ICT-777137

5G-RANGE

5G-RANGE: Remote Area Access Network for the 5th Generation

Research and Innovation Action
H2020-EUB-2017 – EU-BRAZIL Joint Call

D6.2: Prototype Integration

Due date of deliverable: 30th April 2020
Actual submission date: 12th November 2020

Start date of project: 1 November 2017 Duration: 30 months
Lead contractor for this deliverable: Inatel and TUD
Version 1 date 12th November 2020
Confidentiality status: Public

Abstract

The 5G-RANGE project aims for conceiving, develop and implement a mobile network that can support different use cases in remote and rural areas. The evaluation of the conceived network is essention to validate the proposed protocol stack for the physical, link and network layers. In order to achive this goal, a integrated proof-of-concept is necessary. This deliverable will describe the integration of the real-time blocks that will culminate in the 5G-PoC BS and 5G-PoC UE. The use cases that will be exploited by the proof-of-concept are voice and data connectivity, bachhauling and smart farm.

Target audience

The primary target audience for this document is the radio access network research and development community, particularly those with an interest in mobile communication physical and MAC layers. This material can be fully understood by readers with a background in mobile wireless cellular systems, especially those familiar with 3GPP standards for 4G and 5G.

Impressum

Full project title:	5G-RANGE: Remote Area Access Network for the 5th Generation
Document title:	D6.2 – Prototype Integration
Editor:	Moacyr Martucci Junior, USP (BR)
Project Co-ordinator:	Marcelo Bagnulo, UC3M (EU), Priscila Solis, UnB (BR)
Technical Manager:	Luciano Mendes, Inatel (BR), Yaning Zou, TUD (EU)
WP leaders:	Ivo Bizon, TUD (EU), Luciano Mendes, Inatel (BR)

This project is co-funded by the European Union through the ICT programme under H2020.

Executive Summary

This deliverable provides information about the 5G-RANGE proof-of-concept (PoC) prototype integration. The real-time software blocks and components that composes the 5G-RANGE-PoC Base Station and 5G-RANGE-PoC User Equipment (UE) are described. The PoC prototype was designed following the requirements presented in deliverables D2.1 [1] and D2.2 [2], employing the techniques defined in Work Packages (WPs) 3 and 4. The main aim of the PoC is to address the Voice and Data Connectivity, Wireless Backhauling and Smart Farm use cases.

Three main subsystems that compose the PoC prototype are: Physical layer (PHY), Medium Access Control (MAC), and 5G Core Network layers. An overview of the implementation aspects for each of these subsystems and their integration is provided. As an implementation strategy, the prototype is based on a Software Defined Radio (SDR) platform with MAC and PHY functions implemented in a General-Purpose Processor (GPP). This software approach accelerates the prototype development and allowed the researchers to reuse functions developed for the simulation in the real-time PoC. The SDR approach also easies the technological transfer for the market, since the portability of the protocol stack to others hardware solutions is seeamsless.

List of Authors

Ivo Bizon Franco de Almeida (TUD)
Alexandre Carvalho Ferreira (Inatel)
Luciano Leonel Mendes (Inatel)
Wheberth Damascena Dias (Inatel)
Francisco Valera (UC3M)
Iván Vidal (UC3M)
Eduardo Melão (CPqD)
Jorge Seki (CPqD)

Table of Contents

List of Figures

Definitions and Abbreviations

5G-ACRA	5G- Advanced Coding for Remote Areas
5G-I2RA	5G- Inner Receiver for Remote Areas applications
5G-MIMORA	5G- Multiple-Input Multiple-Output techniques for Remote Areas Applications
A + P	Address plus Port
Adj-Rib-In	Adjacent Routing Information Base, Incoming
AJAX	Asynchronous JavaScript and XML
AMC	Adaptive Modulation and Coding
AMF	Access and Mobility Management function
API	Application Programming Interface
ARP	Address Resolution Protocol
AS	Autonomous System
ASIC	Application Specific Integrated Circuit
AUSF	Authentication Server Function
AWGN	Additive White Gaussian Noise
BER	Bit Error Rate
BGP	Border Gateway Protocol
BLER	Block Error Rate
BRAS	Broadband Remote Access Server
BS	Base Station
CB	Code Block
CDF	Cumulative Distribution Function
CDL	Cluster-Delay Line
CGN	Carrier Grade NAT
CFO	Carrier Frequency Offset
CLI	Command Line Interface
CMAC	Cognitive Medium Access Control
COTS	Common of-the-shelf
CP	Cyclic Prefix
CQI	Channel Quality Indicator
CRC	Cyclic Redundancy Check
C-RAN	Cloud Radio Access Network
CS	Cyclic Suffix
CSI	Channel State Information
D2D	Device-to-Device Communication
DCI	Download Control Information
DDoS	Distributed Denial of Service
DL	Downlink
DNS	Domain Name System
DoS	Denial of Service
DPD	Digital Pre-distortion
DSL	Digital Subscriber Line
DSLAM	DSL Access Multiplexer
DSA	Dynamic Spectrum Allocation
eBGP	External BGP
ECR	Effective Code Rate
EESM	Exponential Effective SINR Mapping
ESM	Effective SINR Mapping
FCFS	First-Come First-Serve
FEC	Forward Error Control
FFT	Fast Fourier Transform

FLUTDL	Fusion Lookup Table Downlink
FPGA	Field Programmable Gate Array
GbE	Gigabit Ethernet
GFDM	Generalized Frequency Division Multiplexing
gNB	Next generation Node B
GPP	General-Purpose Processors
HARQ	Hybrid Automatic Request
HDL	Hardware Description Language
HGW	Home Gateway
IANA	Internet Assigned Numbers Authority
iBGP	Internal BGP
IETF	Internet Engineering Task Force
IFPI	Interference Free Pilot Insertion
IMD	Intermodulation Distortion
IPC	Inter-Process Communication
ISP	Internet Service Provider
L2S	Link-to-System
L3	Layer 3
LISP	Locator/ID Separation Protocol
LLC	Logical Link Control
LoS	Line of Sight
LSN	Large Scale NAT
LSM	Link to System Mapping
MAC	Media Access Control
MACC	MAC Control
MCS	Modulation and Coding Scheme
MED	Multi-Exit Discriminator
MIMO	Multiple-Input Multiple-Output
MIESM	Mutual Information Effective SINR Mapping
MQ	Message Queues
MSE	Mean Square Error
MSL	Maximum Segment Lifetime
MTC	Machine Type Communications
MTU	Maximum Transmission Unit
NAT	Network Address Translation
NIC	Network Interface Card
NFV	Network Function Virtualization
NFVI	Network Function Virtualized Infrastructure
NFVO	NFV Orchestrator
NLoS	Non-Line of Sight
NRF	Network Repository Function
NSSF	Network Slice Selection Function
OFDM	Orthogonal Frequency Division Multiplexing
OOBE	Out-of-Band Emissions
OS	Operational System
P2P	Point-to-point
PC	Programmable Computer
PDU	Protocol Data Units
PHY	Physical Layer
PI	Provider Independent
PoC	Proof-of-Concept
PU	Primary User

PSS	Pilot Synchronization Signal
QAM	Quadrature Amplitude Modulation
QPSK	Quadrature Phase-Shift Keying
RA	Resource Allocation
RB	Resource Block
RBG	Resource Block Group
RIR	Regional Internet Registry
r.m.s	Root Mean Square
RFPA	Radiofrequency Power Amplifier
RX	Reception
SAVI	Source Address Validation Improvements
SDR	Software-Defined Radio
SDU	Service Data Unit
SE	Spectrum Efficiency
SINR	Signal-to-Interference-plus-Noise Ratio
SLS	System-Level Simulator
SNR	Signal-to-Noise Ratio
SMF	Session Management function
SRS	Sound Reference Signal
SoC	System-on-Chip
SS	Spectrum Sensing
SUAV	Small Unmanned Aerial Vehicle
TB	Transport Block
TCB	Transmission Control Block
TRL	Technology Readiness Level
TVWS	TV White Space
TX	Transmission
UCI	Unique Client Identifier
UE	User Equipment
UL	Uplink
USRP	Universal Software Radio Peripheral
VIM	Virtual Infrastructure Management
VNF	Virtual Network Functions
VNFM	VNF Manager

1 Introduction

The 5G-RANGE application scenarios envisage economically effective solutions for remote, rural and under-served areas. The features for these solutions, for example, long-range coverage and operation in TV Whitespaces (TVWS), need testing and evaluation to validate their effectiveness and efficiency. The Deliverable 6.1 accomplished the first step to validate the conceived solution for the rural area network, where a system simulator was developed to analyze the physical (PHY) and network layers' performance. The strategy for a second validation step is the construction of a functional prototype as a proof-of-concept (PoC) of the entire system.

