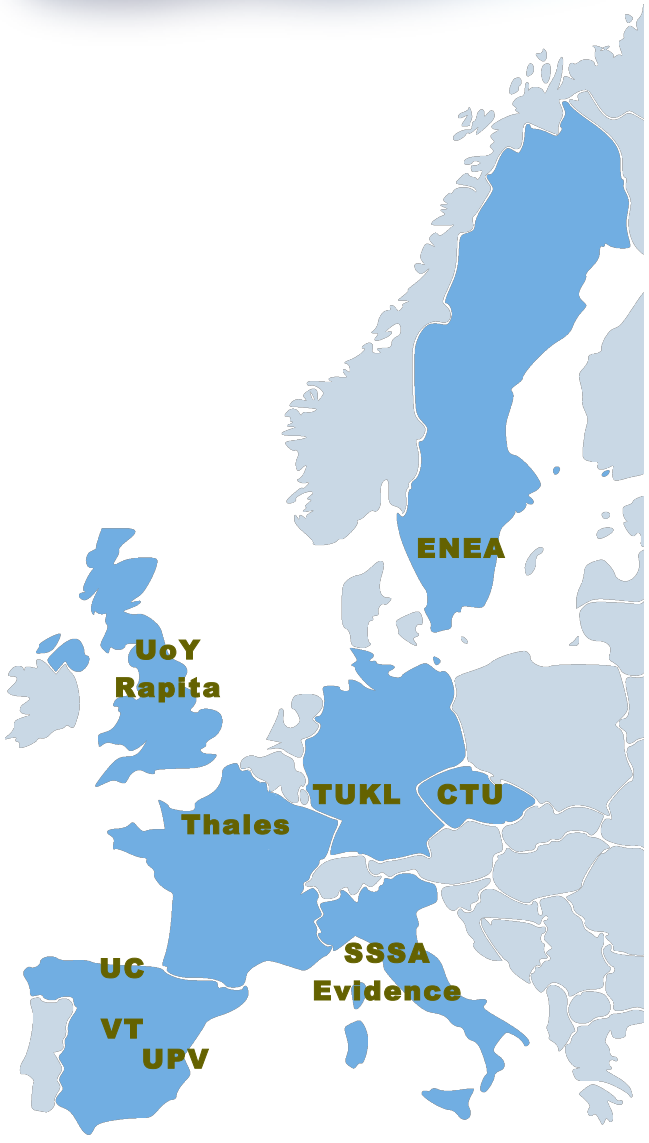
frescor

Framework for Real-time Embedd
Systems based on COntRacts

Information Society Technologies



Bezdrátové sítě IEEE 802.11 v kontraktovém frameworku

Martin Molnár, Michal Sojka

Czech Technical University in Prague
Department of Control Engineering

ºFeb 5, 2008

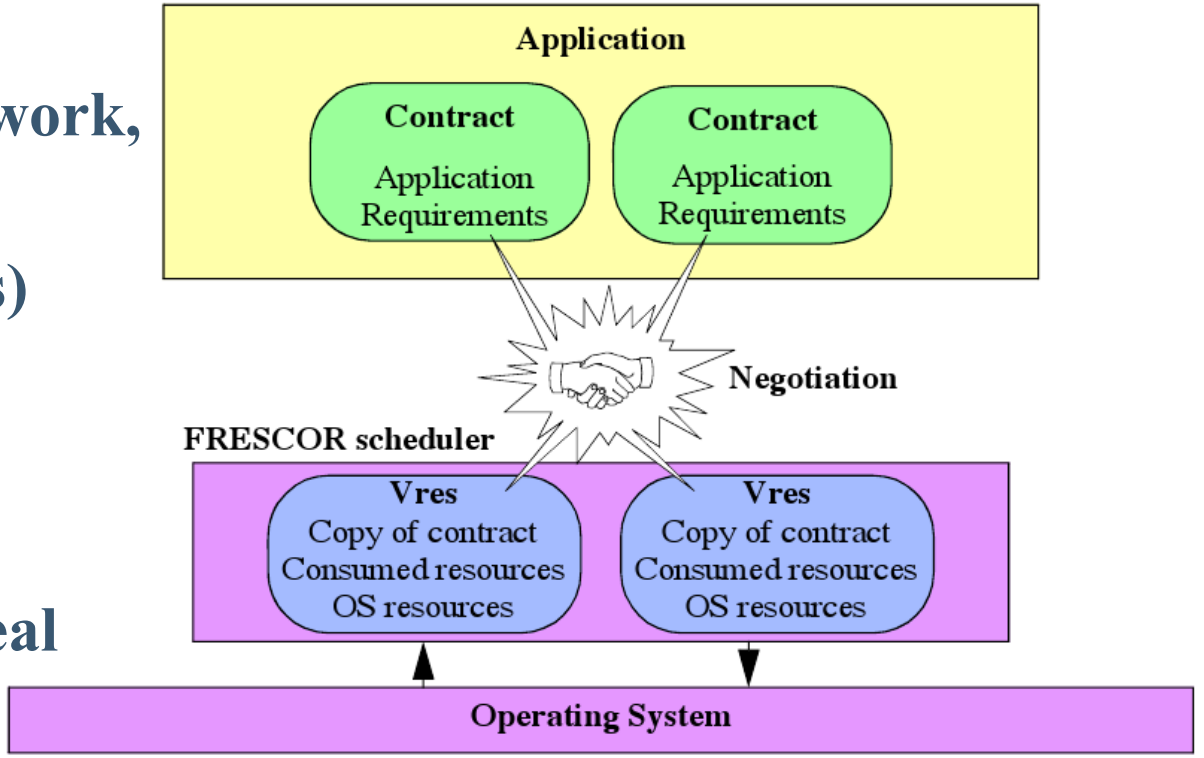
Real-Time Applications

- Functional requirements
  - What to do – determined by the controlled technology
  - Every programmer can fulfill them
- Timing requirements
  - Until when to do it – deadlines
  - Fulfilling them is not straightforward for more complicated applications
    - Multiple threads
    - Shared resources
  - One can use some techniques to find schedulability (response-time analysis)
  - What in case of distributed system? Timing is influenced by communication latency.
  - What about mode changes? What about ...?

