



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

Project Deliverable D2.1

High level requirements for testbeds and software platforms

| | |
|--------------------------------------|--|
| Contractual date of delivery: | 31-03-2015 |
| Actual date of delivery: | 31-03-2015 |
| Beneficiaries: | IMINDS, TCD, CNIT, TUB, NCENTRIC, RUTGERS, SNU |
| Lead beneficiary: | Name of lead beneficiary (acronym) |
| Authors: | Carolina Fortuna (IMINDS), Anatolij Zubow (TUB), Mikolaj Chwalisz (TUB), Adam Wolisz (TUB), Nicholas Kaminski (TCD), Luiz DaSilva (TCD), Peter Ruckebusch (IMINDS), Pieter Becue (IMINDS), Ilenia Tinnirello (CNIT), Pierluigi Gallo (CNIT), Ivan Sesar (RUTGERS), Sunghyun Choi (SNU) |
| Reviewers: | Ingrid Moerman (IMINDS) Robin Leblon (NCENTRIC) |
| Work package: | WP2 – General requirements and showcases |
| Estimated person months: | 6 |
| Nature: | R |
| Dissemination level: | PU |
| Version: | 1.0 |

Abstract:

Starting from a driving scenario, this deliverable defines the high-level functional requirements for radio and network control and according unified programming interfaces. Besides the requirements for radio and network control, we also define functional requirements for the portable testbed that can be deployed at any location allowing validation in the real world and involving real users. This deliverable further identifies the status of the compliance of the testbeds involved in the project with the Fed4FIRE rules, and outlines the plans for conformation. This deliverable finally reports on the questionnaire that was organized to understand the most stringent needs from wireless innovators, and gives an overview of the answers we collected until the end of March 2015.

Keywords:

functional requirements, radio control, network control, intelligent hierarchical control, portable testbed, questionnaire

Executive Summary

This month 3 deliverable reports on the functional requirements of the software platforms to be developed by WiSHFUL in view of radio and network control. A driving scenario for the project is first provided, then the proposed conceptual architecture is described and three unified programming interfaces (UPIs) are introduced: UPI_R for radio control, UPI_N for network control and UPI_{HC} for regular as well as intelligent hierarchical control. For each of the three UPIs, a set of functional requirements is defined. Besides the requirements for the interfaces, we also define functional requirements for the portable testbed that enables the deployment of testbeds on demand in locations that are different from the ones of the original testbeds from the project. Finally, we assess the current status and the planned federation compliance of the participating testbeds, as well as describe the approach and initial results from contacting stakeholders to help prioritize the technical development within the project.

List of Acronyms and Abbreviations

| | |
|----------|--|
| AM | Aggregate Manager |
| AODV | Ad hoc On-demand Distance Vector |
| AP | Access Point |
| API | Application Programming Interface |
| BAN | Body Area Network |
| CPU | Central Processing Unit |
| CSMA | Carrier Sense Multiple Access |
| DMT | Discrete MultiTone |
| DSL | Digital Subscriber Loop |
| DSR | Dynamic Source Routing |
| DT | Delay Tolerant |
| EM | ElectroMagnetic |
| EMS | Experiment Management Server |
| EWMA | See Figure 8 |
| F4F | Federation for FIRE (Future Internet Research Experimentation) |
| FBMC | Filter Bank Multi-Carrier |
| Fed4FIRE | Federation for FIRE (Future Internet Research Experimentation) |
| FRCP | Federated Resource Control Protocol |
| FSM | Finite State Machine |
| HAL | Radio Abstraction Layer |
| HTTPS | HyperText Transfer Protocol Secure |
| I/Q | In phase / Quadrature |
| IEEE | Institute of Electrical and Electronics Engineers |
| IP | Internet Protocol |
| KPI | Key Performance Indicator |
| LTE | Long Term Evolution |
| LTE-A | Long Term Evolution - Advanced |
| LQI | Link Quality Indication |
| PRR | Packet Reception Rate |
| MA | See figure 8 |
| MAC | Medium Access Control |
| MTC | Machine-Type Communications |
| NOC | Network Operations Center |
| OFDM | Orthogonal Frequency Division Multiplexing |
| OLSR | Optimised Link State Routing |

| | |
|-------------------|--|
| OMF | OMF Measurement Library |
| OML | Orbit Management Framework |
| PLC | Power Line Communication |
| QoS | Quality of Service |
| RAM | Random Access Memory |
| RF | Radio Frequency |
| RP | Radio Processor |
| RSpec | Request Specification |
| RSSI | Received Signal Strength Indication |
| RT | RealTime |
| SDR | Software Defined Radio |
| SFA | Slice Federation Architecture |
| SSH | Secure SHell |
| TCP | Transmission Control Protocol |
| TDM | Time Division Multiplexing |
| TDMA | Time Division Multiple Access |
| TSMP | Time Synchronized Mesh Protocol |
| UC | Use Case |
| UMTS | Universal Mobile Telecommunications System (UMTS) |
| UPI | Unified Programming Interface |
| UPI _R | Unified Programming Interface radio |
| UPI _N | Unified Programming Interface network |
| UPI _{HC} | Unified Programming Interface hierarchical control |
| URL | Uniform Resource Locator |
| USB | Universal Serial Bus |
| VPN | Virtual Private Network |
| Wi-Fi | Wireless Fidelity |
| XFSM | eXtended Finite State Machine |

Table of contents

| | | |
|----------|---|-----------|
| 1 | Introduction | 7 |
| 2 | Driving Scenarios | 8 |
| 2.1 | Home usage | 8 |
| 2.2 | Smart City usage | 9 |
| 2.3 | Public Event usage | 9 |
| 3 | The WiSHFUL Concept | 11 |
| 3.1 | Towards a unified radio control software architecture | 12 |
| 3.1.1 | High-level software architecture..... | 12 |
| 3.1.2 | UPI _R - UPI for Radio Control | 13 |
| 3.2 | Towards a unified network control software architecture | 14 |
| 3.2.1 | High-level software architecture..... | 14 |
| 3.2.2 | UPI _N - UPI for Network Control | 14 |
| 3.3 | Towards intelligent radio and network control | 15 |
| 3.3.1 | High-level software architecture..... | 15 |
| 3.3.2 | UPI _{HC} - UPI for Intelligent Control..... | 17 |
| 3.4 | Flexibility Requirements | 18 |
| 4 | Requirements for UPI_R..... | 20 |
| 4.1 | Use cases | 20 |
| 4.1.1 | Coexistence of heterogeneous technologies | 20 |
| 4.1.2 | Intelligent download use case..... | 21 |
| 4.2 | Functional requirements | 22 |
| 5 | Requirements for UPI_N | 26 |
| 5.1 | Intelligent download use case | 26 |
| 5.2 | Functional requirements | 28 |
| 6 | Requirements for enabling intelligence | 31 |
| 6.1 | Intelligent download use case | 31 |
| 6.2 | Functional requirements | 32 |
| 7 | Requirements for portable testbed | 34 |
| 7.1 | Portable testbed architecture | 34 |
| 7.2 | Equipment Portability | 36 |
| 7.2.1 | Hardware requirements..... | 37 |
| 7.2.2 | Transportation & setup of hardware | 37 |

| | |
|---|-----------|
| 7.3 Functional requirements | 40 |
| 8 Testbed requirements: Fed4FIRE compliance..... | 43 |
| 8.1 Fed4FIRE architecture | 43 |
| 8.1.1 Resource discovery, requirements, reservation and provisioning | 44 |
| Central location(s) | 45 |
| Testbed side | 45 |
| 8.1.3 Monitoring and measurements | 45 |
| 8.1.4 Requirements for experiment control | 46 |
| 8.2 Advanced federation..... | 46 |
| 8.2.1 Requirements of advanced federation | 47 |
| 8.2.2 Benefits of advanced federation | 48 |
| 8.3 Light federation | 48 |
| 8.3.1 Requirements for light federation | 49 |
| 8.3.2 Benefits of light federation | 49 |
| 8.4 Associated testbeds | 50 |
| 8.5 Status of WISHFUL testbeds | 50 |
| 8.6 Steps to join the Fed4FIRE federation..... | 51 |
| 9 Consultations with innovators | 52 |
| 9.1 PART 1: Gaining insight in the activities of the industry partner | 52 |
| 9.2 PART 2: Current development practices | 52 |
| 9.3 PART 3: Identifying the technical focus points for WISHFUL | 57 |
| 9.4 Summary of results | 63 |
| 10 Conclusion..... | 65 |
| 11 References | 66 |
| Annex 1 Industry Questionnaire | 67 |

1 Introduction

This first deliverable reports on the work planned in WP2 “General requirements and showcases” and is aligned with three of the four objectives/tasks of the WP (for the 4th objective/task a separate deliverable is planned):

- defines the high-level functional requirements for radio control (Section 0), network control (Section 5) and intelligent hierarchical control (Section 0)
- defines high-level functional requirements for realizing portable testbeds (Section 0)
- identifies the status of compliance with Fed4FIRE (F4F) federation rules for the testbeds involved in the project and their plans for conformation (Section 0)

The document first describes the driving scenario consisting of dense wireless communications that inspires the work in this project (Section 2). We detail of how future wireless communications will look like inside the home, within a smart city and at a very crowded public event such as a concert or riot. Then, Section 0 provides the conceptual architecture for the WiSHFUL project and introduces the Unified Programming Interfaces (UPIs) that enable radio (UPI_R), network (UPI_N) and hierarchical intelligent control (UPI_{HC}). These programming interfaces are central to the project and the flexible control of radio networks that we are aiming for. Therefore, prior to defining the functional requirements for these interfaces, we focus on an intelligent download scenario and we identify the key enablers from the three perspectives (radio, network, intelligence).

The functional requirements for the UPIs introduced in Section 0 are listed in Sections 0 (UPI_R), 5 (UPI_N) and 0 (UPI_{HC}). Additionally, the software modules developed within the project will also enable fully flexible control of portable testbeds. The concept and architecture behind the portable testbed are presented in Section 0. This section also identifies an additional set of functional requirements related to equipment mobility and transportation.

A mechanism for uniformly connecting and performing experiments on fixed and portable testbeds is important for making them accessible to as many users as possible. This will be achieved through Fed4FIRE (F4F) compliance and Section 0 presents the overall F4F federation architecture along with the three federation options. The WiSHFUL testbeds are already fully or partially F4F compliant and full federation is planned as reported in Section 8.6.

During proposal writing, we have discussed with many innovators from small, medium-size and large enterprises from different vertical markets to identify their most pressing needs with the goals of (i) prioritizing the WiSHFUL research and (ii) deriving generic research scenarios and proof-of-concepts that will demonstrate the benefits of WiSHFUL. Most of them have expressed their interest in the WiSHFUL software platforms. Section 0 describes the questionnaire developed for the purpose of continuing the interaction with these stakeholders and gives a preliminary overview of some of the answers that were collected in preliminary consultations with the industrial partners.

2 Driving Scenarios

The scenario considered in our work refers to the emerging wireless ecosystem, where multiple technologies, operators and service providers coexist in the same RF environment characterized by a high-density deployment of wireless devices.

Because of the heterogeneous capabilities of the devices (in terms of spectral bands, coverage, management functionalities, networking models, etc.), the lack of open vendor-independent configuration interfaces, and the conflicting goals of independent providers, the integration of these technologies is currently very limited.

Indeed, wireless devices typically employ one or more radio interfaces implementing specific wireless standards (such as LTE, Wi-Fi, Bluetooth), while in niche contexts more advanced programmable interfaces, based on SDR, may also be available. These devices are generally organized into coexisting wireless systems, including:

1. peoples' body area networks (BAN);
2. their gateways to the Internet (i.e. typically tablets and mobile phones) and the relevant access network;
3. smart buildings, smart cities or more generally speaking, smart environments.

Looking at the future deployments, the BAN is usually composed by wearable well-being or health related sensors, micro-cameras or implants, cognitive and physical augmentation devices (e.g. glasses and retina displays), etc., which are connected to a gateway or mobile device and possibly interact with a robot companion or assistant (particularly in case of elderly or disabled people). The access technologies are mainly represented by cellular technologies and Wi-Fi, while building and city automation systems are equipped with massive low-cost/low power devices, intelligent cameras, and pervasive access points.

Different wireless services with heterogeneous requirements can be simultaneously active in the ecosystem, such as access to the Internet, collection of monitoring or metering data, support of mission-critical machine-type communications (MTC) and other reliable services for security and business purposes. Each service may depend on the specific context and location in people's daily life. Firstly, one rather static and controlled context is the home environment (the same scenario can also be applied to most small office environments). Secondly, a mostly mobile and dynamic context is given by a smart city where users move by foot or by vehicle (the same scenario could be considered for large working environments such as certain factories or production environments). Thirdly, a very dense and somewhat ad-hoc scenario is represented by the participation at a large public event or gathering.

2.1 Home usage

Despite their apparent simplicity, the manageability of home networks presents a compelling problem. Since several wireless technologies operate on the same frequency bands and several home networks can interfere with each other, it is necessary to consider coexistence and coverage problems in typical residential deployments. Adopting different spectrum portions for different applications (licensed cellular frequencies, 5 GHz, 2.4 GHz or lower frequencies such as the TV white-spaces) can help in mitigating coexistence problems.

In addition, devices operating in the home belong to different actors:

- the user (i.e., the home owner), who wants to configure its monitoring and automation system according to personal preferences;
- the Internet service provider (ISP), that installs its gateway at the users' home for offering its services;
- other service providers (e.g., IP-TV, health-care service), that provide a service from outside the home to user devices;

- other utility providers (e.g., the electricity provider), that install smart meters at the users' home for performing remote readings.

Every actor may access a limited fraction of devices and, even when allowed to modify the device configuration, it has a limited control on the device positioning (which is ultimately decided by the users).

Finally, environment and interference conditions are time-varying, because they depend on the presence of users at home. Critical co-existence problems and capacity issues may occur at evening or during the holidays, when the peak traffic demand can significantly degrade overall system performance.

2.2 Smart City usage

Apart from cellular access networks, we assume that a smart city is equipped with a pervasive wireless infrastructure offering local connectivity. We can distinguish two situations: one in which a person walks, and one in which a person drives (or is driven in) a vehicle along the city streets.

Walking in places that are not crowded poses no particular challenges, however walking in very crowded environments such as subway stations, shopping streets, historical centres, etc. can result in low data rates, intermittent connections with the infrastructures, and severe inter-BAN interference. The overloaded network can, however, be optimised by applying a mix of frequency diversity, channel allocation, power control, etc. techniques. Additional services may include consumption of multimedia (including 3D visualizations) information about landmarks retrieved based on the position may be particularly relevant for tourists.

When driving a vehicle, the typical user has limited needs for data. However, when people are passengers in (self-driving) vehicles, they will likely behave similarly as they do on public transportation when commuting nowadays. This involves working possibly over a VPN connection to the corporate network, surfing the web, using social networking apps and watching (streaming) videos. Some of these activities may require a reliable data connection of relatively high bandwidth.

Cars driving in the same area can build a local ad-hoc network for the fast exchange of information in emergency situations or to help each other keep a safe distance. Such applications are called ITS (Intelligent Transport System) applications. In case of an accident, not only will the airbag be triggered, but the police and ambulance service will be informed via an emergency call to a service provider. Cars with this technology are already available. We distinguish different types of communication with vehicles:

- Vehicle-to-vehicle (V2V) communication: local ad-hoc network for direct communication with vehicles close-by to prevent accidents. Cars will also inform other cars about accidents via the ad-hoc network to help them slow down in time, even before a driver can recognize an accident
- Vehicle-to-Infrastructure (V2I) communication: interaction of vehicle with infrastructure for local information, e.g. intelligent speed adaptation (ISA) in the area of traffic lights
- Vehicle-to-Vulnerable Road User (V2VRU) communication: direct interaction between vehicle and vulnerable road user (pedestrian, cyclist), e.g. generating a warning to a truck driver when turning right on a cross-road in the presence a cyclist.

The smart city also has wireless sensors in trash-bins, traffic lights, light poles, parking spaces, etc. While these systems can be optimised geographically not to interfere with each other and mostly use lower frequencies to increase coverage, they still need to be actively managed and, due to their typical low duty cycle, can potentially be used as part of overall communication infrastructure.

2.3 Public Event usage

In case of public events, a dedicated wireless infrastructure (based on Wi-Fi technology or LTE-A in unlicensed bands) is often pre-installed in the hosting venues or public spaces.

Typical participants to the event want interact using social media (for example to tweet their experience, check what other friends share), upload photos and videos, and perhaps also check the (local) news. A class of users represented by the staff manning the venue or some special service staff need reliable wireless services for security or business applications. Emergency applications, which require to be served with the highest possible priority and reliability, may also take place at large public events.

In this context, the design of the wireless infrastructure become extremely complex because of the high-density of users and significant variability of capacity requirements (that can be strongly dependent on location and time). The high density of devices can create severe interference problems between neighbour BANs or BANs and environmental MTC/IoT technologies, which require some coordination strategies for selecting the channels, transmission powers and operating intervals of different network segments. The variability of the capacity demand can be faced by deploying a dynamic network infrastructure, in which access points can be switched on and off, can work on different bands, and can tune their coverage range according to the network status and performance. To this purpose a control network has to be deployed for monitoring and configuration purposes.

3 The WiSHFUL Concept

From the description of the previous scenarios, it is evident that the static planning and configuration of wireless infrastructures and devices (today still faced with intense manual work) can be significantly inefficient.

Moreover, performance optimization cannot be just a matter of the availability of advanced hardware platforms and radio capabilities (such as MIMO, beam-forming, spectrum agility, etc.), but mainly depends on the availability of suitable software platforms for controlling and coordinating radio communication and network protocols within the complex wireless ecosystems.

In this direction, the WiSHFUL project proposes an architecture for wireless systems devised to enable the definition of cognitive adaptations of radio operation and automated run-time network intelligence, by means of flexible and unified radio and network control. With flexible control we refer to the possibility to maximize the configuration space of the devices, exploiting all the radio functionalities and programmable protocol logics that is supported by the radio and platform hardware. With unified control we refer to the possibility to expose platform-independent programming interfaces over very heterogeneous hardware platforms, including standardized technologies and SDR platforms.

Figure 1 summarizes this vision. The figure provides a conceptual view of the blocks foreseen for the resulting architecture. Existing devices feature **radio driver software comprising of PHY and some low-level MAC functionality** and a **network stack comprising some higher-level MAC and upper layer functionality depending on the implementation**. In our vision, the radio and networking software can be extended with control functionality (i.e. radio and network control blocks in the figure) and the extension also exposes the corresponding configuration interfaces that can be used for interaction with both local and global (network-wide or cross-network) entities, such as intelligent control engines or application layer services. The intelligence engines are not shown in this figure (but will be introduced in section 3.3). The clean separation between radio control functions and network protocol functions is envisioned to break the current monolithic implementation of wireless stacks, thus drastically alleviating future radio driver and networking software development efforts, while preserving reliability and time accuracy constraints of radio and networking operations.

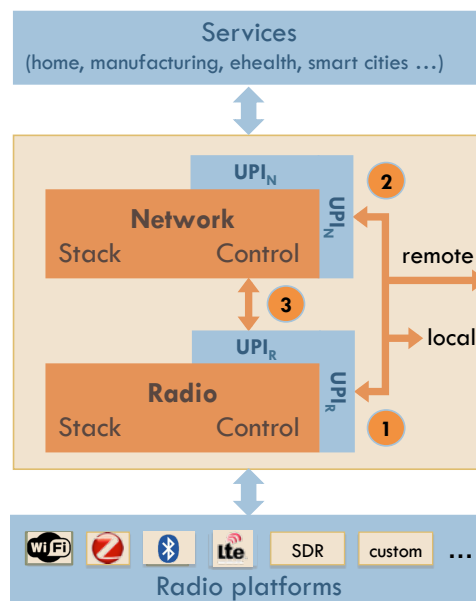


Figure 1 The WiSHFUL architecture.

3.1 Towards a unified radio control software architecture

3.1.1 High-level software architecture

WiSHFUL envisions a novel architecture for programmable wireless devices: while each protocol (or protocol release) supports different features and each vendor implements a specific hardware/software device architecture, there is an interesting common set of capabilities and functions that can be exploited for defining hardware-agnostic programs to be loaded on the wireless devices and to opportunistically change their behaviour. The common set of capabilities can be expressed by **identifying the primitive hardware components** (whose internal structure cannot be modified), which expose a **hardware abstraction layer (HAL)** in terms of input/output signals and configurable parameters.

For example, in SDR architectures the primitive components can be the analog radio-frequency frontend building blocks, including e.g. the power and low-noise amplifiers and DAC/ADC converters, while the mixers, filters, amplifiers, detectors, modulators, can be completely programmed (i.e. controlled) from scratch. This architecture supports the maximum level of flexibility being *constrained only by the available analog frontend, converters and digital processor*. It also implies a very high level of complexity, because the control program has to configure and control all the data processing functions (except for the RF operations).

In contrast to full SDR, WiSHFUL primitive components can be aggregated into more complex sub-systems such as:

1. the *transceiver*, dealing with the reception and the transmission of the frames, according to a set of available modulation and coding schemes;
2. the *transmission queues*, in which traffic flows (data frames) and control & management frames can be separately queued for achieving different medium access performance;
3. the *reception queue*, in which incoming packets can be stored before being forwarded to the host. Optionally, other sub-systems can be included, such as sensors for sampling different types of environmental data, etc.

Rather than being controlled by a given protocol, these sub-systems can be governed by the so-called *Radio Processor (RP)*. The core of the RP architecture is an *Execution Engine* able to read the output signals of the HALs (e.g. the arrival of a new packet from the host) and to react, according to a programmable logic, by sending inputs signals driving the different hardware sub-systems (e.g. transmitting a frame through the air). The reactions to the same output signals may vary according to the system state, which includes the configuration of the hardware and the logical state of the programmed protocol.

This approach allows WiSHFUL to hide the hardware internal details from the control programs and to support very different medium access rules and networking models by trading off flexibility and ease of programming according to the definition of the primitive components to be controlled. It is important to note here that WiSHFUL approach applies equally well to fixed-function radio devices as to SDRs.

