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

Project Deliverable D7.1

Support of Third Party extensions in Year 2

Contractual date of delivery:	31-12-2016
Actual date of delivery:	23-12-2016
Beneficiaries:	CNIT, IMEC, TCD, TUB
Lead beneficiary:	CNIT
Authors:	Pierluigi Gallo (CNIT), Domenico Garlisi (CNIT), Ilenia Tinnirello (CNIT), Ingrid Moerman (IMEC), Peter Ruckebusch (IMEC), Pieter Becue (IMEC), Anatolij Zubow (TUB), Mikołaj Chwalisz (TUB), Anatolij Zubow (TUB), Diarmuid Collins (TCD)
Reviewers:	Spilios Giannoulis (IMEC)
Work package:	WP7 – Testbed extensions
Estimated person months:	4.5
Nature:	R
Dissemination level:	PU
Version:	1.0

Abstract:

This deliverable provides a description of activities done during year 2 for supporting extensions made by third parties to the WiSHFUL testbed facilities.

Keywords:

Testbed extensions, hardware, software, support

Executive Summary

This deliverable summarizes the activities run by WiSHFUL consortium while supporting third parties testbed extensions. **WiSHFUL capabilities have been extended with new hardware and software modules thanks to the open call contributions and to the support given by patrons.** In particular, the extensions considered during Y2 activities fall within the open call 1. These have been provided by Adant, which extended the testbed to support **reconfigurable antenna systems**, the University of Thessaly, which provided support for **LTE Experimentation,** and the University of Perugia, which provided modules to support **software radio transmitters for DVB-T.**

The focus of this document is on the support given by the WiSHFUL Consortium, to all phases of the testbed extension, from the idea till the validation. Brief descriptions of the activities for the design, implementation and deployment of the extensions are included in this document to make clear the context of these activities. Mainly patrons supported the testbed extenders first in providing details about the platforms, the WiSHFUL control framework and its related tools; then during the integration of the new hardware and software modules, the patrons also helped in the design of the extra UPIs and modules, as well as on the functional validation of the new functionalities. Phone calls and face-to-face meetings helped especially during phases of hardware integration and the overall validation of the extensions. The time investment that patrons spent for tutoring and support activities is also detailed quantitatively by the mean of timetables.

List of Acronyms and Abbreviations

BB	Broadband
CPU	Central Processing Unit
DVB-T	Digital Video Broadcasting — Terrestrial
GNU	GNU's Not Unix!
MIMO	Multiple Input Multiple Output
OC1	Open Call 1
OFDM	Orthogonal frequency-division multiplexing
OS	Operating System
RAS	Reconfigurable Antenna System
SDR	Software Defined Radio
UHF	Ultra High Frequency
UPI	Unified Programming Interface
UPI_R	Unified Programming Interface – Radio
USB	Universal Serial Bus

Table of contents

1 Introduction

This document introduces the activities performed by third parties selected for extending the WiSHFUL testbeds. From the 14 proposals that applied for extensions, three were chosen. Their activities were mainly conducted during the second year of the project and required specific support from the WiSHFUL Consortium. Patrons have been assigned to each third party, accordingly to their expertise in the hardware and software components to be extended.

The designated testbed extensions enrich WiSHFUL functionalities in different directions including programmable antennas, LTE and DVB-T.

The first extension introduces reconfigurable antenna systems and their controlling software. These modules increase the flexibility of several WiSHFUL nodes with **programmable radiation patterns,** which provide better performance when the antenna configuration is opportunely tuned. The second extension enhances WiSHFUL adding the functionalities offered by the NITOS testbed. These allow **LTE experimentation**, as for example to realize showcases like the LTE to Wi-Fi offloading. The third extension enables **digital video broadcasting in terrestrial scenarios**, permitting the IRIS platform to work as a digital video broadcaster.

The remainder of this document is organized in three sections, one for each testbed extension. Third-parties' activities are logically grouped in order to report the work dedicated for extending the UPIs, for the definition and implementation of hardware and software components as well as their deployment and setup. Finally, special emphasis was placed on validation procedures for demonstrating the novel functionalities and their integration with the pre-existing facilities. Patrons have facilitated all these actions, providing their expertise in the WiSHFUL framework, its architecture, the available platforms, and the UPI functions. In each of these three sections, one paragraph is dedicated to the description of the assistance provided by the patron, reporting both quantitative and qualitative indicators.

2 RAS - Reconfigurable Antenna System for WLAN platforms

This WiSHFUL extension has been provided by Adant and has been supported by CNIT.

2.1 Extension short description

The RAS extension aimed to extend WiSHFUL capabilities by adding electronically reconfigurable antennas, which are capable of dynamically changing their radiation characteristics. Reconfigurable antennas and cognitive radios are drivers for advanced sensing and adaption to the context. The RAS extension fills the lack of smart antennas, which were not available in WiSHFUL testbeds until now. The functionalities added by this extension allow experimenters to use novel antenna technologies and create directive links by means of novel control functions, specifically designed to drive the reconfigurable antenna systems. This is feasible now thanks to the capability of radiating the energy in different directions. This extension paves the way to new wireless solutions for wireless networks, creates potential for **performance improvements**, **new testbed topologies** and support for **dedicated applications**, such as localization and physical layer security.

2.1.1 UPI extension

The RAS testbed extension enriched the WiSHFUL UPIs for controlling the RAS configuration. The WiSHFUL project offers flexibility for extending UPIs to support and control reconfigurable antenna systems within the WiSHFUL testbed facilities. The RAS extension required a new function in the Unified Programming Interface – Radio (UPI$_R$).

controller.radio.set_sas_conf(band, conf_ant1, conf_ant2, conf_ant3, conf_ant4): this function is used in order to modify the direction of the radiation pattern of four antennas. In fact, the controller can drive up to four radiating elements, which are piloted through the configuration value, an integer between 0 and 8, 0 for the omnidirectional pattern and 1-8 for the eight directional configurations. The band parameter indicates the frequency band of the radiation pattern to drive, which is 2 for 2.4 GHz and 5 for 5 GHz.