Use contract based framework! FRESCOR

Contract

- Application specifies its requirements (CPU, network, disk, ...) in contract
- Negotiates the contract(s) with the scheduler
- Contract is rejected or accepted
- Application access the real resource through Vres (virtual resource)
  - Runtime representation of contract
  - Protects the resource from misbehaved applications



IEEE 802.11 legacy MAC

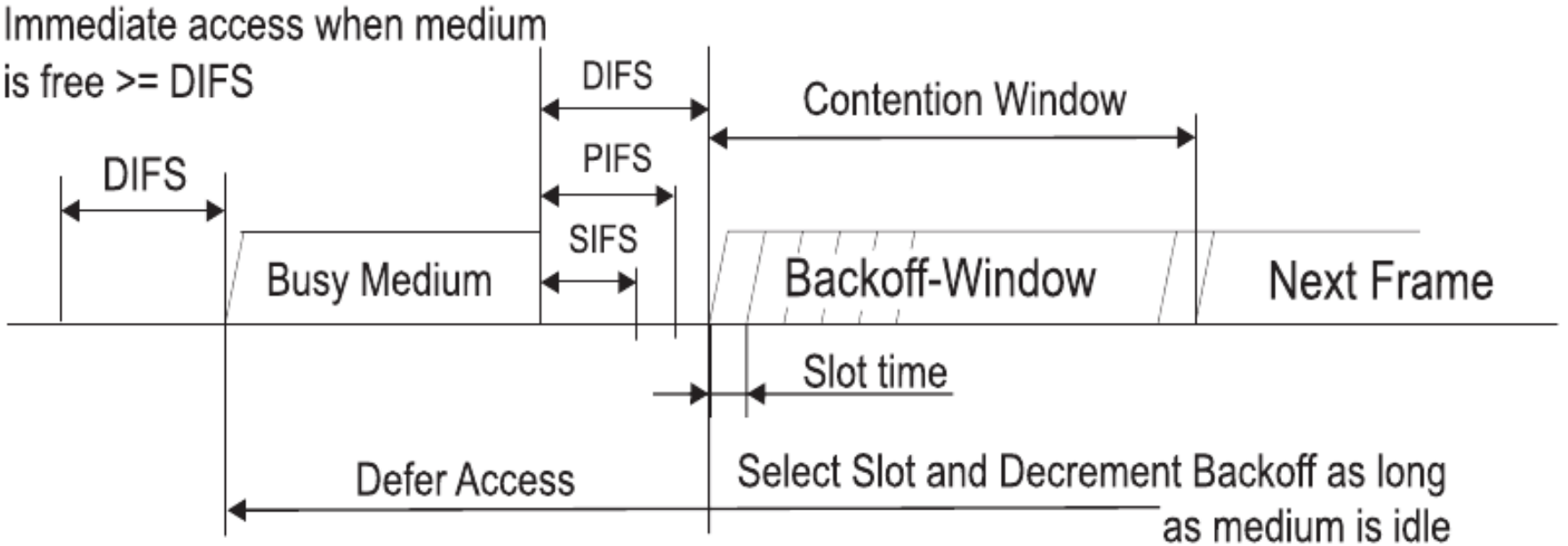
Defines two medium access coordination functions:

- **1. Distributed Coordination Function (DCF)**
  - **mandatory**
  - **Contention-based (CSMA/CA)**
  - **provides asynchronous transmission**
  - **can be used in ad-hoc and infrastructure modes**
- **2. Point Coordination Function (PCF)**
  - **not mandatory**
  - **contention-free**
  - **provides synchronous transmission**
  - **can be used only in infrastructure mode**

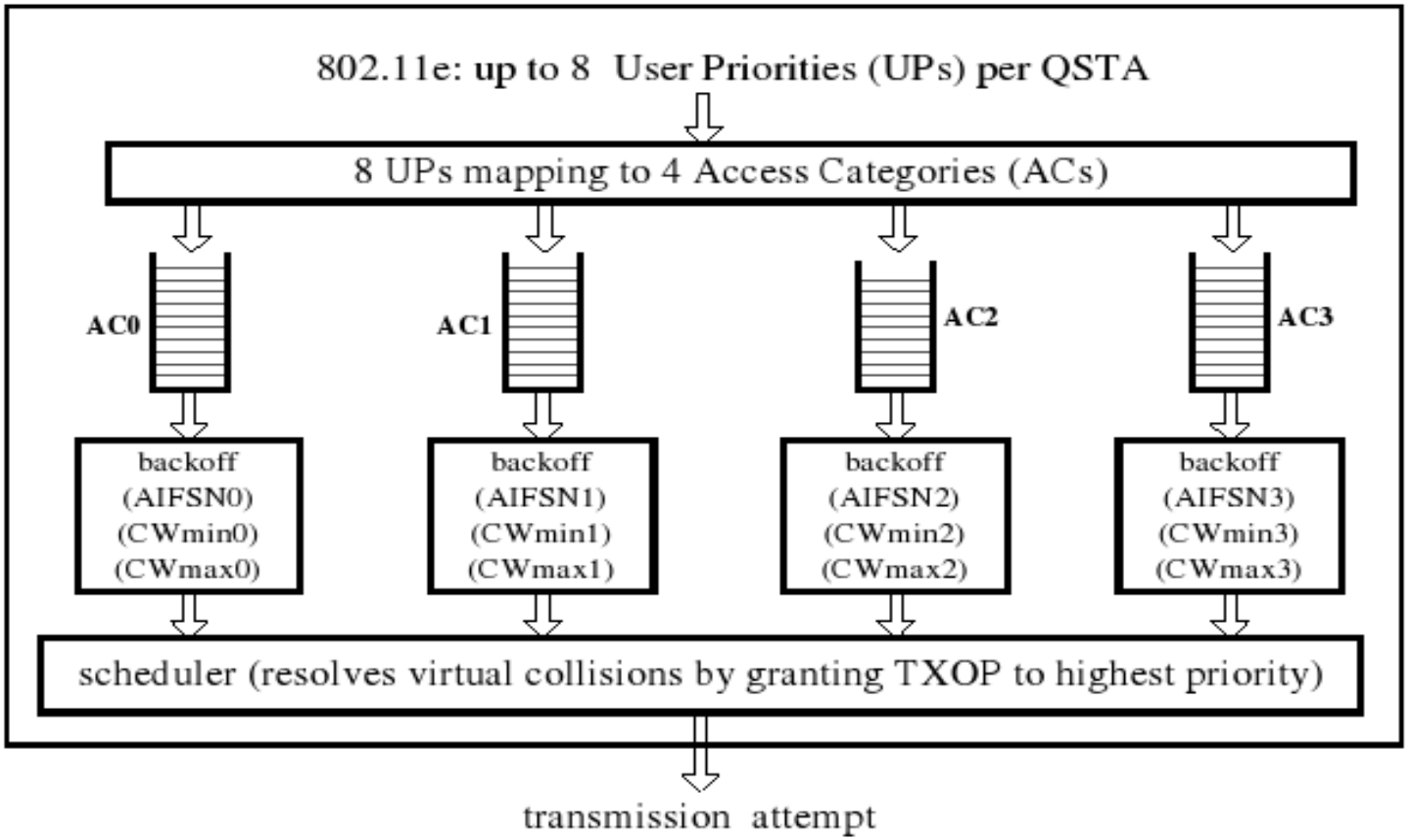
Basic Service Set (BSS) = group of stations(STA's) coordinated by DCF or PCF

