GRANT **AGREEMENT NO.: 732174**

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

Orchestration and Reconfiguration Control Architecture

D4.5: Final 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	31/1/2020
Submission date	31/1/2020
Deliverable lead	TCD
Version	1.0
Authors	Joao F. Santos (TCD), Yi Zhang (TCD), Seyed Ali Hassani (KUL), Achiel Colpaert (KUL), Wei Liu (IMEC), Xianjun Jiao (IMEC), Muhammad

	Aslam (IMEC), Jan Bauwens (IMEC), Ingrid Moerman (IMEC), Walter Nitzold (NI), Martin Danneberg (TUD), Ahmad Nimr (TUD), Shahab Ehsanfar (TUD), Roberto Bomfin (TUD)
Reviewers	Clemens Felber (NI), Andrea P. Guevara (KUL)

Abstract	This deliverable includes the progress and implementation results obtained in Year 3 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.
Keywords	End-to-end, SDR, control plane, radio slicing, virtualisation, reconfiguration

Disclaimer

Copyright notice

Acknowledgment

This report has received funding from the EC under the grant agreement 731274.

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	

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

EXECUTIVE SUMMARY

The adoption of Software-Defined Radio (SDR) and other forms of reconfigurable radio technologies enable the support for multiple types of communication services with distinct traffic requirements, sharing the same underlying network infrastructure. This vision is realised through network slices (NSs), which we define as isolated logical networks instantiated using heterogeneous resources, e.g., computing, transport and radio resources, spread across multiple network segments of a physical End-to-End (E2E) network infrastructure. 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, network segments, and hybrid SDR-SDN functionality, targeting services with distinct performance requirements, e.g. throughput, latency, and reliability. Using these experimental facilities, a network designer is given access to a varied range of physical layer and control plane solutions to reconfigure its own 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 reconfigurability and easier extensibility.

First, this deliverable details the end-to-end network slicing and orchestration developments, demonstrating that SDRs can realise standard-compliant Radio Access Technologies (RATs) and interface with commercial off-the-shelf devices, while still retaining their intrinsic programmability and reconfigurability characteristics, which enables the slicing of the physical SDR for creating multiple virtual radios. In addition, it shows how we can separate the E2E resource management and slicing per network segment while achieving a consistent E2E QoS for E2E NSs.

Next, this deliverable details link control mechanisms for advanced PHY, showing: real-time mmWave beam tracking capabilities for enabling high-throughput traffic on mobile devices; and the implementation of a Time Division Duplexing (TDD) Medium Access Control (MAC) scheme for Generalized Frequency Division Multiplexing (GFDM) systems, enabling multi-user operation with higher-throughput and lower out of band emissions.

Finally, this deliverable reports the advancements in the multi-RAT functionality, demonstrating: the dual-connectivity or aggregation between 4G, 5G and WiFi RATs, which facilitates experimentation on LTE-WiFi interworking; and a monitoring interface and application for assessing the performance and operation of the multi-RAT setup.

TABLE OF CONTENTS

GRANT AGREEMENT NO.: 732174 ..... **1**
EXECUTIVE SUMMARY **3**
TABLE OF CONTENTS **4**
LIST OF FIGURES **6**
LIST OF TABLES **8**
ABBREVIATIONS **9**
1 INTRODUCTION **11**
2 END-TO-END SLICING AND ORCHESTRATION **13**
2.1 Continue the integration of OFDM transceiver with embedded Linux API 13
2.1.1 Motivation 13
2.1.2 Implementation Results 13
2.1.3 Testbed Integration 17
2.2 API for concurrent IEEE 802.15.4 compliant multi-channel transceiver 17
2.2.1 Motivation 17
2.2.2 Implementation Results 17
2.2.3 Testbed Integration 18
2.3 Dynamic heterogeneous slice deployment and coordination for integrated SDR and SDN .. 19
2.3.1 Motivation 19
2.3.2 Implementation Results 19
2.3.3 Testbed Integration 21
3 LINK CONTROL FOR ADVANCED PHY **24**
3.1 26 GHz mmWave link: add simple MAC protocol over the GFDM PHY 24
3.1.1 Motivation 24
3.1.2 Implementation Results 24
3.1.3 Testbed Integration 26
3.2 GFDM MAC 27
3.2.1 Motivation 27
3.2.2 Implementation Results 27
3.2.3 Testbed Integration 30
3.3 Improve and synchronize the adaptation of AnSIC and DiSIC modules 31
3.3.1 Motivation 31
3.3.2 Implementation Results 31
3.3.3 Testbed Integration 31

3.4 Hybrid beam tracking by combining mmWave convertor with Massive MIMO testbed 32
3.4.1 Motivation 32
3.4.2 Implementation Results 32
3.4.3 Testbed Integration 32
4 MULTI-RAT CONTROL 34
4.1 LTE/5G Dual Connectivity Implementation 34
4.1.1 Motivation 34
4.1.2 Implementation Results 35
4.1.3 Testbed Integration 37
4.2 Enhanced Interface for Multi-Instance and Interworking RAT Control 38
4.2.1 Motivation 38
4.2.2 Implementation Results 38
4.2.3 Testbed Integration 41
4.3 Multi-RAT Monitoring Interface and Application 41
4.3.1 Motivation 42
4.3.2 Implementation Results 42
4.3.3 Testbed Integration 42
4.4 Ns-3 Multi-RAT Networking Examples for 5G, LTE and WiFi 42
4.4.1 Motivation 42
4.4.2 Implementation Results 43
4.4.3 Testbed Integration 46
5 CONCLUSIONS 48
REFERENCES 49

LIST OF FIGURES

Figure 1: The overall structure of the deliverable. 12

Figure 2: OpenWiFi architecture. 14

Figure 3: Our hierarchical orchestration architecture, where each network segment has its own specialised orchestrator, and we accomplish cross-network segment orchestration for deploying NSs through a new entity, the hyperstrator. 20

Figure 4: Performance trade-offs for the different types of RAN slices, with respect to the amount of allocated computational and radio resources for HyDRA and OpenWiFi, respectively. 20

Figure 5: Timing diagrams showing the interactions between the entities of our experimental setup required for deploying NSs with RAN slices created with the different radio hypervisors. 21

Figure 6: The experimental setup we developed to validate our hierarchical orchestration architecture. 22

Figure 7: Mobile mmWave link setup. 24

Figure 8: FPGA-based beam steering diagram. 25

Figure 9: Beam steering algorithm. 26

Figure 10: MAC-PHY transmitter. 28

Figure 11: MAC-PHY receiver. 28

Figure 12: Simple MAC. 29

Figure 13: Packet types. 29

Figure 14: Block Diagram for GFDM NI LTE AFW Integration Tx. 30

Figure 15: Block Diagram for GFDM NI LTE AFW Integration Rx. 30

Figure 16: Low-level MAC, the additive sub-section for synchronized analog and digital self-interference control. 31

Figure 17: Schematic overview of the complete mmWave MIMO system. 32

Figure 18: Summed Spectral Efficiency versus the number of users. Dashed and solid lines represent simulated and measured results, respectively. 33

Figure 19: Overview of Multi-RAT experimentation platform with 5G, LTE and WiFi link. 34

Figure 20: ns-3 LTE-LTE DC Implementation overview. 35

Figure 21: 5G-LTE DC in action using ns-3 on real-time SDR. 36

Figure 22: 5G-LTE DC experimentation setup using NI USRP 2974. 37

Figure 23: System Architecture of TestMan ns-3 interface with addressing options. 38

Figure 24: TestMan Server start command structure with target identifier 39

Figure 25: TestMan ns-3 interface command structure enhancements for addressing. 39

Figure 26: TestMan Server Terminal Operation with different target identifier commands. 40

Figure 27: Example implementation of LWA/LWIP control functionality within ns-3 lte-pdcp.cc source with usage of TestMan and ns-3 parameter database [19]. 41

Figure 28: General structure of the Monitoring Interface..... 42

Figure 29: LTE/WiFi/5G Network Scenario. 43

Figure 30: LTE-WiFi Interworking Example Topology in ns-3. 44

Figure 31: LTE-5G Interworking Example Topology in ns-3. 44

Figure 32: Initialization phase and IP/topology description of LTE-LTE/5G ns-3 simulation.... 45

Figure 33: Simulation flow and packet transmission of LTE-LTE/5G ns-3 simulation. 46

LIST OF TABLES

Table 1: OpenWiFi XPU register list. 15

Table 2 The bit patterns for 26 GHz mmWave beam index. 24

Table 3: GFDM PHY Frame. 27

Table 4: PLCP. 27

Table 5: PDU. 28

ABBREVIATIONS

AIFS	Arbitration inter-frame spacing
AGV	Autonomous Ground Vehicle
AnSIC	Analog Self-Interference Cancellation
AP	Access Point
BPSK	Binary Phase Shift Keying
BRAM	Block Random Access Memory
BS	Base Station
BSSID	Basic Service Set Identifier
C-RAN	Centralised-RAN
CN	Core Network
CW_MIN	Contention Window Minimum
CW_MAX	Contention Window Maximum
D2D	Device-to-Device
DALI	Dual Connectivity for ORCA
DC	Dual Connectivity
DiSIC	Digital Self-Interference Cancellation
DMA	Direct Memory Access
E2E	End-to-end
EBD	Electrical Balance Duplexer
eNB	evolved NodeB
EPC	Evolved Packet Core
GFDM	Generalized Frequency Division Multiplexing
GFDMA	Generalized Frequency Division Multiple Access
GUI	Graphical User Interface
IBFD	In-Band Full-Duplex
IFFT	Inverse Fast Fourier Transform
L1-L2	Layer 1 – Layer 2
LWA	LTE-WLAN Aggregation
LWIP	LTE WLAN Radio Level Integration with IPsec Tunnel
MAC	Medium Access Control
MeNB	Master eNodeB
MIMO	Multiple-Input Multiple-Output
mUE	Master User Equipment

NIC Network Interface Card

NP Network Provider

NS Network Slice

OFDM Orthogonal Frequency Division Multiplexing

OS Operating System

P2P Point To Point

PDB Parameter Database

PDCP Packet Data Convergence Protocol

PGW Packet Data Network Gateway

PHY Physical Layer

QoS Quality of Service

RAM Random Accessible Memory

RAN Radio Access Network

RAT Radio Access Technology

RF Radio Frequency

RLC Radio Link Control

RRC Radio Resource Control

SDN Software Defined Networking

SDR Software Defined Radio

SeNB Secondary eNodeB

SFD Start of Frame Delimiter

SGW Serving Gateway

SiC Self-Interference Cancellation

sUE Secondary User Equipment

TDD Time Division Duplex

TN Transport Network

TS Terminal Station

TSF Time Synchronization Function

TX Transmit

TXOP Transmit Opportunity

UD User Device

UE User Equipment

UUID Universally Unique Identifier

1 INTRODUCTION

This deliverable focuses on the CONTROL plane features developed in the Year 3 of ORCA. Each feature is introduced in three aspects: motivation --- why it is developed, implementation --- how it is developed, and testbed integration --- how other people can use it. The deliverable is organized as follows:

First, this deliverable reports the configurable interface offered by the high-level protocol stack of full stack SDR solutions, including (i) **Control Interface for OpenWifi** with the integration of OFDM transceiver with Linux driver, and (ii) the **Control Interface for Multi-Channel Transceiver** with the API adaptation for virtual concurrent IEEE 802.15.4 PHY. The former is used in showcase 3, where network slices are setup and configured via the control interface described here; and the latter is used in showcase 2, where multiple Zigbee transceivers are demonstrated concurrently on a single SDR with end-to-end connectivity realized upon the API described here. In addition, we also present a **Network Slice Deployment and Coordination Mechanism** with a hierarchy of control and management entities, i.e., SDR controller, local network orchestrator and hyperstrator. This framework can map service requests to these parameters to ensure a coherent network slice being setup across multiple network segments, which is the core feature demonstrated in showcase 3.

Then, this deliverable reports features related to link control for advanced PHY. More specifically, it describes the control link for **26GHz mmWave MAC** with beam steering function in real-time, and the **GFDM MAC** using TDMA for multi-user scenario. Both implementations are used in showcase 1, it demonstrates a mmWave link with beam-steering that is capable to cope with mobility of the Autonomous Ground Vehicle (AGV) while maintaining the video streaming. The control plane established upon Self-interference Cancelation (SiC) is also reported, this feature is used in showcase 2 where low latency communication is achieved.

Finally, this deliverable reports the control plane features related to the **Multi-RAT Control** platform. Some of these features are developed as further extensions to work from ORCA open call for extension subjects. The multi-RAT control framework leverages on the TestMan interface to communicate with ns-3 and it is extended for multi-instance usage scenarios. These features are partially shown in the showcase 4 demonstration.

The overall structure of the deliverable is given in Figure 1, where the color code indicates the test facilities where the corresponding features are available and which showcase utilizes the features.

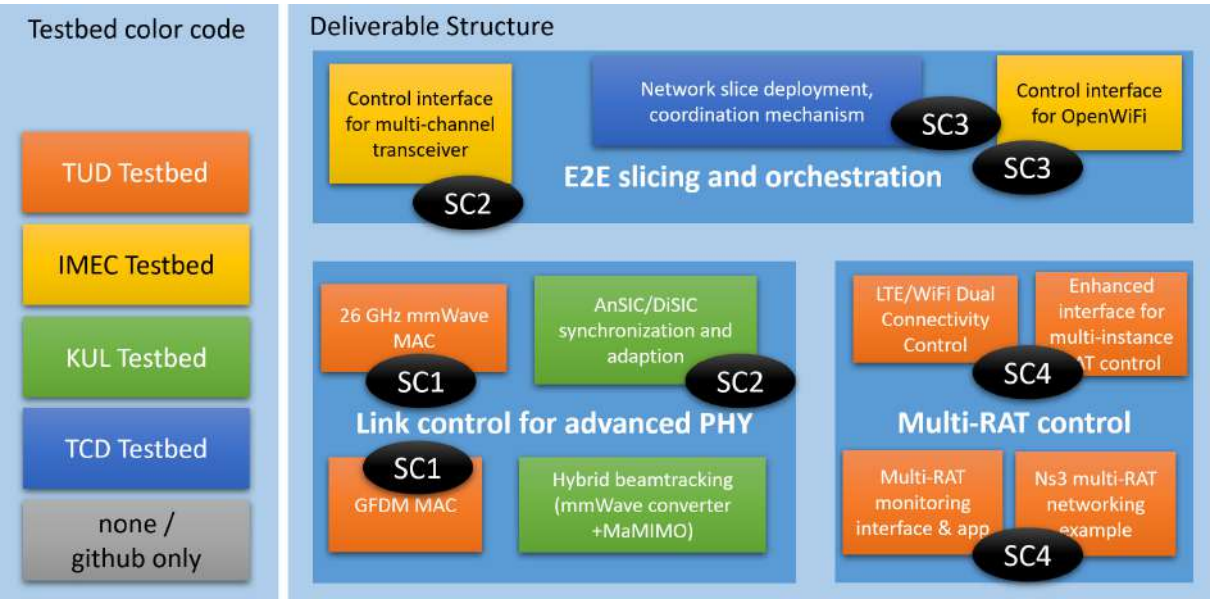

Figure 1: The overall structure of the deliverable.

2 END-TO-END SLICING AND ORCHESTRATION

2.1 Continue the integration of OFDM transceiver with embedded Linux API

2.1.1 Motivation

The work of Orthogonal Frequency Division Multiplexing (OFDM) transceiver integrated with Linux running on embedded ARM of a Software-Defined Radio (SDR) board has led to the release of OpenWiFi[1], an open source project offering full stack WiFi functionality. In order to report the implementation, we will refer to files in this repository to be clear.

Linux mac80211 subsystem [1] as a part of Linux wireless, defines a set of Application Programming Interfaces (APIs) (ieee80211_ops) to rule the WiFi chip driver behaviour. SoftMAC[2] WiFi chip driver implements (subset of) those APIs. This is why Linux can support so many WiFi chips of different chip vendors.

