# ENDING THE ANOMALY

## ACHIEVING *LOW LATENCY* AND *AIRTIME FAIRNESS* IN WIFI

**Toke Høiland-Jørgensen** (Karlstad University), Michał Kazior (Tieto Poland), Dave Täht (Teklibre), Per Hurtig and Anna Brunstrom (Karlstad University)

USENIX ATC 17, July 12-14, Santa Clara, CA

# OUTLINE

- The problem: Bufferbloat (and airtime fairness)
- Our solution: Queueing structure (and scheduler)
- Evaluation results
- Summary and questions

COMPUTER SCIENCE
DATAVETENSKAP

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

# BUFFERBLOAT

Bufferbloat is **pathological and persistent** queueing latency.

Solutions exist to **eliminate bufferbloat** in wired networks:

- The FQ-CoDel hybrid AQM/fairness queueing algorithm
- PIE and CoDel AQMs
- Others appearing

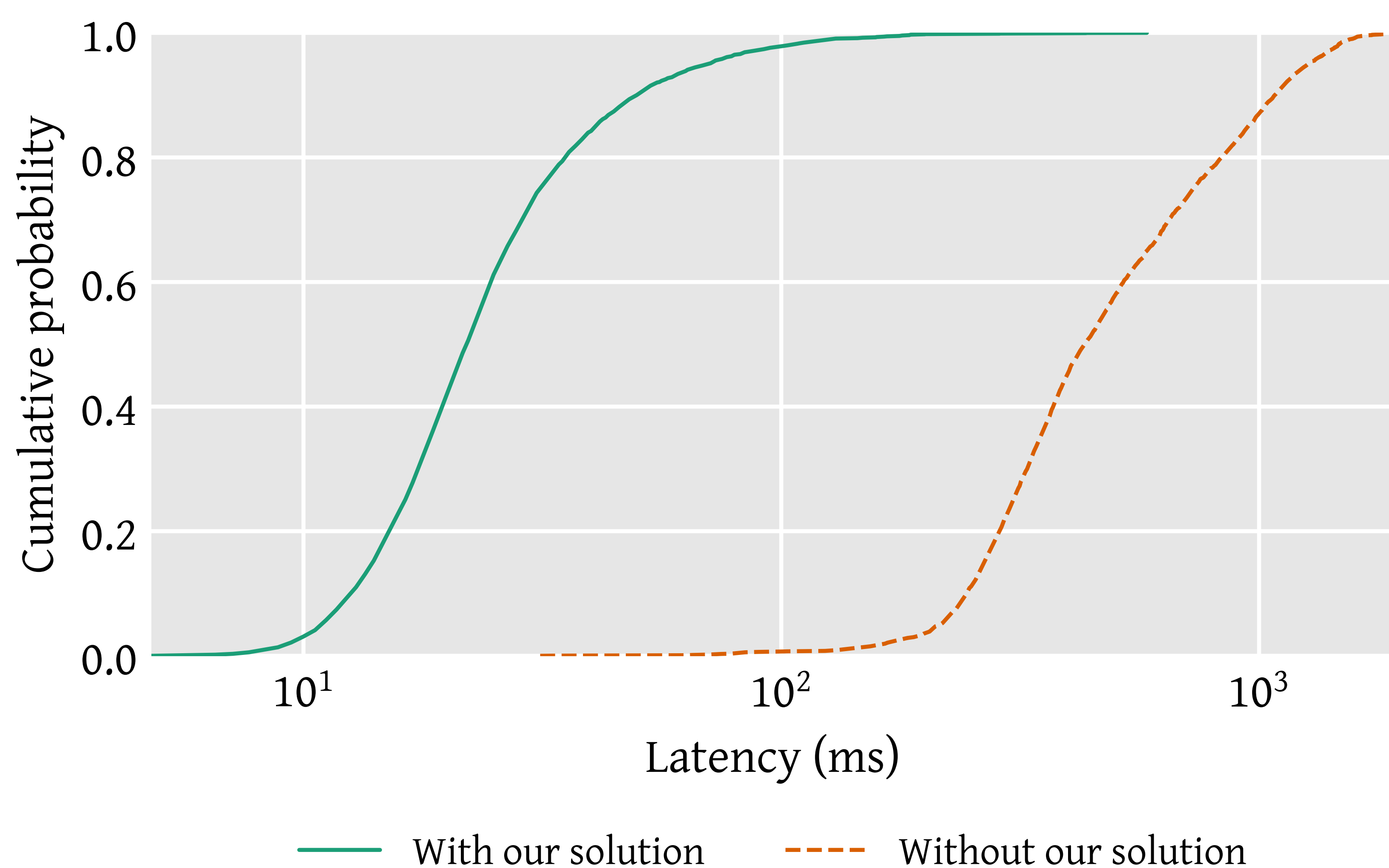
However, on WiFi there we still see **100s of milliseconds** of extra buffering in the stack.



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

# WHAT DID WE ACHIEVE?

- WiFi bufferbloat reduced by an order of magnitude
- (Almost perfect airtime fairness in most cases)
- Light-weight deployable solution, accepted in mainline Linux




COMPUTER SCIENCE
DATAVETENSKAP

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

# 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

1 & 2 means we can't use solutions for wired networks as-is.



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

# OUR SOLUTION

We design a **queueing structure** (and an **airtime fairness scheduler**).

