Wireless Software and Hardware platforms for Flexible and Unified radio and network controL

Project Deliverable D2.4

Results of second set of showcases

Contractual date of delivery:	31-12-2016
Actual date of delivery:	23-12-2016
Beneficiaries:	IMEC, TCD, CNIT, TUB, NCENTRIC, RUTGERS, SNU
Lead beneficiary:	TCD
Authors:	Diarmuid Collins (TCD), Maicon Kist (TCD), Alextian Bartholomeu Liberato (TCD), Ingrid Moerman (IMEC), Peter Ruckebusch (IMEC), Spilios Giannoulis (IMEC), Pieter Becue (IMEC), Anatolij Zubow (TUB), Mikolaj Chwalisz (TUB), Piotr Gawłowicz (TUB), Ilenia Tinnirello (CNIT), Pierluigi Gallo (CNIT), Domenico Garlisi (CNIT), Robin Leblon (NCENTRIC), Sven Zehl (TUB, Changmok Yang (SNU), Sunghyun Choi (SNU)
Reviewers:	Mitch De Geest (NCENTRIC) and Ivan Seskar (RUTGERS)
Work package:	WP2 – General Requirements and Showcases
Estimated person months:	6
Nature:	R
Dissemination level:	PU
Version:	1.0

Abstract:

This public deliverable reports on the results of the second set of showcases that have been implemented. It also includes the specifications of the third set of showcases to be implemented by the end of Year 3. This deliverable reports on activities in WP3, WP4, WP5 and WP6 that are related to the showcases.

Keywords:

Showcases, proof-of-concept, use of UPIs

Executive Summary

This deliverable reports on the showcases implemented in the second year of the project, which exhibit the functionality of the WiSHFUL platform. Each showcase introduces the problem it aims to address, presents an overview of the showcase demonstration, how WiSHFUL UPIs are used, the results or expected results (for showcases being implemented), and future steps. Furthermore, the work outlined herein provides the basis of upcoming scientific publications and other dissemination material as a means of maximizing the impact of the project. Showcases in this deliverable demonstrate the following:

- The LTE-LAA - Wi-Fi coexistence will show how a managed 802.11 Wi-Fi network makes use of WiPLUS to passively *estimate the airtime occupied by LTE-U* at each AP in order to perform better channel assignment and to load-balance client stations across the APs for maximizing the overall throughput/service quality.
- The GNU Radio showcase aims to demo the execution of two independent virtual radios sharing one RF front-end simultaneously.
- The IRIS demo shows the integration with UPIs by changing the parameters on the fly at USRPs.
- The Coexistence of IEEE 802.15.4e TSCH with IEEE 802.11 networks showcase is a demonstration and proof of the solution feasibility and applicability for a cross technology synchronization scheme between TSCH and Wi-Fi networks.
- The Radio Slicing for Virtualized Home Wi-Fi Access Points is currently being implemented with a novel slicing technology to guarantee bandwidth and traffic isolation for the primary users of the home AP in uplink and downlink.
- MAC adaptation in presence of legacy stations extends results presented in D2.3 to consider the possibility that some wireless nodes cannot be directly controlled by WiSHFUL. This showcase proves that the optimization logic can be implemented on WiSHFUL programmable nodes with or without presence of legacy nodes.
- MCS Selection shows that lowering frame loss in highly mobile networks demonstrates how dynamic reconfiguration of frame aggregation and PHY rate adaptation parameters can lower the frame loss.
- The link estimator selection shows that detecting the optimal link estimation algorithm in various network topology scenarios can increase the overall network performance by dynamically selecting the optimal link estimation algorithm supported by a global control program and controlling a sensor network.
- Multihop load aware MAC adaptations proposes and implements a very promising scheme called REACT for mitigating the performance impairments of CSMA/CA protocols in multi-hop topologies based on the dynamic adaptation of the contention process experienced by nodes in the wireless network.

Technical detail regarding the implementation of support for each showcase is left to the appropriate technical deliverables (D3.4, D4.4 and D6.4). Showcases are used to promote the utility of the project by providing convincing scenarios that use WiSHFUL technologies and infrastructure for advanced wireless research and experimentation. They also act as educational, training and tutorial material to support third parties using developed WiSHFUL UPIs. In combination with D2.2, D2.3 and the upcoming D2.5, this deliverable provides a complete picture of WiSHFUL capabilities and effectiveness by applying the technologies developed by the project to areas of active research and problems relevant to the community in a user-friendly and consistent manner. Finally, this document also defines, at a high-level, a list of intelligence showcases to be implemented within the third year of the project. These include:

- Over the air (OTA) updates using GITAR for WSNs

- Extension of MAC adaptation in multi-hop topologies based on directional antenna and multiple path reservations
- Radio-based indoor localization
- Extension of MAC optimizations in high-density scenarios, with online phase
- Interference classification for Wi-Fi nodes on the basis of error patterns, using machine learning
- Radio virtualization with simultaneous transmission and reception
- IEEE 802.11 Overlapping BSS management
- Closed-loop rate control for IEEE 802.11 infrastructure networks
- Context Awareness in spectrum management system-aided SU networks

List of Acronyms and Abbreviations

5G	Fifth Generation
A-MPDU	Aggregate MPDU
AP	Access Point
API	Application Programming Interface
AGC	Automatic Gain Control
BSS	Basic Service Set
BlockAcks	Block Acknowledgements
CSMA	Carrier Sense Multiple Access
CCA	Clear Channel Assessment
COM	Component Object Model
CSI	Channel State Information
CWopt	CW Optimum
CPS	Cyber-Physical Systems
DDOS	Distributed Denial of Service
DSL	Digital Subscriber Loop
DSR	Dynamic Source Routing
EWMA	Exponential Weighted Moving Average
ISM	Industrial, Scientific and Medical
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
LQI	Link Quality Indicator
LQE	Link Quality Estimator
LTE	Long Term Evolution
LBT	Listen Before Talk
LSA	Licensed Shared Access
LR-WPANs	Low-Rate Wireless Personal Area Networks
LTE	Long-Term Evolution
LTE-U	LTE Unlicensed
MAC	Medium Access Control
MCS	Modulation and Coding Scheme
MCDs	Measurement-Capable Devices
MEDCA	Moderated Backoff
MPDU	MAC protocol data unit

MSDU	MAC service data unit
NB-IoT	Narrow Band-IoT
NRDMS	Number of RPL DIO messages sent
NRPS	Number of RPL parent switches
NTP	Network Time Protocol
OTA	Over The Air
PDR	Packet Delivery Ratio
PLCP	Physical Layer Convergence Protocol
PTP	Precision Time Protocol
PUs	Primary Users
REM	Radio Environment Maps
ROT	Radio-on Time
RR	Retransmission Ratio
RSSI	Received Signal Strength Indicator
RRH	Remote Radio Head
RF	Radio Frequency
SFER	Subframe Error Rate
SFs	Slot Frames
SDR	Software Defined Radio
SSIDs	Service Set Identifiers
SAS	Spectrum Access System
SMS	Spectrum Management Systems
TSCH	Time-Slotted Channel Hopping
TSF	Time Synchronization Function
Tx	Transmitter
USRP	Universal Software Radio Peripheral
VNFs	Virtualized Network Functions
Wi-Fi	Wireless Fidelity

Table of contents

1 Introduction

This deliverable reports on the outcome of the second set of WiSHFUL showcases planned in WP2 *General requirements and showcases* and define in the previous deliverable D2.3 [1]. These showcases define relevant and convincing scenarios in view of promoting WiSHFUL framework capabilities. The aim is to show third parties and experimenters the potential benefits of using WiSHFUL infrastructure and software platforms for wireless innovation creation. For example, the IRIS SDR module, which is a highly reconfigurable radio, demonstrates WiSHFUL capabilities by changing Frequency, Gain and bandwidth parameters. Additionally, we also demonstrate some advanced WiSHFUL experimentation capabilities defined by the GNU Radio virtualization component in Section 2.2. These showcases also act as educational, training and tutorial material to support third parties using WiSHFUL UPIs.

Section 2 starts by listing showcases implemented during Year 2 of the project. Showcases are grouped into the following categories:

- LTE-LAA - Wi-Fi coexistence
- Software Defined Radio (SDR) capabilities
- Coexistence of IEEE 802.15.4e TSCH with IEEE 802.11 networks
- Radio Slicing for Virtualized Home Wi-Fi Access Points
- MAC adaptation in presence of legacy stations
- Load and topology aware networking

Section 3 defines a list of showcases to be implemented in Year 3 of the project. These include:

- Over that air updates using GITAR for WSNs
- Extension of MAC adaptation in multi-hop topologies based on directional antenna and multiple path reservations
- Radio-based indoor localization
- Extension of MAC optimizations in high-density scenarios, with online phase
- Interference classification for Wi-Fi nodes on the basis of error patterns, using machine learning
- Radio virtualization with simultaneous transmission and reception
- IEEE 802.11 Overlapping BSS management
- Closed-loop rate control for IEEE 802.11 infrastructure networks
- Context Awareness in spectrum management system-aided SU networks

Finally, section 4 concludes this deliverable.

2 Results of the second set of Showcases implemented in Year 2

This section presents the results from the second set of showcases proposed and defined in D2.2 and implemented in Year 2.

2.1 LTE-U - Wi-Fi coexistence

Cellular network operators are ***offloading traffic in unlicensed*** 5 GHz spectrum using LTE Unlicensed (LTE-U [2]). However, this part of the radio spectrum is also used by existing IEEE 802.11 standards, e.g. 802.11ac, or future Wi-Fi standards, e.g. 802.11ax. Hence, the situation becomes similar to the one in 2.4 GHz ISM band where radio spectrum is shared by different radio technologies, e.g. Wi-Fi, Bluetooth, ZigBee, etc., where numerous studies have demonstrated the adverse effects of interference from different wireless technologies on Wi-Fi.

In particular Jindal et al. [3] showed that, as LTE-U duty-cycling does not implement listen before talk mechanisms (LBT, e.g. CSMA/CA), introduction of LTE-U can disproportionately reduce Wi-Fi throughput performance. Moreover, interference from LTE-U with moderate power can be even more harmful to Wi-Fi than high-power interference [4]. Other studies confirm this [22]. Figure 1 shows results from our own experiments where a single high quality Wi-Fi link was affected by a LTE-U signal source. In particular, it shows the normalized UDP throughput of the Wi-Fi link under LTE-U interference, relative to the non-interfered Wi-Fi link, as a function of different interfering signal strengths, i.e. distances, and LTE-U duty-cycles. We can clearly see the impact of the LTE-U duty cycle on the Wi-Fi performance. Moreover, the Wi-Fi performance is influenced even when LTE-U signal is very weak, i.e. LTE-U source is far away.

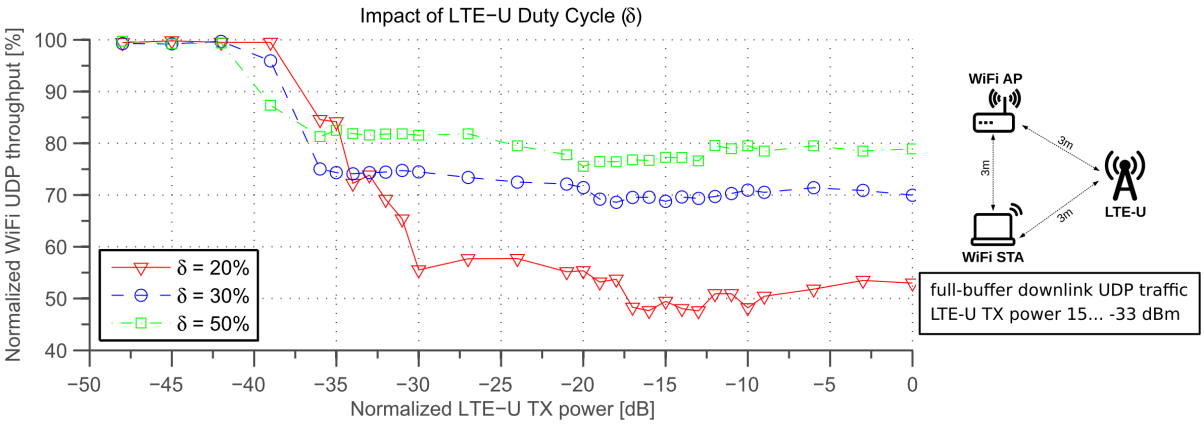


Figure 1. Degradation in UDP throughput of a high quality Wi-Fi link in the presence of LTE-U device operating in the same band and at different signal strengths.

In ***WiPLUS*** [5] we presented a passive low complexity method for ***detection of LTE-U RF devices*** and their duty-cycles in real-time and using only commodity Wi-Fi hardware. The method accurately estimates the airtime occupied by LTE-U, i.e. ON period, at a wide range of LTE-U signal strengths allowing the AP to assess the available airtime for Wi-Fi communication.

In this showcase we demonstrate how a managed 802.11 Wi-Fi network makes use of WiPLUS to passively ***estimate the airtime occupied by LTE-U*** at each AP in order to perform better channel assignment and to load-balance client stations across the APs for maximizing the overall throughput/service quality.

2.1.1 Presentation of UPI Used

In order to support this showcase the Wishful control framework must provide the following functionality:

- UPI_R for passive detection of ***channel occupation by LTE-U*** at each AP,
- UPI_R/N for ***identification of own Wi-Fi network load*** at each Wi-Fi AP,
- ***Global Wishful control program*** which performs Wi-Fi network adaptation to dynamics in LTE-U channel occupation (-> adaptive duty-cycling) and own Wi-Fi network load:
  - Long-term: changing radio channels of APs,
  - Short-term: handover client STAs to less loaded adjacent APs.

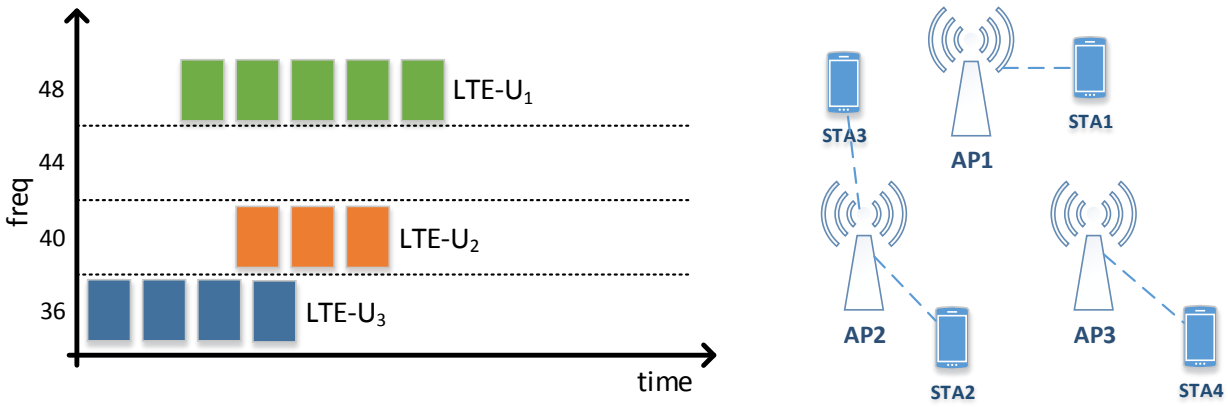


Figure 2. Estimation of channel occupation by LTE-U (left) and example Wi-Fi network (right).

For passive estimation of channel occupancy by LTE-U we implement an UPI_R function which informs the centralized Wi-Fi controller about the current LTE-U channel occupancy in real-time using events. Changes in the network load at each Wi-Fi AP are also reported using events. From this information the global controller is able to adapt to changes in LTE-U duty cycles and own network load by performing RF channel assignment to APs and by triggering STA handover operations.

2.1.2 Application

Figure 3 shows the Wishful framework components involved in this showcase. We have two local control apps reporting information about LTE-U channel occupation and Wi-Fi network load. This information is processed by joint channel assignment and handover control application residing on centralized compute node. Based on the reported data this application performs STA handover operation as well adjusts the AP's channel.

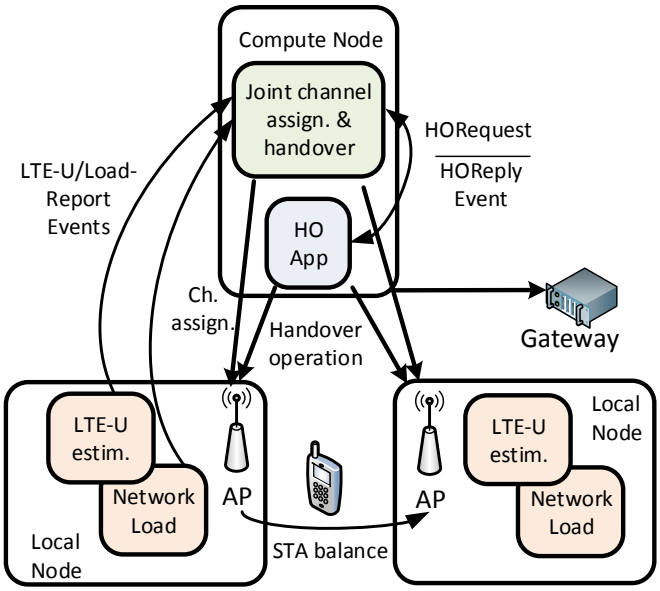


Figure 3. LTE-U Wi-Fi co-existence using Wishful control framework.

2.1.3 Results

We will compare our LTE-U aware channel assignment and STA load-balancing scheme with state-of-the-art solution, which takes only own network load into account. As performance metric we will estimate the average client throughput, throughput fairness and other metrics.

2.1.4 Next Steps

We are currently working on the implementation of the showcase and the setup for the demonstration.

2.2 SDR Capabilities

Software-Defined Radio (SDR) refers to platforms where the baseband processing is performed by software modules running either on field programmable gate arrays, digital signal processors, general-purpose processors or a combination thereof. As a consequence, operation characteristics of the radio technology, such as coding algorithm, modulation type, and channel access mechanism, can be easily changed, simply by loading different software. In addition, multiple radio devices with different characteristics can be replaced by a single SDR.

A typical SDR is modularized in two independent parts: the hardware (which is typically just a RF front-end/RF channelizer) and the software (a set of libraries to ease the development of signal processing algorithms). The most utilized and commercial RF front-end for SDR is the family Universal Software Radio Peripheral (USRP) from the Ettus Research (National Instruments - NI) company. USRPs are connected with a general-purpose machine (a personal computer or a notebook), executing the software responsible for the signal processing. Although several software solutions to perform such task are available, the two most known and utilized by the SDR development community are GNU Radio and IRIS.

We plan to illustrate the capabilities of WiSHFUL with both GNU Radio and IRIS in two independent showcases. With GNU Radio, we will showcase the execution of two independent virtual radios sharing one RF front-end simultaneously; while for IRIS, we will showcase the integration with UPIs by changing the parameters on the fly on the USRPs devices.

2.2.1 GNU Radio Capabilities: Radio Virtualization

The fifth generation (5G) of mobile networks is envisioned to overcome the fundamental challenges of existing cellular networks by providing higher data rates, excellent end-to-end performance, and user coverage in hot-spots and crowded areas with lower latency, and energy consumption. To efficiently support such a vast range of targets, providing a single air-interface "one-size-fits-all" is not a desirable choice. Instead, future mobile networks should become flexible, providing different air-interfaces for particular users and applications. Radio virtualization is considered a promising solution to reach this flexibility by enabling deployment of multiple virtual radios, i.e., air-interfaces, on top of one RF front-end.

The basic premise to radio virtualization is the adoption of a RF front-end dedicated only to frequency digitizing, and a virtualization manager, i.e., a Hypervisor in standard virtualization nomenclature. The Hypervisor is responsible for abstracting the RF front-end into a number of virtual front-ends (similar to what is done with virtual machines), and spectrum, which can be used by different radios. In addition, the Hypervisor is also responsible for scheduling the available resources between the multiple air-interfaces. Mobile operators can quickly introduce new or even experimental services in their production networks without affecting existing users or can follow a number of other deployment scenarios.

In this showcase, we demonstrate the implementation of a Hypervisor implemented in GNU Radio and managed through the WiSHFUL framework. Our showcase demonstration is shown in Figure 4. It will consist of (I) one USRP acting as a Remote Radio Head (RRH) which transmits LTE and NB-IoT data simultaneously, (II) one USRP acting as a LTE mobile subscriber, and (III) one USRP acting as a NB-IoT healthcare data receiver. At the RRH side, we will have a dedicated computer executing our SDR Hypervisor and the software radios for the LTE transmitter and NB-IoT transmitter. Similarly, we will have one dedicated computer for the LTE and NB-IoT receivers.

To provide interactivity in our demo, we will have the LTE virtual radio transmitting a video stream and the NB-IoT virtual radio transmitting data from a healthcare sensor worn by participants. The two displays show the video stream received in the mobile subscriber and the healthcare sensor in the NB-IoT. WiSHFUL framework will be used to manage the configuration of the Hypervisor and the virtual radios through a set of flexible and extensible UPIs.

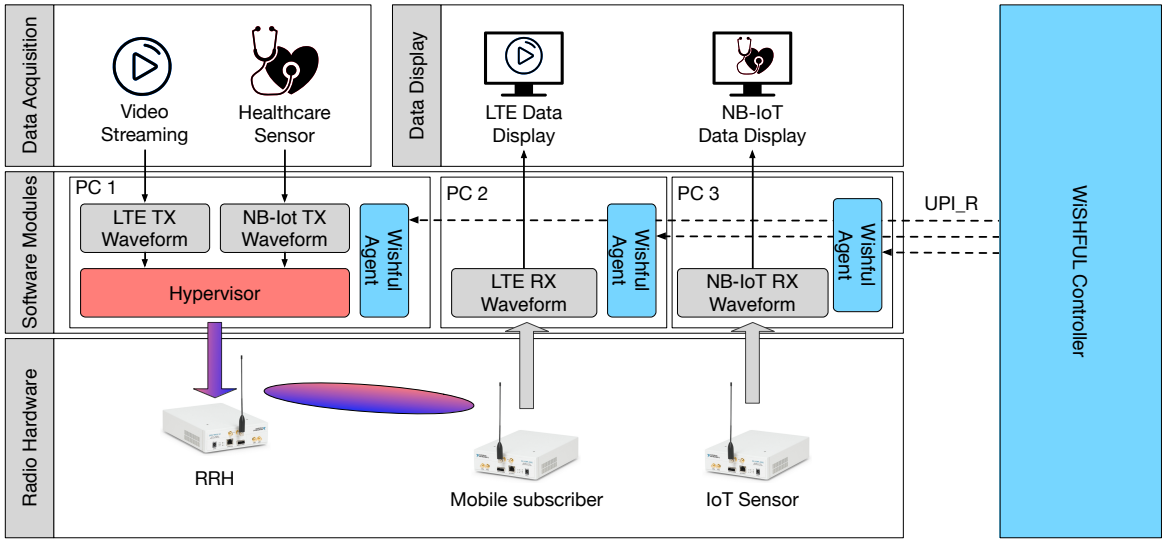


Figure 4 GNU Radio Virtualisation Component

a. Presentation of UPI Used

The UPIs provided by WiSHFUL to set-up and control our showcase, are as follows:

```
def setCenterFrequency(self, center_frequecy)
```

Configures the *center frequency* of a SDR RF front-end. This function returns a message (a python string) on the status of the operation, e.g., "SUCCESS", "ERROR: Frequency out of range", "ERROR: Frequency already in use".

```
def setVirtualRadioCenterFrequency(self, virtual_radio, center_frequecy)
```

Configures the *center frequency* of a given *virtual radio*. This function returns a message (a python string) informing the status of the operation, e.g., "SUCCESS", "ERROR: Frequency out of range", "ERROR: Frequency already in use".

```
def setBandwidth(self, virtual_radio, bandwidth)
```

Configures the *bandwidth* of a SDR RF front-end. This function returns a message (a python string) informing the status of the operation, e.g., "SUCCESS", "ERROR: this bandwidth is not accepted by the current RF front-end".

```
def setVirtualRadioBandwidth(self, virtual_radio, bandwidth)
```

Configures the *bandwidth* of a given *virtual radio*. This function returns a message (a python string) informing the status of the operation, e.g., "SUCCESS", "ERROR: bandwidth out of range", "ERROR: spectrum overlaps with another virtual radio".

These functions are used to change the configuration of the RF front-end (in this case a USRP) and virtual radios. As an illustrative example, consider the case of a NB-IoT virtual experiencing high interference from the LTE virtual radio. Using aforementioned WiSHFUL UPIs, the NB-IoT virtual radio can be configured to change its central frequency to one with less interference.

b. Results

With this showcase we intend to show the flexibility of managing virtual radios through the WiSHFUL UPIs. During the showcase, we use the UPIs functionality to demonstrate the impact of different bandwidths in the data rage of the receiving devices. For example, we can change the bandwidth of the LTE transmitter/receiver pair to accommodate a high-resolution video stream. Similarly, we can change the central frequency of the NB-IoT transmitter/receiver pair and evaluate the interference caused due the proximity with the LTE virtual radio spectrum.

c. Next Steps

The next steps include the implementation of UPIs to create and remove virtual radios on the Hypervisor on the fly. More advanced feature include the encapsulation of virtual radios as Virtualized Network Functions (VNFs) and their management throughput the WiSHFUL Framework.

2.2.2 IRIS Radio Capabilities

The Iris SDR Platform is a software framework for building highly reconfigurable radio networks. This architecture was developed using heterogeneous processing platforms including general-purpose processors, field-programmable gate arrays and the Cell Broadband Engine [6]. Its objective is to provide an environment for flexible reconfiguration of all the software radio components in real-time

with excellent performance. To achieve this goal, the platform offers interfaces to support on-the-fly modification of radio flow-graphs. This model supports adding and developing new radio components for the software platform. Due to the flexibility offered by components developed in software, any part of the radio can be reconfigured on the fly.

While platforms such as Iris SDR offer the necessary capabilities to enable dynamic support and reconfigurability, they can be quite complex for users to learn and utilise. These systems require deep knowledge of network protocols and software architectures, signal processing chains, cognitive radios, C++ libraries and code, and so forth. The WiSHFUL project aims to reduce the knowledge barriers to using advanced SDRs and experimentation facilities.

In this showcase, we aim to demonstrate the integration between WiSHFUL and the IRIS SDR by modifying the frequency on two IRIS SDR nodes, one transmitter (TX) and one receiver (RX), while executing a ping command. Our showcase demonstration is shown in the Figure 5. It consists of (1) two laptops configured with IRIS SDR software and WiSHFUL agent software, (2) one laptop running the WiSHFUL Controller, (3) two N210s with firmware UHD_003.005.005-0, (4) one Ethernet switch connecting the WiSHFUL Controller machine with the Tx and Rx laptops.

Figure 5 Topology used in the IRIS showcase

We demonstrate the following scenario: (I) one USRP transmits a OFDM waveform using TUN/TAP interface connected to one laptop. This task represents the transmitter running the Iris SDR Framework and WiSHFUL Agent software. The Tx node sends ICMP packets to the receiver with an interval constant in time.

The other laptop running the Iris SDR Framework and WiSHFUL Agent will receive the OFDM waveform using TUN/TAP interface via the USRP. This node represents the receiver, which it is going to receive the ICMP packets transmitted by the Tx node.