Figure 2 shows the envisioned Radio Processor architecture with some examples of HALs. In the figure, the HAL of the radio chip is given by a list of parameters that can be set/read for specifying the transceiver configuration, a list of commands activating the transceiver functionalities and a list of signals automatically generated by the transceiver during its operation. As a common processor may differ in terms of supported instruction sets, similarly, different RPs could support different instructions according to the capabilities (i.e. HALs) of the hardware systems to be controlled. The complete set of HALs provided by the different sub-systems represent the Programming Interface that can be exploited for composing the programmable protocol logic. Such logic can be defined in a high-level programming language, such as a C-like code or a state machine and then compiled into an RP-specific machine code.

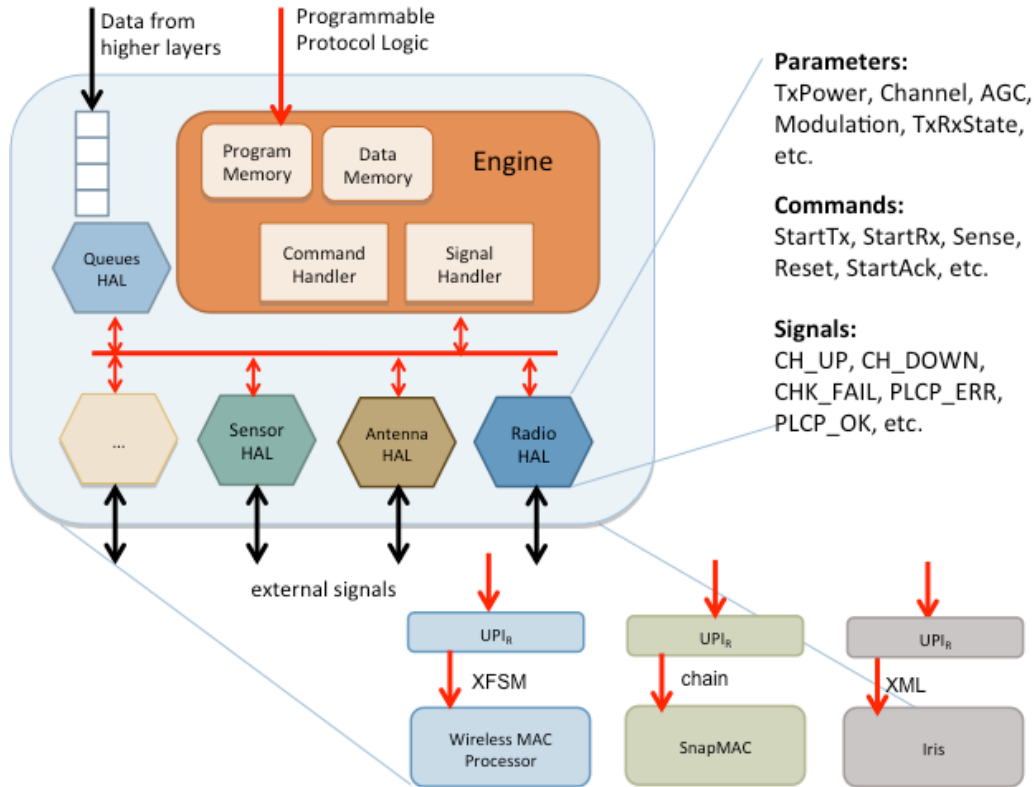


Figure 2: Radio Processor Architecture. The Programming Interface is given by the collection of available HALs.

In order to fully support our *Radio Control vision*, we propose an incremental strategy, according to which we will first work on the integration of the already available architectures in WiSHFUL by defining a **common Unified Programming Interface (UPI)**. Figure 2 depicts the role of the Unified Programming Interface (UPI_R) for integrating the Wireless MAC Processor, SnapMAC and Iris architectures in WiSHFUL. The idea is exposing a common programming model (based on the signals, commands and parameters available in the different hardware blocks) to be mapped onto architecture-specific interfaces and languages.

3.1.2 UPI_R - UPI for Radio Control

The UPI_R enables controllers (standard or intelligent, local, global and hierarchical) to retrieve information from a single radio or a set of radios and to control its/their behaviour. For instance, the controller is able to retrieve information from the lower layers such as binary spectrum occupancy value, raw I/Q samples, RSSI, LQI, duty cycle info and error statistics. The engine can also control the behaviour of the radio such as setting modulation and coding type, power levels, beam forming parameters, transmission timings, etc. Apart from the utilization of a simple parametric control model, we also envision the possibility to define more advanced configurations based on novel abstractions of the radio. In this case, the UPI_R allows programming the radio behaviour according to the supported abstractions and domain-specific programming languages. For example, the radio behaviour can be described in terms of state machines [1] or time-annotated chains of radio chains [2], which are programmed by specifying the hardware primitives to be called when hardware signals are detected in a given logical state.

We also envision the possibility to set-up multiple virtual radio interfaces over the same hardware, similarly to the coexistence of multiple virtual machines on the same computer: a different radio configuration and behaviour can be specified for each virtual interface, provided that the RP allows to manage conflicts due to multiple interfaces requiring simultaneous access to the hardware.

3.2 Towards a unified network control software architecture

3.2.1 High-level software architecture

Devices in the edge network traditionally embody a “vertical” network design, using customized network stacks that are tailored to a specific application domain. A high degree of device and technology diversity, has led to the creation of various development/build/deployment/run-time environments or tools for such devices.

As a consequence, numerous network protocols, stacks, architectures and frameworks have been implemented that in essence provide the same functionality but differ heavily in the required programming environment and tools. Another consequence of the “vertical” network design is that network stacks on devices in different deployments are incompatible with each other, which leads to degraded overall network performance.

Hence, application developers are confronted with complex software tools and often rely on specialized operating systems. Maintenance of such deployments also requires custom tools, which are complex, time-consuming and error-prone in usage. This significantly hampers the development of innovative wireless applications and advanced coordination schemes.

There is a clear need for a ***unified architecture that facilitates deployment and maintenance in the edge network and that incorporates special techniques for node-local and network-wide intelligent control and management***, for example co-existence awareness techniques that reduce interference and boost performance goals like increased capacity, enhanced mobility, increased reliability, low latency communication and low energy consumption. The main problem with the “vertical” network design is that there is no clean separation between control, data and management flows.

In WiSHFUL we envisage a network-wide control and management architecture that supports application development, runtime network control and network management. The architecture must provide a comprehensive solution that operates both at the network and the node level. The network control ***unified programming interface (UPI_N)*** is designed such that it facilitates easy development of novel applications, easy integration of network-wide control functions and easy maintenance of devices in real-life deployments.

The overall conceptual network control architecture is depicted in Figure 3. It can be seen that, similar as with radio control, there exists also a network stack control, which interacts with the upper MAC, network and transport layers. The network stack control provides the UPI_N that can be used by the controller (standard or intelligent; local, global and/or hierarchical) to optimize the performance of the network layers. This performance can be optimized in a distributed fashion by node-local implementations or in hierarchical fashion by interaction between global and local controllers over the UPI_{HC} interface (that will be explained in detail in section 3.3.2).

3.2.2 UPI_N - UPI for Network Control

The UPI_N enables controllers to retrieve and aggregate information from the different layers in the network stack of the wireless network nodes as well as controlling their behaviour. Examples of network information are the topology, the congestion level of links or paths, the selected routing algorithm, the selected MAC scheme, the available traffic classes and flow-level statistics. Network configurations enabled by UPI_N involve the set-up of logical associations/links among the nodes, mapping of traffic flows into queues with different priorities and specific radio configurations, flow control among multiple real/virtual radio interfaces, network coding of different traffic flows, etc. Also for network control, we envision the possibility to introduce behavioural abstractions devised to specify the network configuration in a compact domain-specific language. For example, the rules for forwarding the packets belonging to different traffic flows across different links and radio interfaces can be specified in terms of tables with conditions to be matched and relevant actions.

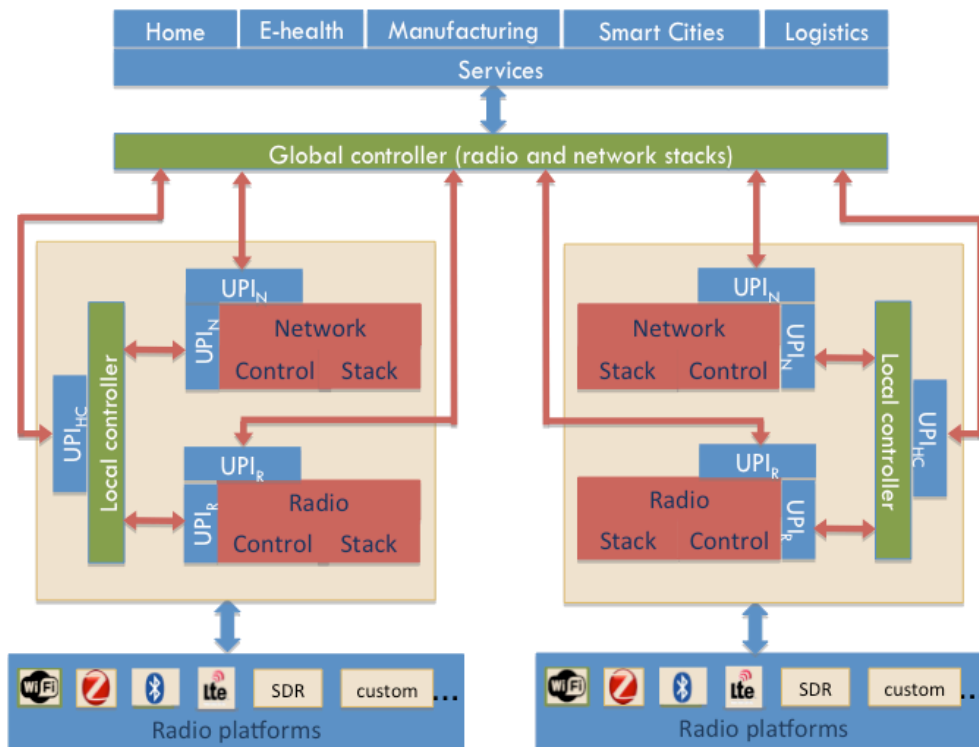


Figure 3: The overall network control architecture

3.3 Towards intelligent radio and network control

3.3.1 High-level software architecture

Intelligent radio and network control refers to increasing the degree of automation, and thus the adaptability of the wireless network in view of improving the overall performance. Instead of using fixed settings advised by data sheets or resulting from manual tuning (i.e. low and high threshold values of a hysteresis), the use of algorithms that are able to adapt to changing environments is referred to as intelligent control (i.e. replacing the hysteresis with an on-line linear regression or artificial neural network).

Within the WiSHFUL project, the aim is to **design and realize a software framework for supporting this type of intelligence** based on decomposing (cognitive) radio network operation into individual data processing tasks. The result is a general, extensible unit for supporting intelligence in wireless applications at either the radio level or the network level [3] [4]. The framework supports local or remote (possibly web based) configuration of the intelligence pipeline in a semi-automatic way where the human is only involved in the composition itself.

The framework provided for enabling the intelligence engine consists of three main components as depicted in Figure 4: Data Collection Component, Intelligence Composition Component and Action Component.

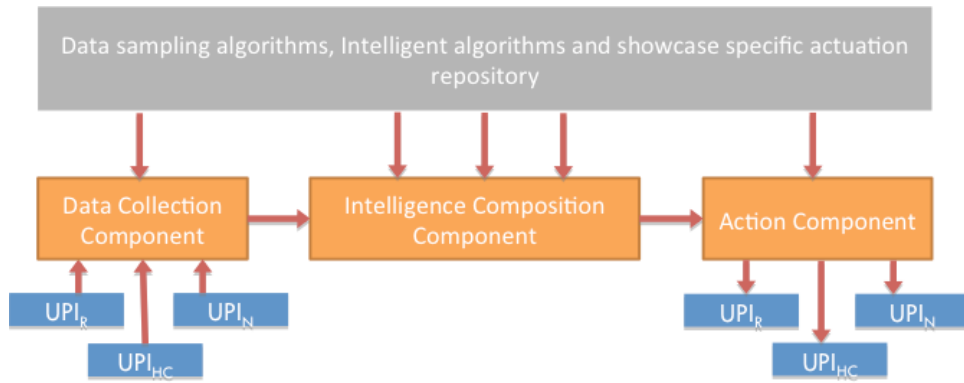


Figure 4 The framework for enabling intelligence in WiSHFUL.

The **Data Collection Component** is a generic software module that interacts with the WiSHFUL UPIs to retrieve data about radio and network stack operation (i.e. instant or aggregated channel occupancy, LQI, RSSI, PRR, etc.). The **Intelligence Composition Module** offers support for composing and configuring several algorithms available in the WiSHFUL Intelligent Algorithm repository into a self-contained intelligence engine that uses the data provided by the Data Collection Component and triggers configuration through the Action Component. The **Action Component** uses the WiSHFUL UPIs to adjust the communication configuration. A communications configuration should be viewed as the output of the intelligence process. Such a configuration can concern individual parameters (e.g. center frequency, back-off delay, etc.), radio processing elements (e.g. filter swapping), or waveform/protocol (e.g. new MAC scheme).

The software components implementing intelligence can be provided as open modules by the consortium, implemented by application developers or a mix of the two. In all cases, they interact with the radio and network stack control through the UPI_R and UPI_N respectively as depicted in Figure 5. The proposed architecture defines the framework for pipelining intelligence-related modules and provides reference implementations. It also supports hierarchical control where the intelligent modules can be structured on different levels. We refer to the collection of these modules as the Intelligent Algorithm Repository in Figure 4. The intelligence framework from WiSHFUL enables the composition of these into a complex engine that finally results in a run-time intelligent controller as depicted in Figure 5. It should be noted that both local and global controllers in Figure 5 include intelligence functionality when compared with respective controllers shown in Figure 3. This functionality consists of algorithms that exhibit a high degree of adaptability (i.e. optimization, learning, etc.).

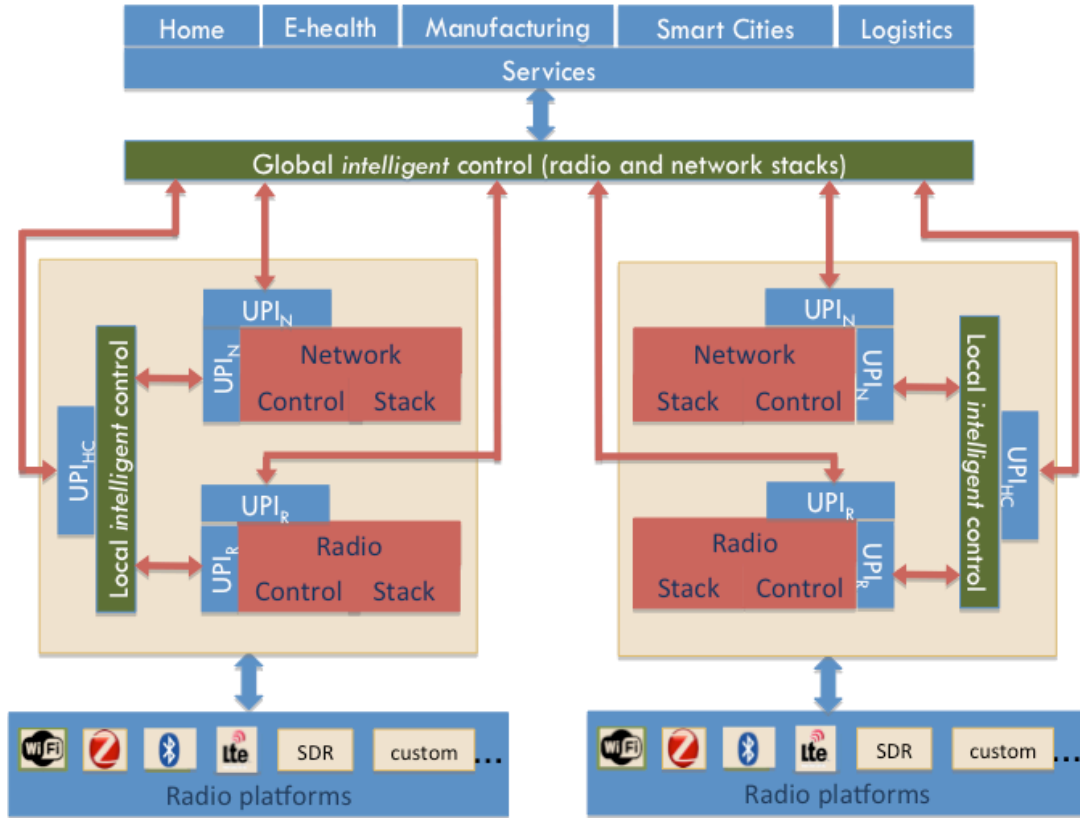


Figure 5: The overall WiSHFUL architecture including intelligent adaptation (global and local controls are intelligent as opposed to basic as in Figure 3)

3.3.2 UPI_{HC} - UPI for Intelligent Control

The UPI_R and UPI_N interfaces can be used by local and global controllers to gather node-local or network-wide information and to dynamically select the most optimal radio and network configurations. To support network-wide dynamic configurations, it is also required to enable some of the basic control services, such as synchronisation, monitoring and triggering of synchronised actions at multiple devices (e.g. for interference coordination). These are enabled through the UPI_{HC}, which enables basic or intelligent controllers to interact in a hierarchical fashion.

Intelligent engines can work on different models and scopes such as local, global or hierarchical (with some forms of central or distributed control). Moreover, they can work separately on the network stack and radio stack, or they can work on the whole device configuration. As an illustrative example, in the following we briefly describe how the typical problem of Wi-Fi and ZigBee coexistence can be faced under the three different optimization scenarios.

Case 1.

Assume that WiSHFUL cognitive network stacks, exposing the UPI_N interface, are deployed on traditional off-the shelf Wi-Fi devices. In the case where Wi-Fi devices interfere with low-power ZigBee nodes and no orthogonal channel is available, the intelligent engine can configure the Wi-Fi legacy MAC for working with a portion of the Wi-Fi beacon interval in contention-free mode. This configuration prevents Wi-Fi nodes from periodically contending for the channel, thus leaving some channel resources to lower-power ZigBee nodes.

Case 2.

Assume that traditional network stacks are deployed on a WiSHFUL cognitive radio, able to implement the Wi-Fi and ZigBee transceiver and to expose the UPI_R interface. In the case where Wi-Fi nodes are involved in a low-rate traffic flow, the intelligent engine can reduce the operation

bandwidth of Wi-Fi nodes (e.g. by down-clocking the OFDM modulator) in order to avoid interference with the overlapping ZigBee channel.

Case 3.

In the deployment case where all devices are based on a full WiSHFUL cognitive stack, exposing both UPI_N and UPI_R interfaces, more advanced optimizations are possible. The intelligent engine can configure all the devices (either the ones implementing the Wi-Fi transceiver and the ones implementing the ZigBee one) for recognizing a common synchronization signal and employing a cross-technology TDMA (as demonstrated in [5]). Since different technologies work on independent time intervals, the same cognitive radio can support two virtual radio interfaces and switch from one mode to another according to the traffic type. This requires an opportunistic configuration of the radio and lower (i.e. time critical) MAC operations, as well as a configuration of the flows classification and forwarding policies.

3.4 Flexibility Requirements

In order to design the WiSHFUL software components for different radio platforms, we started from the analysis of the flexibility requirements emerging from the driving scenarios.

The requirements have been categorized into three groups, which refer to configuration options desirable for wireless infrastructures, end user devices, and SDR platforms.

Wireless Infrastructures. The following, non-exhaustive, set of requirements have been identified for improving the coverage and the capacity of wireless infrastructures.

- Activate more Access Points (APs) while lowering the transmit power of the already active ones (i.e. increase capacity in the space domain).
- Switch to a higher EM frequency, increasing the capacity while lowering the coverage of a single AP (i.e. increase capacity in the space domain: more but smaller cells).
- Increase the number of EM frequencies that are used (i.e. increase capacity in the frequency domain).
- Avoid interference by optimizing the frequency selection so that the APs within the same collision domain use different EM frequencies.
- Downlink optimization: by prioritizing certain application streams the capacity of high priority traffic can be ensured, leaving low-priority traffic for opportunistic delivery (i.e. best effort delivery).

Users' end devices. This set of requirements has been identified for improving the utilization of the wireless resources and coordinating multiple technologies available on the user side. Examples of these requirements are:

- Traffic shaping on the users' end devices allows a more fine-grained planning of the required bandwidth (i.e. demand side control).
- Dynamic frame aggregation allows increasing the number of aggregated frames when the channel is stable. Frame aggregation reduces the protocol overhead and increases the goodput.
- Use a cross-technology TDMA protocol to increase the capacity in the time domain by coordinating the transmissions between different technologies within the same collision domain.
- Incorporate seamless handover to reduce control traffic necessary for roaming from one technology to another. Hard hand-overs (Wi-Fi = break before make) can be replaced with soft hand-overs (UMTS = make before break).
- Based on the type of application stream the most optimal wireless technology can be selected (i.e. automatic technology selection).

SDR platforms. Further optimisations enabled by SDRs are still possible, when backwards compatibility with existing technologies is not required:

- By re-aligning the centre frequencies of each wireless channel more channels can be co-located in the same ISM band.
- By adjusting (e.g. decreasing) the guard bands between wireless channels, more of them can be squeezed into the same ISM band, OFDM and Filter Bank Multi-Carrier (FBMC) providing an upper limit in terms of spectral density
- By dynamically selecting the (de)modulation techniques based on the channel quality, the optimal data rate can be obtained for a given channel quality. Combining this with OFDM, yields DMT, a technique well known in xDSL systems, with opportunities in the wireless domain

4 Requirements for UPI_R

Radio configuration is responsible for the set-up of physical wireless links between the nodes, whose capacity is affected by propagation and interference conditions, bandwidth allocations, and medium access schemes. It can be expressed in terms of a simple parametric model (e.g. channel, bandwidth, power, antenna configuration, modulation/coding format, etc.) and/or in terms of novel abstraction models (e.g. XFSMs) for specifying the low-level MAC rules to access the medium. The interface is also responsible for collecting statistics about the radio performance and local views.