Medium Access: DCF

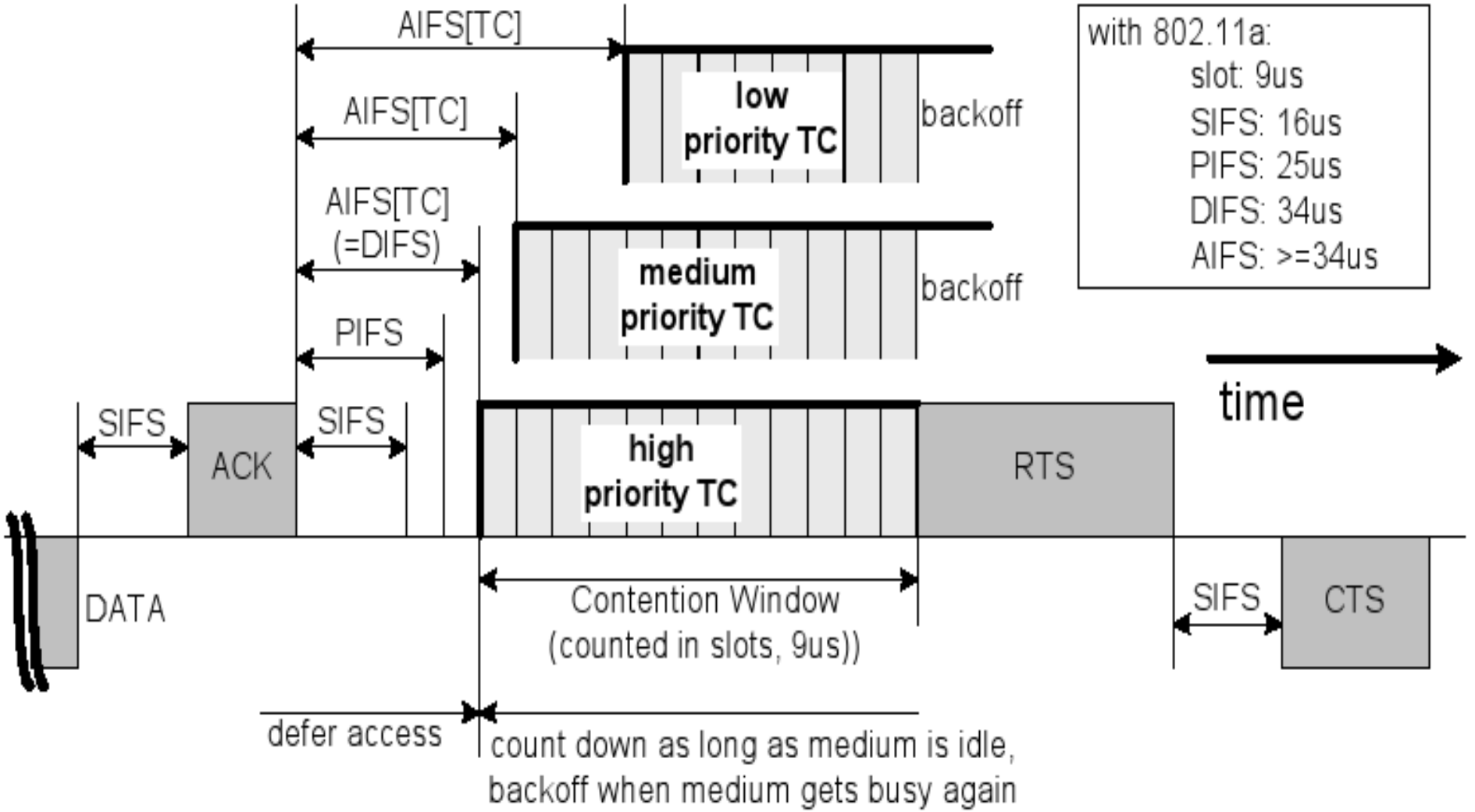
- Backoff window – wait random number of slots
- Collision = two stations start transmitting at the same slot
  - Double the size of Backoff windows
- Problem: all kinds of traffic have the same access policy (priority)



IEEE 802.11e – Medium Access (EDCA)



IEEE 802.11e – Medium Access (EDCA)



FRESCOR WLAN Protocol (FWP)

- FRESCOR Network protocol on top of Wi-Fi networks
- Utilizes IEEE 802.11e (QoS extension)
- Implemented on top of Linux wireless stack (mac80211) – included in 2.6.24

Current status

- standalone FWP prototype (doesn't use any FRSH services)
- WMM tests to familiarize ourselves with behavior of HW and SW
- simple analysis

TODO:

- finish FWP
- integration with the rest of FRESCOR
- compare multiple analysis methods
- tests

Restrictions for Phase 1

- No reliable communication
  - Solution: Use TCP protected by FRESCOR specific Qdisc
- Implemented on top of UDP and in userspace
  - Packet queues and Vreses are better implemented in kernel (lower overhead)
- Rate switching avoided – fixed value 1 Mbit/s for initial test
  - Use nl80211 protocol to obtain current rate and other statistics from drivers
- A simple admission test
  - Decides which contracts to accept
  - Comparison with more advanced techniques necessary

FWP – components I.

libfwp (Frescor wifi library)

- a library linked with every frsh app
- implements :
  - FNA (Frescor Network Adaptation layer) operations
  - vres handling (adding/removing) operations
  - receive and send routines (budget mechanism, BSD sockets)
  - communicates with local FWP agent

FWP – components II.