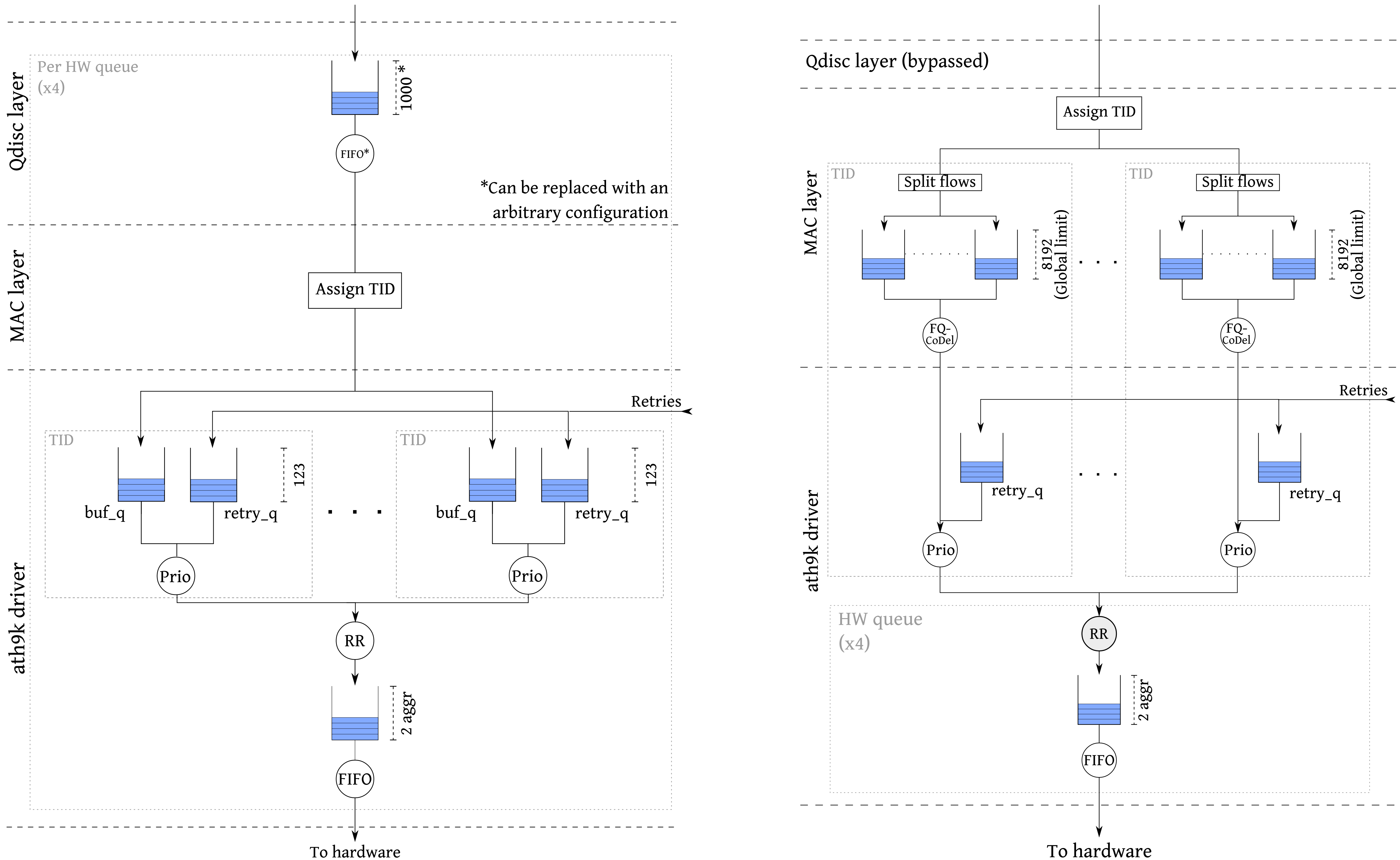
## QUEUEING STRUCTURE

- 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>




Linux kernel queueing structure before and after our modifications.

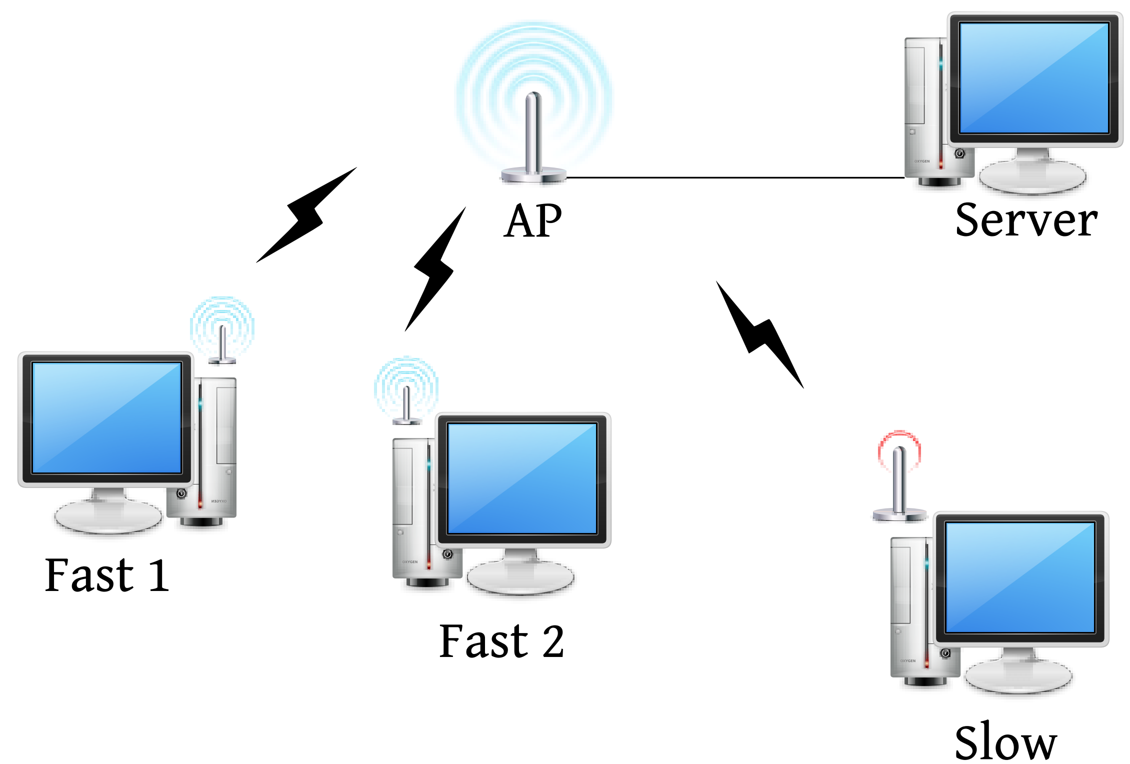
COMPUTER SCIENCE
DATAVETENSKAP

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

# EVALUATION

Four scenarios:

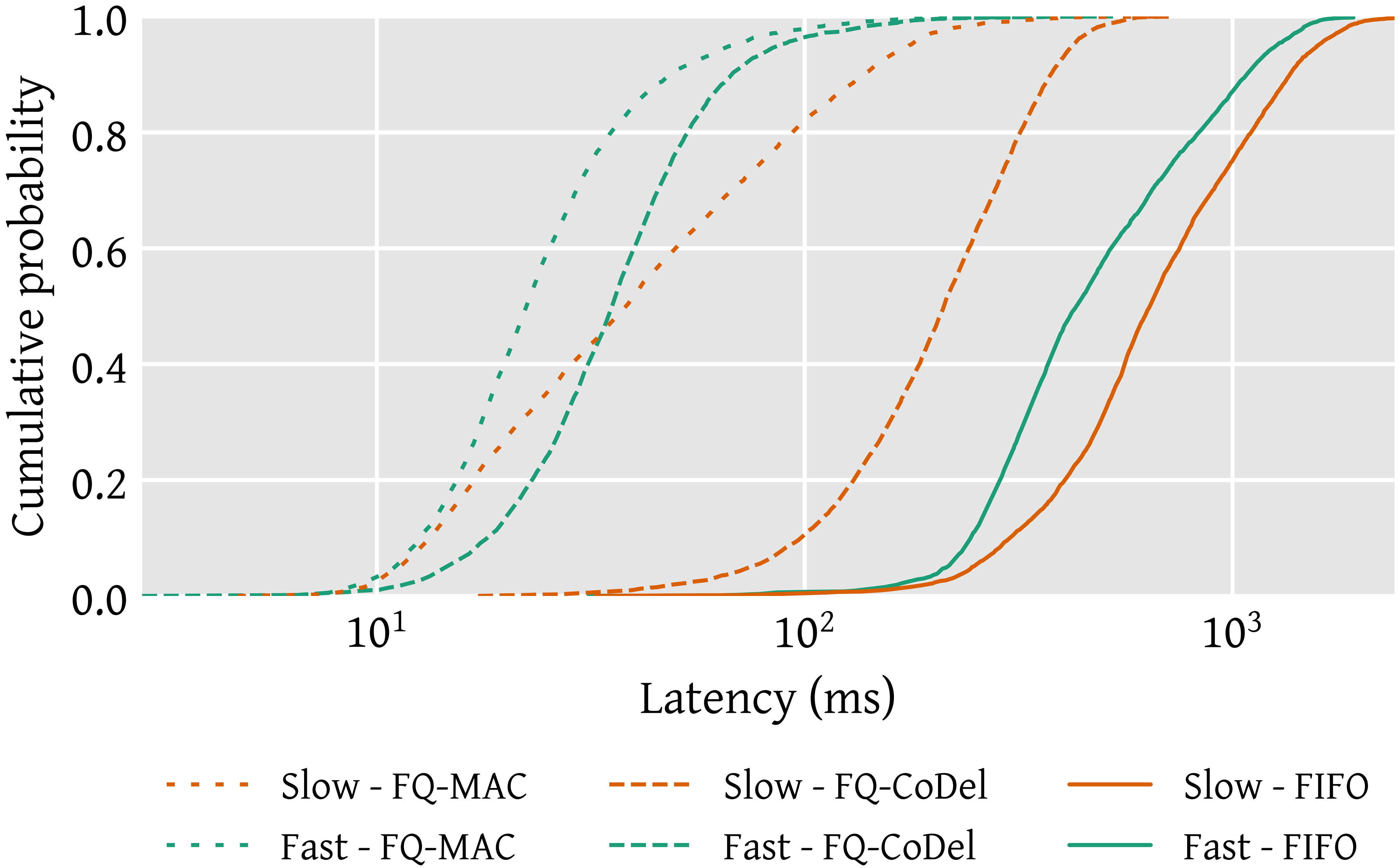
- **FIFO**: Default before modifications
- **FQ-CoDel**: FQ-CoDel qdisc on WiFi interface
- **FQ-MAC**: Our restructured MAC layer queues
- **Airtime fairness**: FQ-MAC + airtime fairness scheduler