In order to identify the requirements for this interface, we analyse some use cases referring to the previously introduced driving scenarios.

4.1 Use cases

4.1.1 Coexistence of heterogeneous technologies

This use case deals with the dynamic reprogramming of nodes belonging to different networks, which coexist in the same physical environment (such as the wireless local area networks and sensor networks that can coexist in the **home scenario**). Because of overcrowding in the ISM bands, a simple planning of different operating channels can be inadequate to avoid inter-network interference. Moreover, performance may degrade significantly if nodes belonging to different networks follow a non-uniform approach for detecting or reacting to the presence of other nodes and technologies.

In such a scenario, a solution can be to support a more granular spectrum allocation policy, i.e. *allocating a given channel resource to each network with network-specific access rules*, such as the possibility to access the spectrum only in periodically allocated time intervals. When an interfering source cannot be controlled (e.g. a network with non-programmable nodes or a micro-wave oven), the medium access protocol of the programmable devices can be tailored to the estimated interference pattern for minimizing channel wastes due to collisions.

Consider for example the scenario illustrated in Figure 6: two wireless computer networks, network 1 and network 2 (employing programmable interfaces), coexist in the same environment and interfere with a micro-wave oven (network 1) or with low-power wireless sensors (network 2), e.g. smart plugs deployed for monitoring energy consumption at home. For the first network, the interference pattern cannot be controlled, because it depends on the power activity of the microwave oven (that is based on periodic on/off cycles during operation). For the second network, we consider two cases in which the sensor nodes can be either programmed or not.

The monitoring functionalities available on each programmable device are able to detect and characterize the inter-technology interference by collecting radio statistics about receiver errors, power measurements, channel occupancy intervals, etc. The statistics can be processed locally and/or by the central controller to have a unified network view. On the basis of these statistics, different coordination actions can be taken by the central or local intelligence.

For example, the central intelligence can enforce the computer nodes belonging to network 2 to access the channel only during a portion of the beacon interval, and the interfering sensor nodes to use the remaining portion. In case the sensor nodes are not programmable, the inactivity intervals of network 2 can be programmed as a complementary function of the estimates of the sensors activities. The same central coordinator can configure network 1 on an independent channel.

When the microwave oven is switched on, nodes belonging to network 1 can independently characterize the activity intervals of the oven and decide to avoid packet transmissions whose transmission time will overlap with this predictable oven interference. To this purpose, local intelligence is programmed to detect predictable interfering patterns and switch to the interference-avoiding protocol in case of interference detection.

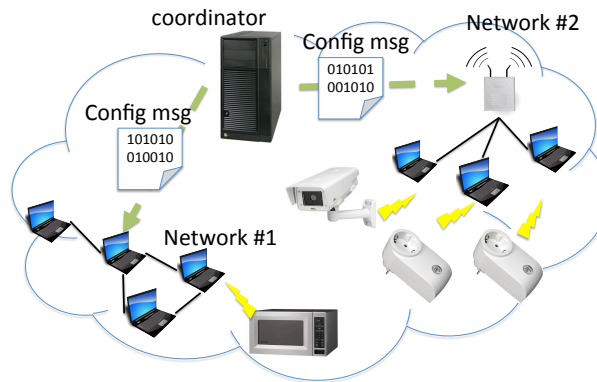


Figure 6 – Home network scenario: two WLANs coexist in the same environment, interfering with a microwave oven (network 1) and low-power sensors (network 2).

4.1.2 Intelligent download use case

We consider a use case with different simultaneous (and high-density) data traffic flows of different types (as in the case of the **public event** scenario). Specifically video streams are the source of major traffic load for the networking infrastructure. While some of the video streams will be real-time (e.g. videoconferencing or real time surveillance in case of alarms), the largest part will be just bulk downloads or uploads, which can tolerate delays.

Traffic flows can be easily categorized for using different transmission queues, which are mapped into different radio configurations according to the specific application requirements. Specifically, the networked radio control is needed to decide when (i.e. scheduling in time) and with which capacity (i.e. assigning of radio resources) the individual flows, corresponding to individual real-time sessions and/or bulk downloads, should take place, and which radio parameters (modulation, coding, frequency, etc.) should be used for each of the transmissions.

Due to the shared wireless medium a proper solution requires complete knowledge of the set of tasks (flows) and an overview of the entire radio environment (e.g. channel load, interference). The resulting radio configuration settings (e.g. individual modulation/coding settings) for each transmission are enforced by the local radio control.

An example illustrating the control flow of the envisioned intelligent download scenario is given in Figure 7. This figure shows the hierarchical control of the system whereas the networked radio controller instructs the local radio controller to execute control tasks, i.e. enforce a radio reconfiguration policy for each flow.

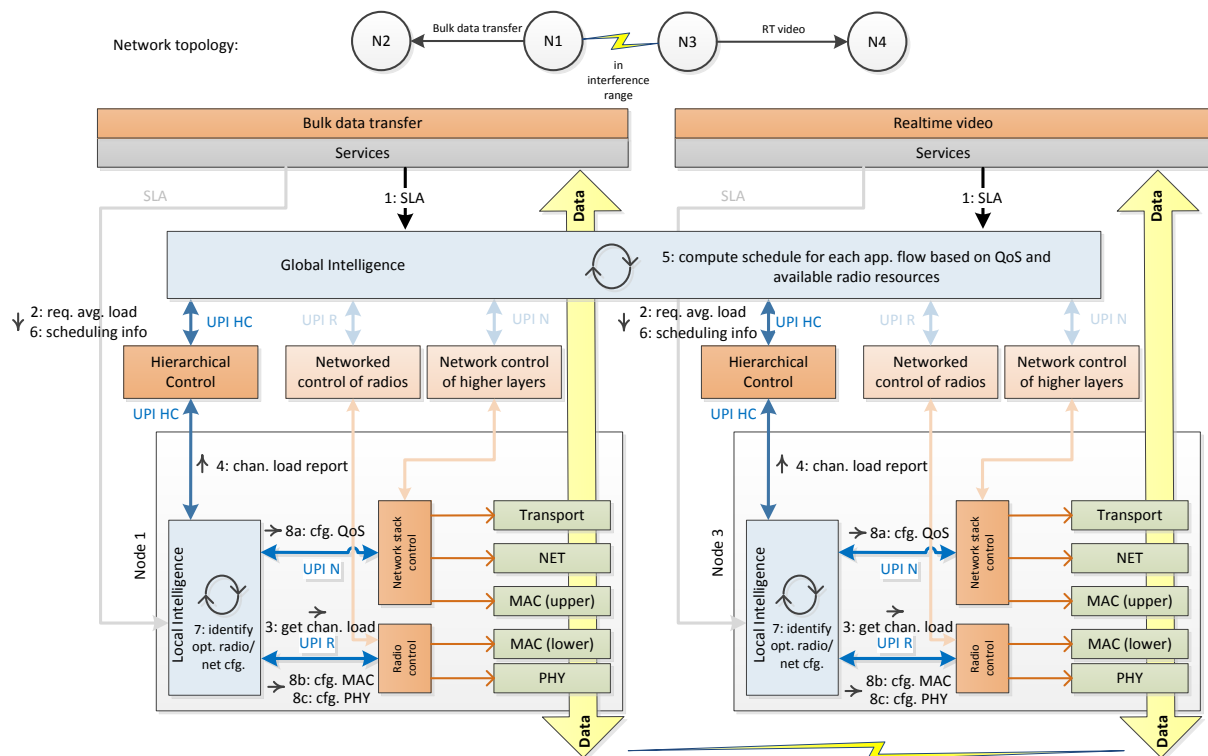


Figure 7 – Radio reconfiguration according to a global network view

4.2 Functional requirements

The UPI_R interface represents the interface towards the node's *primitive components* such as the transceiver (or, alternatively, mixers, filters and SDR execution platforms), the antenna system, the hardware queues, the radio processor, and optionally the sensors. Through this interface, it is possible to read the signals generated by these components and configure the different sub-systems according to their programming model. Different functionalities can be supported according to the primitive components available in the nodes. For example, standard technologies expose pre-configured transceivers and protocols in which only some parameters can be configured, while the Wireless MAC Processor or SnapMAC based nodes expose an internal radio processor for changing the medium access logic.

In general, the UPI_R interface is responsible for the following **operational aspects** of the radio system:

- **Spectrum allocation:** deals with the configuration of the spectrum portion (operating channel, bandwidth, multiple bands or sub-bands, etc.) that can be used by the node;
- **Wireless link set-up:** apart from the available spectrum, wireless links are affected by propagation and interference conditions. It is therefore required to configure the transmission power, the antenna, the transmission format and error protection scheme, on which the radio connectivity critically depends. It is also possible to distinguish between physical connectivity (i.e. wireless links over which a transmission capacity is available) and logical links (i.e. physical links enabled for transmissions).
- **Radio monitoring:** various statistics can be collected from the different radio components, including spectrum occupancy, channel busy intervals, receiver errors, received power and interference levels, the energy consumption, etc.
- **Medium access logic:** in case a radio processor is available, the wireless link performance strongly depends on the rules for timely controlling the access to the shared medium, the acknowledgement policies, and the unicast/multicast delivery modes.

- **Radio virtualization:** on top of the same primitive components, different radio configurations can be configured by exploiting virtualization functionalities. This requires to partition and isolate different hardware resources, and to solve potential access/configuration conflicts.

For each operational aspect, we specify the requirements summarized into the following tables.

| <i>Spectrum allocation</i> | | | | |
|-----------------------------------|---|--|-----------------|-------------------|
| UC# | Actors | UC and short description | Priority | Complexity |
| 4.1 | Local Controller Physical layer | Get spectral capabilities of the RF hardware | Medium | Low |
| | | The local controller must be able to obtain the spectral capabilities of the RF hardware such as: supported bandwidth, supported frequency range(s)... | | |
| 4.2 | Local Controller Lower MAC | Get supported channel configurations | Medium | Low |
| | | The local controller must be able to obtain the channel configurations supported by the system, including channel bonding, subcarrier selection... | | |
| 4.3 | Global Controller Lower MAC | Restrict spectrum usage | Low | Medium |
| | | The global controller must be able to configure the spectrum that can be used by a node | | |
| 4.4 | Local Controller Lower MAC / Physical layer | Change RX and TX spectrum configurations | High | Medium |
| | | The local controller must be able to control the spectrum used for sending and receiving data choosing from the configurations supported according to 3.1 and 3.2 and within the restrictions imposed by 3.3 | | |

| <i>Wireless link set-up</i> | | | | |
|------------------------------------|---------------------------------|---|-----------------|-------------------|
| UC# | Actors | UC and short description | Priority | Complexity |
| 4.5 | Local Controller Transceiver | Get supported link configurations | Medium | Low |
| | | The local controller must be able to obtain the supported link parameters of the transceiver such as TX power, RX gain... | | |
| 4.6 | Local Controller Antenna | Get supported antenna configurations | Low | Low |
| | | The local controller must be able to obtain the supported antenna configurations | | |
| 4.7 | Local Controller Baseband | Get supported baseband functionality | Medium | Low |
| | | The local controller must be able to obtain the supported configurations of the baseband (or | | |

| | | | | |
|------|---------------------------------------|--|--------|--------|
| | | equivalent) such as modulation scheme, coding scheme... | | |
| 4.8 | Local Controller Physical layer | Get MIMO capabilities | Low | Medium |
| | | The local controller must be able to determine the MIMO capabilities of the whole RF system (diversity, beamforming...) | | |
| 4.9 | Local Controller Physical layer | Configure link parameters | Medium | High |
| | | The local controller must be able to control and modify the parameters related to the different RF subsystems as specified in 3.5 – 3.8 | | |
| 4.10 | Global Controller Local Controller | Query and restrict link parameter boundaries | Medium | High |
| | | The global controller must be able to obtain the relevant link information from the local controller and impose restrictions on the used link configurations | | |

| Radio Monitoring | | | | |
|-------------------------|------------------------------------|---|-----------------|-------------------|
| UC# | Actors | UC and short description | Priority | Complexity |
| 4.11 | Local Controller Physical layer | Get raw RF measurements | Medium | Medium |
| | | The local controller must be able to obtain raw RF measurements as reported by the underlying hardware. Depending on the hardware these can be I/Q samples, RSSI values, LQI/LSI estimates, channel occupation, or noise level. | | |
| 4.12 | Local Controller Physical layer | Trigger spectrum scanning | Medium | Medium |
| | | The local controller must be able to trigger spectrum scanning and to get results (e.g. received PHY frames on each scanned channel) | | |
| 4.13 | Local Controller Lower MAC | Sniff and inject raw PHY frames | Medium | Medium |
| | | The local controller must be able to get a copy of each received physical layer frame. Moreover it needs a way to inject externally created PHY frames into the wireless network device. | | |
| 4.14 | Local Controller Lower MAC | Get channel measurements and tune measurement parameters | Medium | High |
| | | The local controller must be able to obtain MAC measurements such as error statistics, channel occupancy, etc., as well as control the parameters of these measurements such as the energy detection threshold | | |

| Medium Access Logic | | | | |
|------------------------------|--|--|----------|------------|
| UC# | Actors | UC and short description | Priority | Complexity |
| 4.15 | Local Controller Upper MAC | Control framing | Medium | Medium |
| | | The local controller must be able to control the segmentation and aggregation of packets, as well as the definition of customized header fields | | |
| 4.16 | Local Controller Lower MAC | Handshake mechanism | Medium | Medium |
| | | The local controller must be able to define the sequence of control packets to be sent in each channel access (e.g. 2-way or 4-way handshake, 2-way handshake with reverse link, etc.) and the control information to be included in RTS/CTS frames. | | |
| 4.17 | Local Controller Lower MAC | Carrier sense mechanism | High | High |
| | | The local controller must be able to modify the collision avoidance policy used by the MAC logic such as the mechanism used (energy-based, preamble-based, virtual...) and the classification method (binary, multi-level). | | |
| 4.18 | Local Controller Lower MAC | Definition of access times | High | High |
| | | The local controller must be able to modify the timing parameters of the MAC logic. This includes but is not limited to: contention window selection, back-off freezing, inter-frame space, multi-node synchronization... | | |
| 4.19 | Local Controller Lower MAC | Definition of ACK policy | Medium | Medium |
| | | The local controller must be able to control the ACK policy used by the MAC logic. This includes artificial dropping, accurate time-stamping, handshaking, frame configuration... | | |
| Virtualization functionality | | | | |
| UC# | Actors | UC and short description | Priority | Complexity |
| 4.20 | Global Controller Local Controller | Definition of partitioning rules | Low | High |
| | | The controller has to allocate the hardware resources for each instantiated virtual interface | | |
| 4.21 | Global Controller Virtual Radio Manager | Definition of access priorities | Low | High |
| | | The controllers must be able to define the hardware access priority of the different virtualized interfaces. | | |
| 4.22 | Local Controller | Conflict resolution | Medium | High |

| | | | | |
|--|-----------------------|---|--|--|
| | Virtual Radio Manager | The virtual radio manager should be able to handle conflicts and notify the local controller when they arise. | | |
|--|-----------------------|---|--|--|

5 Requirements for UPI_N

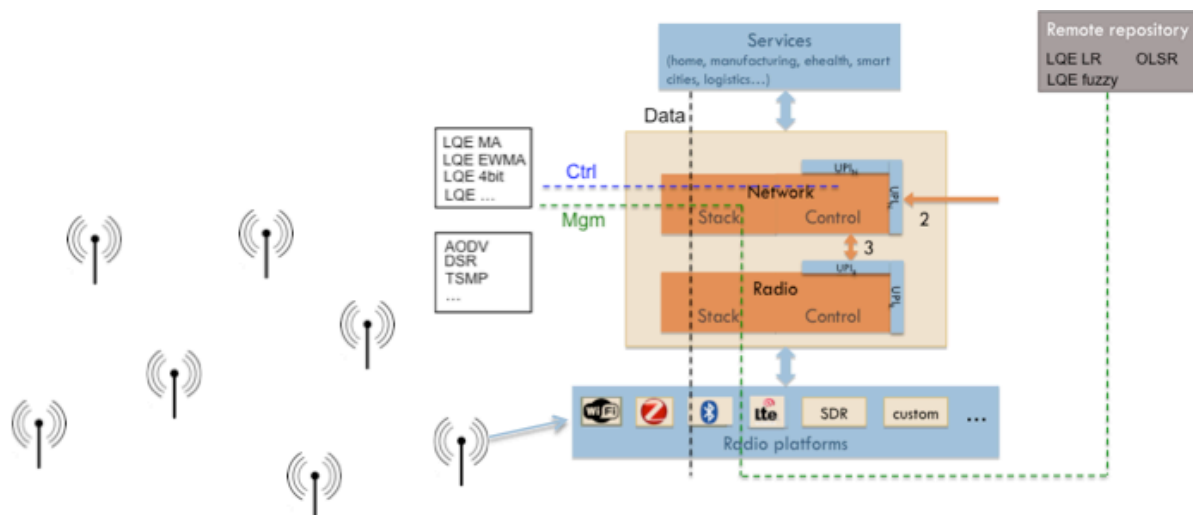
Network configuration is expressed in terms of logical associations/links among the nodes, mapping of traffic flows into queues with different priorities and specific radio configurations, flow control among multiple real/virtual radio interfaces, network coding of different traffic flows, etc. and other network abstractions to be defined within the project.

Note that, differently from network programming in wired networks, where device configuration is basically given by a processing table specifying the rules for forwarding packets, in the wireless domain network programming is more complex, because the physical links between the nodes are not deployed a priori, but depend on radio configuration (modulation, transmission power, antenna port, etc.), mobility and interference.

5.1 Intelligent download use case

If we zoom in on an intelligent download use case as a particular situation in the public event scenario, it can be noticed that the data rates vary, the network density varies and the topology slowly changes due to mobility. The challenge regarding the intelligent handling of application, management or configuration data download consists of enabling downloads at a specified quality of service under all possible conditions. This typically involves 1) adapting/changing the link quality estimator, 2) adapting/changing the neighbourhood management policy and 3) adapting/changing the routing protocol.

Figure 8 presents aspects of how the UPI_R can enable intelligent download scenarios on the data, control and management planes, setting the ground for defining the functional requirements for the overall network control of the WiSHFUL architecture. The figure presents a number of wireless nodes implementing WiSHFUL technology. The data plane (black dashed line) represents the application data flow to/from the terminal. The performance of this plane varies over time and space, thus the Network control through an intelligence engine interacting with it through the UPI_N (arrow labelled with 2) has to constantly improve the performance in terms of end-to-end reliability, latency and throughput. The control plane is illustrated with blue dashed line in the figure. The most straightforward way to achieve this is by tuning parameters such as the time window over which the link quality estimators (LQE) can be computed. The low level parameters required for computing LQEs are obtained from the radio through the UPI_R interface (labelled with 3 in the figure). Additionally, the LQE estimation algorithm can be changed with another one that is readily available on the node. When new software is needed to improve the performance of the network, the nodes can download the new functionality (typically from a remote repository) through the management network (green dashed line in Figure 8) from some remote repository.