In addition to standard WiFi functions, OpenWiFi also supports special configuration, such as the possibility to share the radio between two network slices with configurable time share. This is why OpenWiFi offers a user space tool *sdrctl* to access openwifi specific configuration/functionalities. Note that *sdrctl* does not replace Linux Native WiFi control programs, such as ifconfig/iw/iwconfig. It is meant to be complementary. The following subsections details the implementations

2.1.2 Implementation Results

OpenWiFi driver (sdr.c) implements following APIs of ieee80211_ops:

- **tx**. It is called when upper layer has a packet to send
- **start**. It is called when Network Interface Card (NIC) is up. (*ifconfig sdr0 up*)
- **stop**. It is called when NIC is down. (*ifconfig sdr0 down*)
- **add_interface**. It is called when NIC is created
- **remove_interface**. It is called when NIC is deleted
- **config**. It is called when upper layer wants to change channel/frequency (like the scan operation)
- **bss_info_changed**. It is called when upper layer believe some BSS parameters need to be changed (Basic Service Set Identifier (BSSID), TX power, beacon interval, etc)
- **conf_tx**. It is called when upper layer needs to config/change some tx parameters (Arbitration inter-frame spacing (AIFS), Contention Window Minimum (CW_MIN), Contention Window Maximum (CW_MAX), Transmit Opportunity (TXOP), etc)
- **prepare_multicast**. Function hook is present, currently only an empty implementation.
- **configure_filter**. It is called when upper layer wants to config/change the frame filtering rule in FPGA.
- **rfkill_poll**. It is called when upper layer wants to know the Radio Frequency (RF) status (ON/OFF).
- **get_tsf**. It is called when upper layer wants to get 64bit FPGA timer value (TSF - Timing synchronization function)
- **set_tsf**. It is called when upper layer wants to set 64bit FPGA timer value
- **reset_tsf**. It is called when upper layer wants to reset 64bit FPGA timer value

[1] OpenWifi https://github.com/open-sdr/openwifi

[2] SoftMAC https://wireless.wiki.kernel.org/en/developers/documentation/glossary

- **set_rts_threshold**. It is called when upper layer wants to change the threshold (packet length) for turning on RTS mechanism
- **testmode_cmd**. It is called when upper layer has test command for us. *sdrctl* command message is handled by this function.

Above APIs are called by upper layer (Linux mac80211 subsystem). When they are called, the driver (sdr.c) will execute the referring procedures on the SDR platform. If needed, the driver will call other component drivers, like tx_intf_api/rx_intf_api/openofdm_tx_api/openofdm_rx_api/xpu_api, for assistance.

After receiving a packet from the air, FPGA will raise interrupt (if the frame filtering rule allows) to Linux, then the function *openwifi_rx_interrupt()* of openwifi driver (sdr.c) will be triggered. In that function, *ieee80211_rx_irqsafe()* API is used to give the packet and related information (timestamp, rssi, etc) to upper layer.

The packet transmission is initiated by upper layer. After the packet is sent by the driver over FPGA, the upper layer will expect a sending report from the driver. Each time FPGA sends a packet, an interrupt will be raised to Linux and trigger *openwifi_tx_interrupt()*. This function reports the sending result (failed? succeeded? number of retransmissions, etc.) to upper layer via *ieee80211_tx_status_irqsafe()* API.

The *sdrctl* command allows a user to set/get OpenWiFi low level registers or parameters. As the hardware of OpenWiFi consists of several modules, as shown in Figure 2. Each module has specific registers, that can be set or get.

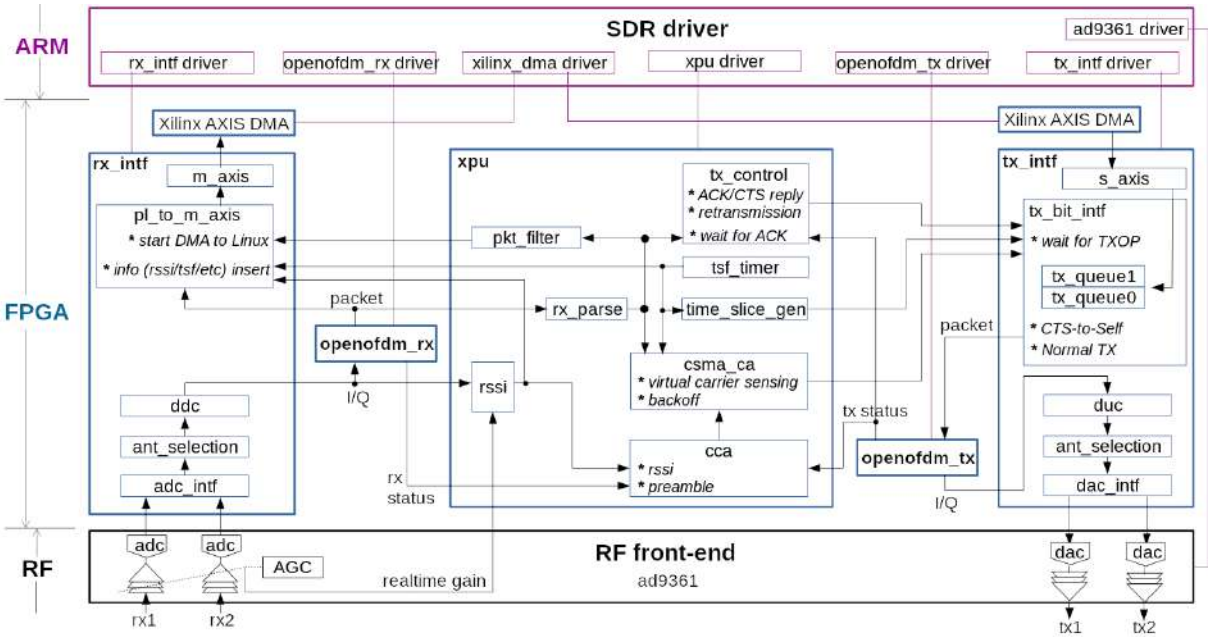

Figure 2: OpenWiFi architecture.

The following command can be used to set/get a register in a module

sdrctl dev sdr0 get reg module_name reg_idx

sdrctl dev sdr0 set reg module_name reg_idx reg_value

module_name drv_rx/drv_tx/drv_xpu refer to driver modules. Related registers are defined in sdr.h (drv_rx_reg_val/drv_tx_reg_val/drv_xpu_reg_val)

module_name rf/rx_intf/tx_intf/rx/tx/xpu refer to RF (ad9xxx front-end) and FPGA (rx_intf/tx_intf/openofdm_rx/openofdm_tx/xpu) modules. Related register addresses are defined in hw_def.h.

For instance, the module drv_rx contains 1 configurable register, it can be used to select rx antenna manually, the following command *sdrctl dev sdr0 get reg drv_rx 1 0* will configure the interface to use RX antenna 0.

The most relevant module is the XPU module, it controls low MAC behaviour of OpenWiFi, the configurable registers are listed below:

Table 1: OpenWiFi XPU register list.

Reg_idx	Meaning	Comments
2	TSF timer low 32bit write	Oly write this register won't trigger the TSF timer reload. should use together with register for high 32bit
3	TSF timer high 32bit write	Falling edge of MSB will trigger the TSF timer reload, which means write '1' then '0' to MSB
4	Band and channel number setting	See enum openwifi_band in hw_def.h. it will be set automatically by Linux. normally you shouldn't set it
11	Max number of retransmission in FPGA	Normally number of retransmissions controlled by Linux in real-time. If you write non-zeros value to this register, it will override Linux real-time setting
19	CSMA enable/disable	3758096384(0xe0000000): disable, 3:enable
20	Tx slice 0 cycle length in us	For length 50ms, you set 49999
21	Tx slice 0 cycle start time in us	For start at 10ms, you set 10000
22	Tx slice 0 cycle end time in us	For end at 40ms, you set 39999
23	Tx slice 1 cycle length in us	For length 50ms, you set 49999

24	Tx slice 1 cycle start time in us	For start at 10ms, you set 10000
25	Tx slice 1 cycle end time in us	For end at 40ms, you set 39999
27	FPGA packet filter config	Check openwifi_configure_filter in *sdr.c*. also: https://www.kernel.org/doc/html/v4.9/80211/mac80211.html #frame-filtering
28	BSSID address low 32bit for BSSID filtering	Normally it is set by Linux in real-time automatically
29	BSSID address high 32bit for BSSID filtering	Normally it is set by Linux in real-time automatically
30	Openwifi MAC address low 32bit	
31	Openwifi MAC address high 32bit	Check *XPU_REG_MAC_ADDR_write* in sdr.c to see how we set MAC address to FPGA when NIC start
58	TSF runtime value low 32bit	Read only
59	TSF runtime value high 32bit	Read only
63	Version information	Read only

More details of these register configurations are documented at https://github.com/open-sdr/openwifi/tree/master/doc.

2.1.3 Testbed Integration

Openwifi is offered as a full stack WiFi solution in ORCA opencall before its release. A dedicated tutorial (https://doc.ilabt.imec.be/ilabt/wilab/tutorials/openwifi.html) is available for people to test Openwifi on ORCA facility. Precompiled images are regularly updated on the testbed, and users have full flexibility to modify/configure OpenWiFi, since it is completely open source.

2.2 API for concurrent IEEE 802.15.4 compliant multi-channel transceiver

2.2.1 Motivation

The virtual transceiver is capable of transmitting and receiving data on up to 8 channels concurrently. The transceiver contains separate TX and RX chains, where TX digital power, and RX digital sensitivity of each channel can be configured independently. The original API towards upper layer for single PHY must be adapted in order to support the multi-channel virtual transceiver. The adaptation is mainly reflected in the header file *radio_driver.h*, and it is explained below.

2.2.2 Implementation Results

On the transmitter side, there are six radio functions, which are defined in radio_driver.h, used to set channel(s), and TX digital power of each channel, while their implementation is done in radio_driver.c file. The functions involved in data transmission using the adapted API are as follows:

1. Function *set_num_txch(int num_of_channel)* configures the number of channels on which the data is to be transmitted. For instance, *set_num_txch(4)* indicates that 4 concurrent channels are selected.
2. Function *set_tx_on_ch(int ch_idx, int on_off_flag)* turns on/off individual channels. For instance, *set_tx_on_ch(3,1)* enables the channel 3 and *set_tx_on_ch(1,0)* disables channel 1
3. Function set_tx_pow_ch(int ch_idx, int bitshift) configures power on an individual channel, for instance, *set_tx_pow_ch(3,1)* decreases the tx power of channel 3 by 6 dB as it shift the samples to the right by one bit.
4. Loading data through ring buffers. The process of loading data is the same as it is currently being done in the single channel case. However, the first byte of data should be the channel number. The format of the packet is : ch_num, data_len, payload. It is worth noting after loading a packet, an interrupt will be generated on pin 90 of the interrupt port towards the ARM processor. which should be cleared by writing to register with the following function call *xil_out8(XPAR_AXI_ZIGBEE_TX_0_BASEADDR + 0x04, 01)*;

On the receiver side, there are four radio functions defined in radio_driver.h used to set channels, and sensitivity of each channel, while their implementation is explained in radio_driver.c file. The receiver is capable of handling the data from eight channels simultaneously. The received data is stored in a Block Random Access Memory (BRAM), which has the capacity of storing the decoded data of up to eight channel.

The functions involved in data reception are as follows:

1. Function *set_num_rxch(int num_of_channel)* configures the number of channels on which data is to be received. For example, set_num_rxch(4) indicates that four concurrent channels are selected.

2. Function *set_rx_pow_ch(int ch_idx, int relative_gain)* sets the Rx digital sensitivity of the selected channel. For instance, *set_rx_pow_ch(3,1)* decreases the RX sensitivity of channel 3 by 6 dB.
3. Function *set_rx_on_ch(int ch_idx, int on_off_flag)* turns individual RX channel on or off. For instance, *set_rx_on_ch(1,0)* disables channel 1, and *set_rx_on_ch(3,1)* enables channel 3.
4. Function *Xil_in32(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08)* returns the channels from which the data has been received. The value should be interpreted as follows, when LSB is set, it indicates channel 0 has packet received, otherwise channel 0 has no data received. Another example, the value 0x02 shows that a packet is received from channel 1. This information comes from the first byte of the received packet this byte represents the channel on which the packet is received. The format of the received packet is as follows: ch_num, data_len, payload. It is possible that while Direct Memory Access (DMA) is reading data, decoded data from another channels needs to be stored into the BRAM. This will results in another interrupt and set the corresponding bit in the *XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08* register. Pay attention that each time *Xil_in8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08)* is called, the corresponding bit of the channel is automatically cleared, so there is no separate command to clear these bits. Hence user should keep reading data from DMA until the register *XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08* has 0 value

Similar to the transmitter, the receiver side has the following interrupts:

1. *Start of Frame Delimiter (SFD) interrupt* on pin 89 is cleared by writing to a register with the following function *Xil_Out8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x04, 01*); Since it is a multi-channel receiver, it can receive more than one channel's SFD at a time. Data (*data = Xil_in8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x04)*) provides information about the indexes of channel(s) which caused the interrupt, with LSB bit indicates 0th channel and so on.
2. The *received packet interrupt* ion pin 62 is cleared by writing to register with the following function *Xil_Out8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08, 01)*; Since it is a multi-channel receiver, it can receive multiple data packets at once, Data (*data = Xil_in8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08)*) provides information about the channel indexes which caused the interrupt. The register's value can be bit mapped to channels, with LSB indicates channel 0.
3. *DMA read packet interrupt* on pin 63 is cleared by the same command used for single channel. Data (*data = Xil_in8(XPAR_AXI_ZIGBEE_RX_0_BASEADDR + 0x08)*) provides the information about the channel indexes whose data is still not read from the RAM of baseband processing unit.

2.2.3 Testbed Integration

A plug and play application example for the multi-channel transceiver at MAC level is given on the testbed tutorial page[3]. Users will need to request access to specific git repositories to test the example on the ORCA facilities.

[3]w-iLab.t document for TAISC on zedboard tutorial https://doc.ilabt.imec.be/ilabt/wilab/tutorials/taisc_on_zed.html#taisc-running-on-zedzynqsdr

2.3 Dynamic heterogeneous slice deployment and coordination for integrated SDR and SDN

2.3.1 Motivation

Public and private networks are evolving and incorporating new types of wireless network segments for serving current and future use cases, e.g., with the addition of mmWave links for extreme throughput, and the expected inclusion of satellite and drone links for ubiquitous connectivity beyond 5G. Henceforth, Network Providers (NPs) will need to integrate these new network segments with their existing network deployments, and efficiently coordinate the use of heterogeneous resources across their extended E2E network infrastructure. However, we observe that the existing E2E orchestrators are not suitable for such scenarios, due to the ossified nature of their centralised E2E network management, and their coarse-grained allocation of resources on wireless network segments.

2.3.2 Implementation Results

We propose a hierarchical orchestration scheme for E2E networks, addressing the oversimplification of the allocation of resources, and the limitation on the support for network segments of existing E2E orchestration solutions, through the coordination between distributed specialised orchestrators, as shown in Figure 3. The hierarchical architecture proposed in this work enables each network segment to be independently managed, using orchestrators tailored for the particularities of individual segments. We ensure a cohesive resource allocation across network segments through a higher-level orchestrator, the hyperstrator, which coordinates the E2E resource allocation and guarantees a consistent E2E Quality of Service (QoS) for the Network Slices (NSs). To the best of our knowledge, this E2E orchestration solution is the first to decentralise both the control operation over the physical hardware infrastructure and the decision intelligence over the allocation of resources. These approaches facilitate the upgrade and replacement of the underlying orchestrators, as well as the inclusion of new network segments, resources, and orchestrators unforeseen at design time.

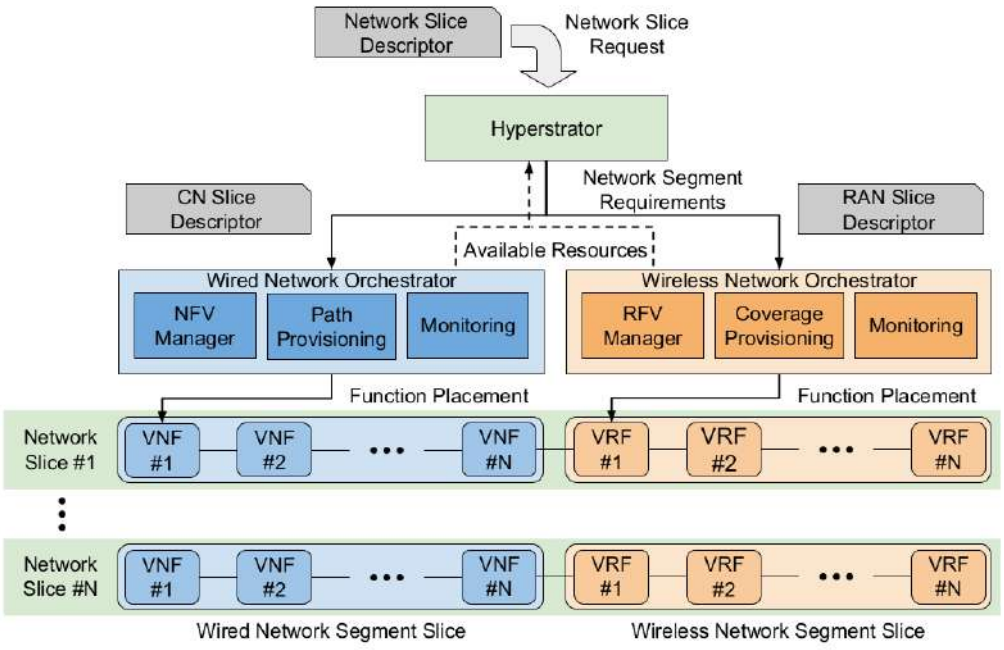

Figure 3: Our hierarchical orchestration architecture, where each network segment has its own specialised orchestrator, and we accomplish cross-network segment orchestration for deploying NSs through a new entity, the hyperstrator.