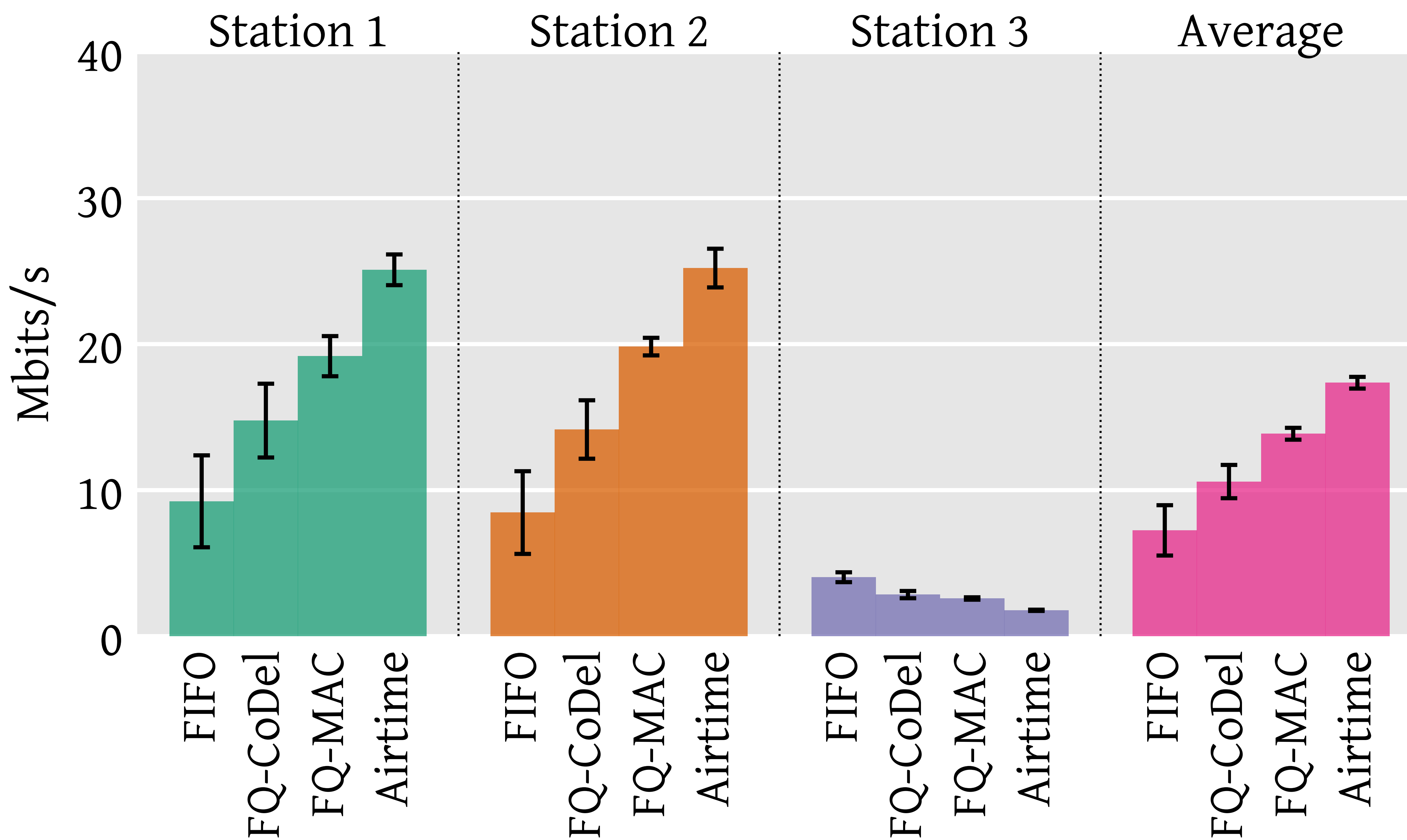

Toke Høiland-Jørgensen <toke.hoiland-jorgensen@kau.se> 

# LATENCY (TCP)




COMPUTER SCIENCE
DATAVETENSKAP

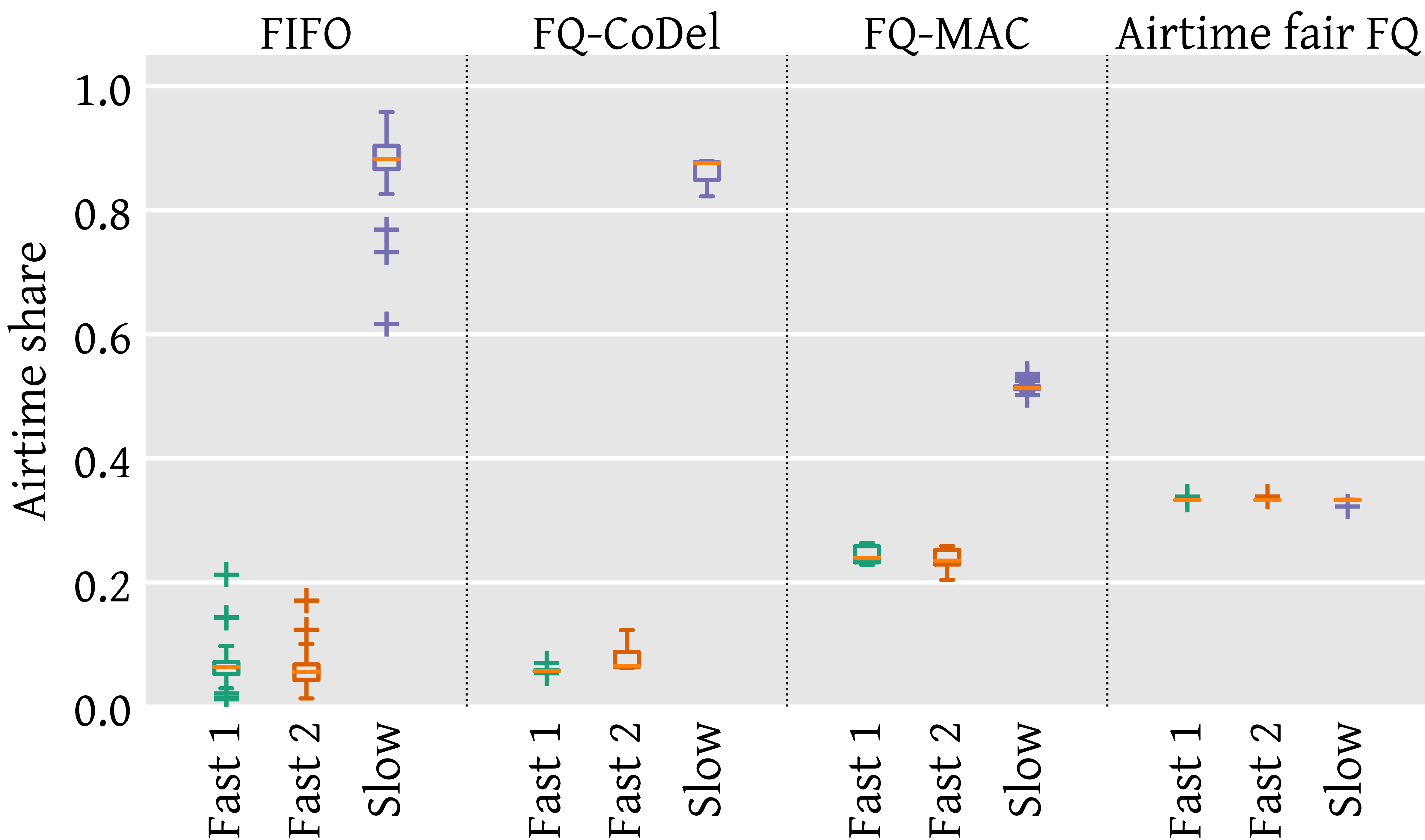
# THROUGHPUT (TCP)