Figure 8 Intelligent download enabled by UPI_N

For such as use case, the requirements for retrieving network-relevant settings and metrics:

- Retrieve higher-level MAC information (e.g. get station capabilities, beacon information elements)
- Retrieve routing and link information (e.g. link layer, address settings, routing tables, neighbour cache, routing rules, queuing discipline settings, traffic filters and classes used with queues)
- Network monitoring interfaces (connection tracking)
- Network monitoring (information (e.g. TCP/IP state, socket types, ports))
- Retrieve derived network performance metrics (e.g. LQE, PLR, Latency, etc.)
- Retrieve size of the aggregation time window (i.e. time window over which network statistics are computed)
- Retrieve list of local modules implementing higher-level MAC logic (e.g. management functionality in IEEE 802.11)
- Retrieve list of local modules implementing routing algorithms
- Retrieve list of local modules implementing LQE algorithm
- Retrieve list of local modules implementing neighbour table construction logic

Requirements for setting network-relevant settings and metrics:

- Set values in higher-level MAC (e.g. configuring/setting beacon information elements)
- Set routing and link configuration (new/set/del link, new/del address, new/del route, new/del neighbour, new/del route, new/del queuing discipline settings, new/del traffic filters and classes used with queues)
- Network configuration (e.g. set buffer size)
- Network configuration (e.g. set TCP/IP window size)
- Set the size of the aggregation time window (time window over which network statistics are computed)
- Select the set of modules to be used for implementing network functionality (i.e. CSMA MAC with OLSR routing and default LQE)
- Replace a module with another (i.e. CSMA MAC with TDMA MAC)

Requirements for updating network modules:

- Add new implementation of higher-level MAC logic
- Add new implementation of networking layer
- Add new implementation of routing algorithm
- Add new implementation of transport layer algorithm

- Add new implementation of LQE algorithm
- Add new implementation of neighbour table construction logic

5.2 Functional requirements

| Functional Requirements for unified control using UPI _N . | | | | |
|--|--|---|----------|------------|
| UC# | Actors | UC name and short description | Priority | Complexity |
| 5.1 | Local Controller Application/ transport layer | Get data flow QoS requirements | High | Low |
| | | The local controller must be able to obtain the QoS requirements of each active data flow | | |
| 5.2 | Local Controller Application/ transport layer | Assign QoS tag to data flows | Medium | Medium |
| | | The local controller must be able to tag data flows with QoS information in order to allow application agnostic monitoring of data flows. | | |
| 5.3 | Local Controller Application/ transport layer | Monitor QoS level of each data flow | High | Medium |
| | | The local controller must be able to monitor the actual QoS level for each active data flow | | |
| 5.4 | Local Controller Application/ transport layer | Restrict data flow QoS level | Low | High |
| | | The local controller must be able to restrict the QoS level of certain data flows | | |
| 5.5 | Local Controller Network/ routing layer | Monitor QoS level of each network layer data path | High | Low |
| | | The local controller must be able to monitor the QoS level for all network layer data paths used by the node in a given time window | | |
| 5.6 | Local Controller Network/ routing layer | Network layer configuration change | High | Low |
| | | The local controller must be able to adjust network layer configuration settings in order to achieve the desired QoS for each data path. | | |
| 5.7 | Local Controller Coordinator Network/ routing layer | Coordinated network layer configuration change | Medium | High |
| | | The local controller must be able to adjust network layer configuration settings in a coordinated fashion (e.g. synchronized configuration change on multiple nodes). | | |
| 5.8 | Local Controller Network/ routing layer | Obtain network layer statistics | Medium | Low |
| | | The local controller must be able to retrieve network layer statistics. | | |
| 5.9 | Local Controller Upper MAC | Monitor QoS level of each data link | High | Low |
| | | The local controller must be able to monitor the QoS level for each link layer data link. | | |

| | | | | |
|------|---|--|--------|--------|
| 5.10 | Local Controller Upper MAC | Adjust data link QoS level | High | Low |
| | | The local controller must be able to adjust the QoS level of a link layer data link (e.g. demand higher priority or guaranteed data rate, etc.) between a given minimum and maximum QoS level. | | |
| 5.11 | Local Controller Upper MAC | Link layer configuration change | High | Medium |
| | | The local controller must be able to adjust link layer configuration settings in order to achieve the desired QoS for the data link. | | |
| 5.12 | Local Controller Upper MAC | Coordinated link layer configuration change | Medium | High |
| | | The local controller must be able to adjust link layer configuration settings in a coordinated fashion (e.g. synchronized configuration change on multiple nodes). | | |
| 5.13 | Local Controller Upper MAC | Obtain link layer statistics | Medium | Low |
| | | The local controller must be able to retrieve link layer statistics. | | |
| 5.14 | Global Controller Local Controller | Obtain required QoS level for each node | High | Low |
| | | The global controller must be able to obtain the QoS requirements for each node in the network. | | |
| 5.15 | Global Controller Local Controller | Install QoS policies | High | Medium |
| | | The global controller must be able to install QoS policies on each node. | | |
| 5.16 | Global Controller Local Controller | Install QoS policy enforcement rule | High | High |
| | | The global controller must be able to install rules that allow the local controllers to enforce QoS policies. | | |
| 5.17 | Global Controller Local Controller | Get aggregated statistics | Low | Low |
| | | The global controller must be able to retrieve aggregated statistics from the local controller. | | |
| 5.18 | Global Controller Local Controller | Monitor network-wide QoS level | High | Medium |
| | | The global controller must be able to monitor the QoS level on a network-wide scale via the QoS tags or by querying the local controllers. | | |
| 5.19 | Global Controller Coordinator Local Controller | Coordinated configuration change. | High | High |
| | | The global controller must be able to adjust configuration settings in a coordinated fashion (e.g. synchronized configuration change on multiple nodes). | | |
| 5.20 | Global | Inter-network policy negotiation and | High | High |

| | | | | |
|------|---|---|--|--|
| | Controller External global controller | coordination | | |
| | | The global controller must be able to negotiate QoS policies with external global controllers responsible other networks. | | |
| 5.21 | Global Controller Local Controller | MAC logic update | | |
| | | Both the global and local controller must be able to update the MAC logic used by the nodes that support the advanced MAC configuration | | |

6 Requirements for enabling intelligence

Intelligence generally refers to the capability of intelligent agents to process information (i.e. stimuli) and make decisions. Artificial intelligence aims to recreate biological intelligence in machines enabling them to automatically process information and perform cognitive functions such as reasoning and learning. This form of intelligence is of great interest to the development of wireless systems for enabling autonomy in both radios and networks.

Within the WiSHFUL project, the aim is to design and realize a software framework for supporting intelligence based on decomposing overall intelligence behaviour into individual tasks. The result of such an approach is a general, extensible unit for supporting intelligence in wireless applications at either the radio level or the network level in line with recent developments in the area of cognitive communications [3][4]. Furthermore, through the introduction of an inter-tier interface, which connects radio level intelligence to network level intelligence, this framework enables hybrid centralized/decentralized intelligence schemes. The framework supports local or remote (possibly web based) configuration of the intelligence pipeline in a semi-automatic way where the human is only involved in the composition itself.

The WiSHFUL framework for intelligence consists of three main components: Data Collection Component, Intelligence Composition Component, and Action Component. The Data Collection Component is a generic software module that interacts with WiSHFUL UPIs to retrieve data from the radio and/or network and pre-processes that data. The Intelligence Composition Component supports the composition and configuration of intelligence algorithms, either provided from the WiSHFUL Intelligent Algorithm Repository or by the user, while providing connection to the other components. The Action Component uses the WiSHFUL UPIs to actualize the decisions from the Intelligence Composition Component.

6.1 Intelligent download use case

Enabling intelligence requires functionality beyond re-configuration of the radio (PHY, lower MAC) and network (upper MAC, NET, Transport, Application) stacks. Take the intelligent download scenario using IEEE 802.11 as an example. In an initial state, we start with standard CSMA/CA and few traffic flows resulting in a small amount of interference. Consider the appearance of a new flow with especially demanding requirements such that CSMA is unable to meet the demanded QoS. In this case a decision must be made on how to address the needs of the new form. For example, we may decide to switch TDM access mode. However, it is important to note that this switching decision is not sufficient on its own; it is not sufficient to switch to a TDM FSM in empty state because each station must determine an appropriate time slot assignment from scratch, which results in outage and further degrading QoS. It is the responsibility of the intelligence to provide a seamless switching from one MAC FSM to another. In particular we have to initialize the state of the newly instantiated TDM MAC with some initial state, like preconfigured station and slot assignment, which is further optimized at runtime. Thus, intelligence requires functionality to select appropriate new configurations, as well as the determination of supporting parameters to allow the new configurations to serve the intended purpose.

Data Collection Requirements are:

- Access to network monitoring (QoS measures, node health, etc.)
- Access to radio monitoring (BER, SNR, etc.)
- Access to network protocols (identification of routes or currently employed protocols)
- Access to radio data (sensing data, acknowledgements, channel gain measurements, etc.)
- Ability to indicate events of interest on radio level (packet transmission/reception, sensing completion) and network level (link usage, route traversal)
- Ability to indicate occurrence of events
- Structure (publish/subscribe, message polling, etc.) for passing perception data
- Ability to allow users to define data pre-processing tasks

- Efficient processing of time series data (aggregation, filtering, etc.)

Intelligence Composition Requirements are:

- Method for receiving data from Data Collection Component
- Method for delivering commands to Action Component
- Access to processor (general purpose processor (GPP), digital signal processor (DSP), graphical processing unit (GPU), etc.)
- Access to supporting libraries
- Method for storing data
- Method for accessing data
- Method for logging data for later export
- Ability to define interaction of intelligence algorithms (e.g. flow-graph design similar to GNU Radio or Labview)
- Access to Intelligent Algorithm Repository

Action Requirements are:

- Access to UPI_N
- Access to UPI_R
- Ability to allow users to define mapping between algorithm commands and actions to be taken
- Determination of time required for each action (for feedback to algorithms)
- Coordinated (synchronized) execution of UPI_R and UPI_N commands, i.e. a command to be executed on each node at the same time

UPI_{HC} Requirements are:

- Flexible and easy-to-use interface between controllers (inter-controller communication)
- Access to injection of information to radios (over the air control cases)
- Ability to allow user defined message structure
- Reliable synchronized message multicast, i.e. control message is received at the same time by the local controllers
-

6.2 Functional requirements

| Functional Requirements for intelligence support | | | | |
|--|---|---|----------|------------|
| UC# | Actors | UC name and short description | Priority | Complexity |
| 6.1 | Data Collection Component Application/transport layer | Get data flow QoS requirements | High | Low |
| | | The Data Collection Component must be able to obtain the QoS requirements of each active data flow | | |
| 6.2 | Data Collection Component Controller | Get parameter settings | High | Low |
| | | The Data Collection Component must be able to determine the current parameter settings for both radio and network elements. | | |
| 6.3 | Data Collection Component Application/transport layer | Monitor QoS level of each data flow | High | Medium |
| | | The Data Collection Component must be able to monitor the actual QoS level for each active data flow | | |
| 6.4 | Data Collection | Identify events | High | High |

| | | | | |
|------|------------------------------------|--|------|------|
| | Component | The Data Collection Component must be able to identify events of interest (as defined by user) to the intelligence algorithms. | | |
| 6.5 | Data Collection Component | Package data for intelligence use | High | Low |
| | | The Data Collection Component must be able to package data in a manner usable by the intelligence algorithms. | | |
| 6.6 | Intelligence Composition Component | Run intelligence algorithms | High | Low |
| | | The Intelligence Composition Component must be able to run intelligence algorithms (from the Intelligent Algorithm Repository) | | |
| 6.7 | Intelligence Composition Component | Compose intelligence algorithms | High | High |
| | | The Intelligence Composition Component must allow users to define the order of intelligence algorithm execution and the flow of data between intelligence algorithms | | |
| 6.8 | Intelligence Composition Component | Issue commands to the Action Component | High | Low |
| | | The Intelligence Composition Component must be able to issue user defined commands to the Action Component | | |
| 6.9 | Action Component | Map commands to actions | High | Low |
| | | The Action Component must be able to able a user defined mapping of commands from intelligence algorithms to radio or network actions | | |
| 6.10 | Action Component | Execution of actions | High | Low |
| | | The Action Component must be able to execute the actions determined from the mapping of commands to actions. | | |

7 Requirements for portable testbed

7.1 Portable testbed architecture

The fixed testbed architecture shown in Figure 9 consists of two main building blocks, namely, i) the experimental box, in the following called WiSHFUL box, on which the actual wireless experiment takes place and ii) the experiment management server (EMS) providing APIs to deploy, execute and control experiments in the testbed. For the communication between the WiSHFUL boxes and the EMS server a wired backhaul, e.g. Ethernet, is used.

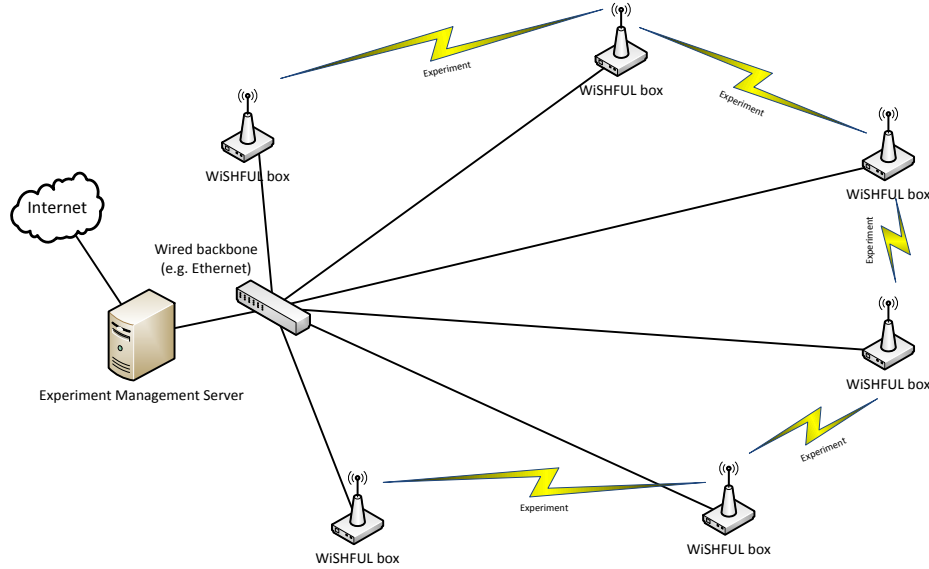


Figure 9. The fixed testbed architecture.

Unfortunately, the mentioned fixed testbed architecture is unsuitable when considering the envisioned portable experimental infrastructure. This is because the installation of a wired backbone especially for large testbeds is in the best case a costly and complex task, and in the worst case difficult if not outright impossible to establish.. Therefore, the portable testbed architecture extends the fixed testbed by an additional block, the WiSHFUL wireless backhaul mesh node, to create a wireless backhaul network replacing the wired one (Figure 10). The objective is to provide the same seamless operation of the testbed infrastructure as in the fixed testbed, i.e. meeting the required KPIs without interfering with the WiSHFUL boxes under experiment (orthogonalized in frequency).

As shown on Figure 10 the WiSHFUL boxes are connected to the backhaul network by wire (e.g. Ethernet). The data channel (blue) of the WiSHFUL backhaul consists of i) control channel for the backhaul, ii) control channel for the WiSHFUL box, iii) measurement data collection. It should be orthogonalized in frequency with the wireless experiment channel (yellow). We do not consider orthogonalization in time by design. Such solutions, although possible, are usually very complex and cannot achieve good performance.

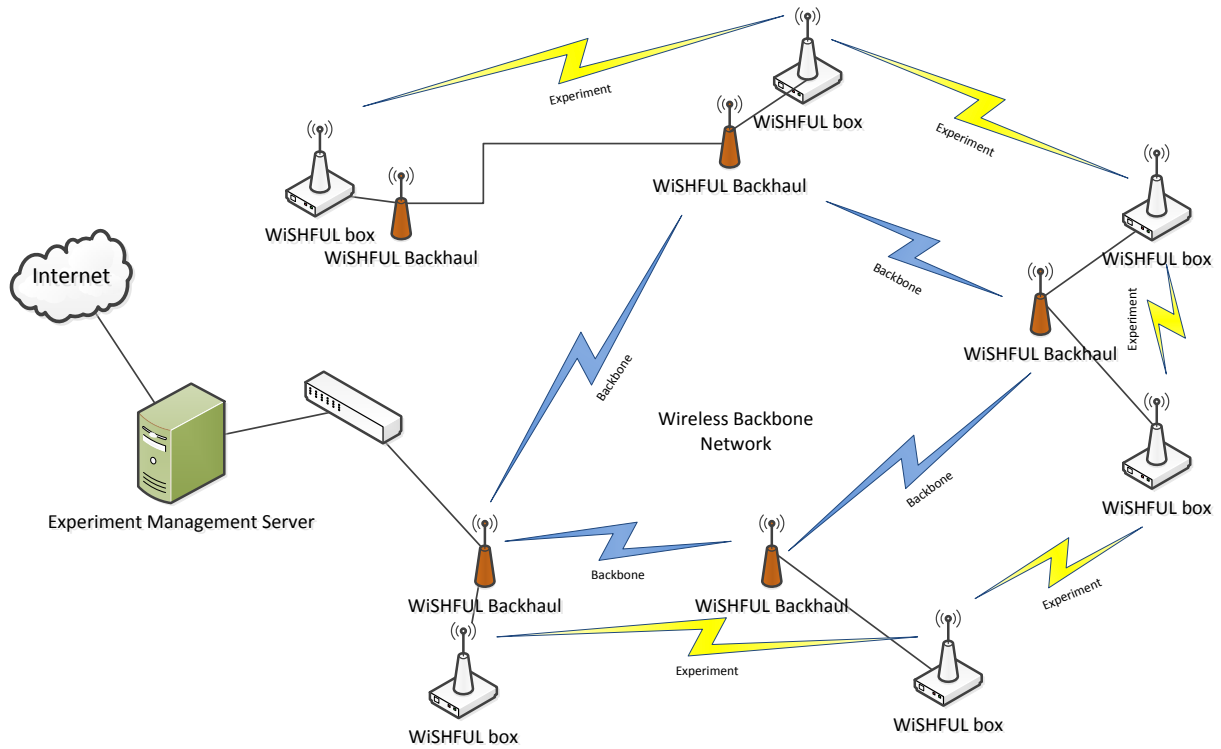


Figure 10. The portable testbed architecture.

To achieve the experiment requirements also the wireless backhaul network need to be controlled in the same way as the wireless network under experiment. Hence, we aim to use the WiSHFUL UPs to control also the wireless backhaul nodes, i.e. the portable testbed is a particular instance of WiSHFUL (Figure 11). The most important difference is that wireless backhaul network has no out-of-band control channel for deployment, execution and control. We actually use the wireless backhaul for the out-of-band control of the experiment running in the experiment testbed. Please note the data flows for controlling the experiment are out-of-band with respect to the actual experiment testbed (also referred to as network under test), but in-band with respect to the backhaul network. The same wireless technology, and protocol stack, for all nodes in the wireless backhaul network are used. Unfortunately, there is a competition between the data to control and monitor the experiment testbed and the data for controlling the wireless backhaul. We hence have to make sure that the data flow requirements are met for both types of control traffic, i.e. for experiment testbed and for wireless backhaul network, hence requiring a joint optimization. The WiSHFUL architecture applied to a portable testbed is shown in Figure 12, where we can identify that the experiment controller can set policies to the wireless backhaul to meet KPIs in both networks.

In some practical applications, a suitable out-of-band control channel may be absent. For this reason it is conceivable, that the network-under-test can realize an in-band control channel. In such situations many additional constraints have to be assured. E.g. the communication peers have to be either in direct communication range or in range of some other node acting as relay. Moreover, the wireless nodes must support at least one interoperable communication stack, which can be ensured when using SDR platform, but may be more challenging when considering COTS IEEE 802.11 and IEEE 802.15.4 devices. It is important to note that these issues, however not touching the portable testbed architecture, can also be inspired by the solutions provided by it.

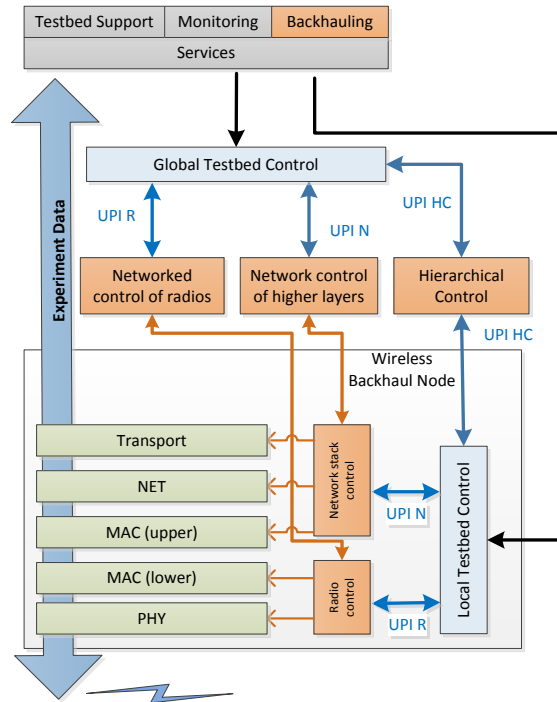


Figure 11. WiSHFUL UPIs are used to control the wireless backhaul of the experiment network.

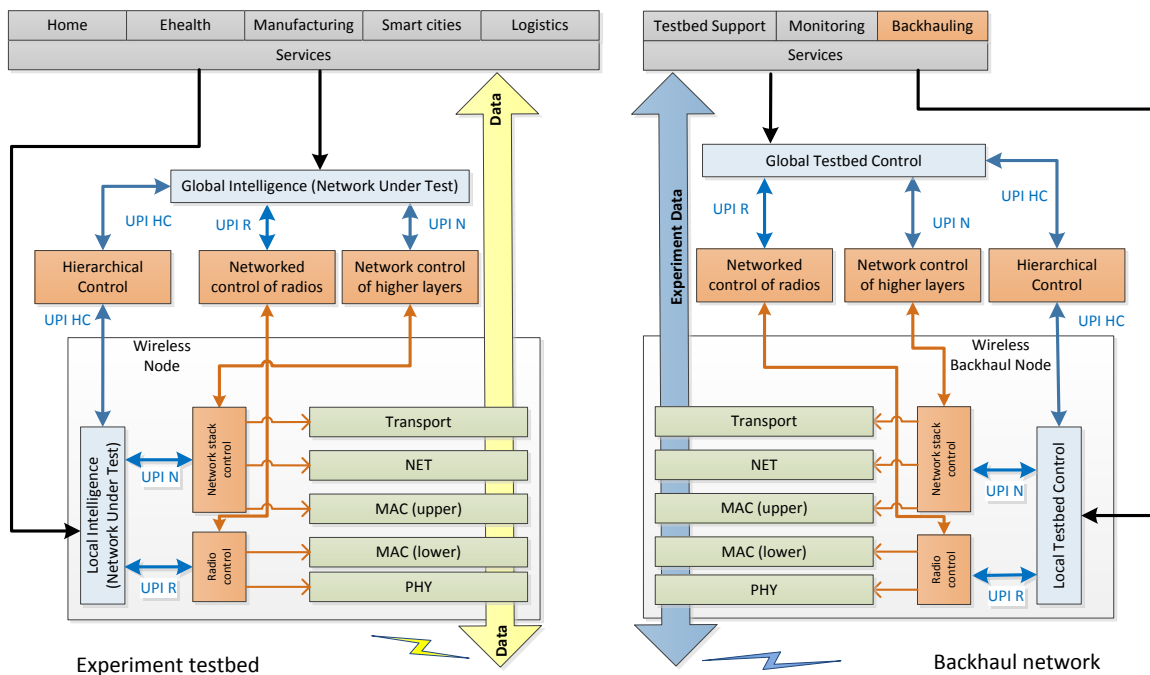


Figure 12. The WiSHFUL architecture when using portable testbed.

7.2 Equipment Portability

One of the main requirements for such a testbed is the portability of the equipment. There are several categories of devices in the proposed architecture:

- EMS: the experiment management server, responsible for the provisioning of the experiment and the control of both WiSHFUL box and WiSHFUL backhaul before, during and after the execution of the experiment.

- WiSHFUL boxes: the devices under test. In first versions of the portable testbed, these will be devices provided by WiSHFUL, to which custom hardware can be added (e.g. a new wireless card or a USB-dongle), or on which experiment software can be installed. Later improvements could allow custom experimenter hardware, with (small) modifications to make the devices compatible with the overall portable testbed architecture.
- WiSHFUL backhaul devices: the devices responsible for providing a backbone wireless network for the control of the WiSHFUL boxes and collecting measurement data.

In the first section, a list of requirements for the hardware is given, in order to ensure the portability of testbed equipment. The second section shows how the equipment can be transported in a reliable way and describes how the transportation equipment (e.g. a flight case) can act as temporary backbone for the hardware.