In the following sections, we assess the resource allocation trade-offs for deploying customised NSs, and the performance of our hierarchical orchestration scheme for provisioning NSs.

2.3.2.1 Creating Customized Network Slices

In this analysis, we consider the performance trade-offs on the resource allocation for deploying customised NSs. The performance of our NSs is mainly constrained by the Radio Access Network (RAN) slices, as our Core Network (CN) slices leverage Gigabit Ethernet connections, capable of delivering around 1 Gbps of throughput with less than 0.1 ms of delay. Hence, we limit our evaluation to analyse the performance of RAN slices using different radio hypervisors developed within ORCA and amount of resources. The use of different radio hypervisor over SDR enables us to create customized RAN slices using different RATs at physical layer and operate both in licensed and unlicensed spectrum. However, the performance of the different types of RAN slices may vary significantly with respect to the amount of allocated resources. Figure 4 shows the results of our measurements, in terms of delay and spectral efficiency, as different types of NPs may be allowed to use different bandwidths. Our results illustrate how the performance of RAN slices scales proportionally to the amount of allocated resources. In addition, it shows two regions of interest, implying that we can use HyDRA's LTE RAN slices for serving communication services that demand high throughput at the cost of higher latency, or OpenWiFi's WiFi RAN slices for serving communication services that demand low latency at the cost of lower throughput.

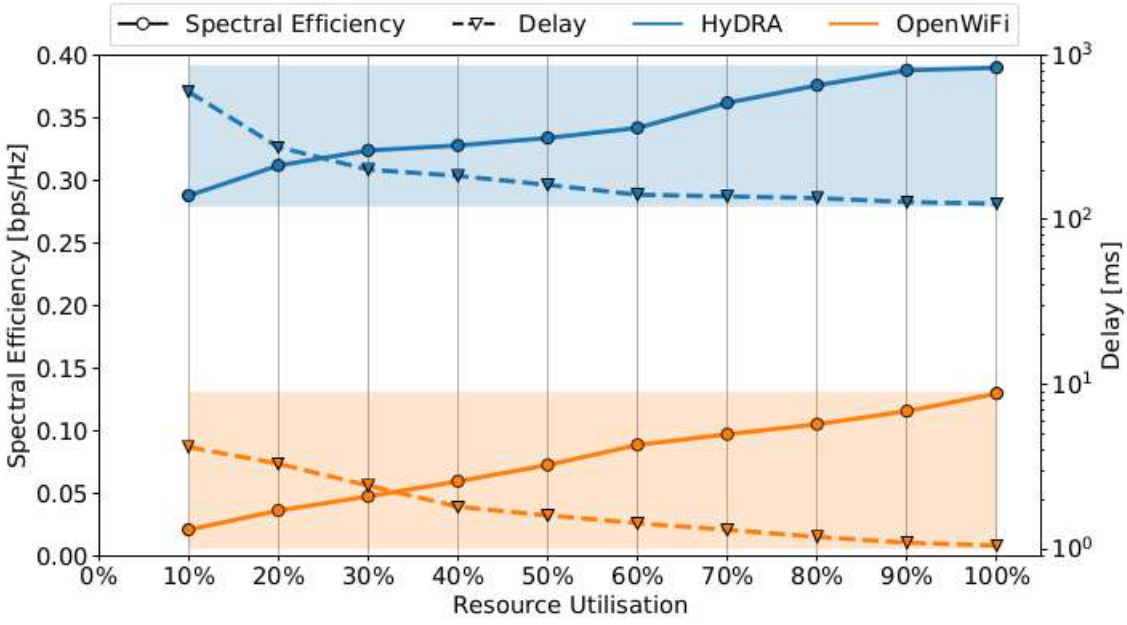

Figure 4: Performance trade-offs for the different types of RAN slices, with respect to the amount of allocated computational and radio resources for HyDRA and OpenWiFi, respectively.

2.3.2.2 Overhead for Provisioning Network Slices

In this analysis, we are interested in the interaction between the elements of our hierarchical orchestration scheme and their delay for provisioning NSs. The distributed nature of our orchestration architecture requires the communication between different entities for coordinating the deployment of NSs, which introduces an overhead in the E2E provisioning procedure. The total E2E provisioning delay is crucial for evaluating the performance of our system, as it dictates the time interval required for deploying new NSs to support communication services. This metric consists in the time interval between an SP sending an NS request, until the NS is available, which includes mapping requirements, allocating resources, and placing functions on all the underlying network segments.

We have deployed NSs with different types of RAN slices, using OpenWiFi and HyDRA, and Figure 5 shows the results of our measurements. The NS requests took less than 50 ms and 300 ms to be fulfilled by the hyperstrator, respectively. We attribute this difference to the pre-allocation of the total number of RAN slices on OpenWiFi due to constraints of its FPGA implementation, whereas HyDRA dynamically allocates RAN slices at run-time on a CPU-based platform. In both cases, the provision of RAN slices was the longest step, contributing with 82.75% and 97.15% of the provisioning delay, while the hyperstrator functionality and the communication with the distributed orchestrators accounted for roughly 1ms, adding an overhead of only 2.13% and 0.35%, respectively. These results show that the hyperstrator can orchestrate our experimental E2E setup and the current implementation of different radio hypervisors for provisioning NSs in real-time, and that our hierarchical orchestration architecture introduces an overhead that can be considered negligible in relation to the total NS provisioning delay.

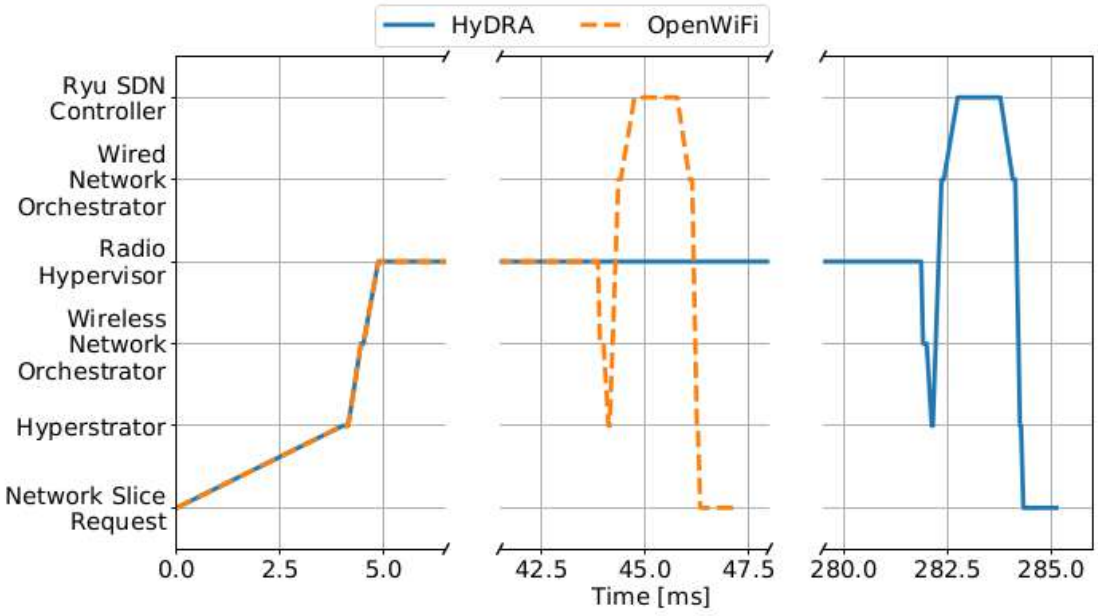

Figure 5: Timing diagrams showing the interactions between the entities of our experimental setup required for deploying NSs with RAN slices created with the different radio hypervisors.

2.3.3 Testbed Integration

We have created an experimental E2E network infrastructure managed by a hierarchy of orchestrators, as illustrated in Figure 6. We leveraged the use of Linux Containers (LXC) on a virtualisation server at TCD's Iris testbed (http://iristestbed.eu/) for realising the majority of elements in our network infrastructure, reducing the physical size and the overall complexity of our experimental setup. The virtualisation server is equipped with an Intel Core i7–4790 processor, 8 GiB of RAM, and three physical

Ethernet ports. Our experimental setup consists of the following components: a wired and a wireless network segments, each possessing a separate specialised orchestrator; a prototype hyperstrator, coordinating the deployment of NSs; and three nodes, serving as the traffic source and endpoint for the NSs. We detail the realisation of these components below.

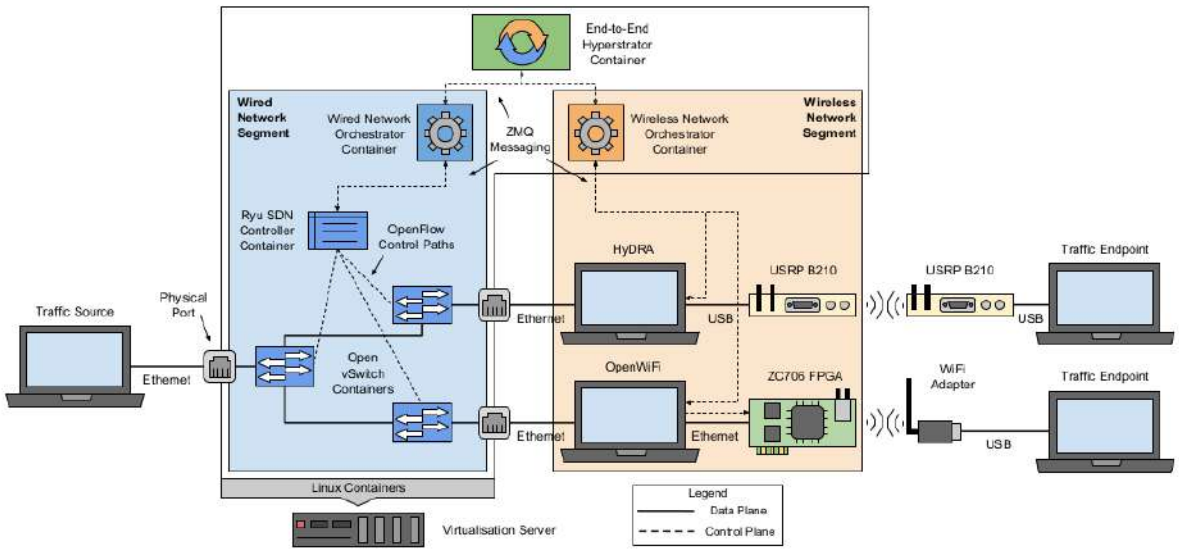

Figure 6: The experimental setup we developed to validate our hierarchical orchestration architecture.

1) Wired Network Segment: We created an SDN-based wired network segment to act as a Transport Network (TN). The data plane is composed of three software switches, implemented in the form of Open vSwitch containers, compatible with the OpenFlow protocol. Each software switch is attached to a physical Ethernet port of the virtualisation server, which is used as the ingress and egress for data plane traffic on the wired network segment. We slice this network segment by creating overlay networks with guaranteed performance using OpenFlow meters. The control plane consists of a homebrew wired network orchestrator and a Ryu SDN controller container. Our wired network orchestrator is responsible for translating the local requirements from the hyperstrator, into the low-level configuration for the switches and network segment slices, i.e., deciding the route and meter configuration; while the Ryu SDN controller is in charge of interfacing with the switches and setting up the appropriate flow rules.

2) Wireless Network Segment: We created an SDR-based wireless network segment to act as a RAN. The data plane is composed of two physical nodes, each equipped with different SDR platforms and radio hypervisors: one node is attached to a USRP B210 and runs HyDRA [2], while the other is attached to a ZC706 FPGA with an AD-FMCOMMS2-EBZ RF front-end and runs OpenWiFi [15]. These radio hypervisors leverage their respective SDR platforms for creating RAN slices isolated at the physical layer, which is not possible using commercial radios [2]. In addition, they employ distinct radio virtualisation mechanisms, allowing us to create different types of RAN slices, e.g., we use HyDRA for creating Centralised-RAN (C-RAN) LTE slices, and OpenWiFi for D-RAN WiFi slices. Each physical node has a connection to the wired network segment through an Ethernet link to the virtualisation server, which serves as the bridge between the wired and wireless network segments. The control plane consists of a homebrew wireless network orchestrator container and the two radio hypervisors. Our wireless network orchestrator is responsible for translating the local requirements from the hyperstrator, into the low-level configuration for the RAN slices, i.e., deciding which radio hypervisor to use for creating RAN slices according to the local requirements; while the respective radio hypervisor is in charge of interfacing with the physical radios and instantiating the RAN slices.

3) Hyperstrator: We developed a prototype hyperstrator, serving as the central point for managing our experimental E2E network infrastructure and instantiating NSs. It operates on a client-server paradigm, with a single server, i.e., one instance of the hyperstrator, serving multiple clients, i.e., the different SPs. We employed the Zero Message Queue (ZMQ) messaging library for both the northbound communication with SPs and the southbound interface towards the underlying orchestrators. The interaction between the SPs and the hyperstrator follows a Create, Read, Update and Delete (CRUD) model, where the SPs can: request new NSs, query information about existing NSs, modify their NSs, and tear down a given NS. The SPs specify the requirements for their NSs in the form of NS slice descriptors, which define the required E2E QoS for the NSs. Then, the hyperstrator queries the underlying orchestrators for the instantiation of network segment slices that can fulfil these service requirements.

In the current state of our implementation, the operation of our prototype hierarchical orchestration scheme is as follows.

- The hyperstrator first requests the creation of a RAN slice to the wireless network orchestrator. Based on the local service requirements, i.e., the required throughput and latency for the radio access, the wireless network orchestrator dimensions a RAN slice, deciding to use either HyDRA or OpenWiFi, and the amount of necessary radio resources, e.g., spectrum bandwidth or airtime. If there are enough radio resources available, the given radio hypervisor instantiates the RAN slice and attributes an IP address to it, serving as a wireless interface for communicating with the respective traffic endpoint.

- Then, the hyperstrator queries the wired network orchestrator for the creation of a TN slice between the traffic source and the new RAN slice. Based on the IP address of the RAN slice and the local service requirements, the wired network orchestrator dimensions the TN slice, deciding the routes, and the amount of necessary transport resources, i.e., flow and meter entries. If there are enough transport resources available, our Ryu SDN controller instantiates the TN slice, serving as a link between the traffic source and the wireless network segment.

- Upon successful creation of the two network segment slices, the traffic source can communicate with one of the traffic endpoints through an isolated and customised NS, which the hyperstrator identifies using an Universally Unique Identifier (UUID). Finally, the hyperstrator returns the UUID of the new NS to the SP, which can use this information to modify or remove the given NS.

3 LINK CONTROL FOR ADVANCED PHY

3.1 26 GHz mmWave link: add simple MAC protocol over the GFDM PHY

3.1.1 Motivation

One targeted implementation of TUD for year 3 was the real-time beam steering functionality for the mmWave frontends described in D3.5 [3]. While D3.5 focused on the receiver part of the algorithm based on energy detection, D4.5 focus on the transmitter beam selection. To this end, the mmWave receiver must exchange information about the quality of the mmWave channel with the transmitter, such that there is full degree of freedom to select a best beam pair. Thus, this section describes a simple MAC layer control protocol that works at a reliable low latency sub 6 GHz wireless link.

3.1.2 Implementation Results

3.1.2.1 Control Channel

The setup for this functionality is shown in Figure 7. The mobile user device sends its data in the uplink through the mmWave band, and the downlink is used to send control information from the Access Point (AP) to the user device at a sub 6 GHz link. In the following, we describe the control channel link.

Figure 7: Mobile mmWave link setup.

The goal of the control channel in the downlink is to send the information to the User Device (UD) about the transmitter beam. That is, the AP runs beam steering algorithm, and informs the UD which beam it should use. In order to have a simplest solution as possible, we use a regular GFDM PHY frame to send the control information over this channel. Currently, both uplink and downlink should have the same GFDM configuration at PHY level. Therefore, the control channel configuration will depend on the uplink configuration. There are 16 possible beams, which means that we need to send four bits of information to the UD, as in the table below.

Table 2 The bit patterns for 26 GHz mmWave beam index.

Beam index	Bits
1	0 0 0 0
2	0 0 0 1
3	0 0 1 0

⋮	⋮
16	1 1 1 1

First, we create a byte whose four least significant bits are mapped from the desired transmission beam, while the most significant bits are made zero. Normally, the payload size of the GFDM PHY is considerably larger than 1 byte. Then we append this byte several times until the amount of required bytes is fulfilled.

3.1.2.2 Beam steering algorithm

Figure 8 shows the diagram of TUD's beam steering FPGA implementation. For simplicity, we always perform an exhaustive search approach for selecting the RX beam. As it was described in D3.5 [3], we take 62,5 μs to probe one beam combination. The scheme is depicted in Figure 9. If a give channel provides energy above a certain threshold, then this beam pair is kept until the channel changes. If all receive beams are probed and the channel remains not suitable for communication, the TX beam is sent over the control channel described in the last subsection. Notice that it takes about 62,5μs*16 = 1ms to probe all receive beams. Thus, in order to keep a reasonable time margin to guarantee that all RX beams are probed for the same TX beam, we set the TX interval of 1,25 ms. This configuration has demonstrated to be sufficient for beam tracking in a mobility scenario. However, we highlight that this parametrization is not fully optimized and will be subjected to further research in TUD.

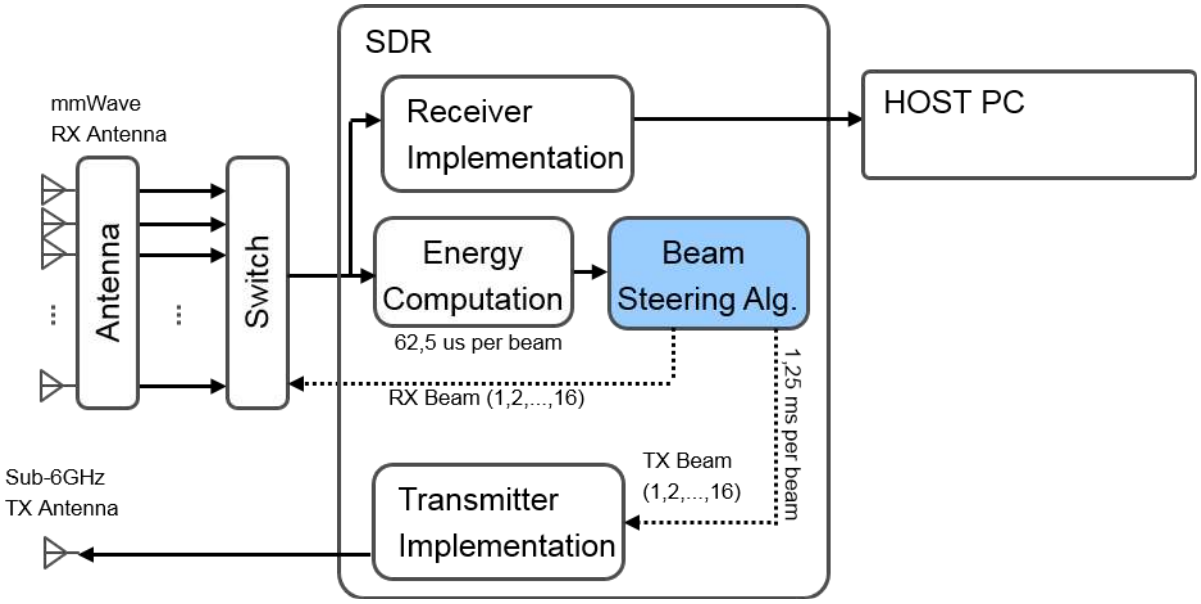

Figure 8: FPGA-based beam steering diagram.

If we consider a fully exhaustive search based algorithm, we can compute the maximum beam tracking time as 1,25 ms * 16 = 20 ms. Alternatively, we developed a faster algorithm that does not probe the TX beam sequentially in an exhaustive search mode. The new solution searches the neighbouring beams of the last successful TX beam, as shown in the bottom left sub block of Figure 9, which depicts the beam steering algorithm. In short, after an interval of 1,25 ms, the TX beam is changed to $< i_{tx\ last} +$

$n >_{16}$ or $< i_{tx\ last} - n >_{16}$, if the amount of beam trials is odd or even, respectively. Therefore, under the assumption that the mobile UD moves slow enough such that the next successful TX beam is a neighbor beam of the last successful beam, we find that the searching time decreases to 1,25 ms *2 = 2,5ms. For low-latency applications that does not require ultra-reliability such as video transmission, TUD's beam tracking algorithm has been shown to be sufficient.

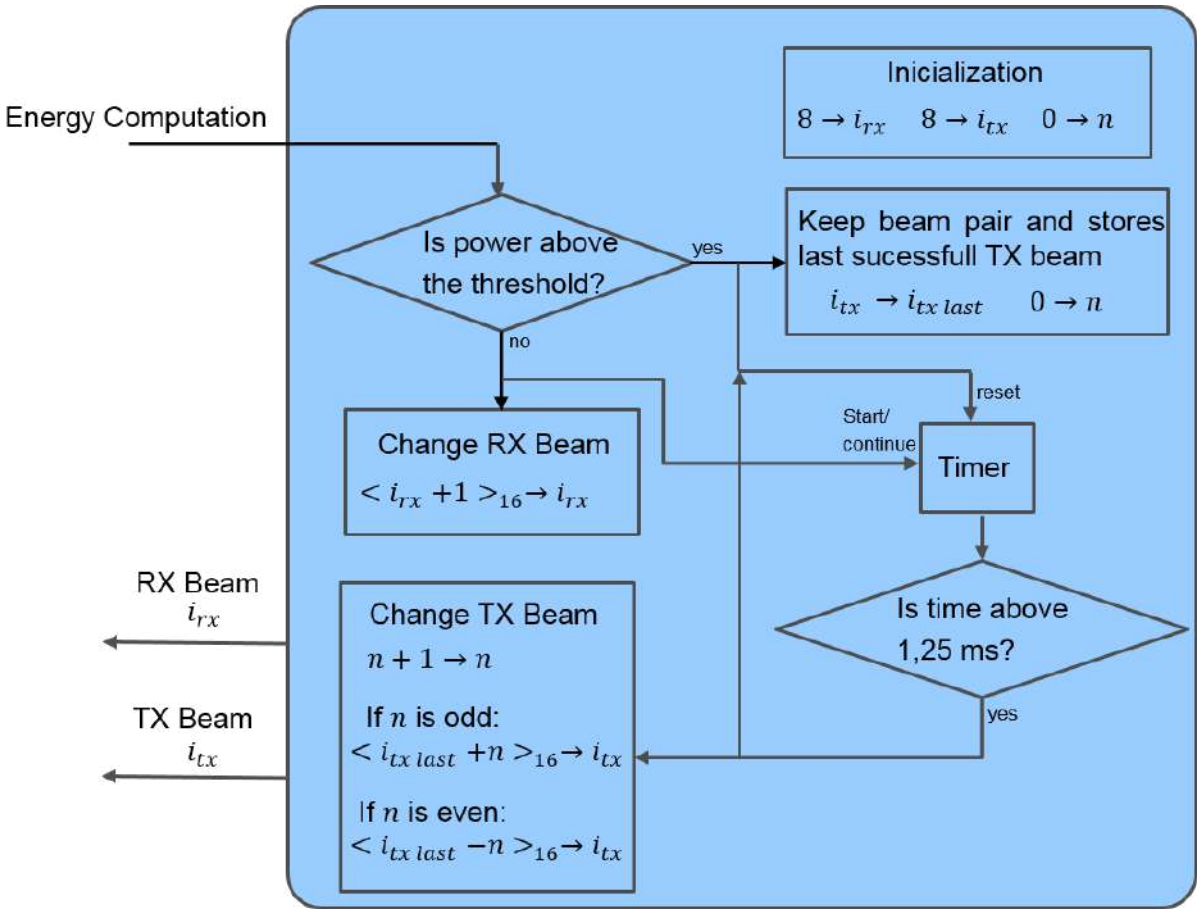

Figure 9: Beam steering algorithm.

3.1.3 Testbed Integration

Further information and source code of the development presented in this section can be found in the testbed's webpage http://owl.ifn.et.tu-dresden.de/orca/mmwave26ghz/.

3.2 GFDM MAC

3.2.1 Motivation

One relevant aspect of TUD's implementation is to allow multiple users to share the medium. This capability allows the design of more realistic experiments with several user terminals, which is suitable for emulating an automated factory with TUD's SDR's. Therefore, this section describes the developments of TUD in the context of multi-user. In particular, the following sections describe the TDMA MAC implementation, as well as the integration with the NI LTE Application Framework [4].

3.2.2 Implementation Results

3.2.2.1 Integration of TDMA MAC with GFDM transceiver

To exploit the flexibility of the GFDM transceiver, TUD developed a PHY frame shown in Table 3, which consists of: a preamble, a Physical Layer Control Protocol (PLCP) and a protocol data unit (PDU).

Table 3: GFDM PHY Frame.

	Modulation	Function
Preamble	Zadoff–Chu sequence: 128 samples + 32 CP	Synchronization and channel estimation
PLCP	OFDM : 128 subcarriers + 32 CP BPSK, ½ code rate	To carry information about the PHY configuration
PDU	Flexible GFDM	Payload of MAC protocol

The PLCP is 6 bytes defined as in the Table 4 below.

Table 4: PLCP.

Field Name	Length [bits]	Function
MCS	4	Coding and modulation options
KM	4	K (subcarriers), M (subsymbols) parameters
K-set	8	Selection of active subcarriers relative to K
PS	4	Pulse shape and precoding options
M_set	2	Indication to the range of active subsymbols
CPL	2	CP-Length options
Length	15	PDU length [byte]
CRC	3	CRC of the header
Tail	6	Tail bits required for convolutional encoder

The PDU is of variable length and contains the fields shown in Table 5:

Table 5: PDU.

Field Name	Length [Bytes]	Function
Header	Protocol-dependent	Addressing, and control information
Payload	Variable length	Data from upper layers
CRC	2	PDU CRC
Tail	variable	Zeros padding for the encoder

At the transmitter, the MAC layer decides which PHY parameters to use, depending on the current channel and the application requirements. The GFDM transmitter is configured to modulate the PDU, and at the same time the PLCP encoder generates the OFDM symbol corresponding to the PLCP header. The preamble, PLCP signal, and the payload signal are multiplexed prior to transmission, as shown in Figure 10.

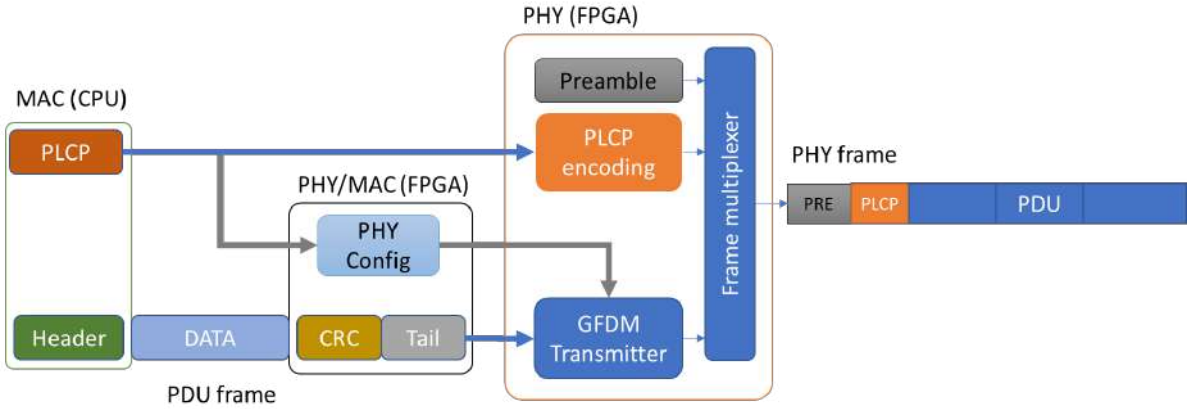

Figure 10: MAC-PHY transmitter.

At the receiver, Figure 11, after detecting a new frame, the synchronization forwards the OFDM symbol carrying the PLCP header to the PLCP decoder. The later samples are buffered until the PLCP is decoded and the PHY is configured. Afterwards, the buffered samples are processed by the GFDM receiver. The CRC block checks the PDU packets and forwards the correct ones to the MAC module.

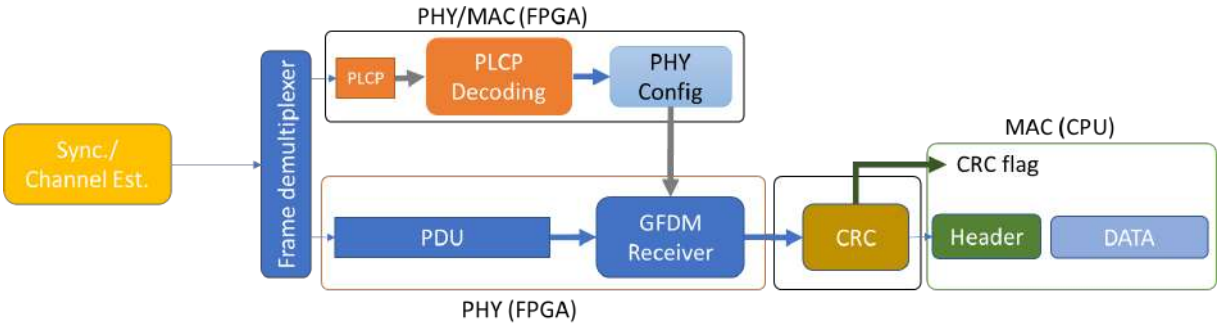

Figure 11: MAC-PHY receiver.

This architecture provides a flexibility to integrate different types of standard and customized MAC protocols.

3.2.2.2 Simple MAC implementation:

For the demonstrator presented in D3.5 [3], TUD implemented a simple MAC protocol for low-latency communication of multiple users. This MAC supports Device-to-Device (D2D) communication as well as routing via the Base Station (BS). Using 5 bit addressing length, this simple MAC can address up to 30 nodes, where ID(30) is reserved for scheduling, and ID(31) for broadcasting.

Figure 12: Simple MAC.

As illustrated in Figure 12, the MAC header consists of

- Frame ID: Which is used to detect repeated frames.
- Direction: To distinguish the node type, UE (0), BS (1).
- Sender ID: The sender address, and it is also used in the polling request.
- Target ID: The receiver address. It is set to 30 for scheduling and to 31 for broadcasting.
- The data length is fixed, but it is reconfigurable.
- Tail: It possesses a 6 bit-length, where ½ convolutional encoder is used.

Figure 13 shows different types of packets used in the MAC protocol. The BS polls the users in a deterministic round robin scheduling approach depending on the latency requirements. Thus, time-critical links are scheduled more often. The polling request is achieved by sending a polling packet from the BS, Direction (1), with no data and the ID of the requested UE is set in the Sender ID filed, whereas the Target ID is set to the scheduling ID (30). For D2D communications, two devices communicate using D2D packets. For indirect communication, the sender sends UL packet with its ID and the target address is (0). The BS then routes the data to the corresponding target by means of DL packet, where the target ID is determined form a routing table.

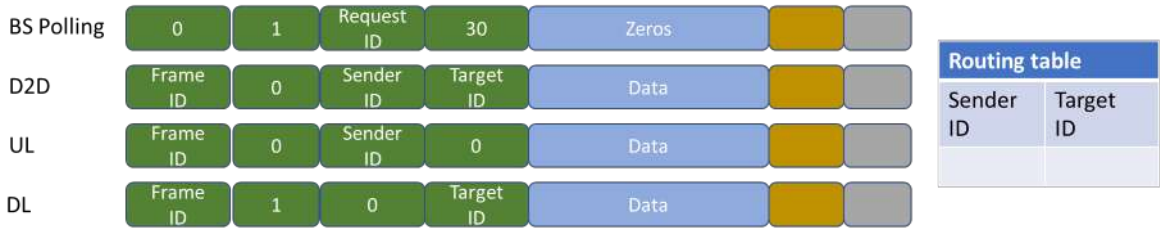

Figure 13: Packet types.

The acknowledgment is avoided by relying on the link stability, which is achieved by sending multiple repeated packets. Accordingly, the frame ID is used to eliminate repeated packets in addition to its functionality in disassembling/assembling larger packets from the upper layers.

3.2.2.3 Integration of GFDMA into the NI LTE Application Framework

In ORCA deliverable D3.5 [3] a new Generalized Frequency Division Multiple Access (GFDMA) scheme based on the research from TUD is described. The new GFDMA scheme performs the allocation after the small size GFDM modulation and then maps the samples from different users to the final GFDM vector. In the LTE standard, the data samples from different users are mapped to the transmitting vector based on the resource block allocation, which is also implemented in the LTE Application Framework from NI [4]. Considering the fact, that one can treat GFDM modem as a pre-coder to add 5G features to the LTE data samples, such as reconfigurable subcarrier spacing, the integration of GFDM modem into LTE Application Framework has been made.

Figure 14 and Figure 15 represent the block diagram of the integration in the downlink LTE at the transmitter and receiver.

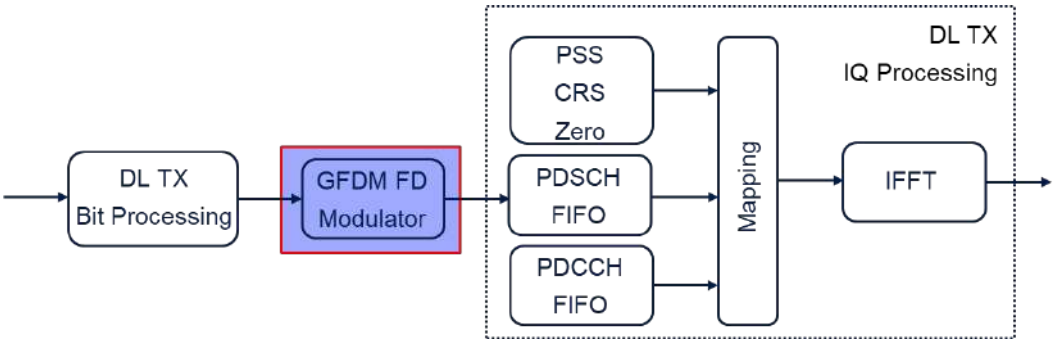

Figure 14: Block Diagram for GFDM NI LTE AFW Integration Tx.

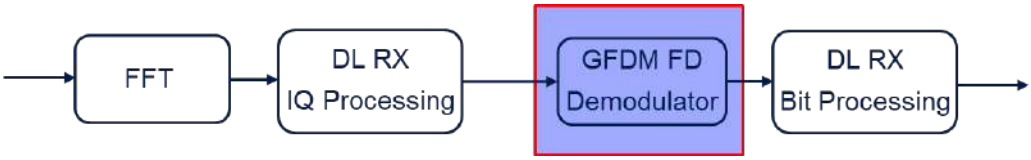

Figure 15: Block Diagram for GFDM NI LTE AFW Integration Rx.

As shown in the figures, the GFDM frequency domain modem is integrated before the resource mapping module at the transmitter and after the resource demapping module at the receiver. Since the LTE Application Framework performs the same modulation scheme as in the downlink, the same integration of GFDM has also been made in the uplink. Based on the research results from TUD, the small size GFDMA scheme has better spectral efficiency. The size of the GFDM modulation is then set based on the PRB size of LTE. Furthermore, the PRB allocation of LTE map the data samples from different users to the final vector before the Inverse Fast Fourier Transform (IFFT) transformation, which transforms the frequency domain GFDM block back into the time-domain. With this procedure, the data samples are enhanced with GFDM features.

3.2.3 Testbed Integration

The development shown is this section is available in the testbed, and the multi-user system can be accessed via the public git repository, https://fusionforge.zih.tu-dresden.de/projects/flexiblegfdmphy/. Documentation and references regarding the GFDM development are also found in the testbed webpage http://owl.ifn.et.tu-dresden.de/GFDM/.

3.3 Improve and synchronize the adaptation of AnSIC and DiSIC modules

3.3.1 Motivation

The KUL's In-band Full-duplex (IBFD) platform benefits from analog and digital self-interference cancelation modules, Analog Self-Interference Cancellation (AnSIC) and Digital Self-Interference Cancellation (DiSIC), respectively. In D5.3, ORCA presents a radar-communication (Rad-com) system which delivers both communication and radar-like functionality, integrated into one platform [5]. In such a platform, an advanced control mechanism is needed to guarantee adequate Tx-Rx isolation required for bi-directional in-band send and receive as well as environmental sensing.

3.3.2 Implementation Results

To achieve the functionality mentioned above, we have employed the KUL's flexible MAC-PHY architecture, in which the straightforward functionalities of the MAC protocol are hard-coded into the FPGA [6]. Figure 16 shows the enhancement on top of the KUL's low-level MAC, that enables synchronized joint analog and digital self-interference cancelation.

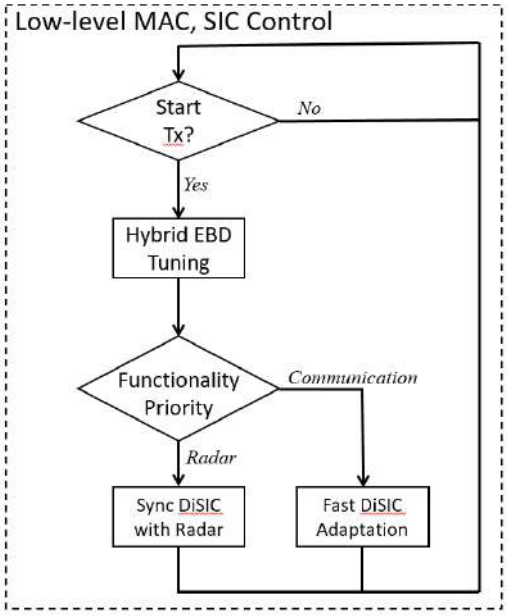

Figure 16: Low-level MAC, the additive sub-section for synchronized analog and digital self-interference control.

As shown in the figure above, the mechanize starts with the packet transmission procedure. The low-level MAC then triggers the hybrid Electrical Balance Duplexer (EBD tuner, represented in D5.3, and adjusts the adaptation rate of the DiSIC module regarding the priority of the functionality. If the application demands low-SNR communication, it adjusts the maximum tuning rate to the DiSIC module. Otherwise, the algorithm freezes the DiSIC's coefficients, allowing the detection of Dopplers with slower frequencies [7].

3.3.3 Testbed Integration

The adaptive AnSIC/DiSIC tuning is integrated into the KUL IBFD testbed (https://www.esat.kuleuven.be/telemic/research/NetworkedSystems/projects/in-band-full-duplex),

allowing four communication nodes to perform IBFD networking experimentation in a non-stationary environment.

3.4 Hybrid beam tracking by combining mmWave convertor with Massive MIMO testbed

3.4.1 Motivation

The target of KUL was to build a system capable of serving multiple users simultaneously over a mmWave wireless link and verify the capabilities of such a system. As KUL already has a Multi-User Multiple-Input Multiple-Output (MIMO) system at our disposal and because TUD provided us with an analog mmWave multi-beam antenna array, we were able to extend the digital MIMO system to a Multi-User mmWave MIMO system. In the following section, this extension is explained.

3.4.2 Implementation Results

The digital MIMO system of KUL is combined with the analog multi-beam antenna array provided by TUD [8]. This array uses a high-dimension 16x16 Butler matrix to obtain 16 beam directions. In our setup we consider a downlink transmission in a single-cell multi-user mmWave MIMO system. The base-station has 16 digital chains and connects each of them the input ports of the multi-beam array. A user has only one digital chain and connects it to one of the ports of the multi-beam array by using a 16-port RF switch. A schematic overview of the base-station and the user can be seen in Figure 17 [9].

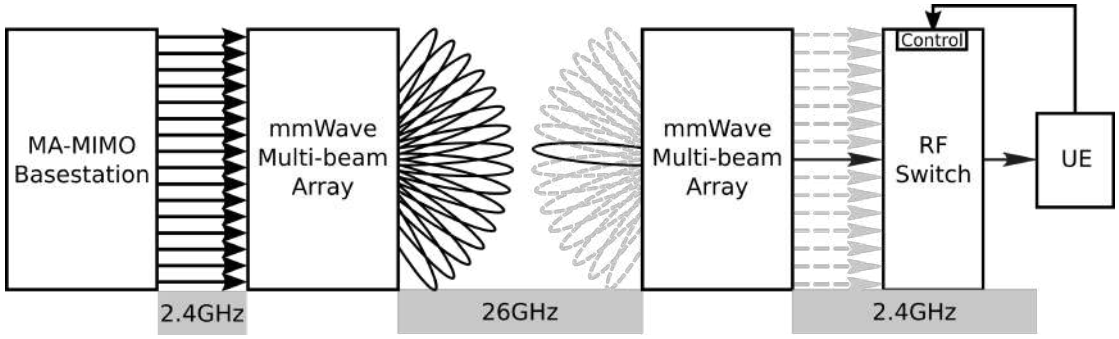

Figure 17: Schematic overview of the complete mmWave MIMO system.

A system model of the architecture is derived and is used to simulate the system's performance. The model is mainly based on existing channel models, but introduces the measured beam pattern of the multi-beam array as to approach the behaviour of the real system. As for the digital part, three different MIMO precoding techniques are considered: maximum ratio, zero forcing and regularized zero forcing. Maximum ratio maximizes the received signal to the user disregarding the interference created at the location of other users. In contrary, zero forcing attempts to minimize this interference. Similarly, regularized zero forcing tries to reduce not only interference but also noise. All previously mentioned techniques are then compared with basic analog beamforming to evaluate the performance gained over a solely analog system. The work in [9] describes this model in full detail.

3.4.3 Testbed Integration

To confirm the model, both simulations and measurements are performed. The simulations consider a number of users up to 16 at random positions in front of the base-station. The performance is evaluated using the spectral efficiency. This can be calculated using channel estimations obtained by the system

model. As for the measurements, the multi-beam array was connected to the testbed, users were placed at random angles in front of the base-station and their channel was measured. Similar to the simulations, spectral efficiency is calculated, but this time by using the measured channel. As a result, both simulations and measurements can be compared, and the validity of the system model can be evaluated. The final results of the integration can be found in [9].

Finally, the spectral efficiency values were compared and showed that the system model was a realistic approximation of the system. Furthermore, the results show that use of digital precoding significantly improves the performance of the analog system, when compared to the case with only basic analog beam selection. The results can be seen in Figure 18. The full explanation and conclusion are described in [9].

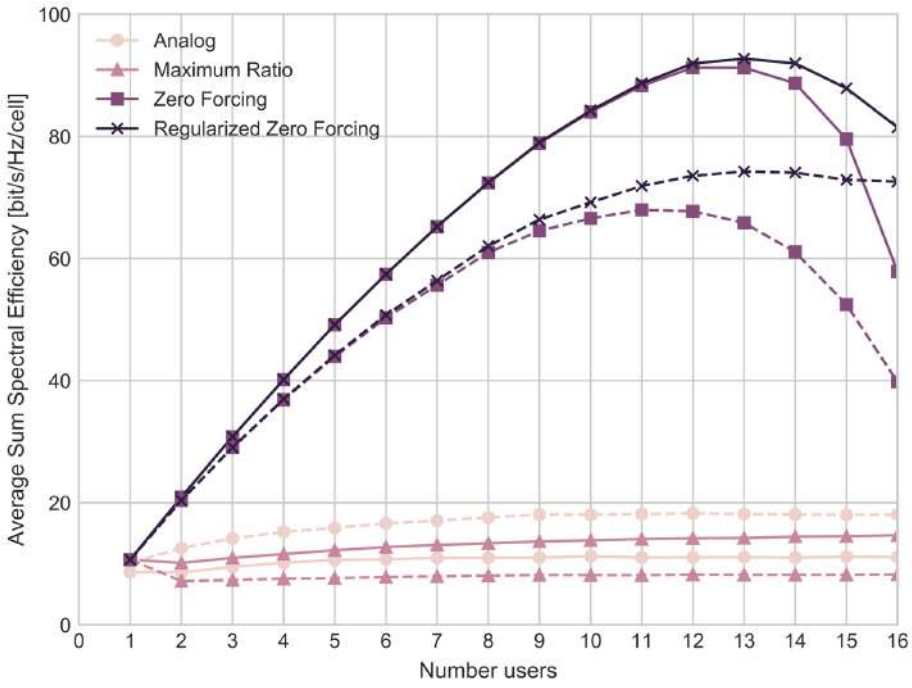

Figure 18: Summed Spectral Efficiency versus the number of users. Dashed and solid lines represent simulated and measured results, respectively.

4 MULTI-RAT CONTROL

The finalization of the Multi-RAT experimentation platform from NI in Year 3 of the ORCA project incorporated different functionalities to accommodate for flexible control of the different RATs.

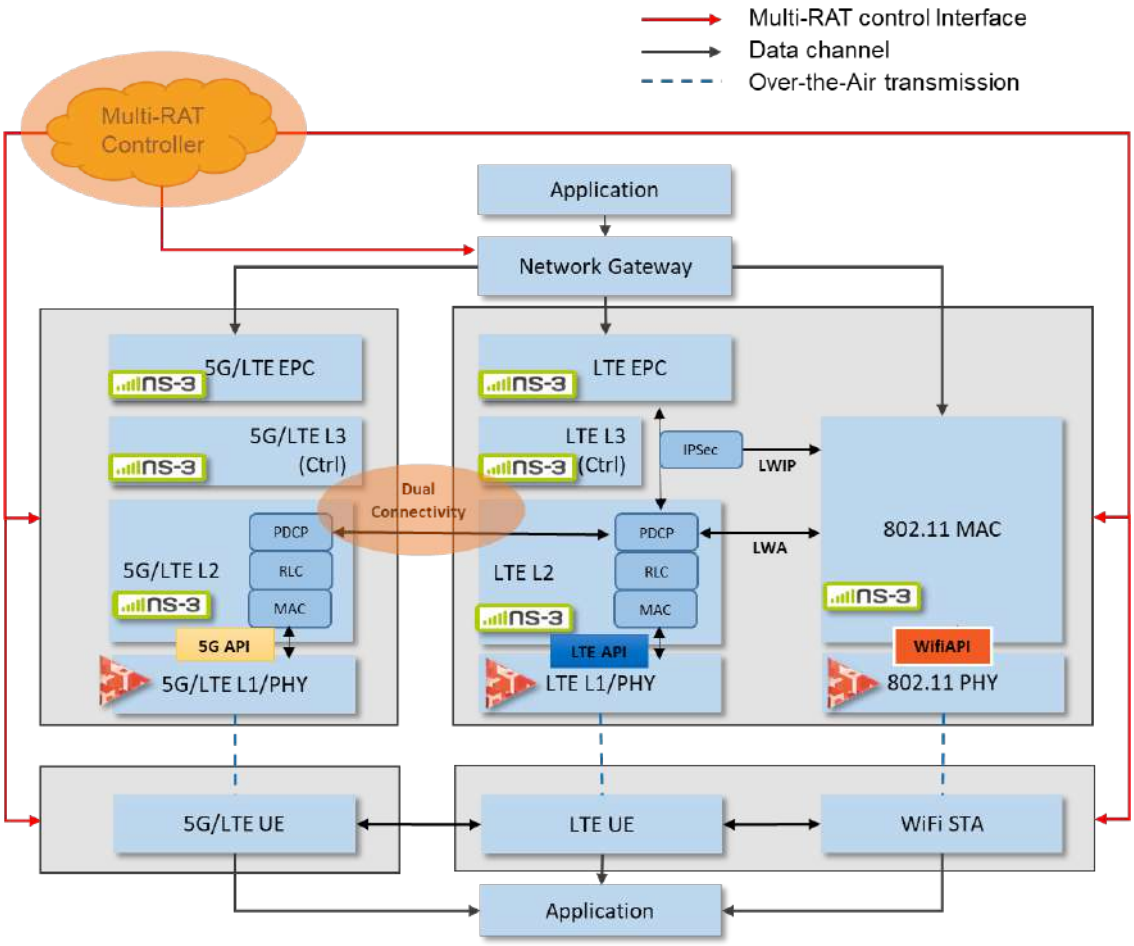

Figure 19: Overview of Multi-RAT experimentation platform with 5G, LTE and WiFi link.

The overall system diagram of the Multi-RAT platform is shown in Figure 19. The focus on the enhancements of the control plane for Multi-RAT functionality incorporated the implementation of control functionality for the Dual Connectivity (DC) feature. This feature was integrated from the Open Call 2 for Extension efforts in the DC for ORCA (DALI) project [10]. As the whole setup became more complex and more nodes and machines were incorporated, the Multi-RAT control interface needed to account for this increased complexity. Therefore, the option to address different ns-3 entities in the network with the TestMan control interface [11] was implemented.

4.1 LTE/5G Dual Connectivity Implementation

4.1.1 Motivation

A primary goal of the Multi-RAT platform within ORCA is not only to give experimenters the chance to do experiments on different RATs jointly in a common network topology but also to exploit and investigate options for interworking between different radio access technologies. As part of this effort,

the LTE-WiFi interworking technology LWA/LWIP (LTE-WLAN Aggregation/LTE WLAN Radio Level Integration with IPsec Tunnel) was implemented in conjunction with Open Call 1 for Extensions in Year 2 [12]. The focus in Year 3 of the ORCA project was on interworking technologies between two LTE stacks, namely DC which is specified in 3GPP Release 13 [13]. This functionality was implemented as part of the Open Call 2 for Extensions.

4.1.2 Implementation Results

An overview diagram of the implementation is show in Figure 20.

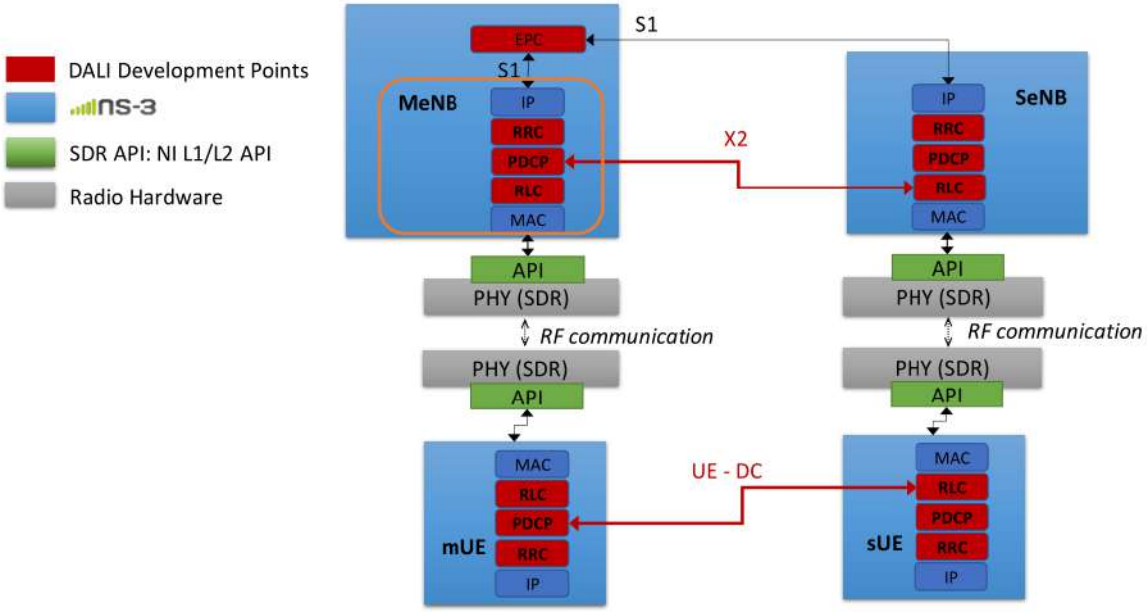

Figure 20: ns-3 LTE-LTE DC Implementation overview.

The figure shows the DC implementation done for the ns-3 [14] based Multi-RAT platform in conjunction with the DALI extension from Open Call 2 for Extensions. The implementation is focusing on LTE-LTE DC. The system consists of a Master eNB (MeNB) and a secondary eNB (SeNB) as well as the corresponding master UE (mUE) and secondary UE (sUE). The master eNB is directly connected to the Evolved Packet Core (EPC) and the DC transmission is initiated from the master eNB. Incoming data is then split within the Packet Data Convergence Protocol (PDCP) layer to be either forwarded through the stack of the master eNB or over the X2 interface towards the Radio Link Control (RLC) layer of the secondary eNB. If data is send via X2 to the secondary eNB, it will be send through the stack and over-the-air to the secondary UE which then will forward the data from RLC back into the PDCP of the master UE using the UE-DC link such that the data can be delivered to upper layers and application of the master UE. As the PDCP of master UE supports packet reordering, the overall transmission of data from master eNB to master UE via a DC split bearer transmission should be transparent to the end user.

While the functionality from the Open Call was based on a DC implementation between two LTE eNBs, the Multi-RAT platform was additionally enhanced to support DC transmission between LTE and the newly integrated 5G link (see [15]). The aforementioned connection between two stacks is shown in Figure 19 between the LTE stack and the second cellular stack that could be either LTE or 5G. The current design of the 5G link in the Multi-RAT platform assumes that only adaptions to PHY and MAC layer were made while higher layers are used from the LTE module. Therefore, the functionality for DC follows the paradigms and implementation of the implementation for the LTE-LTE case.

An exemplary transmission of client/server application over an LTE-5G DC Setup is shown and explained as follows:

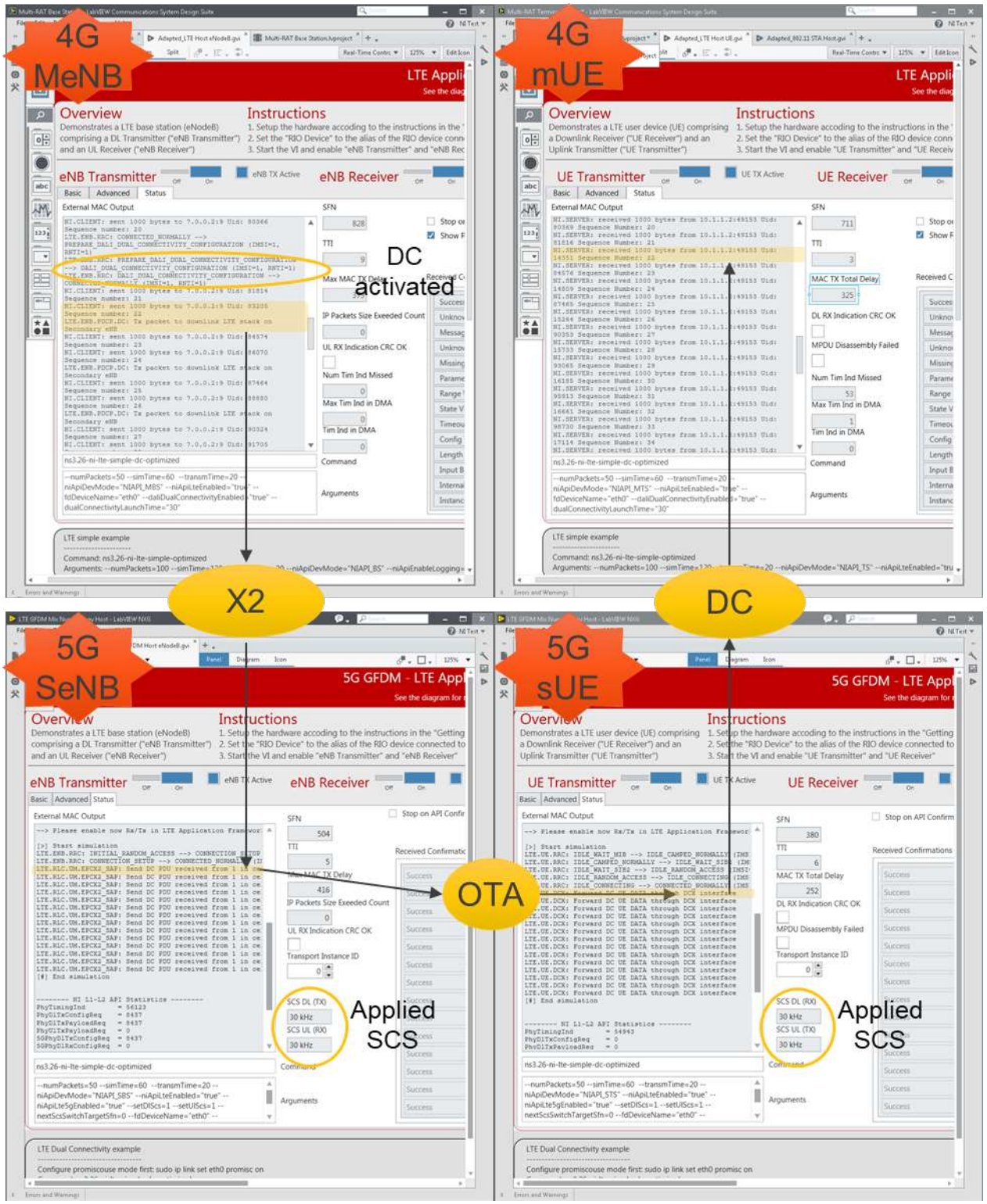

Figure 21: 5G-LTE DC in action using ns-3 on real-time SDR.

Figure 21 shows the graphical user interface (GUI) of the 5G-LTE DC real-time software defined radio (SDR) setup. The upper part of this figure represents the master link with MeNB and mUE which is based on 4G LTE. The secondary link which is based on 5G GFDM explained in detail in ORCA

deliverable D3.5 [15] with SeNB and sUE running on a subcarrier spacing configuration of 30 kHZ is shown in the lower part of this figure. The External MAC Output windows within each node showing the console output of the 4 ns-3 instances running on 4 dedicated SDRs. The hardware setup for this experiment can be seen in Figure 22.
After successful connection setup procedure DC is activated which can be seen in console output of 4G MeNB. In this setup a split bearer configuration is used which forwards every second packet over the DC link. This is a typical scenario where DC functionality is used for data aggregation. In the following the focus is put to the first DC packet with packet number 22. It is transferred to the 5G SeNB using the S1 interface which is implemented as an ns-3 EmuFdNetDevice [16]. This ethernet based interface is an association of an ns-3 device with a physical device in the host machine (CPU within NI USRP 2974). Now the 5G SeNB transmits the packet over-the-air (OTA) to the 5G sUE. Finally, 5G sUE sends the packet to 4G mUE over the DC interface which is also implemented as an ns-3 EmuFdNetDevice. For the 4G mUE the packet reception is completely transparent as already described. Due to split bearer configuration it receives 50% of the packets over the secondary DC based link and 50% over the master link which leads to doubling the throughput.

Figure 22: 5G-LTE DC experimentation setup using NI USRP 2974.

4.1.3 Testbed Integration

The DC functionality for LTE-LTE communication was made available to experimenters in conjunction with Open Call 3 for Experiments where potential experimenters could use this implementation to design experiments for LTE-LTE interworking. The enhancements for DC are also available as part of the Multi-RAT experimentation platform (see [17]) and can therefore be directly deployed in the OWL testbed [18] on suitable hardware.

4.2 Enhanced Interface for Multi-Instance and Interworking RAT Control

4.2.1 Motivation

The Multi-RAT platform in its final stage consists of three different radio access technologies: 5G, LTE and WiFi. As the setup includes a BS and a user equipment for each of these radio access technologies, there is a strong demand for a flexible configuration of the whole platform and its subcomponents. In Deliverable D4.3 [19], the TestMan interface towards ns-3 was introduced as a generic approach to exchange data within a testbed and to send and receive control data to and from ns-3 in the testbed. However, the drawback of the implementation was, that only one instance of ns-3 could be served. Besides that, also limited functionality for controlling RAT interworking technologies were at hand. The evolved implementation of the whole Multi-RAT setup therefore leads to the implementation of a multi-instance capable TestMan implementation for the interface to ns-3. With this, all entities within the topology from Figure 19 can be reached. This is especially of interest if experimenters want to introduce Multi-RAT controller entities or other monitoring functionality to automatically control the Multi-RAT setup. This functionality is the building block of complete control over the different stacks. Complete control also involves configuration of the LTE-WiFi interworking technologies such as LWA and LWIP as well as the newly integrated LTE-LTE and LTE-5G interworking functionality DC. The LTE-WiFi interworking technology and control was successfully used within Open Call 2 by the multiRATsched experiment.

4.2.2 Implementation Results

The system diagram of the TestMan ns-3 interface is shown in Figure 23.

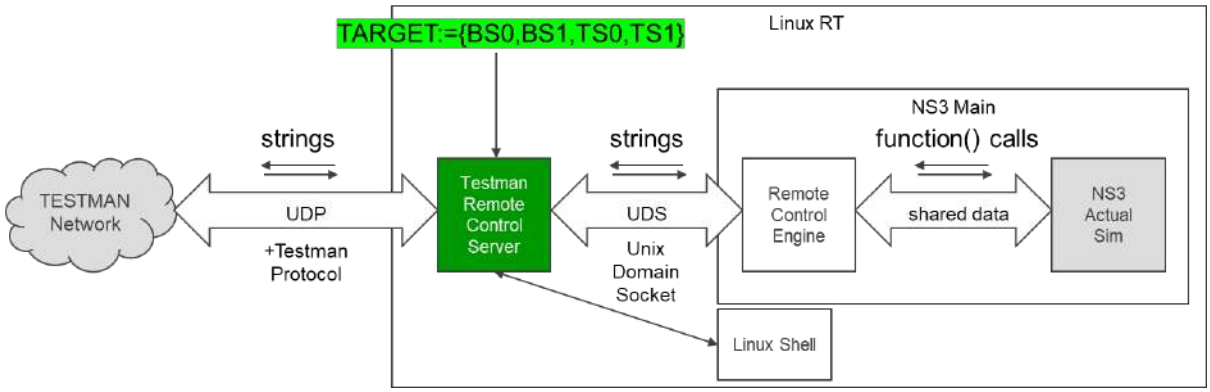

Figure 23: System Architecture of TestMan ns-3 interface with addressing options.

To achieve the aforementioned multi-instance capabilities, the TestMan architecture needed to be enhanced. Therefore, a target identifier was introduced that uniquely defines a certain machine instance. As the Multi-RAT setup consists of BSs and terminal stations (TS), exemplary identifiers are *BS0, BS1, TS0* and *TS1* as depicted also in Figure 23. An exemplary machine will run Linux (Linux RT) as operating system (OS). This is important because a similar Linux-based Operating System (OS) (NI Linux RT) is used for the intended NI USRP 2974 SDR for real time experiments. On this machine an ns-3 instance is running together with a TestMan Remote Control Server instance (green) that connects ns-3 to the TestMan network. While the TestMan server was started without any parameters beforehand, in the new version the unique target identifier is passed as an argument to the server application. This enforces this TestMan server entity to only react on messages that are dedicated to this target identifier.

Figure 24: TestMan Server start command structure with target identifier

Figure 24 shows an exemplary server application start for target identifier *"BS0"* with argument (a) and confirmation of the target identifier that was set (b).

In order to route commands to defined target identifiers within the TestMan network, the command structure needs to accommodate for the newly introduced addressing scheme. Therefore, the command structure was enhanced from the previous version as depicted in Figure 25.

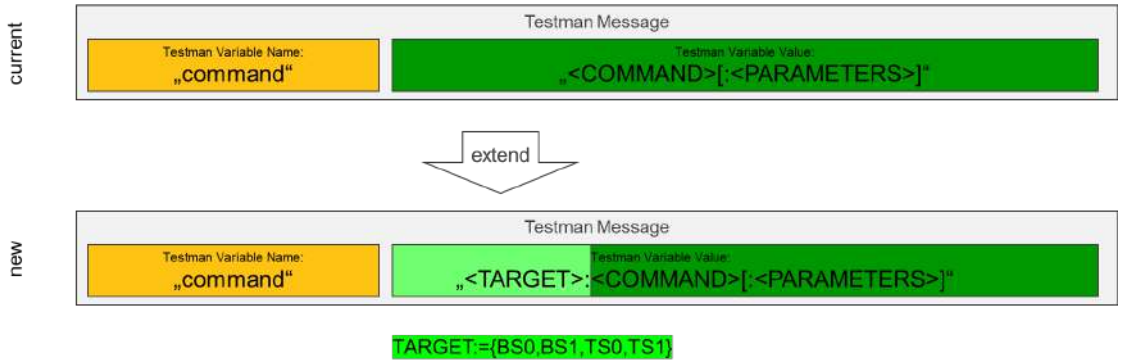

Figure 25: TestMan ns-3 interface command structure enhancements for addressing.

The command structure "current" shows the original structure that incorporates a TestMan-related "command" prefix, followed by control specific commands. These commands are described in [19]. To distinguish between different target identifiers, a new field named <TARGET> was introduced and prepended to the original control specific commands. The prepending operation was chosen as with this, the original functionality of not only controlling ns-3 but also starting shells on the remote machine will also leverage the multi-instance functionality.

Finally, Figure 26 shows an exemplary usage of the TestMan server on a linux machine that potentially needs to be remotely controlled.

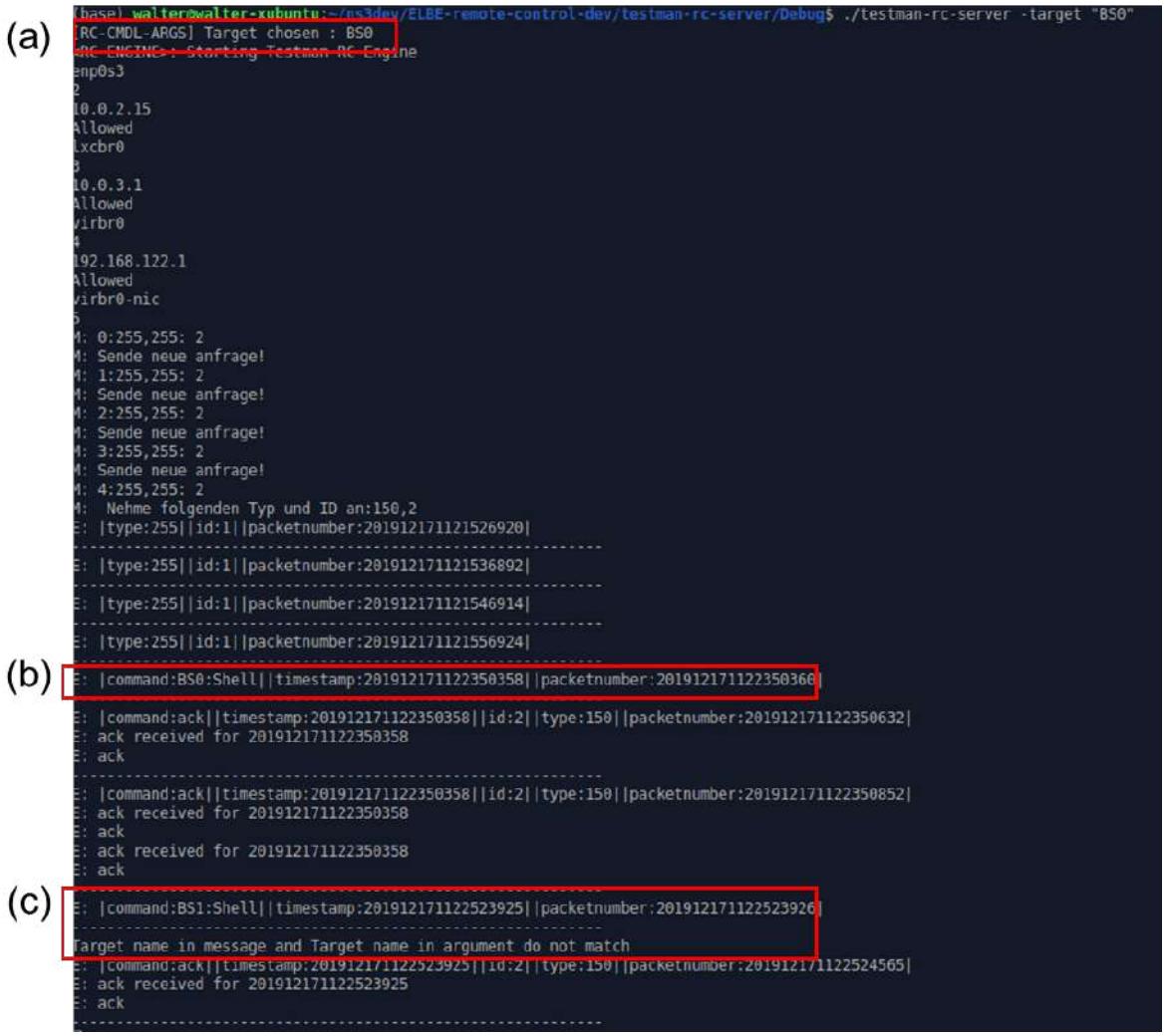

Figure 26: TestMan Server Terminal Operation with different target identifier commands.

The server is started with the command as mentioned in Figure 24. In (a) the specified target identifier is validated. In this example, “BS0” is chosen. Next, a command is sent from a remote host control program which wants to start a linux shell on the particular machine with target identifier “BS0”, see (b). The command with the structure “command:BS0:Shell” is shown to be received by the server and executed without an error. Finally, another remote host control program wants to send a similar control command to the machine with target identifier “BS1”. As the TestMan protocol itself is broadcast-based, such a command will also be received by the TestMan server instance with target identifier “BS0”, as shown in (c). The server of the machine with identifier “BS0” will recognize the command to be ignored as it doesn’t match its own identifier as shown in the terminal.

As part of the OC1 and OC2 for Extensions, the RAT interworking technologies LWA/LWIP for LTE-WiFi and DC for 5G-LTE and LTE-LTE were implemented. While the functionality of the implementations is mainly concerned with the data path, experimenters also need to control the different interworking technologies such that different traffic flows can be switched on and off while the whole Multi-RAT setup is running. Such control functionality can be envisaged as a simple method to incorporate network wide control entities like Multi-RAT and SDN controllers. Red marked interfaces in Figure 19 depict these interfaces to the different ns-3 instances. A simple implementation of a control functionality with the help of the ns-3 TestMan integration (Ref4.3) is shown in Figure 27.

```
if (pdcp_decisionlwa != g_RemoteControlEngine.GetPdb()->getParameterLwaDecVariable())
  {
        pdcp_decisionlwa = g_RemoteControlEngine.GetPdb()->getParameterLwaDecVariable();
        NI_LOG_CONSOLE_DEBUG("NI.RC:LWA value changed! LWA value is : " << pdcp_decisionlwa);
  }
if (pdcp_decisionlwip != g_RemoteControlEngine.GetPdb()->getParameterLwipDecVariable())
  {
        pdcp_decisionlwip = g_RemoteControlEngine.GetPdb()->getParameterLwipDecVariable();
        NI_LOG_CONSOLE_DEBUG("NI.RC:LWIP value changed! LWIP value is : " << pdcp_decisionlwip);
  }
```

Figure 27: Example implementation of LWA/LWIP control functionality within ns-3 lte-pdcp.cc source with usage of TestMan and ns-3 parameter database [19].

The decision variables *pdcp_decisionlwa* and *pdcp_decisionlwip* are internal variables of the current LWA/LWIP PDCP implementation of the ns-3 LTE module in the ORCA Multi-RAT platform. These variables set, e.g., which LWA mode is used. The variables have default values were LWA/LWIP is switched off. With each call of the respective PDCP layer functionality, the TestMan ns-3 remote control parameter database is queried with the call

- *g_RemoteControlEngine.GetPdb()->getParameterLwaDecVariable()*

that reads the LwaDecVariable from the parameter database (Pdb). As the parameter database can be updated on the fly with new values for variables from commands that were send via the TestMan protocol, an experimenter can easily adapt control functionality within ns-3 by adding respective parameters to the parameter database as described in [19]. The control of LWA/LWIP is already implemented in the ORCA Multi-RAT platform. DC can be controlled likewise.

This generic implementation gives an experimenter the freedom to adapt control mechanisms as needed to implement experiments with different control mechanisms and algorithms where the whole control communication is based upon the TestMan remote control functionality with the addition of ns-3 specific remote control and the newly added multi-instance capabilities.

4.2.3 Testbed Integration

The complete Multi-RAT control mechanisms for multiple ns-3 instances can be used as a module in the testbed. If needed, the module for TestMan remote control can be deployed to the respective images within the testbed. As all machines within the testbed are connected to the same network the communication between the different instances can be assured. The accompanying remote control functionality and RAT interworking functionality is part of the Multi-RAT experimentation platform that can be downloaded in its latest version [17].

4.3 Multi-RAT Monitoring Interface and Application

The Monitoring Interface visualizes the workings of the several devices during an experiment. Figure 28 outlines the scheme, which was demonstrated during the 5G Summit 2019 in Dresden.

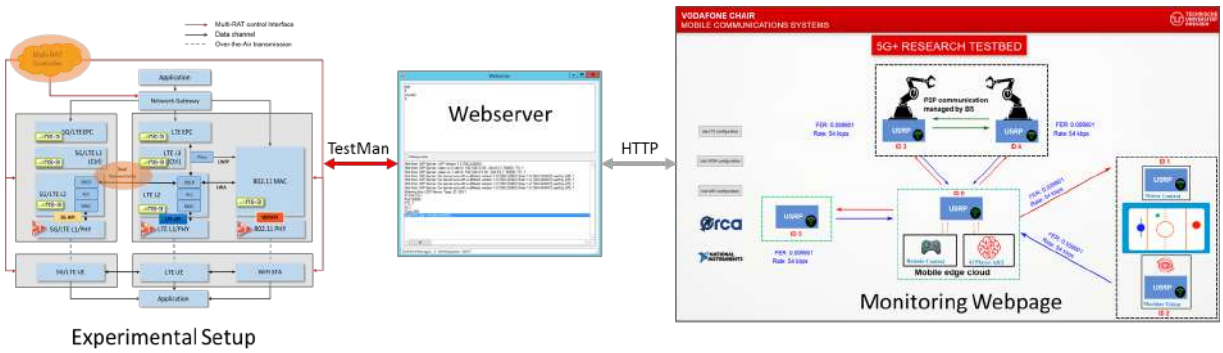

Figure 28: General structure of the Monitoring Interface.

4.3.1 Motivation

The overall Multi-RAT Demonstrator consists of multiple components. Therefore, it is difficult to follow each individual component during an experiment or in an exhibition. Since each component is connected to the TestMan Network, the idea is to use the same protocol to exchange information, such that they can be visualized on a monitoring webpage. Other than dedicated applications, a webpage can be shown on any modern TV screen, smartphone or tablet. Further, the web browser takes care of resizing the content fitting to the respective screen size.

4.3.2 Implementation Results

To realize this idea a modified webserver listens to the message exchanges in the TestMan network. Messages containing a special key/value pair will be understood as valuable for the monitoring webpage. Thus, the content of this message is stored in an internal dictionary acting as a database in the webserver. Now the webpage can request the values via the respective key from the webserver via a standard HTTP request. In addition, data can be sent back to the TestMan network via the reversed way. The webpage needs to send data to webserver via a standard HTTP POST request. Then the message will be split in key/value pairs and send as a message to the TestMan network.

4.3.3 Testbed Integration

A first version of the Monitoring Webpage has been shown during the exhibition at the 5G Summit 2019 in Dresden. However, since the webserver can store any arbitrary data, as long as it is kept in a key/value pair, the Monitoring webpage can be fit to any other demonstrator or to supervise the work during an experiment. This way the application can be reused for future uses.

4.4 Ns-3 Multi-RAT Networking Examples for 5G, LTE and WiFi

4.4.1 Motivation

The completion of the Multi-RAT platform for interworking experimentation was the focus of Year 3 in ORCA. Experimenters therefore need a baseline implementation of specific topologies in ns-3 to study these techniques. As part of Year 2, a generic Multi-RAT experimentation example (see D4.3 [19]) was implemented as a main file for ns-3. This example already incorporated various investigation options, e.g., routing over different paths within ns-3 as well as the LTE-WiFi interworking technology LWA and LWIP. In Year 3, the possibilities were enhanced to include also an exemplary topology for studying interworking technologies between LTE and LTE as well as 5G and LTE.

4.4.2 Implementation Results

The envisaged scenario for the Multi-RAT system is shown in Figure 29.

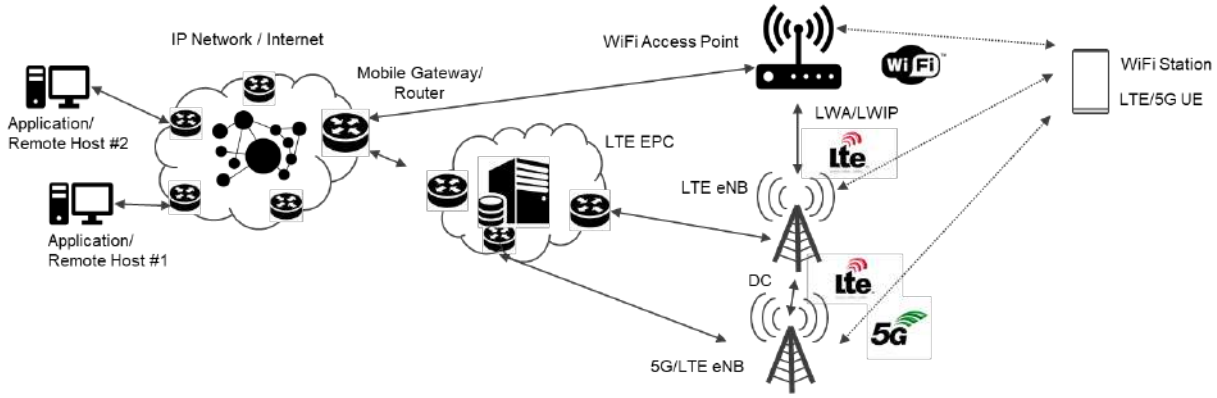

Figure 29: LTE/WiFi/5G Network Scenario.

The topology includes a configurable number of remote hosts that would run different applications. Ns-3 offers a variety of applications that mimic different types of traffic from streaming to bulk transmission also with different transport protocols such as TCP and UDP. The remote hosts are connected to an arbitrary topology for the internet or any IP network. The mobile gateway router is then used to interface the network to the different radio access technologies. In the ORCA Multi-RAT scenario, these radio access technologies are WiFi, 5G and LTE. While the WiFi AP would be directly connected to the internet/network through the router, cellular technologies usually need the separate core network (EPC) to be organized. This core network is connected to the IP-based network and can organize multiple associated BSs. The EPC implementation in the ns-3 Multi-RAT example is based on LTE. The topology would then include several BSs of either LTE or 5G as well as a different number of WiFi APs. Each BS or AP can have several attached WiFi stations or LTE/5G UEs. While originally, LTE/5G UEs and WiFi stations can be separate entities, the goal of a Multi-RAT platform for interworking experimentation is to have a combined device such as a modern mobile smartphone. These devices include several different radio access technologies and so applications can make use of interworking technologies between the different RATs. To accommodate for the above-mentioned interworking technologies, the interfaces between WiFi APs and LTE base stations as well as between LTE base stations and LTE/5G base stations are also part of the network scenario. All connections are IP based such that simple IP routing can be accomplished throughout the complete topology and traffic flows can be easily redirected to different end-points.

To fulfill the above outlined Multi-RAT networking topology, two ns-3 main files for investigations on the two interworking technologies were implemented. Figure 30 shows the topology for networking and interworking between LTE and WiFi. The main file is explained in detail in D3.3 [20].

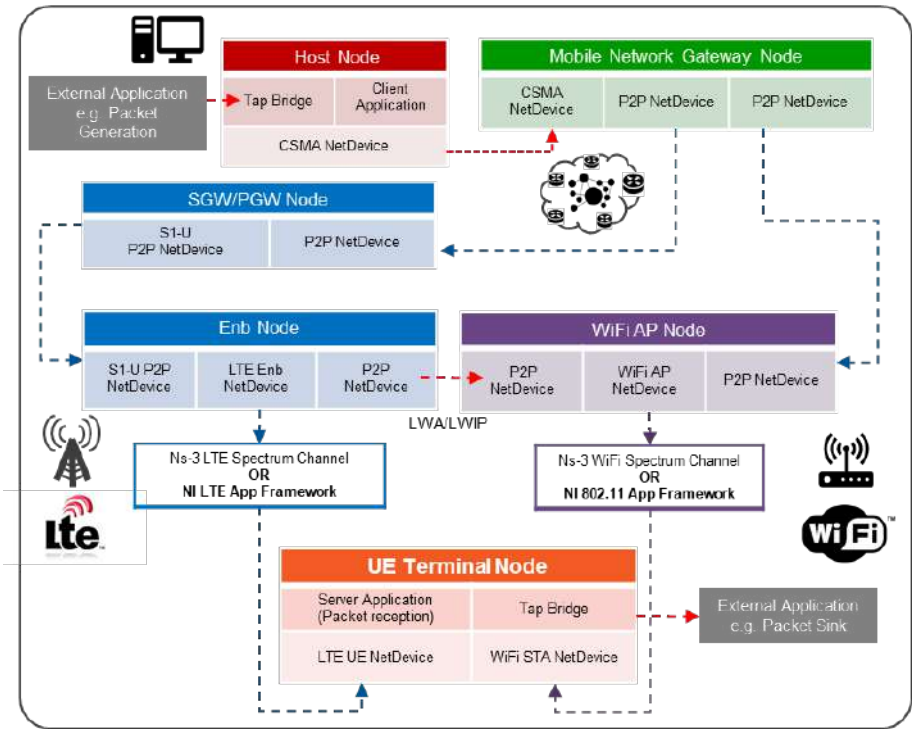

Figure 30: LTE-WiFi Interworking Example Topology in ns-3.

To have similar possibilities and comparability, the structure of the LTE-WiFi topology was adapted to be used for the LTE-LTE interworking example topology as well. The adapted topology can be seen in Figure 31.

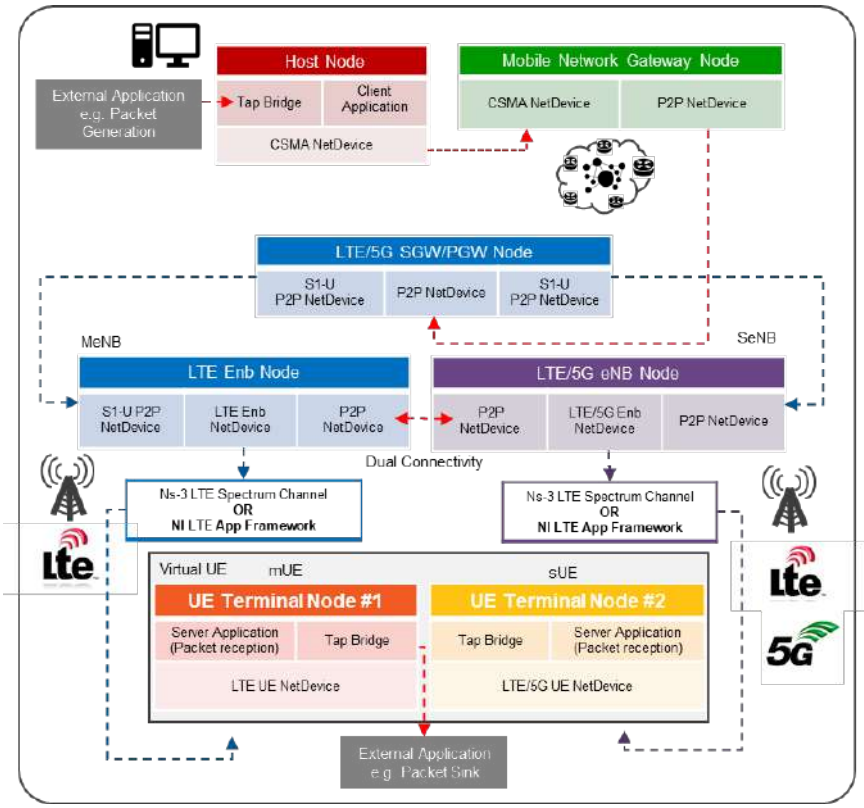

Figure 31: LTE-5G Interworking Example Topology in ns-3.

Key components of the topology remain the same. A remote host node serves as the application that can be used to produce data traffic of different types. Additionally, the option of injecting external traffic from outside of ns-3 is also given with the help of tap-bridges [16]. The host node is then connected to the Mobile Network Gateway Node that serves as an expandable model of the internet. Attached to the network gateway and to the interface of the radio access technologies is the LTE/5G core network, denoted as LTE/5G SGW/PGW (Serving Gateway, Packet Data Network Gateway). This core can serve several BSs via the S1 interface. The example topology implementation uses two base stations. One is used as LTE base station while the other can either be LTE or a 5G base station, depending on the PHY layer that is used when NI SDR hardware is attached to the ns-3 instances via the NI L1/L2 API [21]. Irrespective of the PHY layer, the upper layers of the stack remain LTE. The X2 interface that serves as a connection between base stations is implemented via point-to-point (P2P) net devices [22] and can be used to investigate DC behavior. Finally, two UE nodes that serve as nodes for LTE and LTE/5G are implemented as model for a Multi-RAT capable end device. The connection from the BSs to the UE can be either realized as ns-3 internal spectrum channel implementations or as interfaces to NI SDR hardware via the NI L1/L2 API as real-time over-the-air transmission.

An exemplary run of the LTE-LTE ns-3 main file is shown in the following printout:

```
(Py36) walter@walter-xubuntu:~/ns3dev/ELBE-ns3-dev/build/src/ni/examples$ ./ns3.26-ni-lte-simple-dc-optimized --daliDualConnectivityEnabled=1 --fdDeviceName=enp0s3:

-------- NS-3 Configuration ------------
Running LTE node as:    ENB and UE
LTE API:                disabled
5G API extensions:      disabled
LTE UDP Loopback:       disabled
TapBridge:              disabled
Logging:                enabled
(a) DALI Dual Connectivity: enabled
NI Module Version:      2.1.0
Required AFW Version:   2.2

Init NI modules ...
Start logging thread to file /tmp/Log_Lte_BSTS.txt

-------- NS-3 Topology Information ------
Number of ETH devices      = 3
(b) Number of LTE UE devices   = 2
Number of LTE eNB devices  = 2
Router GW IP Addr          = 10.1.1.1
LTE Net GW IP Addr         = 10.1.2.2
LTE EPC PGW IP Addr        = 7.0.0.1
(c) LTE UE#1 IP Addr           = 7.0.0.2
Client IP Addr             = 10.1.1.2
Server IP Addr             = 7.0.0.2
```

Figure 32: Initialization phase and IP/topology description of LTE-LTE/5G ns-3 simulation.

The example program shows a simulation that uses DC which can be enabled via a command line argument (a). For such a topology, two eNBs as well as two UEs are needed (master and secondary) which can be configured and is shown in (b). As the complete topology is configurable with IP addresses, the remote host node that runs the client application has a specific IP associated (10.1.1.2). The LTE node of UE#1 also gets an IP address associated (7.0.0.2) and acts as the server end point of this end-to-end transmission, see (c). This can be confirmed as the server IP address is the same as the UE IP address. When initial configuration is done, the simulation starts. The respective output is shown in Figure 33.

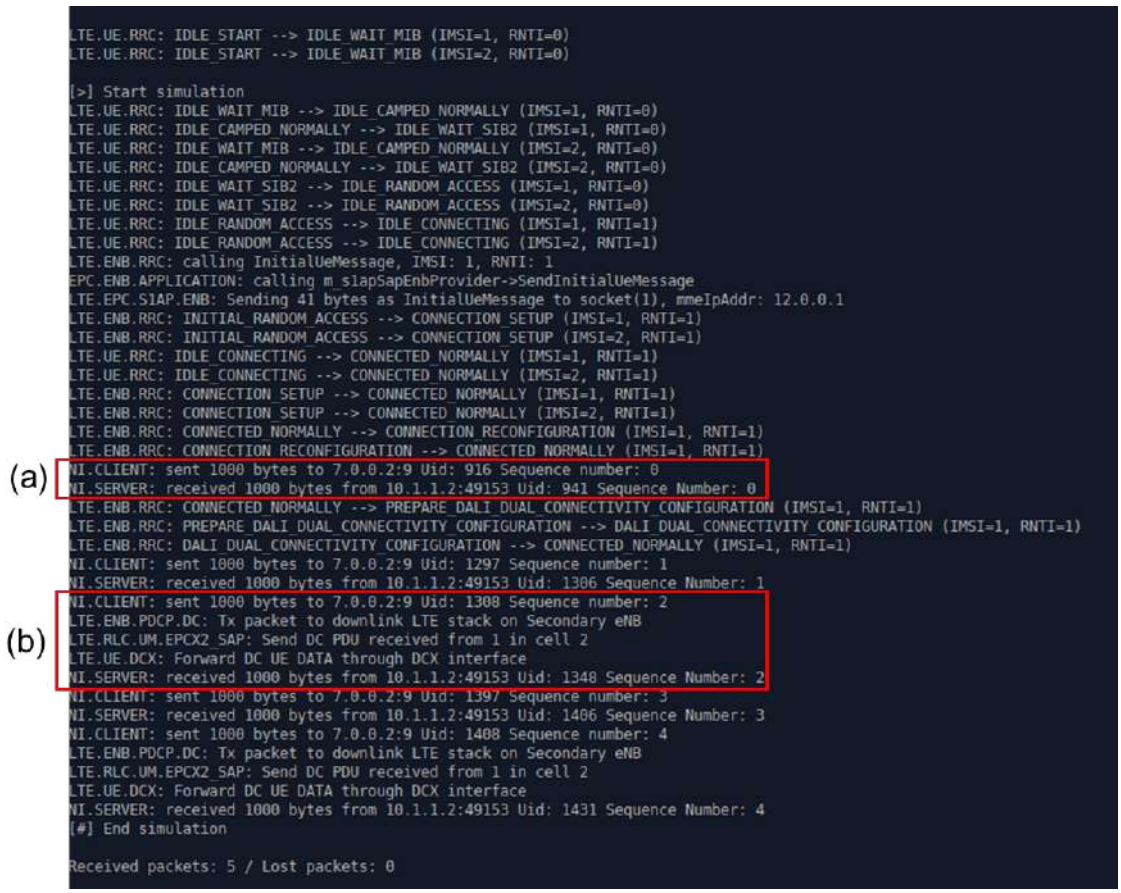

Figure 33: Simulation flow and packet transmission of LTE-LTE/5G ns-3 simulation.

After initial control plane traffic (Call Setup / Attach Procedure) to setup all the attachment of UEs to eNBs with Radio Link Control (RLC) signalling, the packet transmission starts at (a) with a packet that is send from the client (IP 10.1.1.2) to the server at IP 7.0.0.2 which is the master UE. As soon as the DC transmission is launched (see LTE.ENB.RLC: messages between (a) and (b)), packets that are send to the UE at 7.0.0.2 will partially be send over the link of the secondary eNB-UE pair. This is shown in (b) where the packet is forwarded from PDCP to the secondary eNB. Then it is transmitted over the secondary link and afterwards fed back from the secondary UE into the PDCP of the master UE where it will be received by the server application. In this scenario, both LTE links are used. Note that the 5G extension is only implemented on the SDR FPGA PHY layer with its L1-L2 API while the upper layers are used from the LTE stack (see D3.5 [15] for more detail). This means that in pure ns-3 simulation, up to now only LTE simulations can be accomplished as the 5G LENA module is not integrated (see D3.5 [15]). The experimentation setup can be arbitrarily configured to also setup traffic flows towards each UE independently.

With this setup, experimenters can develop different topologies and investigate Multi-RAT and SDN concepts. The gateway node, e.g., can be enhanced to become an OpenFlow switch which enables dynamic reconfiguration of routes.

4.4.3 Testbed Integration

The LTE-WiFi and LTE-LTE example topologies are part of the Multi-RAT platform as dedicated ns-3 main files. These example topologies are published on Github [17]. A researcher or potential experimenter can make use of the topologies in advance for designing the investigation/experiment with the topologies in simulation mode and when finalized, bring the ns-3 based Multi-RAT example

topology code to the testbed to be used with the NI SDR hardware for real-world experiments. The OWL testbed (http://owl.ifn.et.tu-dresden.de/orca/) from TUD is the preferred testbed platform [18].

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

- IMEC – A control plane upon full stack open source WiFi implementation is established, and API is extended for configuration of multi-channel virtual transceiver.
- TCD & IMEC – A hierarchical orchestration scheme for E2E networks, addressing the oversimplification over the allocation of resources, and the limitation on the support for network segments of existing E2E orchestration solutions, through the coordination between distributed specialised orchestrators.
- TUD – Real-time beam steering algorithm with sub 6 GHz control channel for dealing with mobility conditions. In addition, TUD also implemented a simple MAC mechanism that allows a TDD-based multi-user functionality for the GFDM PHY.
- KUL – A hybrid self-interference cancellation scheme which adopts the tuning algorithm based on the functional priority of the device. In addition, the mmWave extension has been incorporated into the MIMO testbed, allowing for the implementation of Multiple-User mmWave MIMO systems.
- NI – An enhancement of DC functionality towards interworking between LTE and 5G as well as control plane functionality enhancements of the TestMan interface to support RAT and RAT interworking control with monitoring capabilities were achieved. Two ns-3 examples for Multi-RAT simulation and experiment development are additionally introduced.

Additionally, a brief description was given on how the aforementioned implementations were integrated in the third year ORCA showcases and in the partners' testbeds.

REFERENCES

[1] mac80211 subsystem basics, https://www.kernel.org/doc/html/v4.16/driver-api/80211/mac80211.html, last accessed on 20th December 2019.

[2] M. Kist, J. Rochol, L. A. DaSilva, and C. B. Both, “SDR Virtualization in Future Mobile Networks: Enabling Multi-Programmable Air Interfaces,” IEEE ICC, Kansas City, MO, 20-24 May 2018.

[3] ORCA Deliverable D3.5, Final Operational SDR Platforms with Advanced Reconfigurability/Reprogrammability Capabilities, January 2019.

[4] LabVIEW Communications LTE Application Framework 2.5, Manual, Oct 2018

[5] Hassani, Seyed Ali, Karthick Parashar, André Bourdoux, Barend van Liempd, and Sofie Pollin. "Doppler Radar with In-Band Full Duplex Radios." In IEEE INFOCOM 2019-IEEE Conference on Computer Communications, pp. 1945-1953. IEEE, 2019.

[6] Hassani, Seyed Ali, and Sofie Pollin. "A Low-Latency Wireless Network for Cloud-Based Robot Control." In International Conference on Cognitive Radio Oriented Wireless Networks, pp. 46-51. Springer, Cham, 2018.

[7] Hassani, Seyed Ali, Andrea Guevara, Karthick Parashar, Andre Bourdoux, Barend van Liempd, and Sofie Pollin. "An In-Band Full-Duplex Transceiver for Simultaneous Communication and Environmental Sensing." In 2018 52nd Asilomar Conference on Signals, Systems, and Computers, pp. 1389-1394. IEEE, 2018.

[8] X. Wang, M. Laabs, D. Plettemeier, K. Kosaka, and Y. Matsunaga, 28 GHz Multi-Beam Antenna Array based on Wideband High-dimension 16x16 Butler Matrix,”, in 2019 13th European Conference on Antennas and Propagation (EuCAP), Mar 2019, pp. 1–4.

[9] A. Colpaert, E. Vinogradov, S. Pollin, “Fixed mmWave Multi-User MIMO: performance analysis and proof-of-concept architecture.”, Manuscript Submitted for publication.

[10] ORCA Deliverable D7.4, “Summary of results of second Open Call for extensions”, December 2019.

[11] TestMan github repository, https://github.com/vodafone-chair/TestMan.git, December 2019.

[12] ORCA Deliverable D7.2, “Summary of results of first Open Call for extensions”, December 2018.

[13] 3GPP Release 13, https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId, December 2019.

[14] ns-3 discrete-event network simulator for Internet systems, https://www.nsnam.org, December 2019.

[15] ORCA Deliverable D3.5, Final operational real-time SDR platforms, January 2020.

[16] ns-3 Emulation Overview, https://www.nsnam.org/docs/release/3.30/models/html/emulation-overview.html, December 2019.

[17] NI ns-3 Application Example, https://github.com/ni/NI-ns3-ApplicationExample, December 2019.

[18] OWL Testbed, http://owl.ifn.et.tu-dresden.de/, December 2019.

[19] ORCA Deliverable D4.3, “Enhanced operational SDR platforms with end-to-end capabilities”, December 2018.

[20] ORCA Deliverable D3.3, “Enhanced operational real-time SDR platforms”, December 2018.

[21] ORCA Functionality: Generalized API, https://orca-project.eu/wp-content/uploads/sites/4/2017/09/ORCA_2.2-08-v1.1.pdf, Dec 2018.

[22] ns-3 PointToPoint NetDevice, https://www.nsnam.org/docs/release/3.30/models/html/point-to-point.html, December 2019.