GRANT AGREEMENT NO.: 732174

Call: H2020-ICT-2016-2017
Topic: ICT-13-2016
Type of action: RIA

Orchestration and Reconfiguration Control Architecture

D4.1: First operational SDR platforms with end-to-end capabilities

Revision: v.1.0

Work package	WP4
Task	T4.1, 4.2, 4.3, 4.4
Due date	31/12/2017
Submission date	22/12/2017
Deliverable lead	TCD
Version	1.0
Authors	Francisco Paisana (TCD), Wei Liu (IMEC), Ingrid Moerman (IMEC), Aslam Muhammad (IMEC), Xianjun Jiao (IMEC), Vincent Kotzsch (NI), Clemens Felber (NI), Seyed Ali Hassani

	(KUL), Roberto Bomfin (TUD), Martin Danneberg (TUD), Peter Neuhaus (TUD), Ivan Seskar (RUTGERS)
Reviewers	Ana-Belen Martinez (TUD), Alessandro Chiumnto (KUL)
Abstract	This deliverable includes the progress and implementation results obtained in Year 1 on the available SDR platforms with special focus on end-to-end functionality. Each research group describes the reconfiguration and radio slicing capabilities they introduced and integrated in the ORCA control plane. This deliverable will also include a plan for functionality to be implemented in the upcoming year.
Keywords	End-to-end, SDR, control plane, radio slicing, virtualization

Disclaimer

Copyright notice

** R: Document, report (excluding the periodic and final reports)*

DEM: Demonstrator, pilot, prototype, plan designs

DEC: Websites, patents filing, press & media actions, videos, etc.

Project co-funded by the European Commission in the H2020 Programme		
Nature of the deliverable:	R*	
Dissemination Level		
PU	Public, fully open, e.g. web	✓
CI	Classified, information as referred to in Commission Decision 2001/844/EC	
CO	Confidential to ORCA project and Commission Services	

OTHER: Software, technical diagram, etc

EXECUTIVE SUMMARY

With the adoption of Software-Defined Radio (SDR) or other forms of reconfiguration radio technology, it becomes possible to support multiple types of communication services with distinct traffic requirements, using the same underlying network infrastructure. This vision is realised through network slices (NSs), which we define as isolated logical networks instantiated using a portion of the hardware and spectrum resources of a physical network. The ability to instantiate multiple NSs in the same underlying infrastructure, and to tailor each NS to satisfy traffic demands of a particular service provides the practical means to realise the 5G vision of a unifying standard for all types of networks.

The overall ORCA objective is to offer end-to-end SDR platform tools and facilities that enable setting experiments and networks with multiple types of communication technologies or NSs, targeting services with distinct requirements (e.g. throughput, latency). Using these facilities, a network designer is given a varied range of physical layer and control plane solutions to reconfigure its network, and better meet its user traffic demands. The provided solutions are diverse, leveraging FGPAs for high computational loads and low communication latencies and host processing for increased configurability and easier extensibility.

This deliverable focuses on the implemented SDR end-to-end capabilities and respective results obtained by the ORCA partners during Year 1. We subdivide these capabilities into the categories of control-plane improvements, where partners describe the parametric reconfiguration functionality provided by their implemented SDR solutions, and radio slicing, which focuses on how these solutions enable network slicing. In addition, this deliverable includes an overview of how the Year 1 end-to-end SDR improvements have been integrated in the partners' testbeds and showcases, and lists the current extension plans for Year 2.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AGC	Automatic Gain Control
AP	Access Point
AXI	Advanced eXtensible Interface
BPSK	Binary Phase Shift Keying
BBU	Base Band Unit
CNN	Convolutional Neural Network
CSMA/CD	Carrier Sense Multiple Access with Collision Detection
DFT	Discrete Fourier Transform
DMA	Direct Memory Access
DSA	Dynamic Spectrum Access
DSP	Digital Signal Processing
GUI	Graphical User Interface
eNB	eNodeB
FD	Full-Duplex
FO	Frequency Offset
GFDM	Generalized Frequency Division Multiplexing
HALO	Hardware in the loop
HLS	High Level Synthesis
IBFD	In-Band Full-Duplex
IP	Internet Protocol
KVM	Kernel Virtual Machine
LTE	Long Term Evolution
LBT	Listen-Before-Talk
MAC	Medium Access Control
ML	Machine Learning
OCM	On-Chip Memory
PHY	Physical Layer
PL	Programmable Logic
PS	Programmable System
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
RTT	Round Trip Time
SDR	Software Defined Radio
SRH	Shared Radio Head
TCP	Transmission Control Protocol

TTI	Transmission Time Interval
TDD	Time Division Duplex
TDMA	Time Division Multiple Access
UD	User Device

1 INTRODUCTION

The wireless industry is facing the challenge of how to expand its market to support multiple vertical industry applications, while meeting their conflicting traffic demands. In late 4G, it can be observed that mobile networks are already being subject to a manifold of technical and service requirements with regard to data rates, latency, reliability, ubiquity, energy-efficiency and cost-efficiency. This diversity of traffic requirements will become even larger with 5G with the introduction of novel application areas, such as factory automation, vehicular communication, and smart grids. FurtTo avoid the cost and energy inefficiencies of deploying segmented single-service networks, network slicing has been suggested by any industry and academic bodies as an appealing solution. It allows the instantiation and control of independent logical networks called Network Slices (NSs), each targeting distinct industry applications, on a single network infrastructure. A NS defines a set of configurations of a network's computational and communication resources that reach all layers of the protocol stack and multiple heterogeneous hardware components. The level of flexibility and scalability promised by slicing may only be achieved through the adoption of more programmable radio and networking platforms, such as Software-Defined Radio (SDR) and Software-defined Network (SDN).

The overall ORCA goal is to offer end-to-end SDR solutions that allow experimenters and network designers to successfully tailor their testbeds or networks to the most diverse and stringent application requirements. In order to do so, ORCA extends the current state-of-the-art SDR capabilities in the two following ways: (i) designing data-plane SDR solutions on heterogeneous hardware platforms that can combine high data rates or low latencies with fast design cycles and high versatility, (ii) providing flexible end-to-end reconfiguration facilities that enable faster and simpler control over multiple SDR nodes and to tailor NSs to target the specific traffic requirements of separate service applications. This deliverable focuses on the latter. ORCA partners will provide a brief overview of the end-to-end capabilities offered by their Year 1 SDR implementations, listing the control and monitoring facilities they add to the ORCA control-plane architecture, and how they support the slicing of a network's radio and hardware resources. The provided solutions are diverse, leveraging the different SDR platforms and computational resources available in ORCA partners' testbeds, and reaching different network elements (e.g. cloud, front-end) and layers of the protocol stack.

1.1 Organization of the Deliverable

The remaining sections of the document are organized as follows:

- Section 2 summarizes the TUD and NI's developed control, monitoring and slicing tools for an mmwave communication system
- Section 3 describes the KUL's network framework for the control of multiple SDR nodes with in-band full duplex capability
- Section 4 describes cloud based robot controller setup implemented by TUD. The robot uses the low latency GFDM PHY that enables the balancing algorithm of the robot to be process remotely. Further, the planned extension with edge cloud computing is briefly described.
- Section 5 summarizes the reconfiguration and slicing functionality implemented by IMEC to support multiple PHY and MAC schemes in a ZYNQ SDR platform
- Section 6 is dedicated to TCD's radio environment monitoring framework, encompassing the the RF signal dataset collection, and the machine learning-based model training, validation and SDR integration facilities to enable RF slicing/waveform classification
- Section 7 describes NI's ns3 based prototyping platform for RAT interworking that enables experimentation with LTE, Wi-Fi and 5G.

Each section is then subdivided into: (i) summary of the implementation results obtained during Year 1, (ii) how the implemented solution is integrated in its respective Year 1 ORCA showcase, (iii) how

the solution is integrated in the respective partner's testbed, and (iv) the extension plans currently envisioned by the partner for Year 2.

2 HIGH THROUGHPUT MMWAVE SYSTEM

The high throughput mmWave system as an outcome of the MiWaveS EU project [1] represents the baseline for further development and testbed integration within ORCA. A short introduction is given in ORCA D3.1 [2] and more a more detailed description can be found in MiWaveS Deliverable D6.5 [3]. Figure 1 shows the offered setup for a bidirectional, closed loop mmWave link. While ORCA D3.1 is focusing on the data plane, this deliverable focuses on the control plane to allow researches parametric (re-)configuration.

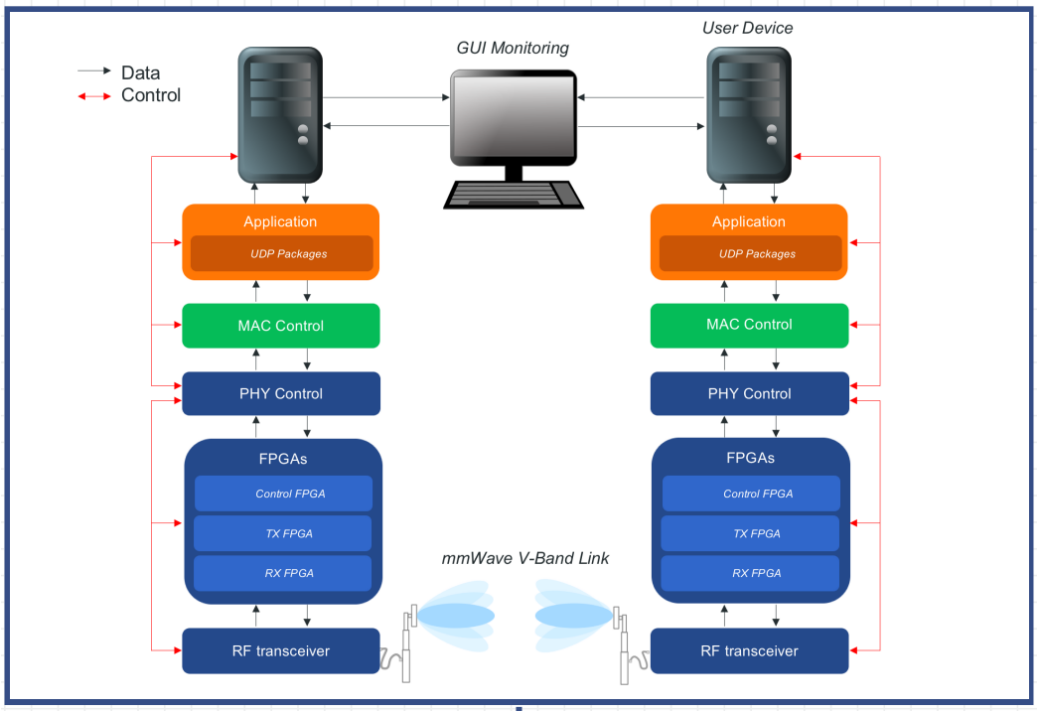

Figure 1: Bidirectional, closed loop mmWave link (V-Band)

2.1 Implementation results

The chapter will give an overview about the bidirectional, closed loop mmWave system using V-Band directive antennas suitable for small cell access. The year 1 implementation results are described in the following subsections with focus on control plane and radio slicing. The main goal was to achieve runtime SDR configuration with demonstration of parametric reconfiguration over testbed backbone [KPI9].

2.1.1 SDR control plane improvements

In order to setup an SDR control plane which includes (re)configuration and monitoring the capacity and status of the links and the network, two possibilities are available

1. Direct Control and Monitoring via graphical user interface (GUI)
2. Remote Control and Monitoring over the testbed backbone

The functionality of (1.) was mainly developed in the MiWaveS EU project [1]. The main Year 1 achievement is (2.) where the interface functionality (1.) was embedded into a remote-control application by NI. This remote-control application enables researchers to modify and monitor PHY, MAC and system parameters during runtime as required for the control plane.

2.1.1.1 Direct Control and Monitoring

This section summarizes the main graphical user interfaces of the mmWave system in order to have direct access to control and monitoring parameters. The main user interface of the access point, used for MAC level demonstrations, is shown below in Figure 2. The user interface provides access to scheduling settings, selection and different beamsteering algorithms. Likewise, it allows monitoring connection states and beamsteering related measures.

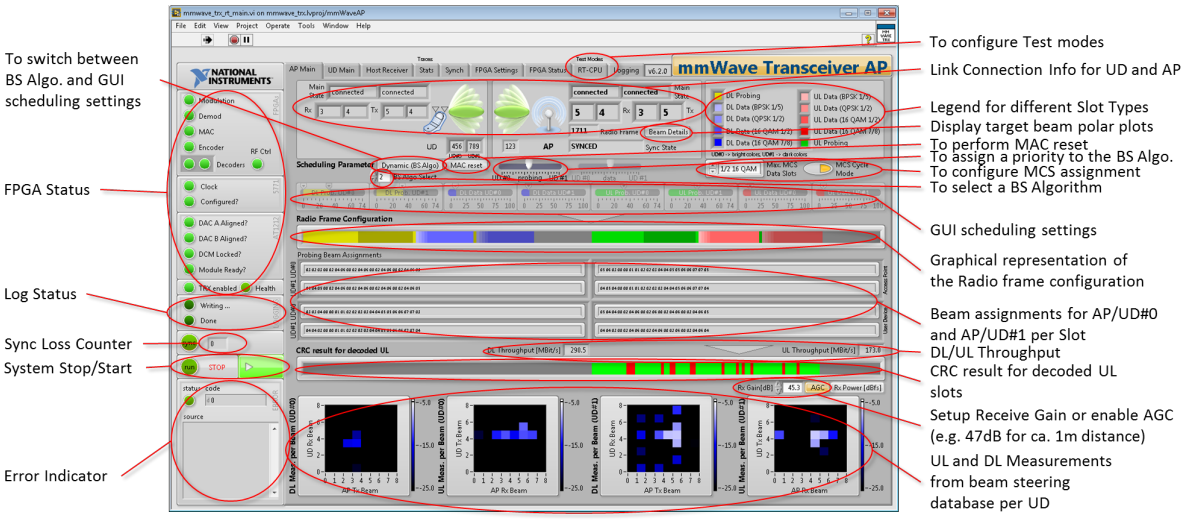

Figure 2: Main access point front panel for MAC level demonstrations.

The beam settings used to transmit data are illustrated in polar plots as shown below in Figure 3.

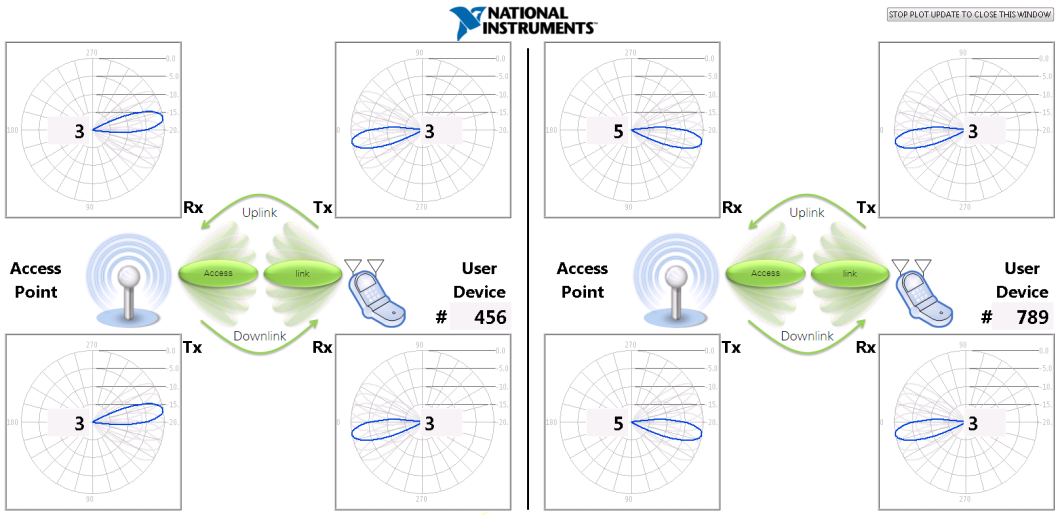

Figure 3: Graphical representation of the current target beam settings at AP and UD for two users.