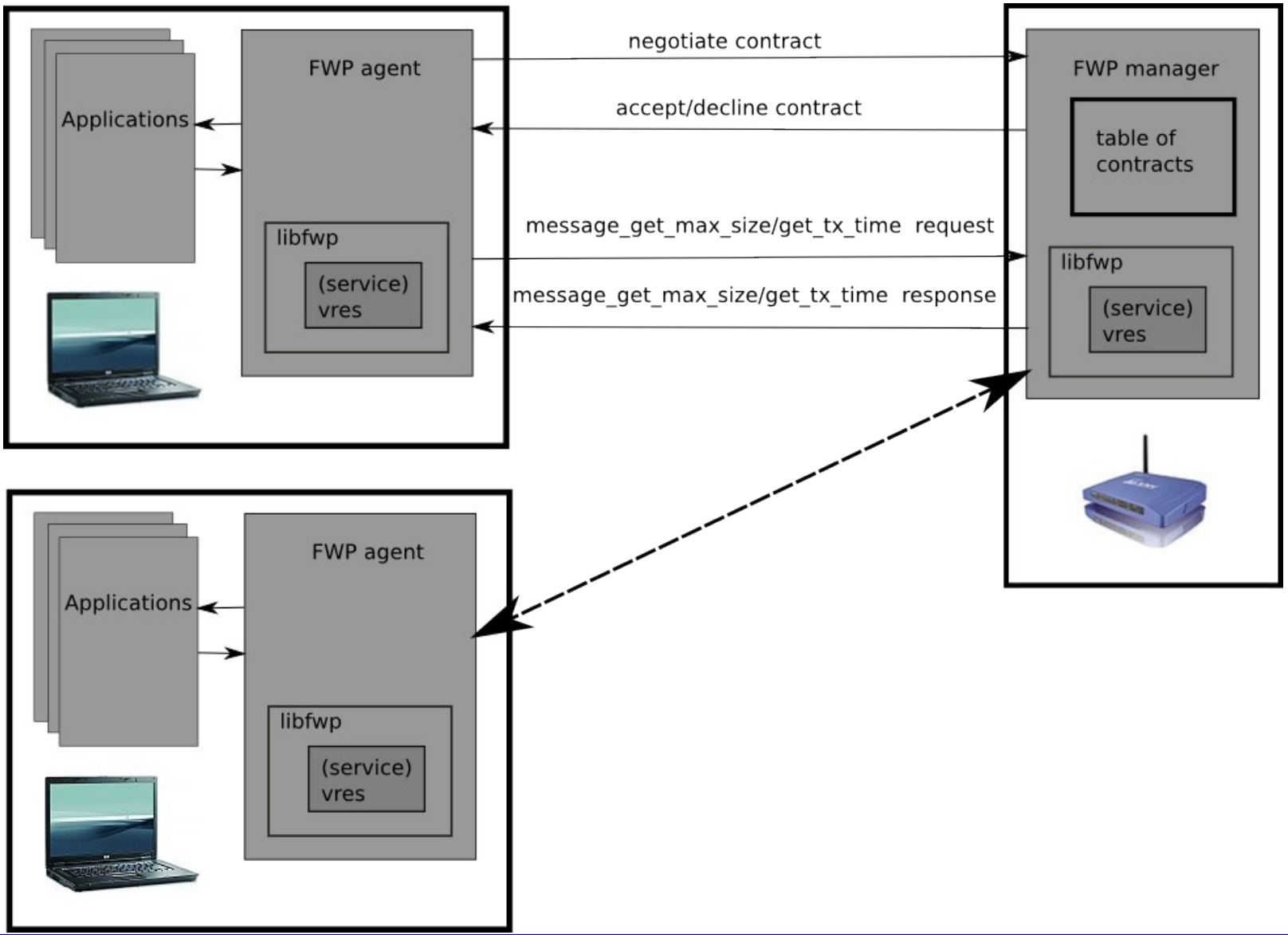
FWP manager

- FRSH process on access point that does admission control and accepts/declines net contracts

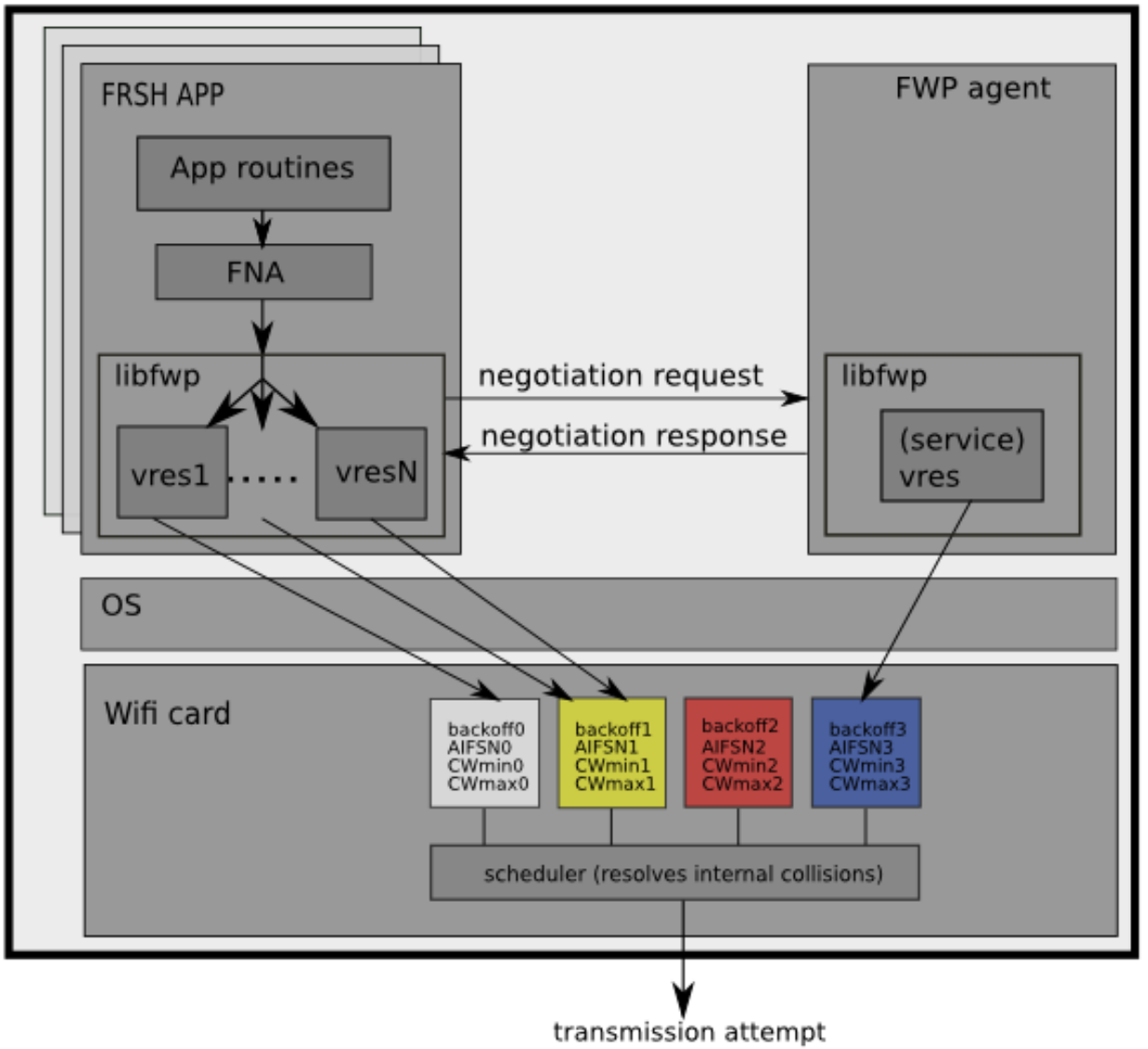
FWP agent

- FRSH process in every station that negotiates network contract on behalf of FRSH application
- In future it may provide some information to manager (e.g. wireless link statistics etc.)

