Per Hurtig and Anna Brunstrom.

Many thanks to Sven Eckelmann, Simon Wunderlich, Felix Fietkau, Tim Shepard, Eric Dumazet, Johannes Berg, Kalle Valo, and the numerous other contributors to the Make-Wifi-Fast and LEDE projects.

EXTRA SLIDES

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

HOW TO GET AIRTIME FAIRNESS IN YOUR DRIVER

Once the patches have been finalised and merged.

- Implement the `wake_tx_queue` driver op
- Set `AIRTIME_ACCOUNTING` HW flag
- Fill in airtime usage in `rx_status->airtime` and `tx_status->tx_time`
  - Should contain time spent on transmission in microseconds
  - Should include failed transmission attempts (on TX)
  - TX: Get from hardware/firmware. RX: Can be calculated

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

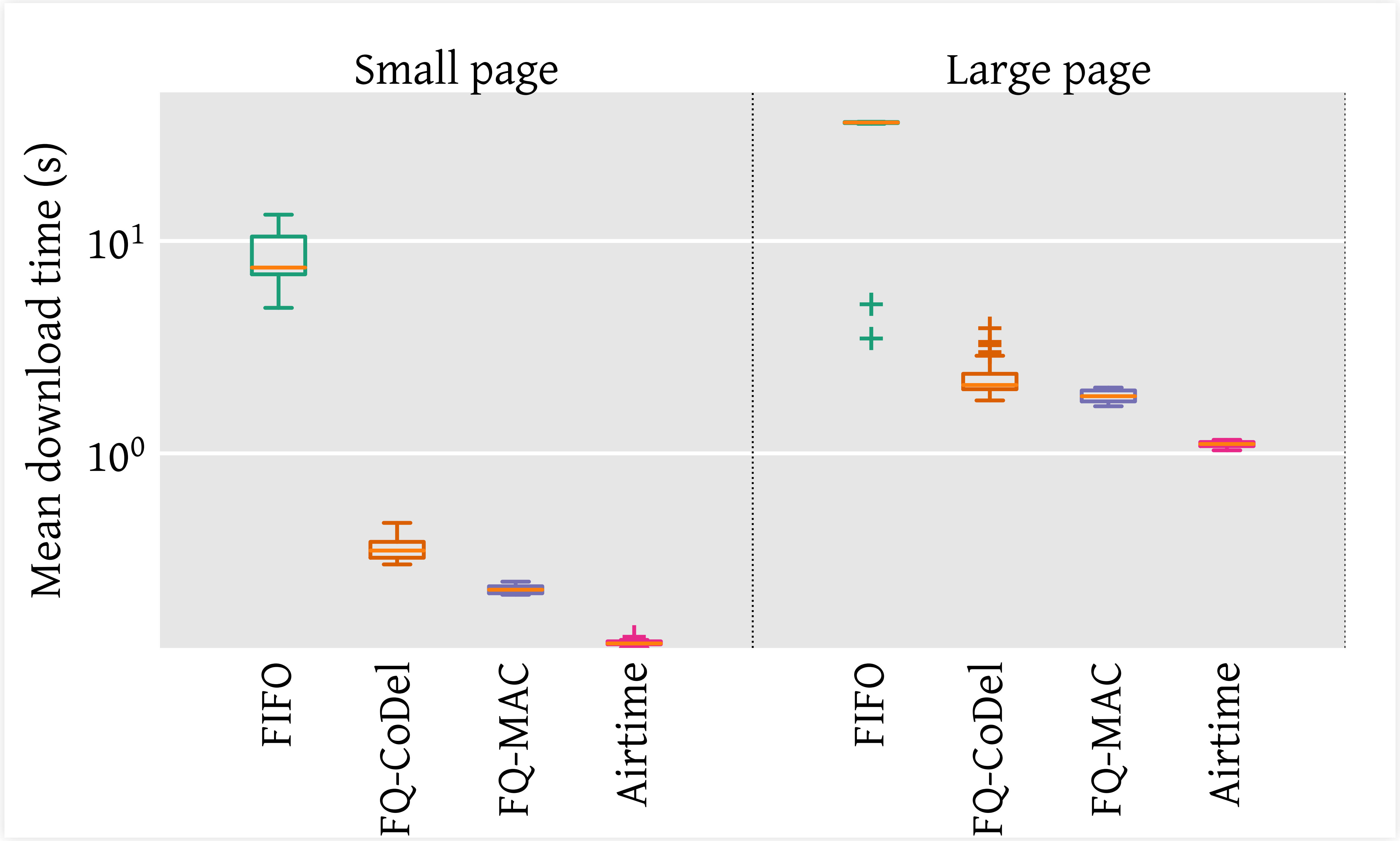
APPLICATION PERFORMANCE IMPACT

We evaluate:

- HTTP page load time performance
- VoIP performance (MOS values)

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

HTTP PAGE LOAD TIMES

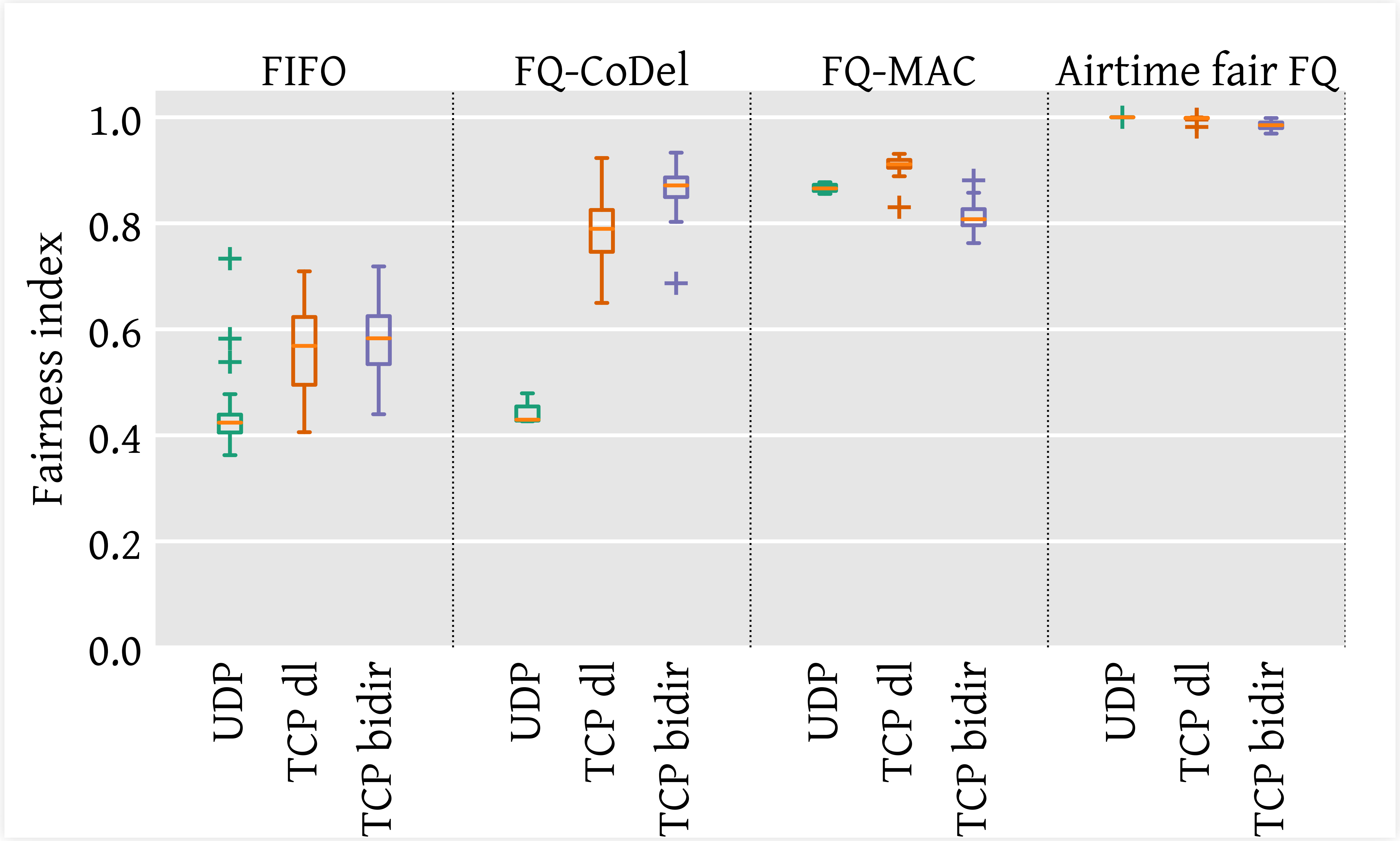


VOIP TEST

	QoS	MOS	Thrp	MOS	Thrp
FIFO	VO	4.17	27.5	4.13	21.6
	BE	1.00	28.3	1.00	22.0