2.1.2 Hardware and Software components

The RAS extension provides both the reconfigurable antennas, the hardware that has been integrated in the testbed and the controlling tools, the software modules in the WiSHFUL framework. The hardware includes RAS radiating elements, the control circuit for changing their configuration, the USB/serial/GPIO interface for control and the enclosing box. The software includes the module developed to extend the existing WiSHFUL framework with the necessary UPIs to control the reconfigurable antenna system from the WISHFUL framework.

2.1.3 Deployment and setup

The RAS extension delivered 48 pole mountable boxes with 48 controllers, most with USB interface and few equipped with serial interface. The provided RAS radiating elements work at both 2.4GHz and 5GHz.

2.1.4 Validation

The programmability of the RAS system has been validated by CNIT through dedicated tests that used the UPI interface from within the WiSHFUL framework. By measuring the received power at sniffing nodes located at specific points around the transmitting node, CNIT validated the effectiveness of the directional configuration. This was changing periodically following the defined configuration policy for directing the RAS. An additional validation was obtained by tuning the transmission power in the RAS enabled system, demonstrating that directional configurations perform better than the omnidirectional one, when the main direction of radiation is opportunely chosen.

2.1.5 Impact on existing WiSHFUL testbeds and Fed4FIRE integration

The RAS extension didn't require specific changes in the WiSHFUL testbeds, since it provided enough flexibility for extending its capabilities and the required hardware interfaces for the RAS system were natively available on WiSHFUL pre-existing hardware (USB and GPIO).

2.2 Support given by the WiSHFUL Consortium – Patron: CNIT

Preliminaries

Preliminary actions for supporting the RAS extension were focused on the definition of key capabilities offered by this hardware extension (interface, number of directional beams, etc.), the performance metrics to be used for validation (throughput, packet loss, …) and some logistics for deployment (position in the testbed, enclosure, …).

The WiSHFUL framework

The patron prepared a presentation to explain the behaviour of the WiSHFUL framework v 1.0, then several live trials were done to explain the whole workflow. After the Consortium decided to switch to version 2.0 of the WiSHFUL framework, the patron shared this new vision and workflow with the RAS team, discussing also the impact on the on-going activities.

Interfaces with pre-existing WiSHFUL components

The interface for controlling RAS antennas has both hardware and software components. The patron discussed details about the hardware interface. In particular it has been discussed to have both USB and serial interfaces to control the behaviour of the smart antennas.

Interoperability between RAS hardware and software components and the boxes available in WiSHFUL testbeds have been studied, (connectors, enclosures, and interfaces as well as OS, drivers, and firmware).

Extending the UPI

The design of the novel UPI functions to control the reconfigurable antenna systems has been coordinated by the patron in order to ensure compatibility and avoid overlaps with what was already available.

Logistics

The patron helped in monitoring progress during the design and development process, providing suggestions and details in case of doubts on capabilities as well as preparing descriptions of the intermediate status. Intermediate reports have been presented by the patron during the plenary meeting held in Palermo, and in Ghent, based on the work provided by the proponent. The number of nodes to be prototyped and deployed on the WiSHFUL testbed, as well as their detailed features have been discussed between the patron and the Adant team, including also practical aspects such as the enclosure to be used for mounting the antennas on the poles.

Validation

The project proponent has been helped by the patron regarding the definition of the best strategies and metrics to validate the new functionalities offered by the testbed extension. In particular, the RX power and the throughput have been defined as key performance indexes to validate improvements offered by the programmable antennas, as well as the speed of the control interface to pilot the antenna behavior.

Table 1 - Time sheet of the support dedicated to the RAS project

Date	Dedicated Time [hour]	Topic
25/03/2016	2	Call to discuss the main action point addressed by RAS extension (bandwidth, interface, enclosure, number of directional beam)
30/03/2016	2	Preparation of the presentation about the wishful framework
04/04/2016	2	Call to plan the developing of RAS driver (USB/Serial)
04/06/2016	2	Call to introduce the framework to ADANT development team in order to insert the RAS module in the WiSHFUL framework
18/06/2016	2	Call to resolve some issue regarding the system in which the RAS module run
28/06/2016	2	Call to introduce new framework 2.0
25/05/2016	2	Call to resolve some issue regarding the new framework 2.0, and discussion about the UPI developed to control the antenna
06/07/2016	2	Preparation of the presentation with the current state of implementation for the WiSHFUL plenary meeting (Palermo)
15/06/2016	2	(Call) According to the meeting feedback, we converge on number of nodes to produce and design for the enclosure. (We addressed an easy installation on testbed)
25/07/2016	8	Preliminary testing of the antenna prototype, to evaluate different configuration (we evaluate Throughput and RX power level for each configuration)
09/05/2016	2	Call to report the current state of implementation of the antenna and enclosure (deliverable of the radiation diagrams)
10/05/2016	3	Prepare presentation to report current state of developing on plenary meeting (Gent) - Search correct connection cable in order to install the antenna
17/10/2016	2	Test the antenna prototype installed on w-ilab2 testbed
18/10/2016	2	Test and push on github the agent module developed by ADANT to control the antenna
11/10/2016	40	Support to ADANT on the developing of the review demo
TOTAL	**75**	

2.3 Documentation and examples for experimenters

The RAS module is available on the github WiSHFUL project [1] and demo scripts for experimenters are available in [2].

3 FIRE LTE EXperimentation using WiSHFUL

This WiSHFUL extension has been provided by the University of Thessaly and has been supported by IMEC.

