GRANT **AGREEMENT NO.: 732174**

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

Orchestration and Reconfiguration Control Architecture

D4.3: Enhanced operational SDR platforms with end-to-end capabilities

Revision: v.1.0

Work package	WP4
Task	Task 4.1, 4.2, 4.3, 4.4
Due date	31/12/2018
Submission date	21/12/2018
Deliverable lead	TCD
Version	1.0
Authors	Joao F. Santos (TCD), Jonathan van de Belt (TCD), Seyed Ali Hassani (KUL), Wei Liu (IMEC), Xianjun Jiao (IMEC),

	Muhammad Aslam (IMEC), Ingrid Moerman (IMEC), Walter Nitzold (NI), Clemens Felber (NI), Vincent Kotzsch (NI), Martin Danneberg (TUD), Ahmad Nimr (TUD), Shahab Ehsanfar (TUD), Roberto Bomfin (TUD)
Reviewers	Wei Liu (IMEC), Walter Nitzold(NI)

Abstract	This deliverable includes the progress and implementation results obtained in Year 2 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 final year.
Keywords	End-to-end, SDR, control plane, radio slicing, virtualisation

Disclaimer

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	
CO	Confidential to ORCA project and Commission Services	

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.

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 FPGAs 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 2. 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 2 end-to-end SDR improvements have been integrated in the partners' testbeds and showcases, and lists the current extension plans for Year 3.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

AnSIC	Analog Self-Interference Cancellation
AP	Access Point
BPSK	Binary Phase Shift Keying
BS	Base Station
CCA	Clear Channel Assessment
C-RAN	Centralised-RAN
DiSIC	Digital Self-Interference Cancellation
E2E	End-to-end
EBD	Electrical Balance Duplexer
FD	Full-Duplex
FER	Frame Error Rate
GFDM	Generalized Frequency Division Multiplexing
GMSK	Gaussian Minimum Shift Keying
GUI	Graphical User Interface
HALO	Hardware in the Loop
HDL	Hardware Description Language
IBFD	In-Band Full-Duplex
IFFT	Inverse Fast Fourier Transform
Imm-ACK	Immediate Acknowledge
IP	Internet Protocol
LTE	Long Term Evolution
MAC	Medium Access Control
M-CORD	Mobile-Central Office Re-architected as a Datacenter
MsgEnc	Message Encoder
MsgDec	Message Decoder
MySVL	Many-to-Many Spectrum Virtualisation Layer
NFV	Network Function Virtualisation
NS	Network Slice
OCM	On-Chip Memory
OFDM	Orthogonal Frequency Division Multiplexing
OSM	Open Source Management and orchestration
PHY	Physical Layer
PNF	Physical Network Function

PRF	Physical Radio Function
PSDU	PHY Service Data Unit
PSK	Phase Shift Keying
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase Shift Keying
RAN	Radio Access Network
RAT	Radio Access Technology
RF	Radio Frequency
RFV	Radio Function Virtualisation
RRH	Remote Radio Head
SDN	Software Defined Networking
SDR	Software Defined Radio
SPI	Serial Peripheral Interface
TDD	Time Division Duplex
VNF	Virtual Network Function
VRF	Virtual Radio Function
XVL	eXtensible Virtualisation Layer

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. To 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 2 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 describes the ORCA vision.
- Section 3 describes NI's efforts to further enhance the direct control of the 60GHz mmWave setup for experimenters together with a beam steering simulation environment as well the integration of the setup in TUD's testbed
- Section 4 describes KUL's in-band full duplex SDR which enable end-to-end networking.
- Section 5 describes TUD's improvements on the SDR control plane as well as the MAC development
- Section 6 describes firstly the IMEC's SDR control plane features to improve the Tx/Rx turnaround time, then presents the initial result for integrating the OFDM FPGA IP cores with embedded Linux on ZYNQ SDR.
- Section 7 describes TCD's many-to-many radio virtualisation, supporting the slicing and aggregation of RF front-ends.

- Section 8 describes NI's enhancements of the ns-3-based Multi-RAT platform with remote control capabilities and an extending Multi-RAT networking example that mimics a more realistic network topology in ns-3.

Each section is then subdivided into: (i) summary of the implementation results obtained during Year 2, (ii) how the implemented solution is integrated in its respective Year 2 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 3.

2 ORCHESTRATING NEXT-GENERATION SERVICES THROUGH END-TO-END NETWORK SLICING

2.1 Towards End-to-End Network Slicing

To cater for the wide range of existing and future network services, with a large diversity in requirements such as reliability, throughput, latency, reconfigurability, etc., future network efforts such as 5G are targeting end-to-end (E2E) network virtualization (also known as network slicing) [1]. Network virtualization allows network resources to be used in a flexible, dynamic, and customized manner, and most crucially, provides isolation between different virtual networks [2]. As a result, each virtual network (or NS) operates as a separate network, and can be individually configured to serve a particular purpose [3]. Services such as critical low latency machine communication in cyber-physical systems, high-speed autonomous vehicle communication, and low-power sensor communication could therefore each be offered a specific network slice, which would be tailored to service requirements, and would guarantee a particular set of performance characteristics.

End-to-end networks can comprise multiple network segments, e.g., radio access networks, transport and core networks, and data centre networks, and these network segments are typically built for different purposes. Network segments also use different media, such as optical fibre, copper cables, and wireless spectrum, and thus employ different technologies and protocols, e.g., xPON, xDSL, and LTE., with unique configurations, policy enforcement and QoS management [4], [5]. Hence, the creation of E2E network slices to provide guaranteed performance requires the slicing of each individual network segment, and the subsequent combination of these network segment slices [4], [6]. This white paper presents the vision of the H2020 ORCA project on combining these individual network slices to create E2E network slices.

2.2 Orchestrating Different Network Segments

Network slices within a network segment are managed by an entity called an orchestrator, that orchestrates the use of network resources and the placement of functionality in a network segment, and also defines the configuration, policies, and management of a network segment. However, as described above, there are fundamental differences between distinct network segments, both in terms of physical resources, and in the way that the network segments are used. As an extreme example, we describe the differences between wired and wireless network segments below.

Firstly, it is not possible to guarantee the information throughput capacity of wireless links, due to the inherent stochastic nature of the wireless medium, whereas the capacity of wired links is predictable and known. Secondly, wireless links are broadcast, and thus have the potential to interfere with other wireless links, whereas in wired links this does not occur [7]. Thirdly, wireless nodes tend to be highly mobile, which means that i) strategies for location and handover are needed, and ii) the traffic can fluctuate significantly as nodes move around in a network.

The communities behind each network segment have their own abstractions and models to manage the particularities of each segment. Therefore, the orchestration of all networks segments by a single community would lead to an oversimplification of the other segments' technicalities, and the loss of fine-grained control over resources. However, the current cross-segment orchestration initiatives are driven by a single community, e.g., the Open Source Management and orchestration (OSM), and the Mobile-Central Office Re-architected as a Datacenter (M-CORD), both are used for deploying

network services in the core network and baseband units in the radio access network, and are led by the NFV (Network Function Virtualisation) and the SDN communities, respectively.

We believe that each network segment should have their own orchestrator, tailored to the segment's particularities, as illustrated in Figure 1. The use of a separate orchestrator for each network segment reduces complexity and breaks down the larger E2E network orchestration problem into smaller parts [8]. In this way, each segment orchestrator can focus on a limited number of well-defined tasks, reducing the software complexity, both in terms of design and implementation. We now highlight the potential tasks of wired and wireless orchestrators.

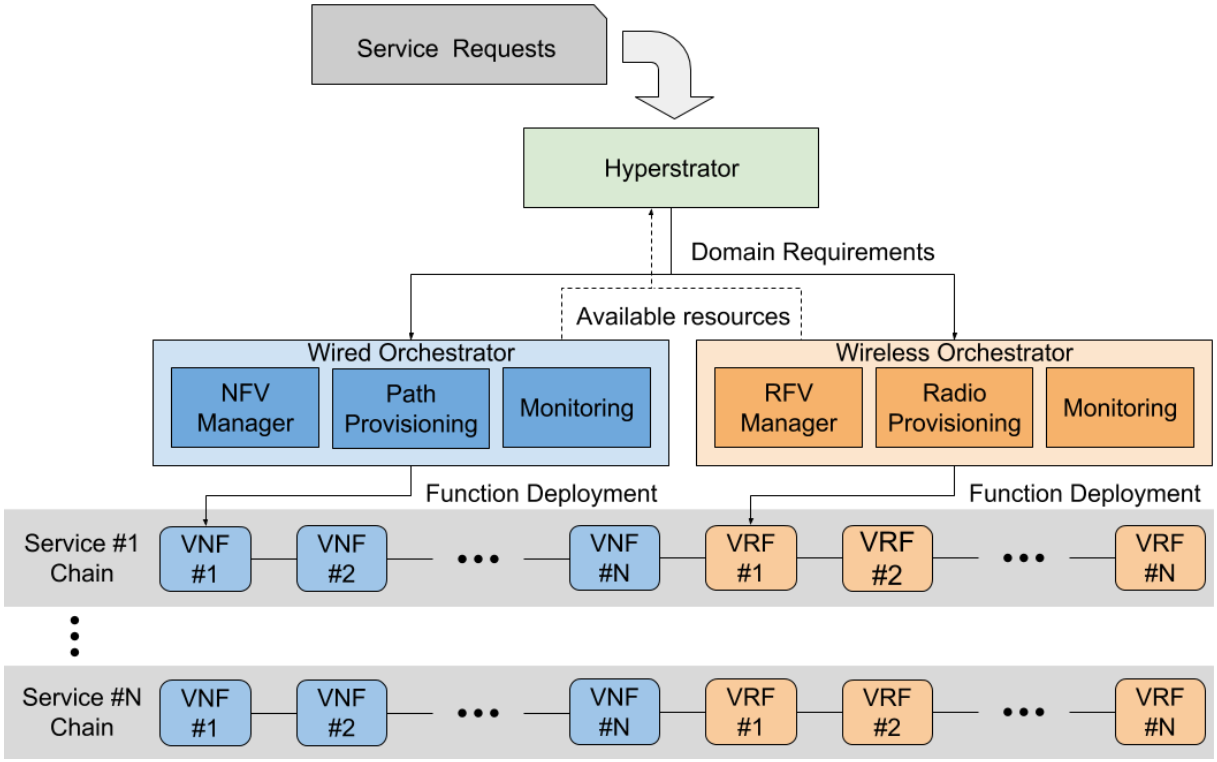


Figure 1: Overview of the Hyperstrator.

The hyperstrator receives service requests and decides the service resource requirements for each network segment, and delegates these to each segment orchestrator. Each orchestrator then provisions resources and deploys services as a chain of virtual functions.

2.2.1 Wired Orchestration

A wired network orchestrator is in charge of provisioning paths and deploying services in a wired network slice, building on SDN and NFV respectively. It should:

1) know the geographical positions, points of presence (the interfaces between networks), storage capacity, computational power, etc. of physical nodes; and the throughput, delay, etc. of physical links;

2) instantiate virtual nodes and virtual links to deploy a network slice to establish a given path;

3) use a network slice to deploy services as a chain of Virtual Network Functions (VNFs) and Physical Network Functions (PNFs); and

4) possess monitoring capabilities to assess the state of the links, nodes, and services.

2.2.2 Wireless Orchestration

A wireless network orchestrator is in charge of instantiating Radio Access Technologies (RATs) and providing radio coverage, leveraging SDRs and Remote Radio Heads (RRHs). It should:

1) know geographical positions, points of presence, computational power, storage capacity, radio front-end capabilities, etc. of physical base stations and access points; the spectrum resources, channel conditions, and the resulting average throughput, latency, etc. of the wireless links; and the maximum movement speed and propagation distance, etc. of radio terminals;
2) provision virtual radio stations and virtual links to deploy a radio slice to cover a given area;
3) use radio slices to deploy radio services, such as RATs, as a chain of Virtual Radio Functions (VRFs) and Physical Radio Functions (PRFs); and
4) possess monitoring capabilities to assess the state of the wireless links, radio stations, and radio services.

The ability to orchestrate E2E network slices across multiple networks segments is still missing, however, which requires a global view of network resources [8].

2.3 Hierarchical Orchestration of End-to-End Networks

Different types of orchestrators are deployed according to the type of resources being managed: wired network orchestrators for managing NFV and SDN and for establishing paths and deploying services; wireless network orchestrators for managing RRHs and SDRs for creating RATs and provisioning radio access. However, E2E network slicing will require a combination of multiple types of orchestrators.

In this scenario, there must be an entity with a global view of the available resources and the capabilities of each orchestrator for establishing and managing flexible E2E networks, leveraging the virtualization of each network segment. This entity would be an orchestrator of orchestrators, which coordinates the interaction between the underlying virtualized infrastructure, namely a hyperstrator.

The hyperstrator would sit on top of different orchestrators for controlling the E2E allocation of resources and management of the entire network. It would be the responsible for mapping high-level E2E network requirements into the requisites for the different networks segments, while each of the underlying orchestrators would then map their own requisites into a realization using the available virtualized resources. The hyperstrator knows the available resources and the status of the current services by gathering information from its underlying orchestrators. Moreover, the hyperstrator must coordinate the combination of slices between network segments for creating E2E NSs. Therefore, it is crucial for the hyperstrator to be aware of the points of presence between network segments, as these are the places where network segments interface and interconnect.

Our proposed hierarchical orchestration scheme is shown in Figure 1: Overview of the Hyperstrator., using a hyperstrator on top of the network segment orchestrators to deploy E2E network services. The underlying orchestrators are responsible for deploying their parts of the service as a chain of virtual or physical network functions and radio functions. Apart from the coordination of resources within an administrative network domain, the hyperstrator could potentially be used for leasing or sharing E2E network resources with other networks operators, through their respective hyperstrators.

2.3.1 Key Benefits

The architecture of the presented hierarchical orchestration scheme based on a hyperstrator and specialized orchestrators is both modular and extensible. Breaking down E2E network orchestration into specialized segment orchestrators designed by domain experts, reduces vulnerability to a single point of failure in the network. In addition, as long as there is a well-defined and open interface between the hyperstrator and the underlying orchestrators, each orchestrator can be easily and independently changed or upgraded.

Furthermore, the separation between systemwide requirements from the requirements of individual network segments facilitates the integration of orchestrators that manage additional (current or future) types of network segments.

Therefore, the architecture we propose enables the end-to-end provisioning of network slices, tailored to the particular requirements of future network services, and guarantees desired end-to-end network performances for different services.

3 MMWAVE SYSTEMS

3.1 Implementation results

3.1.1 SDR control plane improvement

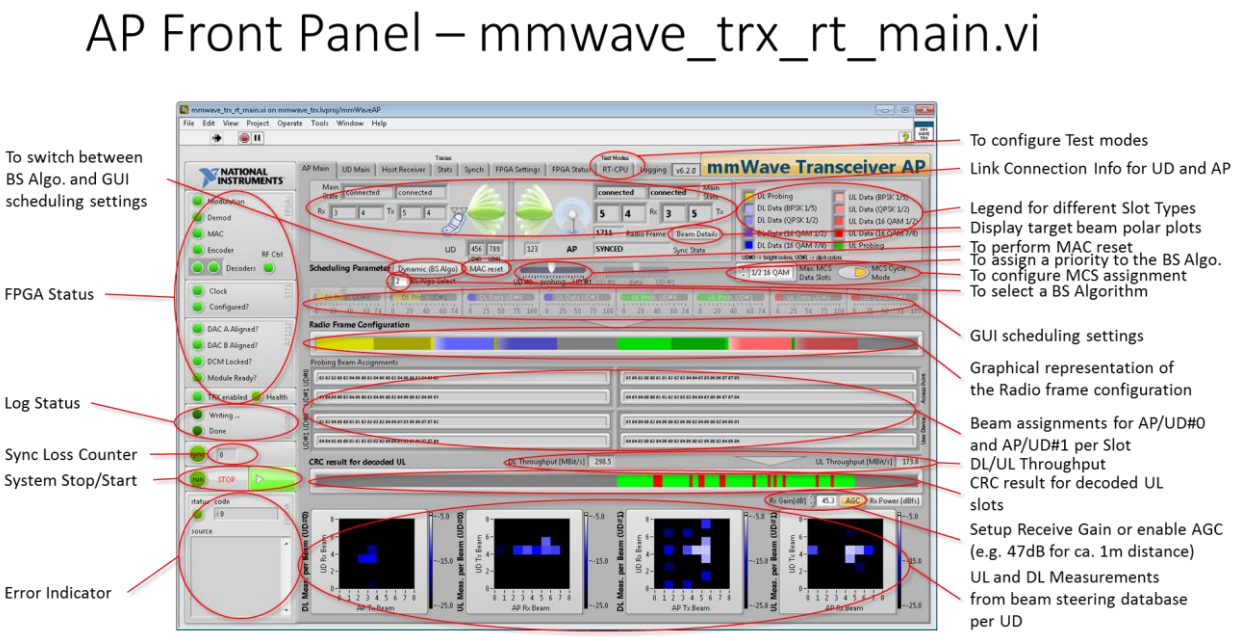
The same way that was reported in deliverable D4.1 [9] for the 60 GHz mmWave system, we have two possibilities available to setup an SDR control plane. They include (re)configuration and monitoring the capacity and status of the links and the network,

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

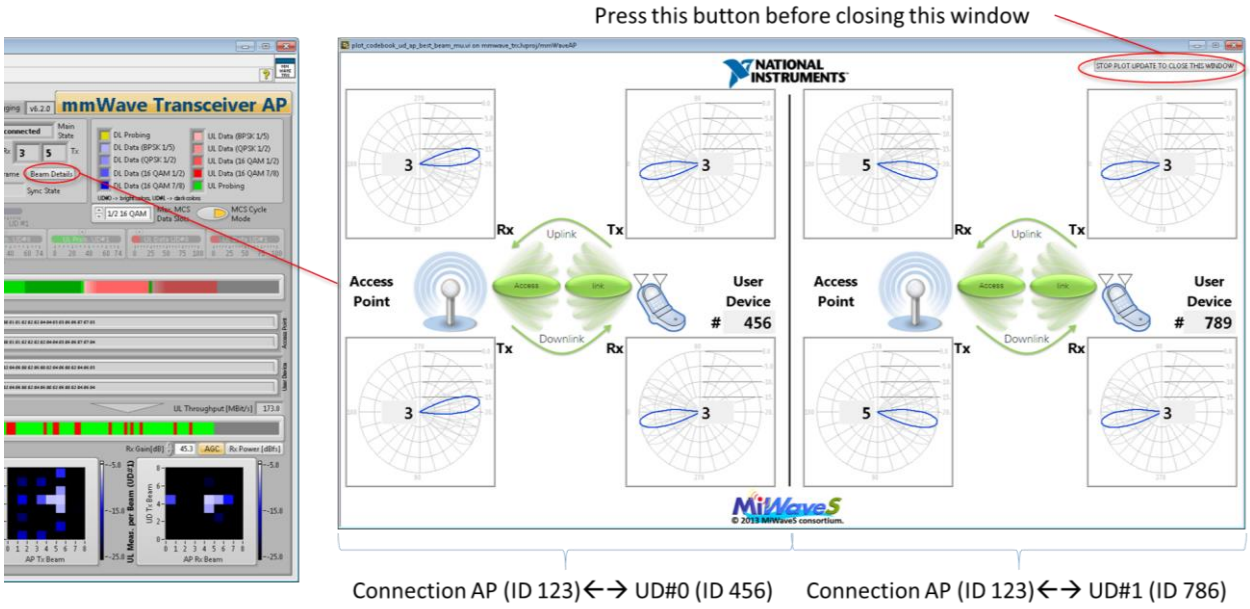
The GUI was developed by NI and is equipped with all functionalities of the 60 GHz mmWave. TUD created a config file which is able to control NI's GUI with a script, allowing a controllable environment for experimentation. The improvements on these real-time configuration methods are described in the following subsections.

3.1.1.1 Direct Control and Monitoring

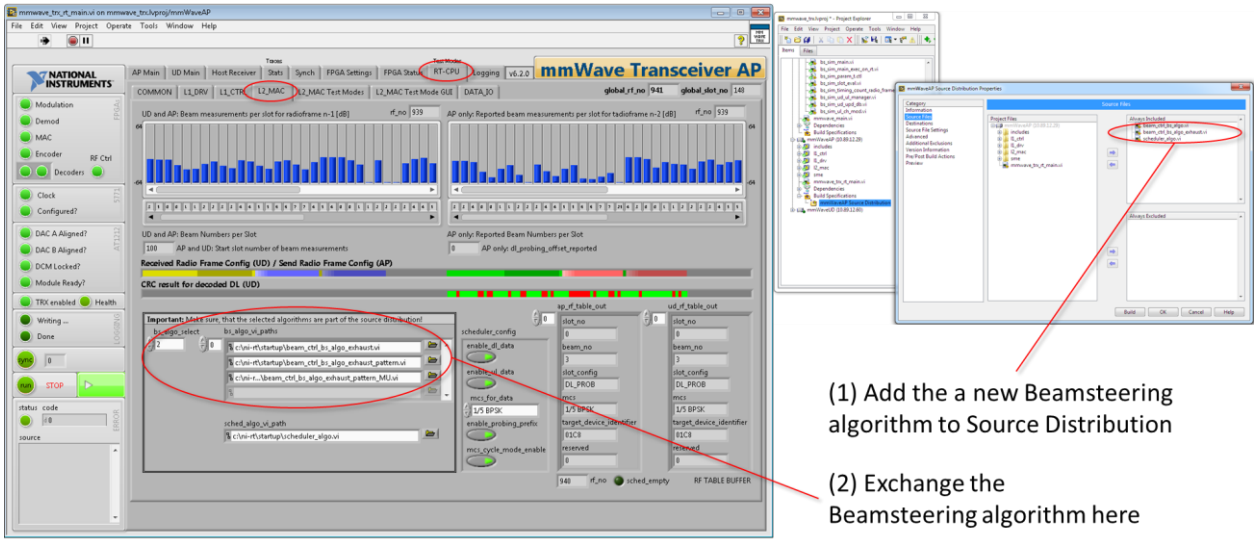
The direct control and monitoring of the real-time mmWave baseband system is possible through LabVIEW front panels as GUI. The front panels offering many different options for mmWave experiments. Subsequently the main front panels for Access Point and User Device are described. Further important panels which are related to beam steering, test modes and statistics are outlined. Front panel elements (Controls and Indicators) can be made available remotely through the testman interface described in Section 3.1.1.2.



AP Front Panel - Polar Plots for Target beams

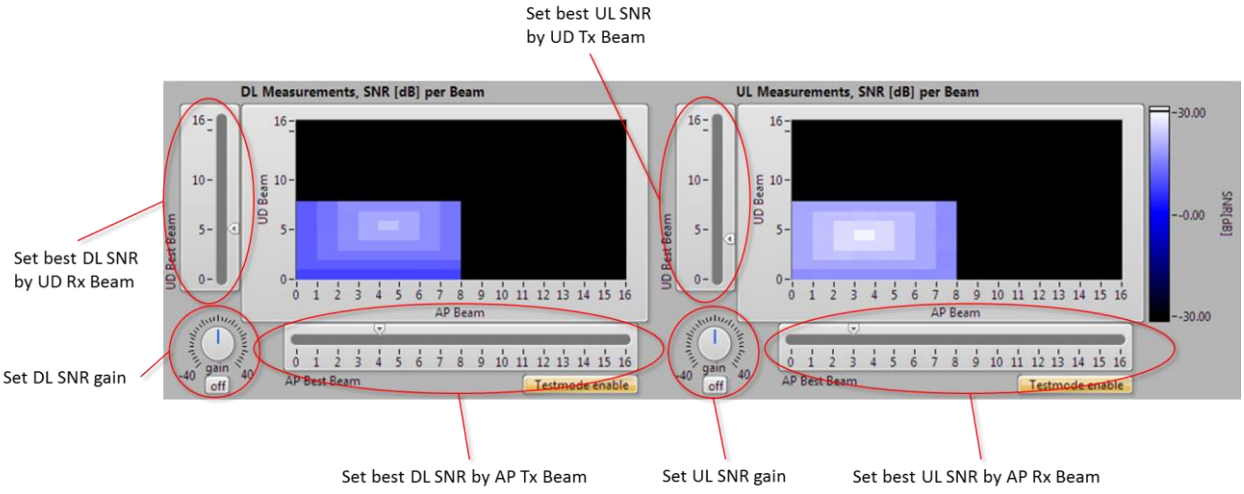


AP Front Panel (Tab RT-CPU → MAC)

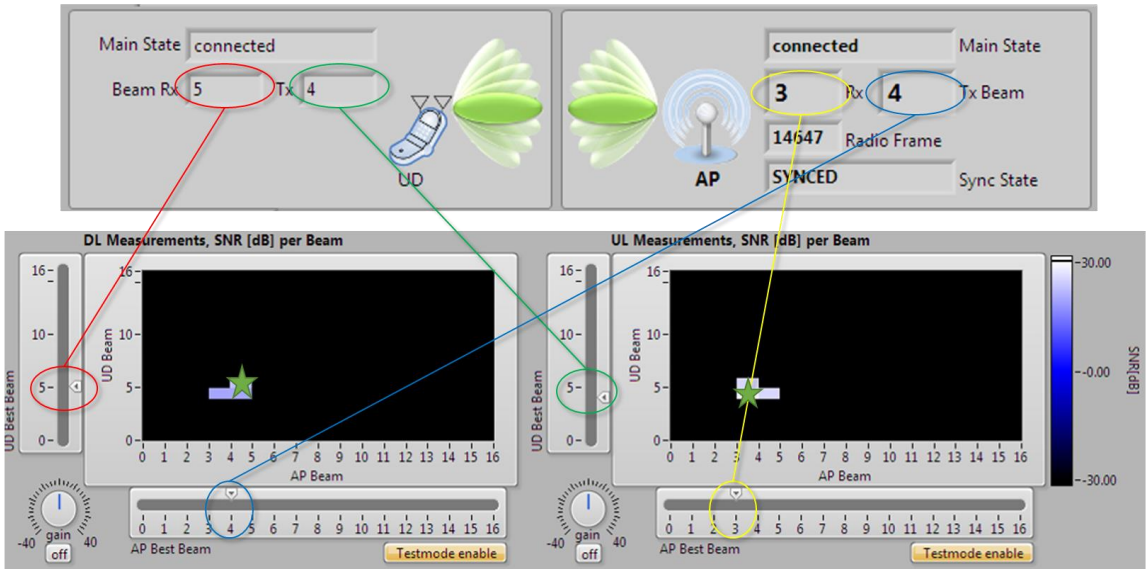


UD Front Panel – mmwave_trx_rt_main.vi

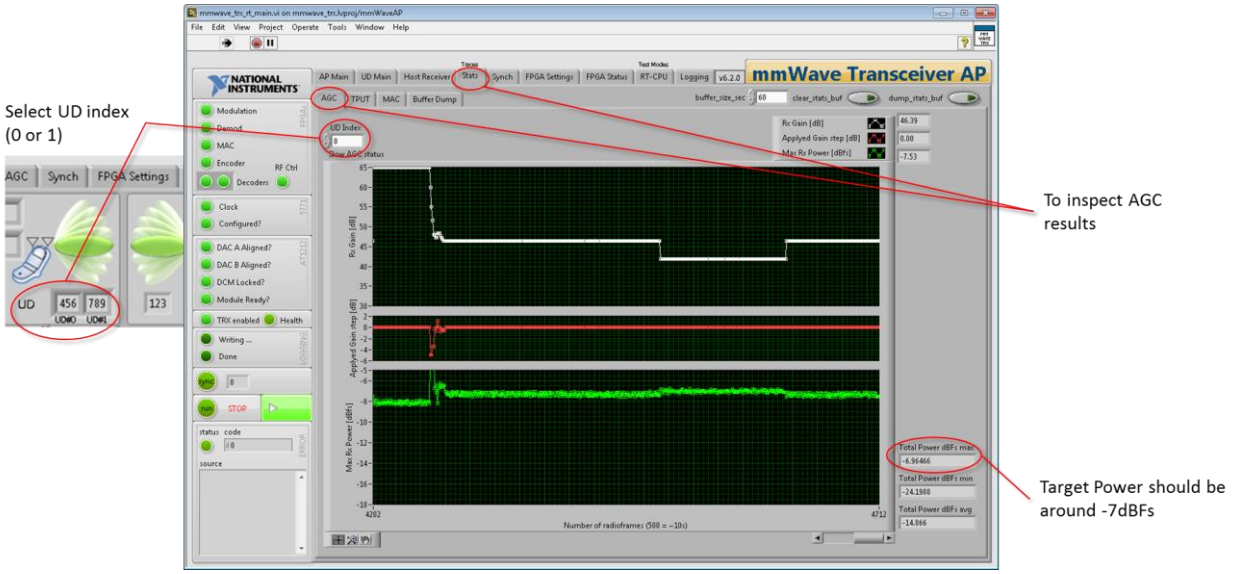
Test modes – Measurements manipulation



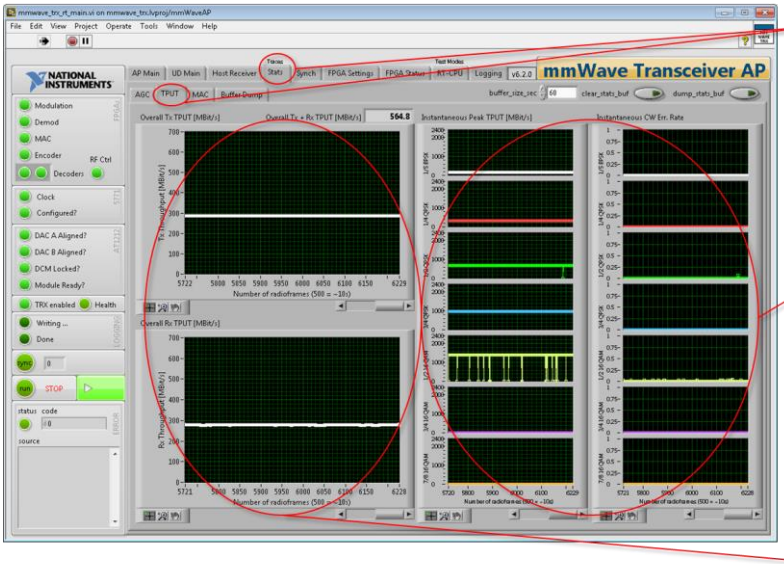
Test modes – Verification



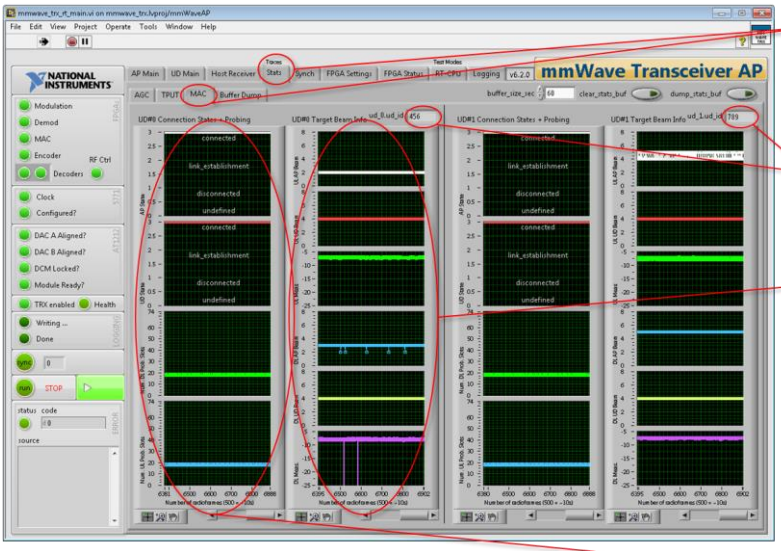
AP/UD Front Panel – mmwave_trx_rt_main.vi – AGC tab



Statistics (Stats) – AP – Throughput



Statistics (Stats) – AP – MAC info



Statistics (Stats) – AP – Buffer Dump

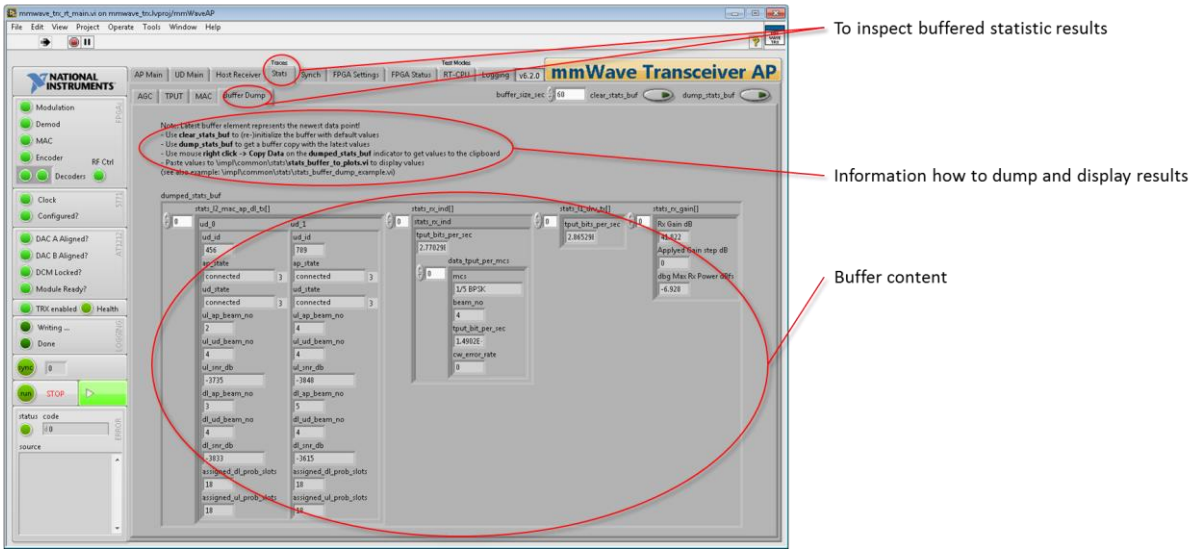


Figure 2: Collection of Frontpanel Views for Manual Control of parameters in the 60 GHz mmWave setup.

The configuration of automatic and manual beam steering functionality is also possible through these front panel interfaces. For automatic beam steering, several beam steering algorithms can be added to the system by defining the path to the algorithm VI see Figure 3. The control "BS Algo Select" which refers to the specified algorithms in "bs_algo_vi_paths", allows dynamic switching between the beam steering algorithms during runtime.

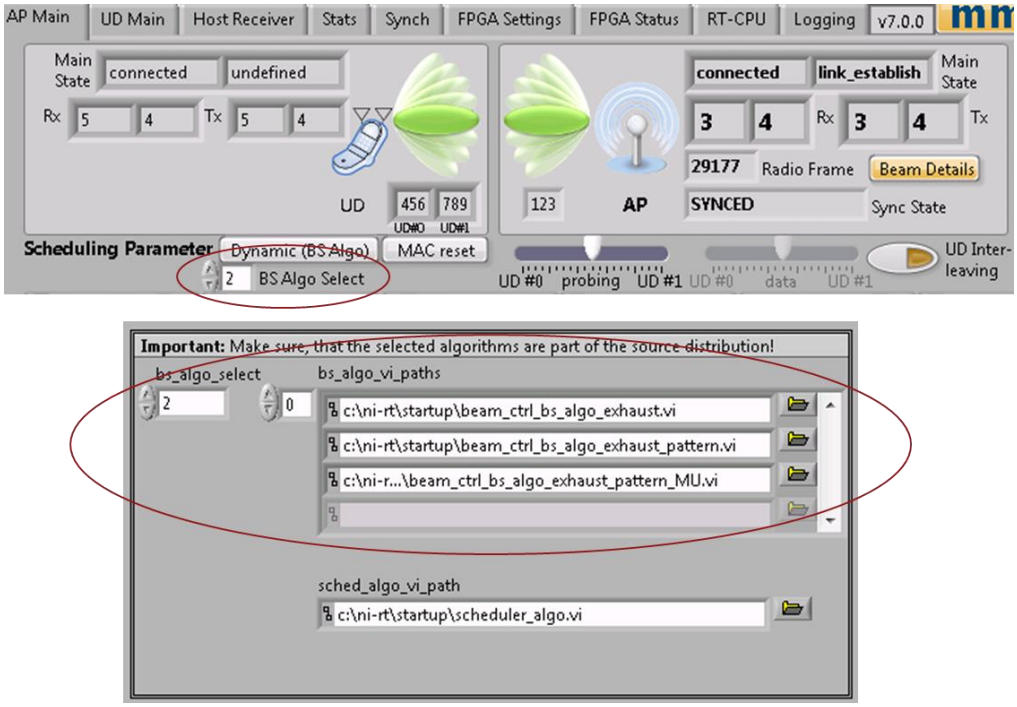


Figure 3: Configuration and selection of beam steering algorithms.

The configuration of manual beam steering can be done by pressing the enable button in control "manual_beam_settings", see Figure 4. Further the Tx and Rx Beam IDs for Access Point and User

Device (UD) can be selected here.

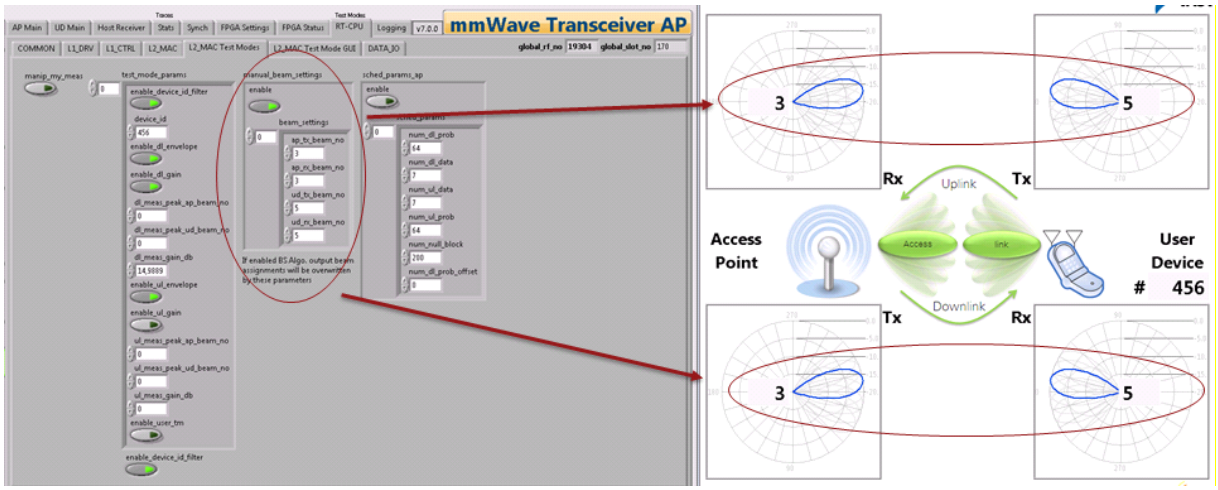


Figure 4: Manual beam steering configuration.

The real-time mmWave baseband allows different codebook configurations for the 60GHz Sibeam transceiver (see D3.1 [10] for details on the transceiver). The configuration can be done at setup time via the "Radio Init" Tab on the main front panel, see Figure 5.

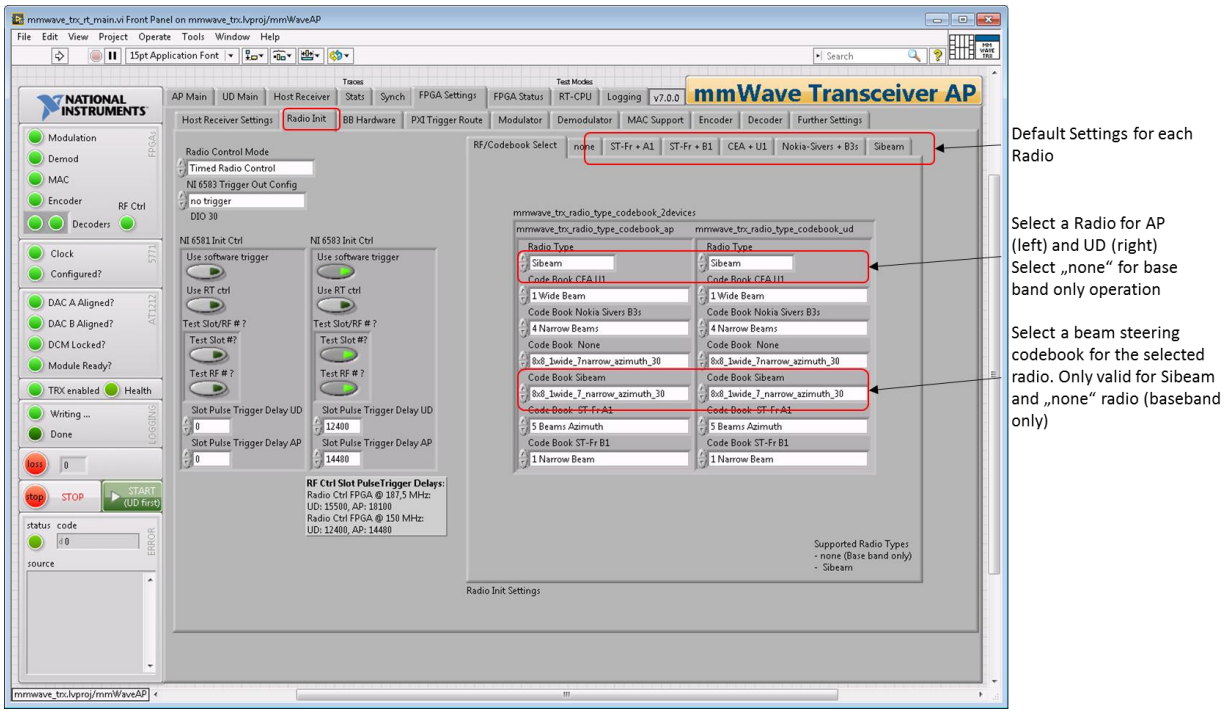


Figure 5: Codebook configuration.

3.1.1.2 Remote Control and Monitoring (TUD)

The same setup used to integrate NI's graphical control and monitoring interface to the testbed network described in deliverable D4.1 [9] was used in year 2. In short the TestMan plugin introduced in the EU-project CREW [11] in responsible for integrating different applications through a pre-defined communication protocol as shown in Figure 6.

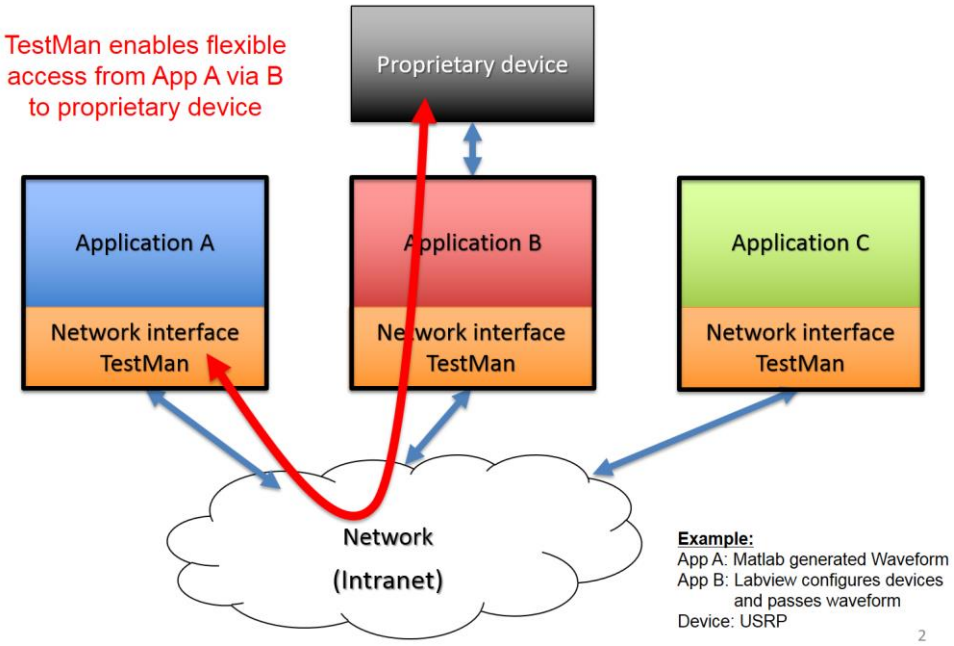


Figure 6: TestMan Concept.

The main achievement of TUD was to provide a complete configuration file that can access the main parameters of the Direct Control and Monitoring Interface. For instance, as a proof of concept, a Matlab script was written, allowing us to re-configure MCS, beam-steering algorithm, manual or automatic beam steering. In addition, the rotation table can also be triggered in order to emulate a mobility scenario. The rotation table is a structure where the antenna is mounted on, it means that we can modify the direction of the beams by moving this structure, emulating then a mobility scenario.

The example script can be seen in Figure 7, where the commands allow:

- Setting and reading MCS: BPSK 1/5, QPSK 1/2, 16 QAM 1/2 and 16 QAM 7/8
- Setting beam steering algorithm: gradient search or exhaustive search, these algorithms are described in Deliverable 6.5 of MiWaveS [12]
- Setting beam tracking: define the amount of beams for tracking, e.g., in the static case, there is no need to probe many beams; while in the mobility case, it is needed to probe more beams
- Moving antenna angle through rotation table: this command allows us to perform a controlled experiment under mobility or static scenario

```
%% Parametrize experiment

% set MSC
resp = send_commmand_mmWave('WRITE:AP:mcs:1/2 QPSK'); disp(resp);
resp = send_commmand_mmWave('READ:AP:mcs'); disp(resp);

% set BS algorithm : TRUE -> gradient, FALSE -> exhaustive
resp = send_commmand_mmWave('WRITE:PC:bs_algo:FALSE'); disp(resp);

% set tracking
resp = send_commmand_mmWave('WRITE:PC:tracking:False'); disp(resp);
resp = send_commmand_mmWave('WRITE:PC:tracking:True'); disp(resp);

% moving angle of user device antenna
angle = 0;
moveAntenna(angle); % synchronize moving apparatus
angle = 10;
moveAntenna(angle); % moving to "angle" degrees
```

Figure 7: Example of script to re-configure mmWave in real-time.

3.1.2 Radio slicing: resource allocation, instantiation and coordination

3.1.2.1 Beam Steering Simulation Environment

As outlined in D4.1 [9] the beam steering and scheduler algorithms are the key for allowing control and (re-)configuration of the radio slice. 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 are possible topics for research and open calls. A beam steering simulation environment, which was initial described in MiWaveS Deliverable D5.2 [13], where the relevant modules can be tested without the need of hardware, allows rapid prototyping of described functionalities.

In ORCA Year 2 this simulation environment was made available to the ORCA facilities. An overview block diagram is shown in Figure 8. The AP (access point) implementation 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. Further a DL Manager is executed which handles DL Control information and adds sideband information such as timing and beam schedule. Channel Measurements are stored in a Beam steering Database.

The UL/DL Channel and System Modelling blocks are modelling system delays and deriving received SNR values and received beam indices using a certain channel model. Here measurements of the Hardware in the Loop (HALO) setup described in D3.1 [10] and D4.1 [9] can be possibly applied by small adaptions of the simulation environment.

The UD implementation models the beam steering relevant parts for the user device such as DL channel measurements which are reported as UL feedback via an UL Manager. Message Encoders (MsgEnc) and Message Decoders (MsgDec) transferring the MAC control structures to/from a byte string back and forth. The main modules which are used in the simulation environment are the same as in the implementation of the real-time mmWave baseband. This allows realistic and comparable simulations.

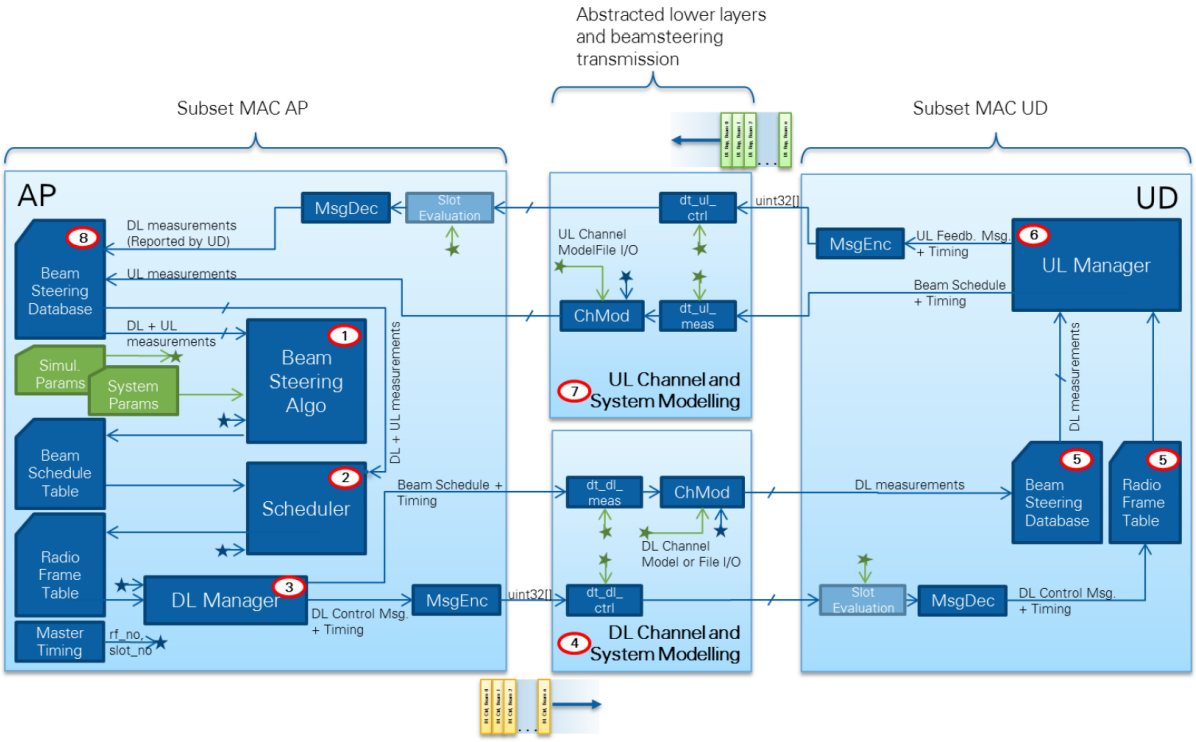


Figure 8: Beam steering simulation environment.

Figure 9 shows the implementation on LabVIEW. A graphical user interface to visualize execution and results is also provided and shown in Figure 10.

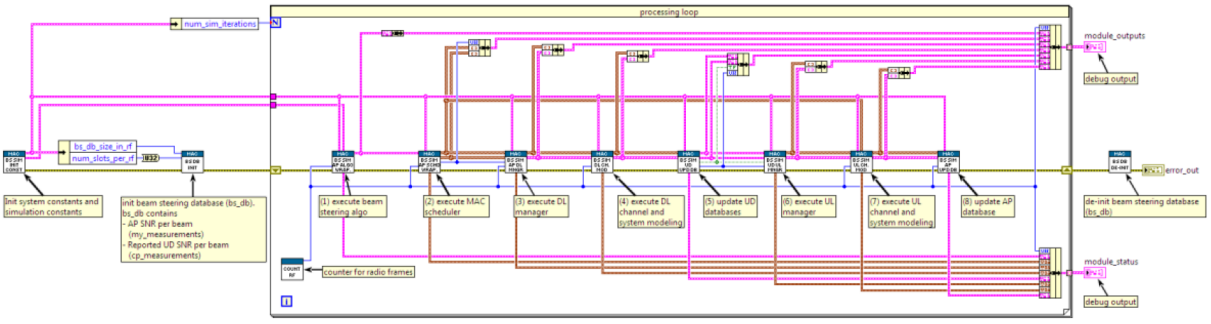


Figure 9: Complete Beam steering Simulation Environment in LabVIEW.

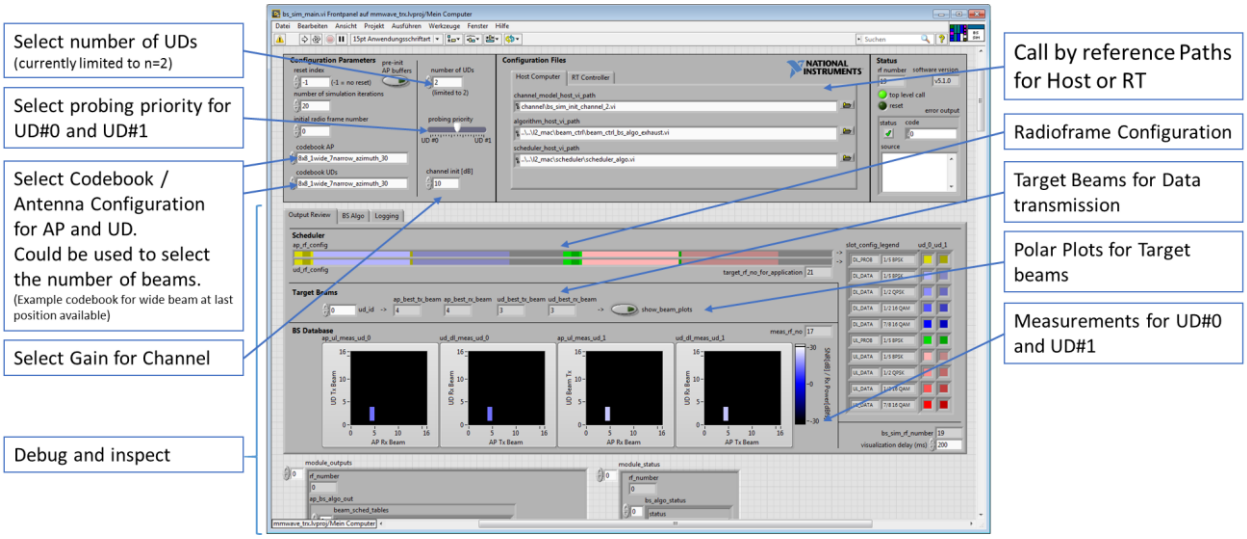


Figure 10: Beam steering Simulation Environment - User interface.

3.1.2.2 Real-Time Beam-Steering Reconfiguration

The real-time mmWave system integrated to TestMan allows relevant experiments where beam steering algorithms can be selected according to the channel behaviour. With the script shown in Figure 7, we set up an experiment where we manipulate 2 parameters:

- Beam tracking (ON/OFF)
- Antenna movement (ON/OFF)

The manipulation of these two parameters leads to the following combination

	Movement **OFF**	Movement **ON**
Beam tracking **off**	The system works fine.	System does not work because the link is too instable.
Beam tracking **on**	The system works but throughput is reduced.	System works fine.

Figure 11: Combinations of Beam Tracking (ON/OFF) and Antenna Movement (ON/OFF).

Using the NI's Control and Monitoring GUI, we can depict the throughput of the received bits at the base station, i.e., up-link. Figure 12 clearly shows that the reconfiguration capability of our mmWave setup leads to a better exploitation of the available resources. For instance, when the movement is turned off, i.e., there is no mobility, it is better to have a minimum set of beams for probing, i.e., beam tracking OFF, such that more slots are used for data, and then throughput is maximized. On the other case, where there are movements, the experiments reveal that beam-tracking is a must in order to keep the link stable.

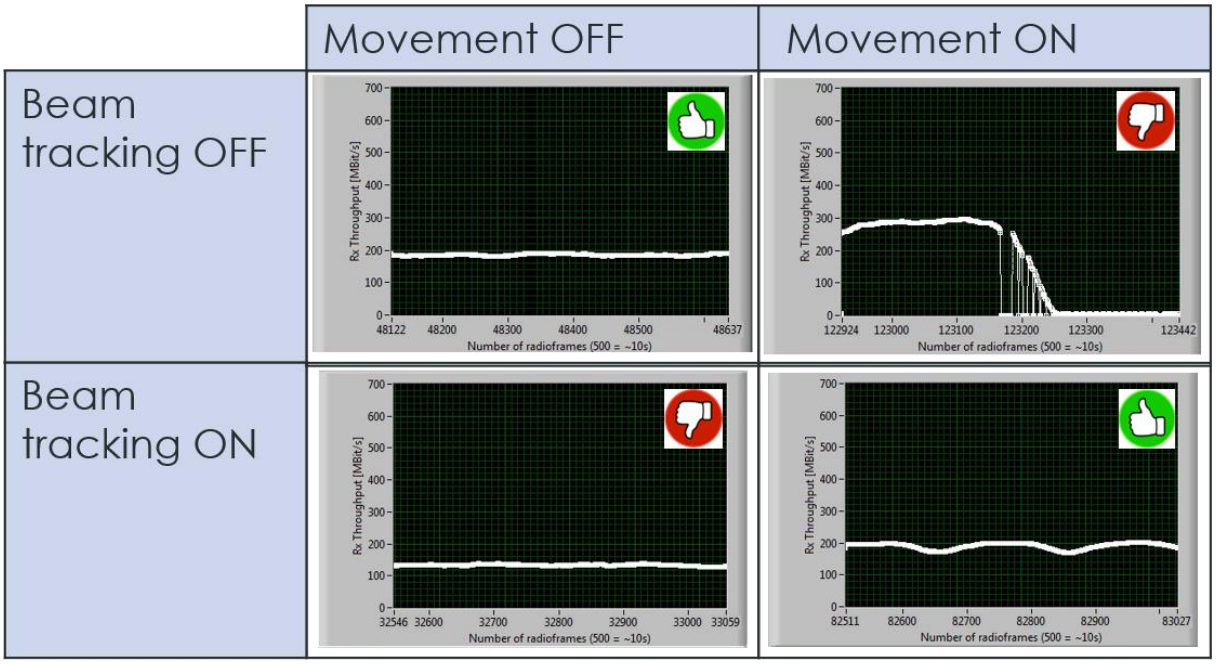


Figure 12: Up-link throughput measurement at base-station.

3.1.3 Testbed integration

These setups are available at the TUD's testbed and are accessible through remote desktop.

3.2 Relation to showcase

The results presented in Subsection 3.1 will be demonstrated in Showcase 1 and 4in the real-time system. The idea is that with this example, a potential experimenter can further work on the setup and design more sophisticated experiments, using the reconfiguration capabilities of the 60 GHz mmWave setup. It also demonstrates how the functionalities can be beneficial for future mmWave communication systems.

3.3 Risk analysis

There is no specific risk of this work package at this moment.

3.4 Implementation plan

For the new 26 GHz mmWave system, we will develop a simple MAC protocol over the GFDM PHY using a predefined TDD scheme to allow the nodes to exchange channel quality information. The MAC will also be used to perform the beam steering algorithm in both beam alignment and beam tracking phases.

The 60GHz mmWave system will not be further extended in Y3 as the 26 GHz mmWave system will be the focus for the final year.

3.4.1 Testbed integration

This system will be made available as part of the TUD testbed. The signal processing of the system is executed in the USRP-RIO SDR with a control PC, which are already part of TUD's testbed.

The focus for the 60 GHz mmWave system in Year 3 will be on the support of possible experimenters with the sophisticated hardware setup to ensure that the correct configuration parameters can be accessed and that experiments will run as planned.

The direct control and monitoring functionality is made available on the front panels of the respective 60GHz mmWave configuration and monitoring program and therefore is accessible for experimenters within the testbed.

4 SDRS WITH IN-BAND FULL DUPLEX CAPABILITIES AND COLLISION DETECTION (KUL)

4.1 End-to-end capable in-band full duplex SDR

This section mainly explains how the achievements in ORCA Y2 have enabled an operational real-time in-band full duplex (IBFD) SDR that can be used for end-to-end low-latency networking. Figure 13 depicts the modular representation of the KUL IBFD platform which is deployed on a NI USRP [14].

In the first year of the project, ORCA presented a communication device equipped with an electrical balance duplexer (EBD) to perform analog self-interference cancelation (AnSIC). This cancelation stage is essential to enable further TX-RX isolation by a complementary real-time digital self-interference cancelation (DiSIC) module. The implemented platform utilizes the improved cross-layer adaptable wireless system (CLAWS) architecture [15] which significantly enhances the real-time performance of the MAC and PHY layers, appropriating them for time-critical messaging in a real-life end-to-end scenario. In this design, the DiSIC, the collision detector block, and the half/full duplex PHY are realized on a Kintex 7 FPGA. Besides, two MicroBlaze softcores are integrated into the FPGA in order to deliver the high-level MAC protocol and the EBD tuning operation. The whole system is controlled by a host PC which not only can accomplish the upper layers tasks but also provides debug and development tools to facilitate network level experimentation.

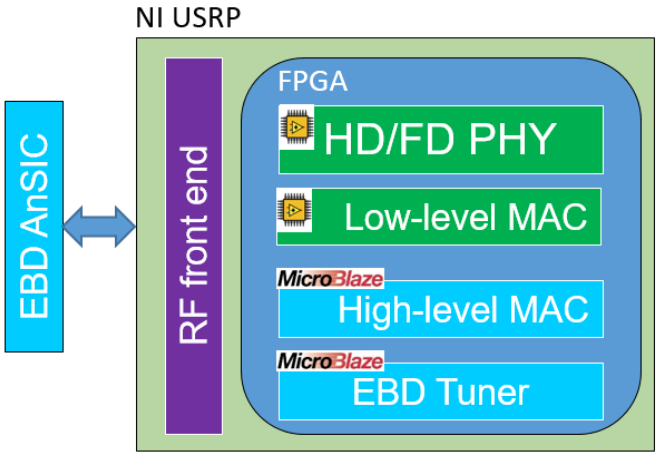


Figure 13: Overview of the KU Leuven in-band full duplex SDR with end-to-end capability.

4.1.1 PHY improvements

Figure 14 illustrates the comprising PHY elements in KUL IBFD platform. The PHY controller plays a crucial role as it interfaces the PHY modules with the MAC layer on one side and provides direct access to the PHY signals from the host PC to facilitate debug and development of real-time algorithms.

It has to be noted that the DiSIC block is implemented by Tampere University of Technology in the framework of ORCA first open call 1 for extension (EXT4).

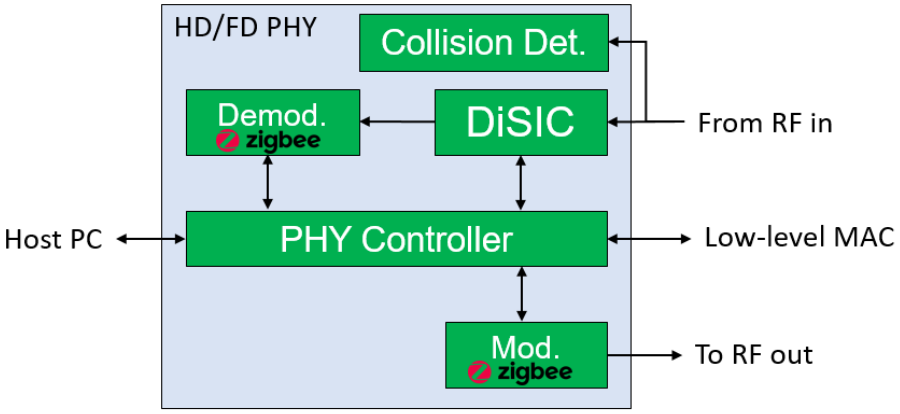


Figure 14: Representation of the implemented IBFD PHY with collision detection capability.

4.1.2 MAC layer improvement

Figure 15 indicates the improved MAC layer in ORCA IBFD capable SDR. The optimized MAC benefits from a bi-layer architecture to achieve optimum latency performance and reliability enhancement which are crucial in a low-latency end-to-end networking scenario..

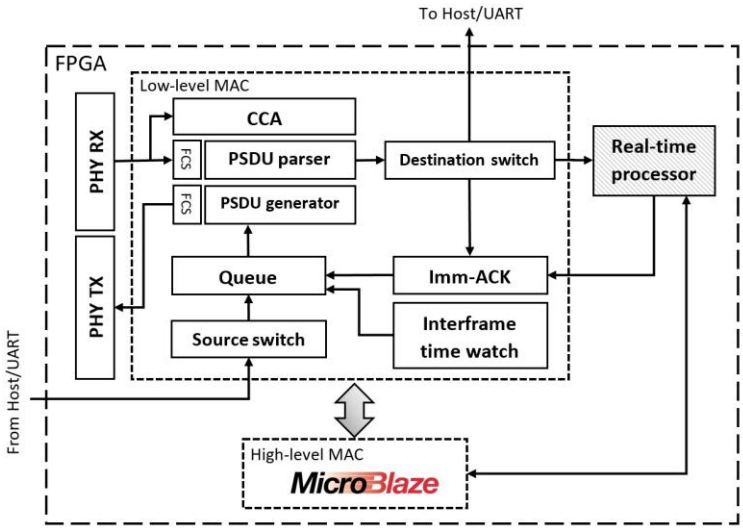


Figure 15: Architecture of the implemented bi-level MAC.

In this structure, the high-level MAC on the MicroBlaze can initiate packet transmission process, run retransmission algorithm, drive the backoff timer, trigger clear channel assessment (CCA) module, configure the payload sources and destination switches, e.g., UART, host PC.

The hardware realized low-level MAC reduces the workload of the MicroBlaze as it performs somewhat non-sequential straightforward jobs of the MAC protocol and reacts to interrupts instantly. For instance, it allows spontaneous Imm-ACK packet transmission. As shown in Figure 15, the low-level MAC is elaborately coupled with the related sections in PHY on one side and interfaced with the high-level MAC on the other.

Generally, the following functionalities can be accomplished by the presented low-level MAC; generating the PHY service data unit (PSDU) and Imm-ACK packet, parsing the PSDU, controlling the packet queue, and measuring the interframe spacing times. This set of components is implemented by LabVIEW FPGA, running at 150 MHz and needs slight level of flexibility.

4.2 Implementation results

4.2.1 PHY layer improvement results

4.2.1.1 EBD tuning enhancement; dithered linear search

The EBD self-interference rejection performance relies on balancing two impedances: a) the impedance of the antenna and b) the dummy impedance network. Since the antenna impedance varies with the dynamics in the environment, a tuning mechanism is needed to adaptively track the environmental changes. To tune the EBD, the IBFD SDR uses the particle swarm optimizer (PSO) implemented in a software module (C++ function). Utilizing this technique, the EBD can be tuned within 1ms [16]. Since the iterative procedure of EBD adaptation imposes distortion to its receiver port, the tuning time has to be minimized to appropriate the EBD for a realistic end-to-end scenario. To this end, an optimized dithered linear search (DLS) algorithm is implemented in the form of a software module. In this implementation, Hadamard sequences are used to search a 4-D solution space. The experimental result shows significant tuning improvement in the sense of speed and resource consumption. Table 1 compares these two tuning techniques.

Table 1: Comparison of particle swarm and dithered linear search EBD adaptation techniques.

Tuning software module	Required memory (32bit-floating point)	Tuning time (ms)
PSO	3200	~1
DLS	20	0.124

To further investigate the performance of DLS-based EBD adaptation, we employed an XY table to move a reflector in 10-50 cm from the antenna. We also defined a threshold of -45 dBm to control the power of residual self-interference. Figure 16 illustrates the measured probability distribution of the remaining self-interference and the tuning time.

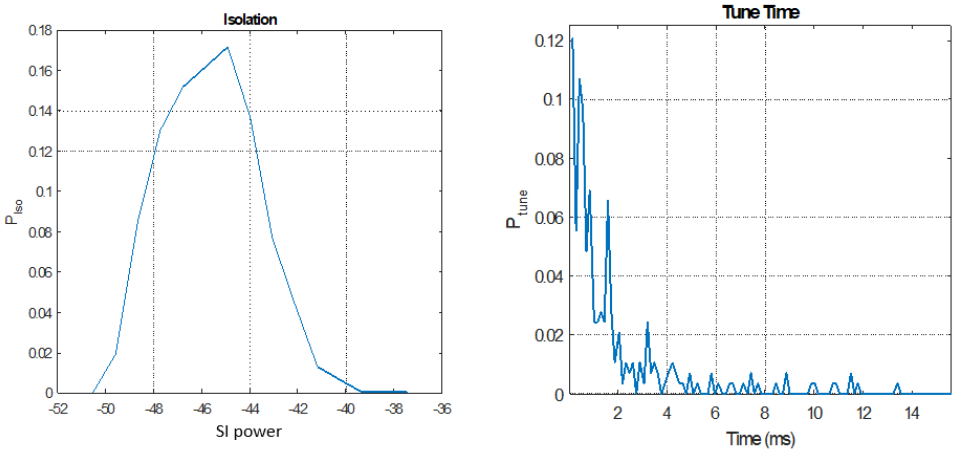


Figure 16: (Left) measured distribution of the residual self-interference power and (Right) tuning time, when the DLS EDB adaptation is triggered by a predefined threshold (-45dBm).

As shown in this figures, the DLS optimizer required 0,124 ms (45 μs for each iteration) to adapt the EBD and maintain the Tx-Rx isolation. This procedure can be done during preamble transmission.

The required tuning rate is also investigated in an indoor scenario. Figure 17 displays the probability distribution of the EBD isolation, measured in a typical office room where the device is placed on a

desk next to a working person. In each experiment, an isolation threshold is determined to trigger the EBD tunning algorithm. The level of the SI and the number of triggers are recorded for 60 minutes to estimate how often the EBD has to be tuned to maintain the Tx-Rx isolation below a certain threshold. It is evident that increasing the isolation threshold raises the required EBD tuning time.

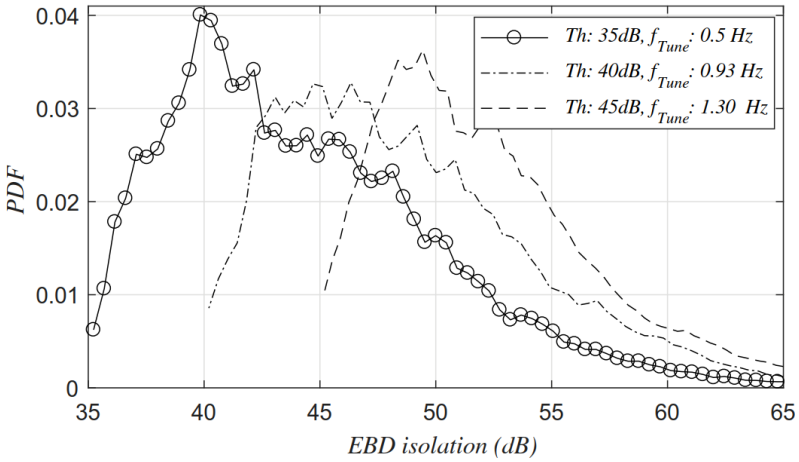


Figure 17: Probability distribution of the EBD isolation performance for various tuning thresholds, measured in an indoor environment.

According to the experimental results in this figure, to obtain 45 dB, 40 dB and 35 dB analog SI cancellation, the EBD requires respectively 1.3 Hz, 0.93 Hz and 0.5 Hz tuning rates.

4.2.1.2 Full duplex functionality

Figure 18 shows the measured results of the real-time self-interference rejection achieved jointly by the AnSIC and DiSIC blocks. This test carried out using a Zigbee-like waveform and a 2-node network. As shown in this figure, the IBFD device can significantly attenuate the self-interference signal to enable reception of the desired signal transmitted from a second party transmitter. More detailed measurement results are presented in [D7.2].

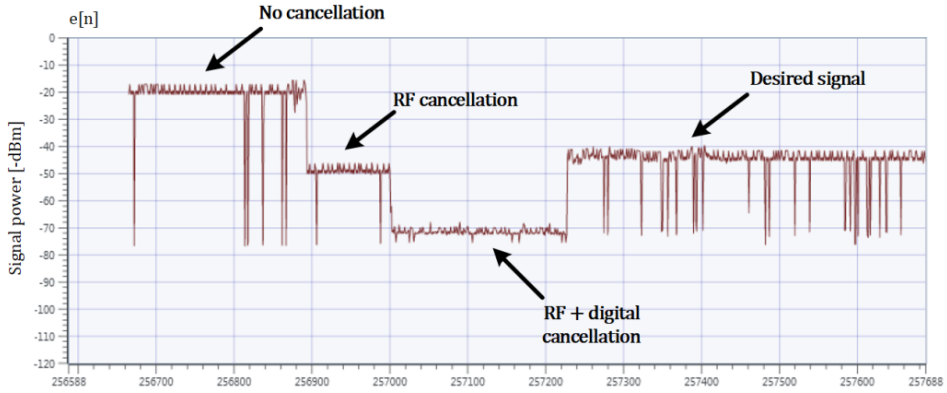


Figure 18: The measured power of self-interference signal with/without self-interference cancelation [D7.2].

To investigate end-to-end capability and evaluate the implemented DiSIC module, we measured the rate of the successfully transmitted packet (120 bytes including the headers and the payload) in a 2-node network. Table 2 lists the measured result in half/full-duplex mode.

Table 2: Measured throughput performance of the implemented full duplex PHY in a 2-node network.

Mode	Successfully transmitted packets	Packet loss rate
Half duplex	75 packet per second	1%
Bi-directional (full duplex)	135 packet per second	5%

4.2.2 MAC layer improvement results

To compare the performance of the implemented two-layer MAC and the fully CPU-deployed MAC, we ran both structures at 150MHz and interrupted them to produce 12-Byte immediate acknowledge packet (Imm-ACK). As listed in Table 3, depending on its working state, the single-layer MAC needs 10 µs to generate and place the Imm-ACK packet in the queue. However, this time may exceed up to 16 µs regarding the working status of the CPU. Whereas, the bi-layer MAC behaves deterministically as it generates and inserts the packet into the queue within 0.41 µs. Due to its efficient implementation hence, the low-level MAC can take up the time-sensitive workload of the MAC layer and also enable reliable networking.

Besides, the realized low-level MAC fits in < 2% of the Kintex7 FPGA while the MicroBlaze occupies 8% of the total logic resources. Therefore, a combination of multiple low-level MACs on top of one CPU module can offer a practical and efficient solution for multi-channel and full-duplex MAC protocols as it provides adequate processing power and multiple interrupt handling ability.

Table 3: Comparison of single layer and bi-layer MAC.

	One layer MAC	Two-layer MAC
Time to generate ACK packet and pass it to the packet queue	10-16 µs	0.41 µs
FPGA resource consumption	8%	< (8.2)%
Packet loss rate (4-node scenario)	3.5%	< 0.4%

4.3 Relation to showcase

In addition to the improved MAC architecture, the full duplex functionality can remarkably enhance the networking reliability as well as round-trip latency. Therefore, this capability is related to Showcase 2 where reliable low-latency end-to-end communication is needed. Besides, in WP5, we study how to enable full duplex operation dynamically based on instantaneous energy efficiency, reliability or latency requirements.

4.4 Testbed integration

On top of the real-time PHY presented in ORCA Y1, the full duplex functionality and the enhanced CLAWS platform are already integrated to the KU Leuven IBFD testbed.

4.5 Risk analysis

- The EBD adaptation rate can influence the performance of the DiSIC. More measurement is needed to investigate this risk.
- Although the non-linearity of the RF head is modeled by the DiSIC module, still some device-dependent parameters, e.g., Tx baseband to Rx baseband delay, are assumed to be constant. Since these parameters greatly influence the SI cancelation, a closed loop mechanize might be necessary to maintain optimum performance.
- The DLS-based EBD tuning relies on the length of the employed Hadamard sequence. While a short sequence fails to converge the adaptation algorithm, the experimental result shows a long one increases the adaptation time. Therefore, optimal length has to be determined according to the number of adaptable coefficients and the size of the search space.

4.6 Implementation plan

For ORCA Y3, we are planning to improve and synchronize the adaptation of AnSIC and DiSIC modules. It is necessary as the performance of the digital canceller is immediately affected by the EBD. This task has to be handled by the MAC layer as it controls transmission and is aware of data reception.

5 FLEXIBLE PHYSICAL LAYER BASED ON GFDM (TUD)

The main focus of the PHY implementation in the year 2 of ORCA is described in deliverable 3.3 with goal in providing a stable and robust communication system. The reported latencies of around 300 µs for an external application using UDP packets to be transmitted over the air leaves enough time to add multi user capabilities to the transceiver while keeping the 1 ms criteria for 5G-similar networks. Further, the implementation has been ported to new hardware platforms to ensure enough processing capabilities to add low latency MAC protocols.

5.1 Implementation results

Two new host platforms are introduced and working with the GFDM transceiver implementation. The first one is the Hewlett Packard Enterprise Edgeline series and the second one is the USRP RIO 2974R from NI. The first platform is intended for edge cloud computing and provides up to 4 computing blades to be used in one edgeline 4000 chassis. In addition, up to four PXI slots for USRPs or FPGAs are available which are shared between the blades. In the testbed several m710 blades with an 8 core Intel Xeon E3-1284L @ 3.2 GHz and 64 GB RAM each are available. Windows server 2012 R2 is used as operation system.

The second platform is the USRP RIO 2974R which has a powerful Intel i7 processor attached to the FPGA based SDR for MAC developments. In contrast to the edgeline systems, a real-time Linux is used as operational system to ensure deterministic execution of MAC protocols.

5.1.1 SDR control plane improvements

In preparation of the integration with higher network layers, a bus system is designed to provide a unified interface for all PHY signal-processing blocks. The main reason is that each signal processing block has a set of different controls and parameters which have to applied cycle accurate to ensure proper functioning of the block. However, changing waveforms requires a good understanding of the individual signal processing blocks. Thus, the bus system presented in Figure 19 abstracts the parameter sets to a single message structure.

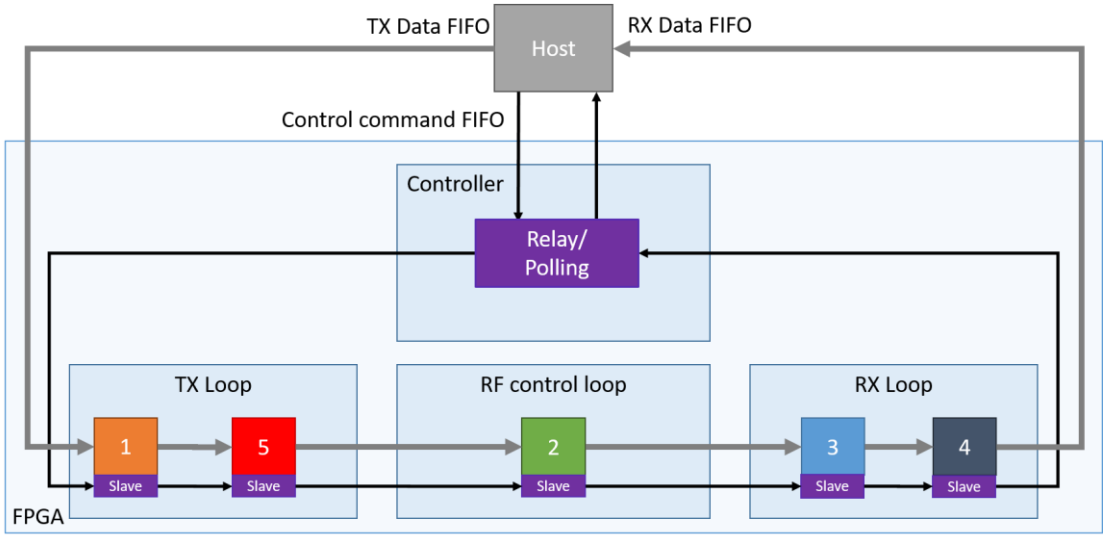


Figure 19: Bus system overview.

Table 4 shows the structure of the messages exchanged along the bus. All the signal processing modules are daisy chained such that the message is passed from on module to the next similar to the SPI bus used in microcontroller architectures. A FPGA controller module handles incoming messages from the host and relays messages back to the host. In case no commands are to be obey, it will trigger each module with dummy packets to return status information in a round robin fashion. Data packets are not passed along the bus lines to keep it free from congestion and are handled separate.

Table 4: Bus message structure.

Unsigned integer 64 word						
2 bit	7 bit	2 bit	4 bit	1 bit	16 bit	32 bit
Unit ID	Module ID	Message Type	Command	Lower / Upper word	Register address	Content
0 – All **1 – TX** **2 – RX** **3 – Other**	0 – reserved to address all modules	0 - Message from Host 1 - Message from FPGA 2 – Message to other submodule	See Table 5	In case 64 bit words have to be sent: 0 – lower 32 bit content 1 – higher 32 bit content	User defined, e.g. could be used as memory address	User defined

Table 5 lists the commands available to be applied on the slave modules of each signal-processing block. They cover all programming or controlling functions needed to operate the PHY. Further, they can be used to obtain status information, including for example complete channel impulse responses to higher layer instances.

Table 5: Bus commands.

ID	Command	Comment
0	Dummy	Slot for the respective module to reply with e.g. status information
1	Reset	Module should reset and clear all buffers
2	Set state	Defines the state in content, e.g. 0 – idle, 1 – configuration, …
3	Get state	*Reserved for future use*
4	Write register	Write a specific register defined in register address with the content
5	Read register	Read a specific register defined in the register address and put it on the bus - *Reserved for future use*
6	Set memory channel	Select the output memory with register address
7	Write memory	Write content into the memory to register address
8	Read memory	*Reserved for future use*
9	Debug level	Set debug/verbose level of the module with content
10	RAW data	[content] amount of u64 samples will follow after this packet. [register address] to define the output channel. Will turn of operation of the bus system and stall the bus.
11	Data	For data transfers without disabling the bus-system.
12	*Not defined.*	
13	*Not defined.*	
14	Command to controller	Specialized commands to the FPGA bus controller.
15	Special command	Defined in register address using content for additional information/variables

The goal of the bus system is to expose a very fine control of the PHY signal processing for external applications. Therefore, the host part of the bus system is connected to the network via network protocols like TestMan. A connection to the Wishful UPI or ZeroMQ is possible. Finally, pipelining for the presented bus system will be studied to further decrease the reconfiguration phases between two different waveform configurations during year 3 of ORCA.

5.1.2 MAC Development

The MAC developments are based on the described bus infrastructure, but keeping control information and payload separated. This subsection covers two types of flexible MAC. The first is based on TDMA-like WiFi protocols. The other is FDMA similar to the LTE protocol. Both designs use the flexibility of GFDM to provide more degrees of freedom such that not only the modulation-coding scheme can vary to adapt to different channel or application conditions, but instead the whole waveform itself.

5.1.3 TDMA MAC

Figure 20 drafts a communication system with three users using three different waveform configurations depending on the application requirements. The services are scheduled in time domain using one access point with a flexible GFDM transceiver implementation.

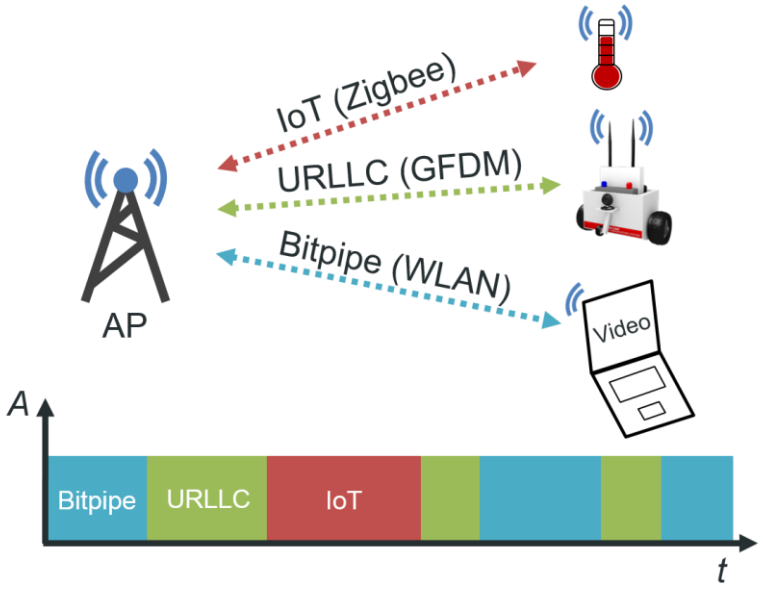


Figure 20: Evolved access point with SDN functionalities offers various services using a reconfigurable and flexible PHY.

The proposed PHY frame structure for this scenario is shown in Figure 21. The structure is similar to standardized systems such as WiFi and Zigbee but adds the freedom to choose the waveform as well.

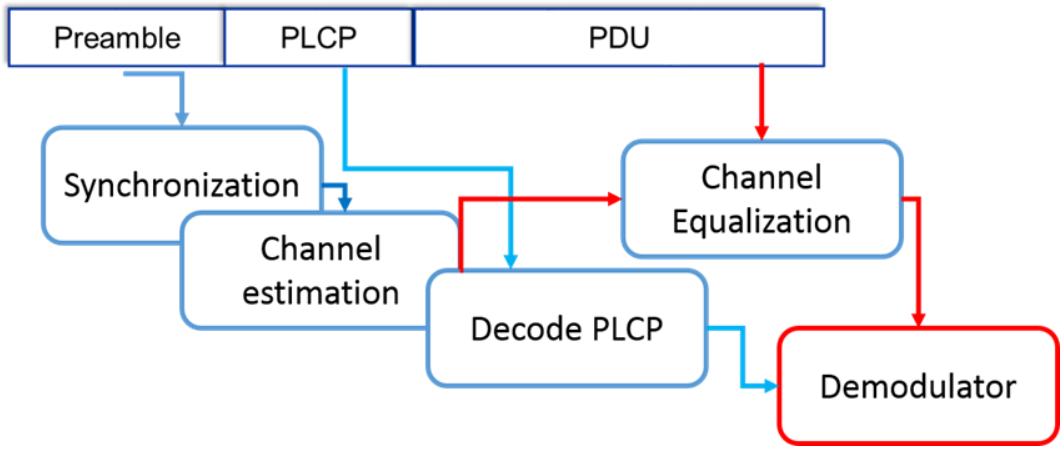


Figure 21: PHY Frame Structure.

The radio frame consist of three components:

- **Preamble**: is a reference signal for synchronization and channel estimation.
- **Physical Layer Control Protocol (PLCP)**: contains a modulated signal with predefined PHY parameters. The data within the PLCP define the PHY parameters used for the payload. Field parameters of the PLCP header are summarized in Table 6.
- **Protocol Data Unit (PDU)**: is the signal containing the MAC layer data. The structure of the PDU file is shown in Figure 22

Further, Figure 21 illustrates how and which order the different information are used. After synchronization, the preamble is forwarded to the channel estimation. Using equalized the PLCP header can be decoded to know with which waveform configuration the PDU is encoded. Finally, the demodulator can resolve the transmitted payload.

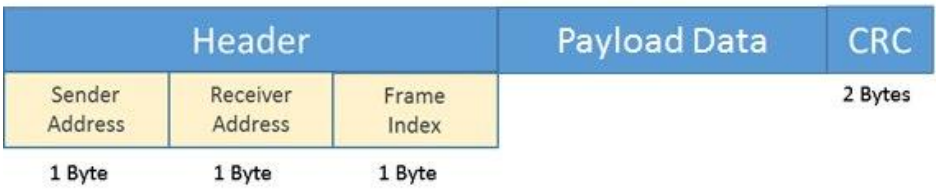


Figure 22: PDU Structure.

To differentiate the waveforms, the PLCP contains the following information:

1. **Coding and modulation scheme**: The QAM modulation is of length 2, 4, 16, 64 and the coding has the possible rates 1/2, 2/3, 3/4, 5/6. We have implemented a fast and flexible QAM mapping and demapping on FPGA.
2. **{K, M} Parameter selection**: The number of subcarriers is selected from the quantity K = {8, 16, 32, 64, 128} and the number of subsymbols from the quantity M = {4, 8, 16} if a Radix 2 modem is used, or the set M = {3, 9, 15} in the case of Non-Radix 2.
3. **Active subcarriers**: the set of active subcarriers divides the available belt into 8 subcarriers, the selected subcarriers are assigned to the set K_{on}.
4. **Number of GFDM blocks**: The number of blocks within a frame is calculated based on the payload length.
5. **Number of fill bits**: Number of additional non-information bits that complete the last block in the frame. It is removed after the decoder.
6. **Cyclic prefix length**: This option offers 9 possible options for different CP lengths depending on the channel.
7. **GFDM Prototype Pulse shape**: 4 possible options for pulse shaping, including OFDM, GFDM, DFT-Spread-OFDM and Chirp waveform.
8. **Cyclic redundancy check of the PLCP header**: 3 bits of cyclic redundancy check (CRC) parity check for the header. Flexible CRC is implemented on FPGA and can be re-used for fast calculation of the frame CRC.

Table 6: Field parameters of the PLCP header.

Filed name	Length [bit]	Function
MCS	4	Coding and modulation scheme
(*K*, *M*)	4	*K*, *M* Parameter selection
K_{on}	8	Aktive subcarriers
NB	8	Number of GFDM blocks
NP	8	Number of filling bits

CPL	3	Cyclic prefix length
PS	2	GFDM prototype pulse shape
CRC	3	Cyclic redundancy check of the PLCP header

For applications with low latency, a polling approach enables deterministic multiple access. With this technique, the access point prompts users to transfer their data using round robin scheduling. The proposed MAC can differentiate up to 254 users.

5.1.4 FDMA MAC

In conventional Generalized Frequency Division Multiple Access (GFDMA), as shown in Figure 23, the assignment is done via the symbol assignment matrix ***D***. In the downlink, each user must demodulate the entire signal and make the assigned data symbol available to the demodulator, which requires the use of the same waveform parameters for all users. In addition, the protection band is set between users with a subcarrier resolution. This approach implies unnecessary calculations, channel estimation is complicated, and spectral efficiency is reduced when the distance between subcarriers is large. To remove this limitation, we propose a flexible GFDAM system. In flexible GFDMA, the available bandwidth is divided into subchannels that can be used by multiple users can be assigned. Each user creates a GFDM-based signal in the frequency domain that fits into the assigned subchannel, as shown in Figure 24.

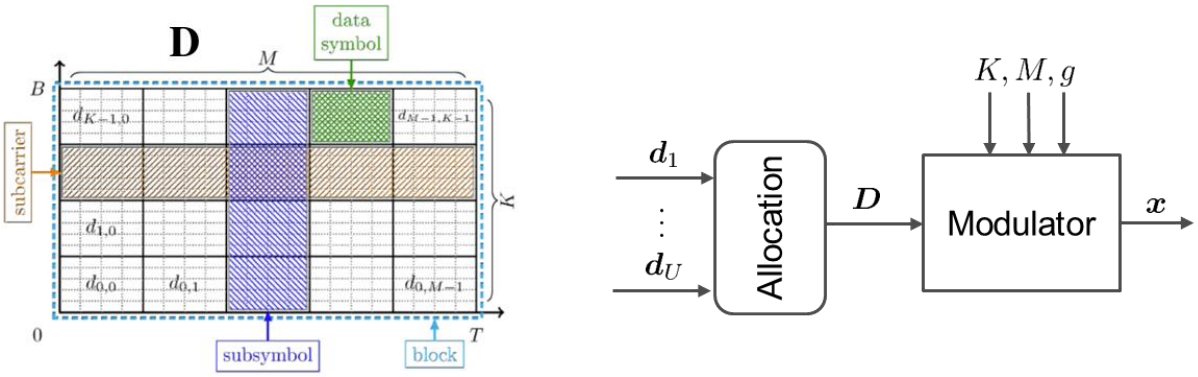


Figure 23 Conventional GFDMA.

Consider a resource grid of multiple OFDM symbols. Each user is assigned a set of subcarriers N_u and a set of OFDM symbols T_u. The set $N_u \times T_u$ contains overall allocated samples indexes, which are further split into a set of data samples $N_u^{(d)} \times T_u^{(d)}$ and a set of pilot samples $N_u^{(p)} \times T_u^{(p)}$. The data samples $\tilde{\boldsymbol{x}}_u$ corresponds to the sample of a FD-GFDM block, which is generated from the data symbols $\mathbf{d}_u$. Thus, $\tilde{\boldsymbol{x}}_u$ represents GFDM-precoded data. The allocation function maps the samples of the GFDM block and the pilot samples $\mathbf{p}_u$ to the resource grid. Actually, this scheme is an extension of OFDMA and SC-FDMA. For OFDMA configuration, the FD-GFDM modulator is bypassed. Moreover, when the FD-GFDM modulator is configured with rectangular FD pulse shape, this scheme produces SC-FDMA. Furthermore, the size of the OFDM symbol can be adjusted via the configuration of the final IDFT block to produce different subcarrier spacing for compatibility with 5G NR. As a result, GFDM precoding can be integrated within an existing OFDM-based system, where all the OFDM-based processing such as synchronization and channel estimation can be reused. The GFDM precoding at the transmitter side is responsible for providing the waveform characteristics, such as low OOB and low PAPR. At the receiver side, additional channel equalization and GFDM demodulation are included prior to the decoder. The flexibility of GFDMA can be exploited to improve the spectral usage in the downlink by using GFDM with well-localized FD pulse shape on the edge subcarriers,

whereas the centre subcarriers can still employ OFDM maintain low-complexity processing. On the other hand, asynchronous multiple access is supported in the uplink by using GFDM waveform for all users. Beside the conventional waveforms, additional OFDM precoding techniques for time and frequency diversity, e.g. OTFS, can be realized.

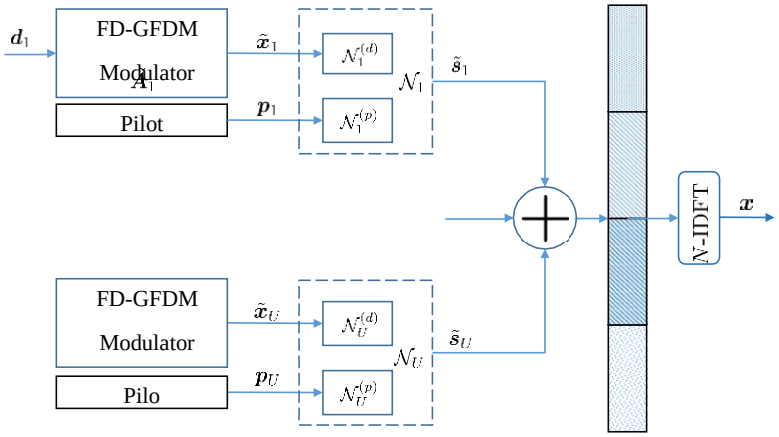


Figure 24 Flexible GFDMA Scheme.

5.1.5 Radio slicing: resource allocation, instantiation and coordination

The developed Multi-Access-Scheme allows using OFDMA or SC-FDMA as special cases. Thus, in the final year radio slicing with very different waveforms is going to be analysed. For example, having GFDM for low-out-of-band emissions at the edges for the spectrum, but using 5G NR (OFDM) in the center for legacy purposes.

5.1.6 Testbed integration

The described implementation will be made available as part of the TUD testbed. The implementation will run on the XILINX FPGA inside USRP-RIO SDR. All platforms support the PHY and higher layer capabilities with a varying processing speed.

5.2 Relation to showcase

The presented implementations are relevant for showcase 2 and showcase 4. Further, the system will be evaluated in the year 3 of ORCA using a set-up build out of several FRANKA EMIKA robot arms to understand the influence of a wireless network into the control loops of an industrial robot application.

5.3 Risk analysis

There is no specific risk observed at this phase of the work package.

5.4 Implementation plan

The presented TDMA MAC will be integrated with the developed GFDM transceiver. Further, other MAC implementations, e.g. from NI will be studied for integration with the transceiver. The FDMA procedure will be integrated into the NI LTE Application Framework, since the GFDM modem can be seen as an OFDM precoding procedure. Both implementations are available for the platforms in the testbed.

6 HYBRID FPGA PLATFORM INCL. FLEXIBLE MAC (IMEC)

6.1 Implementation results

6.1.1 SDR control plane improvements

The improvement of SDR control plane on hybrid FPGA platform (ZYNQ + FMCOMMS2 frontend) has been carried out to reduce latency, which comes down to the two aspects:

1) In year 1 of ORCA the data of radio stack is stored on the DDR memory, in this year it has been shifted to the On Chip Memory (OCM). OCM is a flash memory which performs much faster in terms read and write operation, because the memory access is directly happening on the flash, instead of over multiple stages of cache. It has been tested that this change could bring 200 us improvement.
2) The configuration of radio chip AD9361 was originally achieved via its SPI interface, this is the mode adopted by AD9361 reference design, and in this configuration, the radio chip takes 320 us to switch between transmission and reception status. After studying the datasheet of this chipset, we discovered another possibility. Rather than using SPI interface, there exists IO pins (i.e. ENABLE, TX, RX) of the chip to directly switch states. This method is recommended if the baseband processor module has extra control outputs that can be controlled in real time, allowing a simple two-wire interface to control the state of the AD9361 device. Our baseband IP core is written by HDL, hence it is adapted to support the ENABLE/TXNRX pin control method. After migrating from SPI to direct pin configuration, the Tx/Rx switching time has been brought down to 53.2 us, including the calibration time of various analogue components of the RF frontend. Later on it is observed that the calibration is only needed when Tx/Rx operating frequency is changed. Since the radio frontend remains at the same frequency in our system, this is not necessary. After removing the calibration phase and using direct pin configuration, the Tx/Rx switching time is further reduced to 17.2 us. Lastly the switching time is reduced to virtually zero, by keeping both Tx/Rx path on in the AD9361 chip. More details of the physical layer improvement here is elaborated in Section 6.1.1 of D6.3.
3) The improved Tx/Rx switching scheme is integrated in the TAISC framework. And the switching time in MAC layer is also significantly improved. With the old Tx/Rx switching approach (with SPI bus), the turnaround time is above 400 us. This is measured by a logical analyser, as shown below. Note that the first 3 pins are the SPI bus of TAISC software debugging message, the TRAIN pin contains the measurements of Tx/Rx switching time.

Figure 25: Turnaround time with SPI bus configuration on AD9361at MAC layer.

The measurement of improved Tx/Rx switching shows that the turnaround time is now around 18 us, as is shown in the screenshot below.

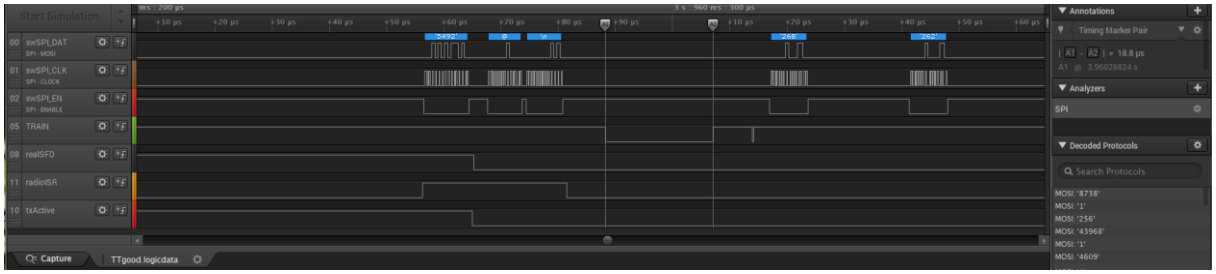


Figure 26: Turnaround Time with direct pin configuration, removing calibration on AD9361 at MAC layer.

6.1.2 Radio slicing: resource allocation, instantiation and coordination

The improvement of radio slicing in the control plane is realized from both host PC side and on the embedded processor itself. On one hand we make the control interface of virtualized radios on ZYNQ SDR easier by exposing a host based interface API. Two function calls in the API are listed below as examples:

1) *int radio_instantiate(uint32_t channel_ID);* This function instantiate a virtual radio on a given channel ID, channel ID is an integer that identifies the logical channel among a set of predefined frequency bands. For now this argument is sufficient, as we mainly consider frequency slicing. The function returns an identifier (radio_ID) of the radio instance if it succeeds.
2) *int send_pkt(void *p, uint32_t pkt_len, radio_ID);* This function sends a packet with certain length on a given virtual radio instance.

On the other hand, an effort has been made to run embedded Linux on the ARM processor for controlling the OFDM transceiver as a Wi-Fi interface. The benefit here is that Linux has rather complete upper layer stack for various of data and control plane functionalities, and it has most rich tools to integrate with existing SDN/NFV functionality. Therefore it is a valuable contribution and at the same time a wise design choice to enable configuration of the OFDM transceiver on the ZYNQ SDR by embedded Linux. An overview of the key building blocks involved in data and management/control functionality of the OFDM transceiver is illustrated in the figure below.

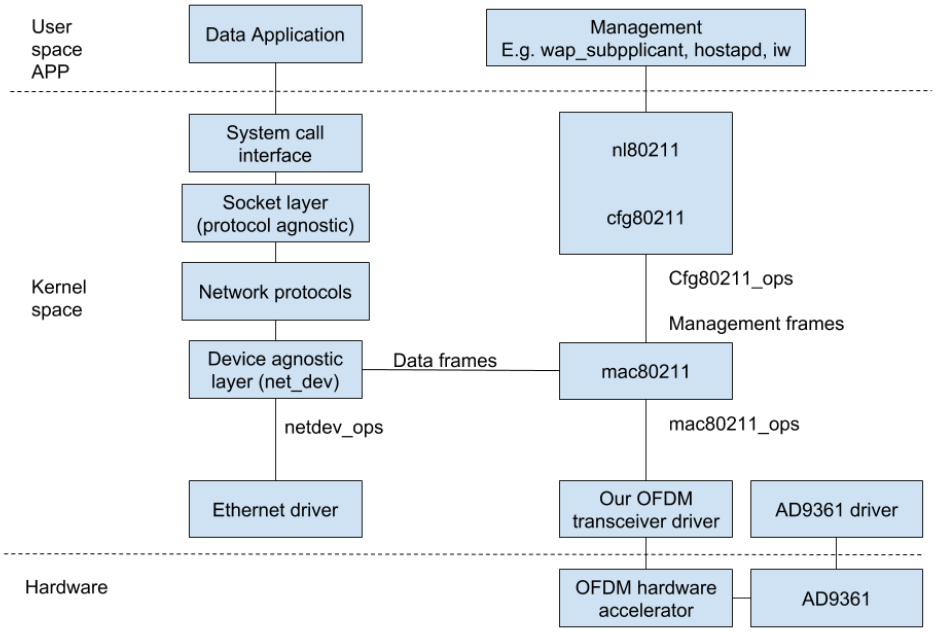


Figure 27: Overview of OFDM transceiver's data and control path integrated with embedded Linux on ZYNQ SDR, adapted from [17].

From top level, building blocks either belong to the user space, kernel space or the hardware. In our case, the hardware block is the OFDM hardware accelerator. The radio frontend is AD9361 chipset, the driver to enable configuration of this chipset in Linux is already offered by Analog Devices [18], together with an embedded Linux distribution that runs on the ARM processor [19]. So the main implementation work identified is to implement our OFDM transceiver's driver. On the ZYNQ SoC, several types of ports (eg, high performance port or general port) on the ARM can be used to make connection with the IP cores running on FPGA More specifically, AXI4 stream bus is used for high speed low latency packet transfer between FPGA and ARM, whereas AXI4 lite bus is used for low latency register access (for PHY/low MAC runtime configuration and status access).

The driver implements a set of functions specified by the *mac80211_ops* interface. The corresponding function is then called by the mac80211 block to handle mainly the high MAC functionalities of IEEE 802.11 protocol in Linux kernel space. The mac80211 block is where the data and management packets are separated for receiving side, or integrated for transmitting side. For the rest of the graph we take receiving side as an example. When a packet enters the mac80211 module it is forwarded to netdev module if it is a data packet, otherwise it is passed upwards towards the cfg80211 block. The data packet is then passed through various processing blocks, finally returned to the user space application which is probably listening on a predefined socket. The journey of a control/management packet is different, it is handled by cfg80211block and nl80211 block, and then passed to some dedicated user space applications, such as wap_supplicant, hostapd, or iw.

The verification of the OFDM transceiver driver is broke down into the following steps:

1) After the embedded Linux is booted on ARM, the OFDM transceiver is recognized by Linux as SDR0 interface. The printout at the end of Linux booting process and the output of 'ifconfig' command is shown below.

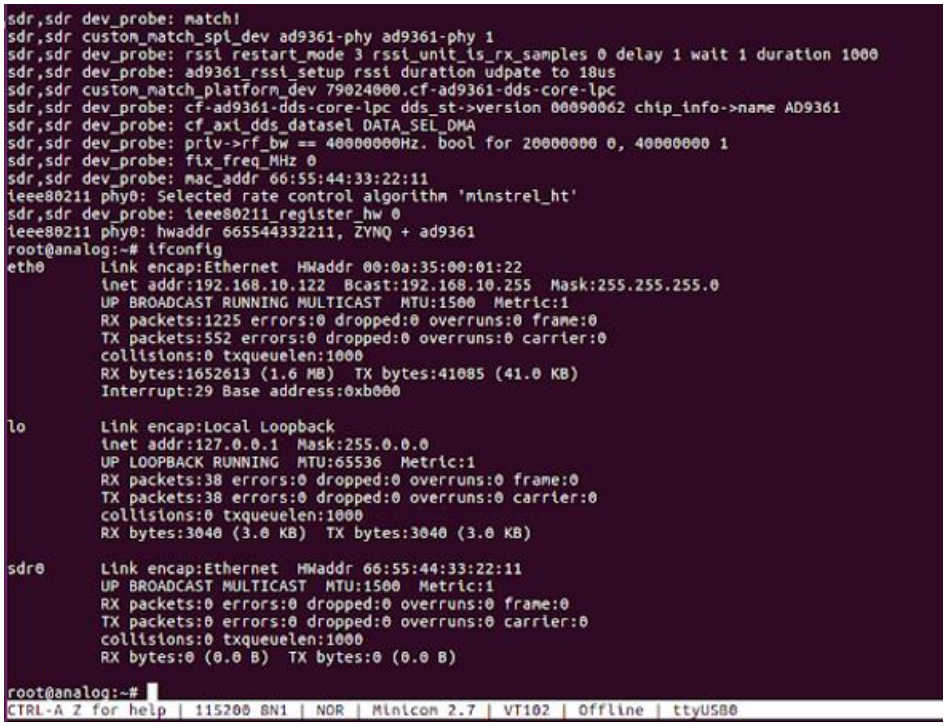


Figure 28: OFDM transceiver recognized as sdr0 interface after Linux booted on ARM.

1) The sdr0 interface can be configured by 'iw' to scan nearby access points, the results are shown in the figure below, due to very long output, only the first obtained AP details are shown completely.

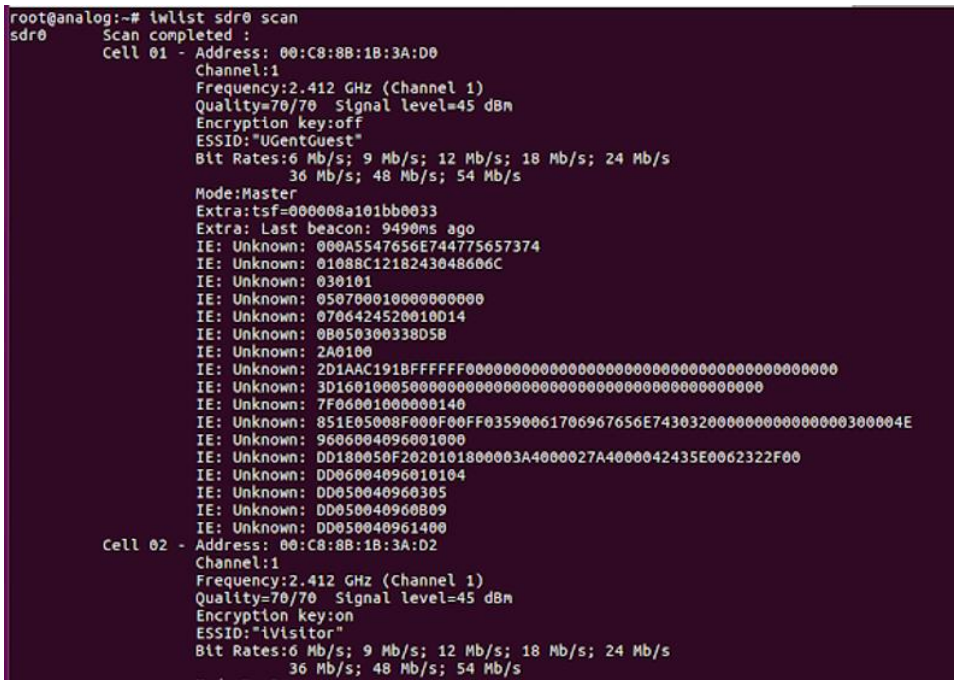


Figure 29: OFDM transceiver is controlled by 'iw' to scan nearby access points.

2) The sdr0 is configured into adhoc mode, and a commercial laptop is able to associate with it. This is achieved by running the following command on the embedded Linux on ZYNQ SDR:

ip link set sdr0 down

iwconfig sdr0 mode ad-hoc

iwconfig sdr0 essid 'sdr-ad-hoc'

ip link set sdr0 up

iwconfig sdr0 channel 8

ifconfig sdr0 192.168.13.20 netmask 255.255.255.0

ifconfig

Then we run 'iwconfig' command again and observe that the adhoc mode is configured, and the cell "96:AC:C8:8C:DA:0D" is up, however there is no station associated with it, the printout of this step is shown in the figure below.

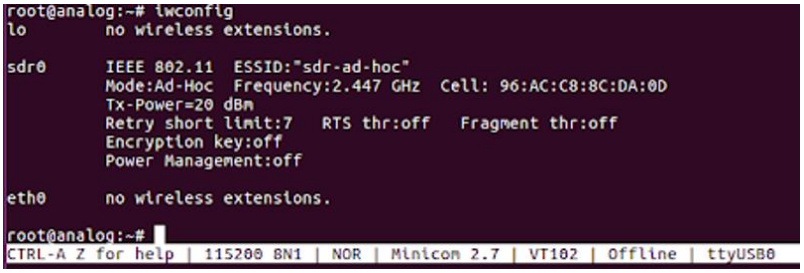


Figure 30 OFDM transceiver is configured as a Wi-Fi station in adhoc mode.

Finally a commercial laptop is used to associate with the SDR, by running the following commands, where the "wlx00c0ca84636a" is the interface name of the Wi-Fi card recognized by the laptop:

sudo ip link set wlx00c0ca84636a down

sudo iwconfig wlx00c0ca84636a mode ad-hoc

sudo iwconfig wlx00c0ca84636a essid 'sdr-ad-hoc'

sudo ip link set wlx00c0ca84636a up

sudo iwconfig wlx00c0ca84636a channel 8

sudo ifconfig wlx00c0ca84636a 192.168.13.13 netmask 255.255.255.0

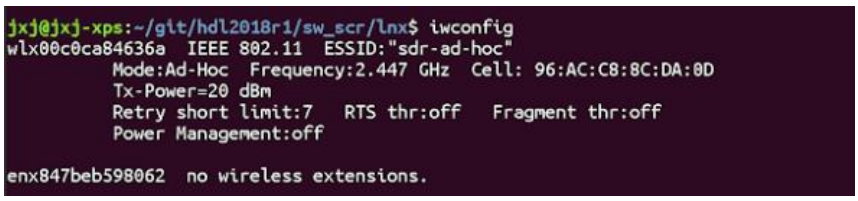


Figure 31 A commercial laptop joined the adhoc network of ZYNQ SDR.

6.1.3 Testbed integration

The improvement on the ZYNQ SDR control plane, more specifically the Tx/Rx switching time, is integrated into TAISC, and testbed users can checkout the latest code in the specific TAISC development code repository to use the new feature.

6.2 Relation to showcase

The feature of improved Tx/Rx turnaround time in the SDR control plane is closely related to showcase 2. This is used to improve the radio link's latency response. The OFDM transceiver's integration with Linux is used to develop showcase 3. In year 1, virtualized radio slices are established by baremetal applications on the ARM, whereas in year 2 this is achieved by applications running in Linux. This new feature makes it more convenient to interface with software in SDN field.

6.3 Risk analysis

There is no specific risk of this work package at this moment.

6.4 Implementation plan

6.4.1 SDR control plane improvements

As stated in WP3 planning, the main tasks of Y3 for data plane is to achieve virtualization of Zigbee physical layer. The control plane is also going to be adapted in order to cope with the multiple virtualized physical layer instances. The integration of PHY and MAC protocol stack is the main development task here.

6.4.2 Radio slicing: resource allocation, instantiation and coordination

The integration of OFDM transceiver with embedded Linux is an ongoing work, and it will be continued in the 3rd year of ORCA. The driver of OFDM transceiver will be continuously improved, driven by the need of upper layer control/management functionality.

6.4.3 Testbed integration

The ZYNQ SDR's Linux control/management of OFDM transceiver is still in early development phase in this year, it is in the planning to release stable features periodically when they are available in the corresponding ORCA software components repository. The same applies for new features of Zigbee virtualization.

7 MANY-TO-MANY RADIO VIRTUALISATION (TCD)

This section describes TCD's Many-to-Many Spectrum Virtualisation Layer (MySVL), a framework for radio virtualisation implemented in Year 2, and the main results obtained. In this deliverable, we describe the general control plane, the radio virtualisation capabilities of this framework, and future steps. For further details on the real-time processing aspects of TCD's contribution to Year 2, please refer to Section 7 of D3.3.

7.1 Implementation results

Throughout this section, we describe MySVL's capabilities with regard to its use for slicing and aggregating the RF front-ends of multiple SDRs. This is in line with its intended use in the Year 2 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 investigating open research challenges in radio virtualisation, such as isolation, signalling and mobility management.

TCD's radio virtualisation implementation is designed to be many-to-many – both in terms of wireless link virtualisation and in terms of Radio Frequency (RF) front-end virtualisation. Many-to-many RF front-end virtualisation enables: (i) the sharing of RF front-ends between multiple independent transmissions/receptions; (ii) the aggregation of multiple RF front-ends for one transmission/reception; and (iii) any combination of the sharing and aggregation. Similarly, many-to-many wireless link virtualisation allows (i) multiple virtual links to share a real wireless link in some manner, (ii) one virtual wireless link to exist over multiple real wireless links (although many-to-many RF front-end virtualisation is required), and (iii) some variation of splitting and aggregation of wireless links. Since both many-to-many virtualisation of RF front-ends and spectrum is possible, we call our implementation the Many-to-many Spectrum Virtualisation Layer, or simply, MySVL.

7.1.1 SDR control plane improvements

MySVL is currently implemented as an out-of-tree GNU Radio block; however, we are developing it as a stand-alone program, able to use any SDR framework, e.g. GNU Radio, srsLTE, LuaRadio, etc.

The overall functionality of MySVL is shown in Figure 32. At the transmitter, MySVL takes in complex samples in the time domain from a number of physical layer inputs, which can be of any type of signal using any type of modulation. The block performs an FFT on each input signal, and next remaps the input frequency bins (the virtual spectrum) to output frequency bins (the real spectrum). Virtual spectrum can be mapped to real spectrum either contiguously or non-contiguously. Then, the real spectrum frequency samples are converted to the time-domain using an Inverse FFT (IFFT) and forwarded to the radio frequency front-end. The MySVL block at the receiver operates very similarly, except instead of taking in complex samples from the PHY, it receives samples from the RF front-end. Subsequently, samples are mapped from the real spectrum to the virtual spectrum, and outputted to the PHY.

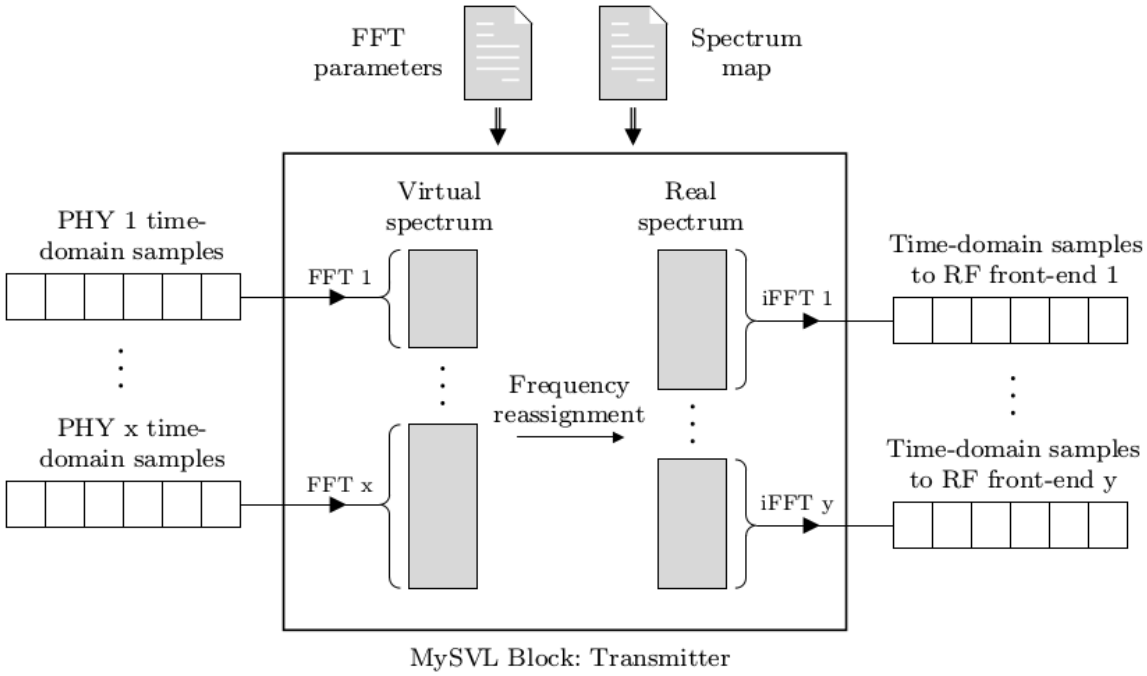


Figure 32: Overview of MySVL functionality.

Time-domain samples from multiple signals are converted to the frequency domain using FFTs – the size of the FFTs will depend on the type of signal. Next, the frequencies contained in the signals are reassigned based on a spectrum map, after which IFFTs convert the frequency domain samples back to time-domain signals. Note that the functionality of both the transmitter and the receiver is identical; only the FFT parameters and the spectrum map are different.

We design MySVL to read in the FFT parameters and the spectrum map from different files; this increases flexibility and independence as it allows the spectrum map to be updated separately from the FFT parameters. In a sense, this is the separation of the isolation and the embedding problem, and the spectrum map can be generated using any embedding technique.

7.1.2 Frame Synchronisation and Non-Contiguous Spectrum Slicing

We now describe the frame synchronization operations necessary to perform many-to-many radio virtualisation.

At the receiver, an FFT is used on the time-domain samples to convert the signal to the frequency domain. However, if the FFT window is not aligned correctly to the time-domain samples, it could cause samples to be mapped wrongly in the frequency domain. As such, it is necessary to synchronise the receiver so that the FFT window is aligned correctly. Especially in the case of non-contiguous slicing and in spectrum aggregation, where the transmitters (or receivers) must be synchronised in time. We call a series of FFT windows a frame, as essentially we want to achieve frame synchronisation.

Although there are many means of achieving frame synchronisation at the receiver, we need to use a synchronisation method that relies on a unique preamble because our implementation of wireless link virtualisation is generic. The reason behind this design choice is that if we use a synchronisation method that does not use a unique preamble, then we could potentially synchronise incorrectly if a user signal is transmitted that relies on the same synchronisation technique. For example, if we use autocorrelation to synchronise in time, then we might get false-positive synchronisations when transmitting an Orthogonal Frequency Division Multiplexing (OFDM) signal, as OFDM also uses auto-correlation for synchronisation. Therefore, we use the cross-correlation estimation block which is already implemented in GNU Radio.

Frame synchronisation is done by cross-correlating incoming samples with a known preamble, a complex Hermitian sequence [20]. A triggered demux block, that we developed, separates the payload samples from the preamble and ensures that the samples going to the MySVL block are aligned with the FFT window.

We tested that frame synchronisation ensures that the FFT window aligns to the correct time-domain samples, ensuring the reliable operation of MySVL for non-contiguous spectrum virtualisation. We allocated an OFDM signal and a Gaussian Minimum Shift Keying (GMSK) signal to non-contiguous spectrum, using frame synchronisation as described. Figure 33 illustrates preamble detection and separation from the payload samples.

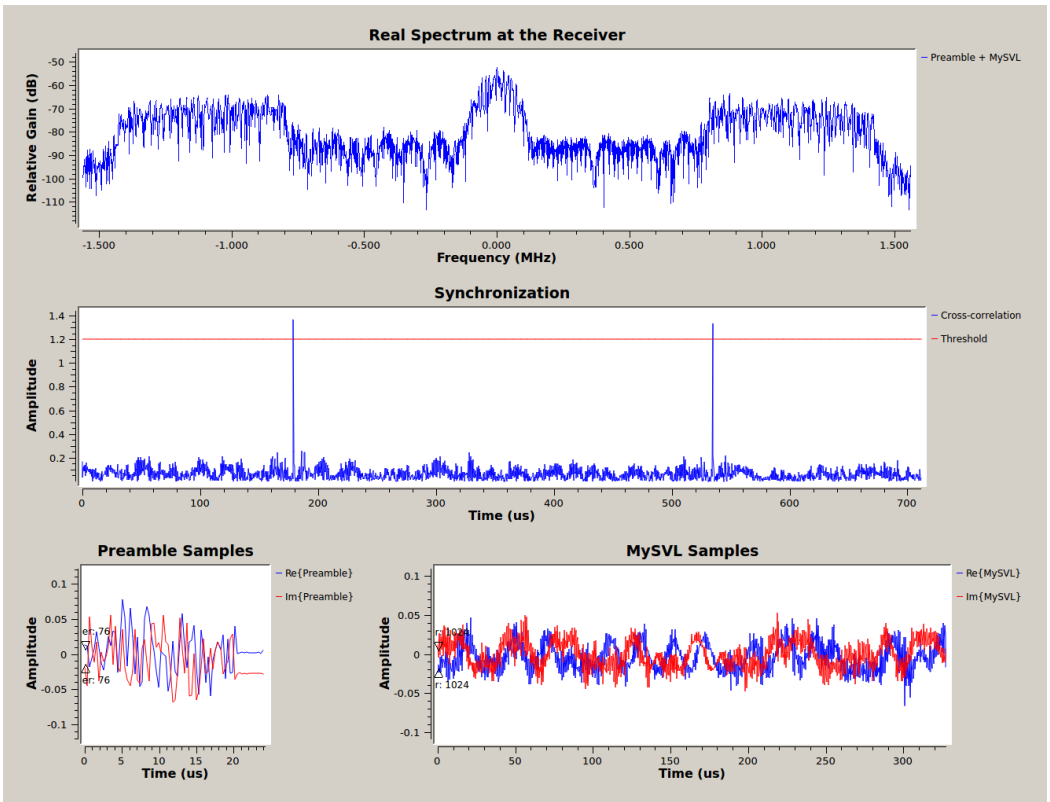
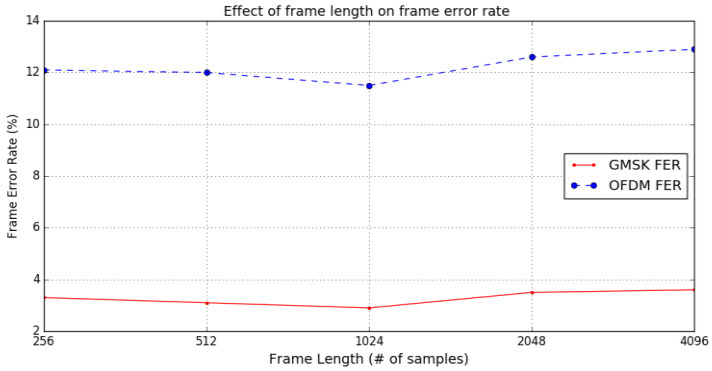


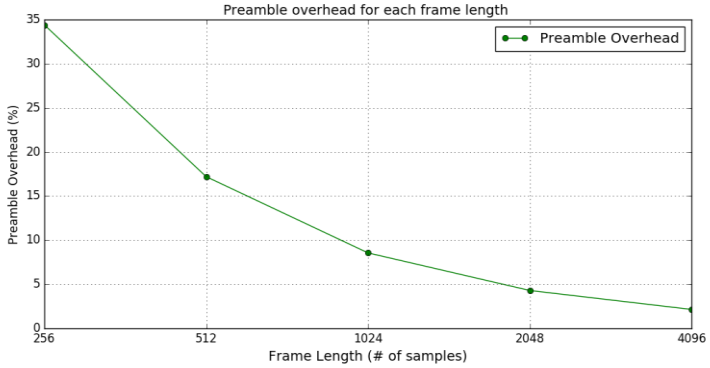
Figure 33: Synchronization at the receiver.

The real spectrum received is shown in the top graph, while the middle graph shows the preamble detection through cross-correlation. The graphs on the bottom row show the separation of the preamble from the payload samples.

We measured the frame error rate of non-contiguous spectrum slicing by comparing the number of bytes transmitted to the number of bytes received correctly for several different packet sizes (or frame lengths) The frame error rates for OFDM and GMSK signals are shown in Figure 34. As we can see from this plot, the GMSK signal has a Frame Error Rate (FER) of approximately 3~3.5%. While the OFDM frame error rate is around 12%, a substantial improvement on the test without the frame synchronisation, where we obtained an FER of 57%. The frame synchronisation massively increased the number of packets received in the non-contiguous case, and moreover allows the non-contiguous case to be used reliably.



(a) Frame error rates for OFDM and GMSK signals for different frame lengths.



(b) The preamble overhead per frame length.

Figure 34: Frame error rates and preamble impact.

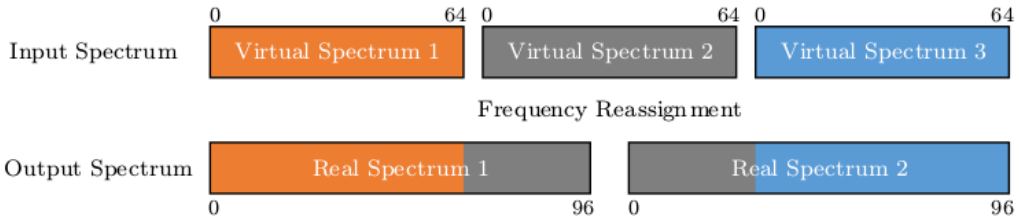
Analysis of the impact of frame length. Although the FER does not change much, the overhead decreases significantly as the frame length increases. The overhead of the preamble decreases as the frame length increases.

7.1.3 Many-to-Many Virtualisation

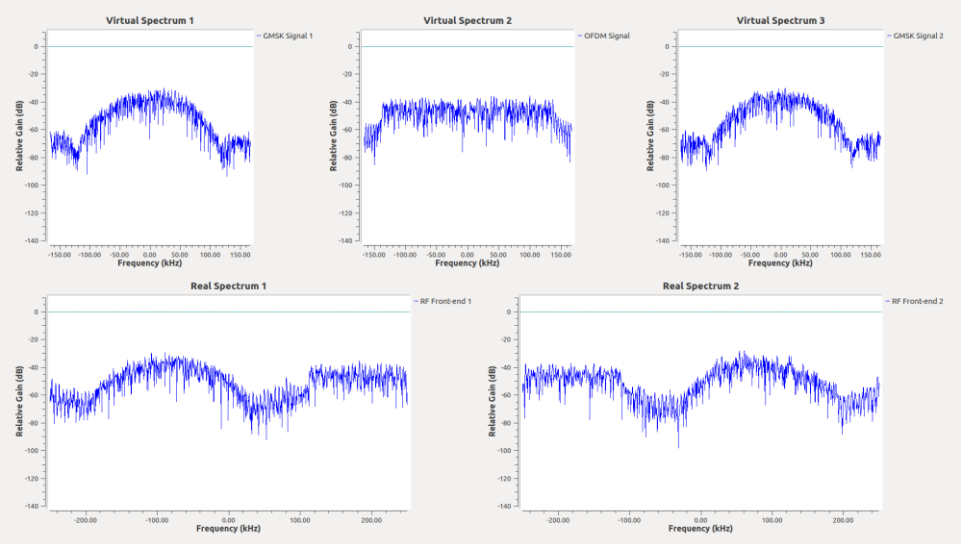
Finally, we show the ability of MySVL to perform many-to-many virtualisation correctly. We use an OFDM signal and two GMSK signals, similar to the example shown in Section 7.1.2. The resulting real and virtual spectra are shown in Figure 35. The files are received correctly in simulation, while

over-the-air we receive packets for a short duration, occurring in longer cycles. The reason for this is that the two RF front-ends have slightly different oscillating frequencies, and therefore they synchronize temporarily, before getting out of sync again. We are working on developing a synchronization method to overcome this.

It is interesting to note that the many-to-many case presents another method of splitting a signal, such as the OFDM signal in this case, which can then be transmitted in separate parts of the real spectrum by the RF front-ends. Similarly, to the use of null signals, this can be used to avoid already existing signals.



(a) Conceptual spectrum map at the transmitter.



(b) Resulting virtual and real spectra at the Tx.

Figure 35: Many-to-many virtualisation.

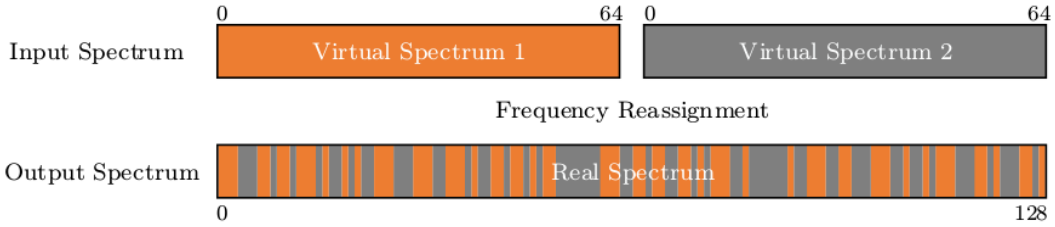
Many-to-many virtualisation allows signals to be split between multiple RF front-ends. This provides another method to avoid interference with other signals, in addition, the use of null signals.

7.1.4 Use Case: Physical Layer Security

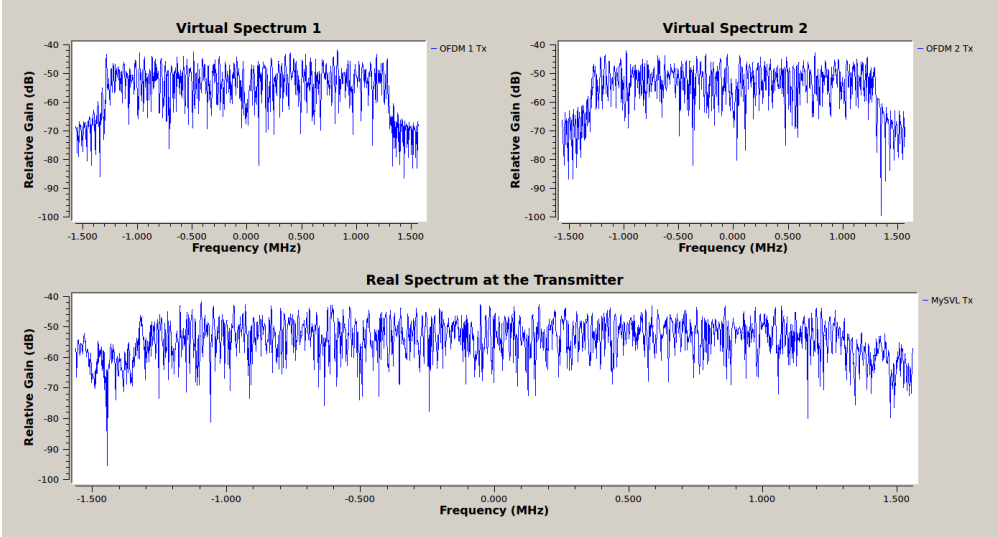
One application of spectrum-level virtualisation could be physical layer security. Physical layer security refers to the concept of exploiting the physical properties of the wireless channel to encrypt information and achieve secure transmission between nodes. We argue that PHY security also includes secure communication through the use of the physical properties of the wireless signal.

To show how physical layer security can be achieved in practice, we demonstrate the use of wireless link virtualisation to achieve security between transmitter and receiver nodes on the testbed. In this

demonstration, two OFDM signals are used, rather than an OFDM signal and a GMSK signal. However, the first OFDM signal uses Quadrature Phase Shift Keying (QPSK) while the second OFDM signal uses Binary Phase Shift Keying (BPSK). We also omit any filtering and resampling operations.



(a) Conceptual randomly-generated spectrum map.



(b) Resulting virtual and real spectrum at the Tx.

Figure 36: Physical layer security through virtualisation.

MySVL can be used to achieve computational physical layer security, by scrambling the FFT bins. In this example, two OFDM signals are interleaved in a way that only the right spectrum map can descramble the two signals.

The spectrum map used for this demonstration is shown in Figure 36a. This spectrum map has two inputs of size 64 and one output of size 128, and thus there are roughly 2.4 × 10^37 possible permutations. The virtual and real spectra for the transmitter are shown in Figure 36b. As before, we measure the correctly received throughput of each signal by examining the number of bytes received and the number of bytes transmitted to determine the frame error rate. The first OFDM signal has an FER of 7.8%, while the second OFDM signal has an FER of 9.91%. The worse performance of the second signal can be explained by the fact that some frames are dropped at the start of the transmission, and since the first signal has a higher throughput, the effect on total throughput is lessened. However, although we were able to use wireless link virtualisation for physical layer security in the case of two OFDM signals, it is not as easy to scramble the signals when different modulation types are used, as the signals will interfere with each other, rather than being orthogonal to each other.

7.1.5 Testbed integration

MySVL was developed and validated in the IRIS testbed, using USRP N210s and X310s. MySVL is available as an ORCA functionality to open call partners, and its source code can be found online at https://github.com/jvandebelt/gr-mysvl.

7.2 Relation to showcase

The Showcase 3 will be divided into three stages: the demonstration of TCD's FFT-based radio slicing capabilities, the demonstrations of IMEC's OFDM-based radio slicing capabilities, and the combination of the two radio slicing approaches. MySVL will empower Showcase 3 with its many-to-many virtualisation capabilities, allowing the creation of virtual radio with non-contiguous or aggregated spectra.

7.3 Risk analysis

There are no identified risks at this stage, with regard to the SDR control plane improvements and radio slicing in the proposed solution.

7.4 Implementation plan

This section describes TCD's plans to extend MySVL, and add Radio Function Virtualisation (RFV) on SDRs for Year 3. We plan to integrate MySVL and the eXtensible Virtualisation Layer (XVL) into one stand-alone program. This unified platform will then be used to instantiate radio functions on top of virtual RF front-ends, ultimately, allowing the creation of virtual radios with both a virtual RF front-end and a virtual radio stack, as shown in Figure 37. For more information about XVL, please refer to Section 7 of D3.3.

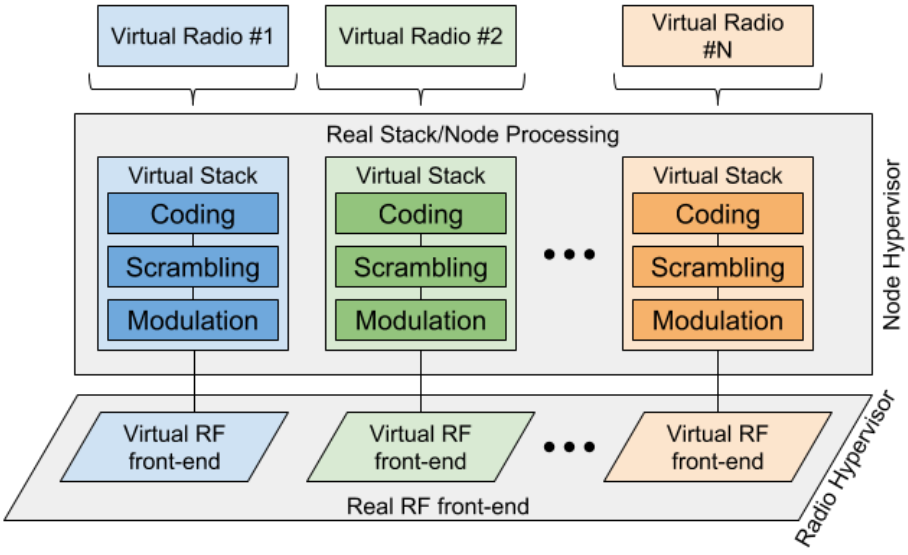


Figure 37: Illustration of virtual radios with virtual RF front-ends and virtual radio function stacks.

7.4.1 SDR control plane improvements

The current implementation of XVL allows the instantiation of virtual RF front-ends on demand, allowing its virtual radio to an operation similar to RRHs, i.e., receiving/transmitting raw IQ samples. We plan to support the remote instantiation and configuration of radio functions on top of these virtual RF front-ends. Hence, providing an SDR control-plane framework that enables the instantiation of virtual radios with a virtual RF front-end and a virtual radio stack.

7.4.2 Radio slicing: resource allocation, instantiation and coordination

The planned extension will support the independent radio virtualisation and configuration of multiple nodes with SDRs. Each node will operate as an independent Base Station (BS) of a Centralised-RAN (C-RAN), which can be virtualised and run different virtual radio functions. Hence, the platform will allow the control and configuration of multiple SDR-enabled nodes, and their respective virtual radios, at a centralised point.

7.4.3 Testbed integration

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

1. Instantiation of virtual radio radios with only a virtual RF front-end, operating similar to RRHs on multiple SDR-enabled nodes.
2. Instantiation of virtual radios with a virtual RF front-end and a virtual radio stack, on multiple SDR-enabled nodes.

8 NS3 BASED PROTOTYPING PLATFORM FOR RAT INTERWORKING (NI)

The goal in Year 2 was to further enhance the Multi-RAT platform for enhanced control plane functionality and networking options towards the overall envisaged platform shown in Figure 38.

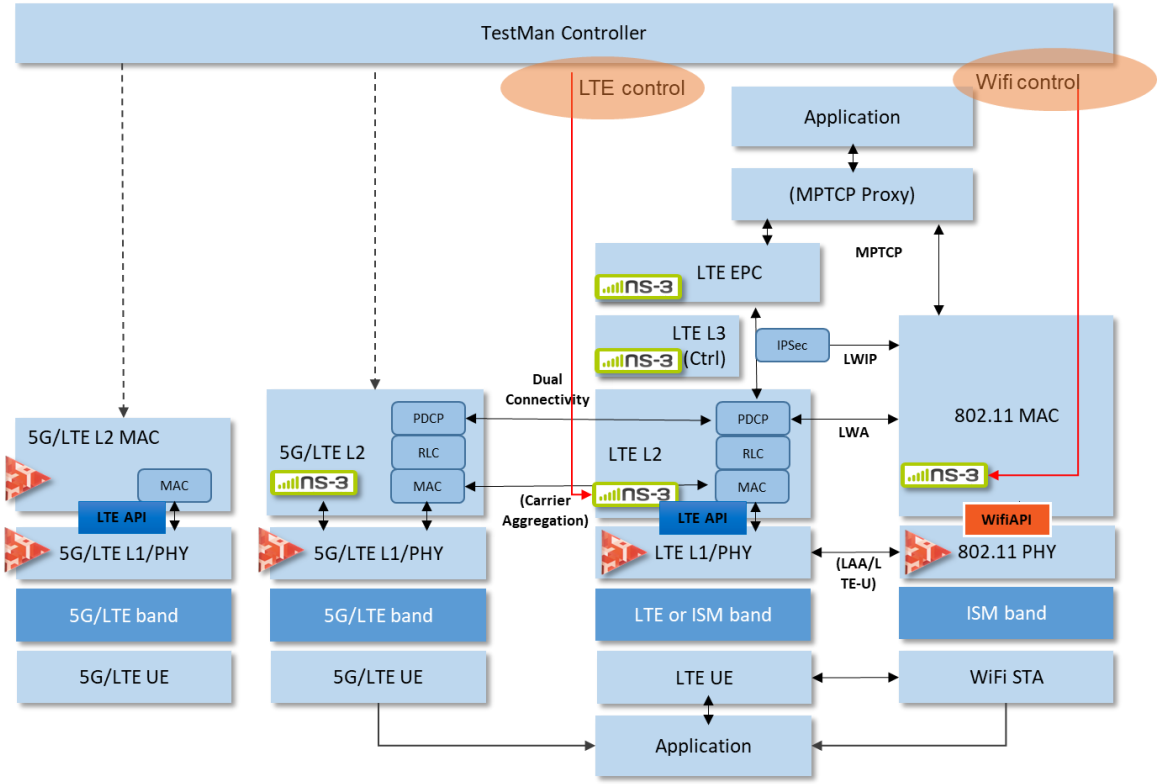


Figure 38: Overview of final RAT Interworking platform with highlighted focus areas from Year 2.

The specific focus of Year 2 was drawn towards two parts of the platform

- To further enhance the accessibility of the Multi-RAT platform parametrization and even further to give experimenters a basic interface for SDN like overall control functionality, the TestMan control interface was enhanced towards the use in ns-3 for LTE and WiFi.
- The basic ns-3 network topology example code of the Multi-RAT platform was streamlined to accommodate a more realistic networking scenario that includes several important entities for networking examples and additionally integrates the LWA/LWIP LTE-WiFi Interworking extension from Open Call 1 for Extensions.

A detailed technical explanation of the aforementioned additions to the Multi-RAT platform will be given in the next subsections.

8.1 Implementation results

8.1.1 SDR control plane improvements

8.1.1.1 Remote control for ns-3

The capabilities of configuring the Multi-RAT testbed remotely have been further enhanced in Year 2. Specific emphasis was put on the configuration of ns-3 as an integral part of the Multi-RAT platform. As the platform involves the use of the newly developed L1/L2 API for LTE and WiFi, the focus shifted towards controlling ns-3 solely and leaving out any direct configuration of the PHY layer. While in Year 1, a parametric configuration of ns-3 was possible with the help of scripts, the efforts of Year 2 were spent on integrating a dedicated module into ns-3 to support reconfiguration of ns-3. Figure 39 shows the systems block diagram of the implementation.

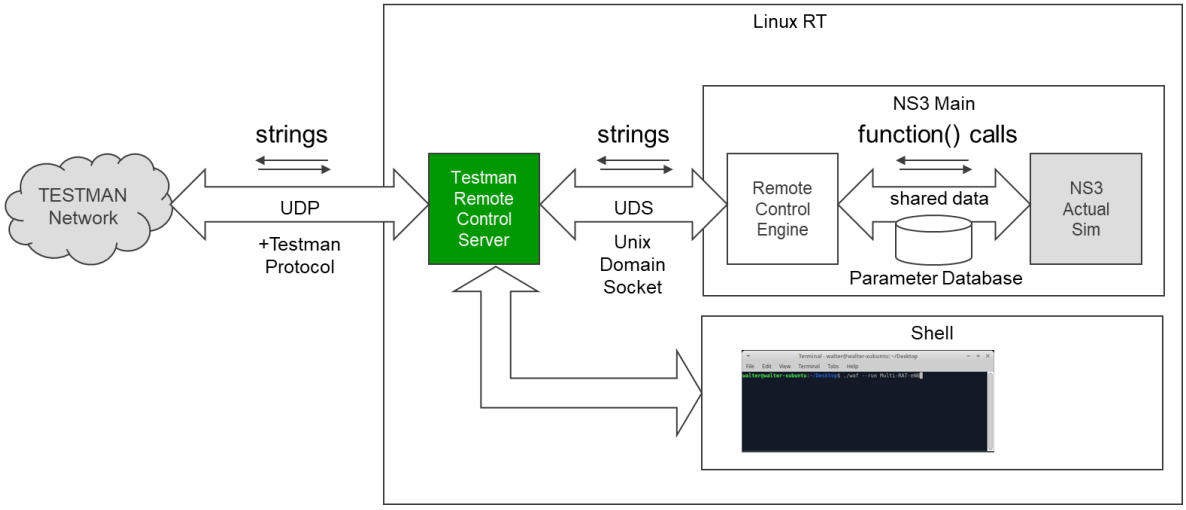


Figure 39: Block diagram of TestMan remote control interface implementation for ns-3.

To allow for configuration of parameters in ns-3, three building blocks had to be implemented and will be shortly described in the following paragraphs.

The *TestMan Remote Control Server* incorporates the TestMan Server application provided by TUD that provides the testbed-wide connectivity and protocol for TestMan. To communicate within Linux with a ns-3 simulation, an interface needed to be incorporated in this module. As ns-3 will run on a Linux RT system in the testbed, the inter process communication was chosen in the form of Unix Domain Sockets for the interfacing.

A *Remote Control Engine* was incorporated into ns-3 as a new module. The purpose of this interface on one hand is to receive and transmit commands from and to the TestMan Server inside the Linux system. Secondly, the module implements a command parser that resembles the command structure of the TestMan protocol. The engine is processing and writing or reading parameters according to the message format into an internal parameter storage class, called *Parameter Database*.

The *Parameter Database* is the integral class that can be included in the modules of ns-3. It provides a centralized storage of parameters such that each module which includes this class, can either read or write parameters from or to that centralized storage. Different modules within ns-3 can share parameter values as well. The addition of a parameter to the TestMan remote control for use in ns-3 is architecturally separated by only adding the respective parameter definition to a single source file and is exemplarily shown in the following and illustrated in Figure 40.

Header File

```
class ParameterDataBase
{
private:
    int parameterInt1;

public:

    void setParameterInt1(int p1);
    void setParameterInt1(std::string p1str);
    int getParameterInt1();
    std::string getStringParamterInt1();
};
```

(1)

(2)

Class implementation (Get/Set)

```
// Demo parameter access methods (2x actual type (int), 2x string)
void ParameterDataBase::setParameterInt1(int p1){
    parameterInt1 = p1;
    return;
}
void ParameterDataBase::setParameterInt1(std::string p1str){
    parameterInt1 = std::stoi(p1str);
    return;
}
int ParameterDataBase::getParameterInt1(){
    return parameterInt1;
}
std::string ParameterDataBase::getStringParamterInt1(){
    return std::to_string(parameterInt1);
}
std::string ParameterDataBase::getParameterByName(std::string pname){

    std::string response = "";

    if (pname.empty()) {
        response = "ERR:EMPTY_VARIABLE_NAME";
    } else if (pname == "ParameterInt1") {
        response = getStringParamterInt1();
    } else {
        response = "ERR_UNKOWN_PARAMETER";
    }
    return response;
}

std::string ParameterDataBase::setParameterByName(std::string pname, std::string pvalue) {
    std::string response = pvalue;

    if (pname.empty() || pvalue.empty() ) {
        response = "ERR:EMPTY_NAME_OR_VALUE";
    } else if (pname == "ParameterInt1") {
        setParameterInt1(pvalue);
    } else {
        response = "ERR_UNKOWN_PARAMETER";
    }

    return response;
}
```

(3)

(4a)

(4b)

Figure 40: Overview of parameter database header files and class implementation with necessary entry points for new parameters.

The addition of a new exemplary *ParameterInt1* needs to be added to the header file and the respective source file of the Parameter Database implementation. Therefore at (1) in Figure 40, the private parameter in the database is defined. This is the memory allocation where the actual value is

permanently stored. Additional set and get functions are declared in the header file, shown at (2). Note that for every value, irrespective of the type, a string accessing method also needs to be declared which is needed to couple the value of the parameter to a string representation mentioned in the TestMan command. The definition of the get and set methods is shown in (3). Finally, the mapping of the internal value of the ns-3 parameter database to the TestMan command is defined in (4a) and (4b) for read and write commands. These methods are directly called by the Remote Control Engine in ns-3. An informal class overview of the ns-3-internal remote control engine is shown in Figure 41.

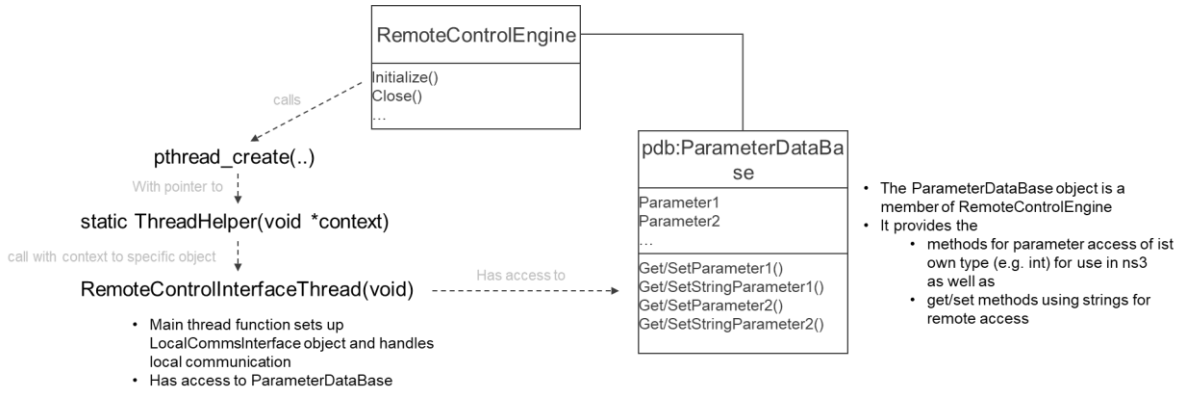


Figure 41: Class overview of the ns-3 remote control engine

To assure proper simulation flow of ns-3 with the remote control engine, the engine is started as a separate thread to parallelize the execution.

As a first proof of concept for the approach, one of the parameters analysed in D4.1 [9] was planned to be implemented, i.e., the MCS. To sync the remote control implementation efforts with the Open Call 1 for Extensions and especially the functionality addition of interworking technologies such as LWA and LWIP into the ns-3 simulator, NI aligned the proof-of-concept implementation to parameters related to this extension. The read and write of parameters on different layers of the ns-3 LTE stack was successfully tested.

While the aforementioned implementation focuses on the remote-control of internal parameters of ns-3, NI additionally implemented the option to start Linux shells on the real-time target where ns-3 will run as depicted in Figure 39. This enables an experimenter to arbitrarily access the Linux RT system and, e.g., start ns-3 with specified scripts which then can be controlled.

To automate the run of specific configurations and scripting of specific experiment sequences, the extensive examples from TUDs TestMan implementation can be used which are available in different scripting languages such as Matlab, Python, C++ and LabView.

Table 7 and Table 8 show a short overview of TestMan commands that can be used to setup and reconfigure the ns-3-based experiment via TestMan interface.

Testman Keyword (testman variable name)	Command (testman variable value)	Parameters	Comment
command	WRITE:<parameter-alias>:<value>	<parameter-alias> … name of parameter defined in implemnetation file of	Writes <value> to parameter with alias <parameter-alias>

		ParameterDataBase <value> … desired value	
command	READ:<parameter-alias>	<parameter-alias> … name of parameter defined in implemnetation file of ParameterDataBase	Read values of parameter with alias <parameter-alias>

Table 7: Testman commands for reconfiguration parameters within ns-3.

Testman Keyword (testman variable name)	Command (testman variable value)	Parameters	notes
command	:SHELL:ID:<NO>: START	<NO> … Shell ID	Start shell with ID <NO>
command	:SHELL:ID:<NO>: WRITE:<PARAM1>	<NO> … Shell ID PARAM1 … String, e.g. shell comand „ls“	Writes the string parameter to shell with ID <NO> (adds line break at the end automatically, invoking command execution) Generally any sh shell commands (switching to bash needs code change, but is easy)
command	:SHELL:ID:<NO>: READ	<NO> … Shell ID	Read everything off the shell with ID <NO> since the last invocation of READ. Truncated to roughly 2000 character. If more characters available, invoke command consecutively and concatenate strings until finished
command	:SHELL:ID:<NO>:: KILL	<NO> … Shell ID	Kill shell with ID <NO> . Possible reasons: Closing Application, Shell script running indefinitely. If shell is supposed to be used afterwards start again
command	:SHELL:ID:<NO>:: RESET	<NO> … Shell ID	Essentially combines :SHELL:KILL and

			:SHELL:START into a single command

Table 8 Testman commands for work with Linux shells on the Linux RT.

8.1.2 Radio slicing: resource allocation, instantiation and coordination

8.1.2.1 Ns-3 Multi-RAT Networking Experimentation Example

One main focus of NI in Year-2 was on setting up a prototyping system allowing for LTE-WiFi interworking experiments utilizing the NI SDR platform. In D4.1 [9] we described the different options for LTE-WiFi interworking evaluated in Year-1. Within Year-2 the Open Call for Extension took place with the goal to extend the ns-3 based prototyping platform with the LWA and LWIP functionality as described in the next section. In addition, NI created a new simulation application in the ns-3 simulator that represents a relevant network scenario including LTE and WiFi links where all the Open Call extension have been finally integrated. Figure 42 shows a sketch of the transmission scenario that NI realized in the new ns-3 simulation example.

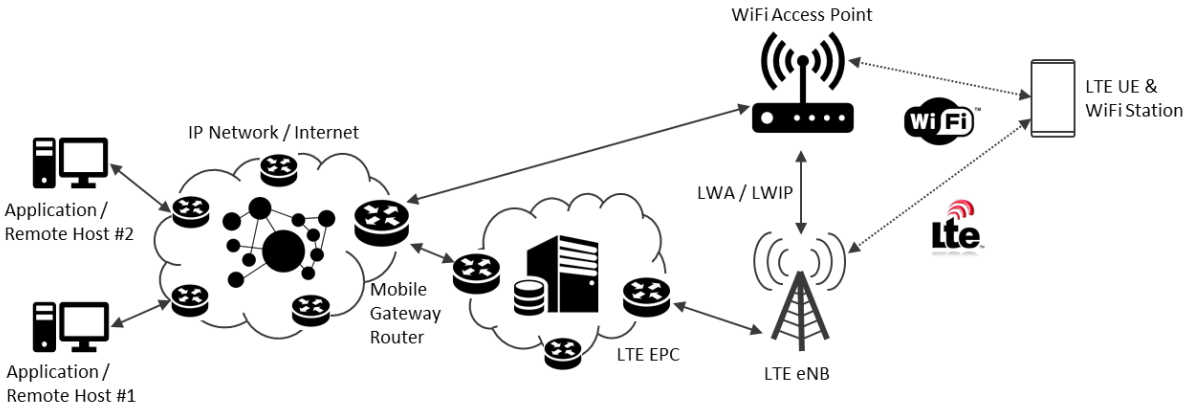


Figure 42: LTE-WiFi Network Scenario.

The idea is to have a configurable number of remote host stations that are connected to the internet via Ethernet as it would be the case is real networks. Within the internet there is a mobile network gateway / router that connects the internet with the wireless access networks where WiFi and LTE are used here that can run in parallel. Furthermore, there exists a configurable number of mobile terminals that include LTE and WiFi interfaces comparable to smartphones and tablets. These mobile terminals can access the WiFi network through the access point or the LTE network through the eNB. The two interfaces can be distinguished by different IP addresses. As a result, one can establish a flexible connection between one remote host station and a mobile terminal just by using a certain IP address configuration. Either on the remote host stations or mobile terminals there can be installed a client or server application that can either generate or receive data packets and hence generating traffic in the network. By choosing the right IP address each station in the network can be reached. The mobile gateway is connected to the LTE and WiFi access networks via point-to-point (P2P) connection that would represent a fibre connection in a real network. Therefore, depending on the routing configuration of the mobile gateway packets will be routed through the WiFi or LTE networks. All networks parts operate on different IP subnets as it would be the case in real networks. Figure 43 shows the realized IP network configuration in ns-3. It already includes the LWA/LWIP extension that will be discussed in the subsequent section in more detail.

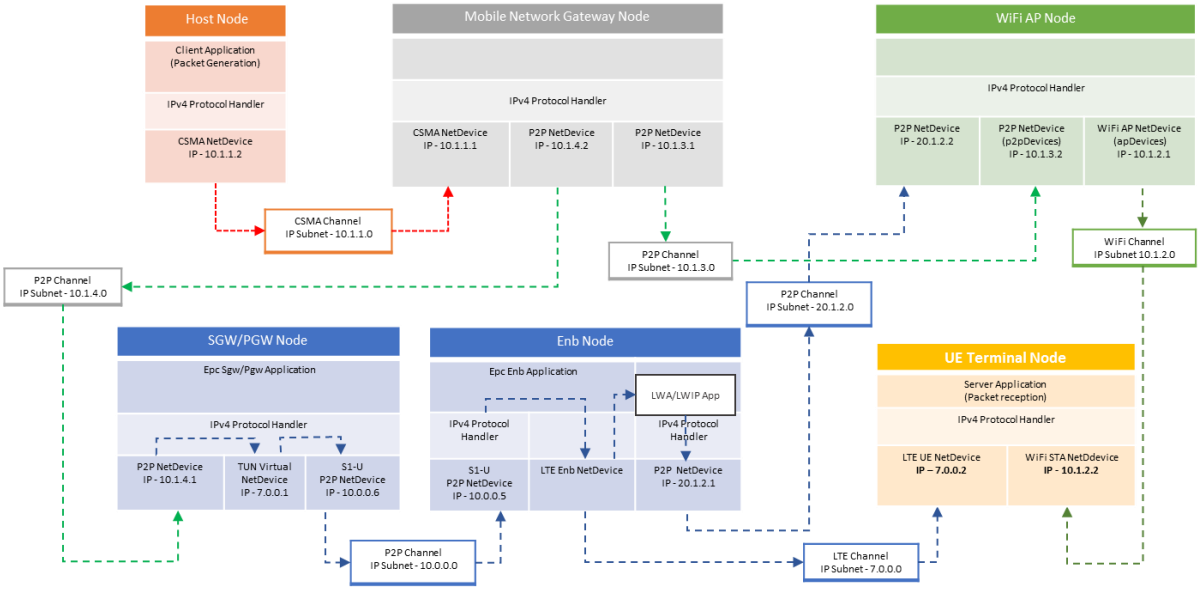


Figure 43: IP Network Configuration in the new ns-3 LTE-WiFi Interworking Setup.

Within the ns-3 program we consequently used the node architecture [21] that is depicted in Figure 44. Each node consists of an application layer, and IP protocol handler as well as a net device. For the net devices implementations interfaces for the most important wired and wireless communication standards are available. It should be noted that each node can include multiple net devices even of same type which allows to create router modules as well as multi standard devices as e.g. needed for emulating realistic mobile devices.

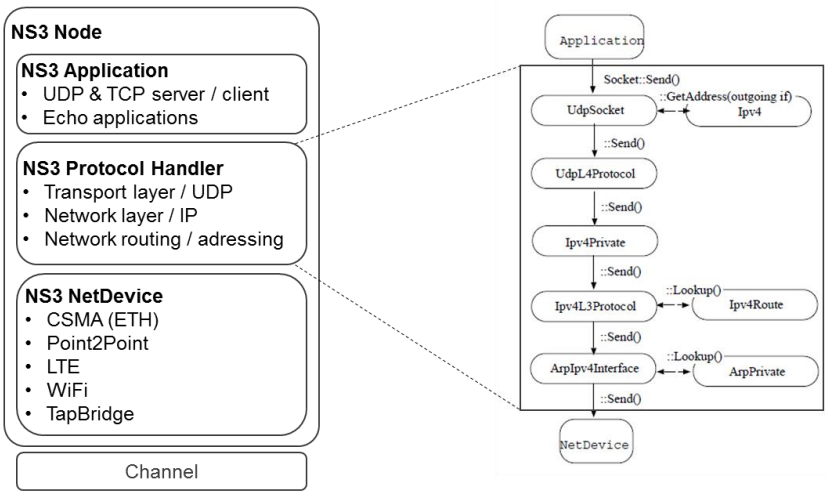


Figure 44: ns-3 Node Architecture

One challenge that has been resolved throughout the implementation of this network configuration are the different routing paradigms that are used in ns-3 for the different network components. While the ns-3 LTE module only supports static routing, the other network components like CSMA, P2P and WiFi modules support global routing. When using global routing, ns-3 is creating all routing tables for the network components automatically based on the IP address configuration provided by the user. In static routing, all routing tables have to be created manually in the code. In the current implementation we used a mixed approach. First, the entire network without LTE is configured using global routing and thereafter static routing is used to connect the LTE Packet Gateway (PGW) to the mobile network gateway. Furthermore, in the remote hosts and mobile terminals the default routes have to be set to

reach the mobile network gateway which itself is configured to distinguish between the LTE and WiFi networks.

Figure 45 shows the console outputs for a common connection between one remote host (IP address 10.1.1.2) running a client application sending two packets to the LTE mobile terminal (IP address 7.0.0.2) running a server application. When looking at the sequence number and Packet Uid one can check that the transmitted and received packets are equal.

```
Use Client Server Config: 2
Number of ETH devices      = 4
Number of WiFi devices     = 3
Number of LTE devices      = 1
Router GW IP Addr          = 10.1.1.1
WiFi Net GW IP Addr        = 10.1.3.2
WiFi AP IP Addr            = 10.1.2.1
WiFI STA#1 IP Addr         = 10.1.2.2
LTE Net GW IP Addr         = 10.1.4.2
LTE EPC PGW IP Addr        = 7.0.0.1
LTE UE#1 IP Addr           = 7.0.0.2
LTE UE#1 STA IP Addr       = 10.1.2.3
Client IP Addr             = 10.1.1.2
Server IP Addr             = 7.0.0.2
Used Client App: 1 -> standard uni directional traffic with one client/server
Start simulation
Client sent 1000 bytes to 7.0.0.2 Uid: 28 Sequence number: 1 Time: 2
Server received 1000 bytes from 10.1.1.2 Uid: 28 Sequence Number: 1
Client sent 1000 bytes to 7.0.0.2 Uid: 44 Sequence number: 2 Time: 3
Server received 1000 bytes from 10.1.1.2 Uid: 44 Sequence Number: 2
```

Figure 45: Reference scenario with remote host sending to LTE interface of mobile terminal.

In order to demonstrate the parallel usage of the LTE and WiFi networks, we created an application that can establish a link to both wireless interfaces at the mobile terminal. When using this application one can distribute and send packets from the remote host application to the mobile terminal application using either the LTE or WiFi networks. Figure 46 shows a similar setup as for Figure 45 with the difference that the remote host (IP address 10.1.1.2) now establishes a connection to the LTE interface (IP address 7.0.0.2) and WiFi interface (IP address 10.1.2.3) of the mobile terminal and sends packets.

```
Use Client Server Config: 2
Number of ETH devices      = 4
Number of WiFi devices     = 3
Number of LTE devices      = 1
Router GW IP Addr          = 10.1.1.1
WiFi Net GW IP Addr        = 10.1.3.2
WiFi AP IP Addr            = 10.1.2.1
WiFI STA#1 IP Addr         = 10.1.2.2
LTE Net GW IP Addr         = 10.1.4.2
LTE EPC PGW IP Addr        = 7.0.0.1
LTE UE#1 IP Addr           = 7.0.0.2
LTE UE#1 STA IP Addr       = 10.1.2.3
Client IP Addr             = 10.1.1.2
Server IP Addr             = 7.0.0.2
Used Client App: 3 -> uni directional traffic with from one client to two destination addresses
Start simulation
Client sent 1000 bytes to 7.0.0.2 Uid: 28 Sequence number: 1 Time: 2
Client sent 1000 bytes to 10.1.2.3 Uid: 29 Sequence number: 2 Time: 2
Server received 1000 bytes from 10.1.1.2 Uid: 28 Sequence Number: 1
Server received 1000 bytes from 10.1.1.2 Uid: 29 Sequence Number: 2
```

Figure 46: Scenario with a remote host sending to LTE and WiFi interfaces of the mobile terminal.

When looking at the server outputs one can see that again the server received two packets from the same client address (10.1.1.2) showing that for the receiver it is transparent whether packets are transmitted over the WiFi or LTE network.

This newly created ns-3 network scenario can be used by experimenters that e.g. would like to investigate SDN principles. The mobile network gateway could also be realized as an SDN capable

Open Flow switch where one can reconfigure the routing tables during runtime dependent on the traffic in the LTE or WiFi networks. Using the supported Tap Bridges in ns-3 also allows to interface to external real IP networks which makes it possible to also include external traffic into the ns-3 network simulation. As this network configuration can run with the ns-3 based SDR prototyping platform described in D3.3 [22] this setup provides a powerful system where researchers can investigate various network scenarios using LTE and WiFi in an SDN like configuration not only in simulation but with real over the air transmission and internal or external traffic.

8.1.2.2 LWA/LWIP Implementation

In the previous section the overall ns-3 LTE-WiFi interworking scenario is described. As explained, one potential experimentation area would be to apply SDN principles to the mobile network gateway to split the traffic between the LTE and WiFi access networks. An alternative that is part of the LTE specification is to split the traffic within the LTE eNB and let parts of the traffic go through the LTE network while another part of the traffic is routed over the WiFi network. This is known as LWA or LWIP connection which has been integrated into the ns-3 LTE module within the Open Call 1 for Extension 3. A summary of the two main techniques is given below:

1. **LTE/WLAN Radio Level Integration (LWIP):** LTE WLAN Radio Level Integration with IPsec Tunnel to split/aggregate the data traffic above layer 2
2. **LTE-WLAN Radio Aggregation (LWA):** LTE-WLAN Radio Aggregation to split/aggregate on PDCP level

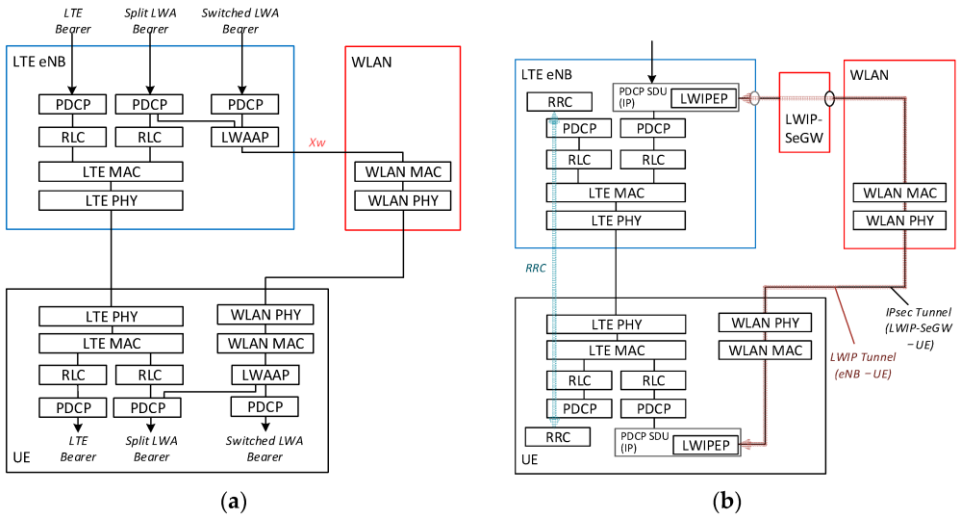


Figure 47: LTE-WLAN interworking architectures in 3GPP Rel.13: (a) LWA in a non-collocated scenario and (b) LWIP [23].

A detailed description of the specific LWA/LWIP implementation in ns-3 can be found in D7.2. After this implementation work was finished, NI integrated these two extensions in the main ns-3 LTE-WiFi interworking scenario. This is also shown in Figure 43. The LWA/LWIP extension on the LTE PDCP layer provides a new P2P link that connects the ns-3 LTE eNB with the ns-3 WiFi AP. One can select the LWA/LWIP configuration using a specific attribute in the ns-3 LTE PDCP object. For the mobile terminal this switching is transparent, i.e. at the application layer all packets will be received independent on the LWA/LWIP configuration as it is intended.

In comparison with Figure 45 and Figure 46 we now configured the LTE-WiFi interworking system with the LWA mode. Figure 48 shows the corresponding console output.

```
Use Client Server Config: 2
Number of ETH devices      = 3
Number of WiFi devices     = 3
Number of LTE devices      = 1
Router GW IP Addr          = 10.1.1.1
WiFi Net GW IP Addr        = 10.1.3.2
WiFi AP IP Addr            = 10.1.2.1
WiFI STA#1 IP Addr         = 10.1.2.2
LTE Net GW IP Addr         = 10.1.4.2
LTE EPC PGW IP Addr        = 7.0.0.1
LTE UE#1 IP Addr           = 7.0.0.2
Client IP Addr             = 10.1.1.2
Server IP Addr             = 7.0.0.2
Start simulation
Client sent 1000 bytes to 7.0.0.2 Uid: 38 Sequence number: 1 Time: 3
LWAAP: Sent packet with Sequence Number 1
Server received 1046 bytes from 20.1.2.1 Uid: 43 Sequence Number: 1
```

Figure 48: Scenario with a remote host sending to LTE interfaces of the mobile terminal in LWA mode.

As one can observe, the remote host (IP address 10.1.1.2) sends packets to the LTE mobile terminal (IP address 7.0.0.2). But in this case the LTE packet is switched by the LWAAP to be sent via the WiFi access network. This is shown by the packet source IP address 20.1.2.1 extracted at the mobile terminal server application which is the IP address of the LWAAP on the P2P link that connects the LTE eNB and the WiFi AP. Again, when comparing the packet sequence number one can see that the same packet is received as transmitted by the remote host client application. The difference in packet size stems from the addition of a PDCP specific packet header.

It is worth mentioning that the new LWA and LWIP features have also been tested and proven with the real-time NI SDR platform described in more detail in D3.3. As a result, the LTE-WiFi interworking platform is now available for the Open Call for experimentation with a variety of options to investigate different traffic scenarios as well as SDN principles including over the air transmission.

8.1.3 Testbed integration

The functionality of the developments in Year 2 will be made available in the testbed. The TestMan interface infrastructure is already deployed in the TUD testbed and missing software components such as the remote control module in ns-3 will be deployed on the testbed as part of a software update to the newest version of the Multi-RAT platform developments.

8.2 Relation to showcase

As described in deliverable D2.1 [24], the showcase for the NS-3 based prototyping platform is Showcase 4: “Interworking and Aggregation of Multiple Radio Access Technologies”.

8.3 Risk analysis

There is no specific risk foreseen at this phase of the project.

8.4 Implementation plan

8.4.1 SDR control plane improvements

The remote-control functionality of ns-3 will be further enhanced in Year 3 such that parameters needed for the specific experiments from Open Call 2 and 3 are made available for remote control. Besides that, the option to directly address specific ns-3 instances is also needed for proper control of many entities in the setup.

8.4.2 Radio slicing: resource allocation, instantiation and coordination

To further align to the ORCA vision of orchestration and SDN approaches, a goal of Year 3 is to investigate the option to use TestMan ns-3 remote-control as a simplified option to implement basic SDN capabilities.

The implementation of Dual -Connectivity for LTE as part of the Open Call 2 for Extensions will be integrated into the overall Multi-RAT networking setup.

8.4.3 Testbed integration

The functionalities for remote-control as well as the Dual-Connectivity ns-3 implementation will be made available for experiments in the TUD testbed through software updates.

9 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 2. Partners provided an overview of the architecture and functionalities of their SDR implementations, motivating the design decisions taken. The implemented functionalities included:

- TUD, NI – further enhanced direct and remote control of the mmWave Setup and Beam Steering simulation functionality
- KUL – full-duplex PHY and bi-layer CLAWS PHY-MAC architecture
- TUD – improvements on SDR control plane based on the bus system concept and MAC development
- IMEC – an improvement in control plane to reduce Tx/Rx turnaround time on ZYNQ SDR, and first integration of Linux with OFDM PHY IP cores in order to better achieve radio slicing.
- TCD – a radio hypervisor that allows the slicing and aggregation of RF front-ends, as well as the combination of both for creating dynamic and flexible radio slices.
- NI – An integration of remote-control of ns-3 via TestMan together with an enhanced ns-3 network topology for more realistic experiments into the direction of the ORCA vision

Additionally, a brief description was given on how the aforementioned implementations were integrated in the second 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 3.

10 REFERENCES

[1] NGMN Alliance, “5G white paper,” Tech. Rep., Feb 2015.
[2] J. van de Belt, H. Ahmadi, and L. E. Doyle, “Defining and surveying wireless link virtualization and wireless network virtualization,” IEEE Communications Surveys & Tutorials, vol. 19, no. 3, pp. 1603–1627, 2017.

[3] 5G America, “5G Americas White Paper Network Slicing for 5G and Beyond,” 2016.

[4] 5G PPP Architecture Working Group and others, “View on 5G architecture,” Tech. Rep., 2016.

[5] N. Sprecher, “Internet engineering task force h. flinck internet-draft c. sartori intended status: Informational a. andrianov expires: January 4, 2018 c. mannweiler,” Tech. Rep., 2017.

[6] 3rd Generation Partnership Project, “3GPP TR 28.801: Study on management and orchestration of network slicing for next generation network,” 3rd Generation Partnership Project, Tech. Rep., May 2017.

[7] K.-m. Park and C.-k. Kim, “A framework for virtual network embedding in wireless networks,” in Proceedings of the 4th International Conference on Future Internet Technologies. ACM, 2009, pp. 5–7.

[8] Michael Howard, “The evolution of SDN and NFV orchestration,” Infonetics Research, Tech. Rep., Feb. 2015

[9] ORCA Deliverable D4.1, “D4.1: First Operational SDR Platforms with end-to-end Capabilities ’’, December 2017.
[10] ORCA Deliverable D3.1, “First operational real-time SDR platforms”, December 2017
[11] CREW project http://www.crew-project.eu, Dec 2018
[12] MiWaveS, Beyond 2020 Heterogeneous Wireless Networks with Millimeter-Wave Small Cell Access and Backhauling, Website, http://www.miwaves.eu, December 2018
[13] MiWaveS, Joint Deliverable D5.2, “Status of Digital Baseband Implementation, Beamforming and Multi-User Transmission”, http://www.miwaves.eu/deliveraables/D5.2.html, Aug. 2015
[14] National Instrument, “NI USRP”, http://www.ni.com/white-paper/12985/en/
[15] Seyed Ali Hassani and Sofie Pollin, “A Low-Latency Wireless Network for Cloud-Based Robot Control” (PDF available here). CROWNCOM 2018, September 2018, Ghent, Belgium.
[16] Vermeulen, Tom, Barend van Liempd, Benjamin Hershberg, and Sofie Pollin. "Real-time RF self-interference cancellation for in-band full duplex." In Dynamic Spectrum Access Networks (DySPAN), 2015 IEEE International Symposium on, pp. 275-276. IEEE, 2015.
[17] Fred Chou, Linux wireless networking: a short walk, https://www.linux.com/blog/linux-wireless-networking-short-walk, on line, last accessed in Dec 2018.
[18] AD9361 high performance, highly integrated RF Agile Transceiver™ Linux device driver, https://wiki.analog.com/resources/tools-software/linux-drivers/iio-transceiver/ad9361, online, last accessed in Dec 2018.
[19] Git repository of Linux kernel tree on ARM, https://github.com/analogdevicesinc/linux, online, last accessed in Dec 2018.
[20] S. E. Kocabas and A. Atalar, “Binary sequences with low aperiodic autocorrelation for synchronization purposes,” IEEE Communications Letters, vol. 7, no. 1, pp. 36–38, 2003.
[21] ns-3 network simulator, https://www.nsnam.org/documentation/, Dec 2018
[22] ORCA Deliverable D3.3, “Enhanced operational real-time SDR platforms”, December 2018.

[23] 3GPP, Release 13 – 36.Series Specifications, http://www.3gpp.org/ftp/Specs/latest/Rel-13/36_series/
[24] ORCA Deliverable D2.1, “Definition of showcases”, March 2017