ETSI WORKSHOP ON FUTURE RADIO TECHNOLOGIES – AIR INTERFACES
27-28 January 2016, Sophia Antipolis France

Unified Radio and Network Control Across Heterogeneous Hardware Platforms

Nicholas Kaminski (Trinity College Dublin), Ingrid Moerman (iMinds-Ghent University), Spilios Giannoulis (iMinds-Ghent University),Peter Ruckebusch (iMinds-Ghent University), (Pierluigi Gallo (CNIT), Anatolij Zubow (Technische Universtät Berlin), Robin Leblon (nCentric Europe), Ivan Seskar (Rutgers University), Sunghyun Choi (Seoul National University), Jose de Rezende (Universidade Federal do Rio Janerio)

Abstract— **Experimentation is an important step in the investigation of techniques for handling spectrum scarcity or the development of new waveforms in future wireless networks. However, it is impractical and not cost effective to construct custom platforms for each future network scenario to be investigated. This problem is addressed by defining Unified Programming Interfaces that allow common access to several platforms for experimentation-based prototyping, research, and development purposes. The design of these interfaces is driven by a diverse set of scenarios that capture the functionality relevant to future network implementations while trying to keep them as generic as possible. Herein, the definition of this set of scenarios is presented as well as the architecture for supporting experimentation-based wireless research over multiple hardware platforms. The proposed architecture for experimentation incorporates both local and global unified interfaces to control any aspect of a wireless system while being completely agnostic to the actual technology incorporated. Control is feasible from the low-level features of individual radios to the entire network stack, including hierarchical control combinations. A testbed to enable the use of the above architecture is utilized that uses a backbone network in order to be able to extract measurements and observe the overall behaviour of the system under test without imposing further communication overhead to the actual experiment. Based on the aforementioned architecture, a system is proposed that is able to support the advancement of intelligent techniques for future networks through experimentation while decoupling promising algorithms and techniques from the capabilities of a specific hardware platform.**

Keywords—experimentation; wireless communication; testbed; control; architecture; scenarios

I. INTRODUCTION

Considering the emerging wireless ecosystem, one is faced with a heterogeneous mix of technologies, operators, and service providers attempting to coexist in a single environment, featuring a high-density deployment of wireless devices. High heterogeneity in device capabilities (in terms of spectral bands, coverage, management functionalities, networking models, etc.) combined with limited open, vendor-independent configuration interfaces complicate achieving the often conflicting goals of independent providers and integration of technologies to provide coherent service. In the emerging wireless ecosystem, wireless devices employ multiple radio interfaces, spanning over several standards (such as LTE, WiFi, Bluetooth) or offering more esoteric capabilities in the form of programmable interfaces, based on software defined radio (SDR) techniques. Developing techniques suitable for this emerging ecosystem require services capable of investigating a broad range of technologies in a unified manner.

To unravel this conundrum, this work identifies driving scenarios to capture the challenges associated with the increasing density and heterogeneity of wireless devices in a concrete and tangible manner. These scenarios are directly presenting a relevant and significant set of requirements for developing the functionality required to experimentally investigate the challenges of future wireless systems. As an initial effort, these scenarios focus on enabling techniques to manage the effects of interference that accompanies future high-density scenarios. Each showcases focuses on a different source for such inter-device interference and displays an approach which requires novel experimentation functionality.

Following the definition of this set of motivating scenarios, an architecture for support future wireless experimentation is presented. This architecture is constructed to both address the requirements of the tangible scenarios, capturing key challenges of future systems, and allow for extension to support investigation of as yet unforeseen challenges. The development of this experimentation framework is the subject of the Horizon 2020 Wireless Software and Hardware platforms for Flexible and Unified radio and network control (WiSHFUL) project [1]. Specifically this project develops Unified Programming Interfaces (UPIs) to provide experimenters a common means of control over a variety of radio hardware platforms running various MAC/Network layer protocols. The UPIs allow experimenters to investigate heterogeneous, high-density scenarios with ease, spanning across from the individual radio device all the way to the entire network scale. As such, herein both the physical, MAC and network layer control aspects of the experimentation architecture are examined.

II. MOTIVATING SCENARIOS

Here a set of scenarios is provided, that present an initial set of challenges associated with interference resulting from future high density, high heterogeneity wireless deployments. This collection of scenarios captures fundamental challenges of future networks in a concrete and tangible manner in order to outline the requirements on supporting UPIs.

A. *Efficient airtime management*

In dense wireless networks, co-channel interference is a fundamental problem, thus efficient airtime management through parameter adaptation and interference avoidance techniques (e.g. time sharing) is increasingly important. This scenario captures the challenges of co-channel interface by examining a potential mechanism for accomplishing such airtime management across IEEE 802.11 WiFi Access Points (AP). In doing so, this scenario addresses the problems of avoiding co-channel interference through adaptation of wireless parameters and efficient allocation of transmission time for co-located APs. Ultimately, this scenario examines questions related to the dynamic control of multiple APs in a coordinated manner.

Specifically, this scenario deals with efficient air-time management by examining the hidden node problem. In IEEE-802.11 (WiFi) networks co-channel interference significantly degrades performance due to packet losses (hidden node problem) and channel contention. Consider the example network given in Figure 1. This scenario assumes four active flows in the following QoS classes – the first three are best effort (BE) while the last one is voice. Each flow is assigned to one of the two APs. Further, let us assume that the two APs, AP1 and AP2, are operating on the same radio channel. In such a case a cell-edge user like STA2 may suffer from interference due to hidden node, i.e. the downlink traffic from AP1 to STA2 will collide with traffic originated at AP2. By solving the hidden node problem, the performance of all nodes in neighboring wireless networks can be increased.