FQ-CoDel	VO	4.17	25.5	4.08	15.2
	BE	1.24	23.6	1.21	15.1
FQ-MAQ	VO	4.41	39.1	4.38	28.5
	BE	4.39	43.8	4.37	34.0
Airtime	VO	4.41	56.3	4.38	49.8
	BE	4.39	57.0	4.37	49.7

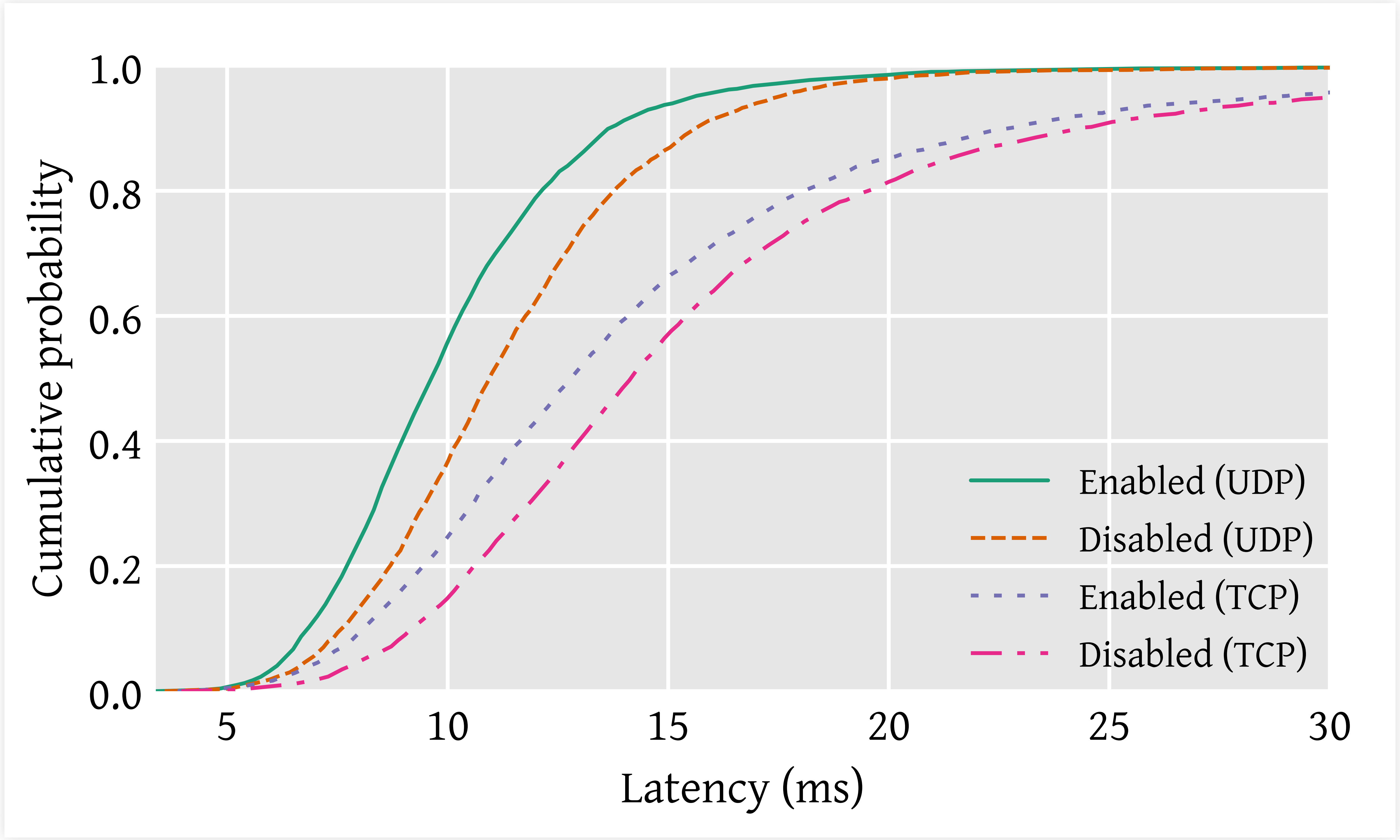
Synthetic MOS values calculated from the ITU-T G.107 E-model.