Several traces are collected synchronously in radio frame intervals, such as automatic gain control (AGC) parameters, throughput, connection states and beam settings. These traces allow to investigate the system behaviour over time, e.g., under the impact of mobility. Furthermore, the traces can be stored in a buffer for offline evaluation. Figure 4 illustrates an example trace showing MAC parameters such as connection states and the beam settings of the connection to two user devices over a duration of 10 seconds.

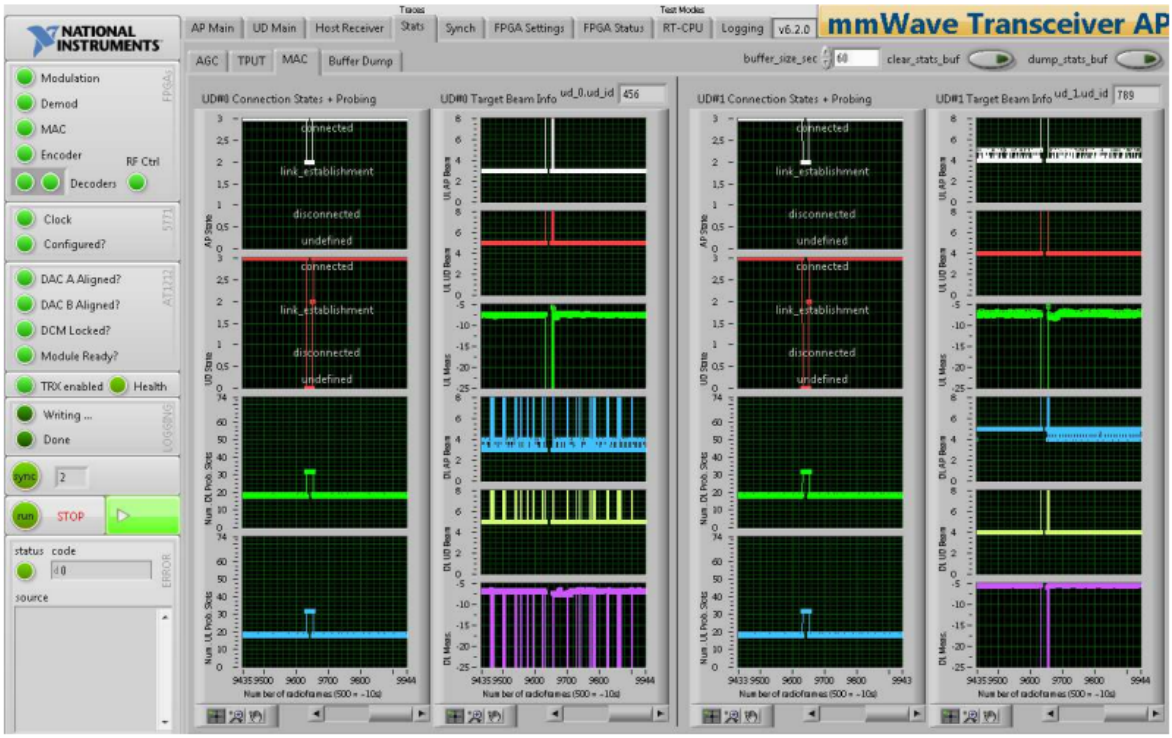

Figure 4: MAC Traces collected at the access point.

Figure 5 illustrates an example trace showing the AGC parameters such as Rx Gain, Applied Gain Step and Maximum Rx power over a duration of 10 seconds during mobility experiments.

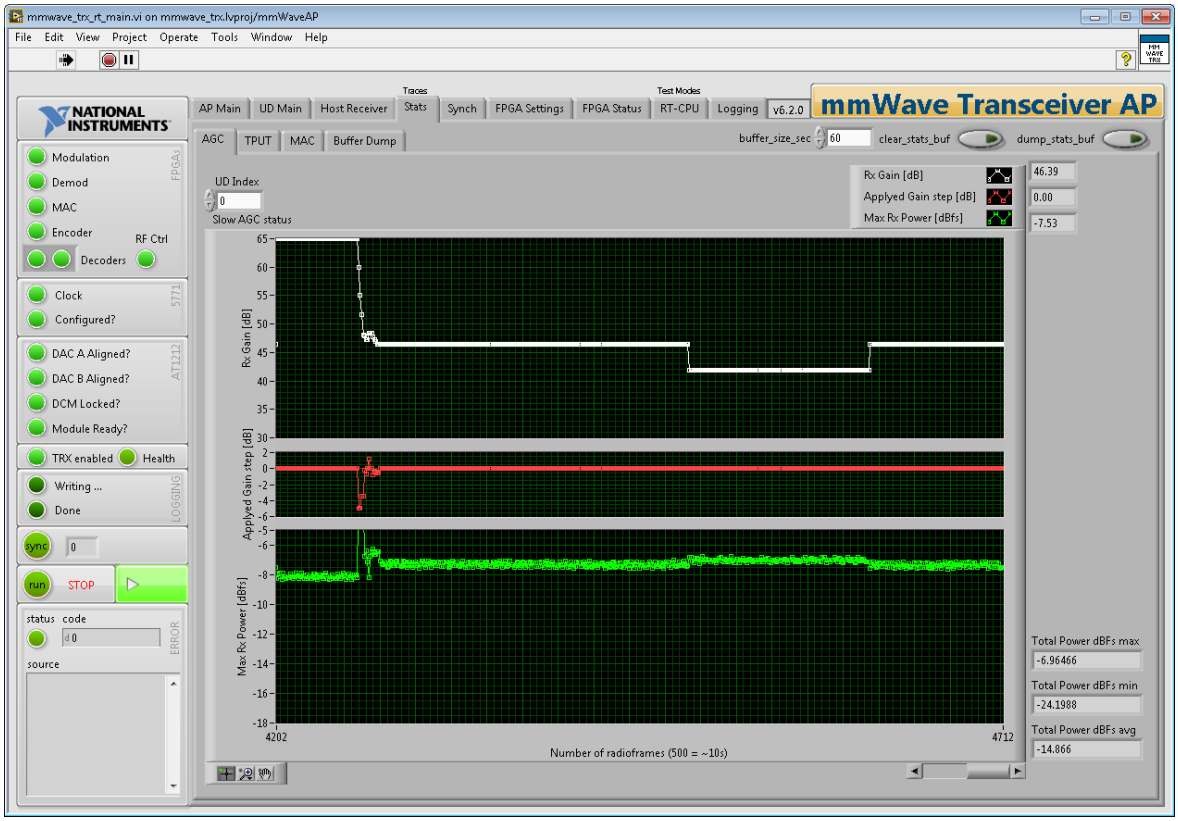

Figure 5: AGC Traces collected at the access point.

2.1.1.2 Remote Control and Monitoring

In order to control and monitor the mmWave SDR in an automated fashion from the testbed itself or a higher layer control application, the graphical control and monitoring interfaces from the previous section will be accessible by using the TestMan plugin (Figure 6) introduced in the EU-project CREW. Therefore, NI developed a glue layer between the proprietary device and the TestMan software (Figure 7). TestMan which was developed by TUD, provides a common interface to exchange data, commands and status messages between different application across a network. With this, script controlled configuration and monitoring are possible. The main concept is illustrated in Figure 6.

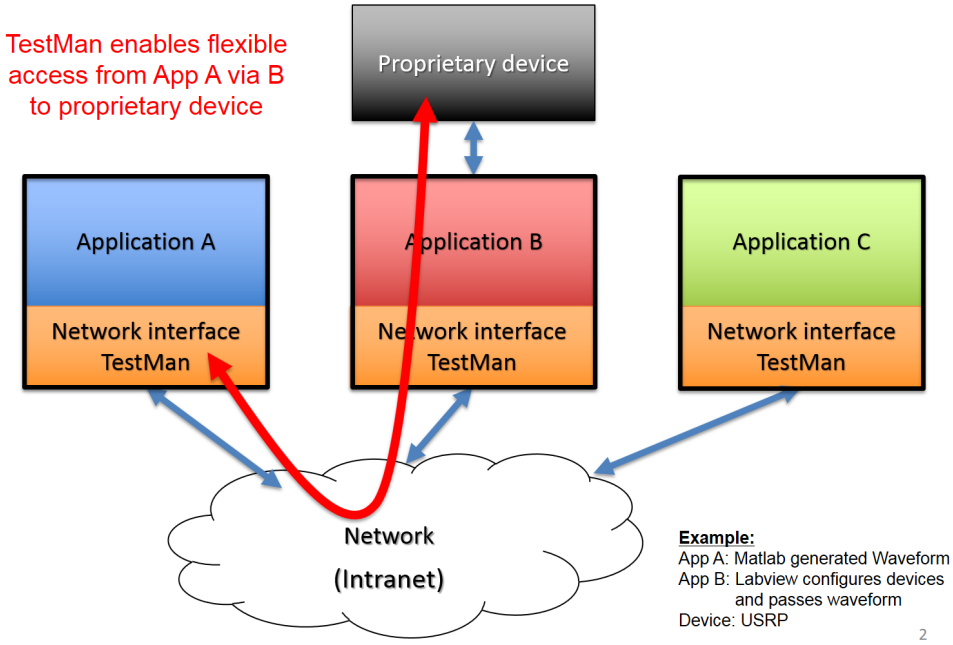

Figure 6: TestMan Concept

The main achievement of NI was to develop a remote access architecture for the mmWave system using and integrating TestMan application and its interface. Figure 7 gives an overview about the remote access architecture used for the mmWave System. Besides TestMan plugin also components such as CtrlCmd Handler are implemented in order to transfer a control parameter from the testbed to a Host Control PC and finally to the mmWave real-time (RT) target.

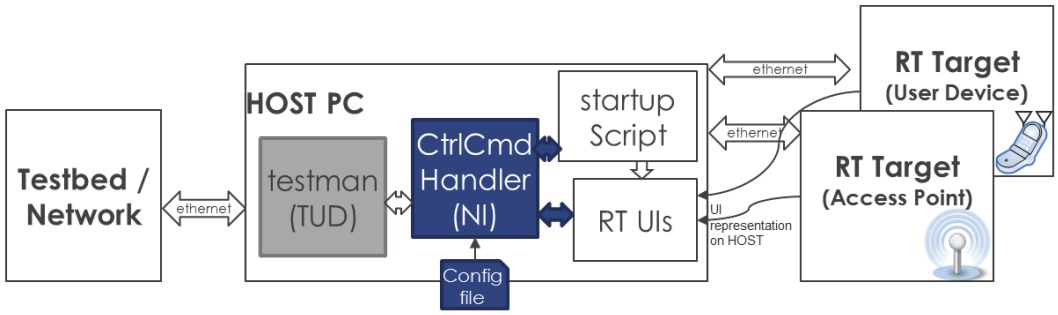

Figure 7: Remote access architecture for the mmWave system using TestMan

Through a configuration file which is consumed by the CtrlCmd Handler the accessible parameters of the Direct Control and Monitoring Interface (2.1.1.1) are defined. With this the remote access architecture is very flexible and can be easily adapted to different scenarios and use cases without changing the code. At first this remote access functionality will be integrated into the Channel characterization environment via HALO (Hardware in the loop), see ORCA D2.2 [4]and D2.3 [5].

2.1.2 Radio slicing: resource allocation, instantiation, and coordination

The mmWave system which was mainly developed in the MiWaveS EU project [1] will be used and extended within ORCA to provide the experimenter capabilities for radio slicing. As a result, for Year 1 the key components and parameters are identified to allow radio slicing and described in the following. For the planned extensions see section 2.3.2.

The single carrier system operating at 750 MHz symbol rate with real time signal processing, supports variable rate transmission using BPSK, 4 QAM and 16 QAM modulation and variable forward error correction code rates. Data rates can be varied between 147 Mbit/s and 2318 Mbit/s. It employs single carrier modulation in TDD mode. The signal structure is divided into radio frames, slots and blocks as shown in Figure 8.

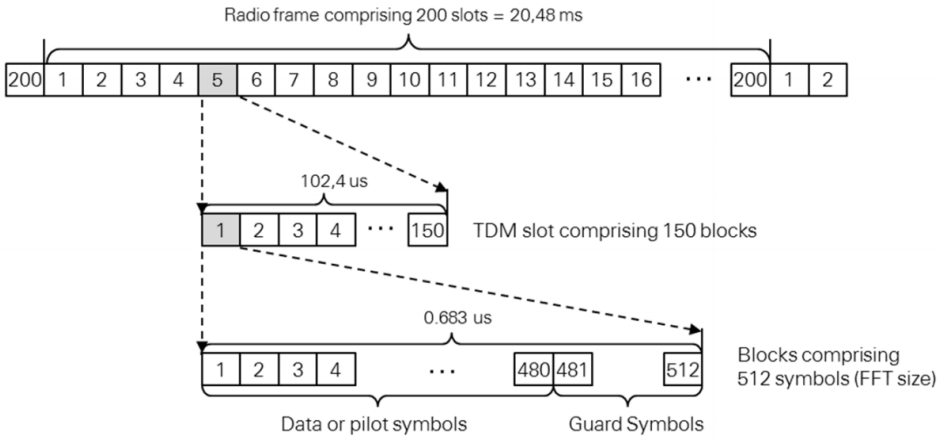

Figure 8: Radio frame structure

Most importantly, a slot plays a similar role as a 3GPP LTE transmission time interval (TTI). However, its duration is 1/10 of the LTE TTI duration (0.1024 µs MiWaveS versus 1 ms in LTE). Shortening the TTI duration by a factor of 10 is a major prerequisite to develop a low latency communication system. Note that a slot is further divided in blocks. Each block carries a guard interval which enables block-wise frequency domain equalization in time-dispersive channels.

Using the real-time mmWave baseband setup, physical parameters can be controlled and (re-)configured on slot level at a time base of 0.1024 us. This is achieved by the MAC layer which is executed on a real-time operating system (LabVIEW RT [6]). Therefore, a radio slice in the context of this mmWave demonstrator can be defined with the following parameters

- Direction: Uplink or Downlink
- Robustness/Throughput: Modulation Coding Scheme (MCS)
- Time: Slot and Radioframe index
- Spatial: Beam index (refers to a code book entry, the code book defines the beam shape and direction)
- Logical: User Identifier (in case of a Multi-User setup, not available in ORCA)

In principle these parameters can be varied manually (e.g. using the HALO setup) or fully automatically using the real-time mmWave baseband setup. With the second variant the current software infrastructure provides two stages of parameter assignments at the mmWave Access Point (AP) which behaves as master, see Figure 9.

1. **Beam steering algorithm** decides about the spatial component and assigns the beam index based on channel measurements
2. **Scheduler algorithm** decides about direction, robustness / throughput, logical and time

assignment and outputs a radio frame table with all relevant PHY parameters

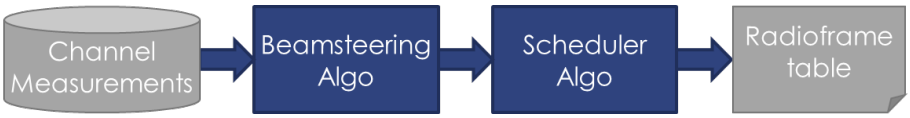

Figure 9: Two stages of beam steering parameter assignments

The actual status of the implementation for Year 1 is, that all the required software infrastructure components such as synchronization, channel measurements, automatic gain control and MAC protocol are available as a result of the MiWaveS EU project [1]. As a result, for Year 1 the key components and parameters are identified to allow radio slicing. Run-time re-configuration of a single slice parameter such as MCS was achieved using the remote control and monitoring tools described in 2.1.1.2. Beam steering and scheduler algorithms for automated slice control are topic for further research. Please see section Implementation Plan (2.3) for more details.