7.2.1 Hardware requirements

The devices used for the WiSHFUL box and WiSHFUL backhaul should meet some requirements in order to assure the portability of the testbed:

- The embedded devices should have compact design and have a robust casing. We envision dimensions of approximately 10cm x 10cm x 10cm (excluding external antennas). Optionally, custom cases for these embedded devices could be created by additive manufacturing (e.g. a 3D-printer).
- The devices should be powered in different ways, depending on the availability of power sources:
 - The devices should preferably be battery powered. Several COTS battery packs are available, which are able to deliver the correct voltage to operate the embedded devices. This option requires no infrastructure to be available on site;
 - If an Ethernet network is available on site, the devices could be powered using Power-Over-Ethernet. The same Ethernet network could then also be used to enable the WiSHFUL backbone, thus eliminating the need for a wireless backhaul network;
 - In the case where an AC power grid is available, the devices can be simply powered from the grid.
- The devices should have at least one Ethernet interface to connect to either the WiSHFUL backhaul devices or to the temporarily available control network (e.g. for provisioning of the devices). If no Ethernet interface is available on the motherboard of the devices under test, one could make use of USB to Ethernet convertors to enable an Ethernet connection.

The requirements listed above for the WiSHFUL boxes and WiSHFUL backhaul nodes could also be applied to the EMS (Experiment Management Server), but could be interpreted more loosely. As there is most likely need for only one EMS per portable testbed instance, this server could be plugged into a fixed AC power supply and could be larger in size. These less strict requirements for the EMS will allow it to be more powerful and enable the storage of large measurement datasets, a high speed uplink to the Internet (if one is available on site) and fast provisioning & control of experiment nodes. If no power supply is available, one could also use one or more small battery powered embedded devices for deploying the EMS.

7.2.2 Transportation & setup of hardware

In order to easily transport the testbed from one location to another, we propose to use flight cases. An example of a flight case and the proposed configuration is shown in Figure 13.



Figure 13: Flight case to transport portable testbed

In Figure 14, the WiSHFUL boxes and WiSHFUL Backhaul devices should be seen as the combination of an embedded PC with their corresponding battery pack. The EMS is considered to be a static part of the testbed and will therefore be permanently attached to the flight case.

To achieve plug-and-play of portable testbeds hardware, a mechanism will be developed using the principle of power line communications (PLC). When plugging in devices (incl. battery pack) to the flight case, which can be seen as the backbone network of the portable testbed, a connection is made to the backbone. This connection will simultaneously take care of recharging the battery pack, while providing a network connection (PLC) between the backbone network and the embedded device that is being plugged in. Ski-like pads in combination with golden spring contacts have been used in previous projects (FP7 OpenLab) and are currently being used to provision the mobile nodes and recharge the battery packs in the iMinds w-iLab.t testbed. This solution simplifies portable testbed management w.r.t to network and power cabling (no need for the testbed user to plug in the network cables or to connect the power adapter). Figure 15 shows the use of spring contacts in the w-iLab.t testbed.

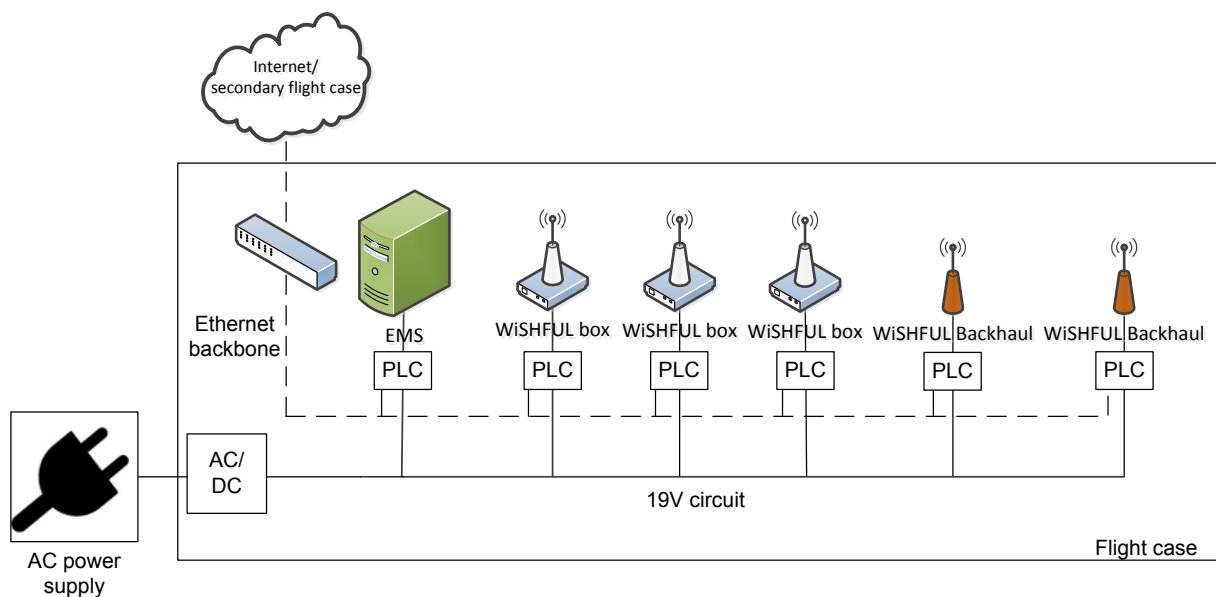


Figure 14: Configuration of devices of the portable testbed in the flight case



Figure 15. Spring contacts used in the iMinds testbed

We assume there is an external AC power supply available for the recharging of the nodes. In case an Internet uplink is available, the EMS could function as a gateway for all nodes in the testbed. This could highly increase the usability of the testbed, as not all software installation has to be done before transporting it to the testing location. In the above configuration, the experimenter can arrive at the test location and do some initial configuration and testing of the nodes before disconnecting them from the wired backbone of the testbed. If there is an Internet uplink available on site, or a 3G/4G dongle is used for these purposes, experimenter could even do the testing from a remote location. Please note that the equipment resides in the flight case for charging purposes, initial installation and testing only, but can later be taken out of the case to be deployed at any location.

If the number of nodes that is needed for the tests exceeds the storage capacity of the flight case, multiple flight cases could be interconnected through the switch at the top left of the diagram (see Figure 14). These secondary flight cases do not have to contain an EMS. Multiple flight cases should preferably be connected in star topology with the primary flight case, containing the EMS, as the centre of the network. A linked-list topology of flight cases could however be considered if the testing area is considerable large and parts of the portable testbed are spread over the site. Figure 16 shows a star topology consisting of one primary flight case (containing the EMS) and two secondary flight cases. Note that this is only feasible if an Ethernet network is available on site, or if one can be installed without too much overhead. If there is no Ethernet network available, all devices are interconnected by using the portable testbed wireless backhaul network.

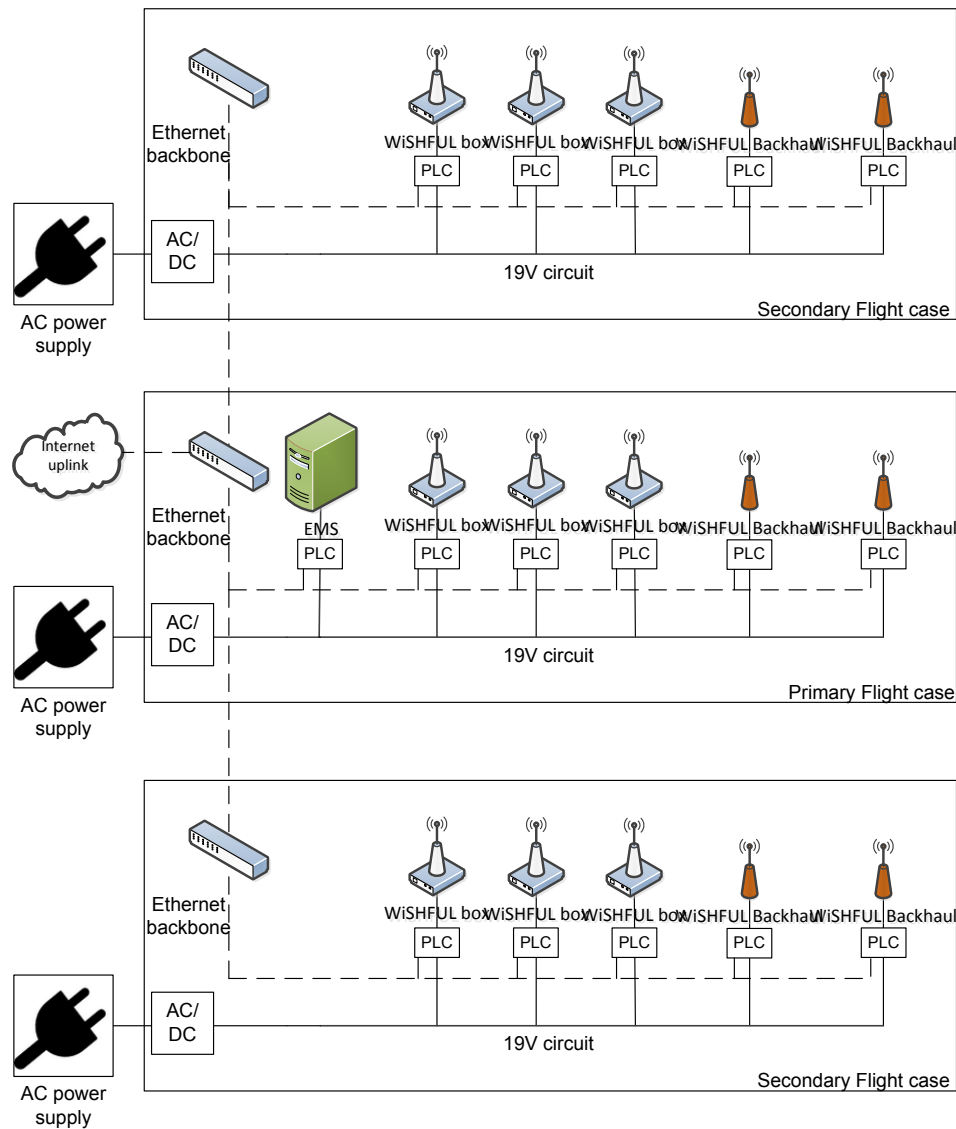


Figure 16. Interconnection of multiple flight cases for larger portable testbeds

7.3 Functional requirements

The portable testbed aims to achieve easy and fast deployment of the testbed at the remote location. This can be for example a conference venue, industrial site or open field. Additionally to the well-defined and standard testbed setup and interfaces, it is responsible for the following operational aspects:

| Functional Requirements for portable testbed | | | | |
|--|---|--|----------|------------|
| UC# | Actors | UC name and short description | Priority | Complexity |
| 7.1 | Experiment Management Server/ Testbed Nodes | Testbed monitoring | High | Low |
| | | Portable testbed has to provide the same level of control and monitoring as normal normal/fixed testbed. | | |
| 7.2 | Global and Local | Wireless link monitoring (normal) | High | Medium |

| | | | | |
|------|---|--|--------|--------|
| | Testbed Control | Portable testbed controller provided by the WiSHFUL backhaul has all monitoring functions as defined in previous sections for UPI. It has to be able to check if traffic channels needed for experiment control & monitoring meet the bandwidth & latency requirements and report the outcome. | | |
| 7.3 | Global and Local Testbed Control | Wireless link monitoring (debug) | Medium | Medium |
| | | As in previous point, additionally the monitoring functionalities have to be propagated to the experimenter for further manual inspection. | | |
| 7.4 | Backhauling Nodes Global Testbed Control | Wireless backhaul interference | High | Medium |
| | | The wireless backhaul network must not interfere with the experiment network. | | |
| 7.5 | Backhauling Nodes Global and Local Testbed Control | Backhaul connectivity | High | High |
| | | The wireless backhaul has to provide a fully connected network i.e. each WiSHFUL box can connect with each other. If the backhaul nodes are not in the direct communication range, relaying has to be provided. | | |
| 7.6 | Backhauling Nodes Global Testbed Control | Backhauling transparency | Low | Medium |
| | | The wireless backhaul network has to be transparent to the experimenter. | | |
| 7.7 | Testbed Nodes Global Testbed Control | Testbed auto-configuration | Low | High |
| | | It should be possible to deploy the testbed with minimal effort. That means nodes should be able to automatically setup the network that can be used by experimenter. | | |
| 7.8 | Testbed Nodes | Plug-and-play of portable testbeds hardware | High | Medium |
| | | It should be possible to insert and remove testbed nodes from the backbone in the flight case without having to unplug cables or remove fixtures/screws. | | |
| 7.9 | Flight case Testbed nodes Controller | Easy transportation & robustness | High | Medium |
| | | The flight case containing the testbed nodes, controller and backbone hardware should be easy to transport and be resistant to external factors such as shocks, rain, dust. | | |
| 7.10 | Flight case | Ventilation, cooling & fire proof | High | Low |
| | | The flight case should have enough ventilation to prevent the hardware from over-heating. Holes in the flight case should be made so that | | |

| | | | | |
|------|-------------|--|------|-----|
| | | no rain or dust can enter the case. Materials inside the flight case should be fire proof in case a short circuit would be caused by a bad connection of the recharging mechanism. | | |
| 7.11 | Flight case | Internet uplink & power source | High | Low |
| | | It should be possible to plug the backbone of the flight case into an existing Ethernet network and connect it to an external AC power supply. | | |

8 Testbed requirements: Fed4FIRE compliance

In order to reduce the threshold for experimentation, the participating testbeds in the WISHFUL project will follow the current de facto standards in FIRE, set by the FED4FIRE project, for testbed interoperability.

The Fed4FIRE project defines three different levels of federation: advanced federation, light federation and associated testbeds. Testbeds that comply with the advanced federation model can be seen as being fully Fed4FIRE (F4F) compliant. The light federation model allows testbeds to implement only a subset of the F4F APIs. Finally, the F4F associated testbeds are only mentioned on the F4F website, with references towards the testbed documentation and contact details. For this last -very basic- federation form, no technical work is required. Depending on the architecture of the testbed, some federation models might not be feasible. The advanced federation model has the most restrictions regarding the testbed architecture.

The following sections shortly describe the F4F architecture, followed by the different types of federation models. Every federation model is presented in more detail by going over the requirements (both technical and non-technical) for the testbeds who want to comply with this model. For every federation model, the possible benefits for the newly federated testbeds are summed up. To conclude this chapter, a brief overview is given of the steps that need to be taken by a new testbed in order to join the F4F federation and be compliant with any of the three federation types.

This chapter was constructed in close collaboration with the Fed4FIRE project (FP7 project number 318389) and is based primarily on F4F D2.1.

8.1 Fed4FIRE architecture

This section gives a brief summary of the architecture used in Fed4FIRE as described in F4F D2.1.

The experimental life cycle forms the basis of the Fed4FIRE architecture and is split up in several phases:

1. **Resource discovery:** discovery of the facility resources by means of a uniform resource description model for all testbeds.
2. **Resource requirements:** detailed specification of the resources required for the experiment. The requirements can be computational, network or storage related, or can even be as specific as the need for software libraries.
3. **Resource reservation:** there can be different models for the reservation of resources:
 - No hard reservation or best-effort (use of a calendar loosely linked to the testbed)
 - Hard reservation (once reserved, one has guaranteed resource availability)
 - Future reservation (one should reserve sufficient time in advance)
 - Instant reservation (one can only do instant reservations)
4. **Resource provisioning:** two models are defined for the provisioning of resources (e.g. the loading of a new operating system, the configuration of network links):
 - Direct provisioning (API): the testbed provides an API where experimenters can instantiate selected resources.
 - Orchestrated provisioning: a functional component decides which resources are the best match for the experiment.
5. **Experiment control:** control of the testbed resources during experiment execution. This can be achieved either by using an FRCP compatible tool or SSH. The experiment control can be composed of predefined interactions and/or commands to be executed on resources, either at start-up or during the experiment workflow.
6. **Monitoring:** a separation is made between monitoring of

- Resources: monitoring of the infrastructure health and resource usage (e.g. CPU, RAM, network traffic, availability of resources).
 - The experiment: monitoring of user-defined experimentation metrics of the solution under test (e.g. throughput and packet loss of a wireless connection).
7. **Permanent storage**: storage of experiment description files or experiment data. This data should remain available after the experiment lifetime.

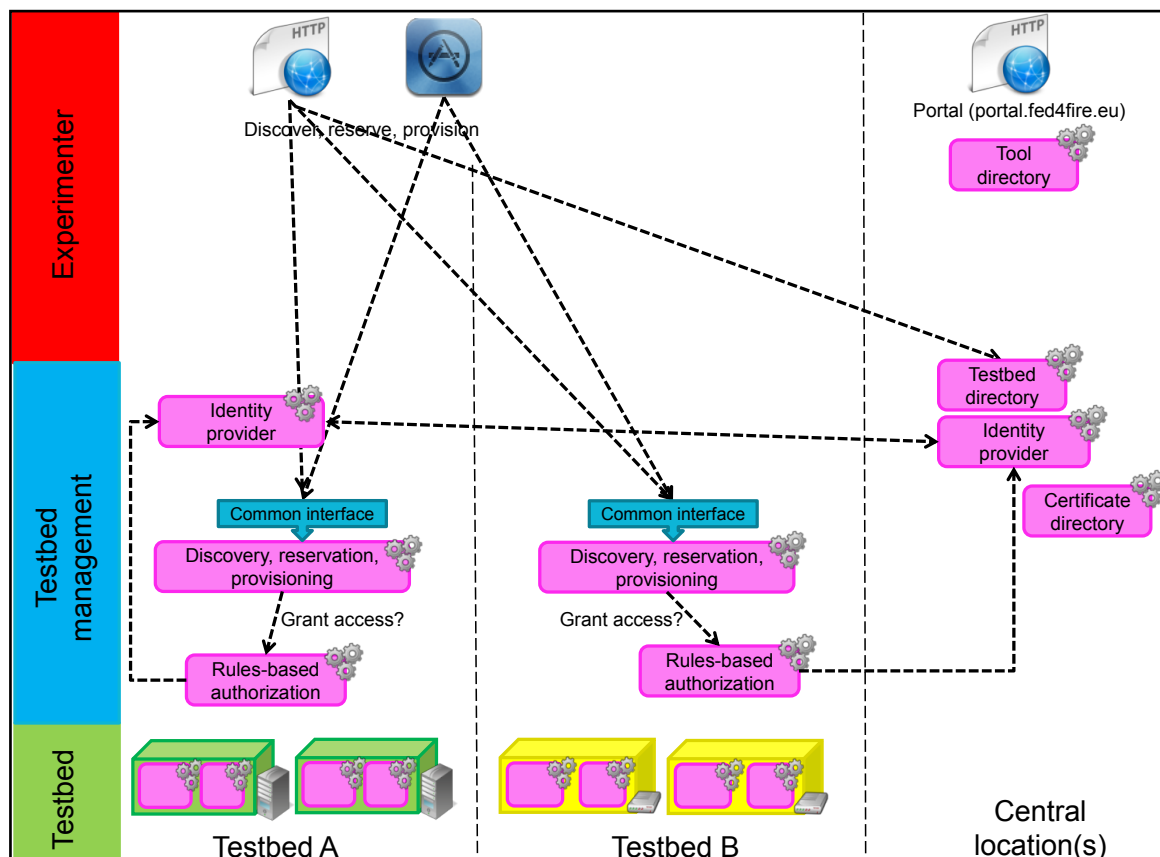
In the following sections, the phases of the experimental life cycle are mapped onto architectural requirements. There are three main sections:

- Requirements for resource discovery, resource requirements, reservation and provisioning (steps 1 to 4 of the experimental lifecycle).
- Requirements for monitoring and measurements (step 6 of experiment life cycle)
- Requirements for experiment control (step 5 of experiment life cycle)

Each topic is detailed in the following chapters and contains a summary of the requirements that are relevant for WISHFUL testbeds. No requirements are given regarding permanent storage, so this step can be seen as optional for testbeds wishing to join the federation.

8.1.1 Resource discovery, requirements, reservation and provisioning

Figure 17 (see F4F D2.1) shows the different blocks that are relevant for the first steps of the experimental life cycle. Some functionality is situated at a central location in the federation, while other functions are implemented at the testbeds. The horizontal layers of the pictures are used to make the distinction between experimenter tools, testbed management tools and testbed resources.



8.1.2 Figure 17. Functional blocks for resource discovery, requirements, reservation and provisioning

Central location(s)

The functionality at the central location is provided by the F4F federation. One or more **portals** (web-based or other) are available to complete the following tasks:

- provide a single point of entry for new experimenters of the federation, including an identity provider and certificate directory. Experimenters who create an account on the portal should automatically be registered at the identity provider. The certificate directory then provides a trusted location of root certificates of these identity providers.
- publish a testbed & tools directory which can be consulted on a website or by querying an API. These directories should contain useful information for the experimenter on the testbed hardware and the available tools.