AIRTIME FAIRNESS



Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

SPARSE STATION OPTIMISATION

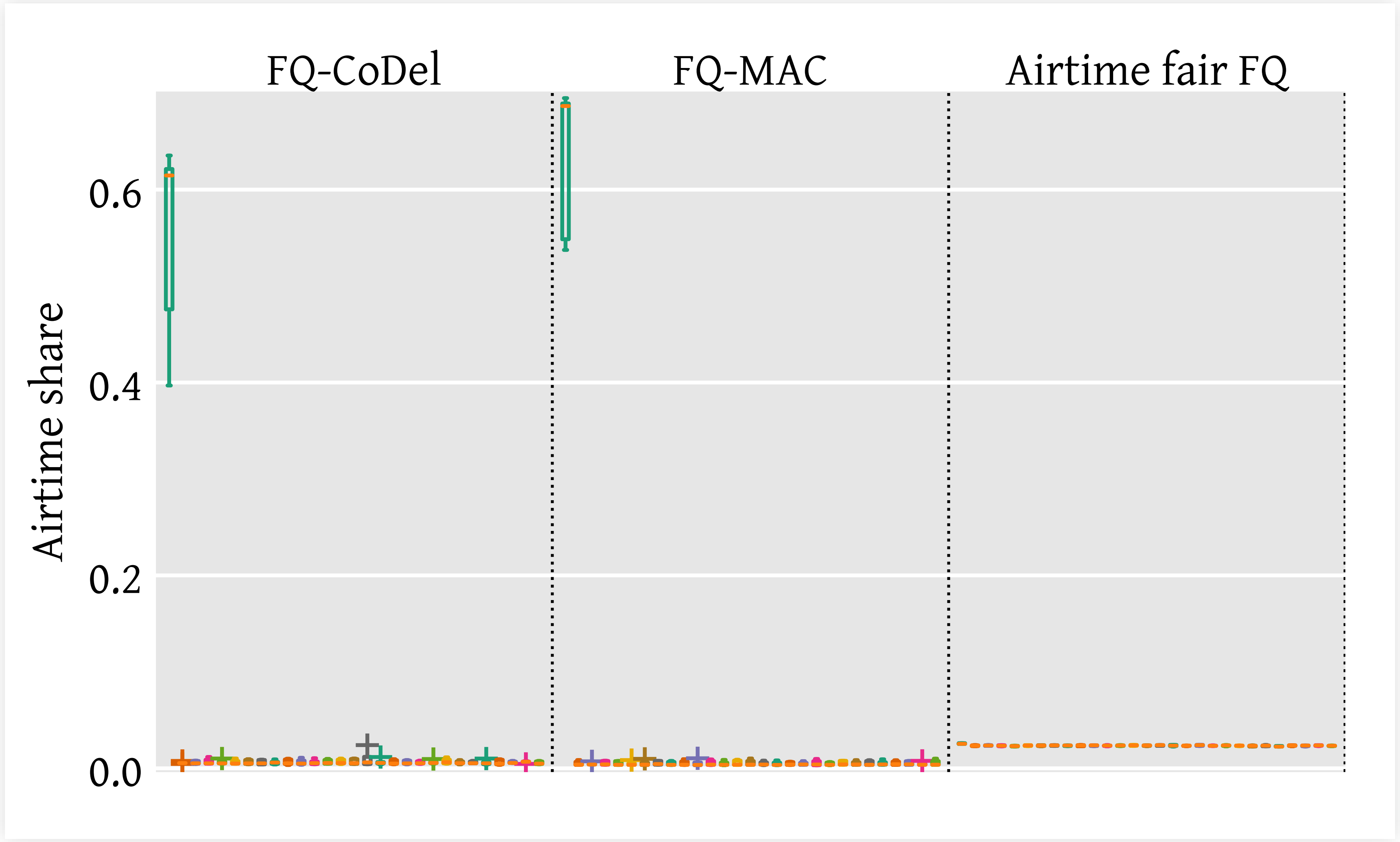