The emulation of manual beam steering functionality is firstly realized using the hardware in the loop (HALO) setup (see ORCA deliverable D3.1 [2]) in combination with the remote access architecture for mmWave system described in previous section.

2.1.3 Testbed integration

The described mmWave system will become available at the TUD OWL testbed

1. For the first Open Call for experimentation starting in ORCA Year 2 the focus is on channel characterization environment via HALO (Hardware in the loop) with real time radio control described in ORCA D3.1 [2], allowing to capture datasets for offline evaluation.
2. In a second step the bidirectional, closed loop mmWave system will be established

These features will be integrated to the testbed allowing access to external users.

2.2 Relation to showcase

As described in ORCA deliverable D2.1 [7], the showcase for the high throughput mmWave system is showcase 1: “High throughput - mmWave”.

2.3 Implementation plan

This section describes the plan for the functionality to be made available through the control interfaces described in section 2.1.1 in the upcoming year, focusing on ORCA KPIs

- mmWave operation - use real time steerable antennas
- bi-directional TDD protocol enabling beam tracking

2.3.1 SDR control plane improvements

For the ORCA Year 2 it is planned to improve the SDR control plane by supporting following offers which are described in more detail in ORCA D2.2 [4].

- runtime reconfiguration of beam steering algorithms
- configuration of automatic and manual beam steering functionality at MAC layer

To achieve this, it is planned to provide the necessary control parameters to the front panel interface which runs on the NI real-time PXI controller [8]. Further, the front panel will be interfaced with the Remote Control and Monitoring module that was developed in ORCA Year 1 and described in section 2.1.1.

2.3.2 Radio slicing: resource allocation, instantiation, and coordination

As outlined in Section 2.1.2 the beam steering and scheduler algorithms are the key for allowing control and (re-)configuration of the radio slice. As already described, parameters such as Direction, MCS, Slot index, Radioframe index, Beam index and User Identifier will define a radio slice for this mmWave demonstrator.

Therefore, different beam steering algorithms and scheduler algorithms can be developed and possibly switched during runtime using the control plane improvement described in 2.3.1. Further, the development of new advanced beam steering algorithms and scheduler algorithms can be seen as possible topic for the second open call for extension. The implementation is planned to focus on development of stable interfaces for the beam steering algorithm as well as for the scheduler. A beamsteering simulation environment, where the relevant modules can be tested without the need of hardware, would allow rapid prototyping of described functionalities. Figure 10 shows the principle concept.

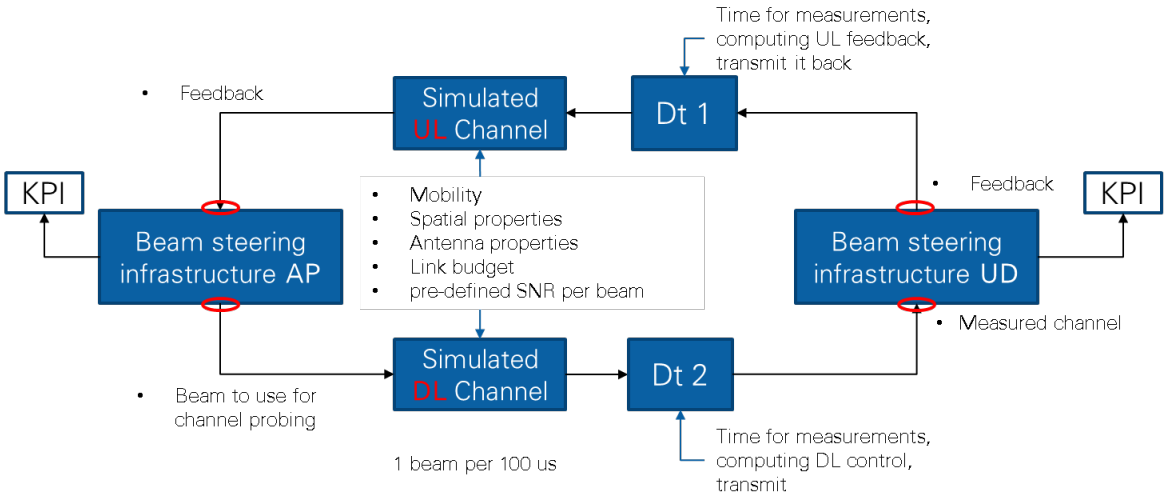

Figure 10: NI Beamsteering Simulation Environment

The beam steering infrastructure AP (access point) covers the execution of the beam steering algorithm which handles all probing slots and the execution of the scheduler which handles all non-probing slots used for data transmission. The simulated UL/DL channel and the Dt 1/2 blocks models UL/DL channel and system behaviour. The beam steering infrastructure UD models the beam steering relevant parts for the user device such as DL channel measurements and UL feedback provision. More details will be described in the upcoming deliverables depending on the relevance.

2.3.3 Testbed integration

The mmWave hardware will support either real-time mmWave basedband setup or HALO. This can be utilised by the experimenters to conduct wither channel characterization experiments for evaluating algorithms offline and later on validating this algorithms with real-time setup.

3 A NETWORK OF SDRS FOR IN-BAND FULL DUPLEX WITH COLLISION DETECTION

3.1 Implementation results

This section presents the implementation results on IBFD network achieved in ORCA Year 1. Figure 11 illustrates the high-level diagram of the IBFD network which enables various end-to-end networking experiments. As shown in this figure, four FD capable nodes are considered in this network to share the spectrum using unslotted CSMA/CD protocol. Furthermore, an extra node plays the role of an external interferer. In the next section we explain how a PC-based control plane enables a wide range of experiments by using the testbed.

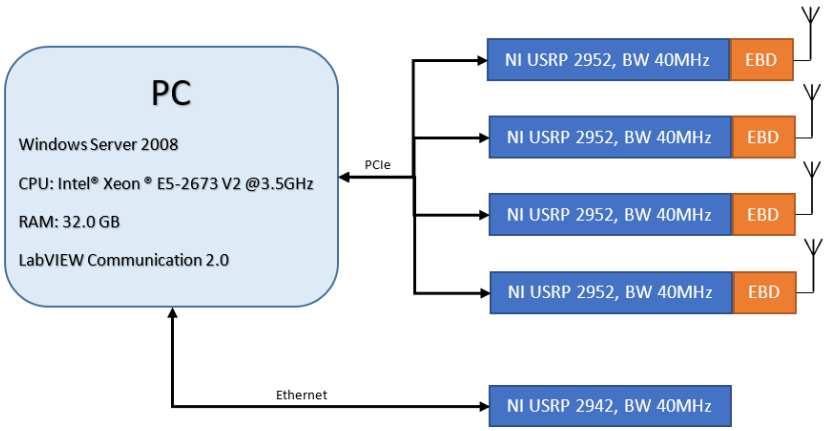

Figure 11: High-level diagram of the full duplex network testbed

3.1.1 PC-based SDN control plane

To control the network, a host interface in Figure 12 is implemented in LabVIEW Communication which enables various test scenarios. This interface allocates address to each one of the nodes and can control their PHY and MAC configurations separately. For example, an experimenter can run 2 nodes with collision detection capability and two others without it and assess their performance separately. It also monitors various signals and performs packet delivery rate measurement.

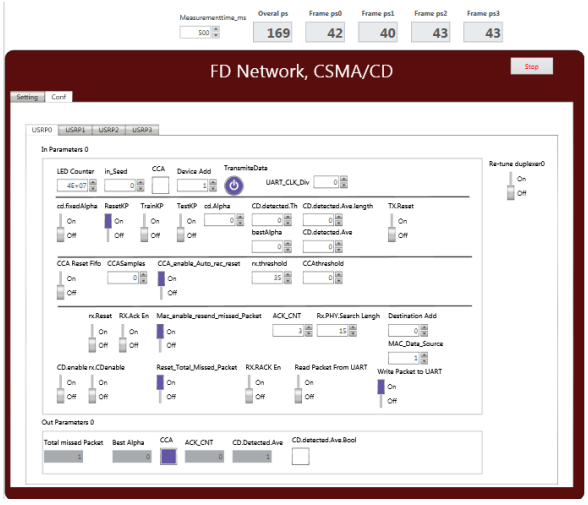

Figure 12: Host interface as the control plane

The control plane can transmit data from three different sources listed in Table 1 and the source is configurable independently for each node.

Table 1: Accessible transmission data sources in the control plane

1	single packet transmission from host interface
2	sequential packet transmission from high-level MAC
3	sequential packet transmission from UART

Figure 13 illustrates the control plane flowchart. The experimenter can select the number of nodes in the test and as shown in this figure, the host iterates over the participating nodes to set the parameters and read the measurement results.

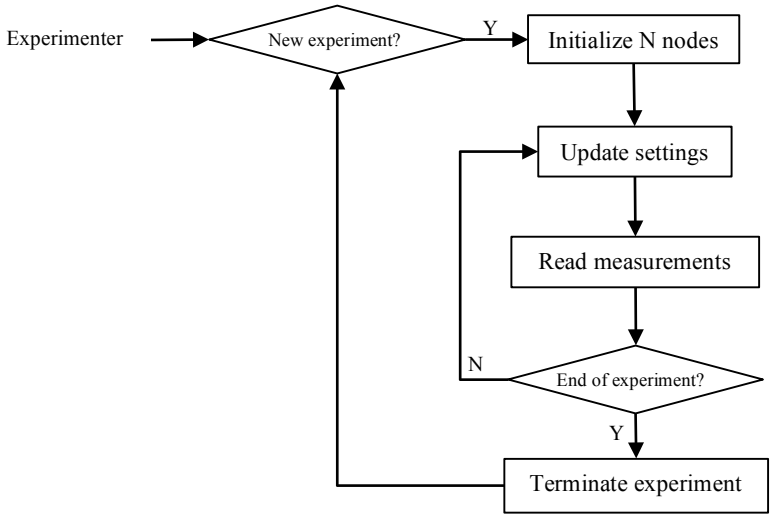

Figure 13: High-level control plane flowchart

Using the host interface, we could measure end-to-end latency of 1.5 - 3ms for the network. This time includes PIC-e interface delay, CSMA backoff period and wave travel time as well as PHY data latency at both transmitter and receiver sides.

3.1.2 Testbed integration

The control plane is already integrated to the testbed allowing full control over the hardware and enables experimenting IEEE 802.15.4 network base on unslotted CSMA/CD.

3.2 Relation to showcase

This setup can be employed for ORCA SC2 as it provides all required facilities for this showcase. In ORCA SC2, 2-3 nodes are considered to be connected to robots, getting sensory data through UART link and pass it to the control node. The control node can send commands to either of the robots. This host interface configures the architecture and routes the UART interface to the PHY and MAC modules enabling robots to send and receive data through the air.

3.3 Risk analysis

Risk analysis whether the end-to-end latency can be measured in details for all the layers accurately. One solution is to perform a part of the measurements inside the FPGA code and the other part by the control plane on the host.

3.4 Implementation plan

For next year, we would like to continue with the development of the control plane in various aspects. Controlling a higher number of nodes, more accurate and detailed end-to-end latency measurement and continued packet transmission from the host are among the targeted developments. We are also planning to update the data plane according to the upcoming extensions in PHY-MAC architecture.

3.4.1 Testbed integration

All the upcoming developments in Year 2 will be updated to the testbed. It means that the planned developments in D3.1, including new configurations in PHY and MAC layers will be accessible in the testbed and experimenters can select the network parameters according to their test scenario.

4 FLEXIBLE PHYSICAL LAYER BASED ON GFDM

4.1 Implementation results

The current implementation enables PHY experiments with a minimal set of MAC functionalities. Figure 14 depicts the status and evaluation setup for the robot case. Here the task is to sense the environment of the robot, in this case the angle of the robot and transmit the measured data to a cloud processor. Inside the edge cloud, the data has to be processed quickly in order to send the right motor control commands back to the robot in time. Finally, the robot has to keep its balance. A second end-to-end application has been demonstrated, using a setup as depicted in Figure 15, where multiple services using different waveform configurations are sharing one transceiver. In particular, the access point has to serve a low latency user as well as a Bitpipe user, which wants to stream video signals. The PHY implementation is adjusted in real-time to apply an optimal waveform configuration for each services.

In both cases the Generalized Frequency Division Multiplexing (GFDM) waveform is employed, which is very suitable for applications under unlicensed bands where several technologies have to share the medium. GFDM provides a flexible framework that is able to generate and receive commonly used waveforms, such as OFDM (LTE, WLAN), SC-FDE (LTE uplink) [9]. In the upcoming year, the focus of the development is to introduce higher network layers up to a powerful edge cloud computing platform.

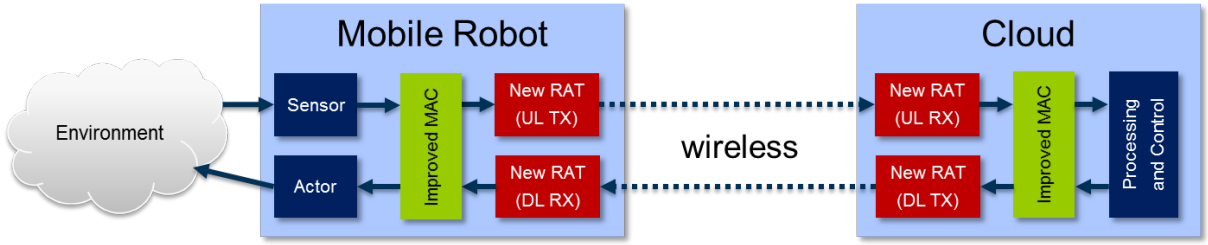

Figure 14: Current evaluation setup for the transceiver

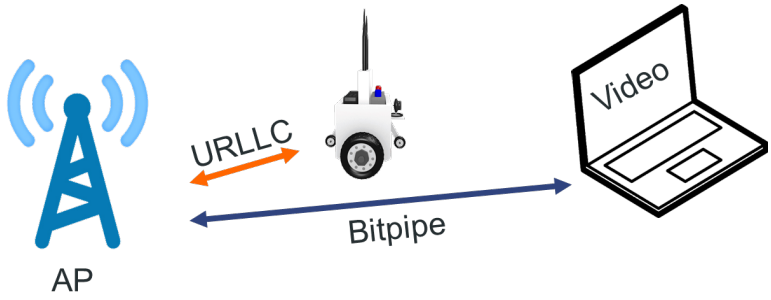

Figure 15: Flexible access point offers various services using a reconfigurable and flexible physical layer transceiver

4.1.1 Testbed integration

The implemented FPGA transceiver is part of the TUD testbed and available via the eWINE software repository as open source project. The used physical layer implementation is described in the EU-project eWINE deliverable D4.1 and in the ORCA deliverable D3.1. Besides the transceiver software, the robot application itself will not be supported by the testbed as a fixed installation due to its mechanical characteristics, but it can be added on request manually for specific experiments.

4.2 Relation to showcase

The presented platform is mainly related to showcase 2 as its focus is on industrial applications. However, also showcase 4 is relevant, especially because in industrial environments multiple wireless networks have to coexist. This circumstance will heavily influence the performance of the network.

4.3 Risk analysis

There is no specific risk observed at this phase of the work package, from a software perspective. The robot itself is sensitive due to mechanical stress and therefore not part of the testbed.