The majority of 5G-RANGE features were implemented in the physical (PHY) and medium access control (MAC) layers. The Network Layer supports the integration of the 5G architecture access network protocol structure into the 5G-RANGE PoC. This document describes the implementation details for each of these layers and the integration between them to enable a functional prototype capable of implementing all the features necessary for the 5G-RANGE PoC realization.

Once the 5G-RANGE system prototype is integrated and ready to operate, its performance evaluation will be compared to the system requirements presented in deliverables D2.1 [1] and D2.2 [2]. The laboratory tests, field tests, and the PoC for the smart farm use case will be presented in the next deliverables (D6.3 and D6.4).

1.1 Deliverable Structure

The remaining of this document is structured as follows: Section 2 describes the implementation of the cognitive MAC layer. Section 3 presents the PHY layer with the implementation details involving the GNU Radio platform and functions, written in C/C++ and Python languages. Section 4 details the network layer, responsible for integrating the 5G core with the 5G-RANGE PoC. Section 5 presents the integration of all layers (MAC, PHY, and network). Section 6 concludes this document.

2 MAC Implementation Details

The software stack for the 5G-RANGE MAC layer was written using C++ programming language with object-oriented approach to representing its main entities. This section will bring details about these entities and MAC functionalities for 5G-RANGE application, using class diagram contents presented in section 2.1.

2.1 Overview

The key elements of MAC implementation are shown in Figure 1, with classes that represent entities related to parameters, interfaces, and control of MAC functionalities, such as control messages construction and scheduling.

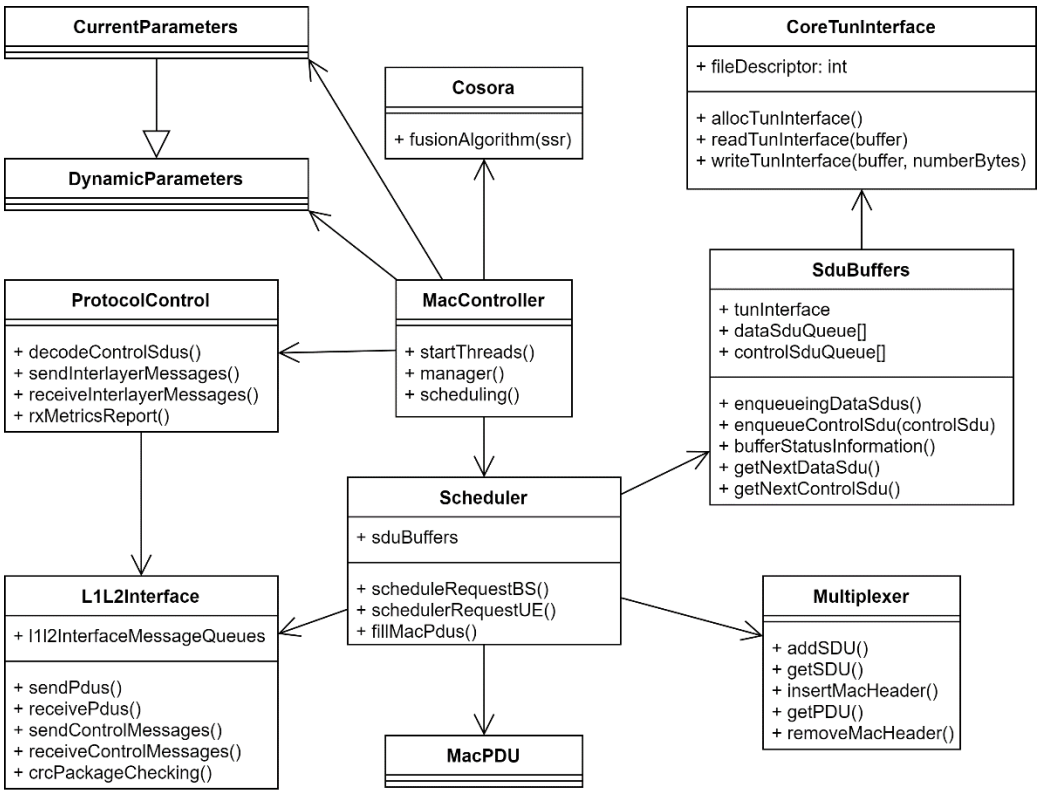

Figure 1. Simplified class diagram of MAC entities implemented for the 5G-RANGE PoC.

It is important to highlight that the same architecture is used on Next generation Node B (gNB) or Base Station (BS) and User Equipment (UEs), i.e., the same entities explained in this section will be used on both sides of 5G-RANGE communication link. Therefore, some entities will accumulate gNB and UE functions, but most of the functionalities are similar on both sides.

2.2 *MacController* class

The main entity of MAC implementation is the *MacController* class. It is responsible for initialising the system, setting its parameters, initialising all threads, and controlling system states based on the three main MAC commands:

- *MACStart.Request*: this command triggers MAC start-up. MAC will perform initial configuration and start system threads with *MacController::startThreads()* procedure.

- *MACConfig.Request*: this command is only available on BS and is responsible for informing the MAC to reconfigure its dynamic parameters with a set of new values. To do this, MAC will pause all scheduler and multiplexer operations and, after setting the new parameters, it will bring back to normal operation.
- *MACStop.Request*: this command will trigger MAC threads termination, releasing all resources and setting MAC process to wait for *MACStart.Request* command again or wait for the process to be terminated.

State machine in Figure 2 brings more detail about *MacController* entity operation and the use of the messages described above. After initialising the process, the *MacController::manager()* function is called and MAC enters *STANDBY_MODE* waiting for *MACStart.Request* command. Then, default system parameters are read and configured setting MAC into *START_MODE* when an interlayer message *PHYConfig.Request* is received from PHY layer. With the response to PHY for this request, system finally enters *IDLE_MODE*, when it can perform transmission and reception of Protocol Data Units (PDUs). If dynamic parameters are changed and *MACConfig.Request* command is place (only on BS), system stops its operation partially to apply the new parameters in *RECONFIG_MODE*. Then it returns to *IDLE_MODE*. Finally, upon the reception of MACStop.Request, almost all resources are released, and threads are terminated leaving MAC into *STANDBY_MODE* again.

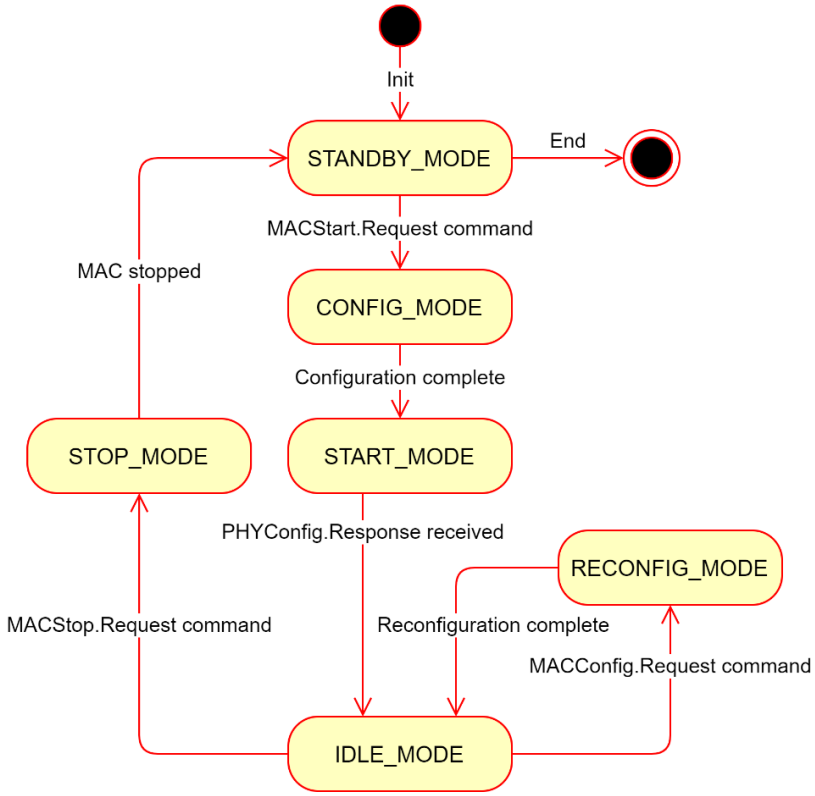

Figure 2. State machine diagram related to *MacController* operation.