COMPUTER SCIENCE
DATAVETENSKAP

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

# APPLICATION IMPACT

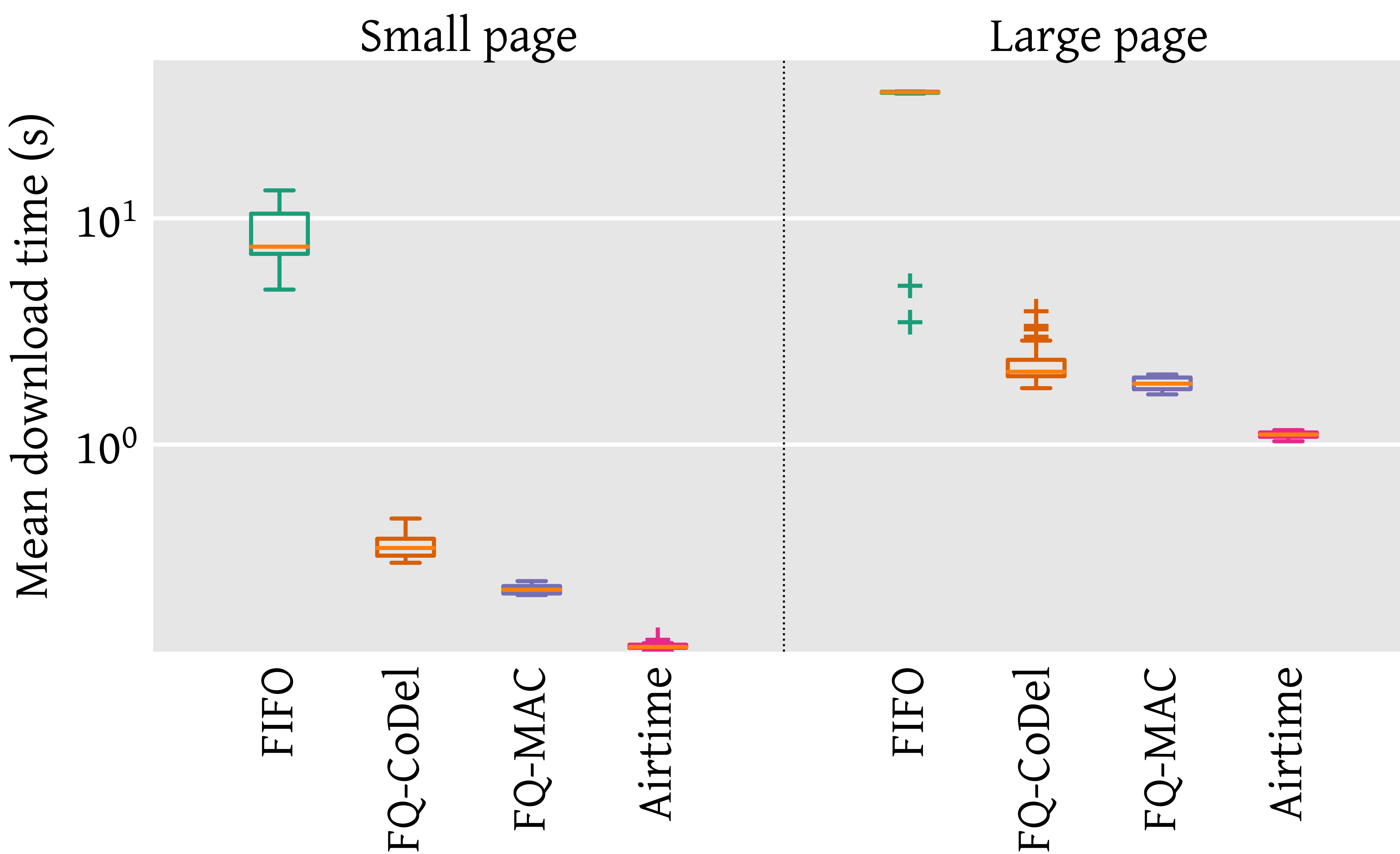
We evaluate:

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



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

# HTTP PAGE LOAD TIMES






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

# 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.



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

# SUMMARY

We have:

- Reduced WiFi bufferbloat by an order of magnitude
- Achieved almost perfect airtime fairness in most cases
- Created a light-weight, deployable solution, accepted in mainline Linux

Code, data and more graphs available at:

https://www.cs.kau.se/tohojo/airtime-fairness/

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



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.