COMPUTER SCIENCE
DATAVETENSKAP

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

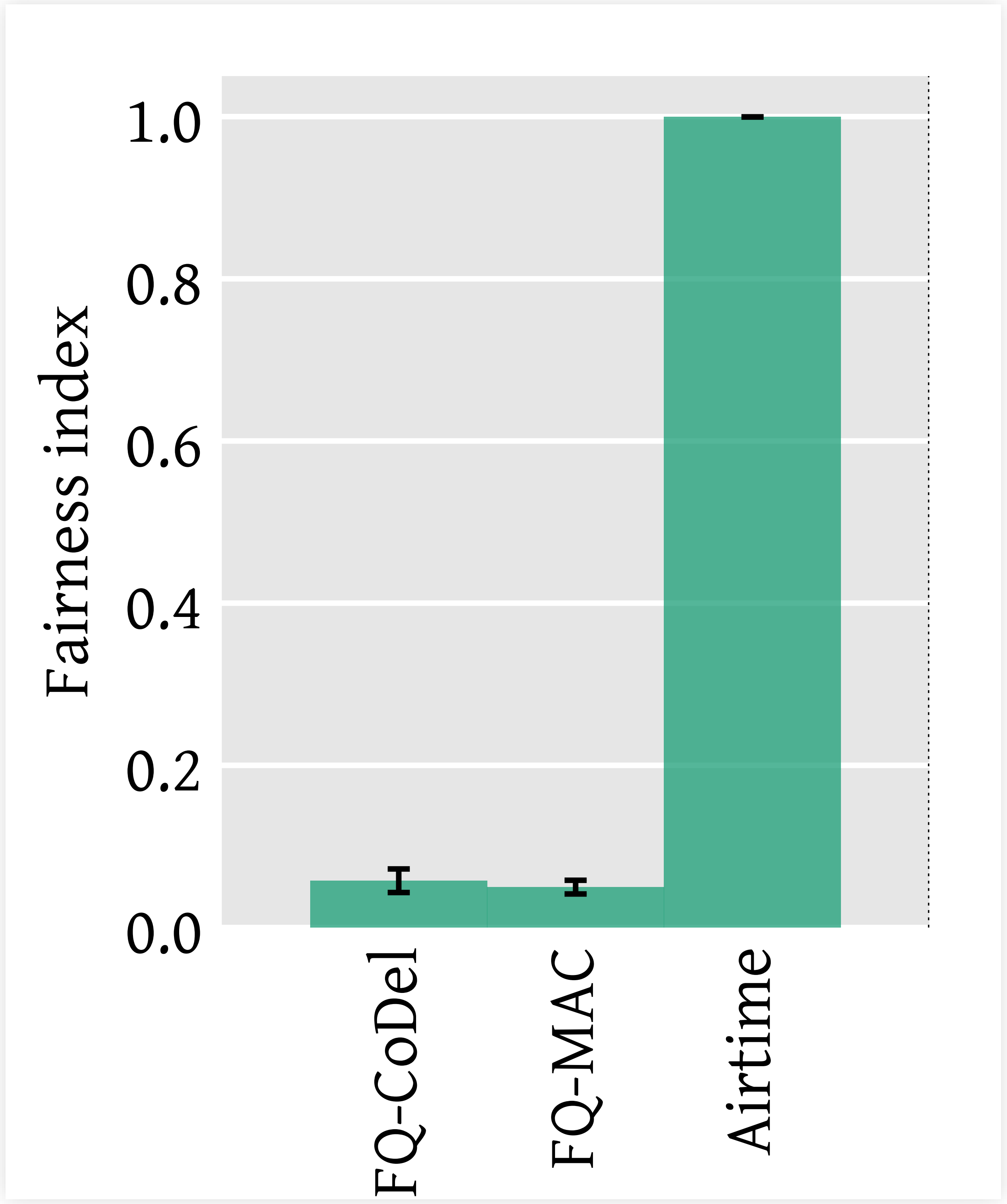
30 STATIONS TEST