2.3 Parameters entities

Two entities are used to store MAC layer parameters: *DynamicParameters* and *CurrentParameters*. The first entity contains parameters that can be changed during the process execution, such as Fusion Lookup Table Downlink (*FLUTDL*) or Modulation and Coding Scheme (MCS) for downlink and uplink. These parameters can be changed in MAC subtasks Cosora and Link Adaptation, respectively, and require system reconfiguration. Furthermore, other parameters are considered “dynamic” if they can have their

values changed by MAC Command Line Interface (CLI) (not finished), for example, uplink reservation, Multiple-Input Multiple-Output (MIMO) configuration, transmission power control, etc.
On the other hand, *CurrentParameters* is an entity extended for *DynamicParameters* to store current execution parameters. It is composed by *DynamicParameters* attributes (because the class was extended) plus static system parameters such as number of UEs configured, numerology, multiplexing scheme (Orthogonal Frequency Division Multiplexing - OFDM or Generalized Frequency Division Multiplexing - GFDM), and others.

The operation with these entities occurs as follows: on *start-up*, the system reads a text archive with all *CurrentParameters* attributes and initiates the system that will consult only *CurrentParameters* object attributes. Then, the *DynamicParameters* entity is started as a copy of *CurrentParameters*, initially, and all necessary parameter alterations are made into *DynamicParameters* object. *CurrentParameters* is only updated when the system enters *RECONFIG_MODE*.

2.4 *Cosora* class

This class is responsible for performing 5G-RANGE's Collaborative Spectrum Sensing Optimized for Rural Areas (COSORA) operations. Its main task is to perform Spectrum Analysis based on Spectrum Sensing (SS) report received from the PHY layer.

2.4.1 Spectrum Sensing (SS) Procedure

Spectrum Sensing procedure occurs at the PHY layer during a silent period, where all UEs in a synchronized way do not perform transmission, and it provides to the MAC layer the Spectrum Sensing measurements, with 1 or 0 per channel, where 1 informs channel in idle and 0 informs channel occupied by Primary User (PU) signal.

UE sends the SS Report to the BS via MAC Control (MACC) Service Data Unit (SDU) (this SDU is explained in section 2.5), and BS receives this report and waits for SS report from the other UEs (if there are any) to perform the Fusion Algorithm. This procedure is shown in Figure 3.

2.4.2 Fusion Algorithm

The BS, after receiving the SS Report from the UE(s), performs the Fusion Algorithm (AND function), using the method *Cosora::fusionAlgorithm()*, and the result is stored in the Fusion Look Up Table Downlink (FLUTDL), as shown in Figure 3. The FLUTDL contains information about the Resource Blocks (RBs) that can be allocated for the UEs for the next Downlink (DL) transmission.

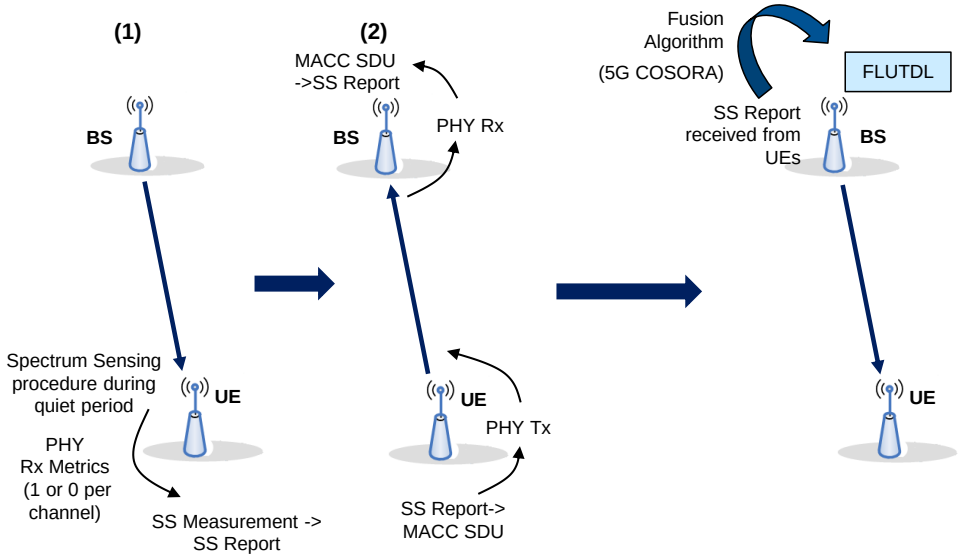

Figure 3. Spectrum Sensing (SS) Report and Fusion Algorithm.

A sample of *FLUTDL* is shown in Figure 4. The bit value 1 means the channel is idle (RBs available for MAC Scheduler Resource Allocation for DL transmission) and the bit 0 means the channel is occupied.

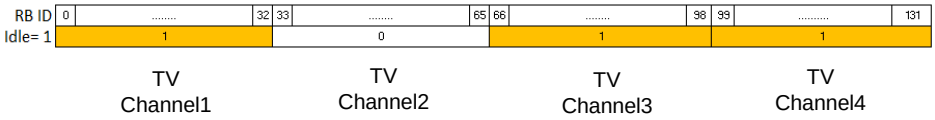

Figure 4. Example of Fusion Algorithm LUT where TV Channel 2 is occupied by a TV signal.

2.5 *ProtocolControl* class and Link Adaptation procedure

This class manages the control messages of MAC system. Control messages can be of two types:

- MACC SDUs: a special type of SDU used by MAC layer on both sides (UE and gNB) to exchange control information between MAC layers. MACC SDUs can carry these types of information:
  - BS to UE: Uplink (UL) MCS; ULReservation that is the RBs allocated for UE UL transmission; Rx Metrics report aperiodic or periodic configuration with its periodicity configuration;
  - UE to BS: Rx Metrics including SS Report, Average MCS of the RBs used in the previous reception, and MCS of each Resource Block (RB) used in the previous reception.
- Interlayer messages: MAC messages to PHY layer, and vice-versa, detailed in Section 5.1.

As mentioned above, *ProtocolControl* deals with user data exchange, and with key control information that is the Rx metrics. These metrics can be periodic or aperiodic, where UE's PHY send to UE's MAC the Rx Metrics. On both cases, these metrics received by MAC layer on the UE side triggers the elaboration of a MACC SDU to report these metrics to the BS MAC layer in the next UL transmission opportunity, using *ProtocolControl::rxMetricsReport()* function.

After receiving a MACC SDU, MAC layer on BS side triggers the procedure *ProtocolControl::decodeControlSdus()* to decode Rx metrics received, and two main procedures are made from here: Link Adaptation and/or Fusion Algorithm.

Link Adaptation occurs on BS side by performing DL MCS calculation based on DL SNR received by MACC SDU from the UE. For that, a table is used for query. If DL MCS must be changed, then *DynamicParameters* object is changed and system schedules a reconfiguration procedure. For the case of UL MCS calculation at BS side, Link Adaptation is triggered aperiodically when a PDU is received from UE (with SNR measured at the BS reception).

ProtocolControl also deals with Interlayer Messages. It defines a thread for permanently receiving these messages from PHY, *ProtocolControl::receiveInterlayerMessages()*. The function *ProtocolControl::snedInterlayerMessages()* is used to send Interlayer Messages to PHY. As mentioned earlier in this section, these messages are defined in Section 5.1.

2.6 *CoreTunInterface* class

The *CoreTunInterface* class was created to manage the interface with Linux Network (L3) layer. This interface is required so that MAC layer can operate with user IP packets as user data, performing sending and receiving operations with these IP packets in 5G-RANGE network.

For that purpose, a kernel virtual device, named TUN, was used together with Universal TUN/TAP Driver, available in Linux OS implementation. More details about the implementation to use this device and the L3 Interface are available in Section 5.2.

2.7 *SduBuffers* class and SDU treatment

5G-RANGE system can operate with multiple UEs connected. This means that gNB must operate with multiple packet destinations and multiple contexts, one for each UE. Therefore, Tun interface (described in section 5.2.2) at the gNB side can receive user data packets (IP packets) addressed do different UEs and this means multiple buffers with user data SDUs (MAC Data SDUs or MACD SDUs). Also, there can be MACC SDUs scheduled for different UEs at the same time. Thereby, a data structure containing SDU buffers is needed to help implementation.

SduBuffers class was created to manage the enqueueing process for MACC and MACD SDUs into queues. It contains as an attribute a *CoreTunInterface* object that will be used by the function *SduBuffers::queueingDataSdus()*, which defines a thread for constantly gathering IP packets present in Tun interface and putting them on *SduBuffers* queues. *SduBuffers* also has a method to enqueue MACC SDUs. To support the scheduling process, *SduBuffers* offers the Scheduler *SduBuffers::bufferStatusInformation()* method to deliver all buffer number of packets and number of Bytes enqueued. To help MAC PDU construction, the primitives *SduBuffers::getNextDataSdu()* and *SduBuffers::getNextControlSdu()* are offered.