Testbed side

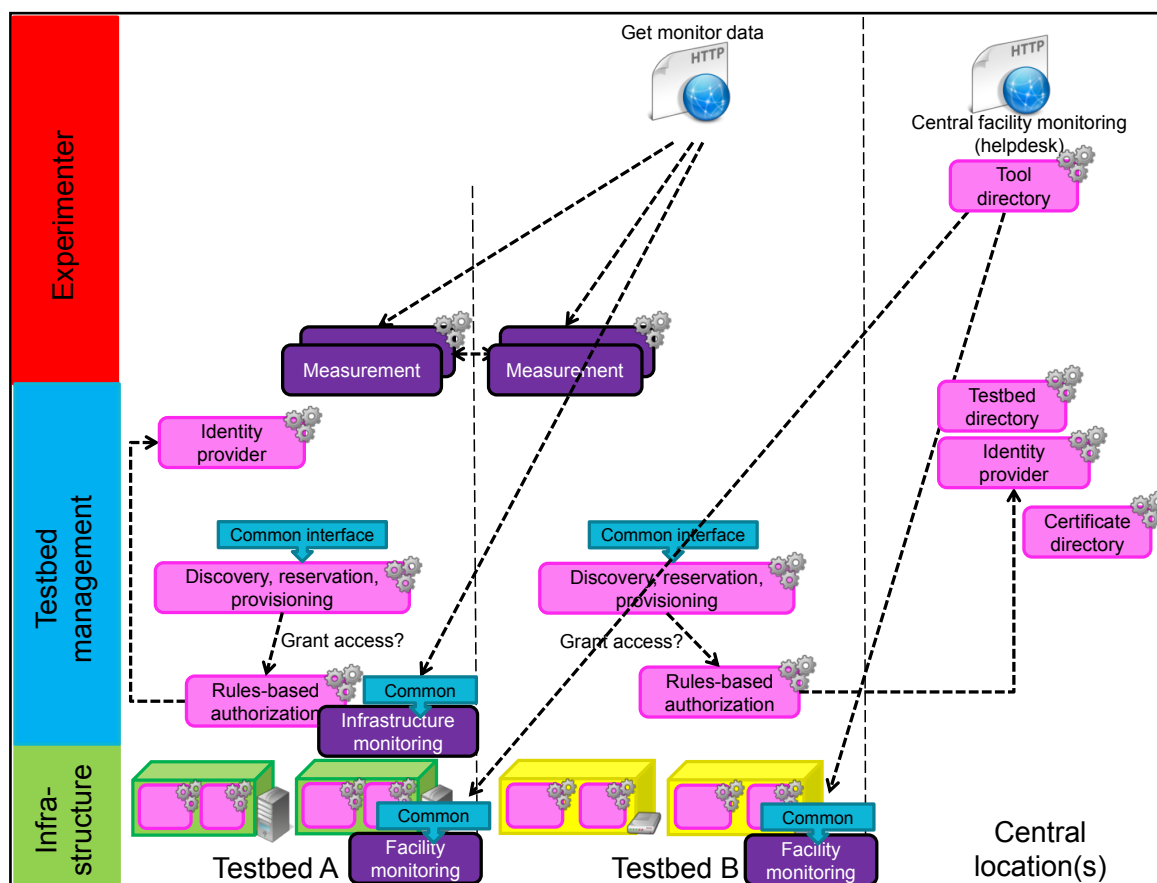
At the testbed side, the following requirements should be fulfilled:

- For every testbed, a component has to be implemented (**Federation AM API** [6]) which does discovery, reservation and provisioning of testbed resources using an SFA interface [7] (based on Geni AM API v3 [8]). The federation AM API communicates with other tools (e.g. jFed [9]) and testbeds by using RSpec [10] containing a uniform description of testbed resources. Identical resources on different testbeds can thus be described by using the same RSpec format.
- A testbed can decide to be its own **identity provider**, or use the identity provider of the federation. For authentication and authorization [11][12] of users on specific testbeds a trust model is used, where identity providers trust each other and specific experimenter properties are included in the experimenter's certificate, which is signed by the identity provider. Using this information, testbeds can do rules-based authorization.
- A testbed can choose to query (and trust) the **central federation certificate directory** to see which root certificates it should trust.

8.1.3 Monitoring and measurements

The figure below shows two types of monitoring (facility and infrastructure) and the measurement of experiment data.

- The **facility monitoring** is used by the experimenter to determine if a facility is still up and running. Most testbeds already do this kind of tests by using a tool like Zabbix [13] or Nagios [14] to monitor their core servers and switches.
- The **infrastructure monitoring** collects data that is useful for the experimenter. An example could be the CPU usage of a shared virtual machine host or the amount of interference in a wireless frequency band.
- The **measurement of experiment data** addresses the logging of experiment specific metrics like packet loss of a certain wireless MAC protocol. The testbed can facilitate this step by providing a database server to store the experiment results (e.g. an OML server [15]).



8.1.4 Requirements for experiment control

The testbed should allow the experimenter to control the testbed resources during the experiment execution. This could be achieved either by allowing the user to log in to the resources using SSH, API calls to a central server, or by using an FRCP [16] compatible tool like OMF [17] or NEPI [18]. A basic form of experiment control is also available in some of the federation tools that take care of the provisioning of resources (e.g. iFed [9]).

The following chapters give a detailed description of the different federation models, with a listing of requirements and benefits for the joining testbeds.

8.2 Advanced federation

The advanced federation model can only be implemented on testbeds where experimenters can gain access towards the individual testbed resources by either using SSH or FRCP [16]. These resources can be shared between different users over time or in parallel. Access to the resources should be dedicated and should implement the concept of credentials [12].

The advanced federation model requires new testbeds to be fully F4F compliant and thus fully integrated into the federation. Experimenters of the federation should be able to control every step of their experiments life cycle, starting with the selection of resources followed by the provisioning step to end with the collection of measurement data. Monitoring of the facility and the infrastructure should also be addressed in this model.

8.2.1 Requirements of advanced federation

There are several requirements for a testbed to join the federation under this model. They are summed up below.

Support for F4F Federation AM API.

New testbeds should implement the federation AM API [6] to open up certain functionality towards experimenters of the federation. In short, this API will translate the SFA [7] protocol towards testbed specific functionality. There are several implementations of this API already available, so the testbed providers do not have to implement everything themselves. Not all functionality in the AM API should be implemented. The mandatory topics are listed here:

- The federation AM [6] should implement authentication and authorization functionality based on X.509 certificates [12]. The AM issues slice and user credentials to allow users to create and run experiments on a (sub-)set of testbed resources. Since the advanced federation model fully includes the testbed into the federation, making it fully F4F compliant, the AM should accept root certificates of the main F4F authorities.
- The AM should expose testbed resources and allow the discovery of those resources by experimenters. As this has to happen in a uniform way, to allow the use of different federation tools, a common XML format was agreed upon: the RSpec definition [10]. The RSpec contains all information the experimenter needs to know about the testbed resources in order to construct his/her experiment.
- Instant provisioning of the testbed resources should be supported through the AM API. Using the RSpec resource description the experimenter obtained in the resource discovery phase, the experimenter can instantly activate his/her experiment if enough testbed resources are available at that time.
- The implemented federation AM API should allow the experimenter to control the experiment after it was started. This can be achieved either by using FRCP control messages (e.g. in combination with the OMF framework) or by allowing the users to manually log in to the nodes by using SSH with public/private keys, which are added to the API calls.

Documentation.

The testbed documentation should be available on a webpage maintained by the testbed providers. It should contain a detailed description of the testbed, including a description of testbed resources in RSpec format and a URL of the server running the federation AM API. The documentation should at least include one tutorial on how to run a basic experiment using a tool provided by the federation. Note that this experiment should not use a lot of resources, since every federation member has to be allowed to run this experiment (see next section on policies).

Policies.

Testbeds joining the F4F federation in the advanced federation model should allow everyone with a valid F4F certificate to execute the basic experiment without extra approval. The testbed can decide for itself if it allows experimenters from the federation to run more complex experiments using more resources.

Facility monitoring.

The federation AM API from all federated testbeds are tested from a central location. If the testbed has internal monitoring tools, a summary can be sent - using OML [15] - towards the OML server of the federation.

Connectivity.

The federation AM should be publicly available over IPv4. SSH logins towards testbed resources should be allowed either over public IPv4 or IPv6. Exceptions for testbeds which can only be reached over VPN connections can be granted, but then the SSH gateway of the F4F federation should be a permanent client of the VPN. The federation tools will then log in to testbed resources by going through the SSH gateway of the federation.

Support.

The testbed has to provide basic support on the testbed functionalities towards experimenters.

Extra federation options.

The F4F federation offers some possibilities to include extra functionality in the testbeds. These are however optional or not yet fully operational in the F4F federation at time of writing this deliverable. They are mentioned in this section for completeness only and are not hard requirements to join the F4F federation (at this time). Some of these extra options include but are not limited to: advanced reservation, infrastructure monitoring, service level agreements, testbed reputation service and experiment control by using FRCP.

8.2.2 Benefits of advanced federation

Some of the benefits of joining the F4F project under the advanced federation model are:

- The federation offers some testing tools for the AM API (e.g. to test credentials) for all testbeds in the federation. These tests are run periodically and the results are shown on the central monitoring dashboard [19]. On a nightly basis, more extensive tests are run to see if it is still possible to discover and provision resources in the testbeds. The tests are monitored by a first level support team that informs the testbed administrators when one or more tests failed.
- The central monitoring dashboard [19] of the federation is available to all experimenters. This makes it very easy for experimenters to check whether a specific testbed is still operational and if there are some free resources.
- At least one federation client tool can support all federated infrastructure testbeds.
- At least one federation authority can issue credentials for experiments.
- The federation SSH gateway is available for all testbeds that cannot offer public IPv4 access to their resources (to bridge e.g. to IPv6, VPNs, etc.)
- The central documentation of the federation contains links to all testbeds.
- Central support tools (Google group [20], NOC) are available for first help and provide a single point of contact for experimenters.
-

8.3 Light federation

The light federation model can be used for testbeds where experimenters cannot gain direct access to the testbed resources, but access is granted by exposing a Web-based API on one or more (central) servers. These servers will then translate the API calls and address the testbed resources. This option does not allow full control over individual testbed resources, but ensures unified access to experimenters, through an API. Similar to the case of the advanced federation model, the concept of credentials should be implemented [12].

8.3.1 Requirements for light federation

Support for Fed4FIRE credentials.

The testbed should support Fed4FIRE credentials in a client based SSL API, which exposes the testbed functionality to the federation. To this end, X.509 certificates have to be adopted, which are issued by one of the authorities from the federation. These certificates can for example be used in a derived PKCS12 version, which can be loaded in a web browser or other HTTPS capable tools. Note that the API in this section refers to the testbed specific API and not the federation AM API as described in the advanced federation model.

Documentation.

The testbed documentation should be available on a webpage maintained by the testbed providers. It should contain a detailed description of the testbed resources, including a URL of the server running the testbed specific API. All the functionality of the API should also be well documented. The documentation should at least include one tutorial on how to run a basic experiment using a tool provided by the federation. Note that this experiment should not use a lot of resources, since every federation member has to be allowed to run this experiment (see next section on policies).

Policies.

Testbeds joining the F4F federation in the light federation model should allow everyone with a valid F4F certificate to execute the basic experiment without extra approval. The testbed can decide for itself if it allows experimenters from the federation to run more complex experiments using more resources.

Facility monitoring.

The testbed API is tested from a central location. If the testbed has internal monitoring tools, a summary can be sent, using OML, towards the OML server of the federation.

Connectivity.

The testbed API should be publicly available over IPv4.

8.3.2 Benefits of light federation

Some of the benefits of joining the F4F project under the light federation model are:

- At least one federation authority can issue credentials for experiments, and the credentials are verified on the testbed specific API.
- Information is provided on enabling PKCS12 authentication in new tools (web based or other). At least one client tool is able to extract PKCS12 credentials from X.509 certificates.
- The central monitoring dashboard of the federation is available to all experimenters. This makes it very easy for experimenters to check whether a specific testbed is still operational and if there are some free resources.
- The central documentation of the federation contains links to all testbeds.
- Central support tools (Google group, NOC) are available for first help and provide a single point of contact for experimenters.

8.4 Associated testbeds

Associated testbeds are not seen as federated testbeds. The testbed is only listed on the Fed4FIRE website, mentioning links towards contact information and testbed documentation. This type of “federation” does not require any technical integration, meaning that no credentials are exchanged between the testbed and the F4F authority and no nightly testing is done. The testbed also has to organize its own support, no help is provided from the federation. Moreover, associated testbeds do not have to provide access towards members of the federation by default.

This type of “federation” requires the least amount of effort from the testbed providers, but the benefits of joining the federation are limited. The learning curve for experimenters from the federation remains the same. They have to create a new account on the testbed and learn to work with the testbed specific tools for setting up and running experiments. No tests are run on a nightly basis, neither is a first line support team (or user community) available to help with various issues.

8.5 Status of WiSHFUL testbeds

This chapter gives an overview of the current federation status of WiSHFUL testbeds. For features that are not yet available, it is indicated whether they are planned or not applicable to the testbed in question.

| | w-iLab.t (iMinds) | ORBIT (Rutgers) | FIBRE Island @UFRJ | IRIS (TCD) | TWIST (TUB) |
|------------------------|---|--|--|---|--|
| Discovery | SFA (GENI AMv3) | XML | SFA (GENI AMv3) | SFA Planned | RESTful |
| Requirements | SFA (GENI AMv3) | XML | SFA (GENI AMv3) | SFA Planned | RESTful |
| Instant Reservation | SFA (GENI AMv3) | Web based (XML) | SFA (GENI AMv3) | SFA Planned | Web based |
| Provisioning | SFA (GENI AMv3) | custom (reservation based exclusive access) | SFA (GENI AMv3) | SFA Planned | custom (reservation based exclusive access) |
| Experiment Control | SSH or FRCP (OMF6) | OMF (5.5 and 6.0) | OMF (5.4) | SSH (FRCP Planned) | SSH / Custom (FRCP Planned) |
| Facility Monitoring | Zabbix+OML (core servers + switches) | OpenNMS + OML | ZenOSS | OML Planned | cacti + collected |
| Connectivity | Public IPv4 for AM Public IPv6 for resources | Public IPv4 + Firewalled IPv4/IPv6 | Public IPv4 for AM Tunnel to private IPv4 for resources | Public IPv4 for AM Tunnel to private IPv4 for resources | Public IPv4/ VPN |
| Documentation | OK | Partial | Partial | OK | OK |
| Policies | OK | ORBIT+GENI | To be updated | To be updated | OK |
| Federation Level | Fully F4F compliant | GENI | Planned full F4F compliance | Planned Full F4F compliance | Planned Full F4F compliance |

8.6 Steps to join the Fed4FIRE federation

Testbeds that want to be part of the F4F federation should follow these steps:

- (1) Get in contact through contact@fed4fire.eu to help you decide on the level of federation and to support you with your developments.
- (2) Decide on which level you want your testbed to be federated (Associated, Light or Advanced):
 - If you want your testbed to be Associated with the federation, skip to step (5)
 - If you want to include your testbed through Light federation, skip to step (4)
 - If you want to include your testbed through Advanced federation, go to step (3)
- (3) Follow the next steps to support the Federation AM API to control your testbed's resources:
 - Implement the Federation AM API on top of your testbed, based on the documentation at <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html>
- (4) Follow the next steps to expose your testbed services through a REST API:
 - Support Fed4FIRE credentials in a client-based SSL API (see <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/credential-sfa.html> and <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/general.html>)
- (5) Send an e-mail to contact@fed4fire.eu with the following information:
 - An URL
 - to the documentation of your testbed
 - An URL to the Terms and Conditions to use your testbed
 - A tutorial for basic usage of your testbed
 - An e-mail address to contact your testbed manager

The Fed4FIRE team will then update the FLS dashboard (<https://flsmonitor.fed4fire.eu/>), write and execute nightly monitoring tests (<http://monitor.ilabt.iminds.be/>) and support experimentation on your testbed through at least one client tool and identification authority.

9 Consultations with innovators

During proposal writing we have discussed with many innovators from small, medium-size and large enterprises from different vertical markets, and most of them have expressed their interest in the WiSHFUL software platforms. Now, at the start of the project, we are in the process of consulting these innovators to obtain precise information on domain-specific applications requirements. First, we prepared two questionnaires (see screenshots of the one for industry in Annex 1), one targeted for industrial experimenters and one to academic ones, accessible at the following URLs:

- Industry:
https://docs.google.com/forms/d/1Y1WvBHHaUecCw0A6_5jL2B5KtnAxMn5szNIJYFIB0Yc/viewform?usp=send_form
- Academic:
https://docs.google.com/forms/d/1UYtes9AZz6mk_S95vb-kB7F70_UgHZnNlyfdknJA2q0/viewform?usp=send_form

These questionnaires have been sent out on various mailing and hereafter we present the partial results. Additionally, we are in the process of organizing a public stakeholders consultation workshop on these topics.

The main goal of this effort is to translate the domain specific application requirements into a set of requirements of the software platforms for radio control and network control to be developed within the project. These consultation rounds (e.g. questionnaires and workshops) will be organized during the course of the project in order to fine-tune the specifications.

The following sections present the partial results of the industry questionnaire. More comprehensive results will be presented at the WiSHFUL workshop at the EuCNC conference in June/July 2015.

Note that the sum of the percentages in some of the following charts exceeds 100% as respondents were able to select multiple answers.

9.1 PART 1: Gaining insight in the activities of the industry partner

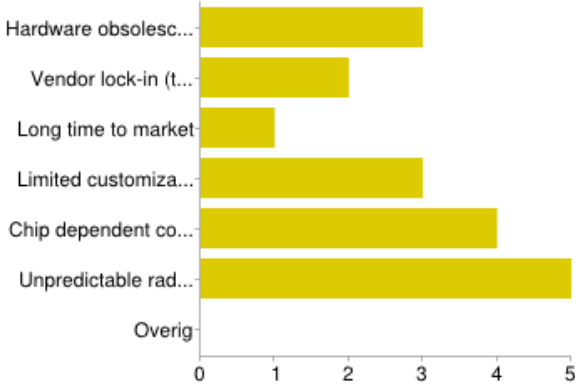
The partial results from European Industry that are currently available include four innovative companies (2 large, 2 SMEs) that classify themselves as radio chip integrator, installer and operator of network equipment and system integrators¹. They target both niche and broad markets and the dominant focus is Wi-Fi (IEEE 802.11 a/b/g/n/ac) and 3G. Also LTE, some other IEEE 802.11 and 15 standards and some sub-GHz ones are on their list.

9.2 PART 2: Current development practices

The most frequent problems they encounter are summarized in Table 1. It can be seen that unpredictable radio behaviour, chip dependent code, limited customization opportunities and hardware obsolescence are frequent among the respondents. These are all aspects that WiSHFUL addresses through the UPIs, re-usable code and the inclusion of SDR elements.

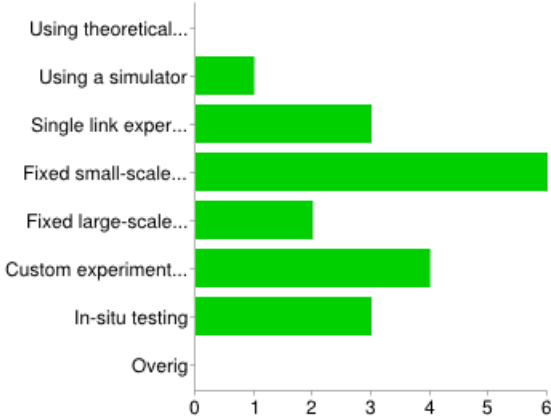
¹ Note that 2 participants to the poll seem to be academic rather than industrial, so altogether 6 people filled the forms.

Table 1 Response results for "Which problems do you encounter during the (re)design of new wireless solutions"

| | | |
|---|---|-------|
|  | | |
| Hardware obsolescence (being forced to switch to new radio chip due to short radio chip availability) | 3 | 42.9% |
| Vendor lock-in (too costly/time consuming to switch to a new radio chip vendor or manufacturer) | 2 | 28.6% |
| Long time to market | 1 | 14.3% |
| Limited customization options (fixed radio behaviour) | 3 | 42.9% |
| Chip dependent code (several software / hardware aspects of your solution depend strongly on a specific chip) | 4 | 57.1% |
| Unpredictable radio behaviour (due to bugs, incomplete documentation, missing features...). Please specify below. | 5 | 71.4% |

The remaining questions and answers of the partial results are available below, without the company names due to privacy reasons.