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

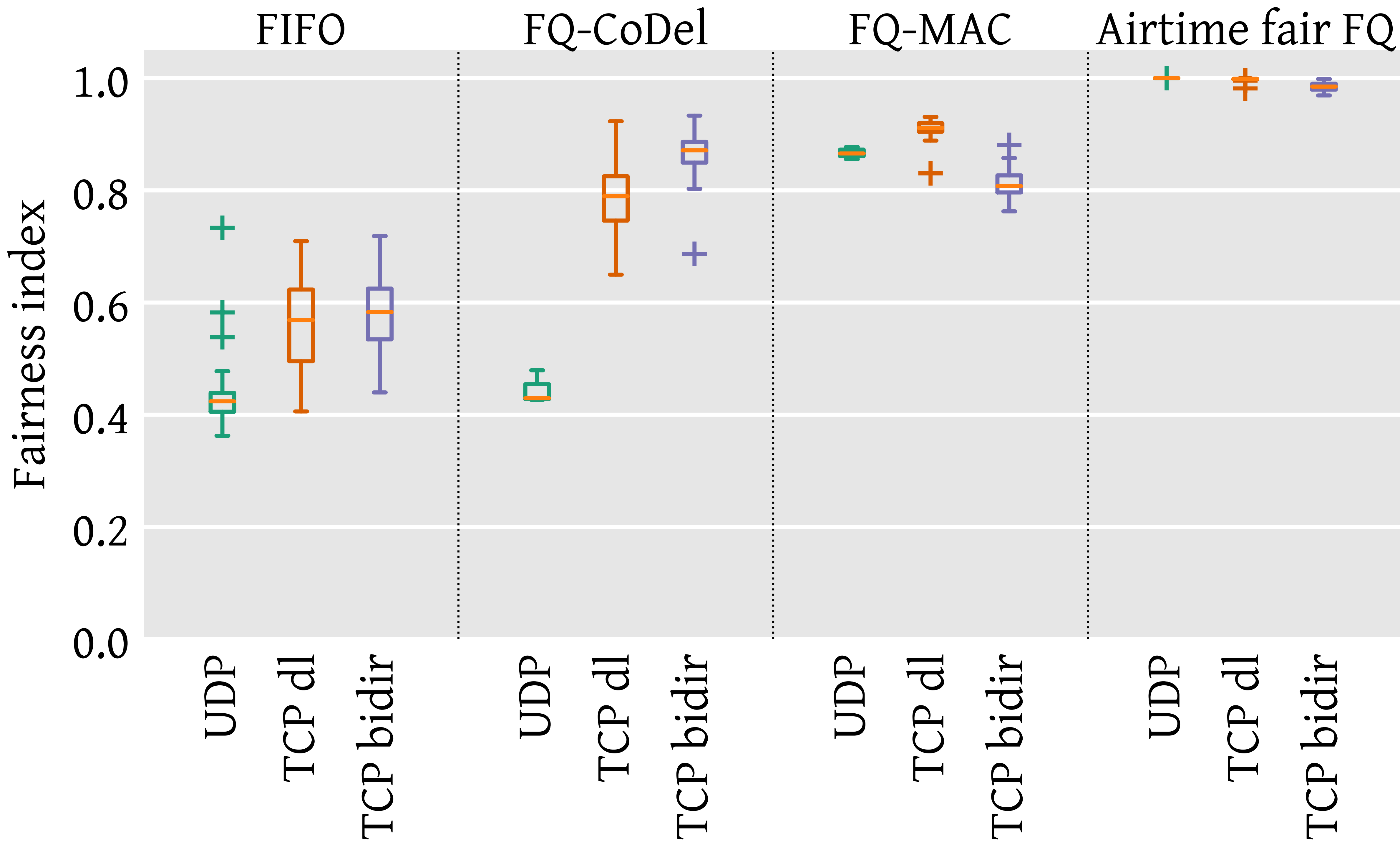
# AIRTIME SCHEDULER

```
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)
}
```



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

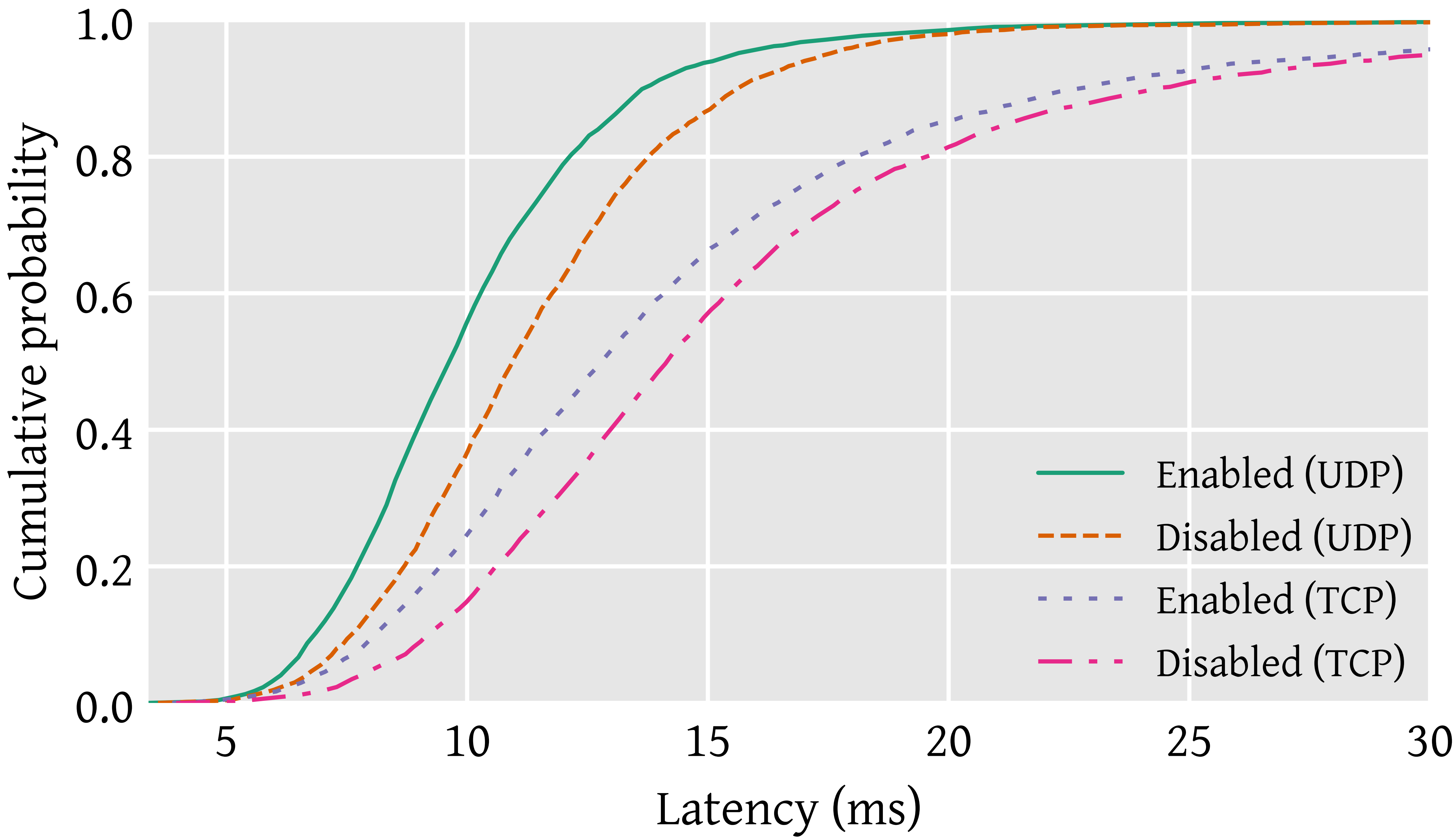
# AIRTIME FAIRNESS




COMPUTER SCIENCE
DATAVETENSKAP

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

# SPARSE STATION OPTIMISATION

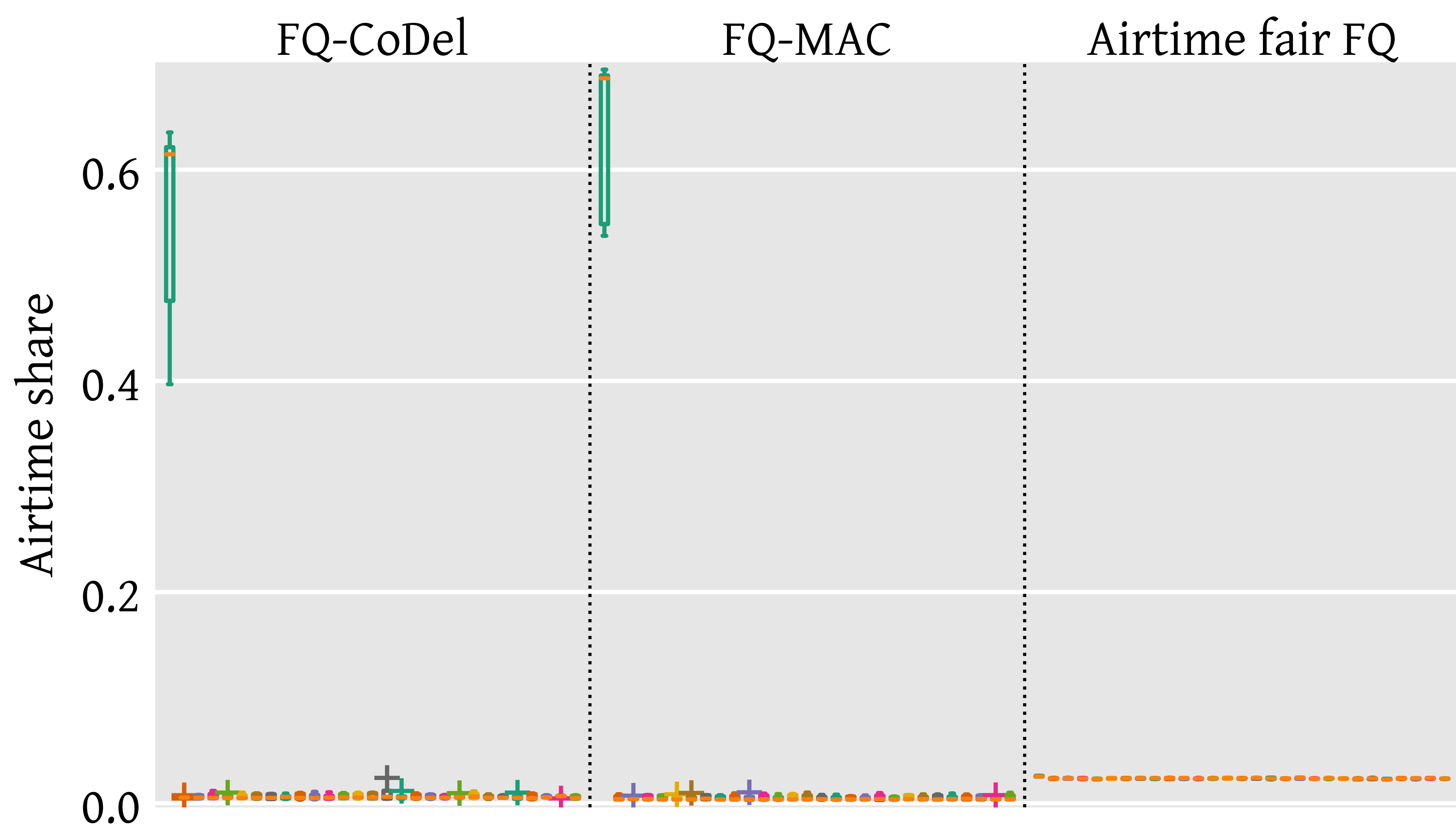



COMPUTER SCIENCE
DATAVETENSKAP

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