An additional feature implemented to support the operation with IP packets was a timeout feature. It is known that late IP packets, after a certain time, lose their validity. This can happen when the system is overloaded, for example, requiring discarding these packets to release space in the queues and the system resources such as the processing time. Therefore, it is convenient to implement a clock resource in the code to timestamp packets and decide if they are waiting too long in the queue.

For this purpose, MAC implements a subframe time as the basic time unit: 4.6 ms. *MacController* launches the thread *TimerSubframe::countingThread()* to count subframes times until the end of system operation. When an IP Packet is inserted into *SduBuffers* queue, it is marked with a timestamp related to subframe number counted by counting thread. A system parameter, Ip Packet Timeout is defined so *SduBuffers* periodically verifies if the first MACD SDU in the queue is later than IP Packet Timeout time. If so, the packet is discarded from the queue. If not, the system verifies again later.

To remove packets from the queue, semaphores were also implemented to support the queue resource sharing between the *Scheduler* (which removes the packet from the queue for transmission) and the MACD SDU discarding thread.

2.8 MAC Scheduler and *MacPDU* class

The activities performed by MAC Scheduler are the final part of MAC operation before transmission. The spectrum allocation and PHY layer parameters must be considered when calculating the spectrum allocation necessary by BS for each UE (downlink) or by the UE (uplink). *MacPDU* class works as a common data structure shared between MAC and PHY and is used to store all information before transmission.

2.8.1 Downlink

The MAC layer scheduler is used for downlink communication and performs dynamic UE selection and spectrum allocation (RBs allocation) for each subframe started upon the reception of interlayer message *PHYTx.Indication*. Figure 5 shows the MAC scheduler procedure for downlink.

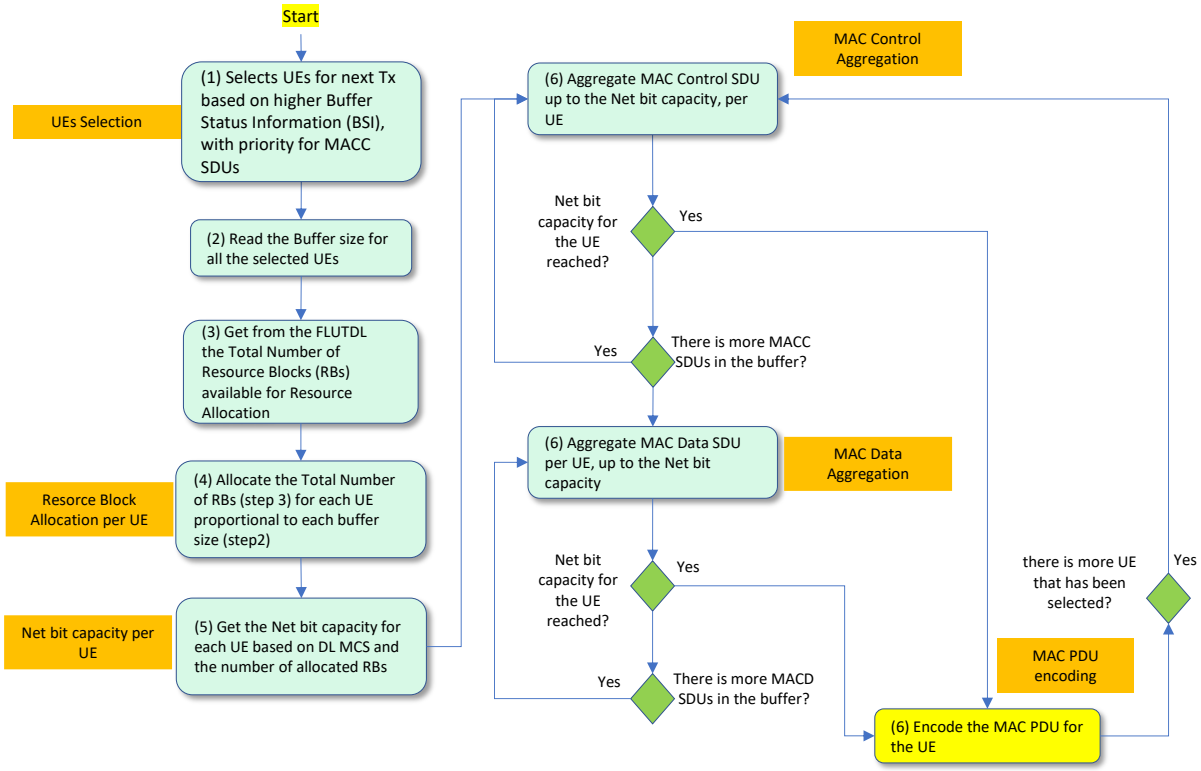

Figure 5. MAC Scheduler procedure (Basic View): UE selection and Multiplexing (aggregation) for Downlink Tx.

When a scheduled process is requested by the BS, the function *Scheduler::scheduleRequestBS()* is called and it first verifies the available RBs considering FLUTDL table. Then, for each UE, based on buffer status information from *SduBuffers*, the number of bits to transmit is calculated and procedures provided by INATEL are used to calculate the amount of RBs necessary for each UE taking into account numerology, MIMO scheme and MCS Downlink. Then, the available spectrum is shared for each UE based on the number of RBs required by each UE with data to transmit.

When the process above finishes, it is necessary to organize each MAC PDU for subframe transmission. For this, *MacPDU* class, shown in Figure 7, is used to store both the multiplexed SDUs and control information for PHY. The method *Scheduler::fillMacPdus()* is called and, for each UE, based on RB allocation decided by the Scheduler, a Multiplexer object is created to perform SDU aggregation. SDUs previously enqueued are aggregated by *Multiplexer::addSDU()* function and *Multiplexer::insertMacHeader()* assemble on PDU buffer, the MAC header that will help for decoding on reception. MAC PDU representation is shown in Figure 6, including MAC Header fields and MAC Payload.

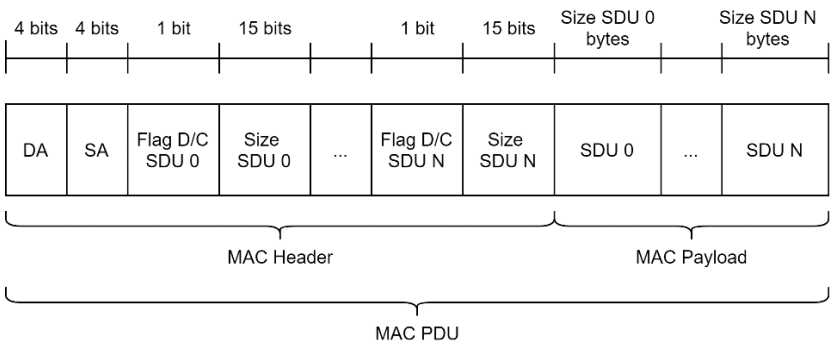

Figure 6. The MAC PDU, with Mac payload and MAC Header fields, with bit extension of each field.

The fields of MAC Header indicated by Figure 6 are described below:

- DA: Destination Address, composed of 4 bits. This address is the same used by the PHY layer to identify a unique UE in the System. BS Address is always zero.
- SA: Source Address, composed of 4 bits.
- Flag D/C SDU i: This field is composed of 1 bit and identifies if the i-th SDU is a MACC SDU or MACD SDU.
- Size SDU i: This field is composed of 15 bits and contains the i-th SDU size in bytes. This information, together with Flag D/C SDU i, is repeated N SDU times.
- SDU payloads are assembled in sequence after MAC Header.

The steps to complete the fields to generate the encoded MAC PDU showed in Figure 6, are made for all UEs selected for transmission, with at least one single PDU for each UE. Here, "at least" means that a UE can be scheduled with two PDUs if, in the middle of this UE spectrum allocation, a "hole" is present into the spectrum because of TV channels signal detected by Spectrum Sensing.

After the MAC PDU fields definition by MAC the *MacPDU* object must be prepared to be sent to PHY. The *MacPDU* class was first elaborated by INATEL and is an agreement between MAC and PHY to use the same data structure. Besides the MAC PDU, *MacPDU* class contains other attributes related to control information that must be exchanged between MAC and PHY, on both transmission and reception procedures. Figure 7 shows these attributes.

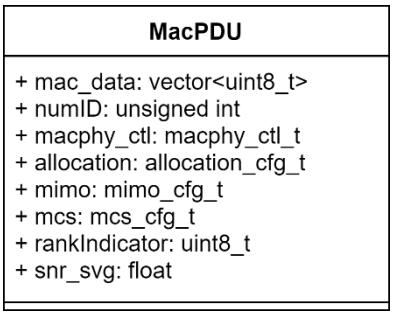

Figure 7. Simplified MacPDU class diagram.

Figure 7 omits some of *MacPDU*'s attributes that are not used by the MAC layer. The attributes listed in the figure are described in Table 1.

Table 1. Attributes of *MacPDU* class.

Attribute	Type	Filled by	Description
mac_data	*vector<uint8_t>*	MAC	MAC PDU buffer, containing MAC Header, MAC Payload and CRC. The latter is filled by *L1L2Interface* class.
numID	*unsigned int*	MAC	Numerology ID.
macphy_ctl	*macphy_ctl_t*	MAC	MAC-PHY control structure, containing the sequence number of the PDU in the subframe.
allocation	*allocation_cfg_t*	MAC	Spectrum allocation, containing UE ID, first RB and number of RBs.

mimo	*mimo_cfg_t*	MAC	MIMO configuration structure, containing information such as number of antennas or if MIMO scheme is multiplexing or diversity.
mcs	*mcs_cfg_t*	MAC	MCS related structure, containing the QAM modulation, the number of information and coded bytes and also power offset.
rankIndicator	*uint8_t*	PHY	PHY informs MAC the rank indicator measured in reception.
snr_avg	*float*	PHY	PHY informs MAC the average SNR measured in reception.

With all attributes of *MacPDU* objects defined, the PDUs are sent to the PHY layer via *L1L2Interface*.

On UE Side, the PDU is received by *MacController::decoding()* thread, with MAC Header and a Multiplexer object created to help remove MAC Header with *Multiplexer::removeMacHeader()* procedure, which returns all information coded into the header to help decoding the aggregated SDUs. The method *Multiplexer::getSDU()* is implemented to help the dissolution of MAC PDU into MAC Control or Data SDUs. Control SDUs are treated by MAC, while Data SDUs are written to Tun interface for Linux IP layer treatment.

Details of the MAC and PHY communication integration are described in section 5.1.

2.8.2 Uplink

The UE uplink MAC Scheduler procedure is different from the BS MAC scheduling because UE has semi-static uplink spectrum allocation. Semi-static stands for a static configuration, that will be changed if, and only if, a MACC SDU is received with uplink reservation parameter update.

The only operation performed by *Scheduler::scheduleRequestUE()* is to verify if the bytes enqueued for uplink transmission require less RBs than the RBs reserved for UE. After this verification, UE fills the unique PDU with the same procedure used by BS on downlink: *Scheduler::fillMacPdus()*. The remaining procedures are also the same and UE forwards the PDU through L1L2Interface.

On BS side, the procedure *MacController::decoding()* captures the PDU, removes MAC Header and decodes the SDUs aggregated. This is the same operation performed on UE after downlink reception.

2.9 *L1L2Interface* class

The L1L2Interface holds the message queues used to provide communication with PHY Layer. These message queues will be defined on integration details section 5.1, in Table 5. The major role of L1L2Interface class is to provide functions to send/receive PDUs and to send/receive Interlayer messages.

For both downlink and uplink, the *L1L2Interface::sendPdus()* method is called after *MacPDUs* elaboration in *Scheduler* and receives these PDUs as a parameter. Right before sending these PDUs to PHY, Cyclic Redundancy Check (CRC) calculation is performed, and CRC calculated is inserted at the end of the MAC PDU. On the receiver side, the PDUs are received via *L1L2Interface::receivePdus()* method and this method will immediately verify if CRC checks the expected value with *L1L2Interface::crcPackageChecking()* function.

The interlayer messages are sent and received with *L1L2Interface::sendControlMessages()* and *L1L2Interface::receiveControlMessages()*, respectively.

3 PHY Implementation Details

As pointed out in D6.1 Deliverable [3], the 5G-RANGE Physical layer (PHY) software implementation utilizes the GNU Radio framework. This section will describe the development details of the functions performed by the PHY layer.

3.1 Overview

The core functions implementation of the 5G-RANGE PHY layer employs C++ object-oriented programming to represent its main entities and core functionality. When compiled, the C++ code can achieve the performance required for real-time operation, also providing a reasonable level of abstraction and encapsulation.

The top-level transmitter chain, developed in the GNU Radio Companion framework has its structure defined through a Graphical User Interface (GUI), as depicted in Figure 8.

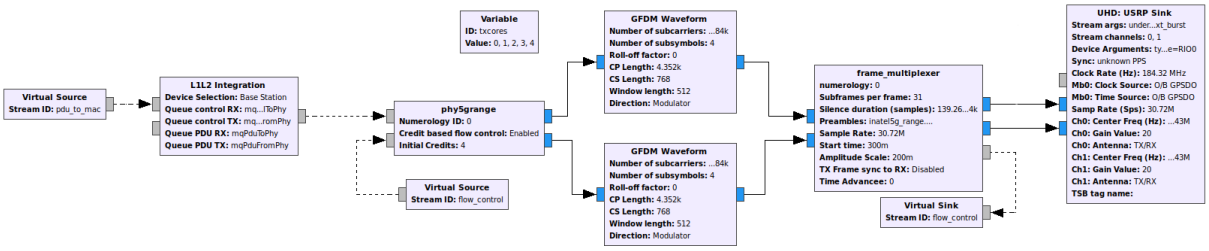

Figure 8. GNU Radio flow graph for the 5G-RANGE transmitter.

The transmitter chain comprises three main GNU Radio blocks, namely the *phy5grange*, the waveform modulator, and the frame multiplexer. Similarly to the transmitter structure rendering, the top-level receiver chain, also developed in the GNU Radio Companion environment, has its structure defined by the diagram shown in Figure 8.

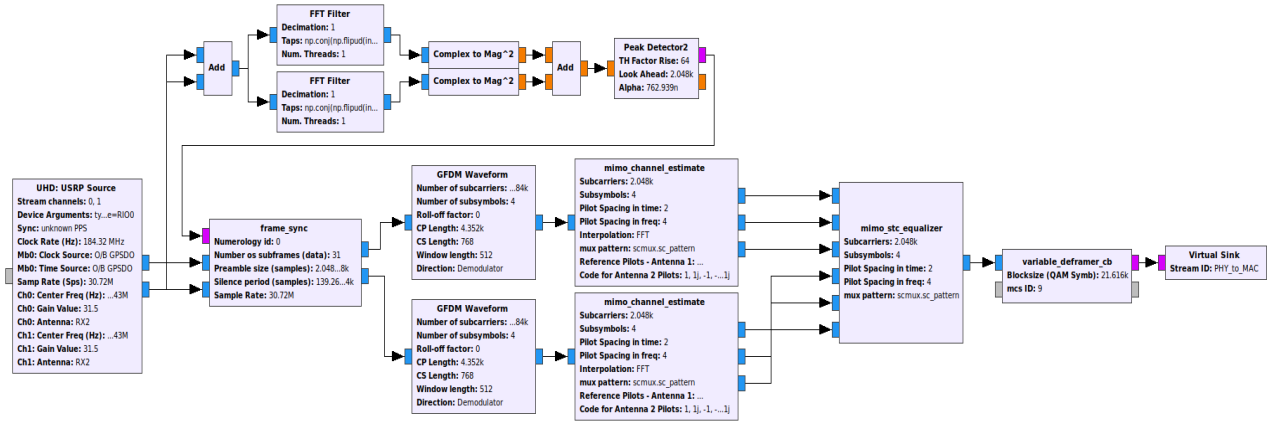

Figure 9. GNU Radio flow graph for the 5G-RANGE receiver.

The receiver chain makes use of shared classes also employed by transmission processing. The main blocks include the synchronization, the waveform demodulator, the MIMO decoder and the deframer (polar decoder and demultiplexer).

3.2 *Phy5grange* GNU Radio block

The *phy5grange* block is responsible for data formatting, transmitting it accordingly to the 5G-Range PHY definitions. It processes the packets originated at higher layers, on a subframe basis, and produces the complex values information symbols according to each transmission antenna. This block is responsible for the Advanced Coding for Remote Areas (5G-ACRA), Multiple-Input Multiple-Output techniques for Remote Areas Applications (5G-MIMORA), the generation of part of the pilot signals used by the Inner Receiver for Remote Areas applications (5G-I2RA) and the PHY framing at the subframe level.

The software structure used in this block encapsulates every signal-processing block in a specific class. Each class's functionality has an implemented function that takes, as an argument, an object holding the input data. The result is stored back to the same data object. This data object sequentially calls all signal processing functions until the required operations are complete and the subframe is ready for transmission.

Such a structure intends to make the top-level code of the transceiver chain intuitive and clean. This approach also aims for making the PHY implementation easier to maintain and extend by adding or replacing the signal processing chain classes.

The *phy5grange block* implementation description, as its main classes and core aspects of development, are presented below.

3.2.1 MacPDU class

The MacPDU class is responsible for storing the data and the PHY layer's information to transmit the data as intended by the MAC layer. This information includes the Modulation and Coding Scheme (MCS), MIMO configuration, targeted UE identification, and the allocated radio resources.

The transmission and reception processes use this class. Channel information, such as the average signal-to-noise ratio (SNR), is also stored in this structure.

3.2.2 Polar Encoder and decoder classes

This class implements the polar encoding functions, necessary to code a MacPDU object's information as instructed by the MAC layer. The interface exposed by this class encapsulates all functionality needed to perform the Forward Error Correction (FEC) coding on the information with the code rate determined by the MAC. Additionally, this class performs the segmentation of data in codewords as needed and realizes the rate matching procedure.

The decoder function implementation makes use of the AFF3CT library [4], which can achieve real-time decoding performance. However, to meet the flexibility required by the 5G-ACRA, the AFF3CT decoder objects need an on the fly configuration at every subframe.

The polar encoding and decoding are computationally intensive tasks. Execution in real-time demands an implementation able to take full advantage of CPU’s parallel processing capabilities. While the AFF3CT library takes advantage of the fine-grained parallelism using the Single-Instruction Multiple Data (SIMD) approach [4], the encoder and decoder classes use thread pools to process many FEC codewords concurrently. This architecture takes advantage of modern processors' multi-core architecture without suffering the penalty of spawning a new thread for every codeword.

3.2.3 Quadrature and Amplitude Modulation class

The QAM modulator class takes the input information bytes after the FEC coding and maps this data into QAM symbols. This operation proceeds two steps. According to the modulation order, the first step is to split the information bytes in chunks of bits or symbols. The second step is to look up the

corresponding complex-valued QAM symbol. Further dividing the look-up process into two identical look-up tables (real and imaginary parts), reduces its size to the modulation order's square root. Therefore, the reduction increases the frequency of use of every position of the look-up table, which, in turn, improves the cache access ratio and throughput performance.

The QAM demodulation process is the opposite way of the modulation. It is responsible for extracting the coded bits from the received noisy QAM symbols. However, instead of bits, this block must deliver a soft metric, the Log-Likelihood Ratio (LLR) for each bit, as the FEC decoder relies on the LLR for robust decoding. The algorithm implemented for the decoder uses an approximated calculation, providing capacity to deliver almost 6 times higher throughput and about 10 times fewer resources on an FPGA with negligible error [5].

3.2.4 MIMO Encoder and Decoder classes

The MIMO encoder and decoder classes handle the MIMO codification as configured by the MAC. These classes can implement MIMO for diversity or MIMO for spacing multiplexing with configurations up to 2x2. The encoder objetct receives the QAM symbols apre provides a data stream for each transmit antenna. The decoder object will combine the data received at each antenna and perform the STC decoder operations, if MIMO for diversity is used, or separate the information from each transmit antenna using zero-forcing algorithm, if spatial multiplexing is employed. In both cases, the channel estate information must be received from the channel estimation block.

3.2.5 Grid class

The grid object is a representation of the resource grid. It offers the storage structure of all resource elements in the grid, but most importantly, it provides an interface to map the MacPDU objects to the resource grid, as requested by the MAC layer.

This class also implements all functions required to map the pilot subcarriers as, described in D3.2 [6]. Figure 10 shows the time-frequency distribution of data and control resource elements.

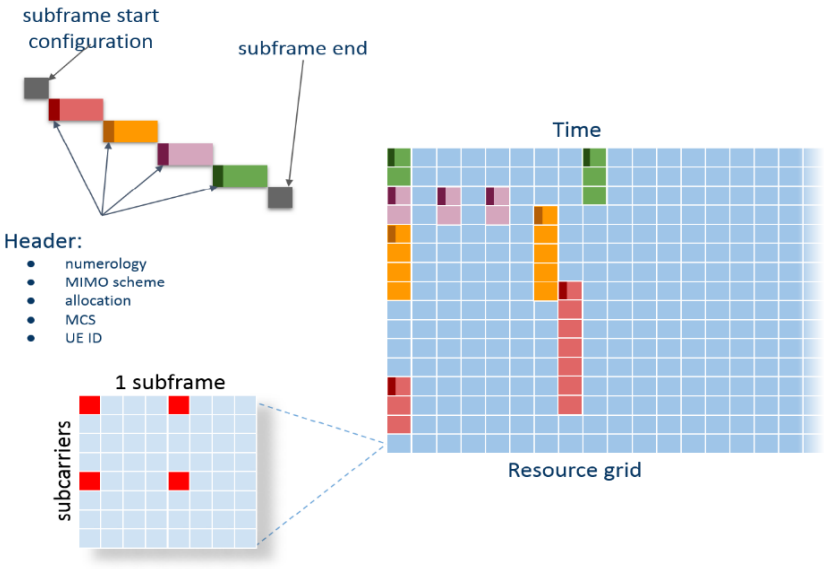

Figure 10. Resource grid with input control allocation

Figure 11 illustrates the receive processing flow from the data extraction of the resource grid up to the MAC layer interface.

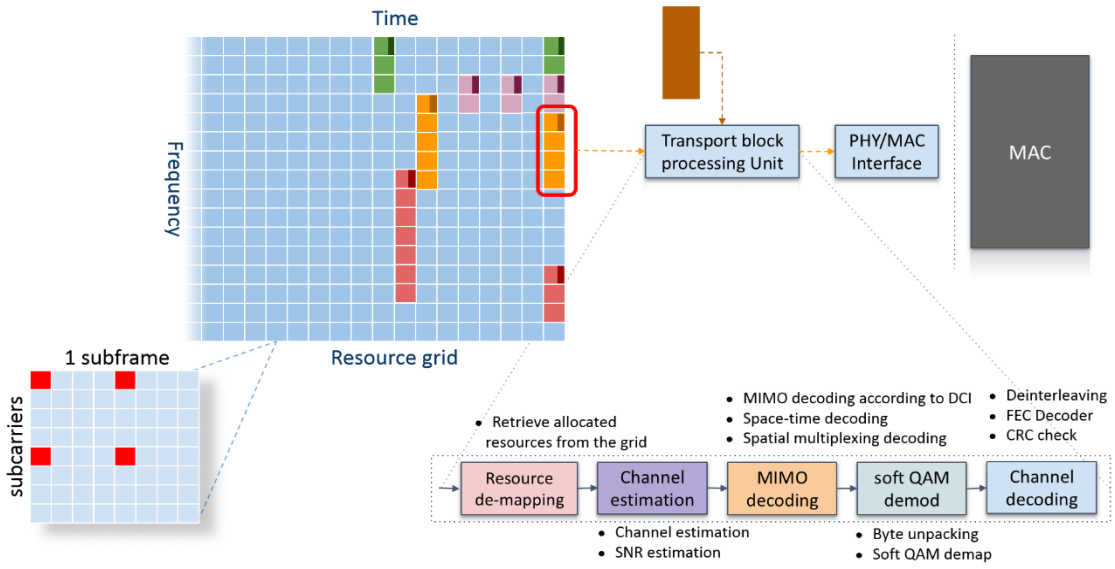

Figure 11. Resource grid with the receiver processing structure.

3.2.6 CSI class

At every subframe, the receiver estimates current Channel State Information (CSI) since this information is required for the MIMO decoding process. The *CSI* class stores this information and provides the functions to implement the 5G-I2RA algorithms related to the respective channel estimation.

The function responsible for retrieving the CSI encapsulates all steps required to derive the channel state from the complex-valued symbols received and stored at the grid object. Additionally, the class provides functions to interpolate the channel frequency response in such a way it executes de estimation, not only at the pilot subcarriers, but also for all data symbols by using an FFT-based interpolation algorithm, implemented with the open-source *FFTW* library [7].

3.3 *Waveform* GNU Radio block

This block is responsible for modulating the complex-valued information symbols, using the waveform parameters according to the selected numerology (number of subcarriers, subcarrier spacing, Cyclic Prefix (CP) duration and number of subsymbols).

This block makes use of a library base, previously implemented in C language. The developed code explicitly takes advantage of Intel Advanced Vector Extensions 2 (AVX2) present on modern x86 processors. Thus, it does not rely on compiler optimizations. The execution is accomplished using Intel Intrinsics, which are C style functions wrapping the SIMD machine code instructions available in the processor.

Despite the successful execution, this approach limits the optimized portion of the source code's portability for other target architectures such as Advanced RISC Machine (ARM). In this case, the software could utilize ARM NEON Intrinsics library.

3.4 *Frame Multiplexer* GNU Radio block

The 5G-RANGE frame is composed of 32 radio subframes, from which 31 carry information. The 32nd subframe period has two parts: The silence period for spectrum sensing and the Primary Synchronization Signal (PSS).

The frame multiplexer block is responsible for the frame's final assembly, concatenating the subframes and the reference signals for each transmission antenna.

Since the 5G-RANGE system uses the TV White Spaces (TVWS) spectrum as secondary user, the PSS takes place only in the free portion of the spectrum, avoiding interference with the Primary User (PU). The frame multiplexer block receives metadata information, called tags in GNU Radio, from upstream blocks at every subframe time slot. It also contains the spectrum mask available for transmission. The PSS is then located accordingly.

The frame multiplexer block includes tags at the beginning of every frame to instruct the SDR block about the exact timing position to start a frame transmission. In this way, it is possible to synchronize all signals in the uplink direction with the frame start in the downlink. Figure 12 depicts a measurement of the transmitted waveforms for uplink and downlink directions.

Figure 12. Synchronization of downlink and uplink directions.

Notice that both frame transmission starts at the same time. This ability is critical to allow many UEs to share the uplink spectrum without interfering in each other.

3.5 Synchronization GNU Radio blocks

The timing synchronization reference, computed on the software-defined transceiver, comes from the cross-correlation between the received signal and a known sequence, the PSS [6]. The peak of this cross-correlation process occurs at the start of the frame, allowing the receiver to estimate the exact position to start all chain operations processing as intended. The block *Peak Detector2,* which is part of the default GNU Radio block set, calculates and identify this cross-correlation peak.

After the frame start identification, the *frame_synchronizer* block splits the 31 data subframes to the waveform demodulator block. The samples belonging to the silence period are delimited and forward to the spectrum sensing block processing. This block also registers the absolute time when the frame start

occurred at the radio. The uplink transmitter will use this timestamp information to trigger the uplink frame start precisely.

3.6 Spectrum sensing GNU Radio block

Within the several existing spectrum sensing techniques, the chosen one for the PoC implementation employs the detection based on cyclostationary properties of the primary signal. The traditional schemes using cyclostationay properties requires previous knowledge of the primary signal waveform for detection processes. They have a robust operation on noisy environments, detecting signals even in low SNR areas.

In the PoC, the primary signal used for the spectrum sensing demonstration is a DTV signal. Because of the available infrastructure, an equipment that generates the ISDB-Tb (Integrated Services of Digital Broadcasting - Terrestrial) signal has been used. Since ISDB-Tb employs OFDM as waveform, the sensing algorithm explores the correlation property of the signal structure and its cyclic prefix to detect the incumbent.

This spectrum sensing detection technique can also identify the DVB-T standard (European standard for DTV) signal. The DVB-T also uses OFDM with cyclic prefix. A significant difference between the two standards is that the channels in DVB-T have 8 MHz of bandwidth while the channels in ISDB-Tb is only 6 MHz wide. The 5G-RANGE signal, using all its bandwidth of 24 MHz, occupies four ISDB-T channels, while the same bandwidth corresponds to only three DVB-T channels.

In the 5G-RANGE structure, a frame is composed of 32 subframes, each one comprising a 4.6 milliseconds duration. The last subframe is a silence timespan for the sensing purpose. In this period, the BS and UEs do not transmit any signal, making possible, for each UE, to use this period for sweeping the channels while looking for the presence of primary users.

The sensing algorithm was implemented in Python and embedded it into a GNU Radio block framework. The block has as input to received samples of the silence period and, as output, it produces a message with the state (occupied or not occupied) for each one of the four 6 MHz channels that compose the maximum 5G-RANGE bandwidth. The L1 to L2 integration block receives this message and delivers the UE’s MAC layer information. The UE's MAC uses the control channel to update the BS with the state of spectrum occupancy in its respective area. After receiving the UE’s updates, a cooperative sensing algorithm, processed in the BS, allocates the available resources to the 5G-RANGE users.

4 5G Core Implementation Details

The 5G core related components are intended to provide end-to-end IP network connectivity to 5G-RANGE user terminals in the proof-of-concept. The main purpose of these components is to provide an implementation for the user-plane protocol stack of a Non-3GPP InterWorking Function (N3IWF), as defined by the 3GPP, to support the connectivity of UE to the 5G core network via untrusted non-3GPP access (as it is the 5G-RANGE access network).

The implementation of these functions has to be included at both ends of the communication across the access network, so, on the one hand, a 5G Core prototype has been implemented in the provider side and a UE gateway (UE/GW) component has been implemented to support UE connectivity.

The implementation details of these components have been described in deliverable 5.3. However, a summary has been included in this document for the sake of completeness.

4.1 5G Core

This component will support the connectivity of any UE and gateways of the proof-of-concept through the 5G-RANGE access network. It has been developed to serve for validation purposes in the proof-of-concept of the project. Hence, the prototype does not provide all the functions and services specified in [8] for the 5G core network. In particular, it does not implement a control-plane protocol stack, as defined by 3GPP.

The 5G Core component has been prototyped as a Virtual Network Function (VNF). This way, it may automatically be deployed by the 5G-RANGE MANO platform. The VNF provides the implementation of the user-plane protocol stack of a Non-3GPP InterWorking Function (N3IWF), defined by 3GPP, to support the connectivity of UE to the 5G core network via untrusted non-3GPP access. In addition, it provides routing functionalities towards external networks. This way, the 5G Core VNF can be understood as a component that provides a basic implementation of the user-plane protocol stack for a N3IWF and a User Plane Function (UPF). The development has been based on the Linux ip-gre module and the ipsec-tools and racoon Linux packages, which provide the required support for Generic Routing Encapsulation (GRE) [9] and IP security (IPsec) [10], respectively.

4.2 UE/GW

The UE/GW component provides the functions of a 3GPP UE. In concordance with the implementation requirements for the 5G Core component, the prototyped UE/GW does not include all the functions and services specified by 3GPP for the 5G system. In particular, it implements the user-plane protocol stack to support access to the 5G core network through untrusted non-3GPP access. Besides, the UE/GW supports IP routing functions. Hence, it may be used to enable the access of multiple end-user devices to the 5G-RANGE access network. The UE/GW component has also been implemented as a VNF.

4.3 Supported protocols and components deployment

The protocol stack enabled by both the 5G Core and the UE/GW is provided below in the figure, along with the resource requirements to support their execution as VNFs. The picture also outlines the role of the 5G Core and the UE/GW in the provision of end-to-end IP network connectivity towards external networks.

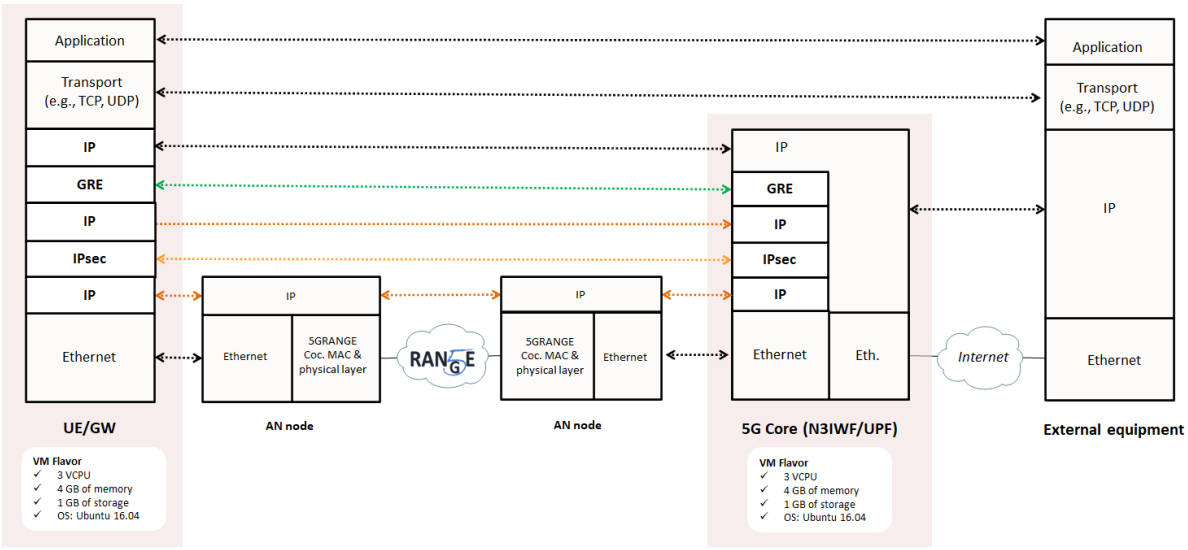

Figure 13. 5G Core and UE/GW supported protocol

To facilitate the deployment of these two components as VNFs and in general, the whole network layer architecture that has been described in D5.2 [11], the Management and Orchestration Platform and the NFV infrastructure of 5G-RANGE have been deployed in the 5G Telefonica Open Network Innovation Centre (5TONIC) [13], located in Madrid, where UC3M participates as a member. An overview of this deployment is shown in the figure below.