Architecture I.

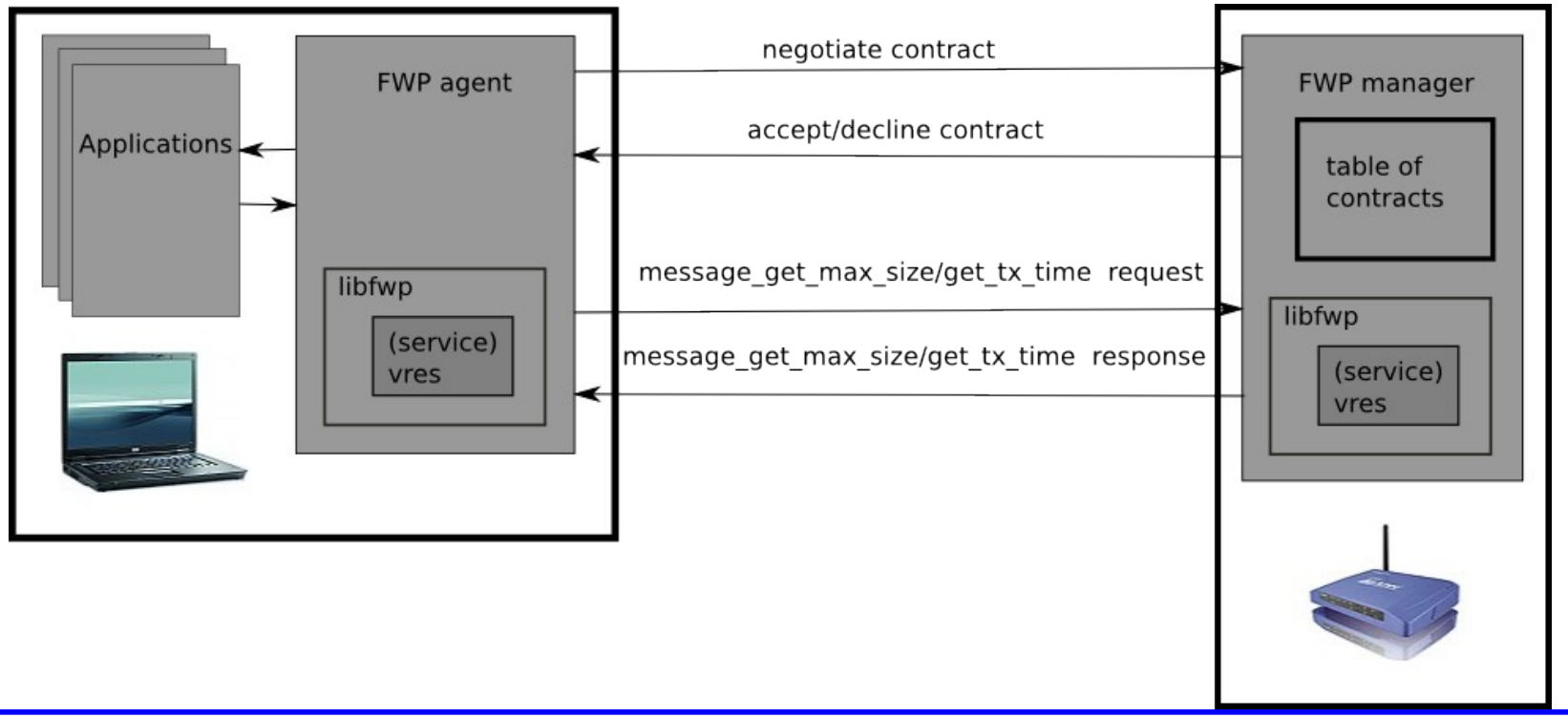


Architecture II. – detailed node

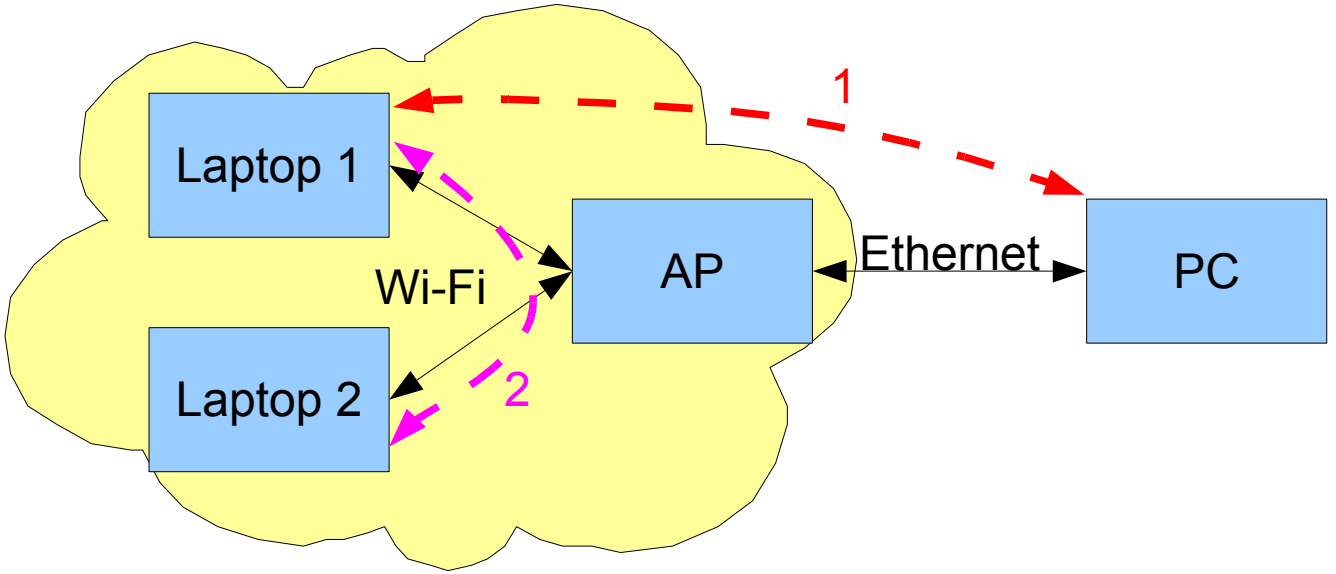


Sequence of contract negotiation

1. An application sends contract to FWP agent through UNIX socket.
2. FWP agent use its network VRES to forward the request to FWP manager
3. FWP manager executes admission test and sends results back to FWP agent
4. FWP agent forwards results to the application
5. If some other nodes are influenced by this change (spare capacity), FWP manager contacts the corresponding FWP agents with the change.