4.4 Implementation plan

The proposed transceiver will be integrated into powerful host processing platforms in two steps. At the TUD Testbed two different devices are available, the NI PXI chassis with a NI 8880 controller (Intel Core i7, 24 GB RAM), and a Hewlett Packard Enterprise (HPE) Edgeline 4000 chassis with four server modules (16 core Intel Xeon, 128 GB RAM) each. Especially the last platform will be interesting for industrial applications as it can host a mobile edge cloud attached to the SDR device.

Figure 16 gives an overview of the proposed system, where different functionalities are spread over different servers. The SDR devices with transceiver implementation is attached via PXIe to the MAC-PHY server, where various software-MACs are instantiated on request. The PXIe connection ensures are low-latency and high-bandwidth data exchange between the SDR and the MACs. These software MACs represent the different network slices offering IoT, Bitpipe or URLLC services, which are sharing one PHY layer implementation and the respective “air-time” foreseen for the service. A Linux real-time OS realizes timely and reliable execution of the MAC functions. In the first step, the NI 8880 controller is used to host the MAC-PHY server to gain a first understanding of the performance of the system.

In the second step, other OS will be hosted on the remaining server modules within the HPE Edgeline chassis. These will facilitate the mobile edge cloud to provide high computing capabilities attached to the access point. Applications requiring strict timings will therefore run close to the wireless network, whereas non-timing critical applications can be offloaded to other cloud platforms in the network or in the internet.

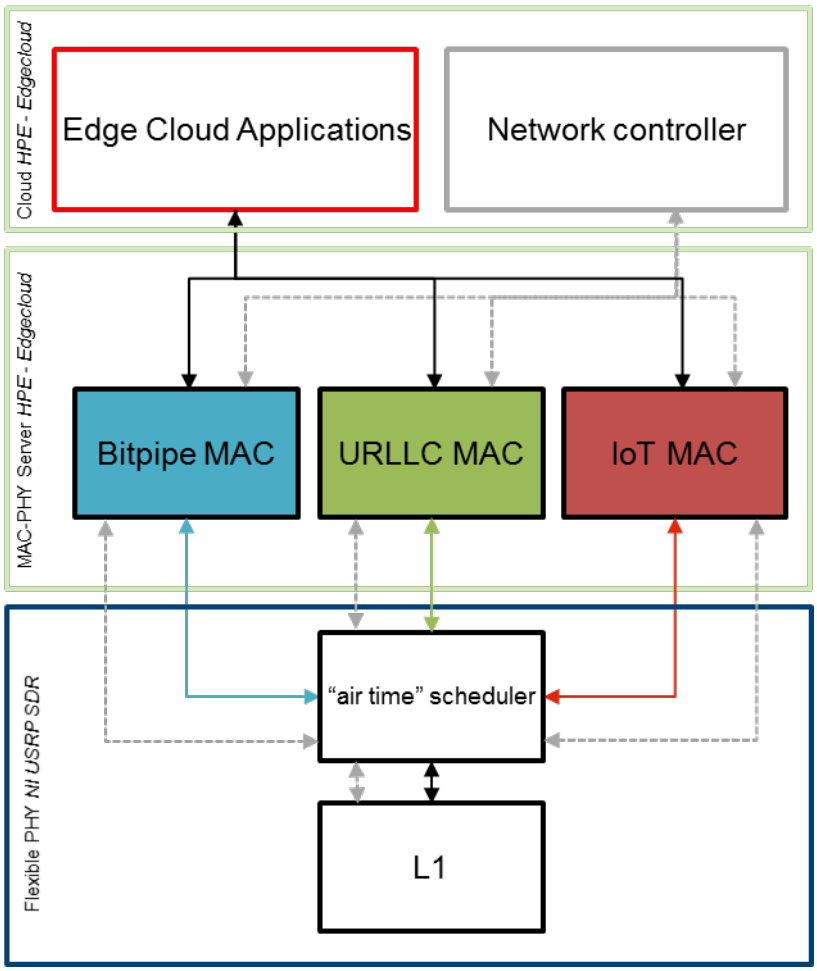

Figure 16: System overview: The data path is marked with a continuous line, the control path with a dashed line. The different MACs can be instantiated in software and then program the PHY according to the requirements.

4.4.1 Testbed integration

The described systems will be made available as part of the TUD testbed. At least one base station will be equipped with a HPE edgeline server and an attached USRP-RIO SDR platform. The user terminals are based on a USRP-RIO SDR with a control PC each and an additional access point will have a NI PXI based chassis. All platforms support the mentioned PHY and higher layer capabilities, only the processing speed will vary. Using the proposed setup, the TUD testbed will enable full stack experimentation, starting from the edge cloud down to a very flexible physical layer supporting a wide range of waveforms. Further, the very fast reconfiguration times allow adapting not only the MCS but also the whole waveform depending on the wireless situation.

5 HYBRID FPGA PLATFORM INCL. FLEXIBLE MAC

5.1 Implementation results

This section describes the Year 1 implementation results on the ZYNQ SDR platform, for more details of this platform and corresponding data plane design, please refer to Section 5 of D3.1. The high level diagram of the design is repeated below to clarify further the description. In general, several hardware accelerators are available that supports run time configuration from the ARM processor.

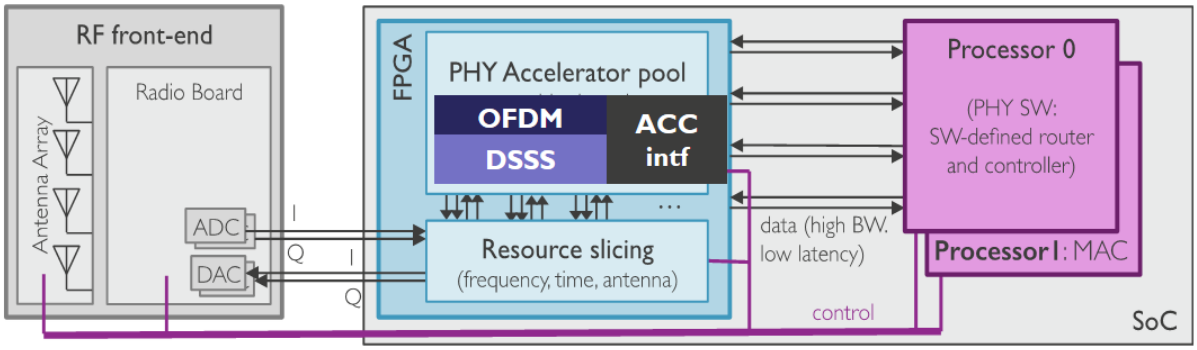

Figure 17: General architecture of the hybrid SDR on ZYNQ

5.1.1 SDR control plane improvements

A 'zoomed in' view of the data and control path of an individual accelerator is shown in Figure 18. For each of the accelerator, control and data path are exposed through the AXI-lite and AXI stream bus interfaces respectively. More specifically, AXI-LITE controls the reading and writing operations of

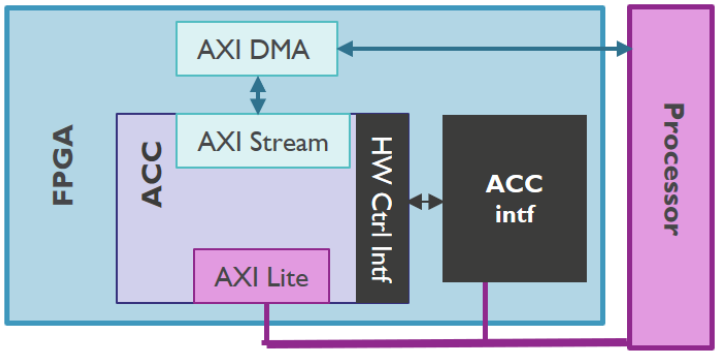

Figure 18: Zoom in view of the data and control path of an accelerator

registers, which controls various parameters in the accelerator. A few parameters of the OFDM receiver accelerator is listed below as an illustration:

Table 2: Example register space of OFDM Rx Accelerator

Register name	Functionality
pkt_det_autocoor_config	Configures the auto correlation of preamble, eg correlation length, threshold, etc.
lts_corr_thresh	Configures the cross correlation threshold with the Long Training sequence

rx_fft_config	Configures various settings of FFT, eg number of subcarriers, cyclic prefix length, the offset of first sample location in each OFDM symbol, etc
rx_chan_est_smoothing	Configures the coefficients of the filter to smooth channel estimation between adjacent subcarriers.

In addition, a customized hardware interface is designed to support additional control signals from the 'ACC intf' block, that could be regularly occurring, eg clock enable signals as mentioned in D3.1 for controlling the operation speed of OFDM accelerator. For certain accelerators that do not have an AXI Stream interface, the data path towards ARM processor could be achieved via the ACC Intf block, which translates normal FIFO like interface into AXI Stream.

The choice of AXI Stream and AXI LITE is made based on solid investigation of their throughput and latency performance. On the ZYNQ SoC, the AXI Stream throughput could easily reach Gbps, though the exact throughput depends on the bus width and the clock speed of each design [10]. A simple measurement is conducted to measure the latency performance with the ZC706 Evaluation. A dummy FPGA acceleration block is designed by Xilinx HLS (High Level Synthesis) tool. The block receives data from PS in streaming manner via Xilinx DMA (Direct Memory Access) module, and stream the processed result (same amount of data as input) back to PS via DMA. It takes 654 clock cycles for this FPGA block to process 128 input samples, each sample is represented as a 32-bit word on the 32-bit AXI stream bus. The configuration interface between PS, PL and DMA controller is AXI_LITE, which is connected to M_AXI_GP port of ARM. Data link is AXI stream, which is connected to S_AXI_HP port of ARM. A DMA controller was used to convert AXI Memory Mapped interface (needed by PS) to AXI Stream interface (needed by streaming mode FPGA accelerator). Clock speeds are as follows: AXI buses and PL run at 200MHz, and ARM cortex-A9 processor run at 800MHz. Xilinx ILA (Integrated Logic Analyzer) is inserted to the design for event recording with 5ns resolution. The ARM software event is recorded by writing special value to PL register and detecting this value via ILA. The latency profiling result is shown in Figure 19.

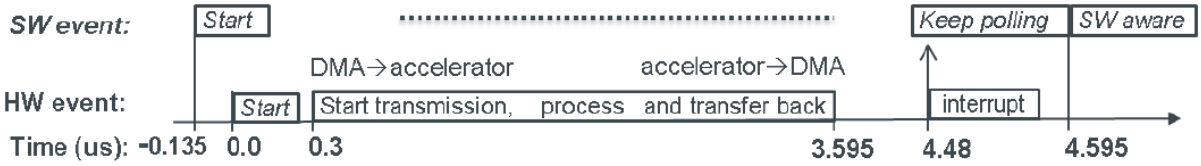

Figure 19: Latency test result of ZYNQ-7000 SoC

We observe the event log of the entire round trip delay test captured by Xilinx ILA. First, the software start DMA transmission at -0.135us; then DMA controller receives the instruction at 0us via AXI LITE register interface; After some internal preparation and buffering operation, the actual DMA transmission on the AXIS bus starts at 0.3us; After 3.295us, which is caused by the accelerator processing latency of 654 clocks at 200MHz, the accelerator completes the data transfer of processing results back to DMA controller in streaming manner; Then DMA controller raises interrupt to PS at 4.48us; Finally, software becomes aware of this event at 4.595µs. So the round trip latency between the FPGA accelerator and software in PS is 4.595+0.135-(3.595-0.3) = 1.435µs. Note that this is superior performance comparing to the latency of USRP variants, and it is even not a significant overhead compared to the Wi-Fi SIFS requirement (16us or 10us). More on the latency measurement is available in [11].

After verify the latency performance of local control loop on a single ZYNQ SDR, an experiment is performed to examine the round trip time of a communication link between two ZYNQ SDR board. The measurement simply uses the concept of 'ping', i.e., one SDR generates a packet and the other replies as soon as the packet is received. The measurement setup is illustrated Figure 20. In order to clearly describe the measurement, the PS of each SDR is drawn twice, once for reception and once for transmission module, similarly the DSSS (ZigBee) accelerator is also shown twice, when it is working in Tx and Rx mode respectively.

At the start (noted as 't0'), SDR1's PS initiates a SW event to DMA to store data for transmission, at time instance 't1', the DMA action is complete, and the accelerator starts to generate IQ samples towards the DAC. At time instance 't2', the packet transmission is complete, SDR2 receives packet, triggers its PS and transmits the reply packet. At time instance 't3', the ZigBee accelerator in SDR1 receives the complete packet, and trigger the DMA transfer towards PS. Finally, at time instance 't4', PS of SDR1

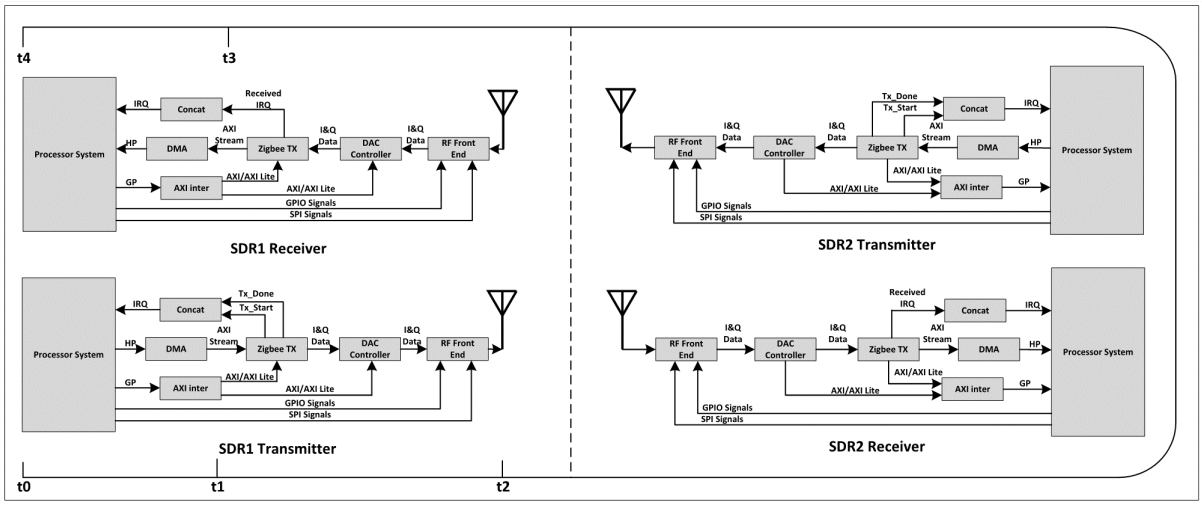

Figure 20: The round trip time between two ZYNQ SDR's measured by 'ping' over ZigBee (DSSS) link.

receives the ping reply from SDR2. The time consumed between each of the aforementioned time instances are shown in Table 3. The measurement is repeated with different packet sizes. It is worth mentioning that the overhead, which is calculated with the measured elapsed time substracted by the expected time (the transmit and reception time when packet is in the air), is independent from the packet size, i.e around 1.9ms.