Table 2 Response results for "How do you currently develop, test, stress-test or demonstrate wireless solutions?"

| | | |
|---|---|-------|
|  | | |
| Using theoretical tools (such as matlab, ...) | 0 | 0.0% |
| Using a simulator | 1 | 14.3% |
| Single link experiments (communication between 2 devices) | 3 | 42.9% |

| | | |
|--|---|-------|
| Fixed small-scale experiment set-up (< 10 testing nodes) | 6 | 85.7% |
| Fixed large-scale experiment set-up (≥ 10 testing nodes) | 2 | 28.6% |
| Custom experiment setup for each new test phase | 4 | 57.1% |
| In-situ testing | 3 | 42.9% |

Table 3 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Non-realistic experimentation conditions”

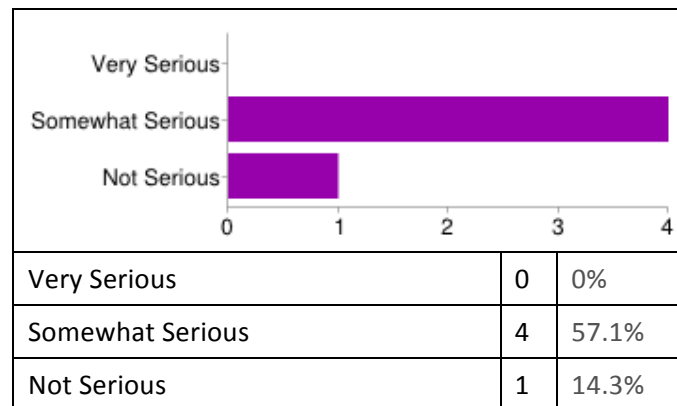


Table 4 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Not enough configuration flexibility ”

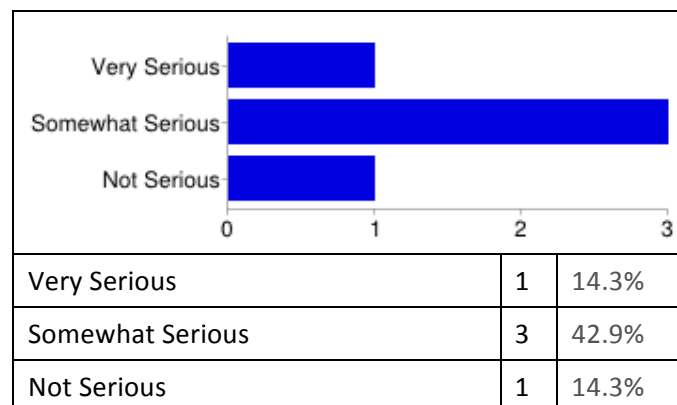


Table 5 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Not enough insight into the behaviour of the solution”

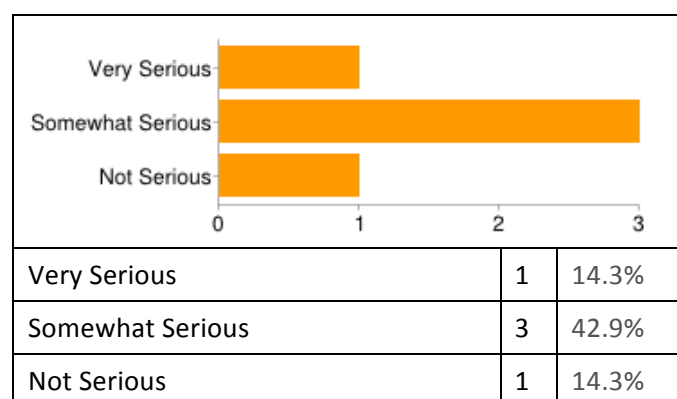


Table 6 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Not enough insight into the behaviour of the environment”

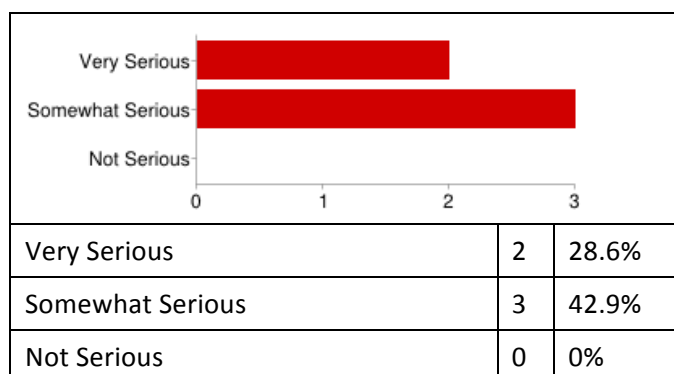


Table 7 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Too costly to set-up experiments”

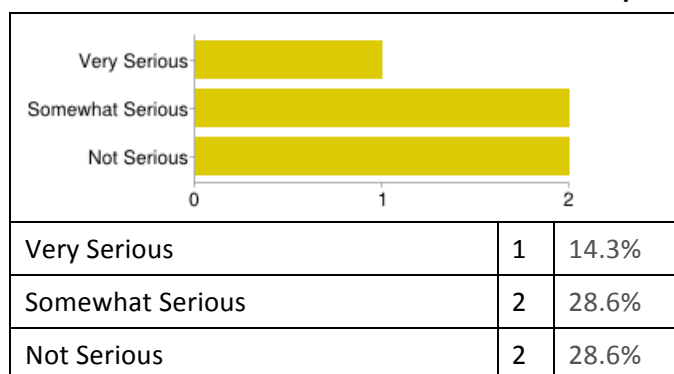


Table 8 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Too small scale experiments”

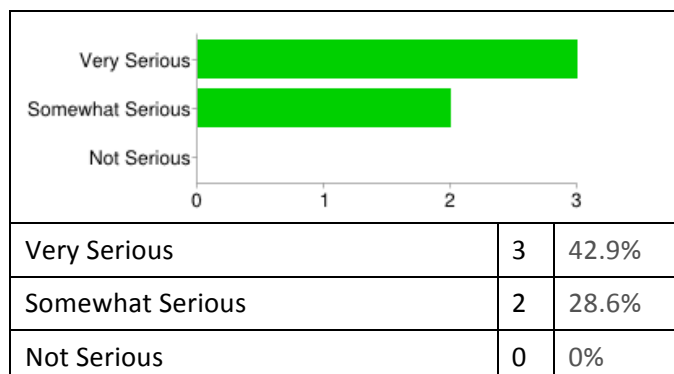


Table 9 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Complexity of writing low-level driver and MAC software”

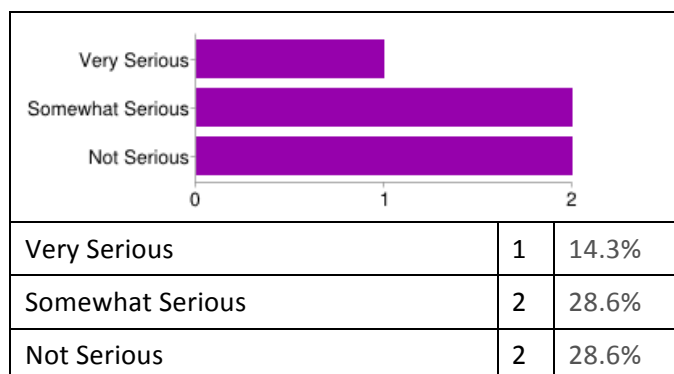


Table 10 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Lack of code reuse at driver and MAC level”

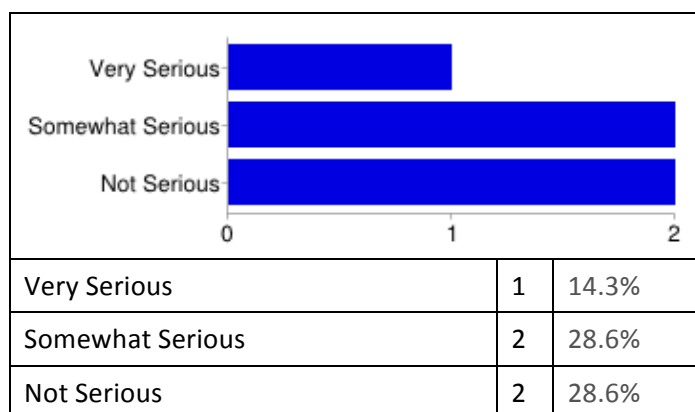


Table 11 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - Difficult to switch towards a new technology ”

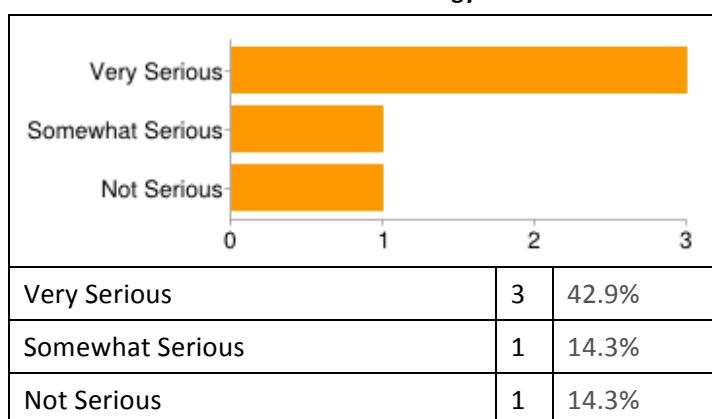
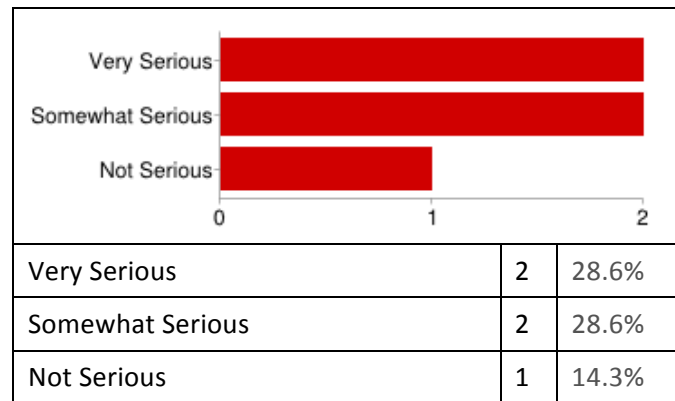


Table 12 Response results for “Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions? - No stress-test possibilities”



9.3 PART 3: Identifying the technical focus points for WISHFUL

Table 13 Response results for “Which aspects of a wireless solution would be most beneficial to control during development, testing, optimization or demonstration of wireless solutions? - Physical layer aspects”

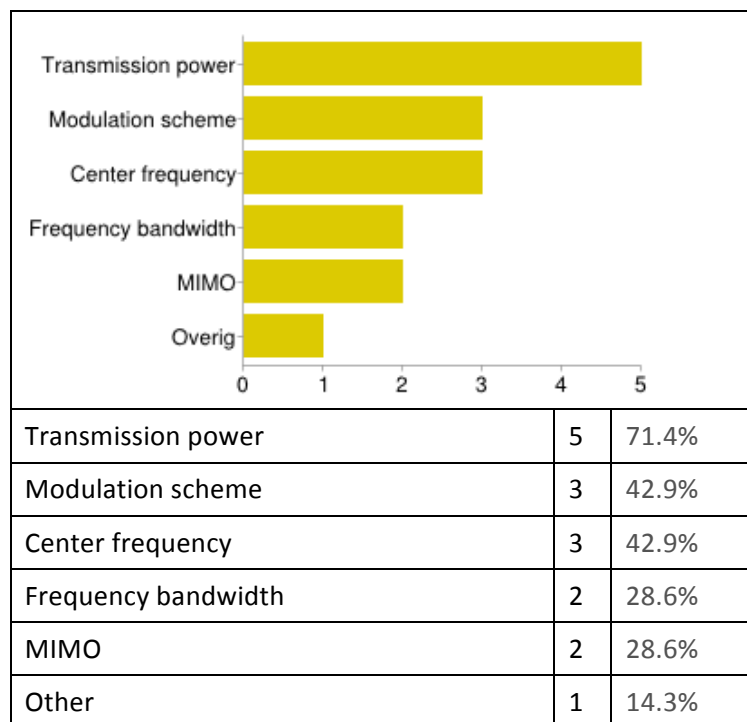


Table 14 Response results for “Which aspects of a wireless solution would be most beneficial to control during development, testing, optimization or demonstration of wireless solutions? - Link layer aspects”

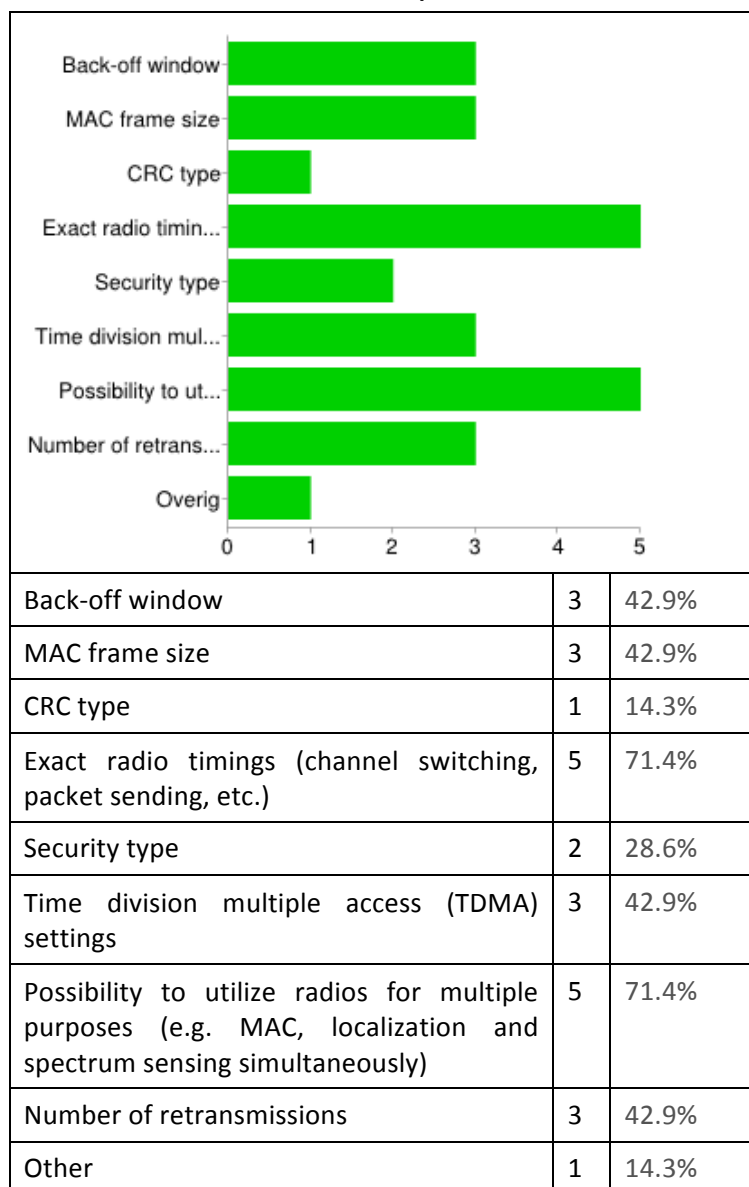
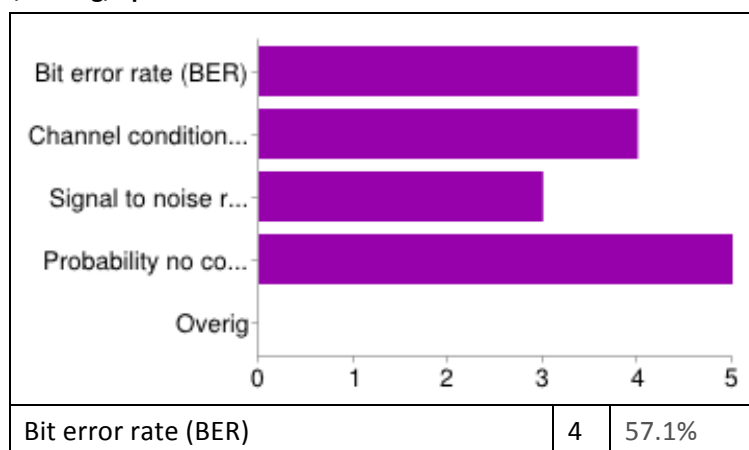


Table 15 Which information from the wireless solution would be most beneficial to obtain during design, validation, testing, optimization or demonstration of wireless solutions? – Physical layer aspects”



| | | |
|---|---|-------|
| Channel conditions / spectrum information (interference, multi-path fading, etc.) | 4 | 57.1% |
| Signal to noise ratio (SNR) | 3 | 42.9% |
| Probability no communication is possible | 5 | 71.4% |
| Other | 0 | 0% |

Table 16 Which information from the wireless solution would be most beneficial to obtain during design, validation, testing, optimization or demonstration of wireless solutions? – MAC layer aspects”

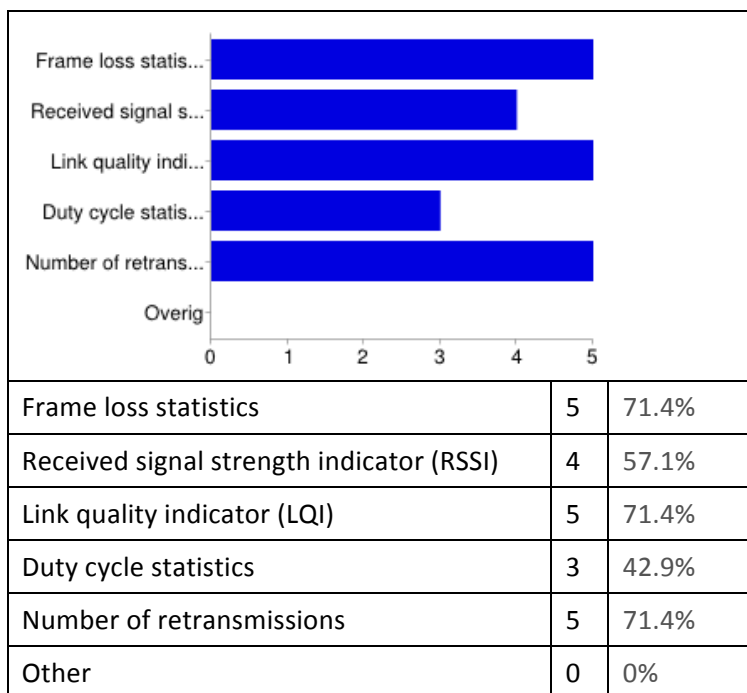


Table 17 Which information from the wireless solution would be most beneficial to obtain during design, validation, testing, optimization or demonstration of wireless solutions? – Network wide aspects”

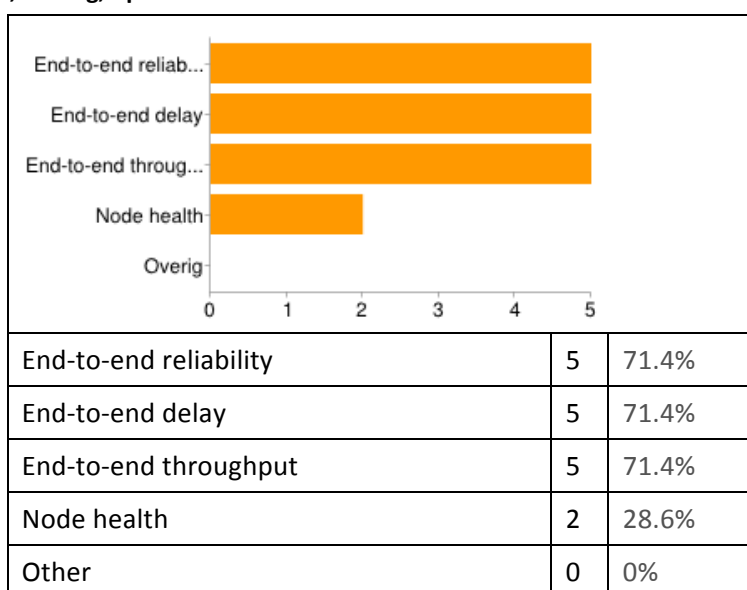


Table 18 What kind of solutions are most interesting for you? - Interference detection and automatic mitigation"

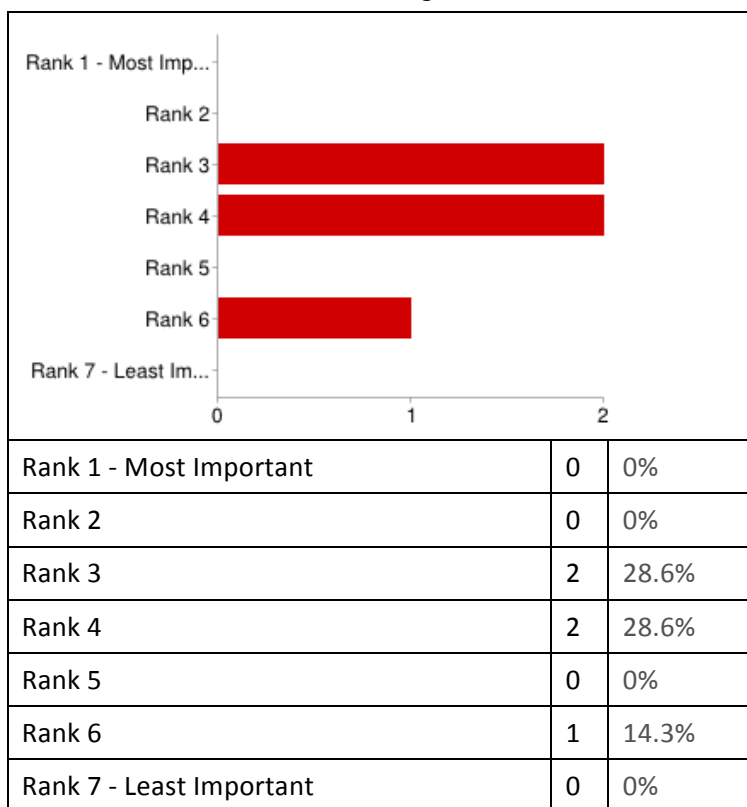


Table 19 What kind of solutions are most interesting for you? - Over the air updating

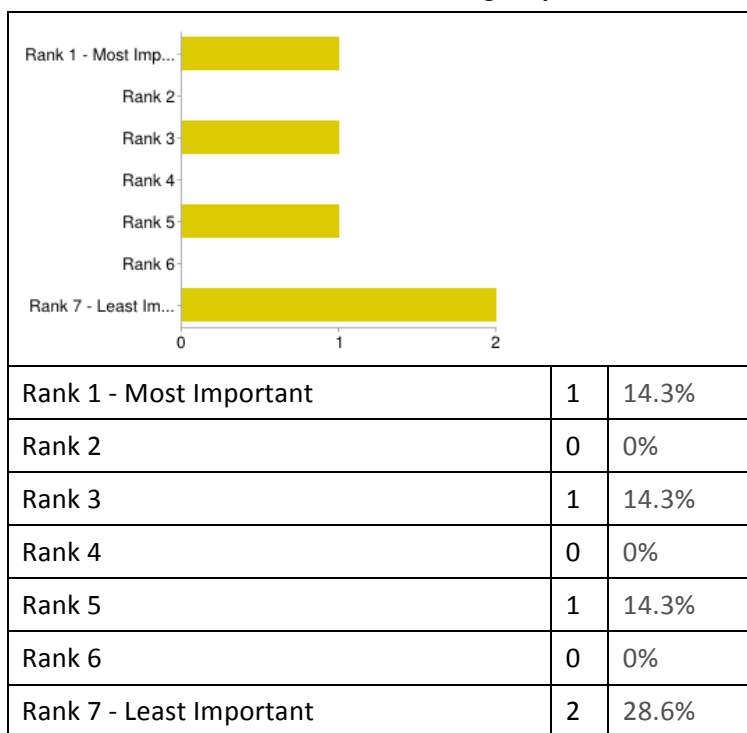


Table 20 What kind of solutions are most interesting for you? - Automatic identification of causes for failing links

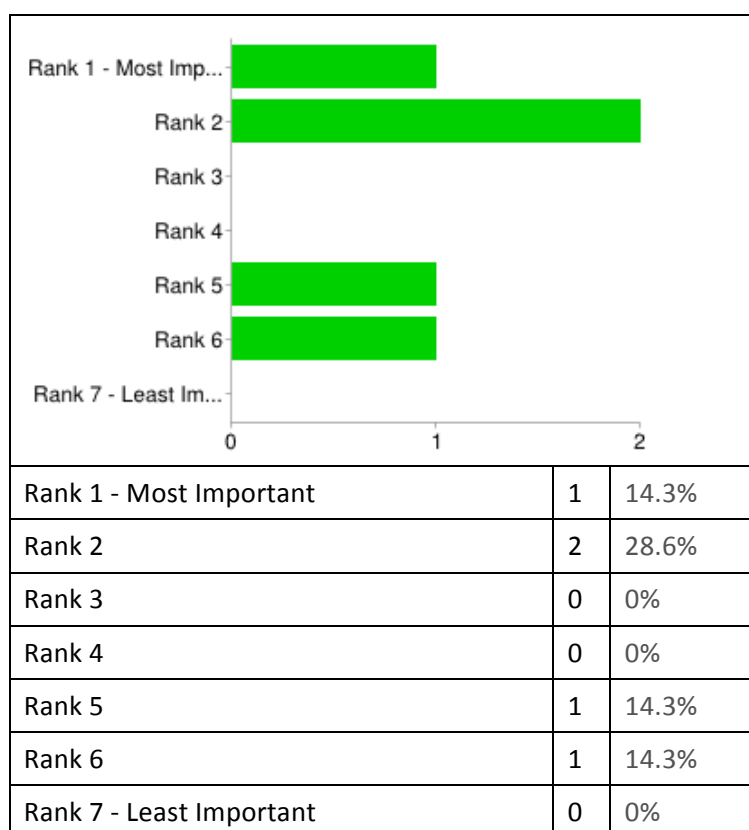


Table 21 What kind of solutions are most interesting for you? - Remote parameter tuning