Experiments



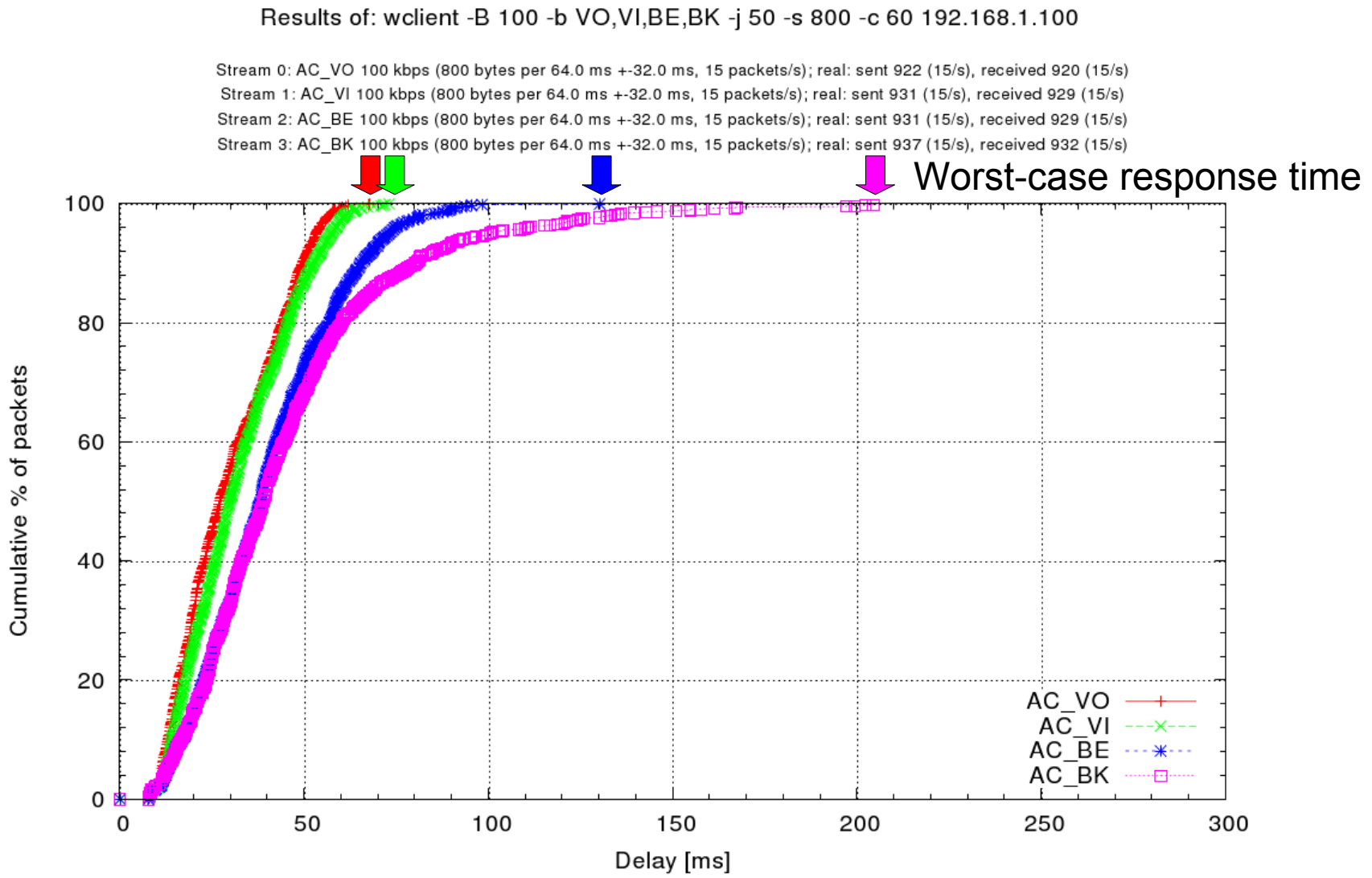
Measuring of communication latency between:

1) Laptop 1 and PC

2) Laptop 1 and Laptop 2

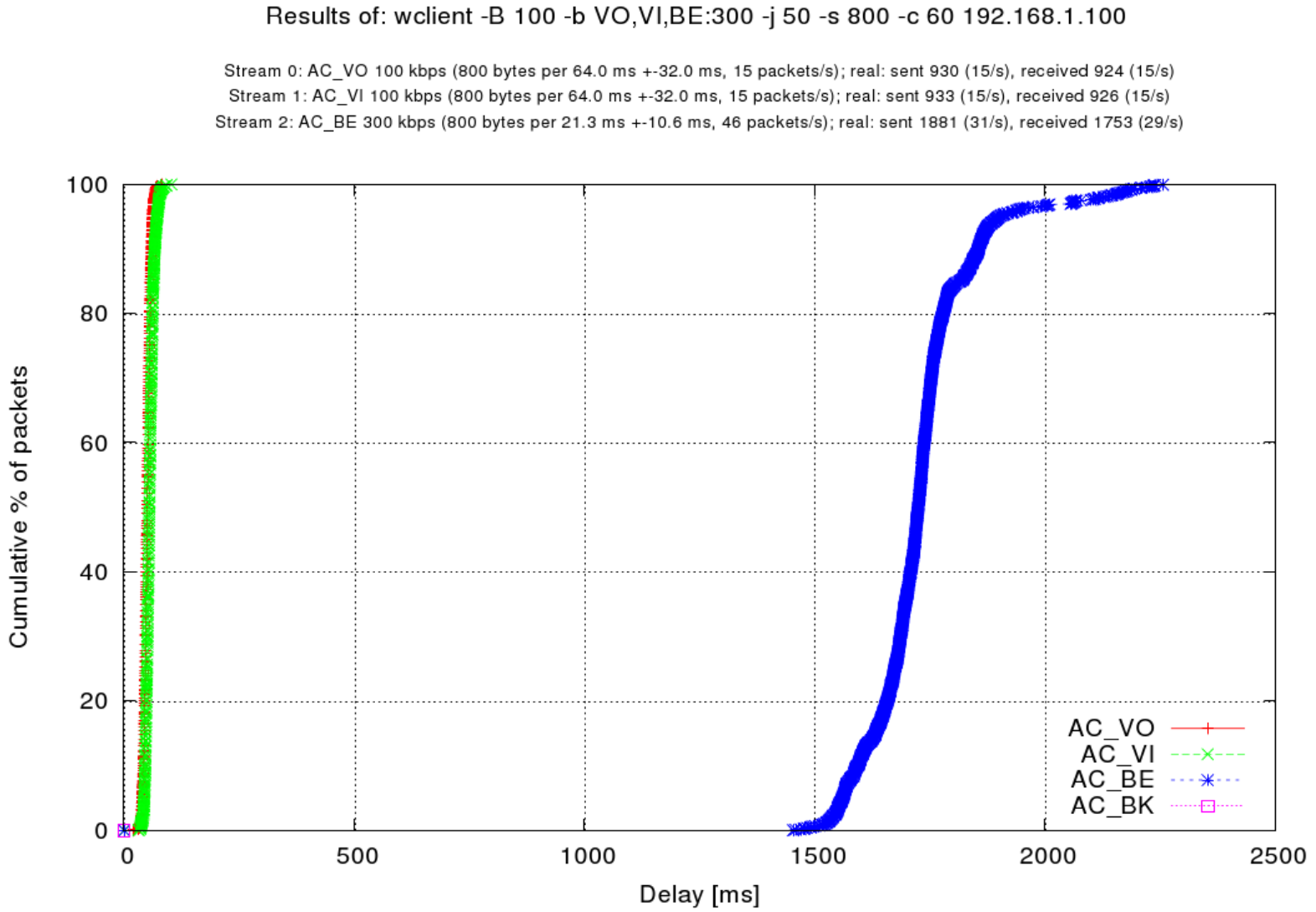
Non-saturated medium

No big difference between the 4 access categories



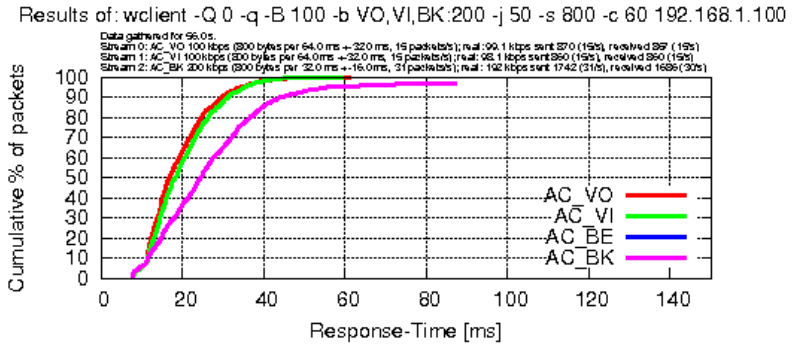
Saturated medium

Only the lowest “priority” is saturated. Big queuing delay.

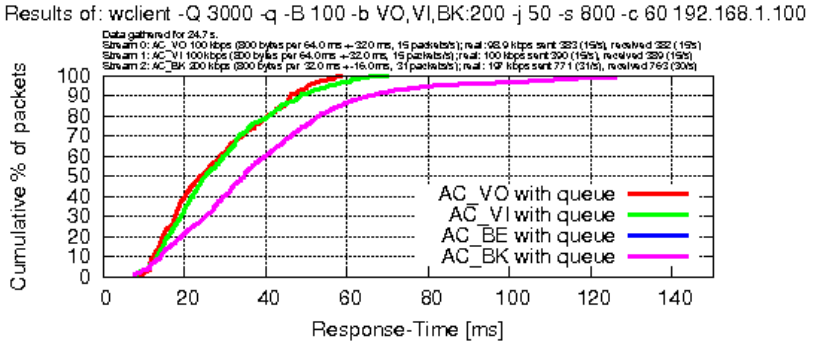


How queue size influences delay (no saturation)

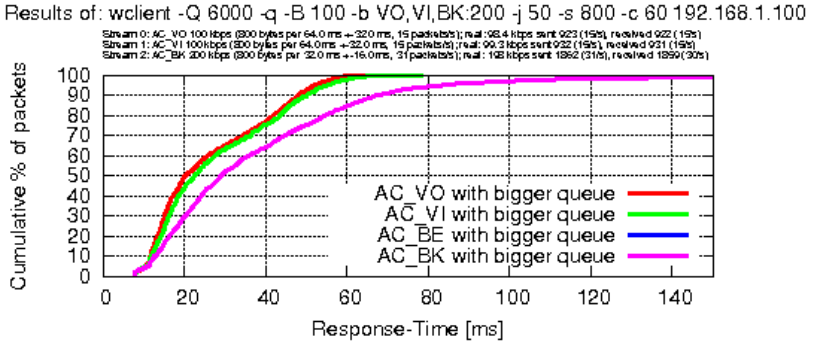
No queue



Small queue

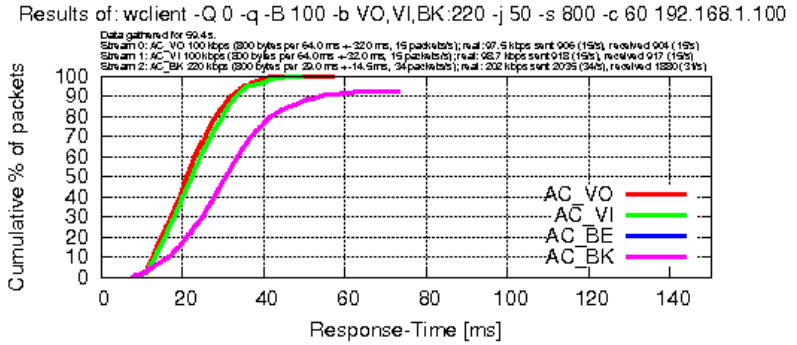


Big queue

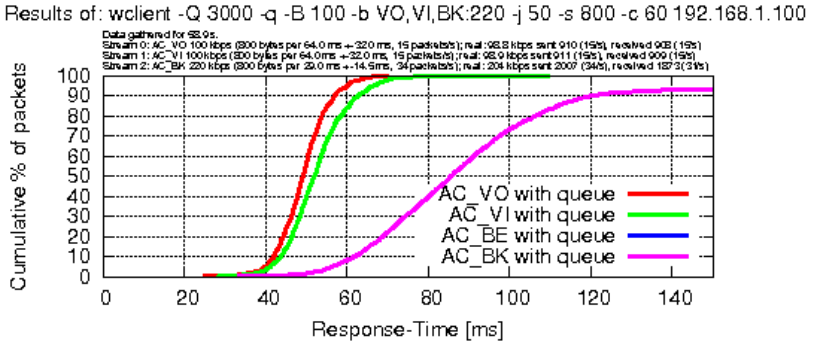


How queue size influences delay (saturation)