3.1 Extension short description

Recent efforts of the research community are concentrating on the standardization of a 5G protocol suite solution that will offer enhanced network coverage and capacity via allowing ultra-dense cell deployments for increased connection availability, lower end-to-end latency, while also integrating existing legacy network deployments. In this context, 4G technologies, will be in the fore for at least another decade through the constant amendments to the LTE standard, currently the key 4G enabler solution. LTE has penetrated the global market as the key 4G solution, able to deliver speeds ranging from 100Mbps to over 1Gbps per cell. In this environment, validation under real-world settings of research algorithms and protocols enabling enhanced end-user experience are of paramount importance. The purpose of the FLEXFUL extension is to offer LTE experimentation equipment located in the NITOS testbed and provide the necessary software modules in order to make feasible the control of the LTE equipment from WiSHFUL. UTH has extended the offered WiSHFUL facilities, by providing its Fed4FIRE compatible, LTE-enabled testbed NITOS, along with the corresponding software extensions to the WiSHFUL Unified Programming Interfaces (UPIs) for configuring and controlling the provided LTE resources. Subsequently, the UPI framework has been appropriately tailored in order to appropriately expose the configuration parameters of the LTE equipment (Base Stations, EPCs, User Equipment), and enable their combination with the rest of the supported resources in a unified fashion. The delivered framework has been evaluated by the execution of an LTE to WiFi offloading experiment, making use of all the extensions developed in FLEXFUL. Finally, UTH will offer a strong commitment regarding the availability of the NITOS LTE testbed as part of the WiSHFUL facilities even after the end of FLEXFUL.

3.1.1 UPI extension

The following list is populated with the new UPIs that are available after the FLEXFUL extension.

a. ***Unified Programming Interface – Radio (UPI_R)***

GetTrackingAreaCode, []	for getting the TAC of the LTE network
SetTrackingAreaCode, [Area_Code]	for setting the TAC of the LTE network
GetPlmnId, []	for getting the PLMNID that is used
SetPlmnId, [plmn]	for setting the PLMNID that is used
GetENBName, []	for getting the eNB name
SetENBName, [name]	for setting the eNB name
GetEci, []	for getting the CellID parameter
SetEci, [cell_id]	for setting the CellID parameter
GetENBType, []	for getting the eNB type
SetENBType, [0/1]	for setting the eNB type (0 is for home, 1 for macro)
GetMaxERABperUE, []	for getting the maximum Radio Access Bearers per UE
SetMaxERABperUE, [num_ERABS]	for setting the maximum Radio Access Bearers per UE
GetPUSCHPowerControl, []	for checking the Power Control on the PUSCH channel
SetPUSCHPowerControl, [0/1)]	for turning on/off the Power Control on the PUSCH channel
GetPDCCHPowerControl, []	for checking the Power Control on the PDSCH channel
SetPDCCHPowerControl, [0/1]	for turning on/off the Power Control on the PDSCH channel
GetSINRPUCCHPowerControl, []	for checking the SINR Power Control on the PUCCH channel
SetSINRPUCCHPowerControl, [0/1]	for turning on/off the SINR Power Control on the PUCCH channel
GetHARQPUCCHPowerControl, []	for checking the HARQ Power Control on the PUSCH channel
GetFreqPUSCHPowerControl, []	for checking the Power Control on the PUSCH channel for HARQ
SetFreqPUSCHPowerControl, [0/1]	for turning on/off the Power Control on the PUSCH channel for frequency selection

GetPUCCHSINRTarget, []	for checking the PUCCH SINR target
SetPUCCHSINRTarget, [int]	for setting the target SINR for the PUCCH channel
GetPUCCHBLERTarget, []	for checking the target BLER for PUCCH channel
SetPUCCHBLERTarget, [int]	for setting the target BLER for the PUCCH channel
GetFreqBandIndicator, []	for getting the Frequency Band
SetFreqBandIndicator, [7/13]	for setting the Frequency Band (ip.access femtocells support bands 7 and 13)
GetEarFcnDl, []	for checking the EARFCN for the DL channel (center frequency)
SetEarFcnDl, [3100/5230]	for setting the EARFCN for the DL channel
GetEarFcnUl, []	for checking the EARFCN for the UL channel (center frequency)
SetEarFcnUl, [21100/23230]	for setting the EARFCN for the UL channel
GetDlBandwidth, []	for getting the bandwidth used for the DL channel
SetDlBandwidth, [2/3]	for setting the bandwidth used for the DL channel (5/10 MHz)
GetUlBandwidth, []	for getting the bandwidth used for the UL channel
SetUlBandwidth, [2/3]	for setting the bandwidth used for the UL channel (5/10 MHz)
GetPhyCellID, []	for getting the Physical Cell ID
SetPhyCellID, [0-503]	for setting the Physical Cell ID
GetPBCHPowerOffset, []	for getting the power offset for the PBCH channel
SetPBCHPowerOffset, [-350 to150]	for setting the power offset for the PBCH channel
GetPSCHPowerOffset, []	for getting the power offset for the PSCH channel
SetPSCHPowerOffset, [-350 to150]	for setting the power offset for the PSCH channel
GetSSCHPowerOffset, []	for getting the power offset for the SSCH channel
SetSSCHPowerOffset, [-350 to 150]	for setting the power offset for the SSCH channel
GetRefSignalPower, []	for getting the transmission power of the cell
SetRefSignalPower, [-15 to (-26)]	for setting the transmission power of the cell
GetNumOfRACHPreambles,[]	for getting the number of RACH preambles
SetNumOfRACHPreambles, [0 to 64]	for setting the number of RACH preambles
GetTxMode, []	for getting the transmission mode
SetTxMode, [0/1/2]	for setting the transmission mode
GetMCSDl, []	for getting the MCS used for the DL channel
SetMCSDl, [0 to 28]	for setting the MCS used for the DL channel
GetMCSUl, []	for getting the MCS used for the UL channel
SetMCSUl, [0 to 26]	for setting the MCS used for the UL channel
GetAdminState, []	for checking whether the cell is radiating
SetAdminState, [0/1]	for turning on/off the cell
GetCQIReport, []	for checking whether CQi reporting is enabled
SetCQIReport, [0/1]	for enabling/disabling the CQi reporting
GetUEReport, []	for checking whether the UE reporting is enabled
SetUEReport, [0/1]	for enabling/disabling the UE reporting
GetUeInactivityTimer, []	for checking the UE inactivity timer
SetUeInactivityTimer, [0/1]	for turning on/off the UE inactivity timer
GetCipherAlgo, []	for checking the ciphering algorithm used
SetCipherAlgo, [EEA0/,128-EEA1,128-EEA2]	for setting the ciphering algorithm used
Restart, []	for restarting the cell
UEAttach, []	for attaching a UE to the LTE network
UEDetach, []	for detaching a UE from the LTE network
UEActivate, []	for activating the UE connection within a PDN
UEDeactivate, []	for deactivating the UE connection within the PDN

b. Unified Programming Interface – Network (UPI_N)

LoadConfigEpc , [	for reverting EPC to its default settings
GetSystemStatus, [	for checking the EPC status
RestartSystem, ['service'	for restarting the EPC

GetAttachedSubscriberCount, [	for getting the number of attached subscribers
GetActiveSubscriberCount, [	for getting the number of active subscribers
GetSubscriberStatus, [imsi	for querying the status of a specific subscriber
DetachUE, [imsi	for detaching a subscriber
ExportDataUsage, ['imsi', 'apn', 'date_year', 'date_month', 'date_day'	for exporting data usage statistics of a specific client
ExportAllDataUsage, [	for exporting all data usage statistics
DeleteAllDataUsage, [	for deleting all data usage statistics
GetLogs, [	for retrieving logs for the EPC
AddSubscriber, ['imsi', 'msisdn', 'k', 'opc', 'subscriberStatus', 'primaryProfile','primaryPDNAllocationType',	
'primaryPDNIPAddress', 'secondaryProfile','secondaryPDNAllocationType','secondaryPDNIPAddress'	for adding a subscriber to the EPC
GetSubscriberList, [	for getting alist with all the EPC subscribers
GetSubscriber, ['imsi'	for getting information about a specific subscriber
UpdateSubscriber, ['imsi', 'msisdn', 'k', 'amf', 'op', 'subscriberStatus', 'primaryProfile', 'primaryPDNAllocationType',	
'primaryPDNIPAddress', 'secondaryProfile', 'secondaryPDNAllocationType', 'secondaryPDNIPAddress'	for updating information of a specific subscriber
DeleteSubscriber, [imsi	for deleting a specific subscriber
AddSubscriberProfile, ['name', 'apn', 'qci', 'arp', 'uplinkAMBR', 'downlinkAMBR'	for adding a new subscriber profile to
the EPC	
GetSubscriberProfileList, [	for getting a list with all the active subscriber profiles
GetSubscriberProfile, ['name'	for getting information about a specific subscriber profile
UpdateSubscriberProfile, ['name', 'apn', 'qci', 'arp', 'uplinkAMBR', 'downlinkAMBR'	for updating a specific subscriber profile
DeleteSubscriberProfile, ['name']	for deleting a subscriber profile
AddENodeB, ['eNodeBID', 'ipAddress']	for adding an eNB to the EPC
GetENodeBList, []	for getting a list with all the added eNBs to the EPC
GetENodeB, ['eNodeBID']	for retrieving information about a specific eNB
UpdateENodeB, ['eNodeBID', 'ipAddress']	for updating the information of a specific eNB
DeleteENodeB, ['eNodeBID']	for deleting a specific eNB
AddAPN, ['name', 'gatewayIPAddress', 'gatewaySubnet', 'primaryDNS', 'secondaryDNS', 'startIPRange',	
'endIPRange','mtu']	for adding a new APN
GetAPNList, []	for getting a list with all the available APNs
GetAPN, ['name']	for getting information about a specific APN
UpdateAPN, ['name', 'gatewayIPAddress', 'gatewaySubnet', 'primaryDNS', 'secondaryDNS', 'startIPRange', 'endIPRange', 'mtu']	for updating the information of a specific APN
DeleteAPN, ['name']	for deleting an APN
GetHSSConfiguration, []	for getting the HSS configuration
SetHSSConfiguration, ['diameterIPAddress']	for setting the diameter address to be used by the HSS
GetMMEConfiguration, []	for getting the configuration of the MME
SetMMEConfiguration, ['name', 'gtpcIPAddress', 's1apIPAddress', 'diameterIPAddress', 'gummei', 'mmeCode', 'mmeGroupID']	for setting the configuration of the MME

GetPGWConfiguration, []	for getting the configuration of the PGW
SetPGWConfiguration, ['gtpuIPAddress', 'gtpcIPAddress', 'ambrDownlink', 'ambrUplink']	for setting the configuration of the PGW
GetSGWConfiguration, []	for getting the configuration of the SGW
SetSGWConfiguration, ['gtpuIPAddress', 'gtpcIPAddress']	for setting the configuration of the SGW
GetGlobalConfiguration, ['name']	for getting the global configuration parameters
SetGlobalConfiguration, ['name, 'value']	for setting the global configuration parameters
GetEpcIpAddress, []	for setting the IP address of the remote MME
GetLocalSctpPort, []	for getting the SCTP port used for the S1AP
SetLocalSctpPort, [port_num]	for setting the SCTP port used for the S1AP
SetEpcIpAddress, []	for setting the IP address of the remote MME
GetPgwIpAddress, []	for getting the PGW IP address
SetPgwIpAddress, [ip]	for setting the PGW IP address
UEAPN, []	for setting a new PDN context on the UE
UEIPaddress, []	for the UE requesting an IP address from the LTE network

3.1.2 HW/SW components

The core of the LTE network is the server side equipment that creates the cells and connects the end user devices to the backbone network and services. The commercial equipment is consisting of 2 small cells by ip.access, SIRRAN's LTEnet EPC, and several LTE USB dongles, configurable via AT-commands and LTE-enabled smartphones.

The Open Source equipment consists of several Software Defined Radio (SDR) front-ends, compatible with the OpenAirInterface (OAI) platform. OpenAirInterface is providing the full LTE stack, configurable as either an eNodeB or a UE, and can run on commodity hardware equipped with a compatible RF front-end. Moreover, Open Source Core Network installations are also present in NITOS, by using the OpenAirInterface-CN extensions.

A wide variety of different LTE dongles, configured differently are present in the NITOS testbed. These consist of LTE dongles, provided by Huawei or ZTE, which are configurable via standardized interfaces, such as AT-Commands or the QMIcli. It is also worth mentioning, that vendor specific versions of firmware that might restrict the experimentation, such as Huawei HiLink that sends all traffic through a NAT process, have been re-flashed by the NITOS team. The Public Land Mobile Network Identifiers (PLMNID) used in the NITOS testbed, are broken down into two groups:

- Dongles with IMSIs for the PLMNID 46099 (NITOS PLMN)
- Dongles with IMSIs for the PLMNID 20398 (Default PLMN for OpenAirInterface - OAI)

Moreover, the testbed is equipped with several mobile phones, all of them running Android, that can be used as UEs. Their configuration takes place over the Android Debug Bridge (ADB), as they are mounted on some NITOS nodes. The complete list of phones is:

- Google Nexus 6
- Google Nexus 5
- Samsung Galaxy I9500 S4
- Samsung Galaxy I9190 S4 mini

3.1.3 Deployment and setup

All of the specific hardware is located in the NITOS testbed that will be offered to experimenters for the remaining duration of WiSHFUL as a fully supported fed4FIRE testbed. Nevertheless the hardware used is also available in wilab-t, Gent so the extension can be deployed and used there as well. This would enable the possibility of conducting LTE and WiFi-ZigBee coexistence experiments in wilab-t using only the WiSHFUL framework to setup and execute such a complex experiment.

3.1.4 Validation

In order to showcase the functionality of the FLEXFUL extensions, we have demonstrated a mobile offloading scenario over the NITOS testbed. The scenario is based on an optimization problem, where we consider that each client has two interfaces for its communication with the internet, either through an LTE network, or through an operator provided WiFi mesh network. As each client has a predefined SLA with the mobile network operator, the operator will need to minimize the LTE bearer allocation, subject to maintaining the client SLAs for the downlink channel, when offloading them to the WiFi network.

The involved equipment makes use of the UPI$_R$ and UPI$_N$ that were developed for handling the LTE resources. More specifically, a set of NITOS nodes equipped with both WiFi and LTE interfaces have been used. NITOS nodes form a WiFi mesh network that uses a layer 2 routing algorithm. All these nodes will be connected to the LTE access network, which will be their default connection to the Internet. Our demonstration showcases the functionality of the mobile traffic to be redirected from the LTE access network to the WiFi mesh network, where some nodes act as gateways for the rest of the nodes. Figure 1 presents the experiment setup used to validate the extension.

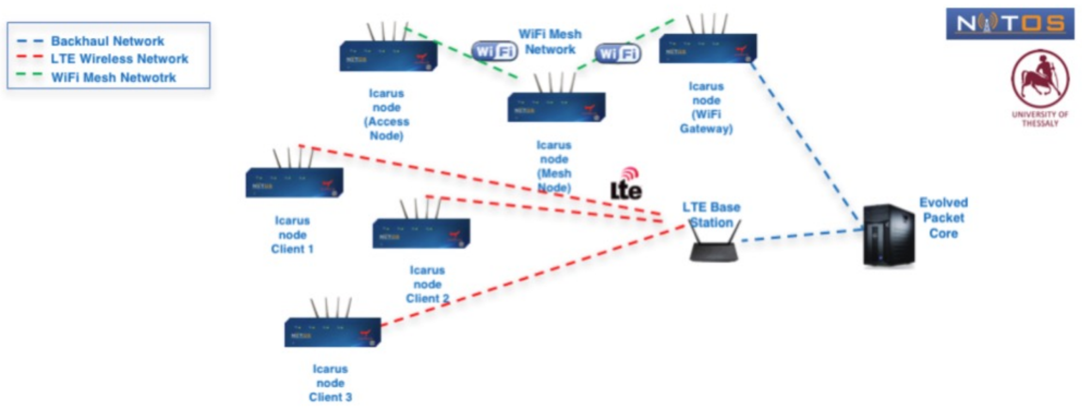

Figure 1. Offloading experiment setup

For the implementation and the validation of the framework, different components needed to be developed. These consist of the following:

- An OpenFlow controller that is handling the bridging between a WiFi gateway node, and the EPC network. The controller is endowed with the process of redirecting the clients to use the LTE/WiFi network, when their SLAs cannot be met, based on the current demand of the network.
- A daemon process, running on the client nodes, able to receive messages from the OpenFlow controller instance and select the interface to be used for sending/receiving traffic.
- Low level scripts used for setting up the 802.11s WiFi mesh network at the NITOS nodes.

Moreover, new have been developed for running such custom commands on the testbed nodes through the UPI framework and the orchestration through a UPI$_G$ script. The following figure is presenting an offloading experiment run when using the SLAs that are defined in Table 4. Based on the SLAs, and for a given experiment run, we would expect that client1, who is the most data hungry would remain to the LTE network for as long as its SLA can be met. When a client connects who will make the aggregated requested network capacity to exceed the total achieved limits (Client 2), the most data hungry node will be offloaded to WiFi. Figure 2 is demonstrating the results from such an experiment run, which validates our approach.