Table 3: RTT measurement with different packet size

PacketSize	Sending Time (ms)		Receiving Time + Waiting Time (ms)		Elapsed Time (ms)		
	PS-DMA (t1 - t0)	Zigbee Tx - RF front end(t2 - t1)	RF front end - Zigbee Rx (t3 - t2)	DMA - PS (t4 - t3)	Measured	Expected	Overhead (ms)
125	0.323	4.5	4.98	0.647	10.45	8.512	1.938
96	0.323	3.59	4.05	0.644	8.607	6.656	1.951
66	0.321	2.64	3.08	0.644	6.685	4.736	1.949
48	0.321	2.06	2.51	0.643	5.534	3.584	1.95
37	0.322	1.71	2.15	0.644	4.826	2.88	1.946
29	0.323	1.45	1.9	0.644	4.317	2.368	1.949
20	0.321	1.16	1.61	0.646	3.737	1.792	1.945

5.1.2 Radio slicing: resource allocation, instantiation, and coordination

5.1.2.1 Host PC based radio slicing

The radio slicing is demonstrated first on host-based SDR platform, where multiple Virtual eNB (VeNBs) are sharing a single USRP as the shared radio head. The overall architecture of the virtual eNB environment is depicted in Figure 22. The system consists of three modules: eNB, mux, and radio front-end. The first module, represented in the left-most part of Figure 22, consists of a set of virtual eNBs (VeNBs). Each VeNB, in turn, comprises the LTE eNB software itself, a virtualization environment, and a guest operating system running on commodity x86 servers. Regarding software, we use srsLTE [12], a highly modular opensource LTE library which is relatively simple to modify and use. In addition, we choose KVM as our virtualization platform and use Linux guests to house the virtualized base station software. In the right-most part of the figure, we represent the actual radio front-ends or shared radio heads (SRH). To this aim, we employ an USRP B210, commonly use for software defined radio [10]. This board provices 56 MHz of real-time bandwidth, a programmable Spartan6 FPGA, and fast SuperSpeed USB 3.0 for connectivity with the eNB software. For tests purposes, we employ a set of additional USRP boards and servers deploying the LTE UE software counterpart that allow us to connect to each of the virtual eNBs. In between, we implement and deploy a mechanism to multiplex signals from the multiple virtual base stations onto the shared radio head (SRH) for transmission (Tx) as well as the ability to split the incoming signal back to the corresponding eNBs, i.e. reception (Rx). To this end, we implement a frequency multiplexing IQ switch that receives IQ samples (digitized radio signals) from the VeNBs and shifts those to different frequency locations in a wider bandwidth. The merged signal, which has higher sampling rate and wider bandwidth than those of the individual VeNBs, is sent to the SRH. To avoid overlapping or interference between the VeNBs, we use a DUC (Digital Up Converter) composed of an upsampling filter and a frequency shifting module. In order to comply with the Nyquist theorem and achieve efficient computation, we set the sampling rate of the final signal as the least common multiplier from the baseband sampling rate of the different VeNBs (e.g., 30 if three VeNBs have sampling rate 3 MHz, 5 MHz and 10 MHz). Finally, it is worth pointing out that the IQ switch/demux is implemented using GNU Radio [13].

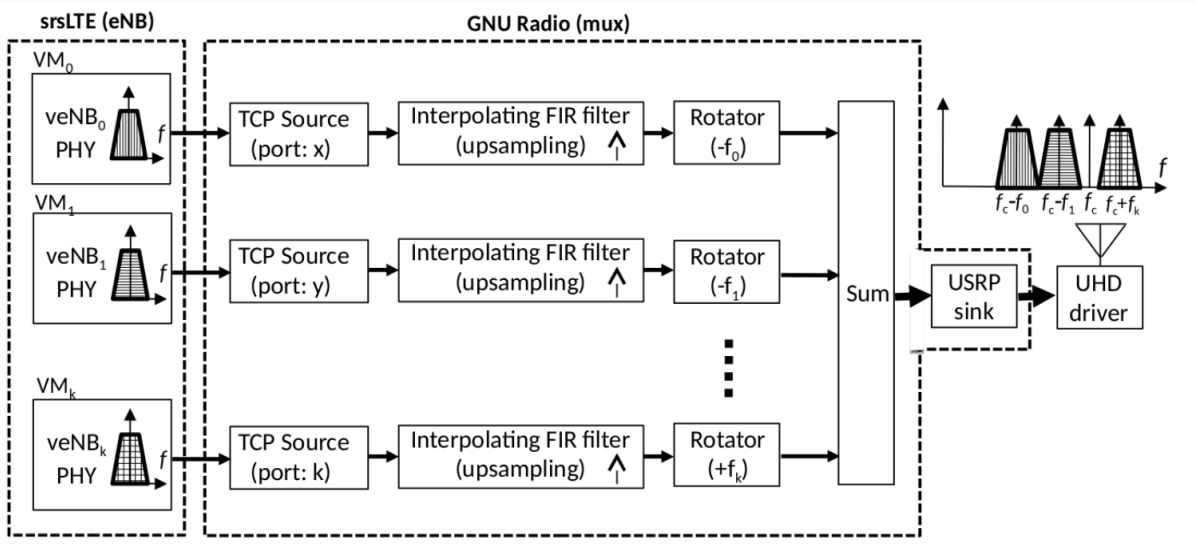

Figure 22: High level block diagram of virtual eNB

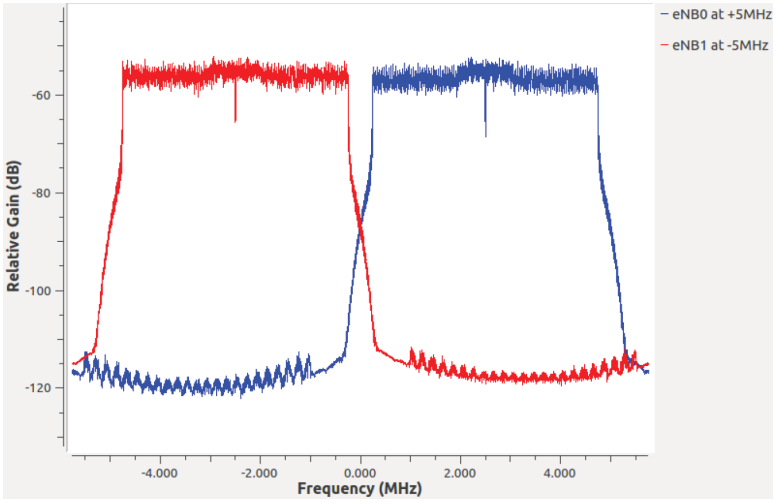

Figure 21: The spectrum of the baseband signal from two 5 MHz VeNBs after DUC and rotator.

The validation consists of two stages, first verify the operation of the IQ switch, and then an end-to-end performance evaluation is conducted. For IQ switch, it is validated with 2 VeNBs operating at 5 MHz baseband sample rate, with effective bandwidth of 4.5 MHz. The spectrum of the baseband signal from two 5 MHz VeNBs after DUC and rotator is shown in Figure 21. According to the figure, the interference level to adjacent channel is about 60 dB lower than the signal in that channel. This is a very good isolation at the transmitter side, since it causes negligible signal to interference and noise ratio (SINR) degradation. The FIR filter is designed in such a way that it offers neglectable interference for both VeNBs. For more details on the host based radio slicing please refer to [14].

5.1.2.2 Radio slicing on ZYNQ SDR

The radio slicing, more specifically frequency slicing is achieved by similar concept in the year 1 of ORCA. The difference is that role of IQ generator is now played by various of hardware accelerators, and the DDC/DUC banks, (de)multiplexers are now implemented also on the FPGA. A ‘zoomed in’ view of frequency slicing on the ZYNQ platform is given in the figure below.

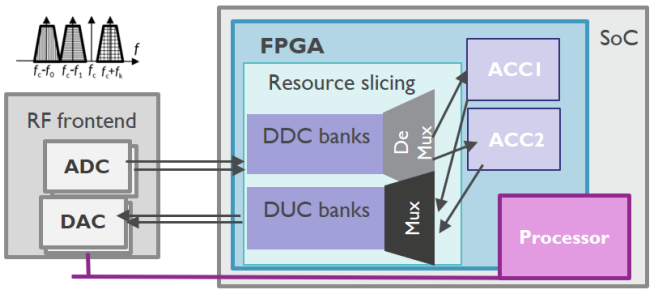

Figure 23: Frequency slicing on ZYNQ SDR platform.

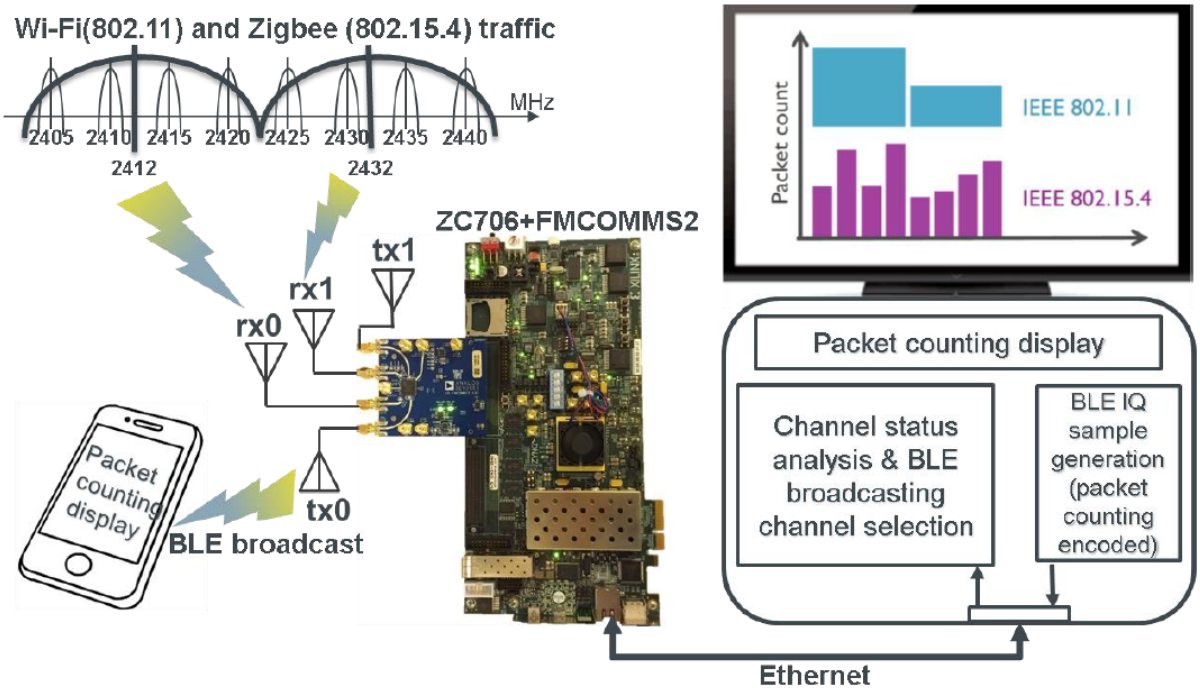

Figure 24: Radio resource slicing on ZYNQ SDR – validation scenario

The validation scenario shown Figure 24 proves the real time operation of radio slicing (frequency, time and antenna slicing), and the capability to build the slices upon the virtualized radio hardware. The SDR platform is composed by Xilinx zc706 evaluation board together with Analog Devices fmcomms2 RF front end. A host computer is connected to the SDR platform for visualization and processing mid-long latency task. Three standards are covered in the demo: Wi-Fi/802.11 20MHz mode, ZigBee/802.15.4 2.4GHz version, and BLE.

Ten virtualized preamble detection instances and one BLE (Bluetooth Low Energy) [15] broadcaster instance are created from resource slicing module and a dual-mode preamble detector (as accelerator) in FPGA under the scheduling software in ARM processor. RF front end has two receiving antennas and two transmitting antennas (rx0, rx1 and tx0, tx1 in Figure 24), and works in 40MHz bandwidth mode. For each rx antenna, resource slicing module creates one 20MHz Wi-Fi channel and four 5MHz ZigBee channels (overlapped with Wi-Fi) by five DDC (Digital Down Converter). Central frequency of each rx antenna's DDC can be tuned independently inside the range of 40MHz bandwidth. In the demo, rx0 is tuned to cover upper 20MHz of 40MHz; rx1 is tuned to cover lower 20MHz of 40MHz. Ten (five per rx antenna) slices' IQ sample are routed to ARM processor, then fed one by one to the preamble detector by control software for preamble searching. 9.6Gbps AXI bus is used to construct on chip network. Dedicated AXI links are setup between resource slicing, preamble detector and ARM

processor. ARM reports Wi-Fi and ZigBee packet counting results of each slice to host computer via Ethernet. A program in computer analyses channel status according to packet counting report, then it will: 1) select a good BLE broadcasting channel from three options: 2402MHz, 2426MHz, 2480MHz; 2) generate IQ samples for BLE broadcasting packet with Wi-Fi and ZigBee packet counting information encoded; 3) Send BLE IQ samples to ARM processor in the SDR platform via Ethernet. Resource slicing module broadcasts the BLE packet into the air via the right tx slice (frequency, timing and antenna) under ARM software control. Finally, a user can read the BLE message, which contains Wi-Fi and ZigBee packet counting information, on their equipment (phone, pad) screen by general purpose BLE scanner App, for instance, LightBlue and BLE Scanner. In the demo, BLE broadcasting is an example of mid-long latency service which is handled by host computer instead of FPGA.

A dual-mode preamble detector (16-sample auto-correlation algorithm) is designed based on similar structure of Wi-Fi and ZigBee's preamble. According to the standards, at Wi-Fi baseband rate 20 Msps and ZigBee half baseband rate 1Msps, both preambles have the form of repeating 16-sample random signal. The only difference is that: Wi-Fi' preamble has 160 samples which is generated by repeating 16-sample signal 10 times; ZigBee repeats 8 times, which results in 128-sample preamble.

C language is used to develop both FPGA and ARM software. By Xilinx HLS C, resource slicing and preamble detector blocks achieve 200MHz clock speed. AXI bus run at 64-bit 200MHz mode. Each AXI 64-bit word contains two IQ samples (16-bit I and 16-bit Q sample for each antenna). To process 512 samples, which means 25.6us under Wi-Fi baseband rate, 512us under half ZigBee baseband rate, preamble detector needs only 1094 clocks, i.e. 5.47us, according to FPGA synthesis report. So, it is fast enough to handle two Wi-Fi slices and ten ZigBee slices. Because the speed of accelerator consuming IQ sample is higher than speed of ten rx slices generating IQ sample. The whole processing system won't lose any IQ-sample/signal, as if there are ten preamble detector running in fully parallel (logically). Different from technology in [26], multiple virtual AP (Access Point) share one Wi-Fi channel by time division, our implementation allows running multiple APs in multiple channels concurrently without implementing standalone FPGA block for each channel.

For the control software, a basic control slot length is 25.6us (512 samples of Wi-Fi), which is also the basic time slot length of resource slicing. 20 slots compose a control period. Different tasks, such as ten preamble detection tasks, one BLE broadcasting task, host computer communication task, tx/rx frequency tuning task, are scheduled in different time slots according to upstream producing rate and latency requirement, just like a computer running multiple programs. By carefully arranging the schedule, there are ten rx instances and one tx instances running in fully parallel from the end user point of view. In this way, a set of radio hardware resources (RF front end, accelerator, etc.) get virtualized to multiple instances.

5.1.3 Testbed integration

The ZYNQ based SDR platform is foreseen in the w-iLab.t testbed, an operating system image with appropriate Xilinx tools is prepared for users to experiment with the SDR communication link. For more details please refer to D6.1, as well as the online repository on the ORCA project website.

5.2 Relation to showcase

Showcase 2 and showcase 3 both use the hybrid architecture (i.e. ARM processor + hardware accelerator) to support radio hardware virtualization, which is needed for flexibility.

The low latency control loop and round trip time profiling is significant for showcase 2, where the robot control link cannot tolerate jitter and delay. The radio slicing and virtualization is essential, especially for showcase 3, where we will use these techniques to coexist with other waveforms and improve the overall spectrum efficiency.

5.3 Implementation plan

5.3.1 SDR control plane improvements

We plan to further reduce the round trip time measured in the SDR ping experiment. We identified that a significant amount of time is lost when the software tries to store data into the hardware accelerator. This could be potentially changed by using a static ram, or also referred to as the On Chip Memory (OCM) of ZYNQ processor.

With the advantages of SDR, we also want to exploit the sampling rate control. Latency can be reduced further by shortening pocket length (increase sampling rate). Or flexible bandwidth can be achieved by changing sampling rate.

Furthermore, a host based interface API will be offered to experimenters, for easy control of the virtualized radios on ZYNQ SDR.

5.3.2 Radio slicing: resource allocation, instantiation, and coordination

The focus of year 1 is mainly on frequency slicing and hardware virtualization, and some basic time division functionalities. Due to the effort of basic PHY communication on SDR, there is limited advances regarding the real virtualization work. Only preamble detections are shared between technologies, the for the complete communication chain, multiple RATs are running concurrently on SDR by using multiple hardware instances.

It is foreseen that in next year we will better supports resource slicing and hardware virtualization, (not only preamble detection but also full transceiver), the virtualized slices will be can be constructed based on flexible waveform (OFDM or/and DSSS).

5.3.3 Testbed integration

The code repository, including the radio control plane for slicing and virtualization, will be updated on regular bases, which will be made available on ORCA portal and on w-iLab.t testbed tutorials. In this way, experimenters will be able to follow the newest developments.

6 RADIO ENVIRONMENT MONITORING

6.1 Implementation Results

This section describes TCD's signal classification framework implemented in Year 1 and the main results obtained. In this deliverable, we describe the general control plane, radio slicing capabilities of this framework, and future steps. For further details on the real-time processing aspects of this implementation, please refer to Section 7 of D3.1.

We provide below the general high-level diagram of our framework. Throughout this section, our description of its capabilities will be mainly centred on its use for identification of RF slices/waveforms. This is in line with its intended use in the Year 1 Showcase 3, where TCD is involved. Additionally, we expect that a considerable percentage of the functionality provided by this framework can also be repurposed for other types of classification problems involving RF signals.

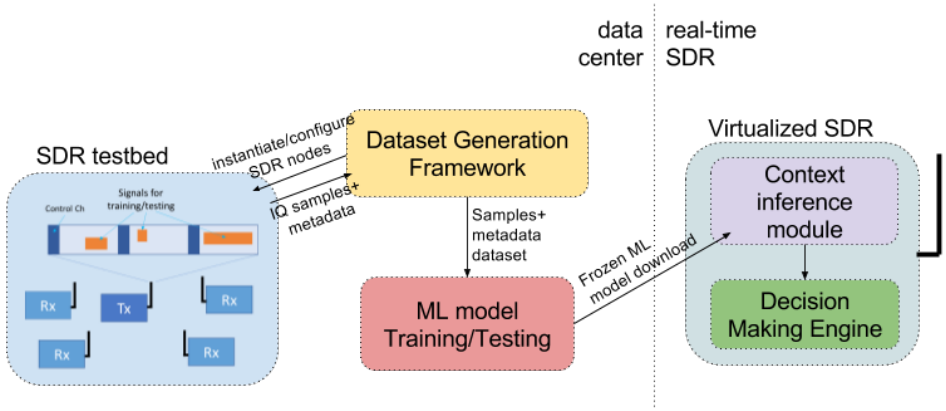

Figure 25: High level diagram of the several components involved in the ML-based signal classifier.

Our framework primarily relies on machine learning (ML) for the design of RF signal classifiers. The design flow of a classifier can be divided into the three main stages: (i) dataset collection, (ii) training and validation, and (iii) inference. During the first stage, the experimenter characterizes the set of possible RF waveforms and respective RF parameters (e.g. frequencies, duty cycle) that its SDR may encounter in the RF environment during its normal operation. For the specified waveforms and parameters, the framework will then generate a dataset composed by several IQ sample files, and associated metadata/labels. This data collection process is managed automatically by the provided framework, which will remotely access, instantiate and configure SDR nodes of the IRIS testbed to perform RF signal transmission/reception and data file storage. The resulting dataset is then used in the following stage to train and test ML-based signal classifiers. Classifiers can then be "frozen", transferred and deployed in real SDR nodes for context awareness.

6.1.1 SDR control plane improvements

In this section, we describe the main functionality and APIs associated with TCD's dataset collection framework. As will be seen in next section, this framework is essential for the design and testing of the machine learning models offered by TCD for the purpose of spectrum coexistence in a multi-radio slice environment.

6.1.1.1 Motivation

To make a supervised ML-based classifier robust to variations in its input, it has to be trained with very large datasets. In areas such as speech processing and computer vision, building such datasets generally requires long, costly, and repetitive tasks of manual sound or image collection, analysis, and labelling.

To circumvent these costs, several off-the-shelf datasets have been made available to the public that ML researchers can use to train and test their own models.

Being a relatively new area, RF signal classification still lacks standardized datasets for ML. This issue is exacerbated by the fact that distinguishing and labelling RF signals can be a more difficult and error-prone process to the human eye than labelling objects in images in computer vision, for instance. The features that distinguish different RF waveforms may be only visible after special pre-processing transformations (e.g. DFT and autocorrelation). Furthermore, effects such as SNR, multipath, and hardware impairments can significantly alter the shape of the signal.

Another challenge specific to RF waveform classification when applied to spectrum coexistence is the degree of detail a radio device needs regarding its environment to efficiently meet its traffic requirements. In general, for efficient coexistence a radio should not only detect the presence of other radios in its surroundings, but also other features, such as their frequency of operation, bandwidth, duty cycle, amplitude. To train a ML model to provide these fine grained features, extensive and rigorous labelling of training datasets would be required, which would not be feasible through simple human eye inspection in a cost-effective amount of time.

To circumvent these issues we designed an automated RF collection and labelling framework, which allows the generation of datasets of different waveforms (e.g. Wi-Fi, PSK, LTE) with distinct RF parameters (e.g. frequency, hardware gains), waveform-specific features (e.g. modulation order, packet length), and DSP transformations (e.g. frequency offset (FO), soft gains, shape filtering, and multipath emulation). Each possible permutation of parameters and waveform types will be translated into an IQ signal that is transferred over-the-air between USRPs in an SDR testbed environment. The received IQ signals and associated parameters are then stored in data files for later access. Additionally, the framework estimates any USRP RF impairments not known a priori, such as SDR FO, clock difference between SDRs, multipath, and SNR levels. These estimated parameters are obtained through periodic exchange of preambles in RF. For further details on the preamble design and robustness, please refer to Section 6 of D3.1.

6.1.1.2 API to generate an RF dataset

To make the data collection framework truly extensible to different types of signals and parameters, the experimenter specifies the types of waveforms and DSP and RF parameters in a declarative manner through a configuration file (.cfg). The data collection script runtime will parse this configuration file and generate a task workflow graph that will be run in a remotely accessed testbed composed by multiple SDRs. The end result is a dataset of samples+metadata files respective to the signals transmitted and received by the testbed SDRs involved in the data collection process.

Each dataset configuration file (.cfg) needs to have the following fields defined:

- Tree of task dependencies – Here the experimenter defines the name and dependencies between each task of the dataset collection process. We define a task as a set of instructions that based on given input parameters generates a file composed by IQ samples and respective metadata/labels.
- Task parameters – For each task, it is specified the names and possible values the parameters involved in the generation of data files can take. Examples of such parameters include the waveform, modulation order, soft gain, hardware front-end address, etc.
- Task scripts – For each task, a script will be run, taking as parameters an iteration of all the possible combinations of parameters provided in “Task parameters”.

6.1.1.3 Framework features

We list below the main strengths of this framework:

- Over-the-air RF samples collection - Machine learning models tend to be very sensitive to the authenticity of the input representation. Models trained on purely simulated IQ data tend to not

generalize well when applied in real RF environments, with all their RF impairments, such as DC/time/frequency offsets, IQ imbalance, non-linear amplifiers, signal shaping and Tx/Rx filtering. Our framework facilitates the collection of over-the-air samples that include those effects.

- Extensibility to any type of waveform, DSP, and RF parameters - This is achieved through a set of scripts that apply each of these signal processing transformations and stores the result in intermediate compressed files. The selection of the types of waveforms, parameters and their values to include in the data collection process is specified through a configuration file.
- Fine grained labelling of signals' features - Thanks to time and frequency synchronization mechanisms put in place through periodic transmissions of preambles, the labels/metadata (e.g. timestamps of transmission, selected channels, SNR) associated to each signal that the Tx USRP transmits will remain valid after the Rx USRP receives and stores them into files.
- Automation of what was previously a long, manual, error-prone process - The number of possible permutations of RF signal shapes/types grows exponentially with the number of parameters the experimenter specifies as valid (e.g. range of frequency and time offsets, types of waveforms).
- Easy to verify the correctness of the produced dataset - We store intermediate results in compressed files whose signals can be easily visualized as spectrograms, DFT or time series in the form of of images (.png), and metadata can be represented as json files. In case some of the data files have been corrupted (e.g. in case of interference), we can select them to be re-run, while keeping other files intact.
- Efficient computational resource use - As the workflow is represented in form of directed acyclical graph, its tasks can be parallelized when multiple cores are available. The experimenter can also interrupt and resume the data collection process at any point.

6.1.2 Radio slicing: resource allocation, instantiation, and coordination

Making use of the previously described data collection/labeling framework, TCD also designed several ML-based classification and feature estimation models that take RF signals as input. These models can be used either for protection of incumbents in dynamic spectrum access scenarios, or for more advanced channel selection schemes that better meet a radio's specific traffic requirements. In a multi-radio slice environment, the provided ML models can identify the type of slices/waveforms and other features (e.g. bandwidth, centre frequency, and time dynamics) present in an SDR's surroundings.

Our main goal, in the context of showcase 3, is to employ these models to assist an SDR at selecting for operation RF channels where other neighbour radios have compatible MAC schemes to its own (e.g. LBT, ALOHA, or TDMA). To design, train and test these ML models, labelled RF samples datasets are required. This can be obtained through the dataset collection framework described in the previous section.

Like the dataset collection, the design, train and validation of ML models can take place offline in a more centralized point of the network. For this stage, we employ currently mature, widespread deep learning frameworks, such as TensorFlow and Caffe. This way we can design, train and test ML models as we would normally would when using these tools locally.

6.1.2.1 Robustness to RF impairments and noise

One particularly relevant design decision for the performance of a ML classifier is the format of its input data. We have experimented with three main formats of data preprocessing: spectrograms, IQ, and amplitude+phase shift. In [16], we have shown that in the context of radar signal detection,

amplitude+phase shift provides the best performance for low SNRs. Some of these results are shown below for the particular problem of detection of radar pulsed signals.

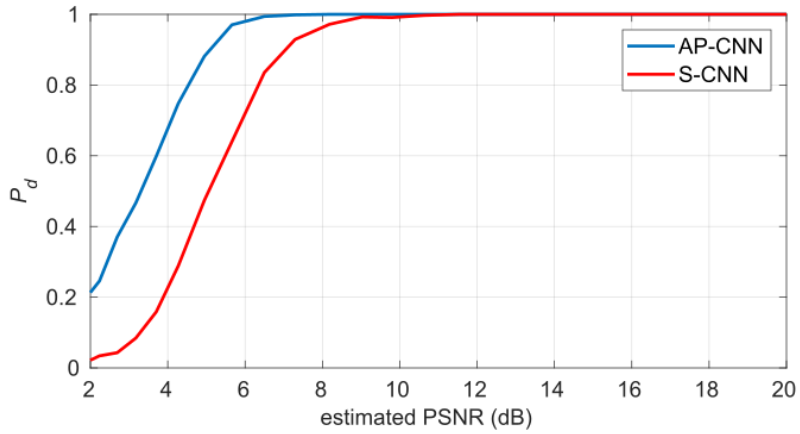

Figure 26: Probability of detection of a radar phase modulated pulse for two different convolutional neural network (CNN) models, one using spectrograms (S-CNN) and the other amplitude+phase shift (AP-CNN) as input data formats.

Additionally, we have shown in [16] that the provided CNN models can be used for distinguishing radar and other types of communication systems like LTE and Wi-Fi. This feature is relevant in Dynamic Spectrum Access (DSA) scenarios, where a secondary user or sensor network has to distinguish signals transmitted by incumbents from those transmitted by other secondary users.

In [16], we demonstrated the capabilities of a CNN model at discerning the channel hopping and duty cycle patterns of another radio transmitter operating in a range of 10 MHz. It was stipulated beforehand that this radio could use 10 different types of channel access patterns, which we denoted as scenarios. Below, we show the probability of correctly identifying these scenarios for different SNR levels. As can be seen, our classifier could achieve error rates of less than 10% for SNRs even below 0 dB.

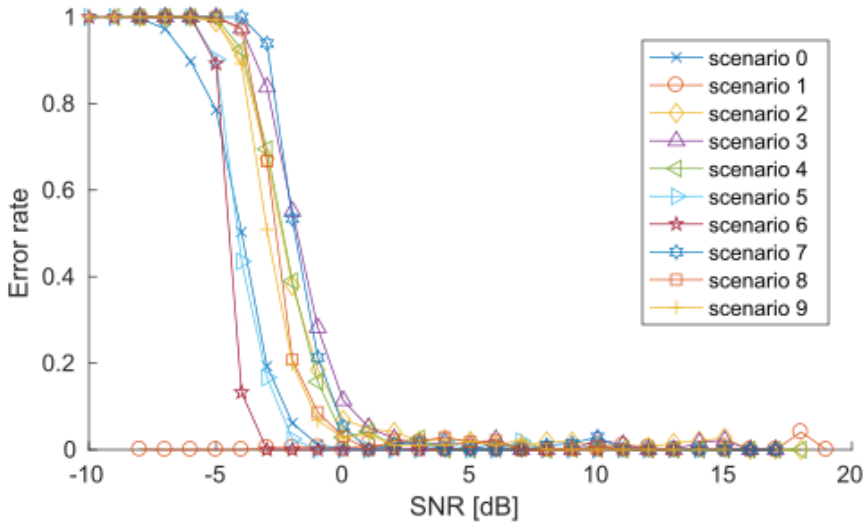

Figure 27: Error rate of the designed CNN model in [16] to correctly discern another radio's channel access scheme pattern.

6.1.2.2 API to reuse of ML models for classification

When picking a deep learning model for classification, the experimenter may pick one of the several that we make available through our testbed or design his/her own. If one of the existing models is picked,

the experimenter has the additional option of picking a "frozen" model that has been trained and can be plugged in an SDR for context inference, or a training "checkpoint", that the experimenter can use as a starting point for the training of a new model. This functionality is already supported by most deep learning frameworks, such as TensorFlow and Caffe.

As seen earlier, the input signal representation is crucial for the performance of the classification task. Hence, we additionally make available to experimenters the functionality of conversion from IQ samples to the chosen input representation through GNURadio blocks and/or python scripts. The experimenter may pick one of the already existing blocks or design his own.

6.1.2.3 Framework functionality

Here we summarize some of the functionality provided by the TCD-designed ML models and framework:

- Waveform classification - The trained models can distinguish between different types of signal waveforms present in a given radio environment, allowing an SDR to make the best decisions to maximize its QoS.
- Waveform parameter estimation - Classifying what type signal is present in a radio environment is not sufficient in most of the coexistence scenarios. For this reason, we also provide feature estimation capabilities that allow SDRs to localize signals' bandwidth, channel, duration, and time.
- Robust performance - Results, so far, as shown that the provided classifiers can identify signals even below the noise floor or in the presence of other types of interferers.
- Reusability - The provided framework allows peaking pre-existing models to be retrained or to be directly deployed in an SDR.

6.2 Testbed Integration

We make available to future experimenters the following functionality:

1. data collection framework to create new RF sample datasets for training and testing new ML-based models. This framework is made available through a git repository.
2. RF signal+metadata datasets, including Wi-Fi, PSK, QAM and other waveform types that TCD generated for the purpose of showcase 3.
3. trained and validated deep learning models and associated preprocessing scripts TCD designed for showcase 3. Each model is provided in the form of a checkpoint for further training/tuning or as a frozen model for inference.
4. integration of the frozen deep learning models with GNURadio, and control through simple APIs. This feature is part of the git repository referred in point 1.

6.2.1 Example of dataset generation process

Here, we show an example of how to use the offered dataset collection framework for the generation of Wi-Fi and PSK signals with different DSP/RF parameters. Our setup consists of two USRPs, one as a Tx and the other Rx, which will be remotely accessed, instantiated and controlled by the dataset generation framework according to a configuration file provided by the experimenter. We illustrate this procedure below.

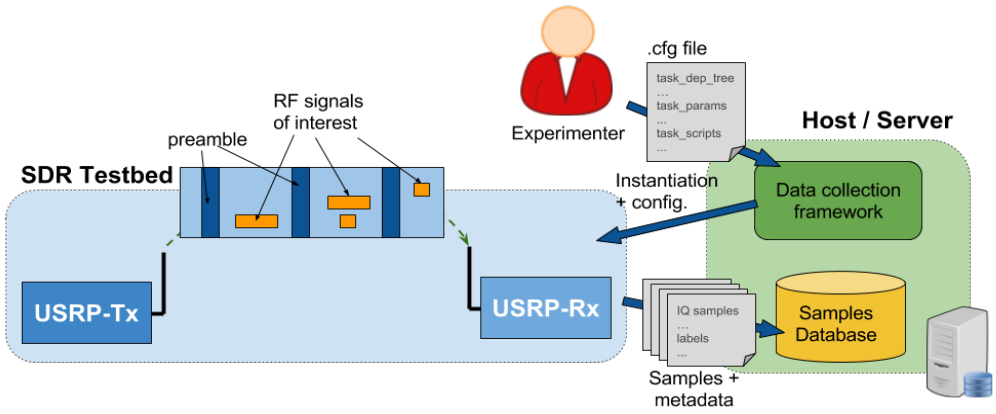

Figure 28: Illustration of a RF signal collection procedure.

The experimenter starts by providing a configuration file (.cfg) where he/she defines the tasks, their dependencies, scripts, and experiment waveforms/parameters. We provide below a simple example of how such fields can be set in a configuration file. We omit parameters that are not relevant for the purpose of this example.

dependency_tree:	**tasks:**	**task_scripts:**
DSPTx: waveform	waveforms:	waveform: wv_script
RF: DSPTx	Wi-Fi:	RF: rf_script
RF_to_spectrogram_png: RF	PDU_length: [500,1000,1500]	DSPTx: dsptx_script
	PSK:	RF: rf_script
	modulation_order: [2,4]	RF_to_spectrogram_png: rf2_spectrogram_script
	DSPTx:	
	CFO: linspace(-0.4,0.4,10)	
	time_offset: linspace(0,1000,10)	
	RF:	
	USRP_tx_gain: range(0,20)	
	USRP_rx_gain: range(0,20)	
	USRP_tx_IP: 192.168.5.20	
	USRP_rx_IP: 192.168.5.21	
	RF_frequency: 3.5e9	
	RF_to_spectrogram_png:	
	img_size: (256,512)	

The data collection framework runtime will parse this configuration file and generate a task workflow graph that will run in the remotely accessed Tx and Rx USRPs. We list below the description of the tasks that will be run as a result of this graph:

1. 5 “waveform” data files are generated using the wv_script; three files with Wi-Fi samples with different PDU lengths, and two files with BPSK and QPSK samples, respectively.
2. For each of the 5 generated waveform files, 100 data files are generated for different permutations of frequency and time offsets.

3. For each of the 5x10000 DSPTx data files generated, 400 RF files are generated with different combinations of Tx and Rx USRP gains. This stage involves two remotely accessible USRP devices, one as a Tx and the other as Rx, which will be instantiated and controlled in parallel by the rf_script.
4. For each "RF" file, a spectrogram image is generated with 256 frequency bins and 512 FFT windows.

We provide below some examples of the spectrograms of the Wi-Fi and BPSK signals collected by the Rx USRP. To demonstrate the correctness of the time/frequency synchronization mechanism put in place, we also show bounding boxes, as yellow squares over each spectrogram, that the data collection framework automatically generated to delimit the frequency and time ranges of each transmission. Other metadata generated by the framework, such as SNR and type of waveform, can easily be accessed through json files.

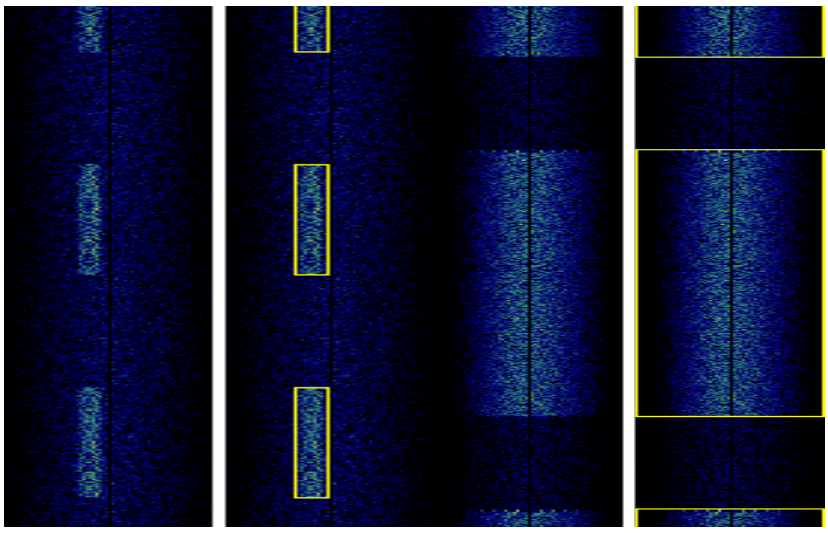

Figure 29: Spectrograms of BPSK and Wi-Fi signals and respective bounding box labels, as yellow squares, generated by the dataset framework to delimit the transmissions in time and frequency. The fact that the bounding boxes match with the signals' frequency and time ranges shows that the synchronization between the Tx and Rx USRPs was successful.

6.2.2 Trained and frozen deep learning models

As the input signal representation (e.g. spectrogram and IQ) and type of deep learning model are essential for the success of a classification task, both of these components have to be well documented

All components necessary to their handling will be made available to future experimenters. A GNURadio block will be provided for the conversion of raw IQ samples to any of the used input signal representations.

6.3 Relation to the showcase

The showcase 3 will be divided into two separate phases: (i) ML model generation, and (ii) over-the-air radio slicing coexistence.

6.3.1 ML model generation

In the first stage, we will demonstrate the functionality related to the dataset generation and ML model training for RF waveform/MAC identification described above. In particular, we will show how an experimenter can define the set of RF environment scenarios and features that are relevant to his

experiment. This metadata is then used to create a deep learning model that is capable to distinguish between different RF waveforms and estimate parameters, such as channel, bandwidth and MAC access features.

For the purpose of this showcase, the experimenter defines in a dataset framework configuration file (.cfg) an RF environment composed by three types of waveforms: (i) 20 MHz-wide CSMA/CA-based WLAN, (ii) 5 MHz-wide ZigBee, and (iii) 1 MHz-wide ALOHA-based single carrier transmitter. The framework will read this file and generate a dataset using one USRP board as Tx and another as Rx. These USRP boards will be located in the IRIS testbed, and be accessed remotely. Several data files will be created with the collected IQ samples and respective labels. The experimenter can inspect the generated files to see that everything is working correctly.

It is based on the generated dataset that the ML classifier model is trained and tested. We will show how the ML model can be configured for the showcase purposes. As the training process is very long taking several minutes or hours, it may be pre-recorded as a video for later illustration during the showcase.

6.3.2 Multi-radio slice coexistence

In the coexistence stage, a "frozen" ML model is integrated with IMEC's virtualized radio and used to infer not only the RF channel allocation and usage, but also type of transmitters. Based on the outputs of the ML model, a list of MAC-compatible channels will be provided to the IMEC's radio to assist it during channel selection procedures. We define as a MAC-compatible channel a range of frequencies, where the coexisting radio devices utilize a MAC access scheme that will not cause service disruption to the IMEC radio and vice-versa.

6.4 Risk Analysis

The results obtained so far provide confidence, but no certainties regarding the accuracy and robustness of the developed RF signal classifiers. There is a risk that the developed classifiers are overfitted to the emulated RF scenarios in the testbed. In such case, these classifiers will not generalize well to different types of impairments or unforeseen scenarios. This is a general limitation of machine learning-based approaches, and can only be circumvented through the use of sufficiently large training datasets, extensive testing, and rigorous benchmarking.

We plan to extend the testing of our algorithms to scenarios with slight variations compared to the ones where the classifiers were trained for. Such variations may include different RF waveform filter shapes, user traffic statistics, and multipath environments. These results will provide us a clearer picture of the generalization capabilities of the proposed techniques.

6.5 Implementation Plan

This section describes TCD's plans for implementing a framework to manage both the radio slicing on SDRs and the association between the radios slices and host nodes in Year 2. We want to enable a dynamical instantiation of radio slices and the configuration of the corresponding connection between the radio slices and host nodes. We plan to develop this framework by combining a radio hypervisor, e.g., HyDRA, with a virtual switch, e.g., Open vSwitch.

6.5.1 SDR control plane improvements

The dynamic instantiation of radio slices will require communication between the current radio slices and the radio hypervisor. The radio slices will inform the radio hypervisor about their network load and the radio hypervisor will decide whether to increase or decrease the number of resources available to them. We will achieve this by extending the radio hypervisor to support the exchange of control messages with the radio slices.

6.5.2 Radio slicing: resource allocation, instantiation, and coordination

Our framework will enable the instantiation of radio slices at runtime, without interrupting the operation of the ones already instantiated, as illustrated in Figure 1. The radio hypervisor informs the radio slice 'A' about the upcoming reconfiguration. After the radio resources are freed, the radio hypervisor instantiates the radio slice 'B'. We plan to use a virtual switch to coordinate the configuration of the connection between the newly instantiated radio slice and its host node.

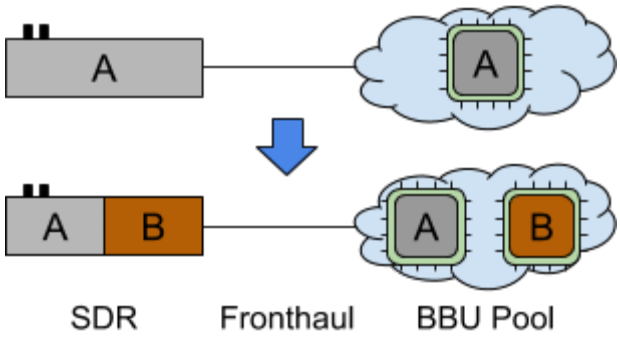

Figure 30: Reconfiguration of an SDR realizing a single radio slice, with the processing being done by a host at a Base Band Unit (BBU) pool.

6.5.3 Testbed integration

We plan to make available to future experimenters the following functionality:

1. Creation of radio slices at run time.
2. Dynamic allocation of resources to radio slices.
3. Coordinated setting of routing rules between radio slices and their hosts.

7 NS3 BASED PROTOTYPING PLATFORM FOR RAT INTERWORKING

Figure 31 below shows the anticipated Multi-RAT system described in more detail in deliverable D2.1 [7]. The basic underlying idea is to enable researchers to do experiments on a Multi-RAT system including Wi-Fi, LTE as well as 5G. This will lead to a better understanding of the practical trade-offs when dealing with existing and new technologies.

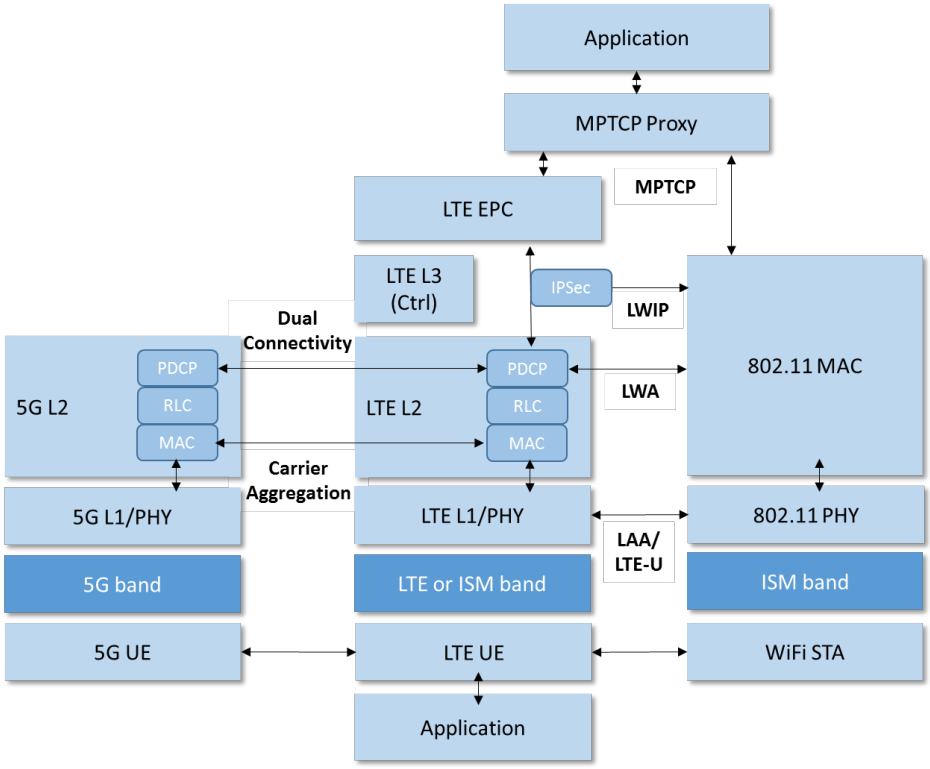

Figure 31:Anticipated Multi-RAT Platform

For the first year the system depicted in Figure 32 was targeted focusing mainly on a combined Wi-Fi and LTE platform allowing for RAT interworking experiments as well as creating parallel 5G link that will be integrated in Year 2. This system will also be used in the first Open Calls for experimentation as described in deliverable D2.2 [4].

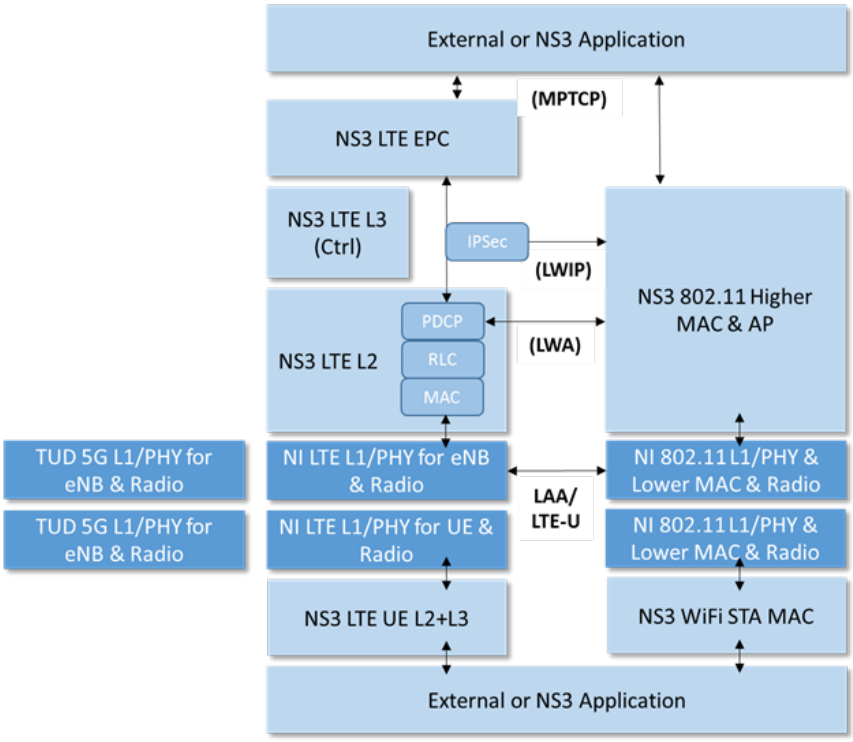

Figure 32: Targeted Year-1 Platform

In the following the status of the system implementation results as well as the plans for Year 2 are described with the specific focus on the control plane.

7.1 Implementation results

7.1.1 SDR control plane improvements

As described in deliverable D2.2. [4], for the targeted platform shown above the goal is to enable a parametric control of the PHY and lower MAC as well as the high MAC and upper layer network protocols. As part of the development efforts within the first year the implementation of the remote configurable TestMan interface by TUD has been ongoing that allows to reconfigure control parameters on the LabVIEW based platform where the NI LTE and 802.11 Application Framework as well as the NS3 based upper layer protocol stack runs.

On top of the implementation of the toolset to remotely control the LabVIEW based platform an analysis started what parameters should be exposed to the experimenter to enable the anticipated show cases. As for the NI LTE Application Framework related physical layer the following parameters can be reconfigured on run time at TTI level (at the eNB):

- UE ID
- Resource allocation per UE
- MCS per UE

Furthermore, the semi static parameters that can be reconfigured are:

- Carrier frequency
- Transmit power

As for the layer 2 that is implemented on the NS-3 platform, the following semi-static parameters can be reconfigured:

- Traffic conditions (IP packet size, time between the packets, external or internal traffic)
- Scheduler algorithm with MCS selection criteria

As the Wi-Fi system has been developed in Year 1, no dynamic parameter reconfiguration can be supported yet.

7.1.2 Radio slicing: resource allocation, instantiation, and coordination

As shown in Figure 31 above, there are multiple interworking options between an LTE and a Wi-Fi system possible. As part of the Year 1 activity several LTE-Wi-Fi interworking options have been evaluated as input for the Open Call for Extension 1 that are listed below (see also Figure 33):

1. **Multi-path TCP (MPTCP):** Multi-path TCP allows to split/aggregate data traffic using a corresponding proxy in the IP network.
2. **LTE/WLAN Radio Level Integration (LWIP):** LTE WLAN Radio Level Integration with IPsec Tunnel to split/aggregate the data traffic above layer 2
3. **LTE-WLAN Radio Aggregation (LWA):** LTE-WLAN Radio Aggregation to split/aggregate on PDCP level
4. **LTE-License Assisted Access or LTE-Unlicensed (LAA/LTU-U):** Use unlicensed Wi-Fi spectrum for LTE transmission

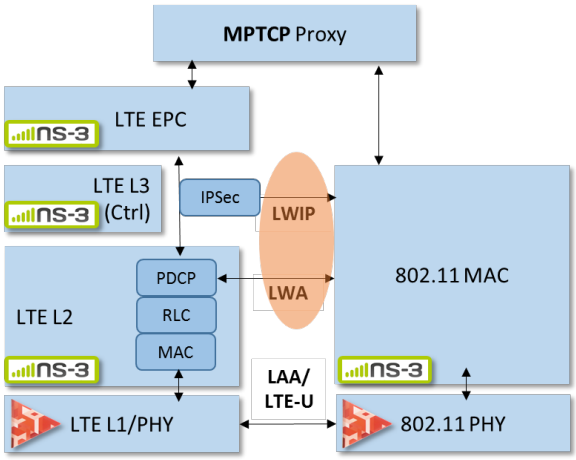

Figure 33: LTE-Wi-Fi Interworking Options

Because option (1) is already supported by the NS-3 simulator and there is also an example available to extend the NI LTE Application Framework to support option (4) in particular option (2) and option (3) are of interest for future extension to provide a complete set of technologies to find the right trade-offs through experimentation.

Option (2) and (3) are two types of LTE-Wi-Fi integration architectures, which are defined in 3GPP Release 13. Figure 34 shows the protocol stacks of these architectures:

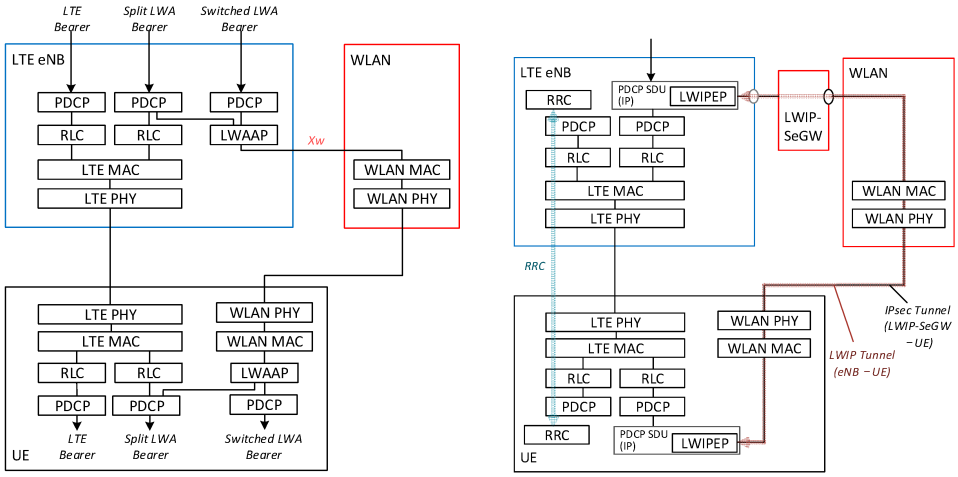

Figure 34: LTE-WLAN interworking architectures in 3GPP Rel.13: (a) LWA in a non-collocated scenario and (b) LWIP 17 [17]

LWA: Similar to the LTE 3GPP Rel. 13 [17] feature of Dual Connectivity in LWA the system aggregates packets between LTE and WLAN on PDCP level. To utilize LTE and WLAN simultaneously split and switched bearers are supported. The PDCP PDU packets sent via WLAN are encapsulated in LWA Adaptation Protocol (LWAAP) which carries bearer identity. The WLAN AP only interacts with the LTE eNB and there is no interaction with LTE core network required. This LWA Extension focuses on the non-collocated deployment scenario where WLAN AP is connected via the Xw interface. WLAN mobility and security aspects are not focus of this Extension and could be simplified. For the LTE-WLAN Aggregation Operation the WLAN Termination (WT) Addition and Release procedures shall be considered. More detailed functional description can be found in 3GPP TS 36.300 V13.8.0, section 22A.1 [17].

LWIP: In LWIP the PDCP SDU packets are sent from the LTE enB to the LTE UE via WLAN using an IPSec tunnel. The WLAN is hidden to the LTE core network (Except for WLAN authentication). In general DL and UL are support via LWIP but without re-ordering functionality. For the LTE DL of a data bearer, at the LTE UE the packets received from the IPsec tunnel are forwarded directly to upper layers. For the UL, the bearer packets sent over the LWIP tunnel are encapsulated using LWIPEP as specified in 3GPP TS 36.361. WLAN mobility and security aspects are not focus of this Extension and could be simplified. More detailed functional description can be found in 3GPP TS 36.300 V13.8.0, section 22A.3 [17].

7.1.3 Testbed integration

As described above, the combination of the NS3 Wi-Fi platform with the NI 802.11 Application Framework is available for the upcoming Open Calls for extension and experimentation. As one core element of the existing Multi-RAT system is based on NS3 also the extension can be seamlessly used as part of the overall experimentation platform.

7.2 Relation to showcase

As described in deliverable D2.1 [7], the showcase for the NS3 based prototyping platform is showcase 4: “Interworking and Aggregation of Multiple Radio Access Technologies”.

7.3 Implementation plan

7.3.1 SDR control plane improvements

In Year 2 new features implemented on the NS-3 based platform that are described in deliverable D3.1 [2] shall be made remotely reconfigurable such as:

- NS3-LTE: Setup MAC scheduler model and configuration
- NS3-LTE: Setup and configuration of MAC and RLC through RRC
- NS3-WIFI: Setup Adhoc or Infrastructure mode, configure various rate control algorithms
- NS3: Several instantiations of LTE and/or 802.11 modules

These features will extend the experimentation options that will be made available through the Open Call for experimentation in Year 2.

7.3.2 Radio slicing: resource allocation, instantiation, and coordination

As in Year 1 the effort of integration the NI 802.11 Application Framework with the NS-3 was the key focus, the implementation of the LWA and LWIP features are part of the Open Call or extension 1. This is one key result expected in Year 2 with the goal completing the LTE-Wi-Fi interworking experimentation platform.

In addition to the LTE-Wi-Fi interworking platform further implementation efforts are planned in Year 2 to support also an LTE – 5G interworking setup that is related to the current discussion of 5G non-standalone operation. In particular the following two options will be investigated:

- **Dual connectivity**: Allows to split/aggregate traffic on PDCP level
- **Carrier aggregation**: Allows to split/aggregate traffic on MAC level

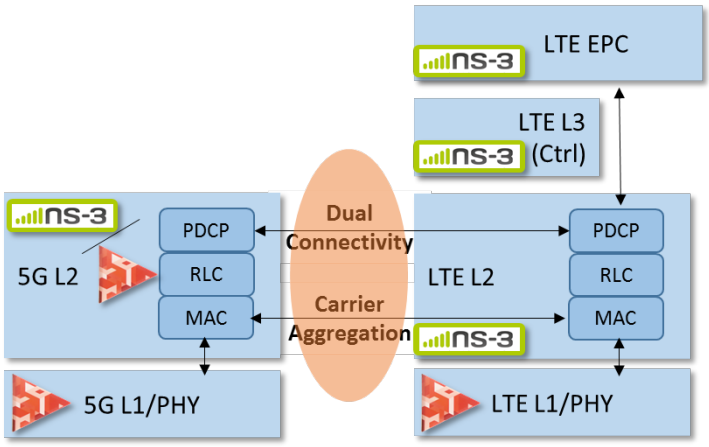

Figure 35: LTE-5G Interworking Options

7.3.3 Testbed integration

The NS3 based changes are an inherent part of the Multi-RAT platform. The targeted implementation for the 5G MAC shall include options for interworking with the LTE stack using e.g. dual connectivity or carrier aggregation.

8 CONCLUSIONS

ORCA focuses not only on extending the current state-of-the-art SDR technology data-plane functionality to achieve higher rates and lower latencies, but also on enabling fine and flexible end-to-end control over SDR networks. ORCA tackles the latter problem in the two following ways. First, it provides a diverse set of configuration, parametrization and monitoring tools that can be run and controlled at different levels of the protocol stack and on different resources of the SDR computational architecture. Secondly, ORCA solutions enable radio slicing, more specifically, the instantiation and tailoring of multiple logical radio networks with distinct features and capabilities on the same underlying infrastructure.

This document described the main contributions made to the ORCA control-plane architecture during Year 1. Partners provided an overview of the architecture and functionalities of their SDR implementations, motivating the design decisions taken. The implemented functionalities included:

- TUD, NI - remote and GUI-based tools for control and monitoring of mmwave systems, and scheduling and beam steering capabilities to support radio slicing
- KUL – A host-controlled framework to experiment with a network of SDRs capable of in-band full duplex collision detection
- TUD – a flexible PHY implementation with lower MAC functionalities
- IMEC – a hybrid FPGA platform with flexible MAC, which enables experimenters to instantiate and configure multiple RATs in a single ZYNQ SDR
- TCD – a radio environment monitoring framework, which provides several tools and algorithms that facilitate RF data collection, machine learning-based RF signal classification model training and testing, and deployment of the generated classification models in SDRs
- NI - An ns3-based prototyping platform for RAT interworking, which enables researchers to do experiments on a Multi-RAT system that includes Wi-Fi, LTE as well as 5G.

Additionally, a brief description was given on how the aforementioned implementations were integrated in the first year ORCA showcases and in the partners' testbeds. Lastly, each partner outlined their plans for the extension of the ORCA control-plane architecture in Year 2.

9 REFERENCES

[1] MiWaveS, Beyond 2020 Heterogeneous Wireless Networks with Millimeter-Wave Small Cell Access and Backhauling, Website, http://www.miwaves.eu, December 2017

[2] ORCA Deliverable D3.1, "First operational real-time SDR platforms", December 2017

[3] MiWaveS, JOINT DELIVERABLE D6.3-D6.4-D6.5, "mmWave Access and Backhaul Link Tests and Presentation of Final Demonstrator", June 2017.http://www.miwaves.eu/deliveraables/D6.3_6.4_6.5.html

[4] ORCA Deliverable D2.2, "Technical requirements of the ORCA test facility", https://static.martel-innovate.com/wp-content/uploads/sites/4/2017/01/ORCA_D2.2_Final_v1.1.pdf, June 2017

[5] ORCA Deliverable D2.3, "Development and integration of showcases in Year 1", December 2017

[6] National Instruments, "LabVIEW Real-Time Module", Website, http://www.ni.com/labview/realtime/, December 2017

[7] ORCA Deliverable D2.1, "Definition of ORCA showcases", https://static.martel-innovate.com/wp-content/uploads/sites/4/2017/01/ORCA_D2.1_Final.pdf, March 2017

[8] National Instruments, "PXI Controller", Website http://www.ni.com/en-us/shop/select/pxi-controller, Dezember 2017

[9] N. Michailow et al., "Generalized Frequency Division Multiplexing for 5th Generation Cellular Networks," in IEEE Transactions on Communications, vol. 62, no. 9, pp. 3045-3061, Sept. 2014.

[10] AXI DMA v7.1 LogiCORE IP Product Guide, PG021, October 4, 2017. https://www.xilinx.com/support/documentation/ip_documentation/axi_dma/v7_1/pg021_axi_dma.pdf

[11] Jiao, X., Moerman, I., Liu, W., & de Figueiredo, F. A. P. Radio hardware virtualization for coping with dynamic heterogeneous wireless environments, at Conference on Cognitive Radio Oriented Wireless Networks (Crowncom), 2017

[12] Ismael Gomez-Miguelez, Andres Garcia-Saavedra, Paul D. Sutton, Pablo Serrano, Cristina Cano, and Doug J. Leith. 2016. srsLTE: An Open-source Platform for LTE Evolution and Experimentation. In Proceedings of the Tenth ACM International Workshop on Wireless Network Testbeds, Experimental Evaluation, and Characterization (WiNTECH '16). ACM, New York, NY, USA, 25–32. https://doi.org/10.1145/2980159.2980163

[13] GNURadio, the Free and Open Software Radio Ecosystem. http://gnuradio.org/. ([n. d.]).

[14] Mendes, J., Jiao, X., Garcia-Saavedra, A., Huici, F., & Moerman, I. (2017). Cellular Access Multi-Tenancy through Small-Cell Virtualization and Common RF Front-End Sharing.

[15] Bluetooth Low Energy, https://www.bluetooth.com/specifications/adopted-specifications.

[16] Ahmed A. S. Selim, Francisco Paisana, Jerome A. Arokkiam, Yi Zhang, Linda Doyle, Luiz A. DaSilva, "Spectrum Monitoring for Radar Bands using Deep Convolutional Neural Networks", IEEE Global Communications Conference (GLOBECOM), 2017.

[17] 3GPP, Release 13 – 36.Series Specifications, http://www.3gpp.org/ftp/Specs/latest/Rel-13/36_series/