# 30 STATIONS TEST

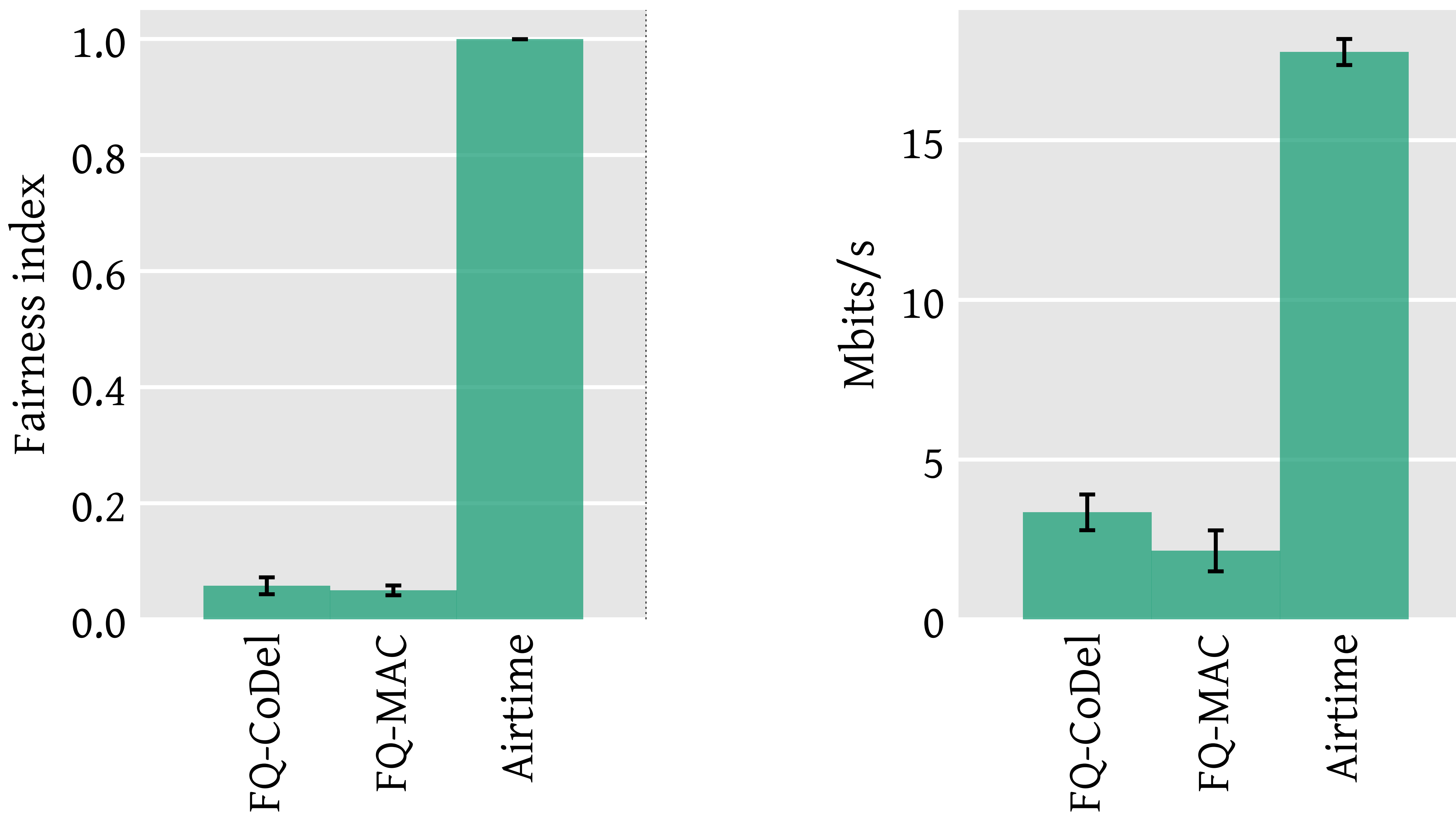
- We cooperated with another lab to evaluate our solution
- 30 station testbed, one slow station (1 Mbps)






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

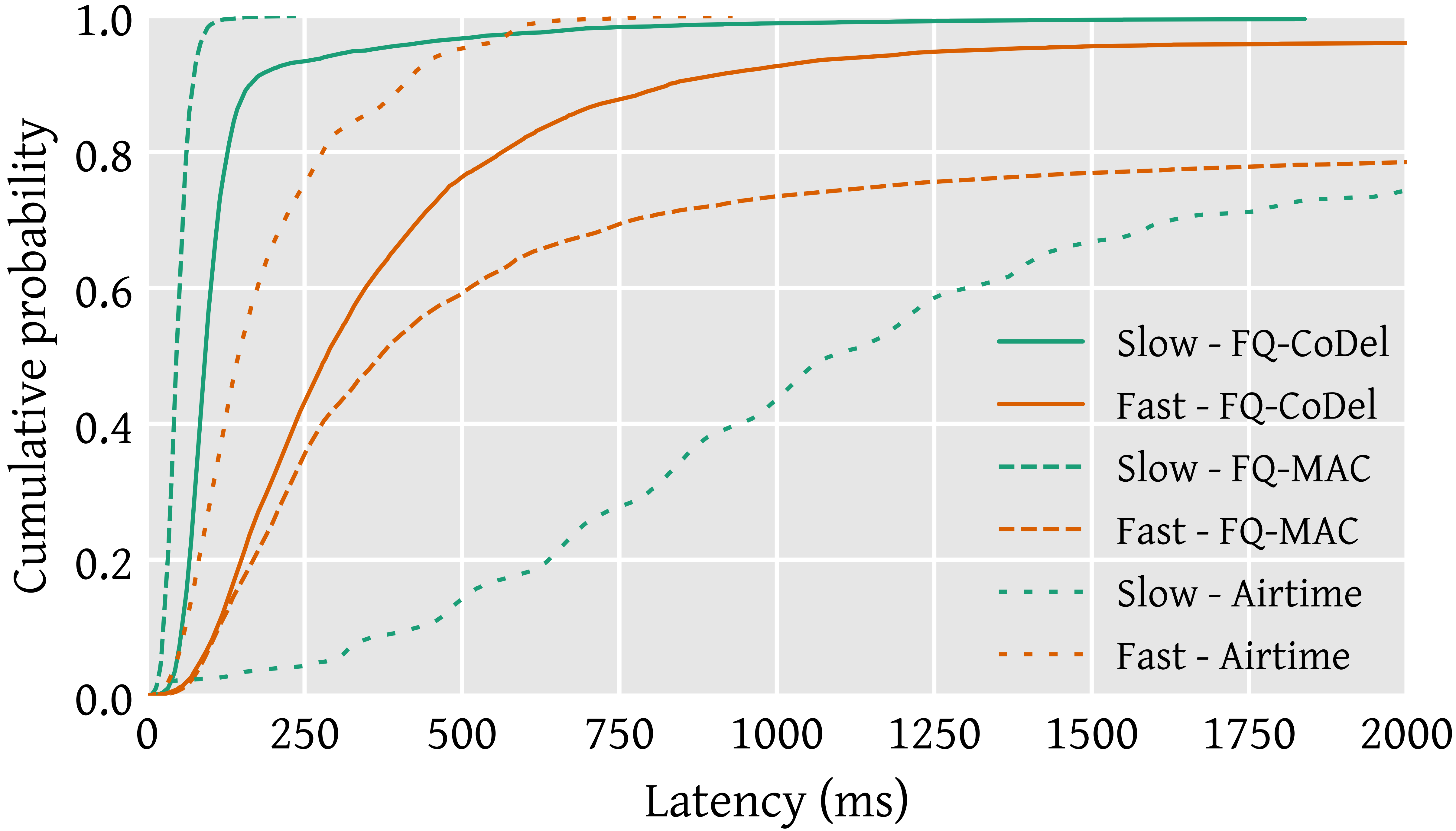
# 30 STATIONS






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

# 30 STATIONS - LATENCY




COMPUTER SCIENCE
DATAVETENSKAP