The WiSHFUL Controller laptop is responsible for sending a request to the Tx and Rx WiSHFUL Agents to change the *Frequency* parameter on both nodes. This node uses the IRIS WiSHFUL APIs to transmit the commands to the Tx and Rx nodes. The controller node supports dynamic reconfigurability of individual parameters of signal processing functionality in real-time on the Tx and Rx nodes. The result is the Tx and Rx nodes changing communicating frequencies on-the-fly, while continuing to send and receive ICMP packets. A video demonstration of this IRIS WiSHFUL showcase is available in [7].

a. Presentation of UPI Used

The UPIs provided by WiSHFUL to set-up and control the showcase is as follows:

```
@framework.bind_function(upis.radio.set_frequency)
def set_frequency(self, set_frequecy)
```

Configures the value of the ***frequency*** *in* SDR. This function returns a message (a python string) informing the status of the operation, e.g., "SET_FREQUENCY_OK", "ERROR: SET_FREQUENCY_NOT_OK".

```
@framework.bind_function(upis.radio.set_gain)
def set_gain(self, set_gain)
```

Configures the value of the ***gain*** in SDR. This function returns a message (a python string) informing the status of the operation, e.g., "SET_GAIN_OK", "ERROR: SET_GAIN_NOT_OK".

```
@framework.bind_function(upis.radio.set_bandwidth)
def set_bandwidth(self, set_bandwidth)
```

Configures the value of the ***bandwidth*** *in* SDR. This function returns a message (a python string) informing the status of the operation, e.g., "SET_BANDWIDTH_OK", "ERROR: SET_BANDWIDTH_NOT_OK".

```
@framework.bind_function(upis.radio.set_rate)
def set_rate(self, set_rate)
```

Configures the value of the ***rate*** *in* SDR. This function returns a message (a python string) informing the status of the operation, e.g., "SET_RATE_OK", "ERROR: SET_RATE_NOT_OK".

b. Results

In this showcase we demonstrate the integration between WiSHFUL UPIs and the Iris SDR Framework. Our goal is to modify the frequency (set_frequency) on the Iris SDR Tx and Rx nodes using the WiSHFUL UPIs. This integration supports individual parameter reconfigurability in real-time.

c. Next Steps

The next steps involve the implementation of UPIs with additional features such as changing cyclic prefix, modulation depth, sub carriers values, antenna type, and so forth, on the fly.

2.3 Coexistence of IEEE 802.15.4e TSCH with IEEE 802.11 networks

The Industrial, Scientific and Medical (ISM) spectrum of 2.4 GHz is a very busy populated frequency spectrum that is *by intention* shared among numerous access technologies. A very popular, if not the dominating technology is the omnipresent IEEE 802.11 (or Wi-Fi). Wi-Fi is by design primarily *optimized* and tuned for high throughput on a distributed best effort basis. On the other hand, the trend towards Cyber-Physical Systems (CPSs) caused increased popularity of many Wireless Sensor Networks (WSNs) technologies – such as WirelessHART, Zigbee, Bluetooth LE etc. – which also share this ISM band. The design of those technologies has been primarily driven by demands for long life thus energy efficiency, relatively low throughput rates and discontinuous data bursts. Devices using

these communication technologies are commonly battery powered and expected to operate over long periods (up to years) without maintenance.

The demand for low transmission power and limited communication ranges comes also as a disadvantage and vulnerability if they have to co-exist with other, much stronger, transmitters (such as Wi-Fi), which can impair communications.

Modern CPSs encompass more and more varied control applications (including industrial applications) with diversified, often very demanding requirements for the quality of communication. Especially in CPS contexts approaches face an increasing emphasis on reliability and latency requirements.

In real world settings it is foreseeable that different co-located wireless technologies will have to gracefully co-exist to fulfill own system goals. Unfortunately, this leads to interference between technologies and degraded performance. The discussion on the *interference* is given in Section 2.1. The related work addresses the problem to deduce something about the behavior of the other technology in order to make the interference less harmful, however with limited success.

The setting for this work is motivated by a demanding version of the above scenario. In particular, we consider coexistence of two prominent technologies that fulfill different goals in the envisioned setting. On the CPS side IEEE 802.15.4e Time-Slotted Channel Hopping (TSCH) [1], which is gaining momentum in both standardization and deployment and constitutes a technology for highly reliable control networks. On the other side the popular WLAN technology, IEEE 802.11 Wi-Fi [2], used for high throughput best-effort data streams. Such combination is quite typical for laboratories, and industrial settings where a plant infrastructure is run over a highly reliable and latency constraint TSCH network, while Wi-Fi networks, occupying the same ISM bands, are used for high throughput multimedia data. The challenge is to ensure that existence of Wi-Fi will not adversely impact any TSCH transmission.

The currently established and well-known mechanisms of the existing technologies – such as Carrier Sense Multiple Access (CSMA) – cannot be applied in such a scenario because of two reasons:

- TSCH slots might be scheduled to fulfill hard timing deadlines that *cannot* not be delayed nor interrupted (by Wi-Fi);
- Unicast transmissions in TSCH do not incorporate Clear channel assessment (CCA) mechanisms and *must* be avoided completely (even if the next slot is yet to begin).

Even with Wi-Fi's listen-before-talk carrier sensing capabilities, it leads to interference with TSCH. The Wi-Fi node has no means of knowing that a scheduled TSCH transmission is about to start claiming the channel. On the other hand, TSCH standard doesn't incorporate means to delay dedicated slots (nor is that desired for deterministic transmissions), as there is no CCA for those. In order not to compromise TSCH transmissions, Wi-Fi needs to be silent for the duration of TSCH transmissions.

The two technologies, although sharing the same ISM band, are incompatible with each other and are unable to decode each other's transmissions. This means there is no intrinsic way for Wi-Fi to learn from TSCH transmissions directly. Some sort of external information exchange is necessary in order to achieve co-existence.

In this study we advocate explicit information exchange and coordination of operation among the TSCH and Wi-Fi networks. We feature an approach in which the schedule, of time-critical but low volume communication in TSCH networks, is communicated to the WLAN systems, which delay their potential transmission in order to avoid interference. The much less critical WLAN traffic, being the best effort service, can tolerate such delays. Our approach opens a number of challenges. Most critical is the necessity to assure a precise time synchronization across the two incompatible systems.

We have designed a complete solution for coordinated cross-technology cooperation between both types of networks including a cross technology synchronization algorithm that allows a Wi-Fi network to acquire time reference signals from running TSCH network. This time reference signal is then used by a Wi-Fi network to keep the channel free for the time of TSCH transmissions and use regular (CSMA based) transmission during the rest of the time. We implement a prototype of the system using COTS Wi-Fi nodes equipped with *ath9k* based IEEE 802.11 wireless chipset, and TinyOS based fully TSCH compatible wireless sensor nodes.

2.3.1 Assumptions

We assume that TSCH and Wi-Fi networks are co-located in the same space and time and are required to coexistence without interference. Both networks are significantly overlapping in coverage and are internally using single hop communication in this scenario, i.e. we do not consider spatial reuse (multi hop scenario will be discussed in Section 9.1). Additionally, we assume that TSCH is running some sort of industrial application and requires high transmission reliability. Thus we would like to make sure that transmissions are not interfered by Wi-Fi, while co-located Wi-Fi runs on best-effort basis requiring mechanisms to disable all Wi-Fi activity during TSCH transactions.

We assume fully compliant TSCH devices [8], using the IEEE 802.15.4e Standard default configurations. It this scenario, it is not necessary for TSCH to follow an artificially specific schedule for our envisioned system to work. More details and necessary schedule properties are discussed in Chapter 5. As there is no possibility to directly exchange data between cross-technology devices, it is necessary to acquire the TSCH link schedule through a separate control channel. Solutions on how such data can be exchanged, is not a focus for this work.

2.3.2 Possible coexistence schemes

There is a number of possible approaches to enable coexistence of both technologies. First of all, we can separate them in frequency and temporal dimensions. We already assume they share the same space and that, as we are talking about incompatible physical layer technologies, we are not able to talk about code dimension.

a. Frequency separation

A very simple approach to let TSCH and Wi-Fi co-exist is to separate them completely in frequency. The first option is to exclude channels used by Wi-Fi network from the TSCH hopping scheme, thus achieving full frequency separation between both networks. This approach is specially promising in case of the high network load in Wi-Fi. On the other hand, in bigger deployments we can observe that multiple Wi-Fi APs are using multiple channels, thus there would be no free spectrum available for TSCH network.

The second approach is to make Wi-Fi avoid the TSCH transmissions in frequency. It is not very practical as the big advantage and interference resilience of TSCH is based on spreading transmissions over the whole ISM band and Wi-Fi does not employ frequency hopping mechanisms.

b. Temporal separation

The other approach is to enable both technologies coexist in time. Basic approach is already there at least on the Wi-Fi side. Wi-Fi stations employ CSMA protocol by default and thus will not start any transmission if they detect sufficient energy in the channel. This is however not a very reliable way of detecting other technologies (like IEEE 802.15.4) and often leads to false negatives.

It is also not practical to make a TSCH network avoid Wi-Fi transmissions in time. First, Wi-Fi sends data opportunistically, so it is not possible to reliably predict when the transmission will occur.

Secondly, TSCH as Time Division Multiple Access (TDMA) protocol does not employ CCA before transmission.

The only possibility is to perform interference avoidance in the Wi-Fi network. Namely, let the TSCH network provide scheduling information so that the Wi-Fi transmissions can be delayed to points in time where no collision with transmission from the TSCH network is guaranteed.

2.3.3 Followed approach

We will focus on the cross-technology TDMA protocol to coordinate the transmission between both types of nodes and reduce interference to a minimum. The system should run a hybrid TDMA scheme on the Wi-Fi nodes, which synchronizes to the TSCH TDMA slots. It then adapts time slots to keep free slots that are implicitly reserved for TSCH network, and uses the remainder for regular (CSMA based) transmission. This schedule, if executed perfectly, guarantees no cross-technology interference.

In our system we consider two networks. A TSCH based wireless sensor network running some industrial application and a Wi-Fi AP with its own stations. Example can be seen on Figure 6. Due to the incompatibility of the PHY layers between IEEE 802.11 and IEEE 802.15.4 it is not possible to directly capture packets from other technology.

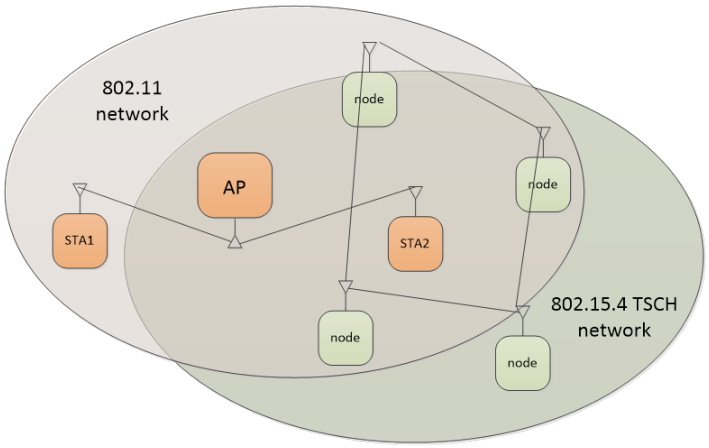


Figure 6 Example illustrating two co-located wireless networks

2.3.4 System design

In the proposed scenario and from the time-keeping perspective, it is possible to distinguish 3 separate time references:

- The IEEE802.15.4e TSCH based sensor network is a tightly synchronized wireless system where a particular node will adapt the clock based on own time parent but has no direct notion of the absolute time reference.
- Linux hosts keep own time reference in UNIX Epoch time and can synchronize it with other nodes over Network Time Protocol (NTP) or over Precision Time Protocol (PTP) where high timing accuracy is required.
- The IEEE802.11 uses Time Synchronization Function (TSF) to allow synchronization between devices in one BSS (Basic Service Set) e.g. AP puts own TSF counter value in the beacon packets, and all stations synchronize to it.

In order to achieve the tight cooperation between Wi-Fi devices and sensor network required for the co-existence use case (as described in D2.3), where Wi-Fi network will cease the transmissions when it would collide in time and frequency with the sensor node transmissions, it is necessary to

synchronize both networks. For start let us assume simplified problem with the focus on synchronization between one sensor node (NXP JN5168 running TinyOS version of TSCH) and one Linux host machine equipped with Wi-Fi card. There are two main options to synchronize those two devices. Over the direct wired connection or indirectly over wireless link. Both of the solutions have advantages and disadvantages. The wired connection promises more accurate and easier solution but requires tight coupling between the devices. On the other hand, the wireless solution is much more flexible in terms of coupling but requires more overhead in the detection of the signals sent by the different and in general incompatible technologies. In the next sections we will analyze both wired and wireless approaches.

The required synchronization accuracy depends on the time slot duration of the IEEE802.15.4e network. Although it is not fixed in the standard, the usual value is 10 ms. It means that the Wi-Fi should be able to cease own transmissions for the time of a TSCH slot. The inaccuracies of the synchronization can be adjusted with additional guard times, but it will additionally limit the performance of Wi-Fi network.

There is still a problem of schedule alignment with the synchronized networks. It is necessary to be able to calculate when the overlapping TSCH channel will be used. In the simple case, if there is a used cell that is scheduled to be transmitted on an overlapping channel, the Wi-Fi transmission will be deferred. In more advanced scenario, the system should be able to detect if the Wi-Fi transmission would harm TSCH nodes that are allowed to transmit in a given cell (i.e. are in the interference range of Wi-Fi). The more advanced scenario should improve Wi-Fi performance (we reduce the time in which transmissions are not allowed) without affecting the TSCH performance.

a. TSCH Schedule

The goal of this part of the work is to create a *spectral fingerprint model* that can be used to cross correlate against the recorded spectral data from the Wi-Fi radio. This requires two independent problems to be solved. Firstly, the schedule of TSCH needs to be available to Wi-Fi. Secondly, Wi-Fi needs to learn, and adapt to, the timing of TSCH to predict when the next active slot of the TSCH schedule is starting. Finally, the structure of schedules has strong implications on the detect-ability of the pattern and hence the ability to synchronize to it. Not all chosen schedules are equally qualified for our spectrum based synchronization approach. They can possess a hidden internal periodicity that can impede the option for synchronization.

The logical representation of all links in the network is called a *schedule*. Links on the nodes are organized in Slot Frames (SFs). Individual nodes only have knowledge of links that they participate in (either as source or destination). A SF has a certain length and repeats seamlessly in time. Multiple SFs can be contained in a schedule, but for this work we assume that only one SF is used. The global schedule therefore repeats with the same periodicity as the SF. Figure 7 shows an exemplary schedule containing only one SF with two links.

Two fundamental ways exist how to construct schedules, which represent also the network topology on the MAC level. Either connections are computed on a *central* computation logic and distributed to the nodes, or links are negotiated between nodes on a peer basis and instantiated in a *distributed* manner. Of course hybrid approaches of both are possible. In industrial settings, where reliability and latency constraints are vital, the *centralized* approach is typically chosen, for being better suited for optimization towards certain goals and constraints. This fact eases the process to acquire the schedule from the central computation logic to make it available to the Wi-Fi synchronization.

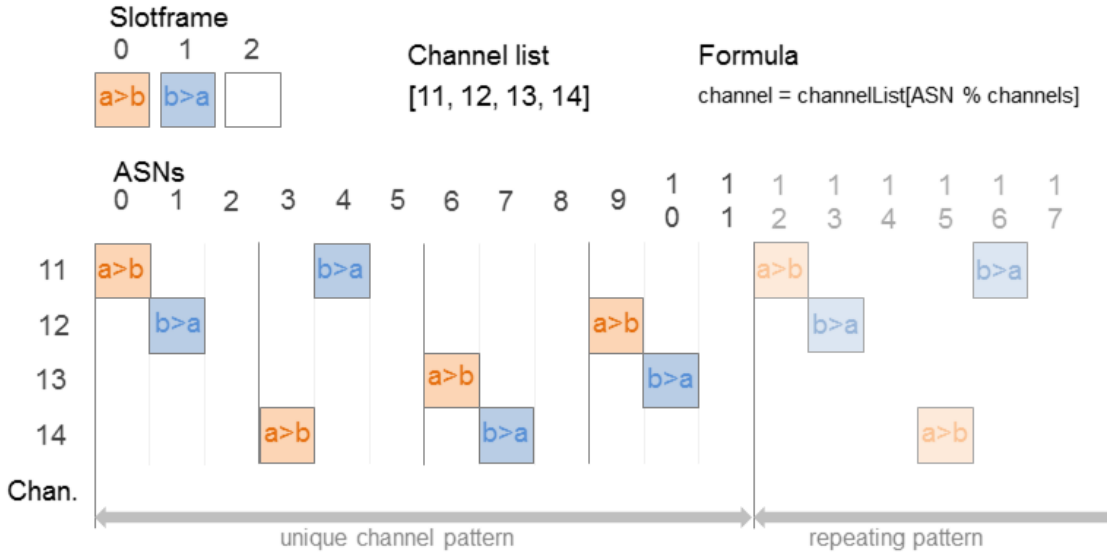


Figure 7 TSCH Schedule in time and channel

In the TSCH standard each node can be part of multiple networks, and thus have multiple schedules operating in parallel. This can of course lead to overlapping slots between multiple schedules. Collisions are resolved based on the schedule priority. In this work we consider TSCH schedules that consist of one SlotFrame (SF) on all participating nodes. This constraint is implemented to simplify the schedule exchange process between TSCH and Wi-Fi and to make the frequency pattern of TSCH more prominent. Implementing multiple SFs like in [9] is per-se possible, but the frequency pattern gets potentially much longer as well (if they are mutually prime to each other).

The foundation of our approach to synchronize Wi-Fi to TSCH is based on the intrinsic unique channel-hopping pattern of a TSCH network. In order to successfully synchronize to this pattern, it has to be acquired first by extracting all necessary information from the TSCH network. This includes the link list, the SF description and the ordered list of channels used for the channel hopping. Out of that it is then possible to compute the sequence of real channels used for that specific TSCH network. We leverage the fact that TSCH applies a random hopping mechanism that is actually repeating after some time. This waives the need to compute the channel hopping sequence for any specific point in time, as it would be for true random hopping. Instead we generate the entire unique sequence and use it to correlate the measurements against. The actual functioning of the pseudo-random scheme and its periodicity is discussed below in Periodicity in time and frequency. The computed sequence of channels for each link in the schedule gives a dataset like the one depicted in Figure 7.

We are specifically interested in the computed sequence of pseudo-random channels, as we use the resulting frequency-spectrum-pattern produced by TSCH to synchronize to the schedule without external synchronization mechanism. Even though the schedule repeats with total number of time slots in the SF, the resulting computed sequence of channels for the TSs repeats with a slower frequency. This is caused by the way the pseudo-random channel hopping is computed. The modulo operator in the channel formula causes the repeating SF to use different channels in successive iterations. The exact shift of channels per iteration depends on the actual lengths of the SF and channel list. There is an upper bound of unique time-channel pattern that is reached when SF-length and channel list length are mutually prime to each other (or prime numbers themselves).

The starting point for schedule model computations is the list of active links in the TSCH schedule. Links of a specific SF are defined with *source node*, *destination node*, *slot offset*, *channel offset*. The sample TSCH link description (code-name *olaf* used for experiments) can be seen in Figure 8. It was created in such a way to be most detectable with a very specific pattern with very low auto-correlation value.

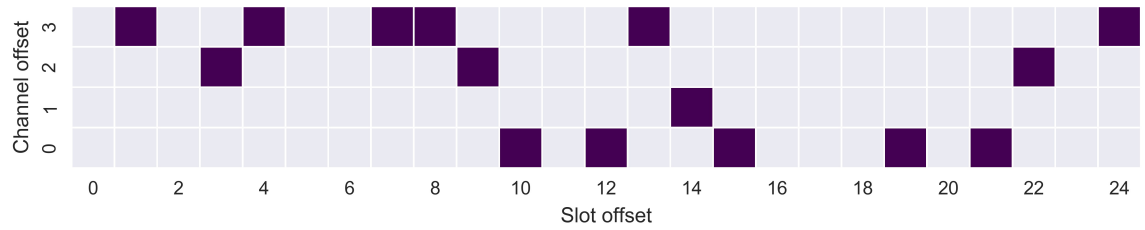


Figure 8 Example TSCH schedule

The relative offset parameters of links can be transformed into real channels and slots. Therefore, the ordered channel list, as well as the length of the SF needs to be known. The resulting data set contains elements with properties *source node*, *destination node*, *ASN* and *channel*. The used channel list in this example had the following channels {23,22,24,21} and the SF a length of 25. Both, number of channels and SF length, are no prime number but mutually prime. Consequentially, length (in Slots) of the resulting unique slot-channel schedule here is 100.

The final step is to transform the computed slot-channel pattern into the time-frequency domain. Meaning the discrete slots need to be spread to actual time values, while the channels are spread to discrete frequency values. Both operations are orthogonal and hence commutative. The resulting data structure contains then data point with properties of *source node*, *destination node*, *time* and *frequency*. This step is necessary for the subsequent cross- correlation with measured FFT data. In time domain the slot wise data needs to be resampled to match the sampling rate of the Wi-Fi spectral scan. The spreading in the frequency domain is a bit more complicated. The frequencies to be used for spreading the channels into are dictated by the FFT sampling of the Wi-Fi radio. A straight forward and simple solution is to ignore any spectral shape of IEEE 802.15.4-PHY transmissions and replace the discrete channel by a rectangular mask of all frequencies that fall into a band with +-1MHz around the channel center frequency. Another approach is to use a more sophisticated spectral model to mimic the received signal strength on the particular set of frequencies that is also received over the air by the Wi-Fi radio. The final generated data for the schedule shown above in Figure 8 is shown in Figure 9. We use this generated data later for cross correlating to the measured FFT data to synchronize Wi-Fi to TSCH.

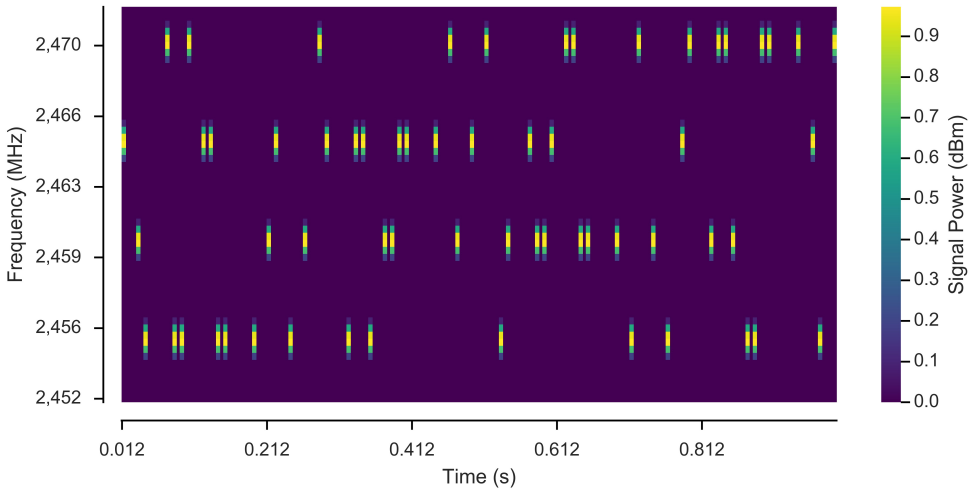


Figure 9 Unique time-frequency schedule model used for cross-correlation

b. Cross technology synchronization

One of the most challenging problems that needs to be solved in order to achieve the cross technology TDMA scheme is achieving a synchronization between two systems. In this Chapter we will focus on solving that problem, by using the *spectral fingerprint model* and correlating it with the *Wi-Fi spectral measurements* acquired from the *ath9k* based device.

Atheros 802.11n chipsets include a built-in spectral analysis features [10]. They are able to report the absolute magnitude of the signal from baseband FFT processing unit. The user of this functionality has an ability to control how often the spectral data is reported to the kernel and though *debugfs* interface to the user land process. The samples are time stamped with the accurate Timing Synchronization Function (TSF) values, as defined in IEEE 802.11. Those are used to extract accurate timing information from the data.

c. Cross technology TDMA

Having the Wi-Fi node synchronized to TSCH network we still need to make use of this synchronization in the hybrid TDMA scheme. As the result of cross-correlation we have achieved proper mapping between TSF, UNIX time and the TSCH slot start. The next step is to use this information to control the behavior of the Wi-Fi nodes.

We base the approach on hybrid medium access architecture [11] to configure behavior of the Wi-Fi nodes. The solution exploits the 802.11 power saving functionality to enable control of the software packet queues.

We are using the proposed and modified *ath9k* kernel driver to control the devices and un/pause software queues based on the detected TSCH schedule and timing. We have extended the PyRIC[1] Python library to support new kernel functionalities. With such addition, and with usage of persistent *Netlink* sockets, it is possible to achieve fast interface between control process running in Python and the kernel driver.

The proposed solution carries the same limitation as in [11] in terms of accuracy of the *Netlink* communication. Additionally, due to the current scheduling limitations, it was necessary to convert TSF timestamps to the UNIX host clock. This can lead to the additional inaccuracies and delays in the execution.

2.3.5 Presentation of UPI used and new

The main contribution of this showcase is a cross technology synchronization scheme between TSCH and Wi-Fi networks. In the first implementation it uses static information about the TSCH schedule, so natural extension is to use WiSHFUL UPI's to extract this information from dynamically changing TSCH network. The solution also uses the *ath9k* spectral scan functionalities already available in the WiSHFUL framework to gather information necessary for the time synchronization.

On the other hand, the synchronization scheme itself can be integrated in the WiSHFUL framework as a service. It doesn't provide any additional UPI by itself but allows for more fine-grained control of two technologies at the same time and in a coordinated manner.

The implementation of the scenario also leads to improvements in the Wi-Fi hybrid TDMA scheme that gives much more flexibility on the control of Wi-Fi node sleep timing, that is not any more controlled by fixed schedule. On the other hand, such solution requires much tighter (and local) control to keep Wi-Fi node operational. Note, too long sleeping period without control will result in the node disassociation from AP, or even AP losing all of its clients.

[1] https://github.com/wraith-wireless/PyRIC

2.3.6 Results

To prove the feasibility of the solution we have designed an experiment consisting of two nodes connected though variable attenuator (simulating wireless channel). One is a sensor node running TinyOS implementation of TSCH standard and sending packets. Second is an Intel NUC node equipped with *ath9k* based Wi-Fi card that is trying to detect TSCH transmissions.

The example data set is shown in Figure 10, and presents data measured by the Intel NUC device with *ath9k* based Wi-Fi card connected over wire with the NXP JN516x node with 93*dB* of attenuation. NXP JN516X was sending packets according to *olaf* schedule.

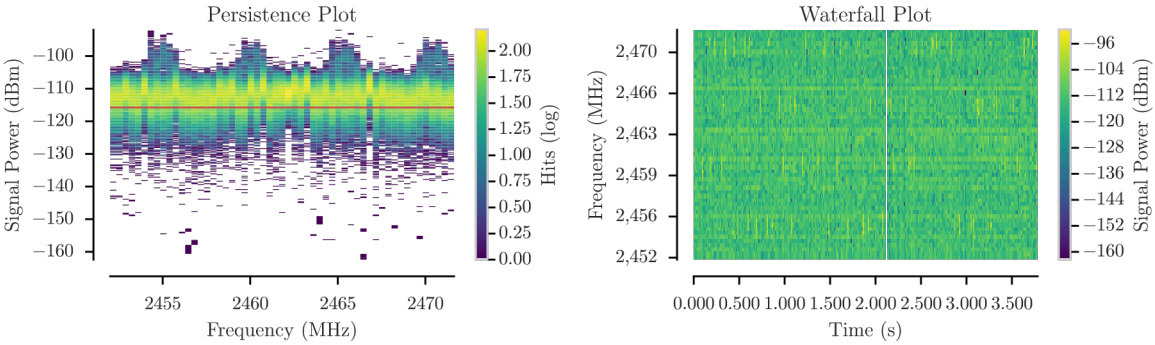


Figure 10 ath9k spectrum scan of TSCH node over cable

Result of the normalized correlation can be seen in Figure 11. If we do peak detection (marks) and map it to the other timestamps in the figure (TSF based) we get following time points: 0.715s, 1.715s, and 2.715s. At this point of time the TSCH schedule (presented in Figure 8) started. The correlated values are a lot larger than three standard deviations from mean and a high attenuation of the cable gives a strong indication of the applicability of the solution.

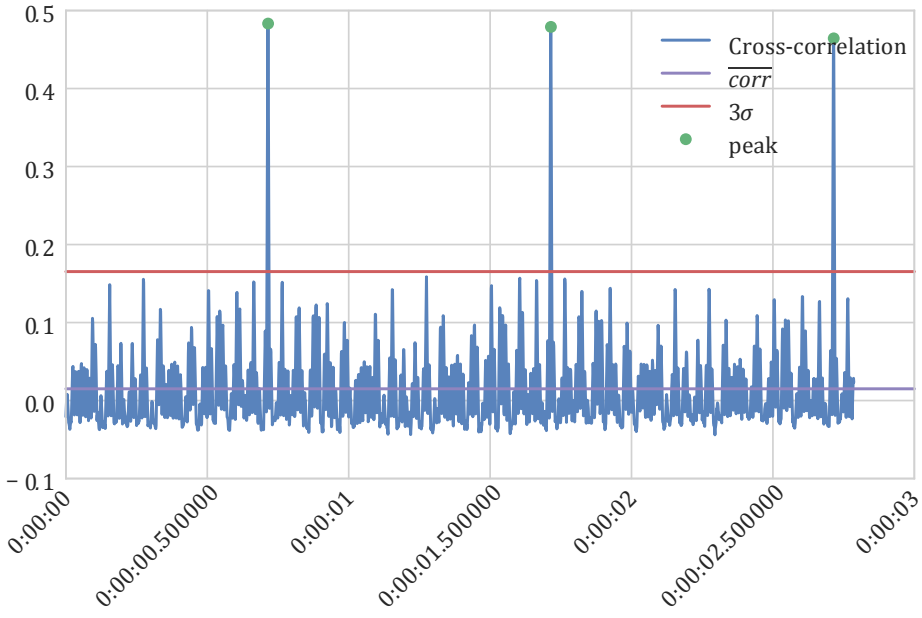


Figure 11 Cross-correlation of measurements with TSCH model

2.3.7 Next Steps

The whole implementation part of this showcase is finished. The next steps naturally include more extensive evaluation and experimentation. Namely, measuring packet losses in both TSCH and Wi-Fi and compare statistics of default operation and with our system running. This should be done with different TSCH schedules and varying traffic types in Wi-Fi (e.g. gradually increasing load in Wi-Fi network).

2.4 Radio Slicing for Virtualized Home Wi-Fi Access Points

Using several virtual wireless networks, identified through different service set identifiers (SSIDs) and basic service set identifiers (BSSIDs) on a single physical Wi-Fi Access Point (AP) is very common in todays' home Wi-Fi networks. The first virtual network usually provides the Internet connectivity for the resident or the owner of the AP. All further virtual networks are, for example, used for so-called "Guest Networks", Community Networks or Public Hotspots of cellular network providers. Albeit all of them are either used by different target groups, or are used to provide Internet access to secondary users. The primary user (AP owner) in the best case should not even be aware of the fact primary network connection is shared. While current approaches, e.g meSDN [12], are focused on traffic shaping or slicing on the wired network side for providing guaranteed bandwidth to primary users, to best of our knowledge, all current approaches are ignoring issues relate to the wireless MAC and PHY layers.

In this showcase we present a solution that considers the characteristics of today's home wireless access networks and applies a novel slicing technologies to guarantee bandwidth and traffic isolation for the primary users of the home AP in uplink and downlink. Figure 1 shows an example use case for this approach. The solution in based on the functionality presented in hMAC [11], which allows to pause and un-pause single software queues of the ATH9k driver on a per link basis.

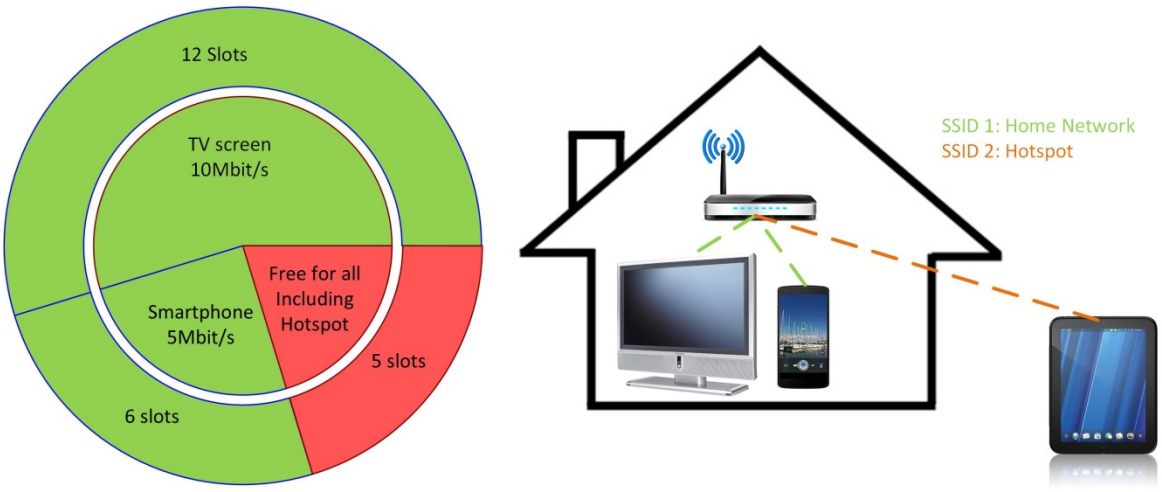


Figure 12 – MAC Layer Slicing in virtualized home Wi-Fi networks, enabled through slotted transmission on MAC layer. Device slices are dynamically adapted using MAC layer information to guarantee bandwidth.

The hMAC utilizes the existing ATH9K driver power saving implementation for the per link queue management. As the power saving functionality is running on the host, no modifications to the registers of the Wi-Fi device need to be done which preserves the standard MAC functionalities on

the device such as CSMA/CA. For this reason, the hMAC does not introduce additional unfairness and is fully compliant with the IEEE 802.11 standard.

We utilize this functionality to enable a slotted mode on top of ATH9K based Wi-Fi hardware. Multiple slots are combined to form a slice that is then assigned to a primary device. End-users are able to set fixed bandwidth guarantees for their primary devices. The radio slicer then takes care of assigning sufficient slots to the slices of the primary devices to ensure that the guaranteed bandwidth is delivered. This is done by considering the currently used physical bitrate of the primary device and the corresponding current success probability plus the current load and interference on the wireless channel. Doing so, every primary device will get its own explicit slice, while the remaining slots can be used by all primary and secondary (e.g. hotspot users) devices.

2.4.1 Presentation of UPI Used and New

In order to support this showcase the Wishful control framework must provide the following functionality:

- UPI_R for passive detection of current ***radio channel load***,
- UPI_R for to ***get currently used PHY rate of each client STA***,
- UPI_R for to ***get probability of successful frame transmission for specific physical rate,***
- UPI_R for ***controlling the hMAC*** (install, uninstall, update slot allocation),
- ***Local Wishful control program,*** which performs periodic slice adaptation based on channel load, client physical rate, transmit success probability and allocated user bandwidth. Moreover, the local control program has to update the slot allocation according to the computed slices by calling the hMAC UPI_Rs.
- **NodeRed Addon** for live visualization of the current throughput, currently used PHY rate and slice size during the showcase presentation as shown in Figure 14.

For passive estimation of channel usage, we implement an UPI_R function that can be called to deliver the currently used airtime of the radio channel. This value is extracted out of the registers of Atheros based Wi-Fi cards. Moreover, we implement a UPI_R function that delivers the currently used physical rate of all associated devices (achieved by iw calls) and a UPI_R function that delivers the probability of a successful frame transmission for this physical rate (achieved by querying the Minstrel rate table). The local control program then dynamically adapts the slice size assigned to each device, uniquely identified via its MAC address, during runtime, based on this information to the desired bandwidth, pre-set before.

2.4.2 Application

Figure 14 shows the Node-RED configuration to visualize the showcase. During the showcase, devices will be sequentially turned on which will be visible for the audience in the node red visualization, e.g. Figure 13. Moreover, it is possible to see that the guaranteed bandwidth for the primary users will always stay above the guaranteed level, while the throughput of the secondary devices may decrease depending on the channel conditions and traffic loads.

2.4.3 Next Steps

We are currently working to complete the implementation of the showcase and the setup for the demonstration.

Figure 13– Node Red live visualization showing troughput, PHY rate and slice size of single Wi-Fi devices

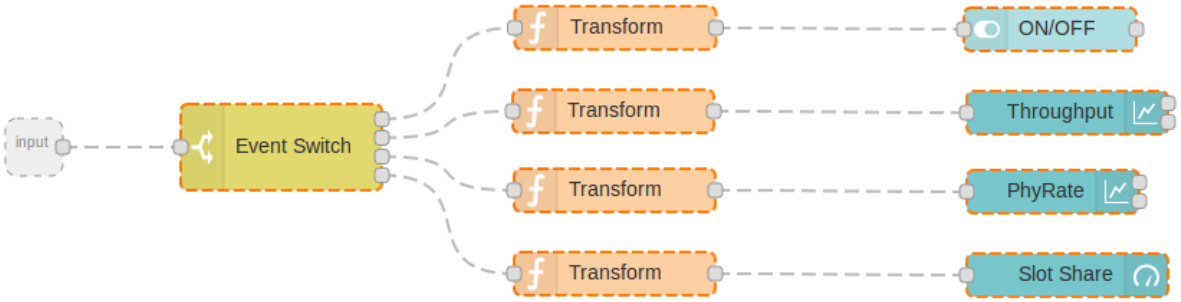


Figure 14 – Node Red graphical representation of the showcase visualization

2.5 MAC adaptation in presence of legacy stations

This showcase generalizes the results presented in D2.3 on the possibility of defining MAC adaptation logic devised to perform optimizations under varying load conditions (*Load & interference aware MAC adaptation* showcase), by dynamically tuning the contention window of the wireless nodes as a function of the number of active nodes, and by switching to a time-division access protocol in case of severe congestion levels. While in Year 1 we considered that the same control program was running on each contending node, supporting the WiSHFUL UPI, in this generalization **we consider the possibility that some wireless nodes cannot be directly controlled by WiSHFUL**. In particular, although the original showcase was demonstrated with both 802.11 and 802.15.4. nodes, we refer to wireless local area networks based on the 802.11 technology. Indeed, **the logic to be deployed for optimizing network performance in presence of legacy nodes has to take into account the**

peculiarities of the radio technology, and the indirect form of control that can be achieved on legacy nodes by using the standard control mechanisms (rather than the WiSHFUL UPI). Note that the coexistence with legacy nodes is a very crucial aspect for the impact of the solutions proposed within the WiSHFUL framework: in fact, we can reasonably assume that wireless nodes supporting WiSHFUL UPI in real deployments can be a negligible fraction of the total wireless devices in short-terms. However, the demonstration of WiSHFUL benefits even in presence of a few controllable nodes can be a valid argument for incentivizing the adoption of WiSHFUL.

Scenario and adaptation logic. We consider a wireless local area network with a single contention domain, in which all nodes are in radio visibility and transmit frames according to the IEEE 802.11 format and modulation schemes. In this domain, some nodes, including the Access Point, are WiSHFUL compatible, i.e. expose the WiSHFUL UPI and are able to run local control programs, while some other nodes follow the legacy 802.11 MAC/PHY protocols. The number of legacy nodes N and WiSHFUL nodes M can be tuned in different experiments; the specific network topology can be completely characterized by the couple of values (N, M).

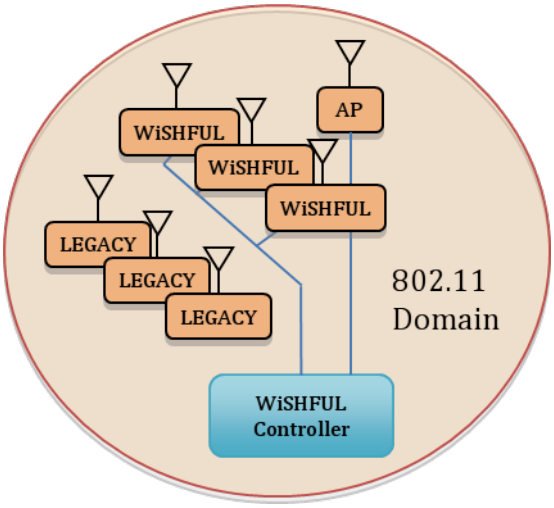


Figure 15 – Network topology used in the showcase

As widely documented in literature, different **optimization functions** and **load metrics** can be considered according to the desired performance metric. Each optimization can be generalized and adapted in presence of legacy nodes. In our showcase, we limit the analysis to optimizations based on the tuning of the contention window values and discuss, from a functional point of view, how addressing coexistence under heterogeneous MAC protocols. Indeed, nodes using heterogeneous contention windows can easily coexist in the network (i.e. they do not disturb each thanks to the carrier sense and backoff freezing mechanism). The only coexistence problem can be an unfair repartition of the channel resource because stations receive a number of transmission grants that, in long terms are inversely proportional to the employed contention window value. Conversely, nodes running CSMA protocols cannot safely coexist with nodes running a TDMA protocol, because scheduled-based transmissions can collide with contention-based transmissions starting before the scheduled time.

Case 1: In Year 1 we used a first tuning function of the contention window, called **Moderated EDCA backoff (MEDCA)**, whose goal is the minimization of the delay jitters on the channel access times. It is well known that these jitters depend on the exponential backoff mechanism, which introduces short-term throughput unfairness among the stations, and significant variabilities on the time between two consecutive channel accesses performed by the same station. Under MEDCA, delay jitters are avoided by adopting a fixed contention window. This mechanism can natively support

coexistence with legacy nodes, as currently proposed in a standard amendment. This is because the fixed value is set to the average contention window value experienced under exponential backoff by the same number of contending nodes. Since the throughput performance of each station depends on the channel access probability, which in turns is only function of the average contention window, the moderated EDCA scheme is able to minimize the delay jitters while guaranteeing the same average throughput of legacy EDCA stations. The only difference is that WiSHFUL nodes working with the fixed contention window will experience lower delay jitters than legacy nodes, without increasing their throughput.

Case 2: In Year 1 we used a second tuning function for the contention window, called **CW optimum (CWopt)**, that enforces all the contending nodes to use the theoretical value which optimizes the throughput performance. Such an optimal value depends on the number of nodes contending in the network, as indicated below, where the number of contending nodes is n_{flow}, and the collision time and backoff slot are given by $t_{collision}$ and t_{slot}.

$$CW = n_{flow} \cdot \sqrt{\frac{2\, t_{collision}}{t_{slot}}} \qquad (1)$$

For generalizing this optimization logic in presence of legacy nodes, we theoretically studied the throughput results that can be achieved in a network with N legacy nodes and M WiSHFUL nodes, by assuming that the transmission probability τ_W of WiSHFUL nodes is an independent variable (which can be tuned by opportunistically choosing a contention window value equal to $2/\tau_W$, while the transmission probability τ_l of legacy nodes cannot be chosen and depends on the exponential backoff mechanism and experienced collision probability. The results of this study are summarized in Figure 16, where the green curves represent the total throughput achieved in the network, the blue curves represent the per-station throughput achieved by legacy stations and the red curves represent the per-station throughput achieved by WiSHFUL nodes. The figures shows different network scenarios, represented by the couple of values (N,M) which quantify the number of legacy and WiSHFUL nodes. Obviously, the (0, 20) case corresponds to the case considered in Year 1, in which all nodes can be controlled by WiSHFUL. From the figure it is evident that a simple throughput maximization cannot be a practical solution. Indeed, maximizing the total throughput can lead to minimizing channel access grants for WiSHFUL nodes, thus reducing the contention level to the legacy nodes only (which achieve throughput results much higher than WiSHFUL nodes). Only for very unbalanced cases (e.g. (1,19)) with a few legacy nodes, throughput maximization leads to comparable results for WiSHFUL and legacy nodes. The figure also shows (with an explicit arrow) the τ_W value selected by MEDCA, which corresponds to exactly the same throughput for legacy and WiSHFUL nodes. We concluded this analysis by excluding the possibility to only work on WiSHFUL nodes for optimizing the network performance, and by considering alternative solutions for indirectly affecting the behaviour of legacy nodes. A closer look to the 802.11 standard reveals that legacy nodes are not constrained to perform exponential backoff rules from the usual CWmin=16 and CWmax=1024 values, but can employ alternative contention window limits signalled by the Access Point in the beacon frame. The contention window values can be differentiated among heterogeneous service classes, including best effort serving class, by means of a dedicated *information element*. The beacon frame can be exploited also for implementing an alternative control mechanism: legacy nodes can be prevented from accessing the wireless channel in some portions of the beacon interval or in some beacon intervals, by specifying the start of a contention-free period that can be ignored by WiSHFUL nodes. Since this mechanism is compatible even with 802.11 legacy stations not supporting EDCA, we decided to explore this solution for using the standard-based signalling mechanisms deployed by means of the beacon frames as an indirect control mechanism for legacy stations. The idea is using the optimal contention window value for WiSHFUL nodes, computed considering only M contending nodes, in the beacon intervals allocated to WiSHFUL nodes, and the legacy exponential backoff in the beacon intervals allocated to legacy nodes. The WiSHFUL enabled Access Point could further

guarantee fairness among WiSHFUL and legacy nodes, by choosing the number of beacon intervals allocated to legacy and WiSHFUL nodes proportionally to the number of per-class nodes.

Figure 17 shows a graphical representation of our envisioned scheme by illustrating the channel access operations performed in 4 consecutive beacon intervals. In this example, we assume that N=M; therefore, the beacon intervals are alternatively allocated to legacy and WiSHFUL nodes (intervals 1 and 3 are allocated for legacy node, and intervals 2 and 4 are used for WiSHFUL nodes employing a fixed contention window evaluated by Figure 16 with n_{flow}=M).

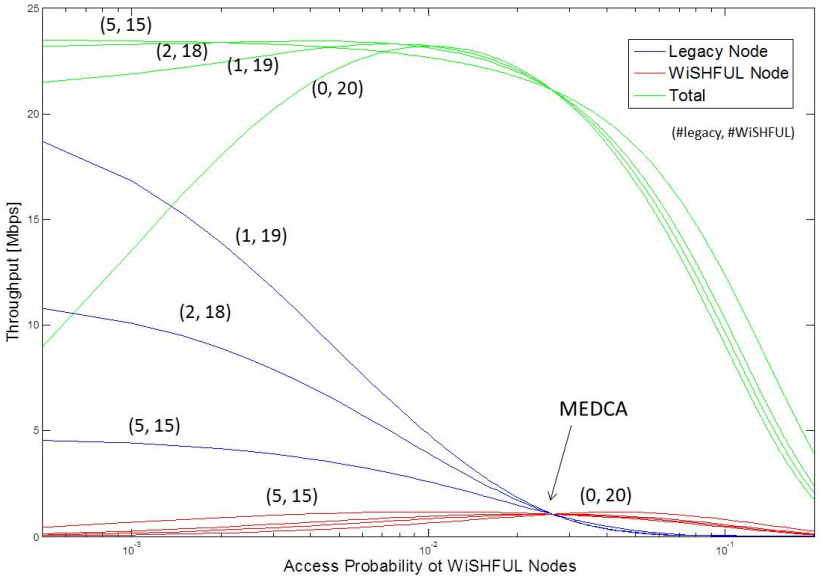


Figure 16 – Throughput results as the WiSHFUL nodes vary their contention window, in case of coexistence with legacy DCF nodes: (x, y) label refers to x legacy nodes and y WiSHFUL nodes.

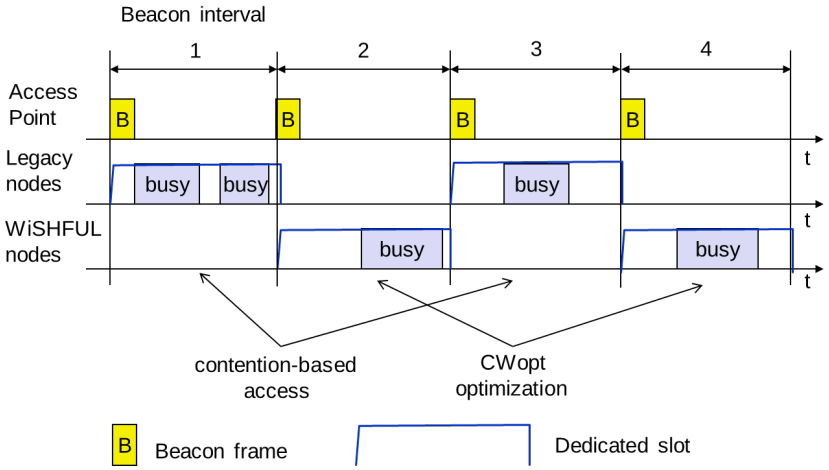


Figure 17 - Graphical representation of the implemented CWopt algorit

Case 3: In Year 1 we considered a last optimization stage, in which WiSHFUL nodes were forced to switch from random-based contention to a TDMA access scheme by a global controller. Although we have not implemented the generalization of this optimization scheme, previous considerations about case 2 suggest an easy solution for also supporting TDMA protocols on WiSHFUL. Indeed, since WiSHFUL nodes and legacy nodes access the channel in independent beacon intervals, it is possible

to also deploy heterogeneous access rules within the allocated channel time. The Access Point can implement a hierarchical scheduling scheme, devised to first allocate beacon intervals to legacy and WiSHFUL nodes, and then to allocate channel slots within the TDMA beacon interval to each WiSHFUL node.

2.5.1 Presentation of UPI used and new

Table 1 reports the list of WiSHFUL UPI functions used for the implementation of this showcase.

UPI	Description / Usage
`hc.start_local_control_program`	Setup and run custom local controller;
`controller.send / controller.recv`	Send and receive message between global and local controller;
`radio.get_measurements`	Get MAC/Radio measurement, extract info on freezing time and packet statistics;
`radio.set_parameters`	Set MAC/Radio parameters; here we set an optimized CW==CWmin==Cwax;

Table 1 - UPI functions used in showcase

The following code in Table 2 shows the control program, which is responsible of injecting the local control logic and performs the node bootstrap phase. The local logic is implemented in the local control program *(myargs)* function, which performs the tuning of the contention window according to the case 1 or case 2 mechanisms. Input arguments are given by the *send* method of *lpcDescriptor* object. For a given list of *legacy_nodes* and *wishful_nodes*, the controller configures the testbed nodes for running legacy DCF or the MAC-enhanced scheme.

```
[…]
lcpDescriptor_wishful_nodes = []
mac_mdl='MEDCA' #possible values: 'legacy-CSMA, MEDCA, CWopt
for ii in range(0,len(mytestbed.wishful_nodes)):
    lcpDescriptor_wishful_nodes.append(

controller.node(mytestbed.wishful_nodes[ii]).hc.start_local_control_program(program
=local_control_program))
    lcpDescriptor_wishful_nodes[ii].send( {'interface' : 'wlan0', 'mac_logic' :
mac_mdl})

lcpDescriptor_legacy_nodes = []
for ii in range(0,len(mytestbed.legacy_nodes)):
    lcpDescriptor_legacy_nodes.append(

controller.node(mytestbed.legacy_nodes[ii]).hc.start_local_control_program(program=
local_control_program))
    lcpDescriptor_legacy_nodes[ii].send( {'interface' : 'wlan0', 'mac_logic' :
'legacy-CSMA'})
[…]
```

Table 2 Control program code

The optimization strategy utilized here is implemented in the following code in Table 3. Two optimization algorithm can be adopted and, for sake of simplicity, we deployed a unique local control program in which are implemented both the tuning mechanisms. The mechanism selection is defined during the controller bootstrap by a configuration parameter. In this case the configuration

parameters can take three possible values: i) legacy-CSMA, ii) CWopt, and iii) (Moderated Backoff) MEDCA.

The main function is customLocalCtrlFunction; it performs three different MAC behavior:

1) Legacy-CSMA: only measurement are provided and no enforcement on MAC behaviour is provided;
2) Moderated Backoff (MEDCA): the local controller extracts periodically a *count_freezing* number from the last measurement returned by the UPI_R monitor function. The *last_count_freezing* is a program variable used for detecting the periodic overflow of the counter. After filtering the measurement, the novel contention window is computed by using an heuristic formula, which relates the average contention window of legacy EDCA stations to the number of backoff freezes, also called IPT (interrupts per transmission). The computed value is filtered to avoid sudden modifications on the station access rates.
3) Optimum contention window (CWopt): local controller retrieves periodically the number of active WiSHFUL nodes. The contention window value is computed as a function of number of WiSHFUL nodes involved in the experiment and depends also on the packet length.

The novel contention window value is enforced by using the UPI_R function responsible of configuring lower layer parameters. The tuning logic is executed independently and locally by each node, by executing the function *customLocalCtrlFunction(myargs).*

```
"""
Custom function used to implement local WiSHFUL controller
"""
def customLocalCtrlFunction(controller, interface, mac_mdl):

    # import modules and library
    import time
    import logging
    import math
    # initialization algorithm variables and node parameters
    b = 0.3
    a = 0.1
    last_count_freezing = 0
    CWMIN = 15
    CWMAX = 1023
    ipt = 0
    cw_f = CWMIN
    cw = cw_f;
    ip_address = controller.net.get_iface_ip_addr(interface)

    #local controller loop
    while not controller.is_stopped():

         #receive message from controller
        msg = controller.recv(timeout=1)
        if msg:
                # to retreive traffic numbers
                n_tx_sta = msg["traffic_number"]
                # to retreive used MAC logic
                alg = msg["mac_mdl"]
            log.warning("num_tx_nodes=%d" % n_tx_sta )

        #get node statistics
        UPI_myargs = {'interface' : interface, 'measurements' :
[UPI_R.COUNT_FREEZING, UPI_R.IPT,
             UPI_R.NUM_TX_DATA_FRAME, UPI_R.NUM_RX_ACK, UPI_R.NUM_RX_ACK_RAMATCH,
             UPI_R.BUSY_TYME , UPI_R.TSF, UPI_R.NUM_RX_MATCH] }
        node_measures = controller.radio.get_measurements(UPI_myargs)
```

```
[…]
        if alg == "MEDCA" :
            """""""""""""""""""""""""""""""""""""""""""""""""
            execute MEDCA algorithm and find new CW value
            """""""""""""""""""""""""""""""""""""""""""""""""
            delta_freezing = count_freezing - last_count_freezing
            last_count_freezing = count_freezing
            ipt = ipt + a * (delta_freezing - ipt);
            targetcw = -0.0106 * ipt ** 2 + 2.9933 * ipt + 18.5519  # determine the
target CW for this IPT
            cw_f = cw_f + b * (targetcw - cw_f)
            cw = round(cw_f)
            cw = int(cw)
            cw = max(cw,CWMIN)
            cw = min(cw,CWMAX)

        if alg == "CW_OPT":
            """""""""""""""""""""""""""""""""""""""""""""""""
            execute CWopt algorithm and find new CW value
            """""""""""""""""""""""""""""""""""""""""""""""""
            Tc = t_data + EIFS; #Collision time
            cw_f = n_tx_sta * math.sqrt(2*Tc / tslot)
            cw = round(cw_f)
            cw = int(cw)
            cw = max(cw,CWMIN)
            cw = min(cw,CWMAX)

        #update CW
       if not (alg == "legacy-CSMA"):
            UPI_myargs = {'interface' : interface, UPI_R.CSMA_CW : cw,
UPI_R.CSMA_CW_MIN : cw,

UPI_R.CSMA_CW_MAX : cw }
            controller.radio.set_parameters(UPI_myargs)

    return 'Local WiSHFUL Controller END'
```

Table 3 Optimization strategy utilized

2.5.2 Results

a. Results with MEDCA algorithm (Case 1)

This subsection reports the results of the experiments based on the MEDCA MAC adaptation. We activated 6 wireless nodes contending under greedy traffic sources towards a common Access Point. For this experiment, we consider a modulation rate of 24Mbps and a frame lengths of 200 byte. The experiment duration is 60 seconds. We first consider a legacy CSMA protocol with exponential backoff. Figure 18 shows the throughput performance achieved by each station and some regular samples of the contention window values (gathered by the WiSHFUL UPI_R) employed by the stations. Although the CSMA protocol in principle should provide an equal share of the network throughput to each station, we can observe some short-term and long-term throughput variability due to the exponential backoff mechanism (short-term) and to the location-dependent interference conditions suffered by each station (long-term).

For three of the nodes, we then activate the local control logic implementing the moderated EDCA backoff scheme. Figure 19 shows that the three stations achieve an average throughput comparable to the one experienced when they were using exponential backoff, but with smaller fluctuations. This is confirmed by the samples of the contention window values, that exhibit very small variations (from 20 to about 25).

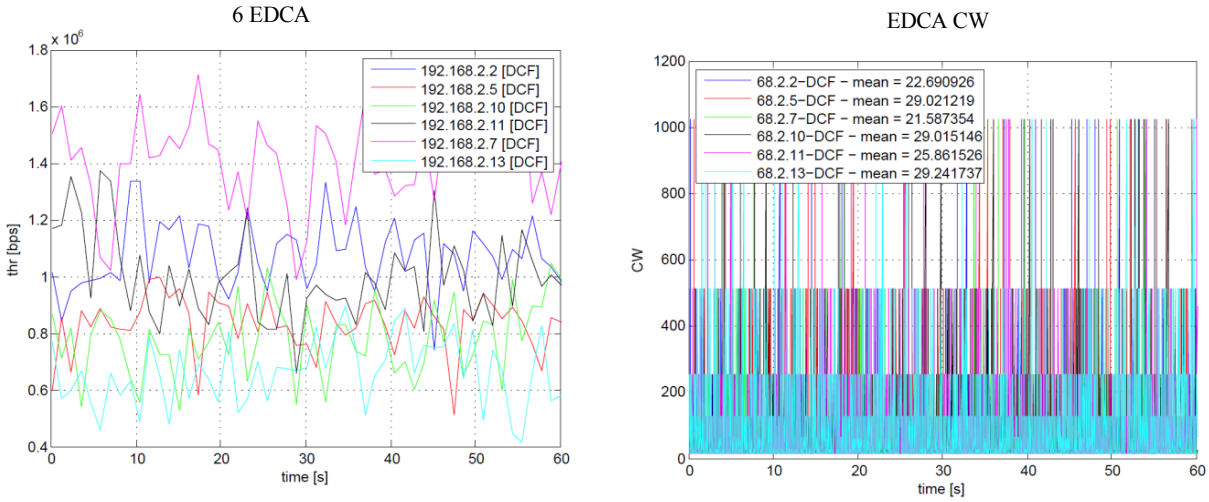


Figure 18 – Throughput performance and Contention Window samples in an experiment with 6 wireless nodes executing CSMA with exponential backoff.

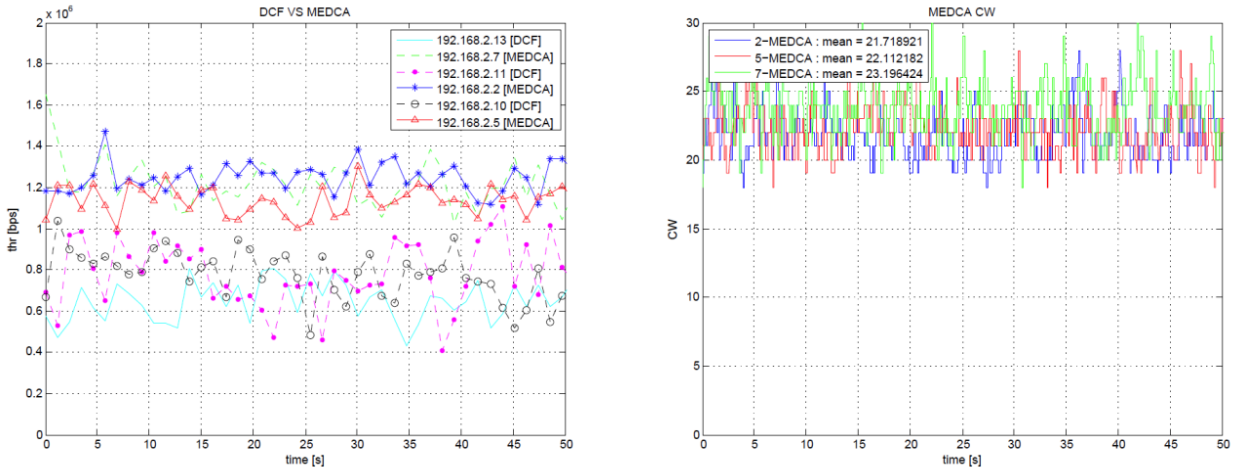


Figure 19 Throughput performance and Contention Window samples in an experiment with 3 stations employing moderated backoff in contention with 3 legacy stations.

b. Results with CWOPT algorithm (Case 2)

This subsection reports the results of an experiment when WiSHFUL nodes employ the CWopt MAC adaptation logic and when they are forced to access the channel during the contention-free periods signalled by the Access Point beacon frames. The network scenario includes a total number of 10 wireless nodes, 5 of which employ legacy DCF and other 5 are using WiSHFUL CWopt MAC logic. For this experiment we used a modulation rate of 24Mbps and a frame length of 1470 byte. The experiment duration is 60 seconds. Figure 20 (left) shows that all the wireless nodes involved in the experiment achieve almost the same throughput, because the beacon intervals allocated to each class of nodes are the same. However, WiSHFUL nodes slightly optimize their throughout performance by using the optimal contention window value. Figure 20 (right) shows the value of the contention window value (namely, 57) evaluated by the CWopt algorithm and used by WiSHFUL nodes.

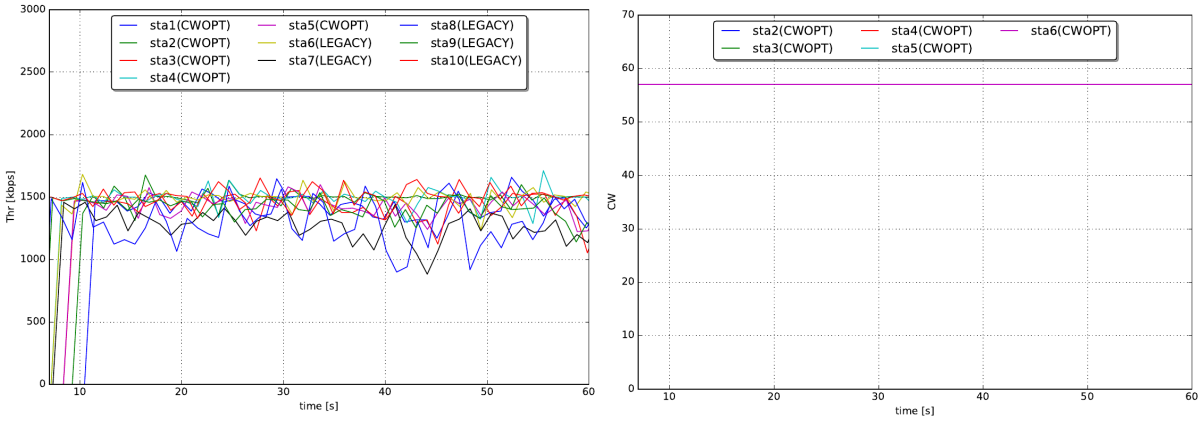


Figure 20 - Throughput performance and Contention Window samples of 5 stations employing CWopt in contention with 5 stations employing exponential backoff

2.5.3 Next steps

In conclusion, in this showcase we proved that the optimization logic can be implemented on WiSHFUL programmable nodes with or without presence of legacy nodes. In particular, we considered a first mechanism, called Moderated backoff, which does not require any coordination with legacy nodes, and a second mechanism, called optimal CW, which requires indirect forms of coordination with legacy nodes based on control mechanisms defined in the standard. Next generalizations of this showcase will involve the solution described as case 3, i.e. the possibility to guarantee coexistence among heterogeneous access protocols, and the definition of scheduling mechanisms to be implemented at the Access Point for dynamically allocating the beacon intervals to WiSHFUL and legacy nodes.

2.6 Load and topology aware networking

In dynamic wireless networks the application requirements vary over-time. Moreover, networks can grow or shrink as a result of node mobility. Network protocols designed for such networks (e.g. 6LowPan and RPL for sensor networks and IEEE802.11e and OLSR for Wi-Fi networks) have built-in support for allowing such dynamic behaviour. The standards, defining these protocols, allow fine-tuning the protocol operation via configuration parameters, enabling different performance metrics trade-offs evaluation. The implementations of these protocols, however, do not provide a unified interface for this purpose. These showcases will demonstrate how the WiSHFUL UPIs can be used to (i) dynamically monitor the network performance and topology; (ii) change network protocol configuration; or (iii) switch routing modules. Two showcases were implemented:

Lowering frame loss in highly mobile networks: illustrates how dynamic reconfiguration of frame aggregation and PHY rate adaptation parameters can lower the frame loss. Because node mobility intensifies the time varying nature of wireless channel, the network stack has to be re-configured according to the degree of the mobility. For this purpose, the node mobility is monitored and, based on the level of mobility, the aggregation level and/or modulation and coding scheme (MCS) index are reduced or increased in real-time. By lowering the aggregation level (e.g. number of frames) or MCS index in case of higher node mobility, the impact on bit errors, due to mobility, can be reduced during frame reception.

Detecting the optimal link estimation algorithm in various network topology scenario's: demonstrates that a global control program, controlling a sensor network, can increase the overall network performance by dynamically selecting the optimal link estimation algorithm. For this purpose, the network topology and performance is monitored and, the optimal link estimator is

choosen in each scenario. In sparse networks, simple link estimators (e.g. objective function 0, ETX) are preferred; in dense network more complex link estimators (e.g. 4BIT, fuzzy LQE) are more appropriate.

2.6.1 MCS Selection

IEEE 802.11 is evolving from 802.11a/b/g to 802.11n/ac in order to meet the much-needed high-throughput demand of smartphones, laptops, and tablet PCs. To achieve high throughput, IEEE 802.11n defines two types of frame aggregation: MAC service data unit (MSDU) aggregation and MAC protocol data unit (MPDU) aggregation. The latter, named the aggregate MPDU (A-MPDU), amortizes PHY protocol overhead over multiple frames by packing several MPDUs into a single A-MPDU. It is generally considered that A-MPDU is more efficient in !-prone environments thanks to the usage of block acknowledgements (BlockAcks) which allow each of the aggregated MPDUs (i.e., A-MPDU subframes) to be individually acknowledged and selectively retransmitted.

Understandingly, it has been believed that longer A-MPDU conveying more A-MPDU subframes always achieves higher throughput by reducing protocol overheads. All the existing studies have found the optimal length of MAC frames based on mathematical analysis and/or simulations, assuming a uniform distribution of errors across an entire A-MPDU.

However, our experimental results reveal strong evidence that the distribution of errors over the entire A-MPDU is not uniform, especially, for mobile users. For example, we observe that when long A-MPDU frames are used, the throughput is reduced by up to two thirds regardless of the channel condition at the receiver in time-varying channel environment, even if an appropriate PHY rate is selected. Furthermore, we find many scenarios where the performance actually degrades as the length of A-MPDU increases due to the limited channel compensation procedure executed by Wi-Fi devices. In such cases, the channel state information (CSI) measured using the physical layer convergence protocol (PLCP) preamble at the beginning of the A-MPDU may no longer be valid for subframes in the latter part of A-MPDU under the time-varying channel. Specifically, the subframe error rate (SFER) may increase as the time gap between the preamble and subframe increases, since automatic gain control (AGC), timing acquisition, frequency acquisition, and channel estimation steps are conducted only during the PLCP preamble reception. It therefore leads to higher SFER in the latter part of A-MPDU than that in the beginning part when the channel condition substantially changes during the A-MPDU reception.

We analyze the wireless channel dynamics considering mobility in IEEE 802.11n WLAN through extensive measurements. From this, we reveal the fundamental problem of existing frame aggregation schemes manifested over a wide range of mobility and IEEE 802.11n PHY features.

a. Presentation of UPI Used and New

The list of WiSHFUL UPI functions used for the implementation of this showcase are shown in **Table 4**.

UPI	Usage
`Wi-Fi.ampdu_length()`	determine A-MPDU length bound in time unit (microseconds)
`Wi-Fi.fixed_rate()`	determine fixed value of MCS for transmission rate adaptation algorithm (Minstrel) can also be turned on

Table 4 – UPI functions used in this showcase

In the control program (Table 5), inputs that exceed the permitted range are adjusted: for A-MPDU length, from 100 to 10240 us, and for MCS, from 0 to 15. For MCS selection UPI, the input '-1' means that the default rate adaptation algorithm (Minstrel) is used.

```
  […]
if length < 100:
    length = 100

elif length > 10240:
    length = 10240

if rate > 16:
    rate = 15
elif rate < -2:
    rate = -1

res_l = device.radio.ampdu_length(phy_dev, str(length))
res_r = device.radio.fixed_rate(phy_dev, str(rate))
[…]
```

Table 5 Control program

b. Results

This subsection shows the measurement results with changing A-MPDU length and MCS in both static and mobile environments. We set the speed of a device to 1 m/s, which is similar with walking speed.

Figure 21 summarizes the average throughput and SFER for varying aggregation time bound which determines the aggregation length of A-MPDU. The aggregation time of 0 μs represents the transmission of a single MPDU without aggregation. As the length of A-MPDU increases, the throughput of the static scenario (0 m/s) increases due to the overhead reduction. However, for an average speed of 1 m/s, the maximum throughput is achieved at 2,048 μs aggregation time bound. When the aggregation time bound is larger than 2,048 μs, the increased SFER induced by the mobility overwhelms the gain from the overhead reduction. Accordingly, the throughput decreases as the aggregation time bound increases.

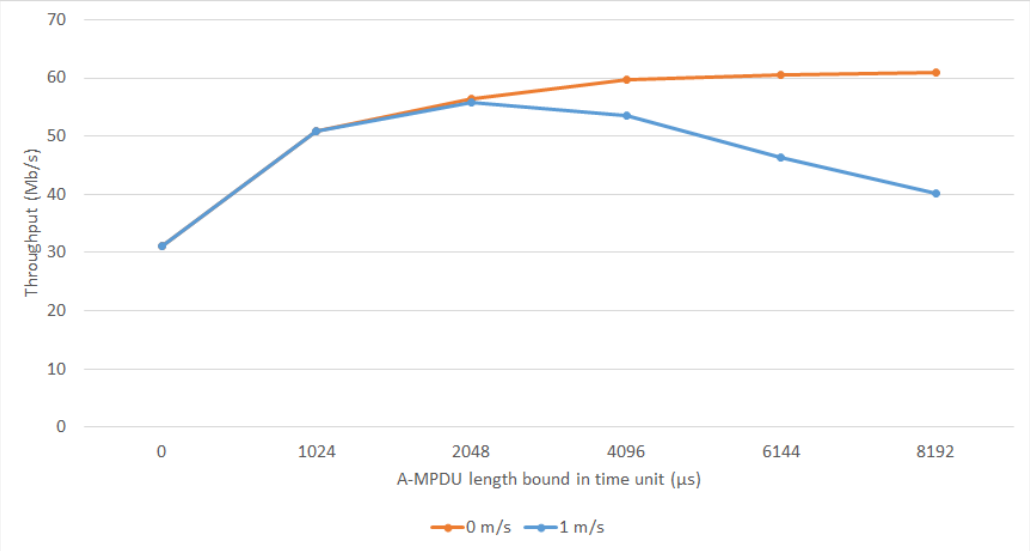


Figure 21 Throughput with different time bounds.

MCS is a critical factor determining receiver performance. Although higher order MCS achieves higher data rate, it is vulnerable not only to the channel degradation but also to mobility, since constellation points are closer from each other and coding gain is generally smaller. To verify this, we conduct experiments by changing MCSs for a given mobility. Figure 22 shows the SFER depending on the subframe location. When a station holds its position and does not move (0 m/s), the SFER remains almost zero in all subframe locations regardless of the employed MCS, because the channel quality between AP and the station is considerably good. However, when the station moves at an average speed of 1 m/s, MCS 4 and MCS 7 employing both amplitude and phase modulation (i.e., 16-QAM and 64-QAM, respectively) show higher SFER in the latter part of A-MPDU, while MCS 0 and MCS 2 which use only phase modulation (i.e., BPSK and QPSK, respectively) achieve stable SFER across the entire subframe locations. That is, MCSs which use amplitude modulation are highly susceptible to mobility. An A-MPDU using high order MCS not only uses longer A-MPDU length for a given A-MPDU duration, but also suffers from higher SFER at the latter part of A-MPDU, especially, when the channel varies within the transmission time of A-MPDU. Therefore, the length of A-MPDU with high order MCS should be carefully determined.

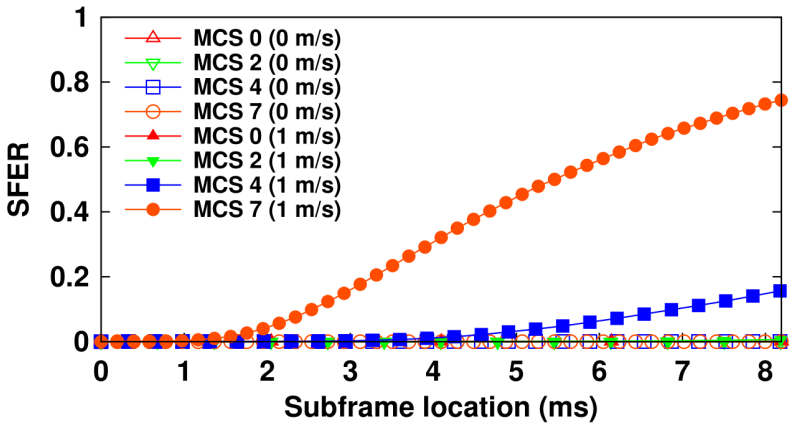


Figure 22 SFER for different MCSs.

c. Next Steps

The next step includes the implementation of UPIs with additional features such as extracting the A-MPDU transmission results from BlockAck and enabling/disabling the mobility-aware algorithm which adapts PHY rate and frame aggregation length in real-time.

2.6.2 Link estimator selection

Link estimators are extremely important in multi-hop wireless networks for obtaining a good network performance because they drive the decisions made by the routing protocol. Many estimators exist but the quality of their estimation depends on the scenario at hand. In this showcase, the impact of the estimator on the network performance for different networking scenarios is investigated. For this purpose, several link estimation algorithms were added to the RPL implementation of Contiki. The following subsection gives a brief overview of the evaluated link estimators.

a. Evaluated link estimators

Link estimation algorithms take the lossy nature and time varying link quality of WSNs into account when determining the best neighbour to forward data to. As discussed in [13], link estimators can be classified in hardware- and software-based algorithms. The hardware based link estimators use quality indicators (i.e. RSSI and LQI) set by the radio driver after packet reception. Software based link estimators such as Four bit [14] and Fuzzy LQE [15], combine information from multiple network layers into a single metric. There are also very simple link estimators, such as the objective function 0 [17], that select neighbours based on hop count. In this section a classification of the evaluated link estimators is given.

Hardware based link estimation algorithms use information provided by the radio (e.g. Received Signal Strength Indicator (RSSI) and Link Quality Indicator (LQI)) to calculate the link metric. Both link layer variables are highly correlated with the Packet Delivery Ratio (PDR) [16] and can be used to model link quality. Due to the unstable nature of wireless communication, the raw values can be aggregated with an Exponential Weighted Moving Average (EWMA) filter [13] to improve the estimation. Because LQI and RSSI are only calculated after packet reception, packet-loss is not taken into account. For this reason, the link quality of less reliable links can be overestimated.

Software based link estimation algorithms The objective function 0 [17], also referred to as hop count, is a very simple software based link estimator that minimizes the number of hops from the source towards the destination resulting in a route where the preferred parent is the furthest reachable node in the direction of the sink. If the quality of the link with this node is poor, a lot of retransmissions and extra packet loss or energy consumption can occur. On the other hand, in stable networks the overhead is limited for maintaining link information. More complex software based link estimation algorithms will tackle the aforementioned issue by using historical and/or cross-layer information to make more intelligent decisions when selecting the best links. ETX based algorithms [18] estimate the number of expected transmissions needed to successfully send a packet to each destination by counting the number of attempts needed in previous transmissions and select the path with minimal ETX. The Four bit algorithm [14] combines information from multiple OSI-layers to calculate link metrics. It uses LQI and RSSI values to account link quality and combines this with the ETX path metric. Fuzzy LQE [15] combines multiple link metrics in a less deterministic way by using membership functions which assign a score in [0, 1] for every link metric. These individual scores are combined with an aggregation function such as the Yagger operator. Multiple metrics are aggregated by calculating a weighted mean (β with a typical value of 0.6) of the worst and average value of each metric. This results in a link score which can be inverted to obtain a link metric for routing purposes.

In the experiment presented here, six different link estimators are used:

- Hardware based link estimators
  - RSSI
  - LQI
- Software based link estimators:
  - Objective Function 0
  - ETX
  - Four bit
  - Fuzzy LQE

b. Presentation of UPI Used and New

The main contribution of this showcase is the ability to change the link estimator at run-time. It has been exposed as a UPI_N parameter for Contiki nodes and can easily be generalized for multi-hop wireless routing protocols designed for other physical layer technologies.

Beside the ability to change the link estimator, the following tasks, each with added WiSHFUL UPI functionality, were required while evaluating this showcase:

- Monitor network performance and topology.
  - Gather knowledge about the topology by observing neighbor table updates.
  - Measure network performance by collecting statistics in the IP and RPL layer.
  - Measure energy consumption by collecting the radio on time from all nodes.
- Configure experiment scenario.
  - (De-)Increase the node density by (de-)activating applications on certain nodes.
  - Modify the send interval boundaries for traffic generating nodes.

Table 6 and Table 7 list, respectively, the UPI functions and attributes used in this showcase.

Table 6 UPI functions used during the link estimator selection showcase.

Function	UPI	Usage
get_neighbor_table	N	Monitor node density by collecting neighbor table information from all nodes.
start_application	N	Start application on a particular node to increase node density.
stop_application	N	Stop application on a particular node to decrease node density.
set_parameters	N	Used to update the link estimator in RPL.
get_measurements_periodic	R/N	Retrieve the radio-on-time, IP and RPL statistics in a periodic manner.
subscribe_events	N	Collect RX/TX events from the application layer.

Table 7 UPI attributes used during the link estimator selection showcase.

Attribute	UPI	Type	Description
RADIO_ON_TIME	R	Measurement	Cumulative measurement indicating the amount of time the radio was on since booting.

RPL_OBJECTIVE_FUNCTION	N	Parameter	Objective function used by RPL to estimate the link quality and calculate path cost.
RPL_STATS	N	Measurement	Cumulative statistics specific to RPL.
IP_STATS	N	Measurement	Cumulative IP layer statistics.
APP_PER_PACKET_RX_STATS	N	Event	Event triggered each time a packet is received.
APP_PER_PACKET_TX_STATS	N	Event	Event triggered each time a packet is transmitted.
APP_SEND_INTERVAL_MIN_MAX	N	Parameter	Allows to define the boundaries for the application send interval between a minimum and a maximum.

c. Results

The results were obtained using the Cooja simulator. First, a standard scenario was defined in which we tested all link estimators to have a baseline for comparing results obtained in other scenario's. During the experiment, the following performance metrics were gathered from the Contiki IPv6 routing layer via the UPI_N interface:

- Generic UPI_N metrics
  - Packet delivery ratio (PDR) at the destination (sink) node.
  - Average radio-on time (ROT) over all nodes in the network.
  - Average number of hops (NH) for each route in the network.
  - Retransmission ratio (RR) at the source node.
- RPL specific UPI_N metrics
  - Number of RPL parent switches (NRPS). This gives a good indication about the stability of the network topology, i.e. if a lot of parent switches occur, the network topology is not very stable. The graphs below show the average number of parent switches per node per minute.
  - Number of RPL DIO messages sent (NRDMS). This gives a good indication about the number of messages that are required to inform nodes about topology changes. This statistic is strongly correlated by the number of RPL parent switches. The graphs below show the average number of DIO messages per node per minute.

Beside the standard scenario, also a dense (more nodes per m2) and a sparse (less nodes per m2) scenarios were used to evaluate different link estimators. The following subsection lists the results obtained for each of the collected metrics in a) the standard scenario; b) the dense scenario; and c) the sparse scenario. The results in (b) and (c) are compared with (a). An overall conclusion is given in the last subsection.

Standard scenario

The standard scenario, illustrated in Figure 23, reflects a typical scenario for wireless sensor networks in which a considerable number of nodes report their sensor values periodically to a so-called sink node, collecting all sensor information and exposing it to the outside world. The send interval boundaries are 20 secs and 40 secs, and for each period, a new transmission delay is chosen within this interval. The scenario consists of 64 sensor nodes in an eight by eight grid. The position of the

sink (at a corner of the grid) and the size of the grid (175m by 175m) was chosen to result in a multi-hop topology. The communication range and interference range settings in Cooja were configured to respectively 45m and 60m. Note that the same experiment can be easily repeated with different topologies and/or sink locations.

Figure 24 shows the results for each of the aforementioned metrics gathered while executing the standard scenario. It can be seen that there is already a clear trade-off between network reliability (chart (a): Packet Delivery Ratio, PDR) and energy consumption (chart (b): Radio On-Time) when choosing of0 or ETX over the more complex Four bit or Fuzzy LQE link estimators. The more complex link estimators introduce more RPL overhead, i.e. require more parent switches (chart (e)) and DIO messages (chart (f)). The average number of hops (chart (c)) and MAC retransmissions per packet (chart (d)) are comparable for all link estimators except for the RSSI based estimator which has a higher hop count and requires more RPL overhead, resulting in a higher energy consumption.

Examining the results a bit closer, it can be seen that while the Four-bit estimator has a higher number of RPL parent switches compared to the RSSI estimator, it has a lower increase in sent DIO messages. This is because the RPL switches are spread more in time for the RSSI estimator. For Four-bit algorithm, most RPL parent switches occurred at the beginning of the simulation. Due to the trickle timer mechanism in RPL, having a lot of parent switches in a short time span only introduced a limited DIO message overhead.

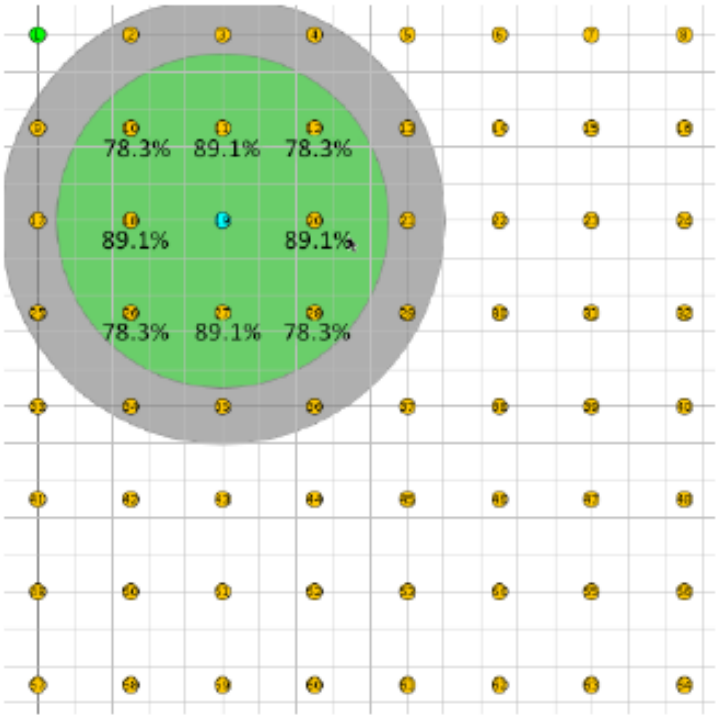
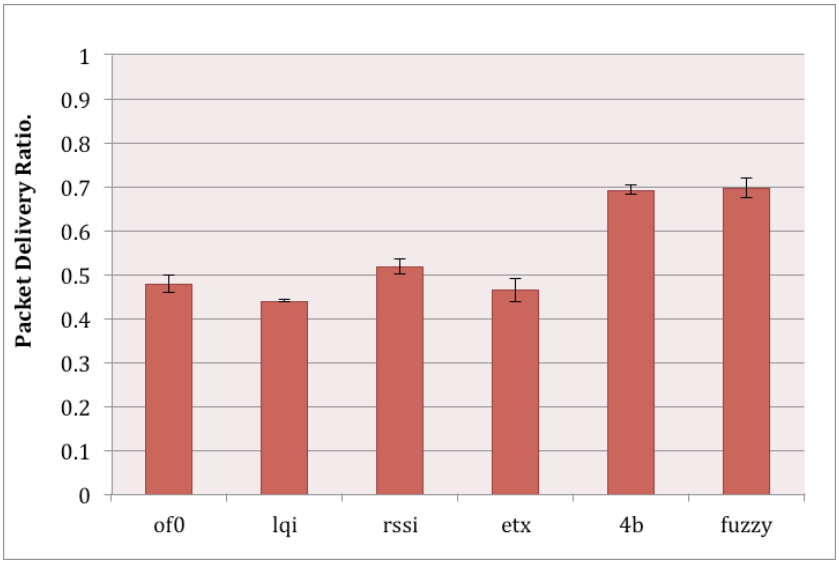
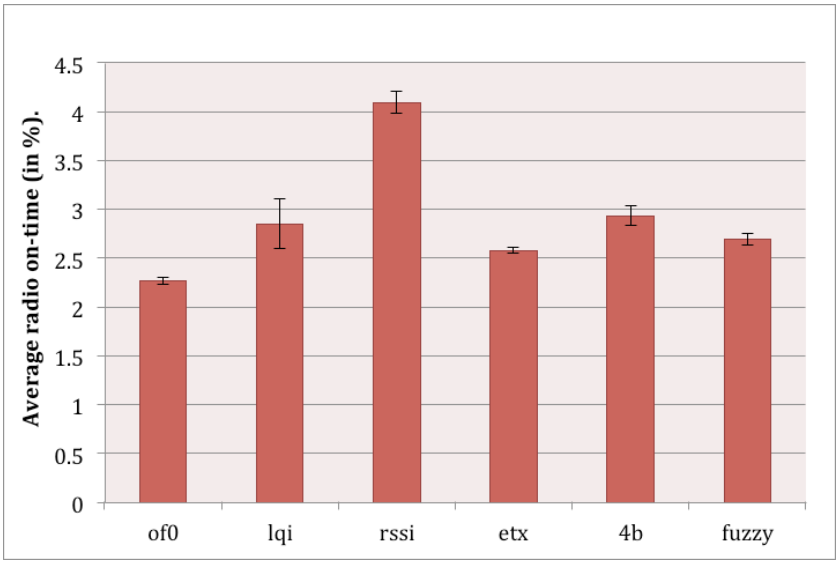


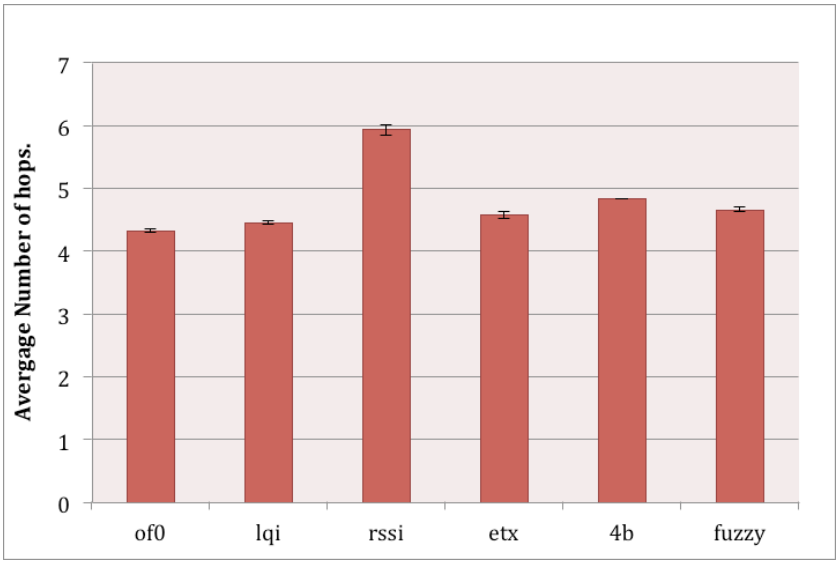
Figure 23 illustrates the standard WSN scenario. An 8x8 grid over an area of 175x175m2. The sink is located on the left corner of the grid. The green circle around a node indicates its communication range, the grey circle its interference range. The other nodes in the green circle have a certain percentage of successful reception rate, defined by the distance according to the default UDGM model used in Cooja.



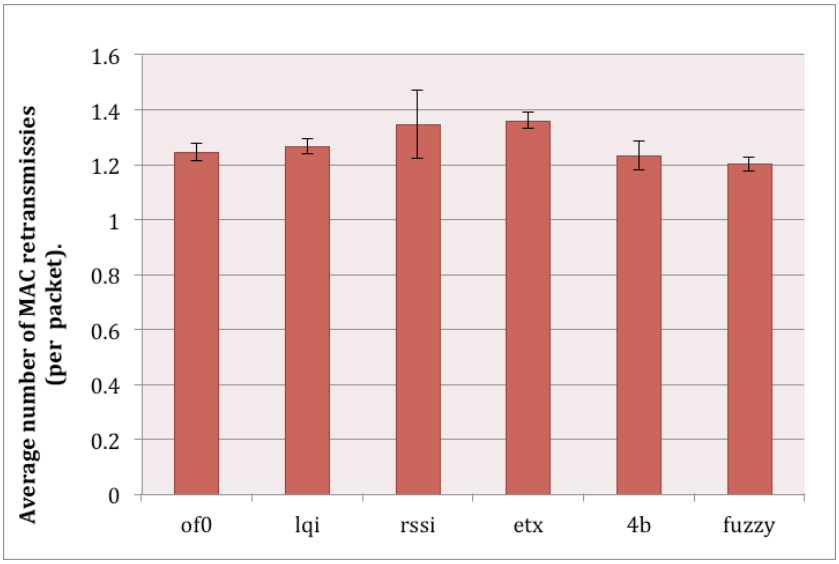
(a) Packet Delivery Ratio



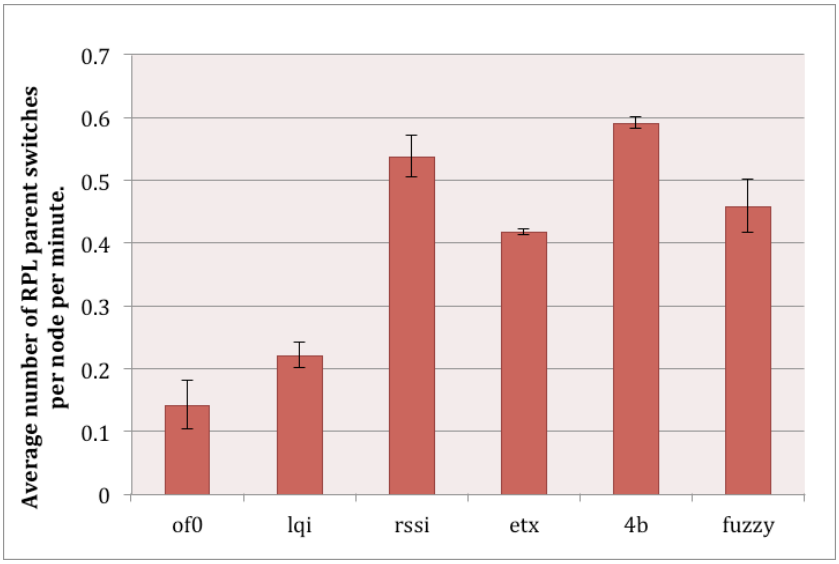
(b) Radio on-time



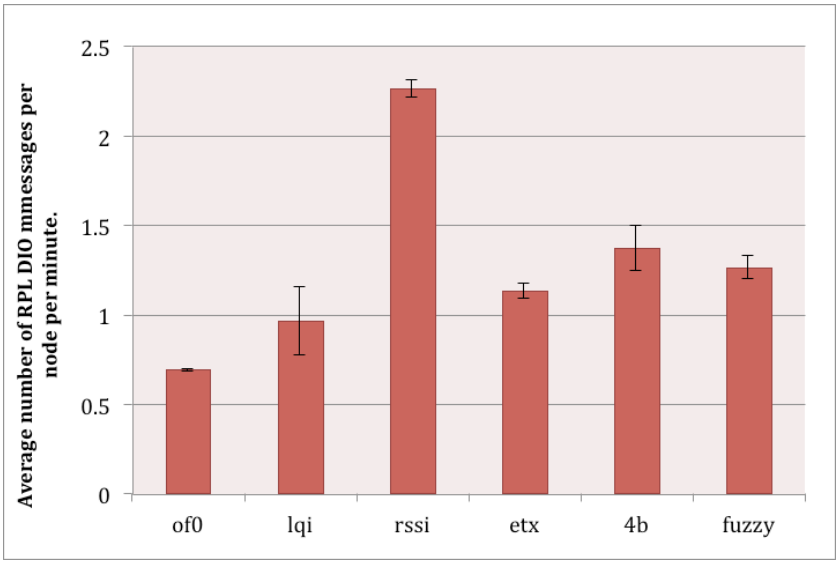
(c) Number of hops



(d) Number of retransmissions



(e) RPL parent switches



(f) Number of DIO messages

Figure 24 Results gathered while executing the standard scenario for different link estimators.

Dense scenario

The dense scenario consists of 144 sensor nodes in a twelve by twelve grid, in the same 175x175 m2 area. The node density is clearly higher because there are more nodes in communication range (green circle) and interference range (grey circle) compared to the standard scenario, as illustrated in Figure 25. The communication range (45m) and interference range (60m) settings are identical compared to the standard scenario. The position of the sink is again chosen in a corner of the grid (upper left). Also the same send interval boundaries (20 secs and 40 secs) are applied.

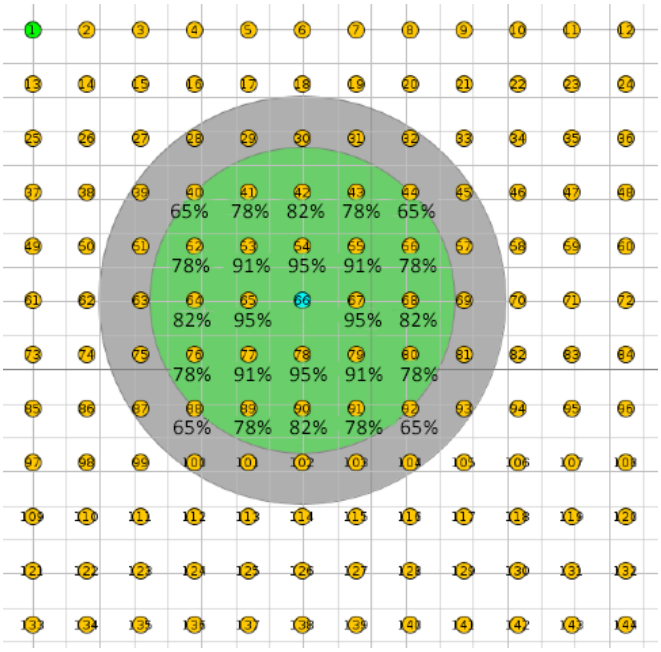
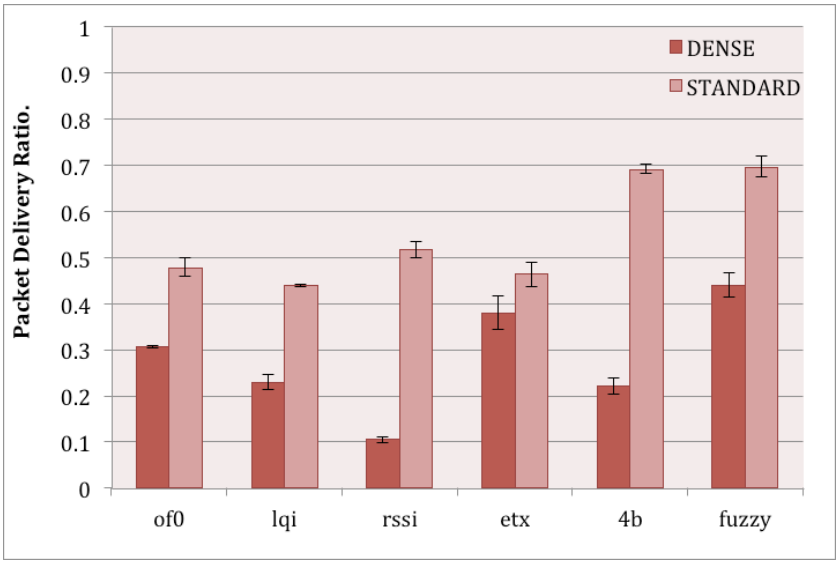


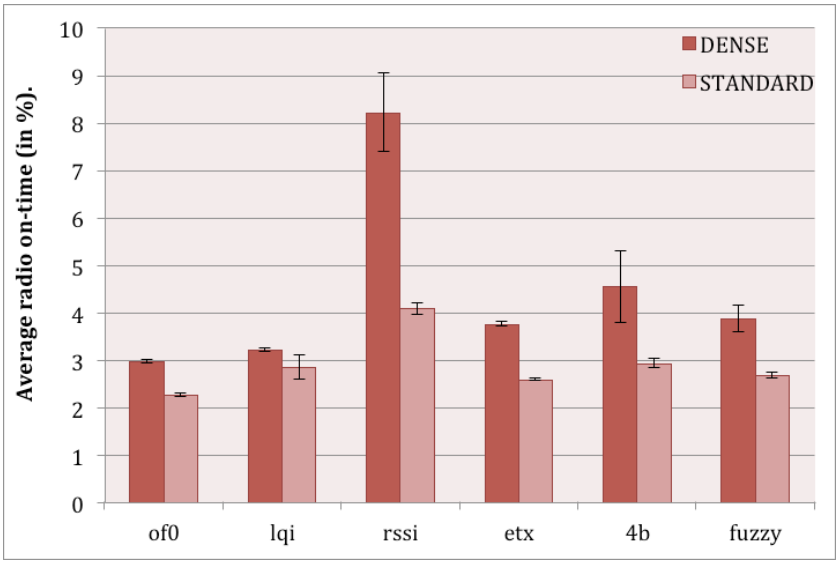
Figure 25 illustrates the dense WSN scenario. An 12x12 grid over an area of 175x175m2. The sink is again located on the upper left corner of the grid. The node density (e.g. number of nodes in communication range (green circle) and interference range (grey circle) is clearly higher in this scenario.

The results shown in Figure 26 compare the performance for different link estimators in the standard scenario vs. the dense scenario. Because there are much more nodes in the same area, there are many more possible paths to the sink. Also internal interference will be higher. Indeed, the results clearly show that the overall network performance is lower for all link estimators in the dense scenario due to the higher interference. The relative difference between the link estimators is also higher in the dense scenario vs. the standard scenario as expected.

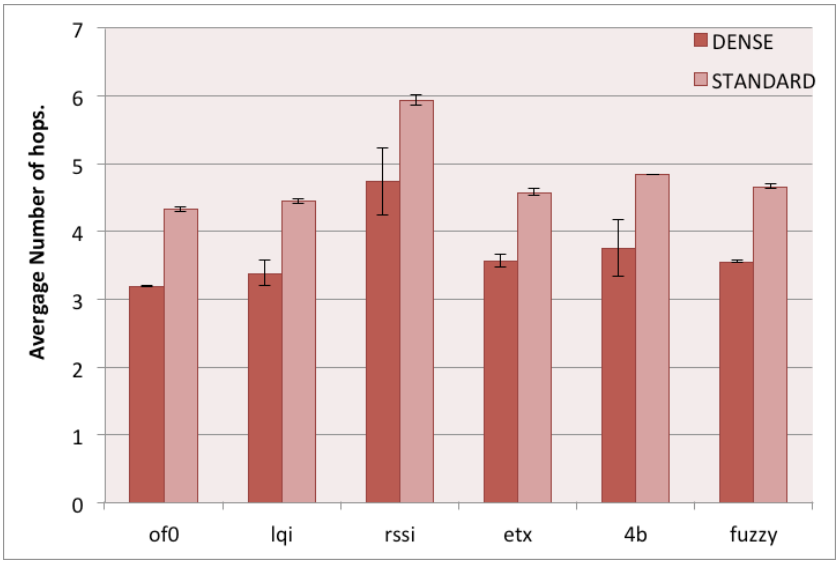
The packet delivery ratio drops (chart (a)) significantly for all link estimators, especially for the Four-bit and RSSI based estimators which is also reflected in the higher number of per packet MAC retransmission (chart (d)). This explains the higher radio on-time (chart (b)), and consequently energy consumption. Fuzzy LQE and ETX show the highest PDR for a limited increase in radio on-time. The RPL overhead w.r.t. parent switching is comparable in the dense scenario vs. the standard scenario, except for the Four-bit and RSSI based estimators (as was the case in standard scenario). Here we again see a substantial increase in RPL parent switches (chart (e)). This does not yield more DIO messages (char (f)) however, indicating that the parent switches occurred in the beginning of the simulation.



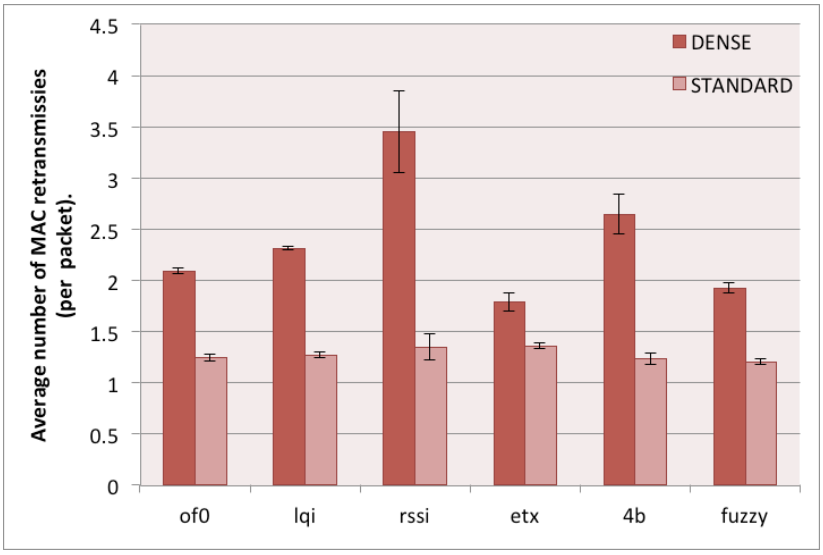
(a) Packet Delivery Ratio



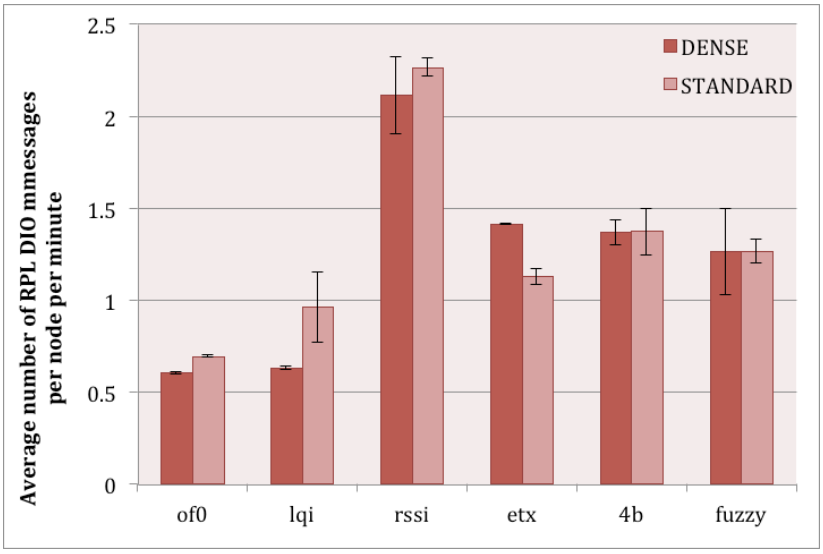
(b) Radio on-time



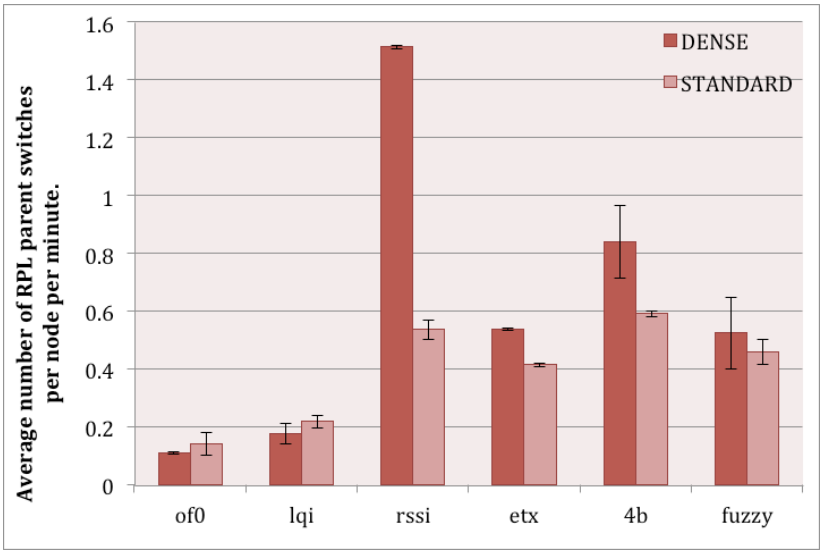
(c) Number of hops



(d) Number of retransmissions



(e) RPL parent switches



(f) Number of DIO messages

Figure 26 Comparison of results between the standard and dense scenario for different link estimators.

Sparse scenario

In the sparse scenario the standard 8x8 grid in an area of 175x175 m2 was used. The node density was reduced by lowering the communication range and interference range settings to 30m and 45m respectively. As illustrated in Figure 27, the number of nodes in comm. and interference range are clearly lower. The same sink position and send interval boundaries were applied.

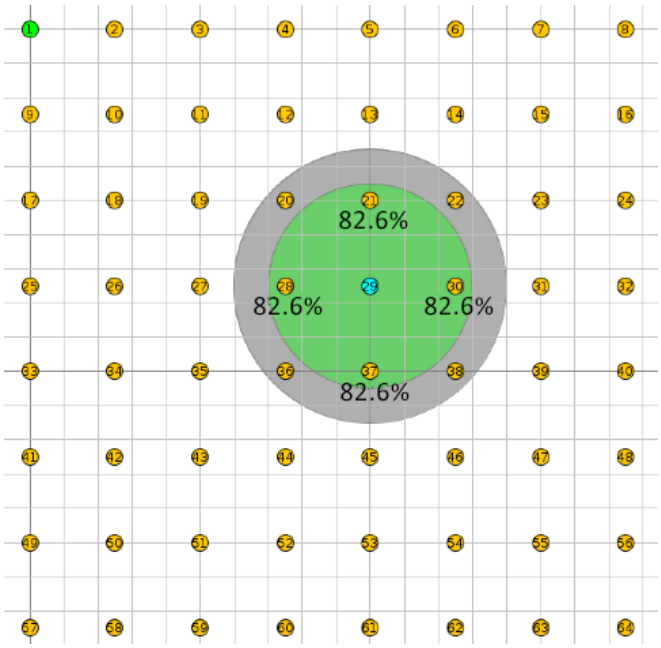
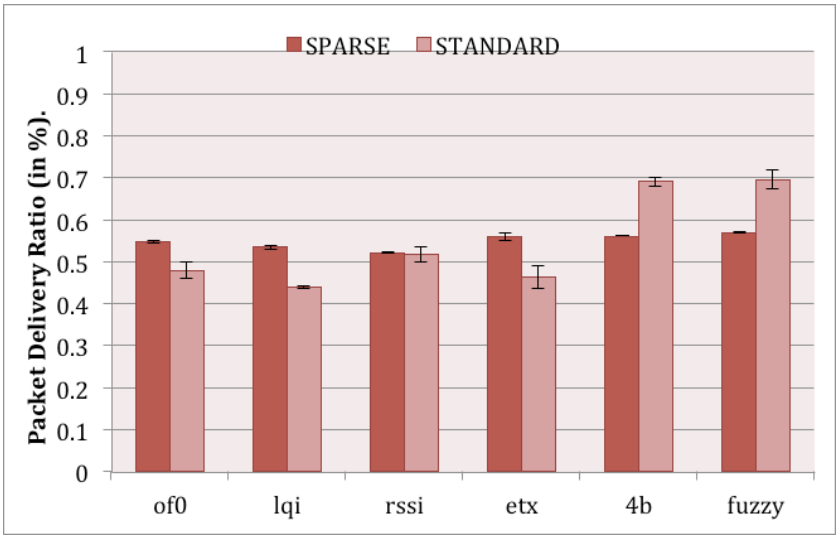


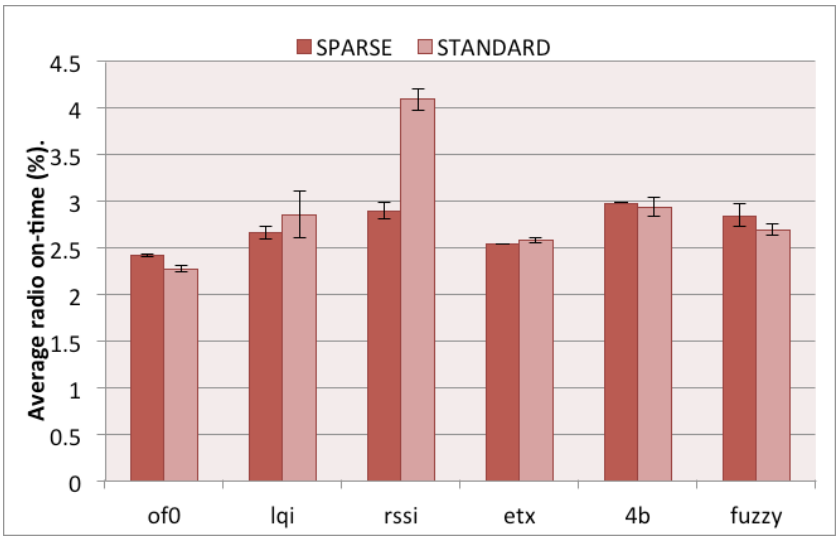
Figure 27 illustrates the sparse WSN scenario. The standard 8x8 grid over an area of 175x175m2 is used but the communication (green circle) and interference ranges (grey circle) are lowered to decrease the node density.

The results shown in Figure 28 compare the performance for different link estimators in the standard scenario vs. the sparse scenario. Because there are much less nodes in the same area, there are also less possible paths to the sink. This results in a higher number of hops required to reach the sink as clearly can be seen in chart (d). The internal interference should however be lower. Because of this, the choice of the link estimator should make less difference. Indeed, the results clearly show that the overall network performance is very similar for all link estimators in the sparse scenario. The relative difference between the link estimators is also much lower in the sparse scenario vs. the standard scenario and the dense scenario, as expected.

The packet delivery ratio (PDR, chart (a)) is nearly equal for all link estimators. The radio on-time (chart (b)) is slightly lower for the simpler objective function 0 or the ETX link estimators compared to the more complex Four-bit and Fuzzy LQE. This can be related to the higher RPL overhead. Significantly more RPL parent switches (chart (e)) are required before the network reaches a stable state. Again, because most parent switches occur in the beginning of the simulation, the effect on the DIO messages (chart (f)) is less substantial.

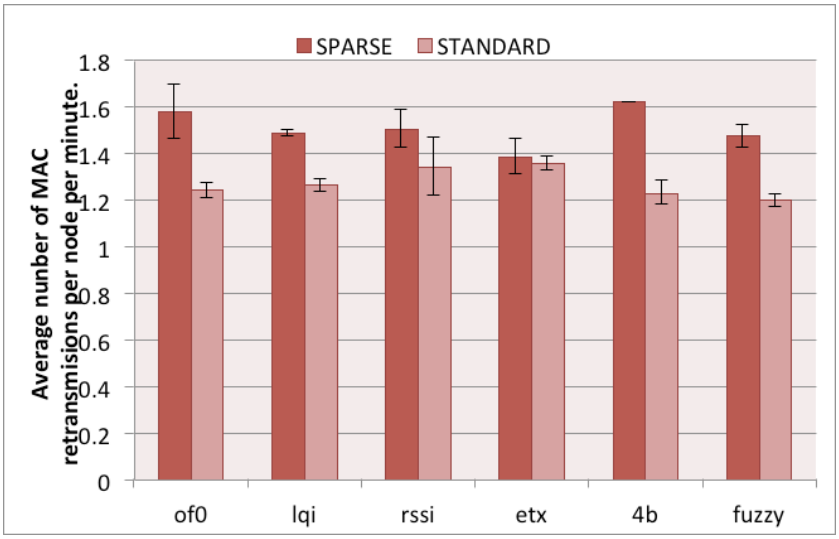


(a) Packet Delivery Ratio

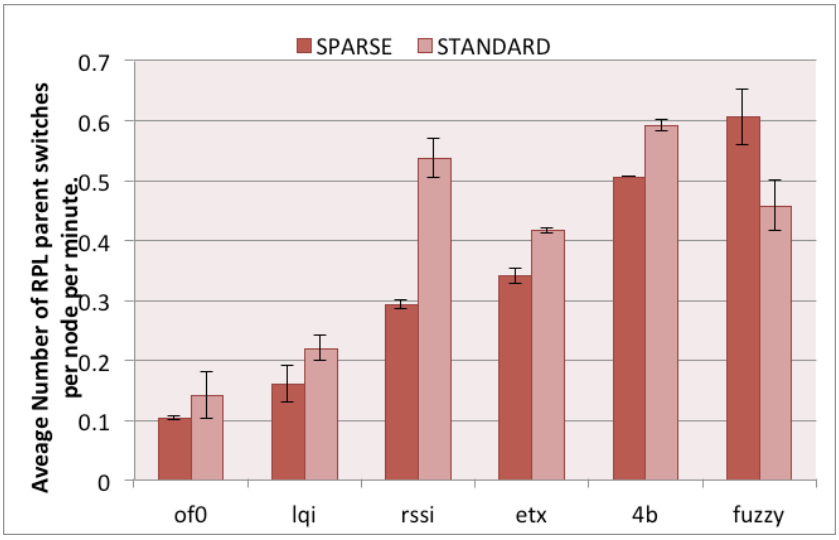


(b) Radio on-time

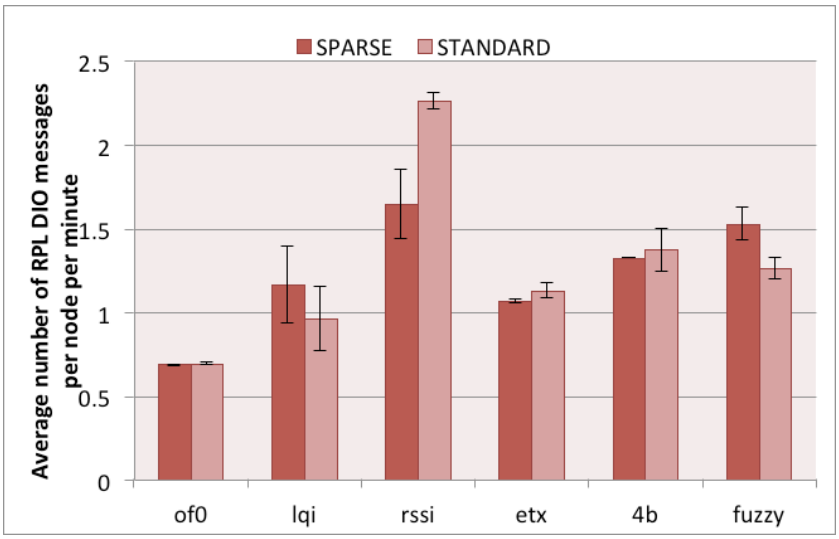
(c) Number of hops



(d) Number of retransmissions



(e) RPL parent switches



(f) Number of DIO messages

Figure 28 Comparison of results between the standard and the sparse scenario for different link estimators.

Overall conclusion

Overall, we can see that the more complex link estimators yield a higher PDR. This requires however more RPL overhead and, consequently, a higher energy consumption. Table 8 ranks the different link estimators based on packet delivery ratio (PDR) and radio on-time (ROT) for the three scenarios. Note that this only gives an indication which link estimator gives the best trade-off between PDR and ROT because the relative differences between the different ranks is not visible anymore. In the standard scenario and dense scenario, Fuzzy LQE could be chosen as best trade-off if energy consumption is slightly less important, otherwise ETX is a better choice. In the sparse scenario, more complex estimators fail to produce significantly better PDR, hence the objective function) or ETX are the most appropriate choices.

Table 8 ranks the different link estimators based on packet delivery ratio (PDR) and radio on-time (ROT) for the three scenarios.

Scenario	Metric	1	2	3	4	5	6
Standard	PDR	Fuzzy LQE	Four-bit	RSSI	OF0	ETX	LQI
	ROT	OF0	ETX	Fuzzy LQE	LQI	Four-bit	RSSI
Dense	PDR	Fuzzy LQE	ETX	OF0	Four-bit	LQI	RSSI
	ROT	OF0	LQI	ETX	Fuzzy LQE	Four-bit	RSSI
Sparse	PDR	Fuzzy LQE	ETX	Four-bit	OF0	LQI	RSSI
	ROT	OF0	ETX	LQI	Fuzzy LQE	RSSI	Four-bit

d. Next Steps

Numerous other aspects could be added in this showcase, enabling us to investigate the impact while changing them and allowing to make more founded conclusions w.r.t. the most optimal link estimator. The following non-exhaustive list gives some potential aspects that should have an impact on the link estimator and that could be added to this showcase.

- Traffic pattern (periodic, asymmetrical, random bursts).
- MAC protocol (TDMA vs. CSMA).
- MAC duty cycle (e.g. number of slots or channel check rate).
- MAX number of MAC retransmissions.
- External interference.
- Node mobility.

Thanks to the advances made in year 1 and 2 of the project, such parameters could also be added. To make this eve more useful for the researchers, a benchmarking toolbox should be added on top of the UPIs that allows the experimenter to execute control programs with different parameter settings.

2.6.3 Multihop load aware MAC adaptations

In recent years, it has clearly emerged that Wi-Fi network performance can dramatically degrade in multi-hop connected topologies and in high-density node scenarios. These conditions are likely to occur when multiple networks coexist or large access infrastructure is deployed. The main reasons for the degradation include the starvation and unfairness phenomena of CSMA-based protocols due to a mismatch in the local views of the wireless medium among the nodes, and due to the high level of contention when the network is heavily congested.

In this showcase, we propose a solution for mitigating the performance impairments of CSMA/CA protocols in multi-hop topologies based on the dynamic adaptation of the contention process experienced by nodes in the wireless network. A distributed protocol, called REACT, is used to negotiate the channel airtime for a node as a function of the traffic requirements of its neighborhood, taking into account bandwidth reserved for the control operations. A mechanism is provided for a node to tune its contention window depending on its allocated airtime. Different from previous schemes, a node's contention window is *fixed in size* unless the traffic requirements of its neighborhood change. The scheme is implemented on *legacy* commercial 802.11 devices exposing WiSHFUL UPIs.

REACT protocol: Channel allocation in wireless networks can be viewed as a resource allocation problem where transmitters correspond to demands and receivers to resources. The general idea on the basis of REACT has been proposed in Figure 29 or slotted systems, for negotiating an allocation of channel airtimes by means of an auction mechanism. Each node runs an auctioneer that maintains an offer, the maximum airtime consumed by any adjacent bidder. Similarly, each node also runs a bidder that maintains a claim, the airtime the bidder intends to consume at adjacent auctions. Through updates of offers and claims, the auctioneers and bidders converge on an allocation of airtimes. The details of the protocol are described in .**Figure 30**. Consider a network with N nodes. Starting from the channel airtime demand (w_1, w_2, … w_N) performed by each node of the network according to the running applications, at the end of the auction mechanism the REACT protocol assigns the normalized airtimes (s_1, s_2, … s_N) that each node is allowed to consume. In order to limit the channel airtime to the assigned value, we designed a scheme for dynamically tuning the contention window as a function of the transmission grants observed in a given monitoring interval, channel busy times and collision rate. Figure 29 shows an example of resource allocation performed by REACT in a network topology with 6 nodes and a specific traffic demand. The main idea of REACT is equally sharing the available channel time among conflicting nodes (which include 2-hops nodes consuming channel time for the desired receiver node) until the traffic demand is satisfied, and further dividing the excess capacity to the nodes for which the demand is not satisfied.

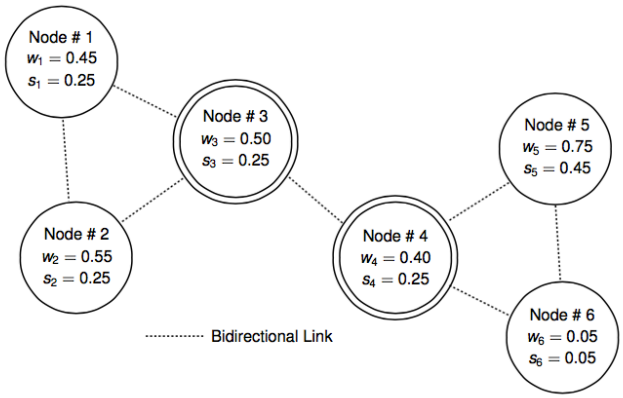


Figure 29 - Example of REACT resource allocation

Tuning of the contention window. In our showcase, we implemented a mechanism for tuning the contention window of each node in order to achieve a desired airtime allocation. The rationale of the tuning mechanism is depicted in Figure 30, which represents a sequence of channel accesses performed by a reference node i in an observation interval C. The airtime of node i is colored blue. It includes RTS, CTS, DATA, and ACK transmissions, as well as the inter-frame spaces between the frames sent in the same channel access (also called transmission opportunity). The observation interval C is divided into sub-intervals c(1), . . . , c(k) delimited by the start of a transmission

opportunity granted to node i. Each interval c(k) includes the airtime of node i and also its backoff expiration time, which in turns depends on the initial backoff value and on the time the backoff is frozen due to the transmission of other nodes. During these intervals f(k), in which backoff is frozen (coloured in gray), the backoff counter is not decremented. Such backoff freezing can be triggered by the physical carrier sense, as well as by the virtual carrier sense mechanism enabled by the reception of CTS frames. Assume that channel accesses are managed by means of a 4- way handshake. Let tx , x in {RTS, CTS, DATA, ACK} be the time to transmit x. The airtime in an access interval depends on the outcome of RTS and DATA transmissions. It can vary from a minimum of tRTS + DIFS when the RTS fails, to a maximum of tRTS + SIFS + tCTS + SIFS + tDATA + SIF S + tACK + DIFS when the transmission is successful.

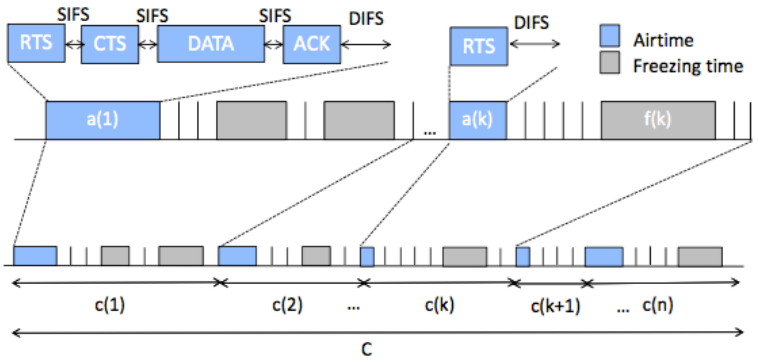


Figure 30 - successive channel transmission for a tagged node-i

From renewal theory, the portion of channel resources allocated to node i may be expressed as:

$$s_i = \frac{E[a]}{E[c]} = \frac{E[a]}{E[a] + E[f] + E[W]/2 \cdot \sigma}$$

where E[a] is the average airtime and E[c] is the average channel access interval. This interval can be expressed as a sum of E[a], the average time the backoff is frozen E[f], and the average initial backoff value E[W] in slots multiplied by σ, the length in time of a backoff slot.

The average airtime can be computed by considering the probability p_{RTS} of a successful RTS transmission, and the probability p_{DATA} of a successful DATA transmission as tRTS + p_{RTS} ·(tCTS +tDATA +2·SIFS) + p_{DATA}·tACK+DIFS. In general, after a successful handshake, it is assumed that data packets are protected by the virtual carrier sense. However, in multi-hop scenarios, it may happen that one of the receiver's neighbors experiences a collision on the reception of a CTS due to a transmission originated by a node hidden to the receiver; in turn, it may interfere with the DATA reception. The average time the backoff is frozen depends on the number of neighbors and on their traffic, while the average backoff value depends on the contention window settings. The channel access probability depends on the average contention window value rather than on the specific backoff algorithm. Let Wi be the contention window value of node i configured during an observation interval C.

Legacy Wi-Fi nodes can estimate the current allocated rate s_i as a function of the total number of channel accesses n, the total time with the backoff frozen F , and total airtime A observed in C:

$$s_i = \frac{\sum_{k=1}^{n} a(k)}{\sum_{k=1}^{n} a(k) + \sum_{k=1}^{n} f(k) + W_i/2 \cdot \sigma \cdot n} = \frac{A}{A + F + W_i/2 \cdot \sigma \cdot n} = \frac{A}{C}$$

where $W_i/2 \cdot \sigma \cdot n$ is an approximation of the total time required for the backoff countdown.

Let W_i^+ be a new setting of the contention window, and let $E[a]^+$ and F^+, respectively, be the prediction of the average airtime and total time backoff is frozen experienced under the new setting of the contention window in a new channel observation interval equal to C^+. If the other nodes used

fixed contention windows, then the dynamic tuning of W_i^+ does not affect the collision probability experienced by the target node. Indeed, the collision probability is given by the probability that at least one interfering node is transmitting, given that the target is also transmitting. It follows that the new Wi+ value has an impact on the channel access probability of station i but not on the average airtime in each channel access (i.e., $E[a]^+ = A/n$), which only depends on the collision probability experienced by RTS and DATA frames. Similarly, we can assume that the fraction of channel time not allocated to node i that is sensed as busy is not affected by the contention window tuning and therefore equals to the previously experienced one, i.e., to F/(C-A) . The total time the backoff is frozen F^+ is obtained as the product between this ratio and the total time not spent in transmission during the interval C^+.

To obtain the desired rate s_i^* in the next tuning interval C^+, it is required to achieve a number of channel accesses equal to $n^* = s_i^* \cdot C^+/E[a]^+ = s_i^* \cdot C^+ \cdot n/A$. Therefore, the total time spent for n^* countdowns of the backoff has to equalize the difference between the channel time not allocated to node i, i.e., $C^+(1 - s_i^*)$, and the time the backoff is frozen. It follows that the contention window can be tuned as:

$$W_i^* = \frac{2}{\sigma} \cdot \frac{A(1 - s_i^*)}{n \cdot s_i^*} \cdot \left(1 - \frac{F}{C - A}\right)$$

The above algorithms for monitoring the channel busy times and allocated airtime and for tuning the contention window as a function of the desired rate can be easily implemented in the WiSHFUL framework on top of the radio UPI, as described in what follows.

a. Presentation of used UPIs and the control program

This section presents a detailed description of the software architecture used for implementing REACT in the WiSHFUL framework, by also specifying the UPIs utilized in the showcase, the experiment setup and the experimental results. The architecture is summarized in Figure 31. The main functional requirements for implementing REACT are: i) enabling the injection of custom frames from user-space for sending the messages used by the auction protocol, ii) interacting with driver-level statistics for measuring channel busy times and airtimes; iii) configuring the node contention parameters (namely, the contention window) dynamically. The overall mechanism is implemented by defining a REACT control program, which includes: a) the REACT protocol logic for managing claims and offers, and b) the dynamic tuning of the Contention Window value as a function of the observed channel parameters. The interfaces between the user-space control program and the kernel-space primitives is natively provided by the WiSHFUL UPIs.

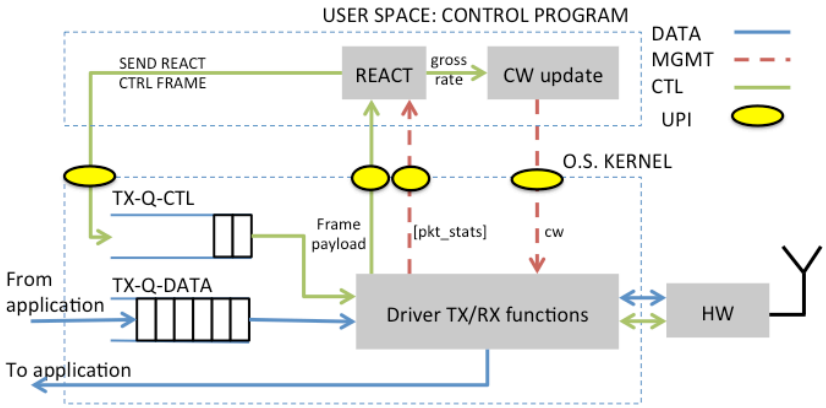


Figure 31 - REACT 802.11 Architecture

The following table (Table 9) summarizes the UPIs used by the REACT control program:

UPI	Usage
`net.inject_frame()`	send custom broadcast frames containing REACT claims and offers
`net.sniff_layer2_traffic()`	receive frames and dissect REACT control messages.
`set_mac_access_parameters()`	set CWmin=CWmax=CW for the data queue, enforcing the desired CW value
`radio.get_measurements()`	get packet transmission statistics: *data_count* and *rts_count* are used to estimate the current freezing time and predict the future one.

Table 9 - UPI list

On each Wi-Fi node, a local control program named **react()** is activated by the global controller (Table 12) with the following input:

```
{"iface":"wlan0","i_time":1,"bw_req":bw_req[i_flow],enable_react":True}
```

where *i_time* is the observation interval to be used for updating the statistics and the contention window value, *bw_req* represents the amount of traffic desired by the node expressed in Kb/s. In order to increase the modularity of the control program, two configuration files are introduced: node_info.csv (**Table 10**) and experiment_info.csv (Table 13). These files contain a tabular description of the nodes and experiment setup, respectively. The global control program inspects the configuration files, setup ad-hoc connections between nodes, activates the REACT algorithm on each node and starts the source traffic by means of iperf applications.

```
#hostname,driver,eth0_ip,freq,txpower,wlan0_ip
nodezotacb3,ath9k,10.11.16.22,5180,1,192.168.0.1
nodezotacb4,ath9k,10.11.16.33,5180,1,192.168.0.2
nodezotacd3,ath9k,10.11.16.24,5180,3,192.168.0.3
nodezotaci3,ath9k,10.11.16.29,5180,3,192.168.0.4
nodezotack3,ath9k,10.11.16.31,5180,1,192.168.0.5
nodezotack4,ath9k,10.11.16.42,5180,1,192.168.0.6
```

Table 10 - node_info.csv

```
#src,dst,bw_req,port,t_start,t_stop
1,2,6000,5012,1,100
2,3,6000,5021,1,100
3,4,6000,5034,1,100
4,5,6000,5045,1,100
5,6,6000,5056,1,100
6,5,6000,5065,1,100
```

Table 11 - experiment_info.csv

For sake of simplicity, the following code snapshot highlights only some relevant parts of the experiment setup.

```
[…]
#SETUP NODES, RUN REACT
nodes_info_path=args['--nodes']
[hosts,driver,eth_ip,freq,tx_power,wlan_ip]=set_hosts(nodes_info_path);
experiment_info_path=args['--experiment_info']
if experiment_info_path:
 [src,dst,bw_req,port,t_start,t_stop]=experiment_setup(experiment_info_path)
for i_flow in range(0,len(src)):
   for jj in range(0, len(nodes)):
     if nodes[jj].ip == eth_ip[int(dst[i_flow])-1]:
       lcpDescriptor =
controller.node(nodes[jj]).hc.start_local_control_program(program=react)
       msg={"iface":"wlan0","i_time":1,"bw_req":bw_req[ i_flow],enable_react":True}
       lcpDescriptor.send(msg)
[…]
```

Table 12 - Global control program: the react control program is activated on each node.

The local control program (shown in Table 13) sent by remote controller to local nodes is composed by three main threads which run continuously:

- *REACT control message sender*: (send_REACT_msg) this thread collects the information required for specifying offers and claims and send REACT control messages by means of custom broadcast L2 messages. Control message are sent at regular intervals of 100ms by using the UPI function **inject_frame().** The L2 message is forged directly by the local control program. At each transmission, the state internal parameters of the node (offers/claims) are updated.
- *Control message receiver and dissector:* (sniffer_REACT) this thread receives L2 frames and dissects REACT messages. The messages are processed in bursts collected at each second, while L2 frame are received and buffered by using the UPI function **sniff_layer2_traffic().** When a new burst of control frames is processed, the local controller updates the list of state parameters (offers/claims) for each neighbor node.
- *Contention windows update:* (update_cw) this thread is based on two independent inputs: i) packet and channel statistics retrieved by means of the UPI function **get_measurement();** ii) desired rate computed by the REACT auction and stored in the internal state parameters. The desired rate is then mapped into a new contention window values as a function of collision rate and channel busy times, as described in the previous section. When a new CW is computed a call to the UPI function **set_mac_access_parameters()** is performed to enforce the new CW value.

```
def main(iface="wlan0",i_time=1,bw_req=0,enable_react=False):

    #INIT REACT INFO
    init(iface);
    try:

        #Thread transmitter
        _thread.start_new_thread( send_REACT_msg,(iface,i_time,bw_req,enable_react )
)
        #thread receiver
        _thread.start_new_thread( sniffer_REACT,(iface,i_time ) )
        #update CW
        _thread.start_new_thread(update_cw,(iface,i_time,enable_react,i_time))

    except (Exception) as err:
          print ( "exception", err)
         pass

    while 1:
       pass
```

Table 13 – REACT local control program main

```
def send_REACT_msg(iface,i_time=1,bw_req,enable_react):
    #TX
    my_mac =str(netifaces.ifaddresses(iface)[netifaces.AF_LINK][0]['addr'])
    while True:
        rate = min((float)( C ),( (bw_req*C)/float(MAX_THR)) );
        neigh_list[my_mac]['w']=rate
        try:
            pkt_to_send={};

            neigh_list[my_mac]['t']=float(time.time())
            pkt_to_send['t']=neigh_list[my_mac]['t']
            pkt_to_send['claim']=neigh_list[my_mac]['claim']
            pkt_to_send['offer']=neigh_list[my_mac]['offer']
            json_data = json.dumps(pkt_to_send)

            timeout = 30 #sec

            for key,val in neigh_list.items():
                if float(time.time())-val['t'] > timeout:
                    neigh_list.pop(key)
                    update_offer()
```

```
                update_claim()

            # REACT variables updated, transmit!
            send_ctrl_msg(iface,json_data)
            tx_interval=i_time/10
            time.sleep(tx_interval - ((time.time() - starttime) % tx_interval)
        def send_ctrl_msg(iface,json_data):
          a=scapy.all.RadioTap()/scapy.all.Dot11(addr1="ff:ff:ff:ff:ff:ff",
addr2=my_mac, addr3="ff:ff:ff:ff:ff:ff")/json_data
          mon_iface="mon0"

controller.net.inject_frame(iface=mon_iface,frame=a,is_layer_2_packet=True,tx_count
=1, pkt_interval=0)
```

Table 14 – first REACT thread: send control message

```
def sniffer_REACT(iface,i_time):

      #scapy.all.sniff(iface=mon_iface, prn=updateAction(iface,i_time),store=0)

      call_timeout=i_time/2
      call_count=2000
      while True:
#        pktlist = scapy.all.sniff(iface=mon_iface, timeout=call_timeout,
count=call_count,store=1)
         # UPI
         pktlist = controller.net.sniff_layer2_traffic(iface=mon_iface,
sniff_timeout=call_timeout, ipdst=None, ipsrc=None)
         for pkt in pktlist:
            try:
               rx_mac=str(pkt.addr2)
               if rx_mac == my_mac:
                  pass
               else:
                  payload=bytes(pkt[2])
                  if 'claim' in str(payload):
                     payload='{'+re.search(r'\{(.*)\}', str(payload) ).group(1)+'}'
                     curr_pkt=json.loads(payload)
                     neigh_list[str(rx_mac)]=curr_pkt;
                     curr_pkt['t'] = float(time.time())
                     update_offer();
                     update_claim();
            except (Exception) as err:
               if debug:
                  print ( "exception", err)
               pass
```

Table 15 – Second REACT thread: control message dissector

```
"""
update CW decision based on ieee80211 stats values and virtual channel freezing
estimation
"""
def update_cw_decision(iface,enable_react,sleep_time):
   #get stats
   global my_mac
   global cw
   global cw_
   global data_count_
   global rts_count_
   CWMIN=15
   CWMAX=2047
   UPI_myargs = {'interface' : 'wlan0', 'measurements' :
[UPI_R.dot11RTSSuccessCount,UPI_R.dot11RTSFailureCount] }
   pkt_stats=controller.radio.get_measurements(UPI_myargs)

   pkt_size=1534

   if pkt_stats:
      if rts_count_ == 0 and data_count_ == 0:
```

```
            data_count = pkt_stats['dot11RTSSuccessCount'] - data_count_
            rts_count = pkt_stats['dot11RTSSuccessCount'] +
pkt_stats['dot11RTSFailureCount'] - rts_count_
            data_count_=pkt_stats['dot11RTSSuccessCount']
            rts_count_=pkt_stats['dot11RTSSuccessCount'] +
pkt_stats['dot11RTSFailureCount']
            return
         data_count = pkt_stats['dot11RTSSuccessCount'] - data_count_
         rts_count = pkt_stats['dot11RTSSuccessCount'] +
pkt_stats['dot11RTSFailureCount'] - rts_count_
         data_count_=pkt_stats['dot11RTSSuccessCount']
         rts_count_=pkt_stats['dot11RTSSuccessCount'] +
pkt_stats['dot11RTSFailureCount']
         tx_goal=0
         I=0
         dd = sleep_time;
         gross_rate = float(CLAIM_CAPACITY)*float(neigh_list[my_mac]['claim']);
         busytx2 =  0.002071*float(data_count) + 0.000046*float(rts_count); #how much
time the station spent in tx state during the last observation internval
         SIFS=16 #usec
         tslot=9e-6 #usec
         freeze2 = float(dd) - float(busytx2) - cw_/float(2)*float(tslot)*rts_count;
#how long the backoff has been frozen;
         if rts_count > 0:
            avg_tx = float(busytx2)/float(rts_count); #average transmission time in a
transmittion cycle
            psucc = float(data_count)/float(rts_count);
         else:
            avg_tx=0
            psucc=0

         if avg_tx > 0:
            tx_goal = float(dd*gross_rate)/float(avg_tx);
         else:
            tx_goal = 0

         freeze_predict = float(freeze2)/float(dd-busytx2)*float(dd-
dd*float(gross_rate))  ;

         if tx_goal > 0:
            cw = 2/float(0.000009) * (dd-tx_goal*avg_tx-
freeze_predict)/float(tx_goal);

         if cw < CWMIN:
            cw_=CWMIN
         elif cw > CWMAX:
            cw_=CWMAX
         else:
            cw_=cw
            cw_= pow(2, round(math.log(cw_)/math.log(2)))-1;

         # ENFORCE CW
         qumId=1 #BE
         aifs=2
         cwmin=int(cw_);
         cwmax=int(cw_);
         burst=0
         if enable_react:
            setCW(iface,qumId,aifs,cwmin,cwmax,burst);
         thr=(data_count)*1470*8/float(dd*1e6);

out_val="%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f,%.4f
" %
(time.time(),dd,data_count,rts_count,busytx2,gross_rate,avg_tx,freeze2,freeze_predi
ct,tx_goal,I,cw,cw_,psucc,thr)

         my_ip=str(netifaces.ifaddresses(iface)[netifaces.AF_INET][0]['addr'])
         #my_ip.replace(".","_")
```

```
        out_file="{}.csv".format(my_ip);
        with open(out_file, "a") as myfile:
           myfile.write(out_val+"\n")
```

Table 16 – Third REACT thread: Contention Window tuning

```
def setCW(iface,qumId,aifs,cwmin,cwmax,burst):

   phy=getPHY(iface);

   #SETCW UPI
   edcaParams = edca.EdcaQueueParameters(aifs=aifs, cwmin=cwmin, cwmax=cwmax,
txop=burst)
   edcaParams = edca.EdcaQueueParameters(aifs=1, cwmin=1, cwmax=1, txop=1)
   # UPI

controller.radio.set_mac_access_parameters(iface=iface,queueId=qumId,queueParams=ed
caParams)
```

Table 17 - Call to UPI to enforce a new Contention Window value

b. Experiment setup and results

Experiments have been conducted in Wilab2 testbed, using ZOTAC Wi-Fi nodes equipped with Atheros chipset AR9xxx family and running the ath9k driver. Experiments aim to analyze the performance gain provided by REACT in presence of multi-hop topologies with hidden nodes. Here we present two different experiments: 1) an experiment running on a topology with three nodes in a chain; 2) a more complex experiment with 6 nodes, whose connectivity graph includes a maximum distance of three hops among the nodes. In each experiment, we compare the percentage of channel airtime (also called access persistency) achieved by each node, under legacy DCF and in presence of the REACT mechanism.

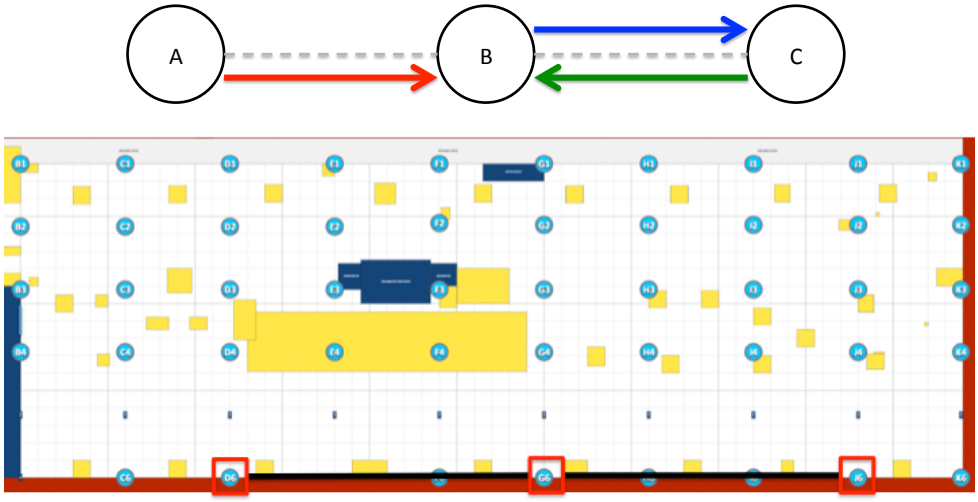


Figure 32 – Network topology of the first experiment: chain of three nodes (node A: ZotacD6, nodeB: ZotacG6, nodeC: zotacJ6).

Figure 32 shows the abstract network topology with a chain of three nodes, where the dashed grey lines indicate the node available links, and the real location of the nodes in the wilabt testbed. Each node sends UDP traffic in saturation mode. Packet size is fixed to 1534 byte and the data rate is fixed to 6Mbps. In the initial experiment, we activate three traffic flows, as depicted Figure 32, by using legacy 802.11 with RTS-CTS. Figure 33 the airtime achieved by each node: it is evident that channel access performance exhibit a clear unfair behaviour. In particular, node B access the channel for

about 41% of the time, while node A and node C succeed in accessing the channel only for about 12% and 23% of the time. When the experiment is repeated using REACT, all the nodes achieve a fair repartition of the channel capacity, with about 30% of channel airtime granted to each one.

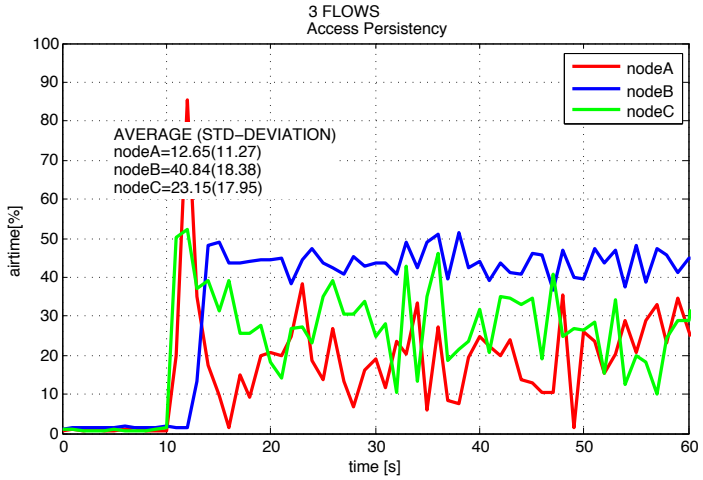


Figure 33 – Normalized airtime achieved under legacy DCF with RTS/CTS enabled in the chain topology.

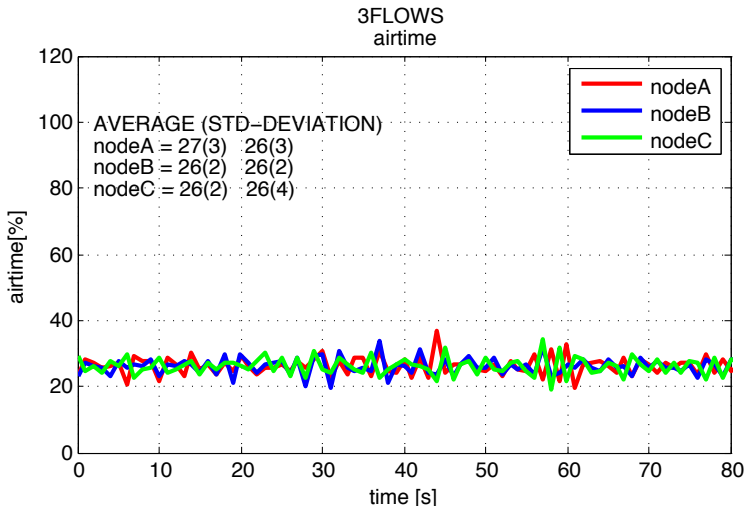


Figure 34 – Normalized airtime achieved under REACT in the chain topology.

In the second experiment, we consider the more complex topology shown in Figure 35, in which we again illustrates the abstract topology, the available links and the active traffic flows (top diagram) and the real positioning of the nodes within the wilab testbed (bottom diagram). In this experiment, we test the REACT performance under dynamic load conditions, by sequentially activating the traffic flows starting at nodes A, B, C, D, E, and F. The effects of the protocol is tuning the contention window as a function of the channel airtime allocated to each node: therefore, in dynamic traffic conditions, the average contention window values change after the activation of each traffic flow, as shown in Figure 36.

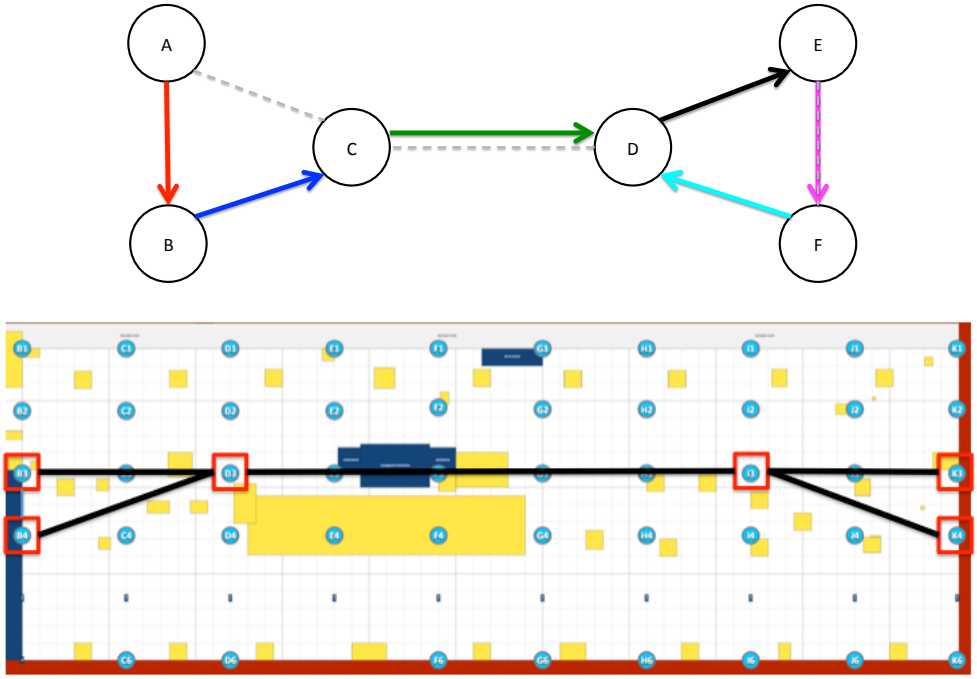


Figure 35 - 6 nodes topology nodeA: zotacB3, nodeB: zotacB4, nodeC: zotacD3, nodeD: zotacI3, nodeE: zotacK3, nodeF: zotacK4

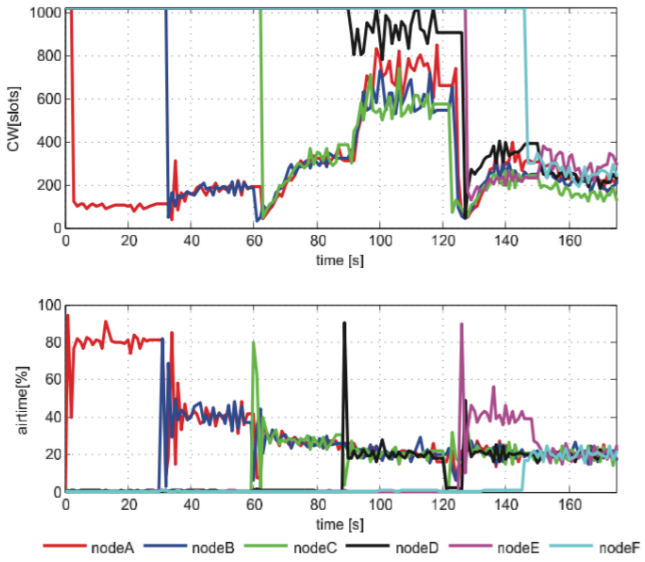


Figure 36 - REACT performance in multi-hop topologies with dynamic traffic conditions: CW values and airtimes achieved by each node.

c. Next steps

The proposed REACT scheme seems very promising for regulating the access rate of nodes in multi-hop topologies, where it is well known that greedy access behaviors can lead to significant performance impairments. We are planning to consider several potential extensions of the scheme for real network deployments. First, we would like to **generalize the concept of channel allocations by considering multi-hop traffic flows**. While the current scheme works by considering each node as independent and by sending claims, which depend only on the local application demand, in real topologies it may happen that the same application flow has to traverse multiple consecutive links. Self-contention of multi-hop traffic flows is a very critical phenomenon for ad-hoc networks. Therefore, it could be very relevant to mitigate self-contention by means of channel allocations

which take into account the application demand across consecutive links. Second, we would like to explore another generalization of the auction mechanism in general topologies in which some neighbor nodes waste channel times of a given transmitted, but send packets that cannot be correctly demodulated. In other words, we would like to **improve the auction robustness in a scenario in which the carrier sense range and the transmission range can be different**. In this case, some claims have to be indirectly estimated by the nodes, which cannot demodulate the REACT control messages. Finally, we want to study, from a theoretical point of view, the stability of the REACT allocations.

3 Definition of Showcases to be implemented in Year 3

The following showcases are envisioned for implementation in the third year of the project. They are described in early form here.

3.1 OTA updates using GITAR for WSNs

More and more Internet-of-thing (IoT) devices are currently being deployed and connected to the Internet. While there are many benefits, there are also numerous possible pitfalls, especially w.r.t. security. Latest example of a serious security problem was the incidents where smart appliances were hacked and repurposed to participate in a massive distributed denial of service attacks (DDOS).

One of the key shortcomings of such devices is that the firmware on most of these devices cannot be updated over-the-air securely. This implies that it is up to the end user to ensure that the firmware of his IoT devices is up to date with the latest security patches. Moreover, software updates are not only necessary for security updates but can also be used to perform bug fixes, feature addition or performance improvements.

The GITAR framework [19] offers a number of building blocks which can be used to enable partial software updates of constrained devices in the IoT. Most importantly, GITAR provides a dynamic linker that can update/add/remove software modules at run-time. Secondly, GITAR enables to convert a static software module into a dynamic software module without requiring any code changes. To facilitate these features, the GITAR framework automatically embeds a component object model (COM) inside the software module.

3.1.1 Overview

In this showcase, the building blocks of GITAR will be integrated in WiSHFUL and used to enable over-the-air (partial) software updates of a WSN devices. The UPI_M interface will be extended with functions that enable it to distribute, install and remove software modules in one or more nodes after deployment.

Given the constrained nature of WSN devices, special attention is required in the protocol used for over-the-air software distribution. It should be possible to evaluate different combinations of mac/routing/transport layer protocols in different scenario's (number of hops, link quality, interference, etc..). For this purpose, the already available UPI_R/N function will be extended (e.g. switching routing/transport protocol is currently not possible at run-time).

To fully exploit the new features GITAR brings into Contiki, some modifications are required in the Contiki network stack. More specifically, currently each layer is statically linked with its upper and lower layer making it impossible to update layers separately. To overcome this, the netstack module in Contiki needs to be refactored.

3.1.2 Goals

The main goal is to extend the WiSHFUL UPI_M interface with functions that enable over-the-air updates of WSN nodes. The second goal is to use/extend the UPI_R/N interfaces for evaluating different software distribution methods. The third goal is to refactor the Contiki network stack in in separate layers so that each can be independently updated and/or replaced.

3.1.3 Breakthroughs

The following list iterates the breakthroughs created in this showcase:

- The possibility to use UPI_M for sending and installing (partial) software updates to constrained IoT devices.
- The ability to evaluate different software distribution methods using UPI_R/N.
- Enabling run-time switching between different versions of a protocol layer in Contiki using UPI_R/N.

3.1.4 Methodology

The following steps will be taken to realize this showcase:

1) Set-up a basic system
   a) Add a basic software distribution method
   b) Integrate GITAR dynamic update capabilities into WiSHFUL framework.
   c) Extend the UPI_M interface with functions to distribute software modules to one or more nodes
   d) Extend the UPI_M interface with functions to control the software installation process.
2) Evaluate basic system
   a) Theoretically analyse the overhead introduced by OTAs.
   b) Verify theoretical analysis by evaluating some scenario's in real-life.
3) Extend basic system
   a) Adding and evaluating different software distribution alternatives.
   b) Refactor Contiki netstack to enable updating and switching separate protocol layers.

3.1.5 Use of WiSHFUL Functionality

During the experimental evaluation of this showcase, WiSHFUL UPI_R/N functionality will be used to monitor the overhead and optimize the behavior of OTA software distribution methods and the UPI_M interface will be extended.

3.2 Extension of MAC adaptation in multi-hop topologies, based on directional antenna and multiple path reservations

3.2.1 Overview

This showcase is an extension of the Year 2 showcase based on the topology-aware and load-aware distributed allocation of airtime in multi-hop topologies. The basic idea of the original showcase was using an auction protocol, called REACT, for computing the airtimes required by the active traffic sources in the network, and a run-time tuning of the contention window for guaranteeing the allocated airtime to each node, while reducing the contention level among hidden nodes.

We plan to generalize the airtime allocation for more general traffic sources. Indeed, in Year 2 each source delivers traffic towards a 1-hop neighbor, while in real ad-hoc topologies it is likely that multi-hop flows are active. These flows cause the well-known self-interference problem in a chain of nodes transmitting the flow frames. Therefore, it could be interesting to perform airtime allocations by considering that the application requirements signaled by a given source node should be propagated along a chain of nodes in a 'multi-hop reservation chain'. A new auction protocol, as well as an estimator of flow demands for the relay nodes, have to be designed.

We plan to also consider another extension based on the availability of programmable antennas, integrated within an OC1 extension. These antennas can be used for generating more complex multi-hop topologies (being the current depth of the network limited to three hops). Moreover, a dynamic reprogramming of the antenna pattern, as a function of the transmission destination, could be performed for mitigating interference in the directions different from the desired one. This feature could require a further protocol extension, based on the introduction of direction airtimes.

3.2.2 Goals

Our goal is designing an effective and robust solution for delivery traffic in ad-hoc networks by mitigating hidden node problems and self-interference generated by traffic flows traversing multiple relay nodes. Our solution is also an interesting cross-layer optimization, in which application requirements and bandwidth demands at relay nodes are mapped into MAC layer (i.e. contention window) and PHY layer (i.e. antenna beam) configurations at run-time.

3.2.3 Breakthroughs

The problem of channel allocations in multi-hop network has been faced from different perspectives, but no solution is currently prominent as the best performing one. Even the concept of channel reservations included into the 802.11s extensions with the mesh deterministic access schemes is not used in practical scenarios because of several technical limits (e.g. problem with node synchronization in mesh networks). We expect that our solution can fill this gap, between theoretical limits and practical feasibility, because it is based on 'loose' node coordination, i.e. on a simple distributed auction protocol which only requires infrequent messages exchanges among adjacent nodes.

3.2.4 Methodology

We will study the extension to REACT, based on multiple airtime allocations of a given traffic flow among the consecutive node relays, by means of a theoretical and/or simulation approach. We will also consider the impact of directional antennas for quantifying channel demands of each node along a specific direction. Once the scheme is defined, it will be validated and refined in real experiments. In parallel, we will study the physical topologies that can be configured in the WiSHFUL wilabt testbed, by using programmable antennas.

3.2.5 Use of WiSHFUL Functionality

WiSHFUL UPI functions will be used for: tuning the contention window of the nodes as a function of the REACT allocations; programming the antenna beams dynamically as a function of the next-hop destination node; programming the antenna beams statically for specific topology requirements; implementing a non-standard control protocol for ad-hoc networks based on the extended REACT protocol.

3.3 Radio-based indoor localization

3.3.1 Overview

Radio-based indoor localization takes advantage of the pervasive coverage of Wi-Fi signals. Despite several solutions that have been proposed in literature, precise localization using Wi-Fi networks is still an elusive target. This is because operational scenarios are fragmented and extremely heterogeneous, therefore a one-size-fits-all radiolocation solution is not feasible. WiSHFUL flexibility, node programmability and controllability allow a *programmable positioning scheme* that can also exploits node coordination.

In this showcase we validate this concept with a radio program that is specifically dedicated to positioning, taking inspiration from passive RADARs and from WIDAR [20]. The exemplary MAC behavior and the proposed scenario are depicted in Figure 37, where several APs coordinated by the global controller running a radio program that, being specifically designed for positioning, has no

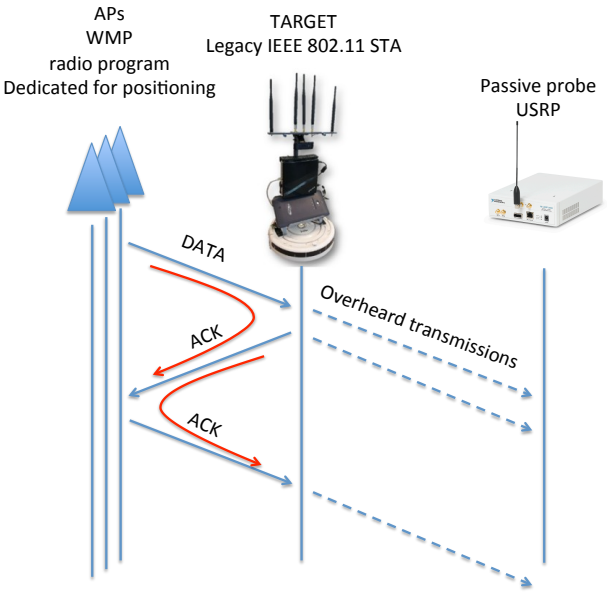


Figure 37 Concept topology for exploiting dedicated radio programs for localization

relevance for data exchange but permits evaluation of RTT in the opposite directions. The target runs standard DCF and it doesn't matter if it is associated to any of the APs. APs send periodic DATA frames that solicit the target. For each received data frame, the target waits for a SIFS and answers with one ACK. As standardized, this happens even if the target is not associated to the AP. The AP radio program is specially designed to answer this ACK with another ACK, sent after a SIFS. The same happens in turns with all other APs. The resulting frame exchanges are overheard by a passive probe, which analyzes the two-way propagation delay over several paths and compensate eventual SIFS heterogeneity between nodes. Moreover, we plan to exploit the possibility to run-time switch from an operating frequency to another in order to evaluate the RTT value experienced with different carrier frequencies.

3.3.2 Goals

The goal of this showcase is to demonstrate the advantages of using radio programs and control programs that are specifically designed for localization over a programmable positioning. In order to keep the scenario realistic, we assume WiSHFUL-enabled radio programmability only on the network side.

3.3.3 Breakthroughs

Realistic and applicable Wi-Fi based radio localization mechanisms exploit protocols of the IEEE 802.11 standards. These protocols have been designed for guaranteeing performance in terms of throughput and fairness among nodes, but localization purposes where not initially addressed. The WMP platform, integrated in WiSHFUL facilities, permits to seamlessly switch between radio programs for data transport and radio programs for positioning, obtaining the best from the latter while maintaining backward compatibility on IEEE 802.11 protocols on the target.

3.3.4 Methodology

The proposed positioning-dedicated radio program resides on WMP-enabled APs in the WiSHFUL testbed, while the target node is a mobile robot equipped with legacy DCF. Dedicated radio programs will be used for soliciting the target and explore the use of machine learning approaches exploiting the WiSHFUL intelligence framework.

3.3.5 Use of WiSHFUL Functionality

The showcase requires several WiSHFUL functionalities: the capability of changing the MAC and tuning its parameters, of adding new frame types, of getting low-level statistics from programmable nodes and low-level radio information from USRPs, used as platforms for advanced sensing.

3.4 Extension of MAC optimizations in high-density scenarios, with online phase

3.4.1 Overview

This showcase is an extension of the Year 2 showcase for MAC adaptation in High-Density networks. The basic idea of the original showcase is to detect pathological topologies in dense networks and enforce appropriate actions. In the original showcase, the detection phase was run offline and was based on simulation data, while the action was implemented online in the WiSHFUL testbed.

In this showcase we will include the detection phase as an online classification, exploiting the WiSHFUL intelligent framework while taking into consideration that the interference conditions can be classified into three main classes depending on the network density [21].

This showcase will extend the archetypal topology used in Year 2 showcase by including more generic dense networks and will explore the improvements due the joint information given by the adjacency matrix (indicating the connectivity between nodes) and the transmission matrix (indicating source and destinations of traffic flows).

3.4.2 Goals

The goal of this showcase is to validate advanced interference detection mechanisms in dense networks, by extending preliminary results obtained with the Year 2 showcase. We will explore in depth the feature space and include the recognition phase in the online experiment, integrating with the inter-BSS MAC and its tuning.

3.4.3 Breakthroughs

The problem of interference coordination has been extensively faced at the node level. The approach of this showcase is to perform an advanced detection of topological issues such as hidden, exposed nodes and flows in the middle. The classification of such phenomena using machine learning is a hot topic that promises performance issues recognition with both centralized and distributed approaches. Phenomena such as hidden and exposed nodes result in similar performance impairments but require opposite actions: limiting the traffic offered by the hidden nodes in one case and stimulating the suffering node in the second one. Classification of phenomena is therefore crucial for best mitigation strategy selection.

3.4.4 Methodology

The advanced detection of pathological interference conditions will be performed through classifiers and neural networks that are defined offline, are trained online and/or offline and used online during the experiment.

3.4.5 Use of WiSHFUL Functionality

In this showcase we will exploit WiSHFUL monitoring capabilities using the Monitoring and Configuration Engine, the intelligent framework for recognizing special interference conditions and control capabilities to enforce and tune the assigned radio program.

3.5 Interference classification for Wi-Fi nodes on the basis of error patterns using machine learning

3.5.1 Overview

The possibility to detect exogenous interfering sources in Wi-Fi networks, such as ZigBee, LTE in unlicensed bands, microwave ovens, and Bluetooth, is a very interesting feature for improving coexistence in ISM bands. We observed that receiver errors generated by exogenous RF signals (i.e. non-Wi-Fi modulated signals) exhibit significant differences (in terms of occurrence probability and error intervals) from the ones generated by collisions with other Wi-Fi transmissions. The errors generated by cross-technology interference have significantly different patterns compared to errors typical of Wi-Fi transmissions. Indeed, in case of wide-band noise and exogenous interference signals, errors may appear randomly at any point during the time the demodulator is active, while for Wi-Fi modulated signals error statistics vary during the frame reception and depend on frame length and rate. Moreover, multiple events can be generated by the receiver during the same interfering transmission in a burst of errors that we define as error pattern. For example, a checksum failure can follow the detection of a good PLCP, or another (failed or not) synchronization trial can be performed after a bad PLCP event according to consecutive resets of the receiver.
In this showcase, we plan to estimate the error statistics for detecting the presence of an interfering source, and then to activate a classifier, working on the processing of consecutive error bursts, for identifying the interfering source among a set of pre-defined sources or for signaling that a specific interfering source is unknown. The classifier will be studied off-line, on the basis of experimental traces collected under controlled interfering sources, for modeling the receiver behavior.

3.5.2 Goals

The goal of this showcase is demonstrating how the WiSHFUL functionalities can be exploited for collecting and processing data traces in different network experiments devised to build data-driven

models of Wi-Fi receivers in presence of various interfering sources and how the data-driven model can be easily implemented in a real network for interference classification.

3.5.3 Breakthroughs

Interference detection of non-Wi-Fi modulated signals is usually based on SDR platforms or simultaneous multi-technology receivers. An approach working with a commercial card in explored in Airshark by using an 802.11n PHY able to read RSSI values at different sub-carriers and by sequentially moving a Wi-Fi monitoring card to the adjacent channels with steps of 5 MHz. In case of sudden disappearance of the RF signals when moving from one channel to the next one, it can be assumed that interference was due to a narrow-band ZigBee channel. Complex algorithms are applied to the RSSI samples for characterizing spectral, energy and pulse signals that are mapped into a technology classification scheme. While this previous works relies on the classical analysis of the frequency and time domains, in this showcase we plan to study the error domain, i.e. the errors produced by the interfering technologies. We expect that such an approach can be much more general, because we are able to train the classifier and to recognize a given interference source without knowing the technical details (bandwidth, pulse signals, energy) of the source physical layer.

3.5.4 Methodology

This showcase is devised to build a specific network intelligence by using machine-learning techniques. The idea is using error statistics and specific error patterns for estimating the timings and the effects of the interfering technology on a given Wi-Fi receiver. The temporal analysis of the receiver events is affected by the receiver implementation because the demodulator reset time in case of false or bad preambles depends on the card internal design and results in a different granularity of consecutive events. It follows that we plan to first train a classifier on the basis of experimental traces collected under controlled interfering sources for designing a data-driven model of the Wi-Fi receiver. Then, the model will be used in real experiments under unknown interference.

3.5.5 Use of WiSHFUL Functionality

We plan to use UPI_R functions for collecting the error statistics and error patterns experienced by Wi-Fi receivers based on the WMP architecture. We will also use UPI_R for generating interfering sources, including LTE modulated signals in ISM bands and non-standard modulations. The WiSHFUL intelligence framework will be used for building the data-driven model and for implementing the new interference classifier.

3.6 Radio virtualization with simultaneous transmission and reception

3GPP is currently standardizing NarrowBand-IoT (NB-IoT). This radio access technology aims to provide cost-effective connectivity services for billions of devices around the world, supporting low power consumption, the use of low-cost devices and provision of excellent coverage. NB-IoT is to be deployed in the same channels used by standard LTE mobile carriers; usually sharing the channel with a LTE deployment. In a previous showcase (see section 2.2.1), we explored the virtualization of LTE and NB-IoT base stations to ease the deployment of these technologies. However, our virtualized base stations had the limitation of being able only to transmit signals towards their clients, i.e., clients could not transmit to their virtual base stations. In this showcase, we will show the extension of the radio virtualization framework that adds capability to virtual radios to also receive signals from the clients, as illustrated in Figure 38.

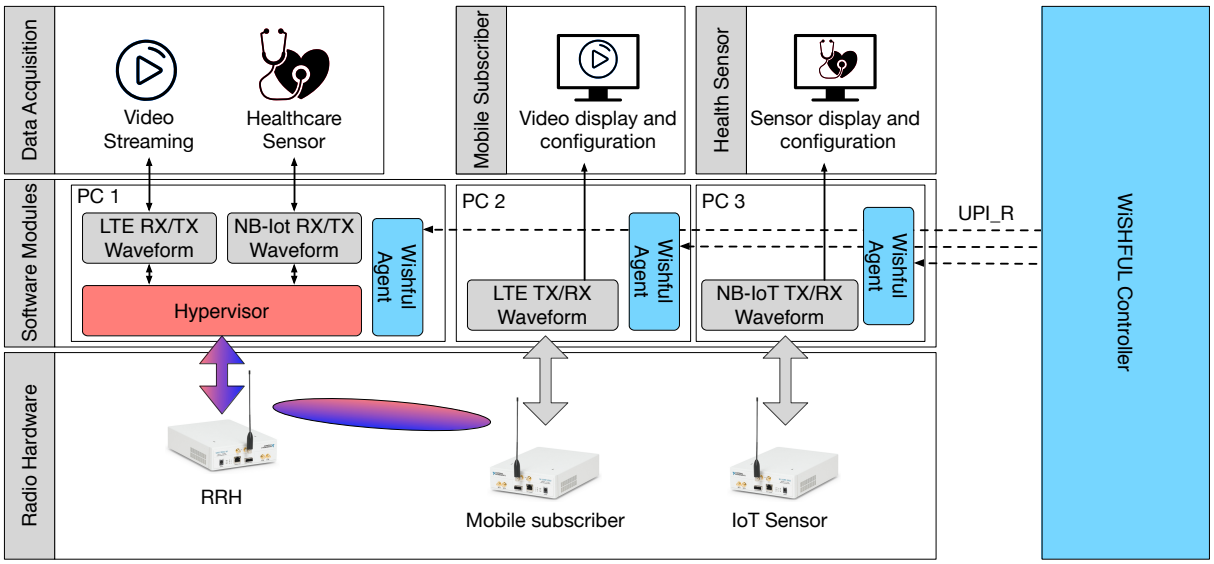


Figure 38 Radio virtualization showcase with transmission and reception

The interesting scenarios, as far as the type and volume of data between the devices for this showcase, can be described as:

- LTE: In the downlink (from LTE Base station to mobile subscriber) we will have a video-streaming; in the uplink (from mobile subscriber to the LTE base station) we will have a feedback regarding the video quality. The video-streaming application will use the information received from the mobile subscriber to adjust the video-quality;
- NB-IoT: In the downlink (from NB-IoT base station to the Healthcare sensor display) we will have the data collected by the Healthcare sensor; in the uplink (from Healthcare display to the NB-IoT base station) we will have a configuration of some parameter of the sensor (time interval between sensor readings for example). The Healthcare sensor will use this information to adjust its configuration accordingly.

3.6.1 Goals

The goals of this showcase is to demonstrate the advantages that radio virtualization can bring to future mobile wireless networks. We use Wishful to enable the configuration of virtual radios and USRP devices. In particular, we will provide access to several parameters of both LTE and NB-IoT radios, and central frequency and bandwidth of the USRP.

3.6.2 Breakthroughs

Radio virtualization is currently being considered in state-of-art architectures for the next generation of mobile networks. We believe that a practical implementation of such mechanisms is the key to analyze its impact on wireless communications.

3.6.3 Methodology

The proposed radio virtualization framework will be evaluated by means of experiments in the IRIS testbed. As the performance metric we will compare the throughput and latency of waveforms with and without the virtualization framework, the difference in SNR with and without the virtualization framework, and the computational overhead of the virtualization framework.

3.6.4 Use of WiSHFUL Functionality

The following WiSHFUL functionality from the GNU Radio module will be used:

- The functionality to change the central frequency and bandwidth of the Hypervisor.
- The functionality to change specific parameters of virtual radio waveforms (number of carriers and MCS in LTE for example).

3.7 IEEE 802.11 Overlapping BSS management

3.7.1 Overview

IEEE 802.11 Basic Service Sets (BSSs) working on the same radio channel are becoming common because of the wide diffusion of 802.11 networks and limited availability of channels. Although carrier sense mechanism in principle does not require any frequency planning, it has been shown that severe performance impairments can occur due to the ***neighbor capture effect***. It occurs when a BSS is between two BSSs which do not hear each other (Figure 39). In presence of greedy traffic, the BSS in the middle can be prevented from accessing the channel indefinitely because it senses the medium permanently busy.

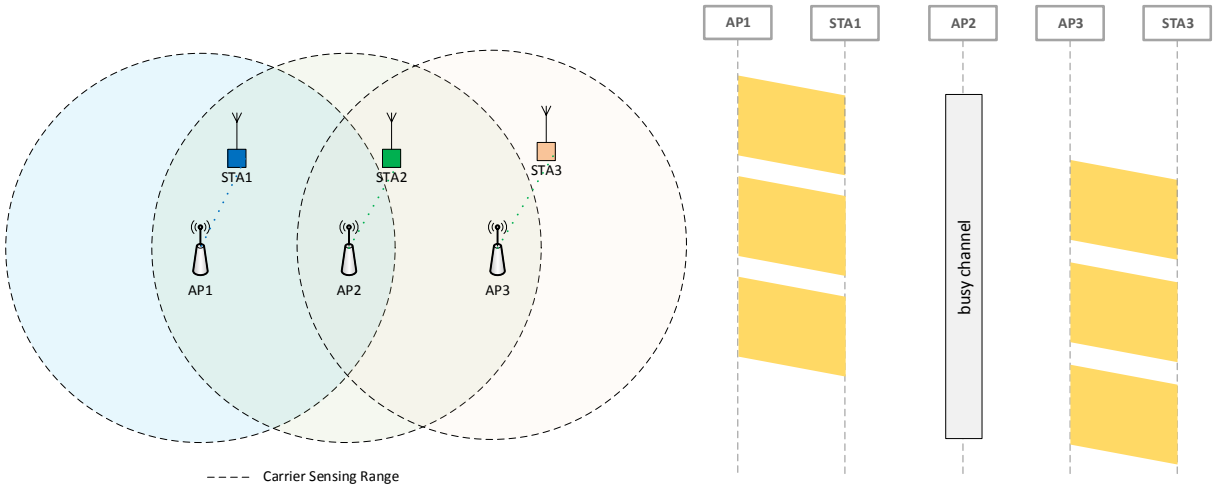


Figure 39. Neighbor capture effect in IEEE 802.11 networks.

To limit the neighbor capture effect and extend admission control / scheduling decisions, a new mechanism called ***OBSS management*** is required.

3.7.2 Goals

The envisioned OBSS management mechanism is based on two main components. The first component is responsible for monitoring (quantifying) the load and interference status of each BSS and for signaling this information to the neighboring BSSs. The latter can be done using either beacon frames or QLoad report frame as currently discussed in the 802.11ax working group. The second component performs the actual OBSS management. In this showcase we will highlight two different approaches. First, a channel selection which takes the neighbour capture effect explicitly into account when calculating the frequency plan. Second, an approach where neighbouring BSSs are cooperating with each other for resource sharing on the basis of such information. In particular, we considered time slot medium access where exclusive time slots are assigned to APs suffering from neighbor capture effect.

3.7.3 Breakthroughs

OBSS management is being actively discussed in the IEEE 802.11ax working group. We believe that the proposed mechanisms and results from this showcase would be of great importance to the 802.11 community and would allow significantly faster evaluation/comparison of the proposed coordination algorithms.

3.7.4 Methodology

The proposed algorithms for OBSS management will be evaluated by means of experiments in an 802.11n testbed. As the performance metric we will compute the throughput fairness (e.g. Jain's fairness index) of each BSS.

3.7.5 Use of WiSHFUL Functionality

The following WiSHFUL functionality from the 802.11 (Wi-Fi) module will be used: a.) the functionality to change the radio channel of BSSs (APs), and b.) the possibility to perform time slotted medium access using the hybrid CSMA/TDMA MAC of the Atheros platform.

3.8 Closed-loop rate control for IEEE 802.11 infrastructure networks

3.8.1 Overview

The rate control algorithms of today's IEEE 802.11 networks are mostly open-loop, e.g. Linux Minstrel algorithm. This means that the Wi-Fi transmitter adapts the bitrate (MCS) based on link-level measurements (probing). Unfortunately, open-lop rate control, while being simple to implement, is inefficient especially in mobile environments as compared to closed-loop approaches. In closed-loop rate control the receiver estimates the rate (MCS) based on measuring the actual channel quality and signaling its value to the transmitter.

3.8.2 Goals

The goal of this showcase is to demonstrate that Wishful enables the required functionality for closed-loop rate control (Figure 40). In particular, we will provide access to Channel State Information (CSI) on a per packet basis, which can be used as input for computation of effective SNR values at the receiver side. From the effective SNR the proper bitrate (MCS) is selected and signaled to the transmitter node, which will use it for the next frame transmission.

In this showcase the signaling will be out-of-band, i.e. using the Wishful control framework. This allows us to study the impact of feedback delay and losses.

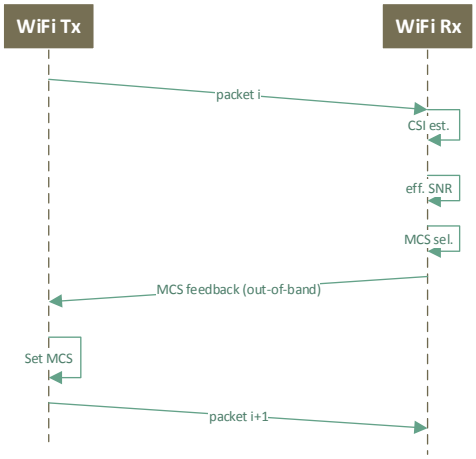


Figure 40. Closed-loop rate control.

3.8.3 Breakthroughs

This showcase will highlight the suitability of using Wishful control platform for studying closed-loop rate control algorithms for Wi-Fi.

3.8.4 Methodology

We plan to implement a specific closed-loop rate control algorithm for Wi-Fi infrastructure networks. The algorithm will be evaluated by means of experiments in an 802.11n testbed. As the performance metric we will compute the throughput and compare it to state-of-the-art open-loop approaches, e.g. Linux Minstrel. Furthermore, we will analyze the impact of feedback delay and loss.

3.8.5 Use of WiSHFUL Functionality

The following WiSHFUL functionality for IEEE 802.11 is required. First, the collection of CSI values on receiver side on a per packet basis. Currently, we support this for the Wi-Fi-Atheros platform. We plan to support it also for Intel-Wi-Fi platform. Second, functionality for inspecting CSI and computing effective SNR from CSI values is required. Third, the functionality for setting the wireless data rates (MCS) at the transmitter side on a per packet basis is required.

3.9 Context Awareness in spectrum management system-aided SU networks

3.9.1 Overview

Over the last decade, regulators, academic and industry bodies have been focusing significant research efforts on the topic of spectrum sharing, as a way to overcome the spectrum crunch problem. Cognitive Radio is one of the possible approaches to enable sharing between different technologies. However, there has been an increasing agreement that cognitive techniques based solely on local sensing may not be enough to protect primary users (PUs) from interference. As a result, regulators have shifted their attention to spectrum sharing approaches that rely on network-wide spectrum management systems (SMS). Examples of SMS include the Spectrum Access System (SAS) in US, Licensed Shared Access (LSA) and TVWS geo-location databases, and radio environment maps (REM). On the other hand, questions remain regarding the use of these management systems in radio environments where the PUs display a highly dynamic channel access behaviour.

In this work, we study different learning and agile decision making techniques that SUs can employ in dynamic radio environments, while supervised and supported by a centralized intelligence spectrum management system. Our system model comprises measurement-capable devices (MCDs), a SMS, and SU networks. The MCDs can be either dedicated sensors or SUs, that collect long-term statistical

data of the radio environment of the PUs' behaviour. The collected statistics, and features from the radio environment are then sent and aggregated at the SMS, which will recognize several distinct sharing scenarios SUs may encounter. For each scenario, the SMS will generate a list of channel access strategies that SUs may employ to avoid interference with the PUs.

3.9.2 Goals

- Design of several long-term channel statistics gathering, and deep learning-based representation algorithms that will be employed by MCDs to characterize PUs' behavior
- Derivation of optimal channel access strategies/techniques and spectrum sensing configurations by the SMS, based on gathered and aggregated data from the MCDs
- Configuration of SUs' channel access and sensing algorithms by the SMS

3.9.3 Breakthroughs

Our vision is to offload power, time and hardware-demanding sensing, classification, and decision making procedures from battery-powered SUs to dedicated sensor networks and SUs that are plugged into power sources (e.g. base stations), and to network-wide intelligent SMSs, respectively. Battery-powered SUs will be configured by the SMS and will only have to perform short-term low-complexity sensing procedures to classify their environment and select the appropriate channel access strategy that minimizes interference to PUs.

Offloading complexity from mobile SUs will allow the use of advanced deep learning classification and Markov Chain-based algorithms that succinctly characterize the PUs' behaviour, and narrow down the set of possible scenarios and channel access strategies a SU may encounter and apply, respectively. Overall, this approach will result in a more spectrally efficient operation by SU networks.

3.9.4 Methodology

Our experimentation scenario will comprise a PU-Tx and a PU-Rx, which can switch between different waveforms and channel access patterns. The PU-Rx will also report to the SMS its experienced interference over time. A SU-Tx in the same radio environment will be in contact with the SMS from where it collects the list of configurations (e.g. possible hopping patterns and parameters) used by the PUs, and consults its caused interference. Based on this information and short-term sensing, the SU-Tx will adapt its channel hopping pattern, transmit power, bandwidth, modulation order, and other parameters to maintaining the interference at PUs below a specified threshold, and maximize the number of packets that reach a SU-Rx. Simultaneously, a sensor will perform long-term sensing, and populate the SMS with possible scenarios, each characterized by a set of PU's features.

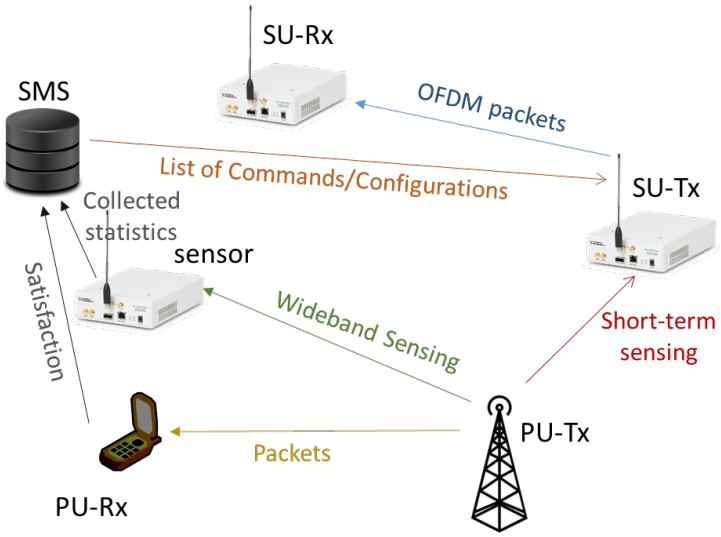


Figure 41 – Illustration of the several agents and interactions present in the context-aware SMS-aided spectrum sharing framework.

3.9.5 Use of WiSHFUL Functionality

WiSHFUL UPIs will be employed to set each element of the experiment running, perform configuration of the PU and SU parameters, such as bandwidth and range of transmit powers, and exchange of control information between PUs, SMS, sensor and SUs.

4 Conclusion

This document summarizes the goals and results of showcases presented in Year 2 of the project. It extends the first set of showcases described in D2.3. The showcases presented herein define relevant and convincing scenarios in view of promoting the WiSHFUL framework capabilities. Each showcase provides a high-level specification of open software platforms for radio and network control driven by domain-specific requirements from different relevant market segments. This document demonstrates the impact of WiSHFUL research and technologies in developing new solutions to resolve challenges found on wireless communication networks. Demonstrations include:

- The IRIS SDR demo integration with UPIs that change parameters on the fly at USRPs.
- The Coexistence of IEEE 802.15.4e TSCH with IEEE 802.11 networks proves the solution feasibility and applicability for a cross technology synchronization scheme between TSCH and Wi-Fi networks.
- We have proved that the optimization logic can be implemented on WiSHFUL programmable nodes with or without presence of legacy nodes.
- The load and topology aware networking showcases demonstrate how the WiSHFUL UPIs can be used to dynamically monitor network performance and topology, change network protocol configuration, or switch routing modules, and mitigate performance impairments.

Additionally, this document discusses extensions or continuations of the work completed so far to further exploit the contributions of the project. These include:

- Implementation of the LTE-U Wi-Fi coexistence showcase.
- Using GNU Radio to facilitate radio equipment sharing.
- Measure packet loss in both IEEE 802.15.4e TSCH and Wi-Fi networks and with implemented showcase system running
- Radio Slicing for Virtualized Home Wi-Fi Access Points.
- Investigate the possibility of guaranteeing coexistence among heterogeneous access protocols and scheduling mechanisms.
- The implementation of UPIs with additional features such as extracting the A-MPDU transmission results from BlockAck and enabling/disabling the mobility-aware algorithm which adapts PHY rate and frame aggregation length in real-time.
- Aspects such as traffic patterns, MAC protocols, node mobility, and so forth can be investigated further as part of the optimal link estimator showcase. Additionally a benchmarking toolbox could be added to support program execution with different parameter settings might be developed.
- We are also planning to consider several potential extensions to the REACT scheme for real network deployments including: Generalizing the concept of channel allocations; improving auction robustness; and the stability of the REACT allocations from a theoretical point of view

This document also outlines the utility and impact of WiSHFUL in project showcases. Technical details of these showcases are outlined in the appropriate technical deliverables D3.4, D4.4, and D6.4. Finally, this document defines a list of intelligence showcases to be implemented in the final year of the project.

References

[1] Project Deliverable D2.3 - Results of first set of showcases, http://www.wishfulproject.eu/sites/default/files/images/review/WiSHFUL_D2.3_TCD_R_PU_2016-01-04_Final.pdf, last access December 5th 2016.

[2] National Instruments, Real-time LTE/Wi-Fi Coexistence Testbed, http://www.ni.com/white-paper/53044/en/, 2016.

[3] Jindal, Nihar, Breslin, Don and Norman, Alan, "LTE-U and Wi-Fi: A Coexistence Study by Google", *Wi-Fi LTE-U Coexistence Test Workshop*, 2015.

[4] National Instruments, Real-time LTE/Wi-Fi Coexistence Testbed, http://www.ni.com/white-paper/53044/en/, 2016.

[5] Olbrich, Michael, Zubow, Anatolij, et al., "WiPLUS: Passive LTE-U Co-Channel Interference Detection using Commodity Wi-Fi Hardware", Technical Report, 2016.

[6] P. D. Sutton et al., "Iris: an architecture for cognitive radio networking testbeds," in IEEE Communications Magazine, vol. 48, no. 9, pp. 114-122, Sept. 2010.doi: 10.1109/MCOM.2010.5560595

[7] Controlling the IRIS SDR Framework using WiSHFUL UPIs, http://www.wishful-project.eu/sites/default/files/Short-Final-WebSite.mp4, last access December 5th 2016.

[8] "IEEE Standard for Local and metropolitan area networks–Part 15.4: Low-Rate Wireless Personal Area Networks (LR-WPANs) Amendment 1: MAC sublayer," *IEEE Std 802.15.4e-2012 (Amendment to IEEE Std 802.15.4-2011)*, pp. 1–225, apr 2012

[9] S. Duquennoy, B. Al Nahas *et al.*, "Orchestra: Robust Mesh Networks Through Autonomously Scheduled TSCH," in *Proc. 13th ACM Conf. Embed. Networked Sens. Syst.*, ser. SenSys '15. New York, NY, USA: ACM, 2015, pp. 337–350

[10] "Linux Wireless: ath9k spectral scan," 2016.

[11] S. Zehl, A. Zubow, and A. Wolisz, "hMAC: Enabling Hybrid TDMA/CSMA on IEEE 802.11 Hardware," Telecommunication Networks Group, Technische Universität Berlin, Tech. Rep. TKN-16-004, nov 2016

[12] Jeongkeun Lee, Mostafa Uddin, Jean Tourrilhes, Souvik Sen, Sujata Banerjee, Manfred Arndt, Kyu-Han Kim, and Tamer Nadeem. 2014. meSDN: mobile extension of SDN. In Proceedings of the fifth international workshop on Mobile cloud computing & services (MCS '14). ACM, New York, NY, USA, 7-14.

[13] Baccour, N., Koubˆaa, A., Mottola, L., Z´u˜niga, M.A., Youssef, H., Boano, C.A., Alves, M.: Radio link quality estimation in wireless sensor networks: A survey. ACM Trans. Sen. Netw. 8(4), 34:1–34:33 (Sep 2012)

[14] Fonseca, R., Gnawali, O., Jamieson, K., Philip, L.: Four bit wireless link estimation. In: Proceedings of the Sixth Workshop on Hot Topics in Networks (HotNets (2007)

[15] Baccour, N., Koubˆaa, A., Youssef, H., Jamˆaa, M.B., Do Rosario, D., Alves, M., Becker, L.B.: F-lqe: A fuzzy link quality estimator for wireless sensor networks. In: Wireless Sensor Networks, pp. 240–255. Springer (2010)

[16] Meier, A., Rein, T., Beutel, J., Thiele, L.: Coping with unreliable channels: Efficient link estimation for low-power wireless sensor networks. In: Networked Sensing Systems, 2008. INSS 2008. 5th International Conference on. pp. 19–26 (June 2008)

[17] Thubert, P.: Objective Function Zero for the Routing Protocol for Low-Power and Lossy Networks (RPL). RFC 6552 (Proposed Standard) (Mar 2012)

[18] Javaid, N., Javaid, A., Khan, I., Djouani, K.: Performance study of etx based wireless routing metrics. In: Computer, Control and Communication (IC4) (2009)

[19] Ruckebusch, P., De Poorter, E., Fortuna, C., & Moerman, I. (2016). GITAR: Generic extension for Internet-of-Things ARchitectures enabling dynamic updates of network and application modules. Ad Hoc Networks, Volume 36, Part 1, January 2016, Pages 127–151

[20] P. Gallo, S. Mangione, and G. Tarantino, “Widar: Bistatic wi-fi detection and ranging for off-the- shelf devices,” in World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2013 IEEE 14th International Symposium and Workshops on a, pp. 1–6, June 2013.

[21] E. Khorov, A. Kiryanov, A. Krotov, P. Gallo, D. Garlisi, I. Tinnirello, “Joint Usage of Dynamic Sensitivity Control and Time Division Multiple Access in Dense 802.11ax Networks”, Multiple Access Communications MACOM 2016

[22] National Instruments whitepaper, “Real-time LTE/Wi-Fi Coexistence Testbed”, http://www.ni.com/white-paper/53044/en/