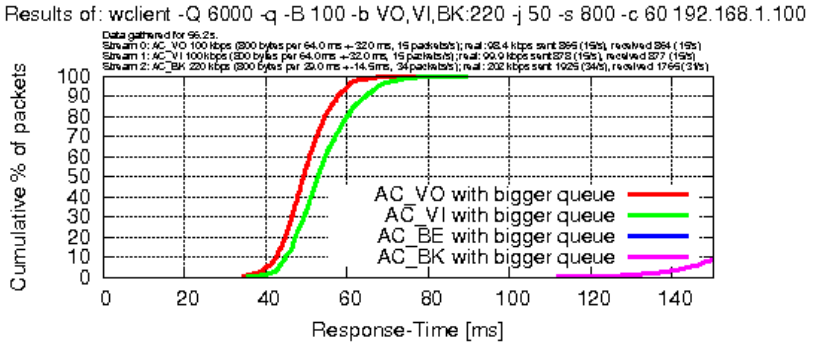
No queue



Small queue



Big queue

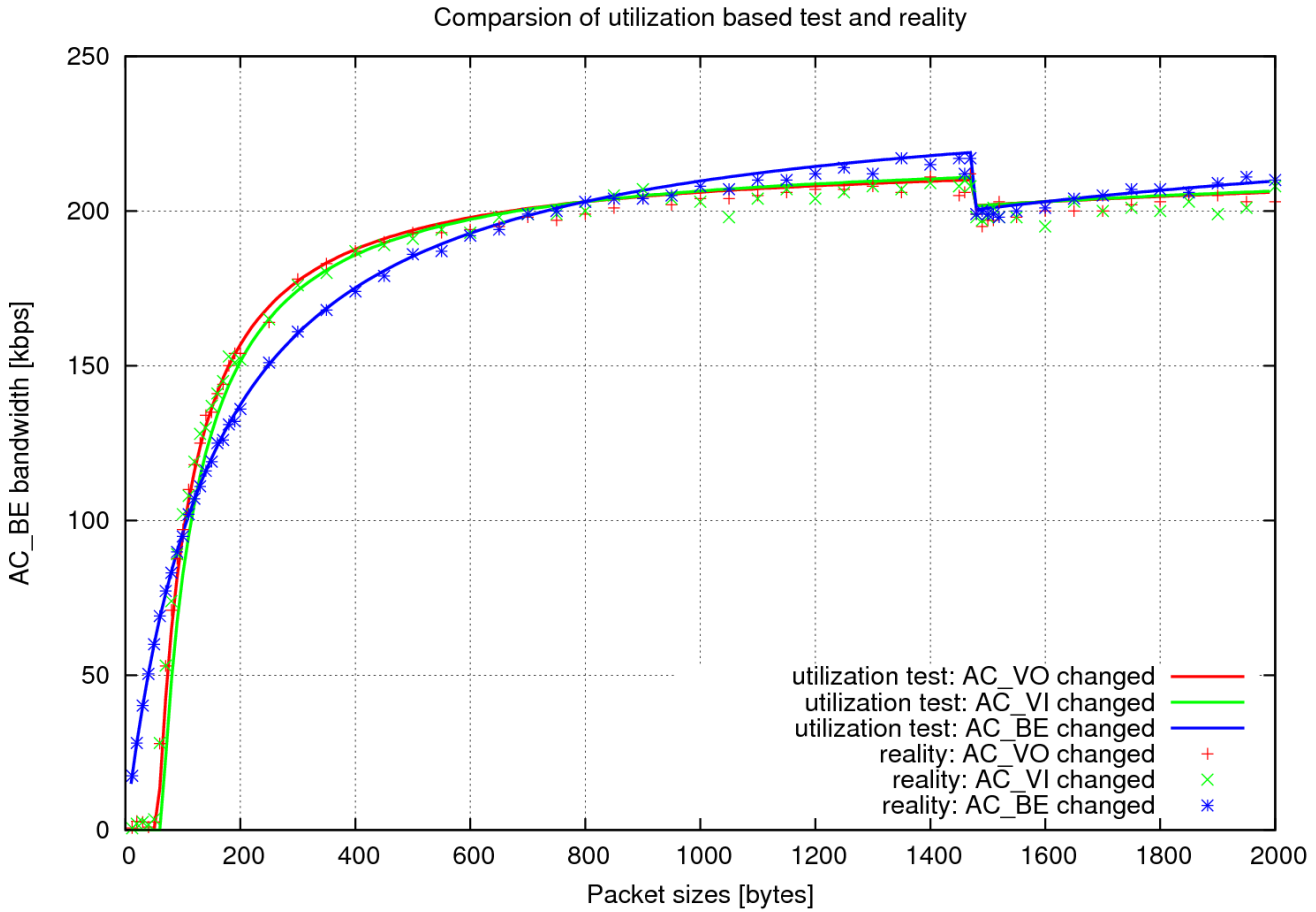


Results

- Do not allow saturation of access categories (AC) for real-time traffic.
- Even if only a low priority AC is in saturation, it influences the response-time of higher priorities (cca 2x greater than non-saturated case)
- Decreasing the number of socket buffers (possibly up to zero) helps, but it degrades performance.
- Simple solution:
  - utilization based admission test to avoid saturation.
    - $\sum(\frac{(packet\ tx\ time + overhead)}{period})$
  - Probability of collision should also be limited in the test.
  - Deadline is handled only in binary manner – greater than 100 ms = accepted (for 1 Mbit/s)

Simple utilization based test and reality

Saturation bandwidth of AC_BE as a function of packet size in 100 kbit/s streams.



Red Line:

- **AC_VO** 100 kbps, 10 – 2000 bytes/packet
- **AC_VI** 100 kbps, 800 bytes/packet
- **AC_BE** saturation, 800 bytes/packet
- Saturation bandwidth was measured during 60 s experiment.

- **Good match with these 3 particular experiments**
- **This match corresponds to 96% utilization according to our admission test**
- **Some empirical constant are used to estimate the influence of collisions**