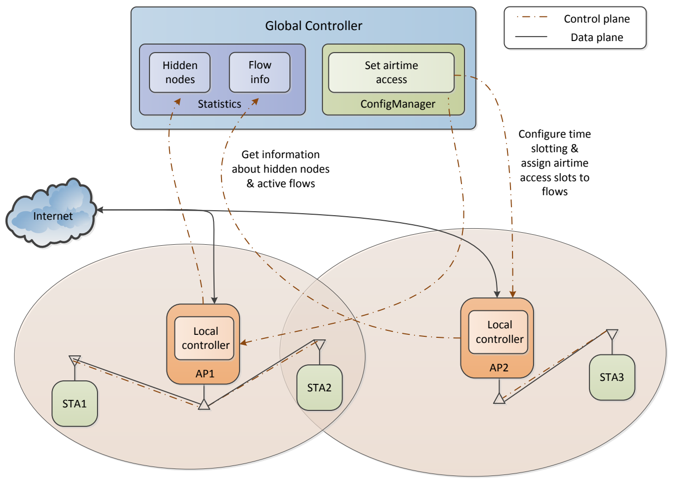

Figure 1 Traffic-aware 802.11 airtime management scenario.

Given the necessary supporting functionality, the challenges of this scenario may be addressed by monitoring the performance of each AP. Such monitoring would make degradation associated with inefficient management clear, allowing the rescheduling of flows to avoid inference. To accomplish this goal, global monitoring of network performance would be required. Specifically, some control entity would need the ability to monitor the active flows for detecting hidden nodes and to define appropriate channel access patterns and time slots for solving the hidden node problem.

B. *Co-existence of heterogeneous technologies*

In dense wireless networks, the co-existence of heterogeneous technologies using the same wireless resources is challenging. In particular, there is an increasing need for heterogeneous technologies to be able to co-exist within the same or overlapping frequency bands. Accomplishing this naturally requires the dynamic adaptation and harmonization of spectrum allocation and access across separate wireless technologies. This will enhance the performance in both networks and make the QoS characteristics (throughput, latency, reliability) more predictable. This scenario specifically considers the example networks IEEE-802.11 (WIFI in 2.4 GHz band) and IEEE-802.15.4e (TSCH) illustrated in Figure 2. The simultaneous operation of both networks in close proximity will inevitably lead to performance degradation due to interference. This is because of contention-free explicit scheduling of radio resources in TSCH (timeslotted channel hopping) and the unreliability of carrier-sensing (listen-before-talk) mechanism used in WiFi, rendering WIFI unable to sense any wireless transmission of the other technology. The QoS in both networks can be increased by making them aware of each other.

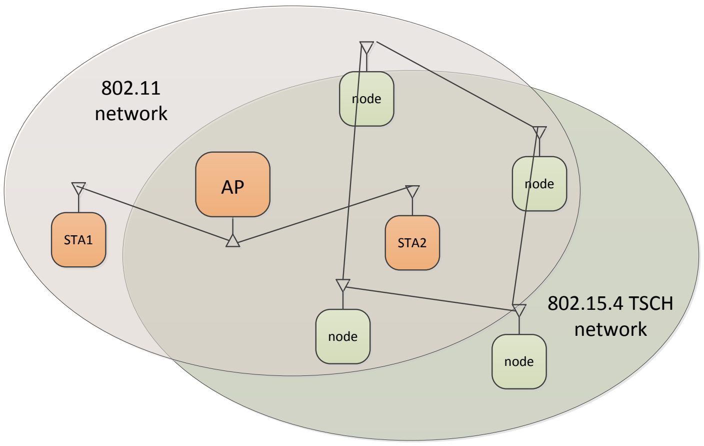

Figure 2 Example illustrating two co-located wireless networks of different technology.

One can imagine multiple co-existence schemes for WiFi and TSCH. Some basic schemes can be implemented by only modifying the sensor network. More advanced, and also promising, schemes require cooperation between the networks. This scenario examines a traffic-aware interference avoidance scheme where, depending on the network load in both networks, other decisions are made. For such a scheme two possible cases, illustrated by Figure 3, must be considered. In the first case the sensor network is highly loaded. Here it is more efficient to perform any interference avoidance in the WiFi network, reducing the overhead on the more loaded network. To accomplish this, the sensor network would need to provide the scheduling information to allow the WiFi network to delay transmissions to points in time where no collision would occur. In the second case the network load in the WiFi is high, suggesting that excluding the spectrum used by WiFI from the hopping scheme of the sensor network is a more promising approach to co-existence.

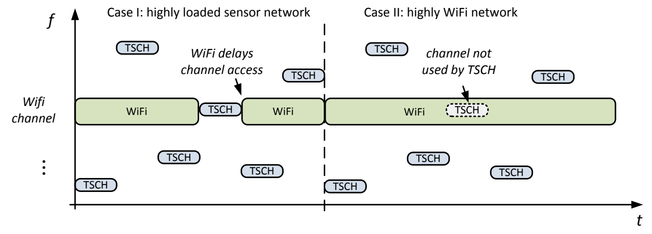
f
Case I: highly loaded sensor network
Case II: highly WiFi network
TSCH
WiFi delays channel access
channel not used by TSCH
Wifi channel
WiFi
t

Figure 3 The proposed co-existence scheme for avoiding interference between WiFi and TSCH.

Another approach is to use a cross-technology TDMA protocol to coordinate the transmission between both types of nodes and reduce interference to a minimum. The system runs a TDMA radio program on the WiFi nodes, adapts time slots to traffic requirements, keeps free some slots that are implicitly reserved to TSCH, and uses the remainder for transmission, in order to minimize cross interferences.
To support the experimental investigation of this scenario, a great deal of functionality is required. A mechanism for the discovery of co-located wireless networks within interference range is certainly necessary to identify whether a problem exists. Furthermore, a range of mechanisms to support mutual network awareness is required, including the ability to share information regarding network load between networks and to make the MAC schedule of the TSCH network available as well as the share the wireless channel used in the IEEE 802.11 network. Moreover, mitigation functionality must be available, potentially including the configuration of spectrum access in the WiFi network, configuration of channel exclusions within the TSCH network, time synchronization between both networks, and the tuning of MAC parameters according to frames size and slot allocation.

C. *Load and interference aware MAC adaptation*

This scenario considers a number of wireless nodes coexisting in the same environment with traffic flows that are added dynamically. At first, the nodes run a contention-based access protocol (either a backoff-based protocol with a constant, but tunable, contention window or a persistent access protocol with tunable channel access probability) and adapt the configuration of the contention-based access protocol to support more traffic flows. When the traffic demand gets higher, tuning the parameters of the contention-based scheme no longer succeeds in supporting more traffic flows. At this point, the nodes need to switch to a TDMA protocol to support more traffic flows. As such, this scenario examines the inherent tradeoffs between various approaches to MAC protocols.