Table 2. SLAs for the offloading experiment

Client ID	Requested SLA
Client 1	30 Mbps
Client 2	10 Mbps
Client 3	25 Mbps

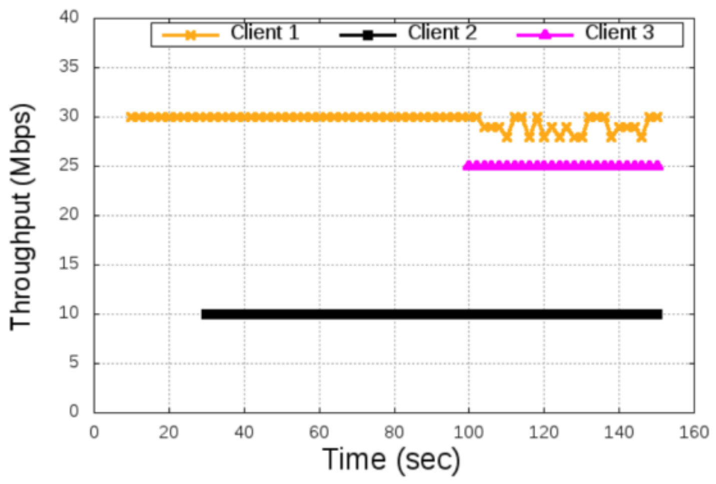

Figure 2. Throughput measurement for 3 clients running the offloading experiment.

3.1.5 Impact on existing WiSHFUL testbed and Fed4FIRE integration

The University of Thessaly (UTH) group is a very experienced group in developing tools for FIRE testbeds. Therefore, the FLEXFUL extension was very straightforward for the UTH engineers, as long as they got acquainted with the existing framework structure. The tools that where extended were easy to adopt, and support large scale experimentation. Moreover, as long as they are setup correctly, they are very reliable, and also interoperable with many other applications that are already supported by the framework.

Through the adoption of the WiSHFUL UPI framework, NITOS testbed has become another FIRE island that is offered via the WiSHFUL project. UTH adopted all the extensions that were supported for the compatible equipment, and thus the WiFi testbed is now offered as well. Up to now it seems a very good solution for combining LTE and WiFi resources in a single experiment. Experimenters are now able to login on NITOS, and load the WiSHFUL specific images on the nodes that will enable such functionality and interoperability with the rest of the consortium.

Given the fact that NITOS is one of the few testbeds all over the world, which provides the possibility on experimenting in pioneering LTE equipment, the incorporation of its resources into the WiSHFUL environment offers a significant added value to the project. It fosters the creation of a realistic experimentation environment where manufacturers, content providers, developers and researchers can take advantage of the WiSHFUL facilities to test their applications, services and radio configurations in novel scenarios that involve LTE technology. This can lead to a better understanding of the behaviour of applications, and software in general, running on commercial mobile devices under a huge range of LTE radio access configurations. FLEXFUL expands the features offered to the experimenters, extending the range of available LTE equipment and allowing WiSHFUL experimenters to research in heterogeneous environments using multiple wireless technologies.

NITOS' addition to the WiSHFUL facility contributes to the sustainability of the heterogeneous approach, which is an emerging research field in 5G discussions. Based on the existing research conducted in UTH, FLEXFUL can also contribute to the definition of requirements and roadmaps towards extending WiSHFUL testbeds to support 5G projects both in the context of FIRE+ and in H2020 5G calls. The experience that the UTH team has on developing services for the testbed can lead to the creation of spinoffs for developing new technical solutions. The testbeds that are available in WiSHFUL and FIRE in general can widely assist in drastically reducing the cost of developing new testbeds for benchmarking applications and testbeds.

3.2 Support given by the WiSHFUL Consortium – Patron: IMEC

In this section the support given from the consortium and the patron is presented in detail.

Preliminaries

Preliminary actions for supporting FLEXFUL extension were focused on the definition of key capabilities and parameters offered by the LTE hardware and software that should be exposed by the extension through WiSHFUL to the end user.

The WiSHFUL framework

The patron described in detail the WiSHFUL framework v 1.0, then several live trials were done to help the experimenter familiarize with the overall workflow. After the Consortium decided to switch to version 2.0 of the WiSHFUL framework, the patron helped the FLEXFUL team to get acquainted with the new framework, pinpointing changes and the necessary adaptions that needed to be conducted, discussing also the impact on the on-going activities.

Interfaces with pre-existing WiSHFUL components

There was no need to interface with pre-existing WiSHFUL components as there was no previous LTE support in WISHFUL. FLEXFUL mapped the decided parameters and control knobs to existing UPIs or created new ones when such a mapping was not possible.

Extending the UPI

The design of the novel UPI functions to control the entirety of the LTE network has been coordinated by the patron in order to ensure flawless integration within the existing UPIs and avoid duplication or overlapping in UPI definition and functionality.

Logistics

The patron helped in monitoring progress in the design and development process, providing suggestions and details in case of doubts on capabilities as well as preparing descriptions of the intermediate status. Intermediate reports have been presented by the patron during the plenary meeting held in Palermo, and in Ghent, based on the work conducted by the FLEXFUL team.

Validation

The project proponent has been helped by the patron about the definition of the best procedures and experiment tests to validate the new functionalities offered by the testbed extension. In particular, a showcase exhibiting the overall functionality of the extension through controlling an LTE network and offloading traffic to WiFi has been designed and deployed.

In the following table the overall detailed support given to the proposer is presented.

Table 3 - Time sheet of the support dedicated to the FLEXFUL project

Date	Dedicated Time [hour]	Topic
14/04/2016	3	Call to discuss the initial steps for bootstrapping the proposal, managerial support for contract signing etc.
18/05/2016	3	Call to present and explain WiSHFUL 1.0 framework
20-30/05/2016	6	Multiple emails communication to explain further the WiSHFUL framework 1.0
2/06/2016	4	Help for preparation of presentation for the plenary meeting in Palermo
13/06/2016	3	Call to present and explain WiSHFUL 2.0 framework
14-30/06/2016	8	Email support for WiSHFUL framework 2.0
12-19/07/2016	11	Troubleshooting connector module incompatibilities, refactoring UPIs exposed from FLEXFUL
15/09/2016	4	Presentation preparation for plenary meeting in Gent
19/10/2016	3	Conference call for review meeting demo design
26/10/2016	3	Conference call for final report draft skeleton and discussion
1-7/11/2016	6	Support for review meeting presentation preparation
18-20/11/2016	6	Support for refactoring the code into modules and uploading it into the WISHFUL github repository.
25/11/2016	4	Discussion on the tutorial of FLEXFUL and preparation of its draft.
TOT	**64**	

3.3 Documentation and examples for experimenters

The entirety of the code extension is open for use to experimenters and is being merged in the master branch github of WiSHFUL at https://github.com/wishful-project/. Documentation and tutorials are also under final revision and will be made available at github as well.

4 DVB-T software radio transmitter eXtension for IRIS

This WiSHFUL extension has been provided by the University of Perugia and has been supported by TCD.

4.1 Extension short description

The DVB-TX-IRIS Extension was developed by the Department of Engineering of the University of Perugia (UPG). It is a WiSHFUL Open Call 1 (OC1) project extension to the WiSHFUL framework. It leverages the functionalities offered by the WiSHFUL Iris software defined radio (SDR) platform on TCD's IRIS testbed.

The DVB-TX-IRIS Extension is designed to stream digital base-band (BB) samples to an SDR device. The DVB-TX-IRIS Extension uses Ettus USRP N210 to emit the signal, which can be received and decoded by off-the-shelf DVB-T receivers, such as commercially available TV sets, set-top boxes, and USB dongles. Since the existing modules of the Iris SDR framework do not allow generation of a DVB-T compliant TV signal, the project initially focused on the addition of this functionality to Iris SDR Framework. IRIS's modular architecture supports high re-programmability permitting the easy additional and configuration of a TV broadcasting transmitter. SDR offer uncomplicated upgrade to accommodate future broadcasting standards, without requiring significant hardware changes in the transmission network.

4.1.1 UPI extension

The WiSHFUL Project offers flexibility using UPIs to support and control real-time DVB-T broadcasting on TCD's IRIS testbed. The UPI extension developed for DVB-TX-IRIS used the Unified Programming Interface – Radio (UPI$_R$). In the remainder of this section we describe the UPIs developed:

upis.radio.set_frequency(freq): this function is used in order to modify the carrier frequency of the emitted DVB-T signal. The *freq* parameter value should be a string representing the carrier frequency expressed in Hz: the valid range is between 400 MHz and 4400 MHz, but the practical range is in the UHF TV band, i.e., between 474 MHz and 794 MHz. This parameter directly modifies the local oscillator frequency of the USRP SDR device daughterboard.

upis.radio.set_gain(*gain*): this function is used in order to modify the gain of the power amplifier located along the DVB-T signal generation chain. The *gain* parameter value should be a string representing the power gain expressed in dB: the valid range is represented by numbers in the range between 0 dB and 31.5 dB, with a step of 0.5 dB, but practical values should be limited to under 20 dB in order to not saturate the final power amplifier and distort the emitted analog signal. This parameter directly modifies the programmable attenuator gain of the USRP SDR device daughterboard.

upis.radio.set_outpower(*power*): this function is used in order to modify the power of the generated digital BB DVB-T signal. The *power* parameter value should be a string representing the expected power expressed in percentage: the valid range is represented by numbers greater than 0% but practical values should be limited to under 50%-60%. This parameter determines the level of amplitude clipping that the DVB-T OFDM signal experiences before floating-point to fixed-point conversion in the DVB-T transmission chain: theoretically, with a level of 100%, there is 99.7% probability that the signal is not clipped. Please note that the final emission power of the DVB-T analog signal is determined by the combination of the effects of the amplifier gain parameter and of the digital BB power parameter.

4.1.2 Hardware and Software components

The DVB-TX-IRIS Extension enables Digital Video Broadcasting−Terrestrial (DVB-T) compliant digital TV broadcasting to be performed with off-the-shelf hardware, such as a regular PC with a multi-core CPU, an Ettus SDR device, and open source software, such as the Ubuntu operating system, the Iris

SDR platform, and the WiSHFUL framework. The performance results show that DVB-TX-IRIS can run in real-time on a laptop computer (Intel Core i7 @ 2.4 GHz CPU, 4 core/8 threads, 8GB RAM).

The DVB-TX-IRIS software components were implemented in modules. These modules are constituted by a group of Iris components that are mapped to the constituent sub-blocks of the DVB-T standard transmission chain. Each sub-block is instantiated inside a separate Iris engine, so as to achieve maximum parallelism and processing speed. During the software development, the C++ Iris SDR Framework components have been validated by comparison with a MATLAB/Octave implementation of the DVB-T standard. The developed Iris DVB-T module can be controlled either by means of XML configuration files, or by means of the WiSHFUL framework, thanks to the WiSHFUL UPIs.

4.1.3 Deployment and setup

The University of Perugia with support from the TCD team, during the development of the DVB-TX-IRIS extension, has used the IRIS SDR and computing facilities of the CONNECT's IRIS wireless testbed at TCD. The testbed is composed by 16 Ettus USRP N210 devices that are physically arranged in a regular grid on the ceiling of the TCD laboratories. Each USRP device can be controlled by means of a dedicated Ubuntu VM. The frequency used in USRP was 400 MHz – 4.4 GHz.

The benchmark was run on an Intel Core i7 2.4 GHz CPU with 8 GB RAM and Ubuntu 16.04 OS, and the programs have been compiled with the GNU C++ compiler v. 5.4.0.

The development of the DVB-TX-IRIS extension has been done specifically for the Iris SDR platform and the IRIS testbed at TCD. They used several SDR devices, namely the Ettus USRP N210 with the SBX120 daughterboard. The extension ran inside a Linux virtualized environment remotely at TCD.

4.1.4 Validation

The DVB-TX-IRIS code has been validated using automated testing provided within the Iris SDR framework. In particular, using routines such as *cmake* and ctest, in order to automatically perform a number of tests on code sanity and expected outputs.

Additional tests included, "smoke" tests (to verify that the code compiles and links correctly into an executable), crash-proof tests (to verify that the produced executable is able to run without crashing), black-box tests (to check whether the number and data types of inputs and outputs match the expected ones, without knowledge of the particular implementation of the task in the sub-block), and white-box tests. White-box tests are based on the particular processing task that is performed by the sub-block, and can be used to perform data buffers size checks.

Moreover, for every sub-block implemented, a parallel version of the processing task using a high-level interpreted language such as MATLAB or Octave was used. Given a predefined input data set, the MATLAB sub-block and the compiled C++ sub-block were fed with the same input, and the relevant outputs were obtained.

The GNU Radio *uhd_fft* application was also utilised. The aim was to verify the emitted spectrum signal quality.

Furthermore, Rohde & Schwarz real-time spectrum analyser present in the TCD testbed laboratory was used to verify the spectrum signal transmitted.

Finally, the TCD team validated the DVB signal transmitted with a TV and set-top-box digital receiver.

4.1.5 Impact on existing WiSHFUL testbed and Fed4FIRE integration

Not applicable.

4.2 Support given by the WiSHFUL Consortium – Patron: TCD

Preliminaries

Support was given in form of training using videos, basic documentation, and direct Q/A to introduce the Open Call partners to the IRIS SDR framework. In particular, the online tutorials and documentation provided the basic instructions for the setup of some basic IRIS and testbed demos, such OFDM transmission and reception. More specific questions regarding parallelization and optimization in IRIS were answered in private via email.

The WiSHFUL framework

The patron worked together with the University of Perugia team to help develop the WiSHFUL Controller and Agent using the WiSHFUL Framework version 2.0. The patron shared this framework with the Perugia team. In additional, TCD also provided support needed to use the WiSHFUL Framework UPIs.

Interfaces with pre-existing WiSHFUL components

The Iris framework required extra consideration during the code design phase to support the WiSHFUL UPIs. Iris is very modular and the new DVB-T Extension was flawlessly added to the existing Iris core and modules.

The patron needed to develop a new extension to support the WiSHFUL Framework with Iris by developing a new interface using a pre-existing model. TCD used the WIFI components available in (UPI_R) of the WiSHFUL Framework version 2.0 as a template for this development.

Extending the UPI

The design of the novel WiSHFUL UPI functions to control the reconfigurable Iris SDR Framework has been coordinated by the patron in order to ensure compatibility and avoid overlaps with what was already available. This UPI extension was presented in section 4.1.1.

Validation

The University of Perugia has been helped by the patron with the definition and list of best strategies and metrics to validate the new functionalities offered by the IRIS testbed extension. This support included bi-weekly meetings, SDR and IRIS testbed training, on-site support, and so forth.

Table 4 DVB-TX-IRIS Support

Date	Dedicated Time [hour]	Topic
5th May 2016 to 20th October 2016	16	General progress meetings
October 2016	16	Testbed Setup, training, set-top-box configuration and installation, support to reserve, use and access Iris testbed
October 2016	20	Training to use the WiSHFUL Controller, Agents and Iris SDR Framework using a video and the spectrum analyzer demonstration
October 2016	24	Support in the development WiSHFUL Controller including scripts to hold the DVB-T parameters
October 2016	16	Support of Open call review in Brussels

TOTAL	**92 hours**

4.3 Documentation and examples for experimenters

The University of Perugia has made all code and examples available on github [3]. They also developed a tutorial to teach experimenters how to use the DVB-TX-IRIS extension, which is available on github [4].

5 Conclusion

WiSHFUL capabilities have been extended during year two by means of new hardware and software modules after the first open call (OC1). These allow experimenters to use novel features, functions and tools that were not included in the WiSHFUL testbed. Such enhancements have been made possible thanks to patrons, chosen among the WiSHFUL Consortium, which tutored OC1 third parties, providing their expertise, suggestions and operational help. The functionalities offered to experimenters, and the integration with pre-existing testbed facilities have been specifically tested. Finally, patrons supported third parties for preparing the open call review, where the effectiveness of the new modules has been proven.

The extension that introduced reconfigurable antenna systems has been validated measuring different power levels and performance by opportunely tuning the antenna configuration. The LTE-enabled extension has been validated through an example of offloading traffic from LTE to Wi-Fi. The extension that offers the DVB-T transmission chain to experimenters has been validated through a TV and a set-top-box digital receiver.

Despite of the heterogeneity of functionalities and hardware platforms integrated within the extensions, which also correspond to heterogeneous background of the involved research groups, our experience demonstrated that in all the cases it was easy to familiarize with the WiSHFUL control framework, even when moving from one implementation to a new one, and to integrate the new modules. Some efforts have been dedicated to the general design phase, in order to apply the WiSHFUL vision, based on simplification and unification of programming models for different network elements and technologies.

6 References

[1] Module for RAS extension - https://github.com/wishful-project/module_ras_antenna

[2] RAS Demo scripts for experimenters

https://github.com/wishful-project/examples/tree/master/ras_antenna_wilab

[3] DVB-T code and examples https://github.com/wishful-project/examples/tree/master/iris/dvb-tx-iris.

[4] Tutorial about the DVB-TX-IRIS extension

https://github.com/wishful-project/module_iris/tree/master/dvb-tx-iris/doc.