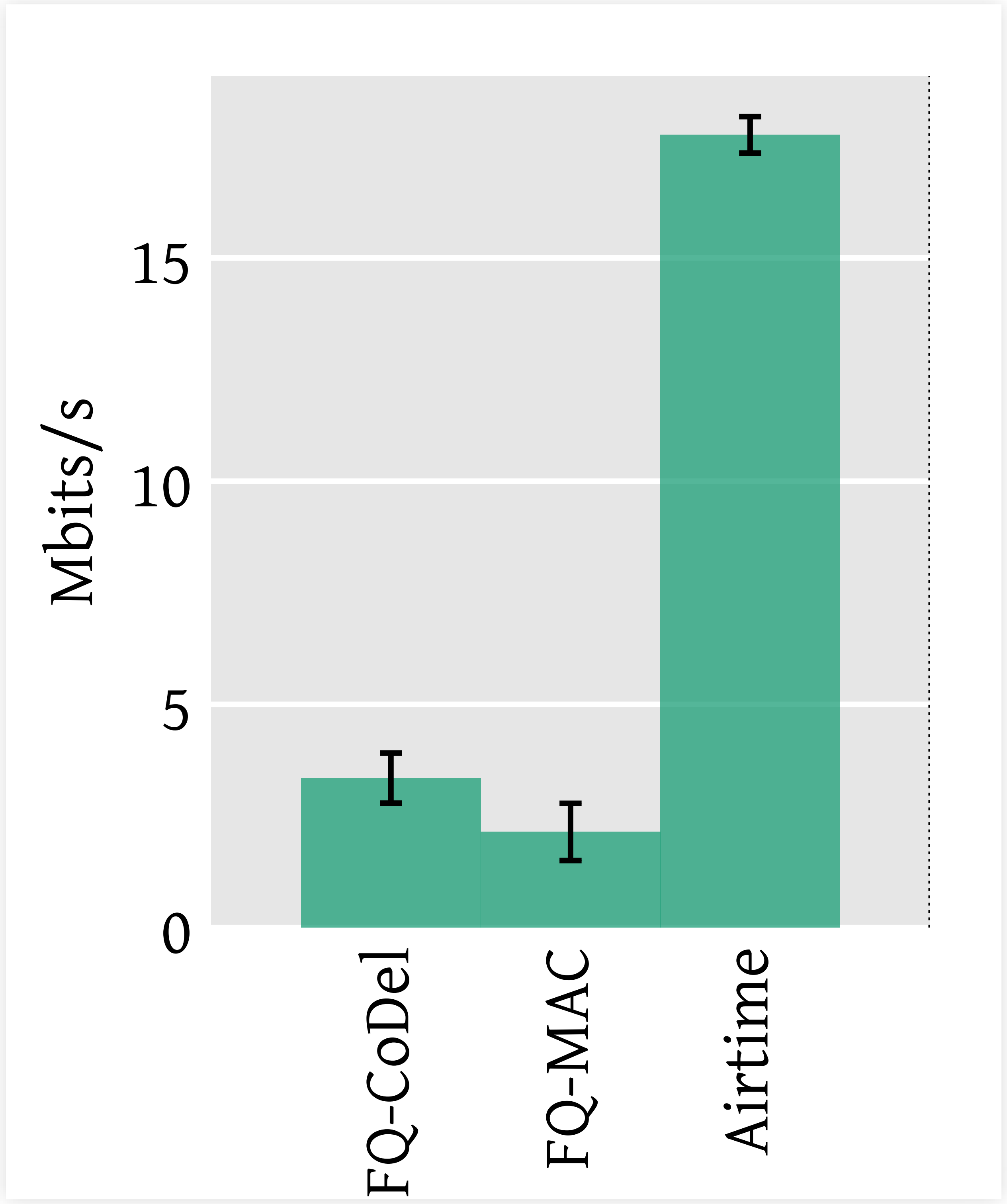
- We cooperated with another lab to evaluate our solution
- 30 station testbed, one slow station (1 Mbps)



Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se>

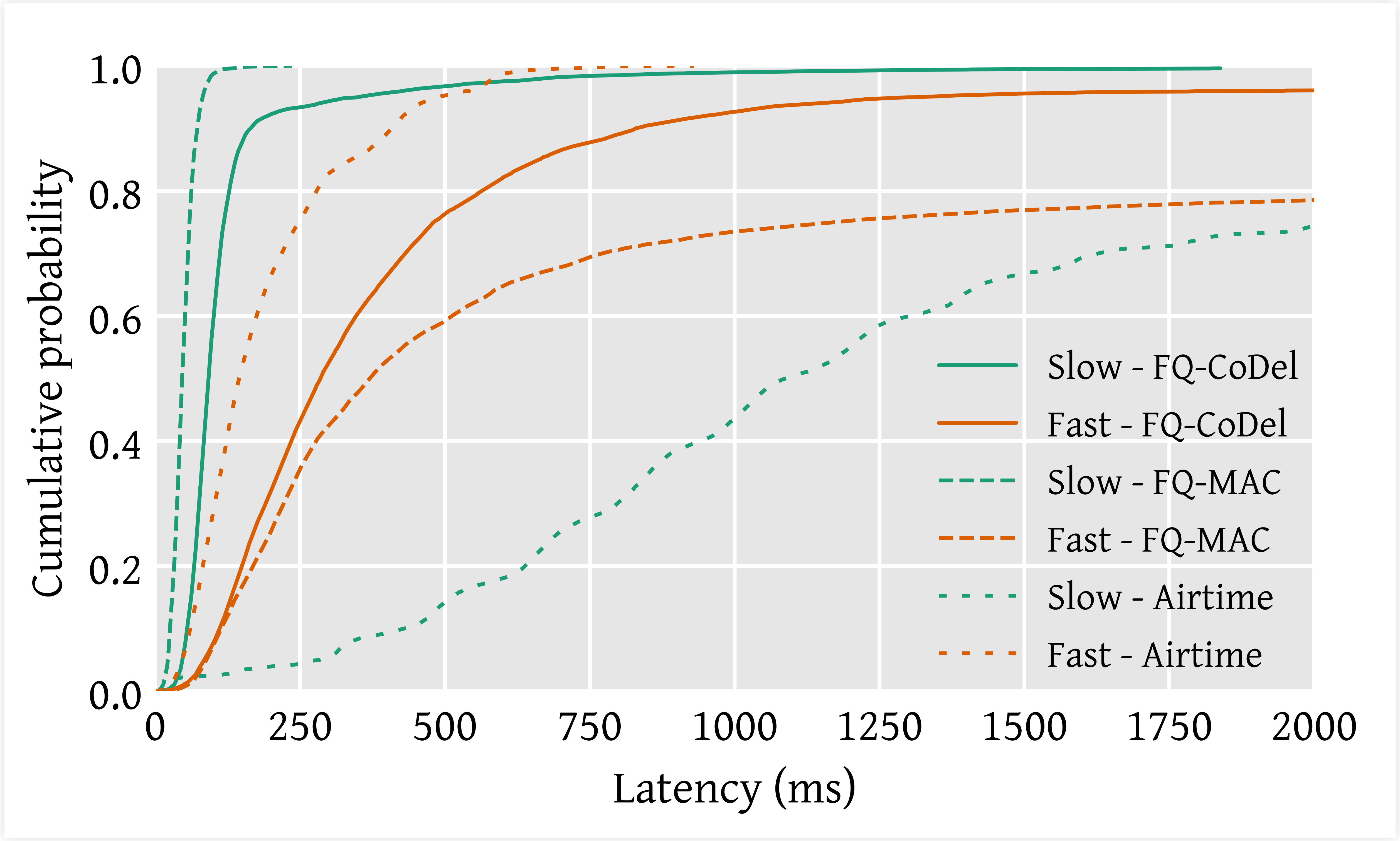
30 STATIONS





COMPUTER SCIENCE
DATAVETENSKAP

30 STATIONS - LATENCY



AIRTIME SCHEDULER

```
function on_tx(pkt) {
  station = get_station(pkt)
  station.deficit -= pkt.duration
}

function on_rx(pkt) {
  station = get_station(pkt)
  station.deficit -= calc_dur(pkt)
}

function schedule(hwq) {
  if full(hwq) { return }

begin:
  station = list_head(station_list)

  if station.deficit <= 0 {
    station.deficit += quantum
    list_move_end(station, station_list)
    goto begin
  }
  if !station.queue {
    list_del(station)
    goto begin
  }
  queue_aggregate(station)
}
```

CONVERTING MAC80211 TO TXQS

Johannes outlined a plan for converting mac80211 to use TXQs:

- Introduce a compatibility layer
- Convert multicast PS buffering
- Use TXQs for offchannel frames
- Handle monitor mode
- Handle non-data frames for stations & vifs
- Remove all the now-dead code
- Convert more infrastructure to TXQs