This scenario examines the utilization of control programs, implementing a cognitive adaptation strategy, across different hardware platforms in order to both fine-tune the contention-based protocol on the local level and to switch to a TDMA based protocol when the traffic demands cannot be supported anymore using the contention-based protocol. Specifically, addressing the challenges of such a dynamic situation suggests the use of a hierarchical control scheme that focuses on adapting the radio functionality. Naturally in order to maintain coherence in the network, this scheme must operate over any participating platform, as illustrated in Figure 4.

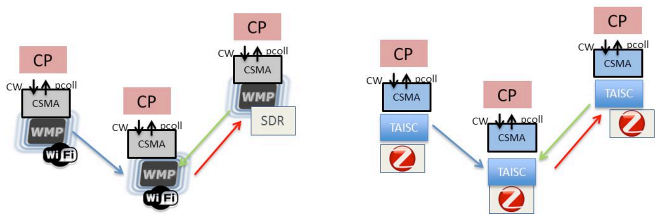

Figure 4 Deployment of a single local program across several platforms.

As a concrete example, initially consider 10 active wireless nodes using a CSMA MAC with a backoff time of 500 slots. As the number of nodes is increased in steps of 10, eventually a clear drop in the total number of frames successful transmitted will occur. To stabilize frame drop rate, it is certainly possible to tune the system by increasing the size of the backoff time to 2000 slots. Should the number of nodes again increase, however, another drop in performance will occur. At this point, the network is so dense that the CSMA protocol is unable to operate and a switch to some other scheme, such as TDMA, is required. Examining such a scenario and the various solutions for parameter tuning or MAC protocol updating certainly requires the exposure of MAC parameters and even MAC protocol definition to some controlling entity.

D. *In-situ Testing*

Wireless testbeds are imperative for testing innovative technologies such as protocols, hardware, and several other modules of any wireless solution. Many of these technologies will serve in dynamic wireless environments and under challenging conditions. For sake of maintainability and experiment repeatability, however, testbed infrastructure often is fixed. Relocating nodes is difficult since their power supply and/or network connections are mounted on wall sockets. The testbed environment is thus less dynamic and the conditions are more stable making the evaluation of experimental wireless solutions in testbeds less realistic.
A portable testbed that can be easily deployable on remote, real-world locations is clearly necessary. Such a testbed would need to be straightforward to deploy where needed, include rugged equipment and self-contained power. Furthermore, a wireless mesh backbone to ensure connectivity between the nodes would be required to allow operation in a variety of

environments. This backbone would need to employ the sort of interference management suggested by previously discussed scenarios. Finally, the portable test must operate in a transparent manner to allow users to examine the phenomena of interest.

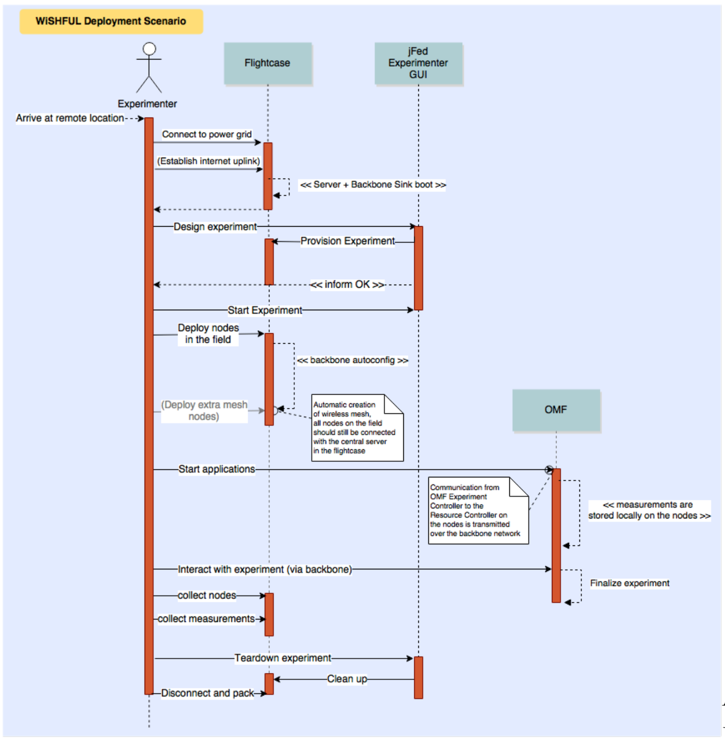
WiSHFUL Deployment Scenario
Experimenter
Flightcase
jFed Experimenter GUI
OMF
Arrive at remote location
Connect to power grid
(Establish internet uplink)
<< Server + Backbone Sink boot >>
Design experiment
Provision Experiment
<< inform OK >>
Start Experiment
Deploy nodes in the field
<< backbone autoconfig >>
(Deploy extra mesh nodes)
Automatic creation of wireless mesh, all nodes on the field should still be connected with the central server in the flightcase
Start applications
Communication from OMF Experiment Controller to the Resource Controller on the nodes is transmitted over the backbone network
<< measurements are stored locally on the nodes >>
Interact with experiment (via backbone)
Finalize experiment
collect nodes
collect measurements
Teardown experiment
Clean up
Disconnect and pack

Figure 5 Sequence diagram for the deployment of the portable testbed.

Taking the successful Fed4FIRE approach [2] as a model for the use of testbed, the following steps, illustrated in Figure 5, would be required on the portable testbed during experiment life-cycle:

1. When the experimenter arrives at the location, the flightcase is plugged into the power net and the servers and switches inside start to boot. Optionally, the experimenter can connect the switch uplink to the internet.
2. As the servers boot, the backbone also configures itself automatically. It creates a wireless mesh among the nodes.
3. When everything is up and running, the experimenter launches the jFed tool from a laptop that is either inside the flightcase or connected to the central switch. The experiment is designed or loaded from a previous run.
4. jFed will perform the needed actions via the testbed management server and the nodes will be provisioned with the desired software.
5. After this process, the user is informed and the actual experiment is started.
6. The user will deploy all nodes in the field; they remain connected and accessible via the wireless backbone.
7. If there should be a bad wireless link between one or several nodes, an extra backbone node can be added to optimize the mesh network.
8. Via OMF, the experimenter starts his experiment. OMF will make the calls to the nodes over the backbone network. These calls can include (but are not limited to) the setup of a wireless interface, the changing of channels or the starting of an application.
9. While the experiment is running, the measurements are stored locally on the nodes.
10. As the experiment finishes, the experimenter can collect all nodes and properly dock them in the flight case, physically connecting them again with the core network.
11. The measurements are fetched from the individual nodes and the experiment can be torn down. If the throughput of the wireless mesh network is high enough, or the amount of measurement data is low, the measurement can be transported over the wireless backbone in real time to a database server in the flight case.
12. jFed will ask the central testbed server to clean the nodes up, and the flightcase can be closed and plugged out.

E. Intelligent Download with WIFI Tethering

Recently, with rapid growth of number of smartphones and mobile devices equipped with various wireless technology interfaces, tethering popularity gains more and more popularity. It is a very convenient, ad-hoc and low-cost wireless Internet access technology. In most cases Wi-Fi tethering is used, which allows sharing the Internet connection provided by 3G/4G technology with other devices (eg. laptops) using WiFi network.
However, in most cases, there is a limit on amount of data that cellular subscriber can download every month. After exhaustion of available data transfer, user connection can slow down significantly or in worse cases, one can be charged for any extra data he/she downloads. As long as the user is aware of all his/her network transmissions, there is no problem. Unfortunately, it can happen that operating system and applications will perform upgrades and download a huge amount of data that user may not even notice until he/she gets a bill from telecom company. These concerns suggest the need to recognize and defer any "unnecessary" traffic flows that require downloading a huge amount of data to a later point in time. Operating system updates (e.g. Windows/Linux) are perfect example here. As such, this scenario investigates the intelligent use of available connectivity by filtering download requests based on context.
To further explore the concept behind this scenario, consider filtering "unnecessary" traffic flows when being connected to a tethering AP, as depicted in Figure 6. The operation of this

scenario can make use of IEEE 802.11u that defines Generic Advertisement Service (GAS). GAS is a mechanism that delivers information to the STA from advertisement services. It allows stations to obtain information about network services. The standard defines a number of advertisement protocols that can be used with GAS: Access Network Query Protocol (ANQP), Media Independent Handover (MIH), Emergency Alert System (EAS) as well as proprietary vendor specific protocols (which will be most useful for us). The GAS mechanism allows the STA to know in advance the AP capabilities, even before associating with it, which is an important feature for this showcase.
With the help of GAS we are able to block the "unnecessary" flows already on the WiFi end-user terminal (e.g. laptop). After reception of the specific IE (information element) from the tethering AP, the terminal should translate it into local firewall filtering rules and apply them. One possible way to achieve that is use of netlink interface and netfilter framework provided by Linux (used by iptables). Note, here both the tethering AP as well as the WiFi end-user terminal need to be WISHFUL-compliant. In a second option, which does not require the end-user terminal to be WiSHFUL-compliant the blocking of "unnecessary" flows is performed in the tethering AP which is fully transparent to the end-user terminal.

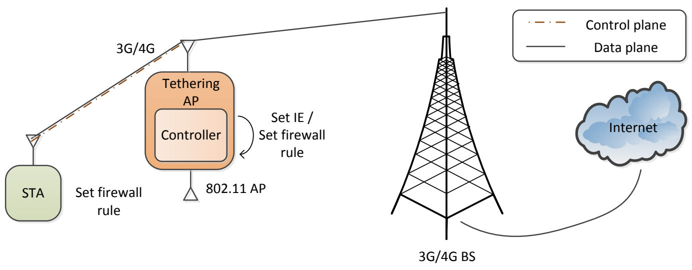

Figure 6 Controlling WiFi tethering operation.

For the first option, a means of programing the Information Elements (IE) send in the beacon frames of the tethering AP is required. Moreover, the end-user terminal requires functionality to read the received IEs, as well as to program the firewall, i.e. reject all outgoing traffic to a specific remote host.
For the second option, the tethering AP must be able to detect and block "unnecessary" flows.

F. *WiFi Offloading*

Although the capacity of cellular networks constantly increases thanks to technological enhancements, the throughput they provide can turn out to be insufficient, because traffic demand increases even faster. On the other hand, most of mobile devices are not only equipped with LTE interface, but also with a WiFi chip. It is therefore promising to offload traffic from mobile networks to WiFi and use them as an extension to the cellular network. This is one of the features in 5G that gains a lot of interest by telecom operators. The main reasons for this approach are the high data rates and availability of WiFi networks in urban environments.
Currently, mobile devices only implement a limited method for deciding when to offload traffic to WiFi, as show in Figure 7. When a mobile terminal discovers and connects to a WiFi network, it steer all its traffic over the WiFi network. The main drawback of this solution is the lack of QoS considerations that can lead to situation where a mobile device will switch from a high data rate cellular connection to a low data rate WiFi connection.
In current networks, operators cannot influence the decision on mobile stations to offload traffic, but the idea is so appealing, that some activity by several standardization forums was taken. They propose operator-controlled WiFi, which are deployed and managed by an operator and/or its partner. In 3GPP Release 12, some WLAN/3GPP inter-working aspects were standardized. Their aim is to provide the network operator control mechanisms to steer traffic offloading in both the downlink and uplink. There is a clear need of functionality to support the investigation of adjusting parameters at the mobile station such as receive power level threshold. When the received power is higher than this threshold mobile can offload its traffic via a network operated controlled WiFi AP.

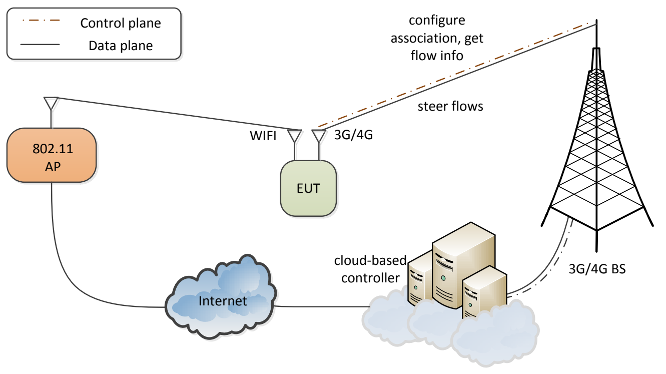

Figure 7 Controlling WIFI offloading.

To support WiFi offloading, several functions are needed. First, the network operator will need to define parameters (e.g. receive power thresholds) that will be send to the mobile STAs. These parameters allow STAs to connect to the best network (WiFi or LTE). Second, the network provider must be able to decide which flows should be offloaded from cellular to WiFi networks. For example, all VoIP traffic might stay in the cellular network, because it provides a more robust and reliable connection, and all flexible flows using the TCP protocol (eg. file transferring, web browsing) are offloaded to WLAN.

II. WISHFUL ARCHITECTURE

Experimentation is certainly a vital tool in the development of future wireless solutions. Furthermore, as illustrated by the above discussion of scenarios for future wireless networks, a large variety of functionality must be supported to investigate

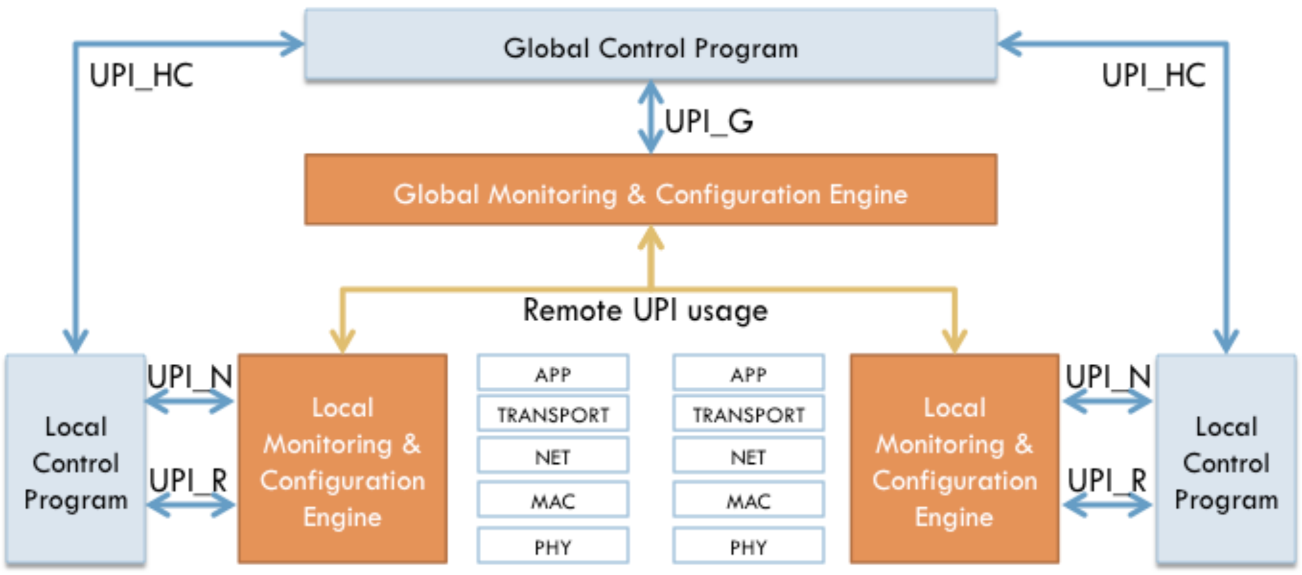

Figure 8 Conceptual diagram of WiSHFUL architecture

the challenges most relevant in the advancement of wireless communications. Moreover, the increasing diversity of wireless solutions and competing radio technologies, along with the ever more stringent requirements on the reliability of test results, has caused wireless test facilities to evolve to be exceedingly complicated imposing steep learning curves for new experimenters. Therefore, as the need for investigating a broad range of scenarios grows, so does the difficultly in doing so.

For these reasons, the WiSHFUL project directly targets lowering the experimentation threshold by developing flexible, scalable, open software architectures and programming to prototype novel wireless solutions. Specifically, WiSHFUL develops mechanisms for unified radio control to provide developers with deep control of physical and medium access components without requiring deep knowledge of the radio hardware platform and unified network control to allow the rapid creation, modification, and prototyping of protocols across the entire stack. These mechanisms chiefly take the form of Unified Programming Interfaces that operate across a range of hardware platforms. In this way WiSHFUL empowers experimentation facilities with the capability to experiment with emerging wireless technologies.

A. Major Entities

The WiSHFUL architecture, displayed in Figure 8, contains several entities designed to support the investigation of future networks. First and foremost within this architecture are the collection of UPIs, with each UPI providing specific functionality to experiments. The radio interface (UPI_R) consists of a set of functions that ensure uniform control of the radio hardware and lower MAC behavior across heterogeneous devices. The functions provided herein take a generic form in order to provide experimenters with consistent operation over hardware specific implementations. The network interface (UPI_N) parallels the UPI_R with a set of functions that provides uniform control over the upper MAC and higher layer protocol behavior across various devices. Again, the UPI_N consists of generic functions to provide a consistent and straightforward experimentation experience across heterogeneous platforms. The global interface (UPI_G) extends the reach of the control provided by both the UPI_R and the UPI_N across several devices in a coordinated and generic manner. The generic functions of UPI_R, UPI_N, and UPI_G are supported by Monitoring and Configuration Engines (MCEs) that contain and manage the platform specific implementations of UPIs within WiSHFUL empowered facilities. Naturally, the UPI_R and UPI_N are supported by a local MCE, while the UPI_G employs a global MCE. Finally, the hierarchical control interface (UPI_HC) enables hierarchical control between control programs structured in a standard manner. Note that this interface does not directly interact with hardware, but rather provides experimenters with the means to explore hierarchical control by offering a convenient method of inter-control communication.

The separation between radio and network functionality occurs within the MAC layer of the OSI stack. In particular, WiSHFUL considers the Upper MAC and higher layers within network control functionality, relegating the Lower MAC and lower layers to radio control functionality. The Upper MAC is responsible for inter-packet states that are not time critical, including framing and management functions, where some form of negotiation between nodes is required. The Lower MAC, on the other hand, directly interacts with the PHY transmission and reception operations, where minimization of processing latency is certainly critical. Typical Lower MAC functions include sending and receiving data, back-off, interframe spacing, and slot synchronization. As such, this distinction reflects the focus on inter-device coordination within the network control and more direct hardware operations within the radio control.

B. *User Control*

The interfaces of the WiSHFUL architecture are designed to support the user in controlling wireless hardware and the accompanying protocol stacks. WiSHFUL views user control as being embodied in control programs, which are either local or global in nature. In general Control Programs (CPs) are user defined software that implements the controlling logic for a wireless experiment and makes use of the UPI_R and/or UPI_N for hardware/protocol control. Local Control Programs (LCPs) are those that use the local information and abilities of a single device, while Global Control Programs (GCPs) interact with a group of devices.

The WiSHFUL architecture supports a two-tier control hierarchy. These two tiers work in a coordinated manner, being orchestrated at the global level. Indeed, global control programs can instantiate local control programs on wireless nodes, performing a sort of control by delegation, or can act directly on the wireless nodes in a coordinated manner. Control by delegation is needed when the reconfiguration decisions or the parameters to be monitored have strict time constraints, which cannot be guaranteed by the control network. In fact, the physical channel used for conveying control messages to/from the global controller can be unreliable and introduce some latencies. Since radio performance depends on highly variable network conditions (e.g. channel propagation, fading, interference, access timings, etc.), control by delegation is particularly important for radio control. The architecture also supports hybrid approaches, in which some control operations are managed at the global level, while some others are delegated to wireless nodes. The coordination between global and local control programs is achieved by employing the UPI_HC. Currently, the WiSHFUL framework follows a proactive approach. A CP has to trigger the execution of UPI functions on the wireless node under control. This polling-based approach might be not sufficient for each control program application. Therefore, for the future we plan to support also a reactive approach. Here the user can define a trigger where when a certain condition is fulfilled a registered callback function is executed.

C. *Hardware Interfacing*

Figure 9 illustrates how the WiSHFUL radio control works on three different platforms, namely the Iris SDR framework [3], TAISC [4] and WMP [5]. The global MCE runs remotely on a Linux machine and allows implementing node configuration that depends on network-level decisions and can be executed in a time-coordinated fashion among multiple nodes. Each of the WiSHFUL enabled nodes runs a local MCE that offers the same local services and the same UPI functions on different platforms by means of a specific Connector Module. This unified approach unloads the experiment from the burden of dealing with a multiplicity of configuration and utility tools, (e.g. iw, iwconfig, iptables, iwlist, iperf, b43fwdump, etc). These tools, indicated in Figure 9 as Local Control Services, are heterogeneous upon platforms/operating systems and depend on the hardware and software configuration of the device under test.

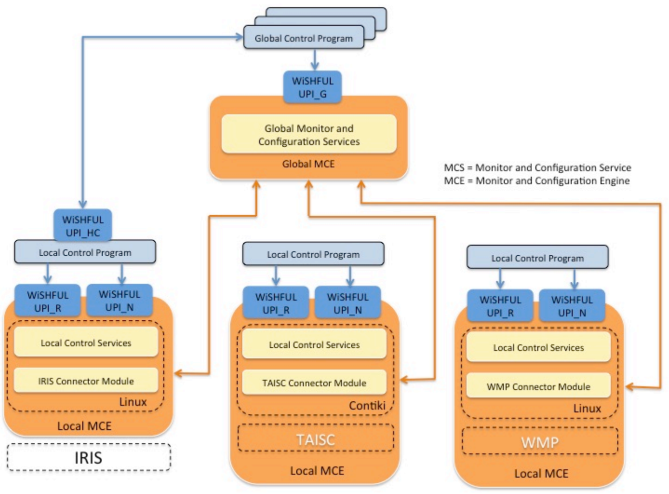

Figure 9 WiSHFUL architecture, UPIs and supported platforms

The Connector Module operate in conjunction with local MCEs to expose the uniform UPI functions on different hardware and software radio platforms. The module achieves two main goals: i) diverting platform-independent UPI calls to platform-dependent implementations and ii) providing a unified way to deal with a plethora of tools provided by heterogeneous operating systems (e.g. iw, iwconfig, iptable) or platforms (e.g. bytecode-manager for the WMP). Note that UPI functionality may or may not be supported by every platform, depending on the capabilities of the platform and the implementation of the Connector Module.

Figure 10 illustrates the interaction from MCE to the Connector Module and subsequently the radio platform. The local MCE delegates each UPI call to the appropriate Connector Module that executes the call using platform-specific sub modules. Currently, all local MCEs and connector modules are implemented in Python, except for Contiki sensor nodes that, in addition to the Python implementation, also have a native implementation using GITAR [6]. The native implementation is used when the sensor nodes are decoupled. In case they have a Linux host PC (e.g. in testbeds) the Python implementation can be used. This allows to easily prototype wireless solutions for sensor networks that can also work in real deployments, when the host PCs are not available.

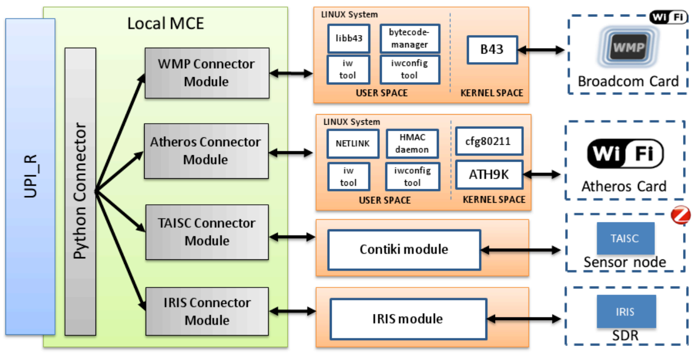

Figure 10 WiSHFUL adaptation modules

D. Basic Services

Alongside, the UPIs themselves, the WiSHFUL framework offers a number of basic services that are summarized here.

1) Node Discovery

A GCP often requires functionality for automatic node discovery. WiSHFUL provides the protocol developer an easy way to define the set of nodes he wants to control. Any wireless node belonging to the same experiment group can be controlled by a GCP using the WiSHFUL UPIs. From that set of nodes the user can either select all of them or just a sub-set.

2) Execution Semantics

The WiSHFUL MCE (local and global) supports two execution semantics. The first is a synchronous blocking UPI call where the caller, i.e. the CP, is blocked until the callee, i.e. any UPI function, returns. The second option is an asynchronous non-blocking UPI function call. Here any UPI call returns immediately. The caller has the option to register a callback function so that he can receive the return value of the UPI call at a later point in time.

3) Time-Scheduled Execution of UPI Functions

Besides the possibility of immediate execution of UPI functions either using a blocking or non-blocking scheme, the WiSHFUL MCEs also provide the possibility for time-scheduled execution of UPI functions at a particular point in time. This is important if nodes need to coordinate their actions in time, e.g. a set of nodes must perform a time-aligned switching to a new channel. The possibility for time-scheduled execution of UPI functions is especially important for GCPs if a non-real-time backbone networking system like Ethernet is used. In such networks we cannot expect that the WiSHFUL control commands are received by all nodes at the same time, e.g. due to CSMA non-deterministic behavior. Moreover, network congestion and delay are also reasons for providing hierarchical control over UPI_HC between local and global control programs.

4) Remote Execution of UPI Functions

WiSHFUL provides full location transparency. Any UPI function can be executed either locally by a LCP or remotely by a GCP. In the latter case, the WiSHFUL global MCE transparently serializes (marshaling) all input and output arguments. The calling semantic for both the local and remote calls is call-by-value. This has to be considered when extending the UPIs with additional functionality. Finally, as with the local execution also the execution of remote functions can be time-scheduled. This is especially important if a given UPI function needs to be executed at the same time on a set of wireless nodes.

5) Time Synchronization

A wide range of WiSHFUL applications like the centralized control of channel access requires a tight time synchronization among wireless nodes. The way the wireless nodes are time synchronized is platform and architecture-dependent. Basically, we distinguish between systems based on whether a backbone network exists. Here in order not to harm the performance of the wireless network the nodes are time synchronized using the backbone (e.g. Ethernet) and some time-protocol like PTP. Wireless nodes without a backbone have to rely on other techniques for time synchronization (e.g. GPS).

6) Packet Forgery, Sniffing and Injection

WiSHFUL provides a wide range of functionality for packet forgery, sniffing and injection. A control programs can use this to create and inject network packets into the network stack of a node or to receive copies of packets. All WiSHFUL nodes support the sniffing and injection on IP layer (layer 3).

7) Deployments of new UPI functions

WiSHFUL provides an open and extensible architecture, which can be easily extended by new UPI functions. Any new introduced UPI function can be implemented in a different way for different platform and software architecture. Therefore, in WiSHFUL for each platform there is separate connector module, as discussed above. Moreover, some WiSHFUL platforms support the deployment and execution of new UPI functions "on-the-fly" from the GCP.

8) Global Control

To enable remote usage of UPI functions using the UPI_G interface, a system is required that supports remote procedure calls. For this purpose the UPI functions must have a generic signature that facilitates serializing function arguments during remote UPI calls. Such types of function definitions are very flexible but error-prone in usage. For this reason, more user-friendly versions are also required that shields end-user from the complexity by offering strongly typed interface descriptions. The WiSHFUL architecture thus defines two levels of UPIs: i) a generic interface that allows to facilitate serialisation of arguments and ii) a UPI helper interface that wraps the functions in more strongly typed versions. The second version is offered in Python by Helper classes, in C by Helper functions.

III. CONCLUSION

Advancing wireless communications requires overcoming several challenges. Herein, several such challenges have been examined in the form of motivating scenarios. These scenarios outline a number of requirements on experimentation platforms for investigating the future of wireless communications. The WiSHFUL project directly addresses these challenges and requirements by defining a framework to support unified experimentation across several platforms.

ACKNOWLEDGMENT

The research leading to these results has received funding from the European Horizon 2020 Program under grant agreement n645274 (WiSHFUL project).

REFERENCES

[1] Fortuna, C, et al. (2015). Wireless Software and Hardware platforms for Flexible and Unified radio and network controL. European Conference on Networks and Communications (EuCNC 2015), Workshop on 5G Testbeds and Hands-on Experimental Research, Paris, France, June 29, 2015

[2] Wauters, Tim, et al. "Federation of Internet experimentation facilities: architecture and implementation." European Conference on Networks and Communications (EuCNC 2014). 2014.

[3] Sutton, Paul, et al. "Iris: an architecture for cognitive radio networking testbeds." Communications Magazine, IEEE 48.9 (2010): 114-122.

[4] Bart Jooris; Eli De Poorter; Peter Ruckebusch; Peter De Valck; Christophe Van Praet; Ingrid Moerman; TAISC: a cross-platform MAC protocol compiler and execution engine; Under submission for Computer Networks.

[5] Tinnirello Ilenia, et al. "Wireless MAC Processors: Programming MAC Protocols on Commodity Hardware." In INFOCOM, 2012 Proceedings IEEE, 1269–77. IEEE, 2012.

[6] Ruckebusch, P.; De Poorter, E.; Fortuna, C.; and Moerman, I. (2015). GITAR: Generic extension for Internet-of-Things Architectures enabling dynamic updates of network and application modules. Ad Hoc Networks, January 2016, Vol. 36, Part 1, Pages 127–151

ABOUT THE AUTHORS

Nicholas Kaminski currently is a Research Fellow at the CONNECT (formerly CTVR) Telecommunications Research Centre at Trinity College Dublin (Ireland). He conducts research focused on extending the bounds of wireless technology by deploying targeted intelligence to act in harmony with flexible radio systems. He advances distributed radio intelligence through experimentation-based research and is co-principal investigator on several European projects that create and develop experimentation platforms and services for an open infrastructure. Dr. Kaminski received his master‘s degree in Electrical Engineering in 2012 and his Ph.D. in 2014 from Virginia Tech, USA. During this time he was funded as a Bradley Fellow at the Bradley Department of Electrical and Computer Engineering at Virginia Tech. He is a co-inventor on a preliminary patent application for his work extending the utility of cognitive radios.

Ingrid Moerman received her degree in Electrical Engineering (1987) from the Ghent University, where she became a part-time professor in 2000. She is a staff member of the research group on Internet-Based Communication Networks and Services, IBCN where she is leading the research on several interdisciplinary research projects. At the national level, Ingrid Moerman has an extensive experience with leading SBO, FWO, and iMinds ICON projects. At the European level, Ingrid Moerman is in particular very active in the FP7 FIRE research area, where she is participating in OpenLab, Fed4FIRE, EVARILOS, FORGE, and FLEX. In the FP7 research on Future Networks, she is involved in LEXNET and SEMAFOUR. In Horizon 2020 she is coordinating WiSHFUL. Ingrid Moerman is author or co-author of more than 500 publications in international journals or conference proceedings.

Spilios Giannoulis is a post doc researcher in the IBCN group of iMinds-UGent. He received his Diploma in Electrical and Computer Engineering in 2001 and his PhD (2010) in the area of routing for wireless mobile sensor networks. Since 2001 he has been a researcher involved in several E.U. and national R&D projects at Industrial Systems Institute (ISI) and the Applied Electronics Laboratory (APEL), University of Patras, Greece. From 2005 until 2014, he was also lecturing at the Technological Educational Institute of Patras, Greece. His main research interests are in the fields of wireless networks, mobile ad-hoc networks, wireless sensor networks, especially MAC and routing protocols, QoS provisioning, cross-layer and power aware architecture design, with relevant publications in international conferences and journals.

Peter Ruckebusch received his B.S. and M.S. in Computer Science from Hogeschool Ghent faculty engineering, Belgium. Peter has many years of experience in developing network solutions of networked embedded systems, especially wireless senor networks. Since 2011 he has been a Ph.D. student at University Ghent, iMinds in the department of Information Technology (INTEC). He has been/is collaborating in several national and European projects. His research topics are situated in the low-end of the IoT mainly focusing on re-configurability and re-programmability aspects of protocol stacks for constrained devices in IoT networks.

Pierluigi Gallo is Assistant Professor at the University of Palermo since November 2010. He graduated with distinction in Eelctronic Engineering in July 2002 and worked at CRES (Electronic Research Center in Sicily) until 2009. His work was dedicated to QoS in IP core routers, IPv6 network mobility and wireless networks. His research has focused on wireless networks at the MAC layer and 802.11 extensions, localization based on the time of arrival and cross layer solutions. P. Gallo has worked in several national and European research projects: ITEA- POLLENS (2001-2003) on a middleware platform for a programmable router; IST ANEMONE (2006-2008) about IPv6 mobility; IST PANLAB II on the infrastructure implementation for federating testbeds; ICT FLAVIA (2010-2013) on Flexible Architecture for Virtualizable future wireless Internet Access, CREW (2012-2015) on cognitive radios and software defined networks, WISHFUL (2014-to date) on unified control of hardware platforms for wireless experimentation. He is the unit coordinator for the national project "Smart health 2.0", on cloud computing for e-health.

Anatolij Zubow is a senior researcher in the Telecommunication Networks Group at the Technische Universtät Berlin, Germany. He received his PhD degree from Humboldt Universtät zu Berlin in 2009. His main research interest is cooperative mesh networking in wireless ad-hoc networks as well as cognitive radio. In addition he worked on several WLAN and WiMAX related projects.

Robin Leblon obtained a Master of Science in Computer Engineering in 2010 from Ghent University after spending a year at Universtat Politècnica de Catalunya- Barcelona Tech. Robin started as an R&D engineer at nCentric in 2011 working on the ROBIN-R8 (offshore wireless mesh node). Currently he is responsible for the R&D department working on different wireless networking products.

Ivan Seskar is Associate Director at WINLAB, Rutgers University responsible for experimental systems and prototyping projects. He is in charge of experiment deployment on the NSF-funded Mobility First Future Internet Architecture project. Ivan is one of the co-PIs and a lead project engineer for ORBIT, with responsibility for development, integration, and deployment of the radio grid emulator system for which the team received 2008 NSF Alexander Schwarzkopf Prize for Technological Innovation. He is currently the PI for the NSF GENI Open WiMAX base station project that has resulted in deployments at several US universities and for the "meso-scale" Open-Flow campus deployment at Rutgers University, co-PI on the NSF funded SciWiNet (research MVNO) project, and co-PI of a couple of cognitive radio projects. His current research interests are in future internet architectures, mobile and ad-hoc networks, wireless testbeds, and cognitive radios. Ivan is also co-founder and CTO of Upside Wireless Inc.

Sunghyun Choi is Full Professor at the Department of Electrical and Computer Engineering, Seoul National University (SNU), Korea. Before joining SNU in 2002, he was with Philips Research USA. He was also a visiting associate professor at Stanford University from June 2009 to June 2010. He received his B.S. (summa cum laude) and M.S. from KAIST in 1992 and 1994, respectively, and his Ph.D. from the University of Michigan in 1999. His research interests are in the area of wireless/mobile networks. He co-authored over 180 papers and holds over 110 patents. He co-authored "Broadband Wireless Access and Local Networks: Mobile WiMAX and WiFi", Artech House, 2008. He served on program and organization committees of numerous leading wireless and networking conferences. He is currently serving on the editorial boards of IEEE Transactions on Mobile Computing and IEEE Wireless Communications, and served as guest editor of IEEE Journal on Selected Areas in Communications. From 2000 to 2007, he was an active voting member of IEEE 802.11 WLAN Working Group. He received numerous awards including the Presidential Young Scientist Award (2008), IEEK/IEEE Joint Award for Young IT Engineer (2007), and KICS Dr. Irwin Jacobs Award (2013). He is a Fellow of IEEE.

Jose de Rezende received the B.Sc. and M.Sc. degrees in Electronics Engineering from Universidade Federal do Rio Janeiro (UFRJ) in 1988 and 1991, respectively. He received the Ph.D. degree in Computer Science from Université Pierre et Marie Curie in 1997, where he was an associate researcher during that year. Since 1998 he has been an associate professor at UFRJ. His research interests are in wireless networks distributed multimedia applications, multi-peer communication, high speed and mobile networks, and Future Internet. He has served in the editorial board of Ad Hoc Networks from Elsevier since 2006.