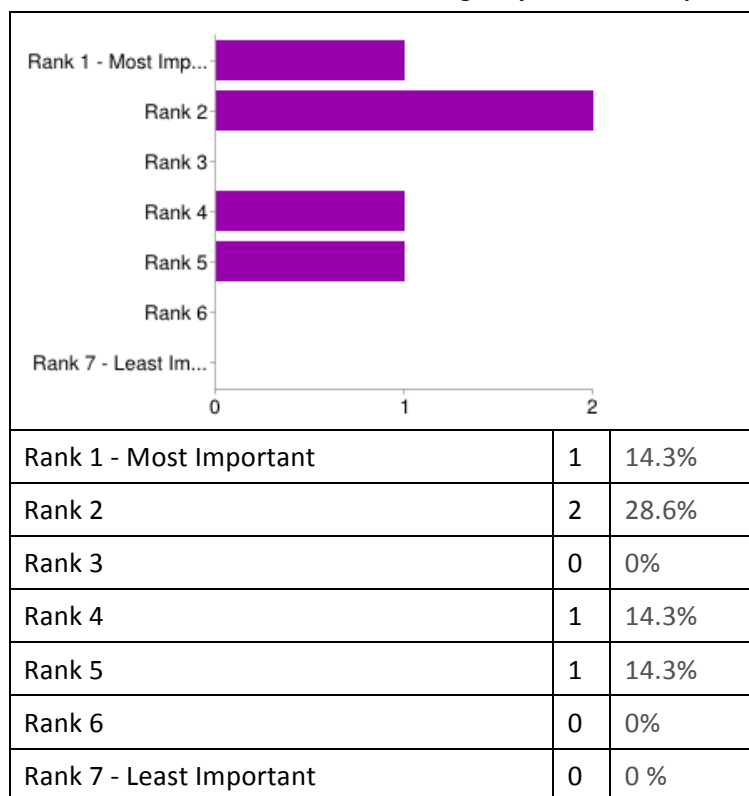


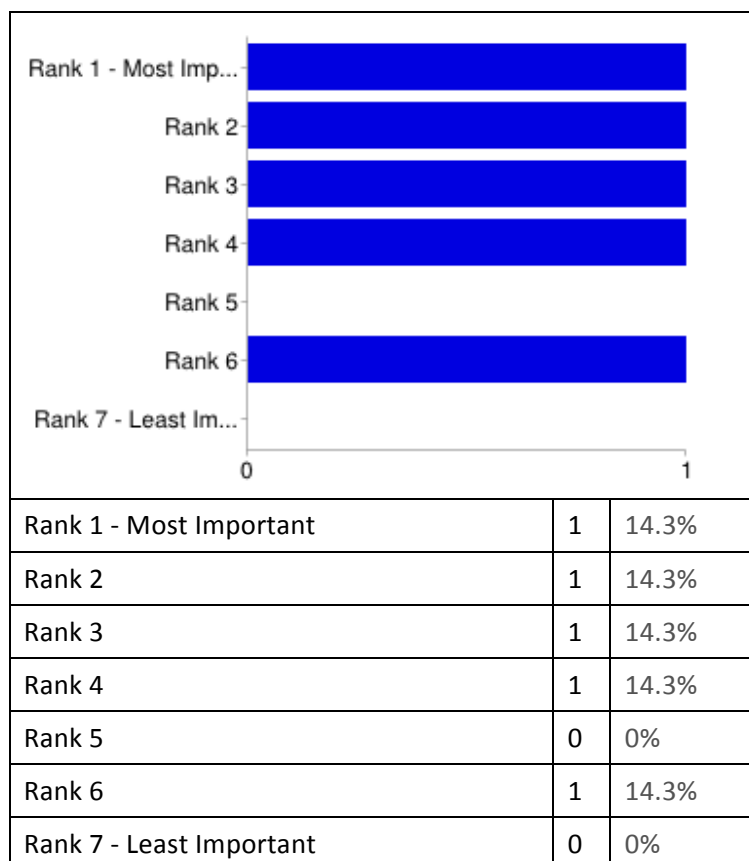
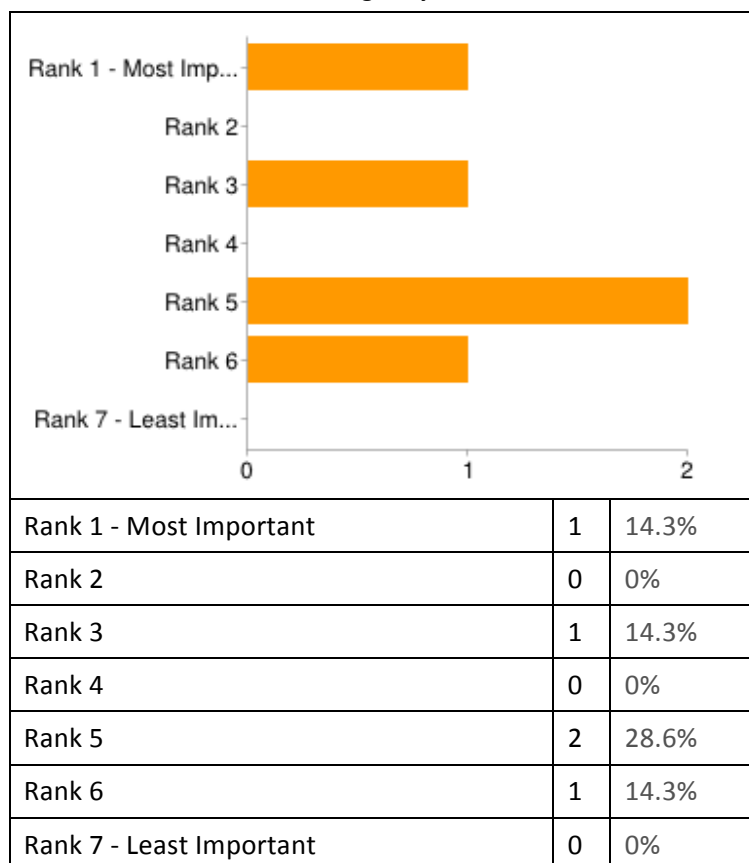
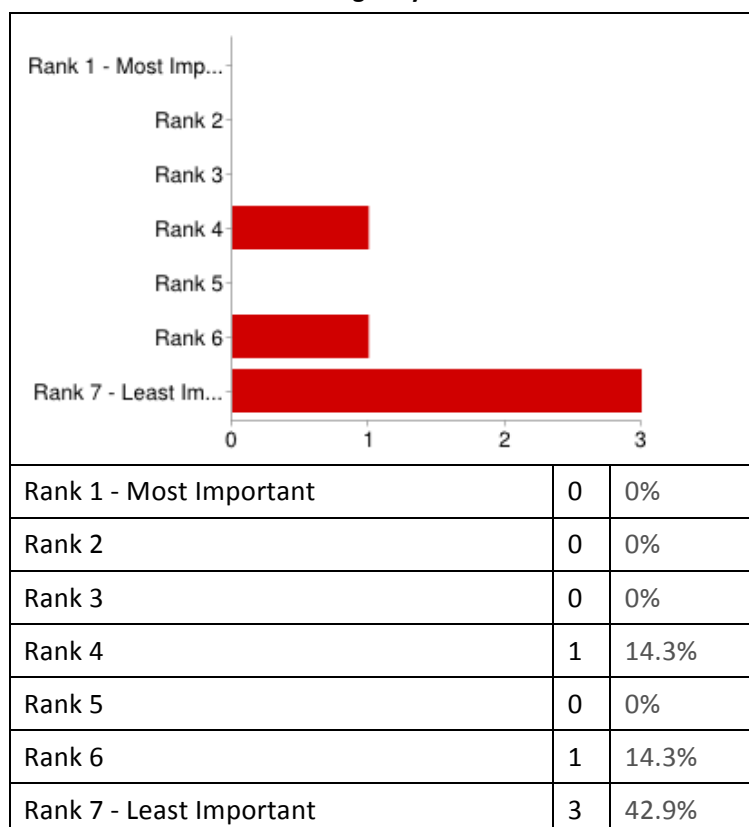
Table 22 What kind of solutions are most interesting for you? - Early warning for malfunctions**Table 23 What kind of solutions are most interesting for you? - Automatic corrections of errors and failures**

Table 24 What kind of solutions are most interesting for you? - Remote collection of low level radio statistics

9.4 Summary of preliminary results

The cost for setting up experiments is often very high, limiting validation of wireless solutions to a fixed small-scale test set-up. Wireless developers further recognize that small-scale testing is a serious limitation. Having no possibilities for stress-testing wireless solutions is also a problem.

Most developers consider non-realistic experimentation conditions and the behavior of the testing environment as minor issues, in contrast with what has been observed by FIRE experimenters. Wireless developers seem to underestimate these issues. However, they further recognize they often lack insight into the behavior of a wireless solution

The lack of configuration flexibility is considered as a more serious issue.

The difficulty to switch to a new technology is a very serious issue for most of the wireless developers. Writing low-level driver and MAC software seems to be a complex task for many wireless developers.

The following aspects would be most **beneficial to control** during development, testing, optimization and demonstration of wireless solutions:

- **Physical layer aspects:**
 - high priority: transmission power, modulation scheme and center frequency
 - medium priority: lesser extent, frequency bandwidth and MIMO
- **Link layer aspects:**
 - high priority: exact radio timing, possibility to utilize radios for multiple purposes
 - medium priority: back-off window, MAC frame size, TDMA settings, number of retransmissions

- low priority: security type

The following **physical layer information** would be most **beneficial to obtain** during design, validation, optimization or demonstration of wireless solution

- high priority: probability that no communication is possible, BER, channel condition/ spectrum information
- medium priority: SNR

The following **MAC layer information** would be most **beneficial to obtain** during design, validation, optimization or demonstration of wireless solution

- high priority: frame loss statistics, received signal strength, link quality indicator, number of retransmissions
- medium priority: duty cycle statistics

The following **network layer information** would be most **beneficial to obtain** during design, validation, optimization or demonstration of wireless solution

- high priority: end-to-end reliability, end-to-end delay, end-to-end throughput
- low priority: node health

The **wireless solutions** that are **most interesting**, in order of decreasing importance, are:

1. Remote parameter tuning
2. Automatic identification of causes for failing links
3. Early warning for malfunctions
4. Interference detection and automatic mitigation
5. Automatic corrections of errors and failures
6. Over the air updating
7. Remote collection of low level radio statistics

These are the preliminary results of the questionnaire and will be updated when more responses are collected and processed.

10 Conclusion

In this deliverable, we have identified a driving scenario that reflects future wireless developments, and aligns well, often overlapping, with the vision for future 5G wireless communications. This scenario (encompassing home, smart city and public event) enables us to identify concrete showcases to be investigated more in depth during the lifetime of the project (to be reported in the upcoming D2.2). We have also defined the conceptual architecture of the WiSHFUL project as well as the required unified programming interfaces (UPIs) and their functional requirements. These will lead to detailed architectural specification and specification of the technical, rather than functional, requirements in the upcoming D3.1 and D4.1. Similar progress has been achieved with respect to the architecture and functional requirements related to the portable testbed concept and will be further developed and reported in D6.2.

Since all the WiSHFUL testbeds are expected to be federated, we have assessed their current status and concluded that w-iLab.t is already full Fed4FIRE compliant, ORBIT is GENI compliant and the remaining three are in some state of Fed4FIRE compliance and will achieve full compliance soon.

Finally, the WiSHFUL project is planning to maintain contact with industrial partners that were consulted during the proposal time, as well as expand the reach and discussions with other partners. Their feedback is very important for prioritizing the research and developments being carried out in the project. As such, the final section of the deliverable reports on the very preliminary results of a questionnaire created to maintain this communication and obtain feedback. It can already be seen from these preliminary results that some of the problems that WiSHFUL addresses, such as offering common API (UPI) and for minimizing the vendor lock-in are very important for SMEs.

11 References

- [1] I. Tinnirello, G. Bianchi, P. Gallo, D. Garlisi, F. Giuliano, F. Gringoli: Wireless MAC processors: Programming MAC protocols on commodity Hardware. IEEE INFOCOM 2012
- [2] Pieter De Mil, Bart Jooris, Lieven Tytgat, Jeroen Hoebeke, Ingrid Moerman¹, Piet Demeester. SnapMac: A generic MAC/PHY architecture enabling flexible MAC design. Ad Hoc Networks, Volume 17, June 2014, Pages 37–59
- [3] Young, Alexander R., et al. "CSERE (Cognitive System Enabling Radio Evolution): A modular and user-friendly cognitive engine." Dynamic Spectrum Access Networks (DYSPAN), 2012 IEEE International Symposium on. IEEE, 2012.
- [4] Fortuna, Carolina, and Mihael Mohorcic. "A Framework for Dynamic Composition of Communication Services." ACM Transactions on Sensor Networks (TOSN) 11.2 (2014): 32.
- [5] De Valck, P., Moerman, I., Croce, D., Guiliano, F., Tinnirello, I., Garlisi, D., De Poorter, E., et al. (2014). Exploiting programmable architectures for WiFi/ZigBee inter-technology cooperation. EURASIP JOURNAL ON WIRELESS COMMUNICATIONS AND NETWORKING, 212, 1–13.
- [6] Description of the federation AM API (available at <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/federation-am-api.html>)
- [7] Slice Federation Architecture 2.0 (available at <http://groups.geni.net/geni/wiki/SliceFedArch> and <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/general.html>)
- [8] GENI Aggregate Manager API Version 3 (available at http://groups.geni.net/geni/wiki/GAPI_AM_API_V3)
- [9] jFed, a Java-based framework for testbed federation (available at <http://jfed.iminds.be/>)
- [10] RSpecs, Resource Specification documents (see <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/rspec.html>)
- [11] ABAC Credentials (see <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/credential-abac.html>)
- [12] SFA Credentials (see <https://fed4fire-testbeds.ilabt.iminds.be/asciidoc/credential-sfa.html>)
- [13] Zabbix monitoring solutions (see <http://www.zabbix.com/>)
- [14] Nagios monitoring solutions (see <http://www.nagios.org/>)
- [15] OML, a generic software framework for measurement collection (see <http://oml.mytestbed.net/projects/oml/wiki/>)
- [16] Federated Resource Control Protocol (see <https://mytestbed.net/projects/omf6/wiki/ArchitecturalFoundation2ProtocolInteractions>)
- [17] OMF, cOntrol and Management Framework (see http://omf.mytestbed.net/projects/omf/wiki/OMF_Main_Page)
- [18] Network Experiment Programming Interface (see <http://nepi.inria.fr>)
- [19] Fed4FIRE First Level support monitor (see <https://flsmonitor.fed4fire.eu/>)
- [20] Fed4FIRE experimenters Google group (see <https://groups.google.com/forum/#!forum/fed4fire-experimenters>)

Annex 1 Industry questionnaire



WISHFUL questionnaire

The purpose of this questionnaire is to identify the most pressing needs from the industry with the goals of (i) prioritizing the WISHFUL research to be in line with the industry requirements and (ii) derive generic research scenarios and proof-of-concepts that will demonstrate the benefits of WISHFUL in light of the industry needs.

All questions have an optional text field that can be used for clarification purposes or to add additional remarks.

The Wireless Software and Hardware platforms for Flexible and Unified radio and network control (WiSHFUL) project will lower the wireless experimentation threshold by developing flexible, scalable, open software architectures and programming interfaces for prototyping novel wireless solutions for a variety of applications ranging from healthcare to smart cities, supporting players in high value-add markets with considerable growth potential. Key features of WiSHFUL include (1) unified radio control, providing developers with deep control of physical and medium access components without requiring deep knowledge of the radio hardware platform and (2) unified network control allowing the rapid creation, modification, and prototyping of protocols across the entire stack. WiSHFUL will also create a testbed-on-the-move, consisting of portable facilities that can be deployed easily and efficiently at any location, allowing validation of innovative wireless solutions in the real world (with realistic propagation and interference characteristics) and involving real users.

Please provide your name

Please provide your e-mail address

Continue »



WISHFUL questionnaire

PART 1: Gaining insight in the activities of the industry partner

Question 1: Please give the name and website of your company or institution

Question 2: Please indicate your primary type(s) of activity.

Multiple can be selected

- ☐ Radio Chip Designer
- ☐ Radio Chip Manufacturer
- ☐ Radio Chip Integrator
- ☐ Installer of networked products
- ☐ Operator of networked products
- ☐ System Integration
- ☐ End user
- ☐ Policy maker
- ☐ Research
- ☐ Other:

Question 3: What is your main target market?

- ☐ Broad market
- ☐ Niche markets

« Back

Continue »



50% completed

PART 2: Current development practices

Question 4: Describe example activities/use cases/challenges related to your wireless network activities

Please provide a link to a website or additional information

Question 5: Which communication technologies are interesting for your activities?

- ☐ IEEE 802.11 a/b/g/n/ac (ISM)
- ☐ IEEE 802.11 af (TV white space)
- ☐ IEEE 802.11 ah (sub GHz)
- ☐ IEEE 802.15.4 (Zigbee, WirelessHART, Thread, ...)
- ☐ IEEE 802.15.1 (Bluetooth)
- ☐ Weightless (TV white space)
- ☐ LoRa (sub GHz)
- ☐ SigFox (sub GHz)
- ☐ 60 Ghz based
- ☐ Satellite based
- ☐ 3G
- ☐ LTE
- ☐ Software Defined Radio (SDR)
- ☐ Other:

Question 6: Which problems do you encounter during the (re)design of new wireless solutions

- ☐ Hardware obsolescence (being forced to switch to new radio chip due to short radio chip availability)
- ☐ Vendor lock-in (too costly/time consuming to switch to a new radio chip vendor or manufacturer)
- ☐ Long time to market
- ☐ Limited customization options (fixed radio behavior)
- ☐ Chip dependent code (several software / hardware aspects of you solution depend strongly on a specific chip)
- ☐ Unpredictable radio behavior (due to bugs, incomplete documentation, missing features, ...). Please specify below.
- ☐ Other:

Question 7: How do you currently develop, test, stress-test or demonstrate wireless solutions?

- ☐ Using theoretical tools (such as matlab, ...)
- ☐ Using a simulator
- ☐ Single link experiments (communication between 2 devices)
- ☐ Fixed small-scale experiment set-up (< 10 testing nodes)
- ☐ Fixed large-scale experiment set-up (>=10 testing nodes)
- ☐ Custom experiment setup for each new test phase
- ☐ In-situ testing
- ☐ Other:

Question 8: Which problems do you currently encounter during the development, testing, optimization or demonstration of wireless solutions?

8a: Please rate the seriousness of the following problems

| | Very Serious | Somewhat Serious | Not Serious |
|---|-----------------------|-----------------------|-----------------------|
| Non-realistic experimentation conditions | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Not enough configuration flexibility | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Not enough insight into the behavior of the solution | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Not enough insight into the behavior of the environment | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Too costly to set-up experiments | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Too small scale experiments | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Complexity of writing low-level driver and MAC software | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Lack of code reuse at driver and MAC level | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Difficult to switch towards a new technology | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| No stress-test possibilities | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

PART 3: Identifying the technical focus points for WISHFUL

Question 9: Which aspects of a wireless solution would be most beneficial to control during development, testing, optimization or demonstration of wireless solutions?

9a: Physical layer aspects.

- ☐ Transmission power
- ☐ Modulation scheme
- ☐ Center frequency
- ☐ Frequency bandwidth
- ☐ MIMO
- ☐ Other:

9b: link layer aspects.

- ☐ Back-off window
- ☐ MAC frame size
- ☐ CRC type
- ☐ Exact radio timings (channel switching, packet sending, etc.)
- ☐ Security type
- ☐ Time division multiple access (TDMA) settings
- ☐ Possibility to utilize radios for multiple purposes (e.g. MAC, localization and spectrum sensing simultaneously)
- ☐ Number of retransmissions
- ☐ Other:

Question 10: Which information from the wireless solution would be most beneficial to obtain during design, validation, testing, optimization or demonstration of wireless solutions?

10a: Physical layer aspects.

- ☐ Bit error rate (BER)
- ☐ Channel conditions / spectrum information (interference, multi-path fading, etc.)
- ☐ Signal to noise ratio (SNR)
- ☐ Probability no communication is possible
- ☐ Other:

10b: MAC layer aspects.

- ☐ Frame loss statistics
☐ Received signal strength indicator (RSSI)
☐ Link quality indicator (LQI)
☐ Duty cycle statistics
☐ Number of retransmissions
☐ Other:

10c: Network wide aspects

- ☐ End-to-end reliability
☐ End-to-end delay
☐ End-to-end throughput
☐ Node health
☐ Other:

Question 11: What kind of solutions are most interesting for you?

11a: Please rank the importance of the options

| | Rank 1 - Most Important | Rank 2 | Rank 3 | Rank 4 | Rank 5 | Rank 6 | Rank 7 - Least Important |
|--|-------------------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|--------------------------------|
| Interference detection and automatic mitigation | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Over the air updating | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Automatic identification of causes for failing links | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Remote parameter tuning | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Early warning for malfunctions | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Automatic corrections of errors and failures | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |
| Remote collection of low level radio statistics | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> | <input type="radio"/> |

11b: Please indicate other solutions you would like to see: