WiSHFUL: enabling coordination solutions for managing heterogeneous wireless networks.

Peter Ruckebusch, Spilios Giannoulis, Domenico Garlisi, Pierluigi Gallo, Piotr Gawłowicz, Anatolij Zubow, Mikołaj Chwalisz, Eli De Poorter, Ingrid Moerman, Ilenia Tinnirello, and Luiz DaSilva

***Abstract*—The paradigm shift towards the Internet-of-Things results in an increasing number of wireless applications being deployed. Since many of these applications contend for the same physical medium (i.e. the unlicensed ISM bands), there is a clear need for beyond-state-of-the-art solutions that coordinate medium access across heterogeneous wireless networks. Such solutions demand fine-grained control of each device and technology, which currently requires a substantial amount of effort given that the control APIs are different on each hardware platform, technology and operating system.**

In this paper an open architecture is proposed that overcomes this hurdle by providing unified programming interfaces (UPIs) for monitoring and controlling heterogeneous devices and wireless networks. The UPIs enable to create and test advanced coordination solutions while minimizing the complexity and implementation overhead. The availability of such interfaces is also crucial for the realization of emerging software-defined networking approaches for heterogeneous wireless networks. To illustrate the use of UPIs, a showcase is presented that simultaneously changes the medium access control (MAC) behavior of multiple wireless technologies in order to mitigate cross technology interference taking advantage of the enhanced monitoring and control functionality.

An open source implementation of the UPIs is available for wireless researchers and developers. It currently supports multiple widely used technologies (IEEE-802.11, IEEE-802.15.4, LTE), operating systems (Linux, Windows, Contiki) and radio platforms (Atheros, Broadcom, CC2520, Xylink Zynq, …), as well as advanced reconfigurable radio systems (IRIS, GNURadio, WMP, TAISC).

***Index Terms*— wireless networks, heterogeneous, cross technology interference, software architecture, experimentation, monitoring, network control, radio control, coexistence, cooperation**

I. INTRODUCTION

THE paradigm shift towards the Internet-of-things (IoT) will result in an increasing number of interfering devices that operate in the unlicensed spectrum, especially given the recent interest of the 5G community to also use the same ISM bands. Coexistence will be a huge challenge, as many heterogeneous networks have to cooperate to share the same spectrum efficiently. To this end, advanced coordination techniques must be developed that allow mitigating cross-technology interference.

Currently, multiple custom tools are used to configure and monitor wireless networks and each type of device requires a different toolset. For this reason, controlling a heterogeneous set of wireless devices is cumbersome at least and often demands a considerable effort to get acquainted with the different hardware platforms and corresponding configuration tools.

The proposed control architecture offers the possibility to create and test coordination techniques while minimizing the complexity and implementation overhead, thereby fostering innovations in a challenging research domain. For this purpose, it relies on the following key enablers:

Unified programming interfaces (UPI) allow reconfiguring various features of the network stack and monitoring its state without the need to have deep knowledge of the software and hardware particularities of each platform. The UPIs enable the design of technology-independent control programs (CPs) on top of different hardware and software platforms.

Context aware execution of UPIs enables to define exactly where, when and how a UPI call must be executed. This allows to change a particular configuration value on a group of nodes at a specific time in a synchronized manner.

Connector modules transform each UPI call into one or more platform specific calls thereby hiding the complexity of the underlying tools and/or APIs.

Hierarchical control enables to create multi-level control loops spanning multiple and possibly heterogeneous networks. Hierarchical control allows CPs to delegate control between each-other and to create custom control flows.

The UPIs and the control architecture are integrated in several federated wireless experimentation facilities. They are

The revised paper was submitted for final review on 16/06/2017. "This work was supported by the European Commission Horizon 2020 Programme under grant agreement n645274 (WiSHFUL) and the FWO SBO "SAMURAI: Software Architecture and Modules for Unified RAdIo control"' project.".

Peter Ruckebusch, Spilios Giannoulis, Eli De Poorter and Ingrid Moerman are with the Department of Information Technology - IDLAB - Ghent University – IMEC, Ghent, Belgium (e-mail: first.last@ugent.be).

Domenico Garlisi, Pierluigi Gallo and Ilenia Tinnirello are with the Department of Energy, Information Engineering and Mathematical Models CNIT - University of Palermo, Italy (e-mail: first.last@dieet.unipa.it).

Piotr Gawłowicz, Anatolij Zubow and Mikołaj Chwalisz are with the Department of Telecommunication Systems - TKN - Technische Universität Berlin, Germany (e-mail: {gawłowicz,zubow,chwalisz}@tkn.tu-berlin.de)

Luis DaSilva is with the Department of Telecommunications, Electronic & Elect. Engineering - CONNECT – Trinity College Dublin, Ireland (dasilval@tcd.ie)

offered as an open source tool to the research community and were already successfully deployed both in- and out-side testbed facilities. In this paper, a high level overview (Section IV & V) of the architecture is given together with the results of several experimental showcases (Section VI).

The implemented showcases demonstrate that the proposed architecture simplifies control of standardized technologies, while still offering advanced control of future reconfigurable radio systems.

II. Representative use case

The difficulty of efficiently managing coexisting wireless networks increases significantly when multiple technologies are considered. As a representative use case, this paper will consider an example where coexistence between IEEE-802.11 Wi-Fi and IEE-802.15.4 TSCH (Time Slotted Channel Hopping) is managed by separating them in the frequency and time domain. As such, different frequencies and timeslots must be allocated to networks that are in each-others interference range. To realize this, advanced monitoring, coordination and configuration techniques are required. Moreover, it must be possible to exchange control messages and maintain some level of synchronization between the different devices.

Building such a system is a non-trivial task and requires the use of different domain specific expertise: Linux and Wi-Fi management tools on one hand, and embedded OS (Contiki / openWSN / …) and programming knowledge on the other hand. Moreover, to apply the same solutions to different technologies (Bluetooth for example) or different operating systems (Windows, Unix, TinyOS, ..) would require to re-implement the same control logic all over again.

The proposed architecture aims facilitating control in all aforementioned scenarios by providing the necessary building blocks. First, the unified programming interfaces (UPIs) allow to re-use the same control logic in different set-ups. Second, the context aware execution of UPIs support building solutions that require fine-grained control. Third, the connector modules simplify the process of extending the architecture towards new technologies and platforms.

III. Related Work

A. Control architectures

The need for fine-grained control of communication networks is becoming increasingly apparent. This is well demonstrated by the interest of the scientific community in solutions that enable software defined networking, (SDN). OpenFlow[1], for instance, is a good example of an SDN-enabler because it allows researchers to control routing, without knowing the internals of vendor-specific implementations. OpenFlow, however, focuses on controlling the forwarding rules between devices (switches, routers and wireless access points) connected by means of pre-installed links (usually wired).

Recently, a number of solutions were proposed that enable software defined wireless networks (SDWN) such as 5G-EmPOWER[2], OpenSDWN[3] and Sensor OpenFlow[4]. The latter two focus on enabling SDWN in a single technology (i.e. IEEE-802.11 and IEEE-802.15.4 respectively). 5G-EmPOWER is broader in scope and provides programming abstractions for managing both Wi-Fi access points and LTE eNodeBs. However, not a single architecture exists today that can facilitate true cross layer control (from PHY layer up to network layer and in some cases up to the presentation layer of the OSI model) in a unified way across multiple wireless technologies. Our proposed WiSHFUL architecture aims to go further by providing abstractions for any device and wireless technology. Furthermore, to the best of our knowledge, our architecture is the first to include reconfigurability of the MAC and PHY layers which strongly affect the link availability and capacity. As such, the WiSHFUL architecture addresses this gap by offering full-stack cross-layer and cross-network control of reconfigurable wireless networks.

The WiSHFUL architecture was first conceptually presented in[5][6]. Now we focus on the novel features such as context aware execution and hierarchical control that allowed us to implement and evaluate the experimental showcases, illustrating how to build cross-technology coordination solutions.

B. Federation of experimentation facilities

Since most SDN solutions have been evaluated in wireless testbed, the federation of (wireless) testbeds [7][8] gained much attention over the last years. Federated testbeds aim to accelerate experimental research by providing easy reservation of experiment time slots as well as the corresponding access to resources (radios, spectrum monitoring, mobile robots, etc.) residing in different testbeds. Despite the clear progress that has been made, executing an experiment still requires manual combination and integration of different vendor or technology specific tools to reconfigure and monitor the devices under test. This imposes a huge burden on the experimenters since they need deep knowledge of the tools at hand, even for setting up a novice experiment. The proposed WiSHFUL architecture builds further on top of testbed federation tools to support easy experimentation using heterogeneous systems to a user base with a diverse skill-set.

C. Reconfigurable radio systems

The proposed architecture supports commonly used operating systems (Linux, Contiki) for standard wireless technologies (IEEE-802.11, IEEE-802.15.4). In addition, the architecture also supports emerging state-of-the-art standards (such as ETSI-RRS[9]) and novel reconfigurable radio systems that allow more fine-grained control over the radio than is possible with typical off-the-shelf radio chips. Currently, four advanced open reconfigurable radio systems are supported: Wireless MAC Processor (WMP) for IEEE-802.11 radios[10], Time-Annotated Instruction Set Computer (TAISC) for IEEE-802.15.4 radios[11], GNU radio and the Implementing Radio in Software (IRIS) for software defined radios (SDR)**Error! Reference source not found.**.

These novel architectures allow the design of state-of-the-art techniques[13] for managing coexistence between devices. For instance, they enable to separate medium access in the time domain, effectively allowing to enforce a cross technology TDMA scheme. However, although they are very flexible, several of these frameworks lack proper documentation and require learning yet another programming language and programming framework, thereby imposing a steep learning curve on wireless researchers and developers before they can be used. The availability of simple, cross-technology WiSHFUL UPIs remedies these shortcomings and allows integration of these advances platforms with traditional radio platforms.

IV. WiSHFUL ARCHITECTURE AND CONCEPTS

To lower the threshold for building coexistence solutions, a novel control architecture was designed and created within the WiSHFUL project. The left side of Figure 1 illustrates the main architectural blocks discussed in this section. The simplified code snippets on the right side exemplify a remote control program (upper), UPI definition (middle) and a connector module (lower).

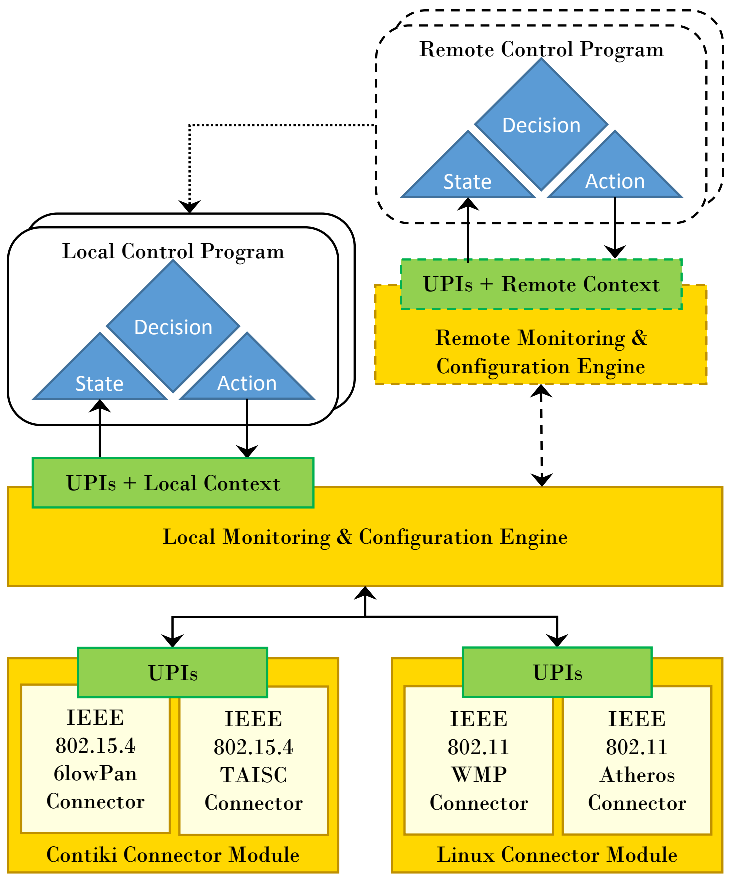
A. Control Programs

The control programs (CP, top of the figure) execute the user-defined control logic. They build up a view on the network state by collecting monitoring information which can be used to drive decisions leading to configuration actions. For this purpose they use a set of Unified Programming Interfaces (UPIs) in a particular execution context.

The control programs can be used locally, on the node, and/or remotely, within a subnet of nodes or across different networks. Control programs can be simple rule-based scripts, but can also comprise more intelligent components, allowing to build a fully self-organizing network.

By allowing interactions between control programs (dotted arrows) it is possible to implement a hierarchical control logic where local CPs execute time-sensitive control loops, while remote CPs gather information from- and take decisions on a group of nodes.

The upper code snippet demonstrates how a remote control program uses the UPIs to configure the Wi-Fi network on a particular IEEE-802.11 channel and blacklist the overlapping IEEE-802.15.4 channels in the TSCH network. The example also illustrates how an execution context can be attached to a UPI function.


Remote Control Program
Decision
State
Action
UPIs + Remote Context
Remote Monitoring & Configuration Engine
Local Control Program
Decision
State
Action
UPIs + Local Context
Local Monitoring & Configuration Engine
UPIs
IEEE 802.15.4 6lowPan Connector
IEEE 802.15.4 TAISC Connector
Contiki Connector Module
UPIs
IEEE 802.11 WMP Connector
IEEE 802.11 Atheros Connector
Linux Connector Module

Remote Control Program

```
# create remote engine
control_engine = create_control_engine()

# discover Wi-Fi and TSCH nodes
wifi_nodes = control_engine.discover_nodes("wifi")
tsch_nodes = control_engine.discover_nodes("tsch")

# change channel on Wi-Fi nodes in 5 sec
control_engine.create_context(wifi_nodes, now + 5)
control_engine.set_channel(6)

# blackist overlapping channels on TSCH nodes
control_engine.create_context(tsch_nodes, now + 5)
control_engine.blacklist_channels([16,17,18,19])
```

UPI definition

```
#generic UPI functions
class phy(upi):
  # generic set channel function
  def set_channel(channel)

# IEEE80211 specific UPI functions
class IEEE80211(phy):
  # IEEE80211 set channel
  def set_channel(channel)
```

Linux IEEE-802.11 Connector Module

```
# binding of local function to UPI functions
@bind_function(upis.phy.set_channel)
@bind_function(upis.phy.IEEE80211.set_channel)
def linux_connector_set_channel(self, channel):
        # Linux specific implementation using iw
        cmd = "iw phy" + self.phy + "set channel"
        cmd += channel
        return self.execute_command(cmd)
```

Figure 1 A high level overview of the WiSHFUL architecture (left side) and example code snippets (right side). The architecture features both local and remote control, as well as context aware execution. For each platform and technology, connector modules adapt generic UPI calls to platform specific calls. The upper code snippet demonstrate the use of UPIs in a remote control program. The lower code snippet illustrates how generic UPI calls are mapped to platform specific calls.

B. Unified Programming Interfaces

The UPIs (green blocks) provide generic hooks that enables controlling the behavior of the network stack on a heterogeneous set of nodes by exposing common functions to monitor and configure networked devices in any layer of the protocol stack (i.e. from PHY to APP). Both request (pull) and event-based (push) UPIs, are provided for monitoring the state and performance of the network.

There is a 2-tier unification for protocol control interfaces:

1. A unification across different platforms and implementations (e.g. the same IEEE-802.11 parameters provided in an identical way for Windows and Linux platforms).
2. A unification across technologies and protocols with similar behavior (e.g. CSMA parameters for both IEEE-802.11 and IEEE-802.15.4).

The UPIs also include meta-information that allows to reason on logical connections between different implementations (e.g. set_channel on IEEE-802.11 and IEEE-802.15.4). The example snippet in the middle illustrates the 2-tier unification of UPIs for the *set_channel* function.

The UPIs focus on common control functions, which are found in most typical radio platforms and networking standards. For control features that are not yet supported across multiple technologies, we offer the possibility to support them as technology/platform specific APIs in an intuitive manner.

C. Monitoring and configuration engine

The monitoring and configuration engines (MCE, dark yellow blocks) implement the core WiSHFUL services required for controlling one or more wireless nodes. Since the nodes have diverse capabilities and can reside in different networks, providing such services is a non-trivial task. The MCEs provide the following core WiSHFUL services:

- Remote execution: UPIs can be executed both locally and remotely on one or more nodes using remote procedure calls.
- Context-aware execution: it is possible to specify exactly **how** (blocking or non-blocking), **where** (one or more nodes in the same or different networks) and **when** (exact time or relative delay) UPI functions are executed.
- User-defined control flows: the architecture allows establishing a dedicated control channel between CPs thereby enabling custom interactions. In addition, control logic can be injected on-the-fly, allowing delegation of control between CPs.
- Support services such as node discovery and time synchronization that work across different networks and on platforms with different capabilities.

More details concerning the discussed services can be found in [14].

D. Connector modules

The connector modules (light yellow blocks) transform the generic UPI calls to platform specific calls. They are implemented on each platform and for each technology. In most cases they are a simple wrapper around existing configuration tools such as *netlink* and *iw* suffices. In other cases custom extensions are required to enable the functionality of UPIs.

The connector modules are dynamically loaded by the Monitoring and Configuration Engine based on the platforms and technologies used in the set-up. This implies that the set of active UPIs changes over time and can be tailored towards the specific needs of a solution.

The example in the lower code snippet illustrates how the Linux *iw* command is wrapped in the platform specific *set_channel* function. This function is then bound to both the generic and IEEE-802.11 UPI function *set_channel.*

V. UPI ENABLED CONTROL PLANE IN WIRELESS EXPERIMENTATION FACILITIES

The control plane extensions offered via the UPIs allow optimizing the QoS in all networks under control, not only by considering node-local and in-network optimizations but also by taking into account the cross-technology interaction (e.g. interference) between the different networks.

Figure 2 demonstrates how a hierarchical control plane can be built using the WiSHFUL architecture. The control programs (blue shapes) can be executed on different logical levels, allowing to place delay-sensitive operations close to the hardware while maintaining a broader, network-wide or cross-network view on a higher level. The figure depicts three logical levels of control: node-local, in-network and cross-network. Each level can directly use the UPIs (dashed arrows) or delegate control to another level (dotted arrows). For instance, a cross-network control program can directly monitor single devices or delegate monitoring processes to the local level and work on aggregated values to reduce the amount of data to be transferred over the network.

A. UPI control channels

Two types of control channels can be employed to enable monitoring and configuring nodes across different networks. Beside the default UPI control channel, i.e. between a (local or remote) control program invoking UPIs, and the node through the MCEs, it is also possible to set-up communication channels between control programs of different levels (node-local, in-network and cross-network). These communication channels can be used to share information and delegate control functionality between different control programs.

This enhances the flexibility in creating the control programs because researchers can, for instance, choose to aggregate monitoring information on the node-local level and only forward information in a custom format. It is also possible to execute certain configuration tasks node-locally on the fly triggered by a central control program.

B. UPI multi-level control loops

The ultimate goal of the UPIs is to enable the creation of multi-level control loops that can span between different networks. In each level, a control program uses UPIs to monitor the network performance and state. Based on this information, the CPs can decide to change the network behavior by executing configuration commands, employing UPIs. The types of control loops made possible by the proposed architecture are presented

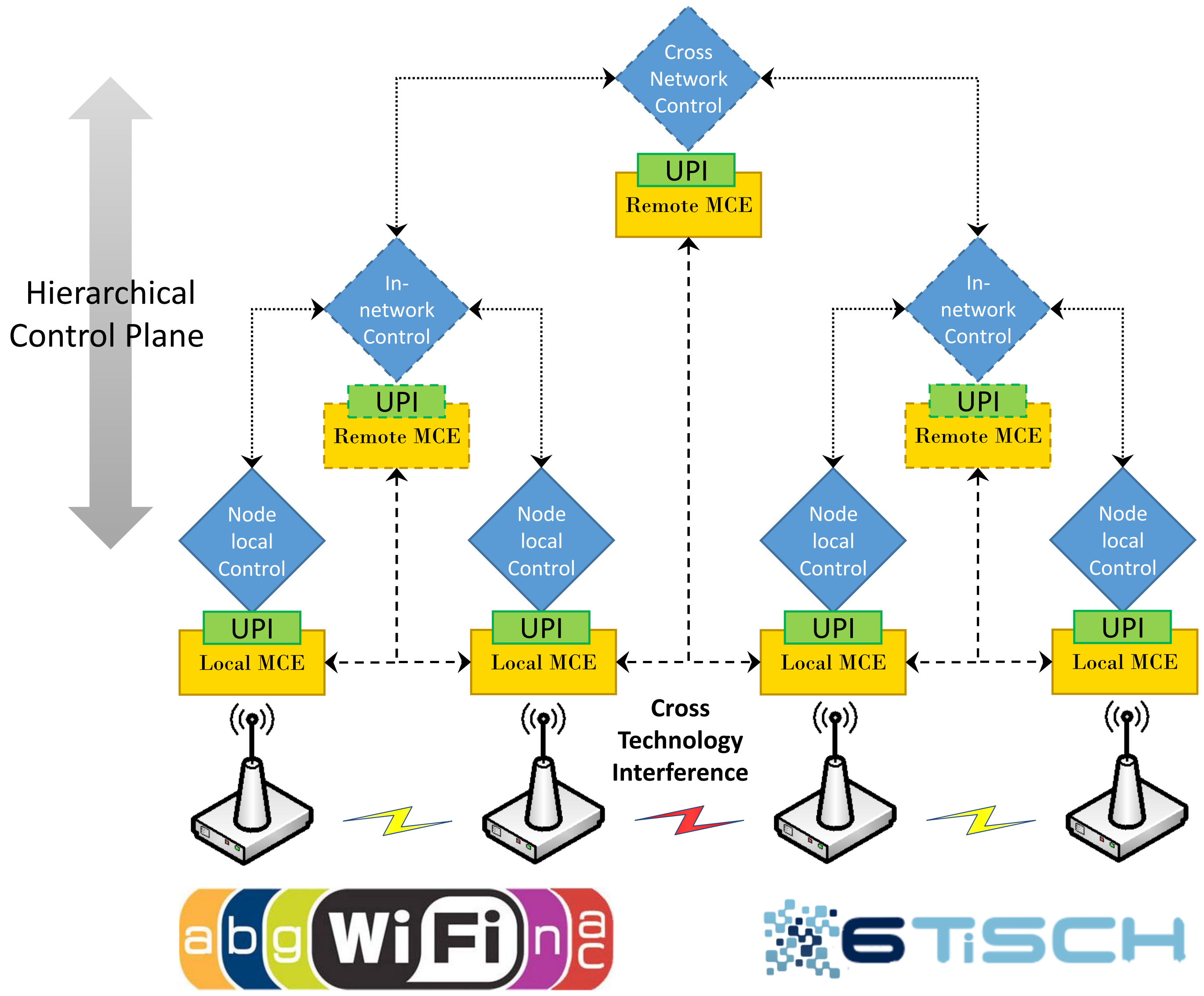


Figure 2 illustrates the possibility to build a hierarchical control plane using the WiSHFUL architecture. Two types of control flows are enabled: 1) UPI based, between control programs and UPIs; or 2) User defined, between control programs.

below:

1) Node-local control loop

The first level provides the possibility to create a node-local control loop where local decisions are made based on information observed locally via the UPIs or received from other control programs via a user-defined control channel. The node-local reconfiguration always uses the UPIs directly. This local approach is efficient to implement quick reactions to the rapidly changing context. The delay of a local UPI call is usually in the order of microseconds, depending on the complexity and the CPU speed.

2) In-network control loop

The second level enables to control all nodes in a logical network (i.e. the nodes are in the same "subnet" and use the same technology). Now, network-wide monitoring drives decisions and configuration settings are changed on a single or on a group of nodes in the network. The information can be retrieved using UPIs remotely or from the node-local CPs. Similarly, network reconfiguration commands can be done remotely, using the UPIs, or via control delegation. The delay of a UPI call inside a network is typically in the order of milliseconds, depending on the network latency and bandwidth.

3) Cross-network control loop

In many cases, control is required across network and technologies (e.g. interference avoidance between different technologies in in the ISM band). For this purpose, the architecture allows creating a cross-network control loop that regulates the medium access between different networks. The interactions are similar to the in-network control loop except that they can now span multiple networks. The typical delay of a UPI call across different networks is in the order of 100' milliseconds and is mainly influenced by the latency of the backbone network.

C. Supported experimentation facilities

The WiSHFUL architecture is currently fully supported in the imec iLab.t[1], TU Berlin TWIST[2], Rutgers University ORBIT lab[3] and TCD Iris[4] wireless experimentation facilities. **Table 1** lists the communication technologies, operating systems (OS), hardware platforms and drivers controlled using the UPIs. With minimal effort, UPI support can be given to experiment facilities that use (a subset of) the technologies listed below. Support for other technologies such as Bluetooth, LoRa and SigFox is planned in the near future.

Technology	*Operating System*	*Hardware platform*	*Hardware driver*
IEEE-802.11	Linux, Windows	Atheros, Broadcom	Ath9k, NDIS driver, WMP
IEEE-802.15.4	Contiki, TinyOS	MSP430, ARM-Cortex-M	Contiki TinyOS, TAISC
LTE	Linux	Linux Server Femtocell	SIRRAN EPC LTE 245F
SDR	Linux, Windows	USRP, Xilinx ZedBoard	Iris, LabView, GNU radio

Table 1 Main overview of supported technologies, operating systems, hardware platforms and drivers.

In terms of memory overhead, the full WiSHFUL framework requires only 0.75% of the 512 kB ROM and 3% of the 32 kB RAM on the employed embedded Zolertia Remote Cortex-M3 devices, making it feasible to support WiSHFUL even on constrained devices.

D. In-band versus out-of-band control channels

To support solutions beyond experimentation, the control channels can be set-up both out-of-band and in-band. The in-band control channel shares the (wireless) communication channels of the devices with the data flows while the out-of-band control channel uses the backbone network provided by the experimental facilities for transferring control flows. Using the latter approach, it is possible to separate the control flows physically from the data flows, thereby allowing evaluating control strategies without impacting the applications.

In real-life deployments (when no testbed backbone is available), however, only in-band control channels can be employed, introducing overhead and impacting the performance of the network. The WiSHFUL architecture supports in-band control channels and allows evaluating the impact of the control flow overhead.

VI. Experimental Showcases

In this section, the strengths of the WiSHFUL architecture are demonstrated by listing results that were obtained when conducting several advanced wireless experiments. Without the presented architecture, a deep knowledge of the particular details of each platform and related tools would have been required. Thanks to the WiSHFUL architecture, each showcase only required creating a generic control program which could then be used repeatedly during experimental validation and evaluation.

The showcases are grouped and discussed by topic. The results shown in this section were obtained on the imec w.iLab.t testbed using 32 RM-090 (MSP430 CPU based) sensors equipped with a CC2520 IEEE-802.15.4 radio, running Contiki/TAISC; and 8 embedded Linux devices equipped with a Broadcom IEEE-802.11b/g card running WMP.

A. Load and topology aware MAC adaptations

This showcase illustrates how the UPIs can be used to apply the same MAC adaptations on two different platforms and technologies, investigating their applicability in a heterogeneous set-up and evaluating the differences between technologies. It is important to note that in both cases, the same control programs were used.

Figure 3 compares the overall network throughput (blue line is RX throughput, green line is TX attempts, dashed black line is number of senders) for both technologies in two phases, initially a CSMA/CA protocol with a contention window optimization algorithm is applied and, in a second phase, a TDMA protocol is activated. In this experiment, the active traffic flows were increased gradually by activating the senders one-by-one until a pre-defined maximum, after which TDMA is activated.

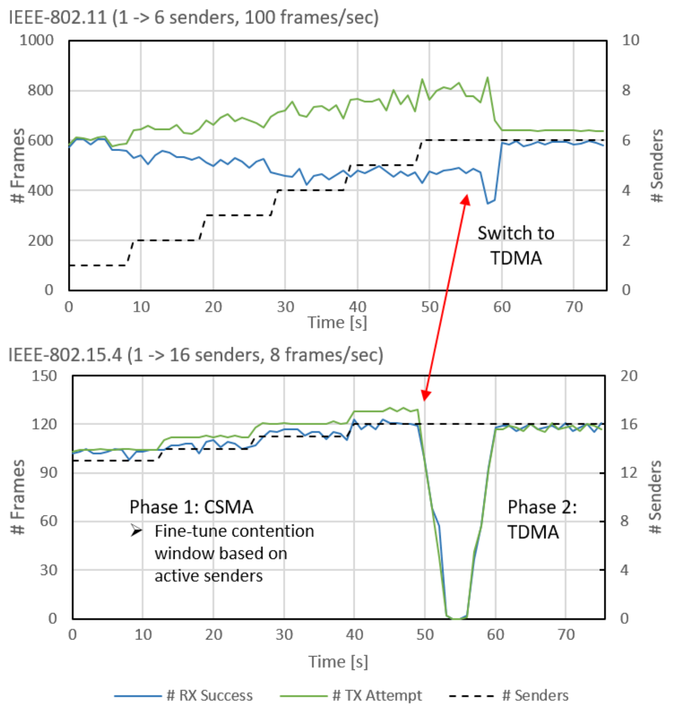


Figure 3 The graphs show the number of received frames (blue) vs. the number transmitted frames (green) for an increasing number of senders (dashed black). This experiment was conducted both on IEEE-802.11 nodes (upper chart) and IEEE-802.15.4 nodes (lower chart).

[1] http://ilabt.iminds.be/ [accessed on 12/06/2017]
[2] http://www.twist.tu-berlin.de/ [accessed on 12/06/2017]
[3] http://www.orbit-lab.org/ [accessed on 12/06/2017]
[4] http://iris-testbed.connectcentre.ie/ [accessed on 12/06/2017]

The applied algorithm adapts the CSMA/CA contention window based on the number of active traffic flows in the network. It can be expected that after a while, applying this technique does not yield a higher RX throughput and packet loss starts to increase due to collisions. At this point, it is more efficient to switch to a TDMA protocol. The exact tipping point depends on many factors such as number of senders and the application data rate. Figure 3 shows a snapshot of such a tipping point during an experiment.

B. Co-existence of heterogeneous technologies

This showcase demonstrates that the WiSHFUL architecture can be used to implement advanced strategies to solve the use case presented in Section II, i.e. coexistence between IEEE-802.11 Wi-Fi and IEEE-802.15.4 TSCH. This showcase exploits the hierarchical control features as well as the built-in synchronization support. Moreover, it also illustrates how the architecture supports both standardized platforms and technologies, as well as state-of-the-art frameworks.

Two different approaches were evaluated. The first solution uses the standard channel blacklisting feature in IEEE-802.15.4e TSCH, to avoid channels used by the IEEE-802.11 Wi-Fi network. The second solution uses a state-of-the-art implementation where a time-slotted MAC (TDMA) is applied in both networks based on a shared synchronization beacon and TDMA schedule.

The upper part of Figure 4 shows the overall network throughput in the blacklisting scenario (blue line is RX throughput, green line is TX attempts, red line is TX request fails). The results clearly show that the throughput of the IEEE-802.15.4 nodes drop in case of IEEE-802.11 interference. This is mainly due to synchronization loss caused by interfered beacons. After the blacklisting of interfered IEEE-802.15.4 channels, the throughput stabilizes again to the level before adding IEEE-802.11 interference.

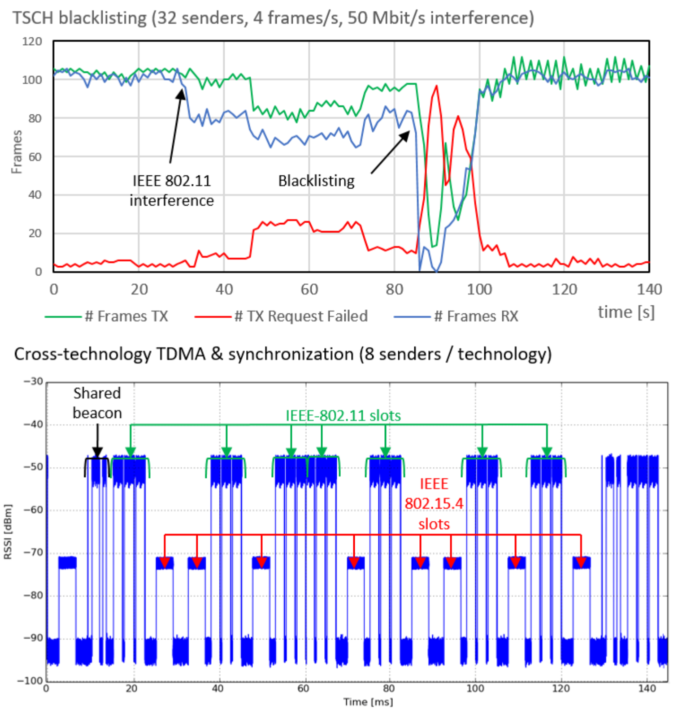


Figure 4 shows results from two experiments that evaluate coexistence strategies. In the first experiment (upper part), the channel blacklisting features of the TSCH (Time Slotted Channel Hopping) MAC is used to avoid channels with high IEEE-802.11 interference. The second experiment (lower part) illustrates a solution where a TDMA schedule and synchronisation are shared across heterogeneous technologies.

The lower part of Figure 4 shows an energy plot obtained by a USRP device operating in energy detection mode, while testing the second solution. The results clearly demonstrate that an IEEE-802.15.4 network can be synchronized using a cross-technology beacon sent by a TDMA MAC implementation of an IEEE-802.11 network. The IEEE-802.15.4 nodes use energy detection to search for a particular beacon pattern transmitted by the IEEE-802.11 access point. The WiSHFUL architecture allows distributing both the beacon pattern and cross-technology TDMA scheme amongst both IEEE-802.11 and IEEE-802.15.4 nodes, enabling separation of both networks in the time-domain. A more detailed discussion of this particular experiment can be found in [13].

VII. Conclusions

In the context of 5G, coexistence is a huge challenge, as many heterogeneous networks will have to cooperate to share the same spectrum efficiently. To this end, solutions are required that allow detailed networks insights, fine-grained network control and management, etc. The WiSHFUL framework offers the possibility to create and test such solutions while minimizing the complexity and implementation overhead, thereby fostering innovations in a challenging research domain.

Foremost, the WiSHFUL architecture offers a unified set of programming interfaces (UPIs) on top of a heterogeneous set of technologies, platforms and protocol stacks, thereby drastically reducing the time and complexity typically required to build innovative solutions.

Furthermore, the architecture offers the possibility to execute control logic on different hierarchical levels (i.e. node-local, in-network or cross-network) in a context-aware manner. This enables defining exactly where, when and how the UPIs are used. The presented cross-technology TDMA scheme fully exploits these features in order to synchronize and coordinate medium access between IEEE-802.11 and IEEE-802.15.4 nodes while retaining the ability to reschedule the slot allocation within the TDMA superframe at runtime.

The design of the architecture also incorporates the possibility for extensions towards new platforms and technologies. This requires only the creation of connector modules implementing and/or extending the UPIs for the particular platforms or technologies. For instance, adding support for controlling LTE networks was not a huge effort, allowing to investigate future 5G challenges such as coexistence between LTE and other technologies in the ISM band.

Finally, all solutions are publicly available as open-source implementations on https://github.com/wishful-project[5] .

ACKNOWLEDGEMENT

This work was supported by the European Commission Horizon 2020 Programme under grant agreement n645274 (WiSHFUL) and the FWO SBO "SAMURAI: Software Architecture and Modules for Unified RAdIo control"" project.

REFERENCES

[1] N. McKeown *et al.*, "OpenFlow: enabling innovation in campus networks," *SIGCOMM Computer Communication*, vol. 38, no. 2, March 2008, pp. 69-74.

[2] R. Riggio *et al.*, "Programming Abstractions for Software-Defined Wireless Networks," *IEEE Transactions on Network and Service Management*, vol. 12, no. 2, June 2015, pp. 146-162.

[3] J. Schulz-Zander *et al.*, "OpenSDWN: programmatic control over home and enterprise WiFi," in *Proc.* 1st *ACM SIGCOMM SOSR*, Santa Clara, CA, USA, June 17 - 18, 2015.

[4] T. Luo, H. P. Tan and T. Q. S. Quek, "Sensor OpenFlow: Enabling Software-Defined Wireless Sensor Networks," *IEEE Communications Letters*, vol. 16, no. 11, Nov. 2012, pp. 1896-1899.

[5] C. Fortuna *et al.*, "Wireless Software and Hardware platforms for Flexible and Unified radio and network controL," in *Proc. 2nd EuCNC*, Paris, France, June 29 – July 2, 2015, pp. 712-717.

[6] N. Kaminski, *et al.*, "Unified Radio and Network Control Across Heterogeneous Hardware Platforms," in *Proc. ETSI Workshop on Future Radio Technologies: Air Interfaces*, Sophia Antipolis, France, Jan 27-28, 2016, pp. 1–10.

[7] M. Berman *et al*., "GENI: A federated testbed for innovative network experiments," *Computer Networks*, vol. 61, 2014, pp. 5-23.

[8] W. Vandenberghe *et al.*, "Architecture for the heterogeneous federation of future internet experimentation facilities," in *Proc. Future Network & Mobile Summit*, Lisbon, Portugal, Jul. 2013, pp. 1-11.

[9] Y. Jin *et al.*, "ETSI reconfigurable radio system: Standard architecture and radio application," in *Proc. 7th ICTC*, Jeju Island, Korea, Oct 19-21, 2016, pp. 1094-1097.

[10] I. Tinnirello *et al.*, "Wireless MAC Processors: Programming MAC Protocols on Commodity Hardware," in *Proc. 31st IEEE INFOCOM*, Orlando, FL, USA, March 25-30, 2012, pp. 1269-1277.

[11] B. Jooris *et al.*, "TAISC: a cross-platform MAC protocol compiler and execution engine," *Computer Networks*, vol. 107, 2016, pp. 315-326.

[12] P. D. Sutton *et al.*, "Iris: an architecture for cognitive radio networking testbeds," *IEEE Communications Magazine*, vol. 48, no. 9, Sept. 2010, pp. 114-122.

[13] P. Ruckebusch *et al.*, "Cross-technology wireless experimentation: Improving 802.11 and 802.15. 4e coexistence," in *Proc. 17th IEEE WoWMoM*, Coimbra, Portugal, June 21-24, 2016, pp. 1-3.

[14] P. Gawłowicz *et al.*, "UniFlex: A Framework for Simplifying Wireless Network Control," to appear in *Proc. IEEE ICC*, Paris, France, May 21-25, 2017.

Peter Ruckebusch (peter.ruckebusch@ugent.be) received his M.Sc. in Computer Science from Hogeschool Ghent Faculty Engineering, Belgium. Since 2011 he has been a Ph.D. student at University Ghent, IMEC, IDLab, 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.

Spilios Giannoulis [M] (spilios.giannoulis@ugent.be) received his M.Sc in Electrical and Computer Engineering (2001) and Ph.D. (2010) from the University of Patras. Since 2015 he is a postdoc researcher at University Ghent, IMEC, IDLab, in the Department of Information Technology (INTEC). He is involved in several EU projects. His main research interests are mobile ad-hoc networks, wireless sensor networks, especially flexible and adaptive MAC and routing protocols, QoS provisioning, cross-layer and power-aware architecture design.

Eli De Poorter (eli.depoorter@ugent.be) received his M.Sc (2006) in Computer Science Engineering and Ph.D. (2011) from Ghent University. He is now professor at the Department of Information Technology at Ghent University. He is currently also coordinating several national and international projects. His main research interests include wireless network protocols, network architectures, wireless sensor and ad hoc networks, future internet, self-learning networks and next-generation network architectures.

Ingrid Moerman (ingrid.moeman@ugent.be) received her M.Sc. in Electrical Engineering (1987) and Ph.D. (1992) from Ghent University, where she became a part-time professor in 2000. She is also a staff member at IDLab-UGent-IMEC, where she coordinates research activities on mobile and wireless networking. Her research interests include IoT, LPWAN, cooperative networks, cognitive radio networks and flexible hardware/software architectures for radio/network control and management. She has a longstanding experience in coordinating national and EU research funded projects.

Domenico Garlisi (domenico.garlisi@dieet.unipa.it) received his M.Sc. in Telecommunication Engineering (2010) and Ph.D. (2014) from the University of Palermo. From May 2015, he is working as researcher for CNIT (Consorzio Nazionale Interuniversitario per le Telecomunicazioni). He is also working as collaborator researcher and software developer for TTI-Lab (DEIM) at University of Palermo, on smart grid and smart mobility. His current research interests include performance evaluation and medium access control in Wireless LANs, included mesh network.

[5] https://github.com/wishful-project [accessed on 12/06/2017]

Pierluigi Gallo (pierluigi.gallo@dieet.unipa.it) received his M.Sc. with distinction in Electronic Engineering (2002) and Ph.D (2006) from the University of Palermo, where he is an Assistant Professor since 2010. His works and interest focus on Wireless Networks, particularly on MAC layer and its localization applications. He has contributed to several national and European research projects including ITEA POLLENS, IST ANEMONE, IST PANLAB II, ICT FLAVIA, H2020 CREW and WiSHFUL.

Ilenia Tinnirello (ilenia.tinnirello@dieet.unipa.it) received her M.Sc. degree in telecommunications engineering (2000) and Ph.D. (2004) from the University of Palermo, where she is currently an Associate Professor. Her research activities have been focused on wireless networks, and in particular on the design and prototyping of protocols and architectures for emerging reconfigurable wireless networks. She has been involved in several European research projects, among them FP7 FLAVIA, H2020 WiSHFUL, Flex5gWare and Symbiote.

Piotr Gawłowicz (gawłowicz@tkn.tu-berlin.de) received his M.Sc in Electronics and Telecommunications from AGH University of Science and Technology, Krakow, Poland in 2014. Currently, he is working as researcher at TKN Group at TU Berlin, Germany. He is the author or co-author of several technical papers. He has been involved in national and European research projects. His research interests include software defined networking, wireless networks and simulation tools.

Anatolij Zubow (zubow@tkn.tu-berlin.de) received his M.Sc. in Computer Science (2004) and Ph.D.(2009) from Humboldt University Berlin. He is a senior researcher at Telecommunication Networks Group at the Technische Universität Berlin since March 2013, where he is coordinating the research activities in the areas of Cognitive Radio, Wireless Access Networks and Software-Defined Networking. In the past he did research in the area of wireless ad-hoc mesh and self-organized networks.

Mikolaj Chwalisz (chwalisz@tkn.tu-berlin.de) received his M.Sc. in Electrical and Computer Engineering from the Warsaw University of Technology and in Computer Engineering from the Technische Universitat Berlin in 2011, where he has been a PhD student since. His research focus is on coexistence and cooperation of heterogeneous wireless networks. He is actively involved in European and German founded projects working on experimentally-driven solutions and testbed orchestration.

Luiz A. DaSilva [FM] (dasilval@tcd.ie) received his M.Sc. in Electrical Engineering (1988) and Ph.D. (1998) from the University of Kansas. Since 2014, he is Professor at Trinity College Dublin. His research focuses on distributed and adaptive resource management in wireless networks, and in particular radio resource sharing and the application of game theory to wireless networks. He is leading research projects funded by the National Science Foundation, the Science Foundation Ireland, and the European Commission.