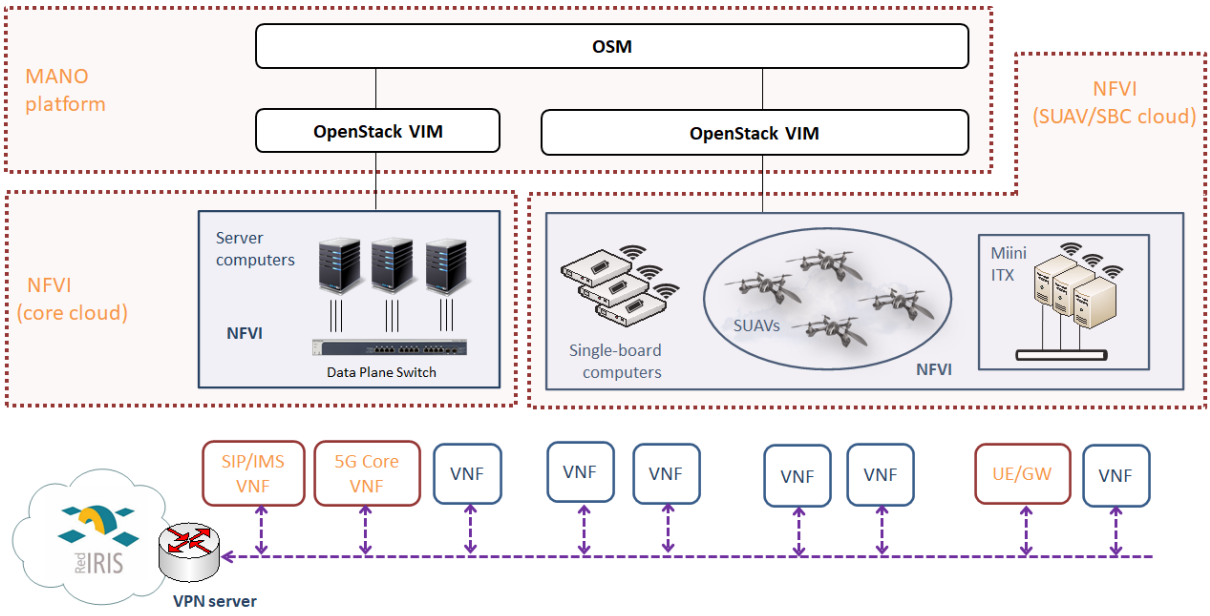

Figure 14. The 5G-RANGE MANO platform at 5TONIC

More details about these deployments can be obtained from D5.3 [12] together with different installation instructions.

4.4 Components validation

The validation of these components has also been detailed in D5.3 [12].

In addition, two different articles have been published including different validation scenarios performed with these components [14] and [15], and an additional article has just been submitted to the IEEE Communications Magazine journal showing the integration of the PHY layer and the Network layer PoC components.

5 Prototype Integration Details

The 5G-RANGE PoC comprises three components: PHY, MAC layer, and Network layers. Due to the dependency on each other, the MAC and PHY layers need an Application Programming Interface (API), as described in deliverable D6.1 [3], allowing a frequent message exchange among these two entities.

The Network layer, on the other hand, uses the L3 transport services provided by the MAC layer to connect the UE to the network Core.

5.1 PHY – MAC Integration Details

The communication between the MAC and the PHY is accomplished by exchanging control messages and messages containing the PHY's payload data. In deliverable D6.1 [3], an overview of the MAC/PHY API was described. However, during the implementation, some changes were required to allow efficient communication among these entities.

An updated table with the control and data messages exchanged between MAC and PHY is presented in Table 2, along with the sequence of these messages, for Transmission (TX) and for Reception (RX) in the gNB and UE sides. Besides message carrying MAC PDU, there are a set of control messages that carry PHY layer parameters that enables its configuration, and PHY information for MAC. These MAC/PHY control messages are *BSSubframeTX.start*; *BSSubframeRX.start*; *UESubframeTX.start*; and *UESubframeRX.start*. Previous control messages parameters have the scope of use at the local PHY and MAC layers, or at the Transmission side. Contents of these messages are present in Table 3.

There is another set of control message that is part of the MAC Control (Ct), which carries specific configurations for each of the UEs, and must be used on both sides, TX and RX. This MAC Ct is prepended to MAC PDU in the communication between MAC and PHY.

Table 2. Messages exchanged between MAC and PHY.

gNB/UE, TX/RX	Types	Direction	Description
gNB TX sequence	*PHYTX.indication*	PHY -> MAC	PHY informs MAC layer to transfer MAC PDU(s)
	BSSubframeTX.start	MAC -> PHY	PHY configuration parameters
	MAC PDU with MAC Ct	MAC -> PHY	MAC PDU transfer with MAC Ct prepended
	SubframeTX.end	MAC -> PHY	MAC PDU transfer end
UE RX sequence	*UESubframeRX.start*	PHY -> MAC	RX Metrics
	MAC PDU	PHY -> MAC	MAC PDU transfer
	SubframeRX.end	PHY -> MAC	MAC PDU transfer end
UE TX sequence	*PHYTX.indication*	PHY -> MAC	PHY informs MAC layer to send MAC PDU
	UESubframeTX.start	MAC -> PHY	PHY configuration parameters
	MAC PDU with MAC Ct	MAC -> PHY	MAC PDU transfer with MAC Ct prepended
	SubframeTX.end	MAC -> PHY	MAC PDU transfer end
gNB RX sequence	*BSSubframeRX.start*	PHY -> MAC	RX Metrics
	MAC PDU	PHY -> MAC	MAC PDU transfer
	SubframeRX.end	PHY -> MAC	MAC PDU transfer end

Table 3. MAC/PHY control messages parameters list.

MAC/PHY messages	Parameters	Description
BSSubframeTX.start	NumUEs	Number total of UEs in the system/network
	NumPDUs	Number of MAC PDUs to transmit in the next subframe
	FLUTDL	Fusion Spectrum Analysis LUT (downlink)
	ULReservation	RBs UL for each of the UEs (uplink). Number of vectors according to NumUEs:
	Numerology	PHY Layer Numerology
	OFDM/GFDM	PHY Layer Numerology OFDM/GFDM
	RxMetricPeriodicy	CSI Periodicity for Wideband SNR, in number of subframes
UESubframeTX.start	ULReservation	RBs UL RB Reserved for the UE
	Numerology	PHY Layer Numerology
	OFDM/GFDM	PHY Layer Numerology OFDM/GFDM
	RxMetricPeriodicy	CSI Periodicity for Wideband SNR, in number of subframes
UESubframeRX.start	WindebandSNR	132 SNRs per RBs, with RBs within the UL Reservation range
	Rx Metrics	Bitwise indicating Ch 1 to Ch4 Spectrum Sensing Measurement (1= idle; 0= occupied)

As commented in section 2.8.1, the gNB DL transmission can have multiple MAC PDUs for each UE destination, and it can also have more than one MAC PDU for a single UE destination. PHY layer requires that the RB allocation for one MAC PDU must be contiguous, meaning that within a group of RBs required for DL transmission to a UE, if there is a group of RBs not available due to the presence of a PU, the MAC scheduler performs the segmentation of the MAC PDU, resulting in a non-contiguous allocation of RBs with separate MAC PDUs with destination to the same UE.

Figure 15 presents an example for gNB DL transmission with two MAC PDUs to be sent to UE 2, and one MAC PDU to be sent to UE 1. These MAC PDUs with MAC Ct are transferred from MAC to the PHY layer individually, with MAC PDU becoming Transport Blocks (TBs) in the PHY layer. The MAC PDU structure is described in section 2.8.1, Figure 6.

MAC Ct is prepended on MAC PDU in the TX side, to transfer control information to configure the PHY layer to transmit, and this control information is also transported along with the MAC PDU (as a TB) to the RX side, where the PHY layer uses it to configure the reception of this MAC PDU. MAC Ct control information content can be seen in Table 4. For gNB DL transmission, MAC Ct carries control information for the Downlink Control Information (DCI) on the PHY layer. For UL transmission, MAC Ct carries control information for the Uplink Control Information (UCI) on the PHY layer.

The interface between MAC and PHY is composed of four message queues, as described in Table 5.

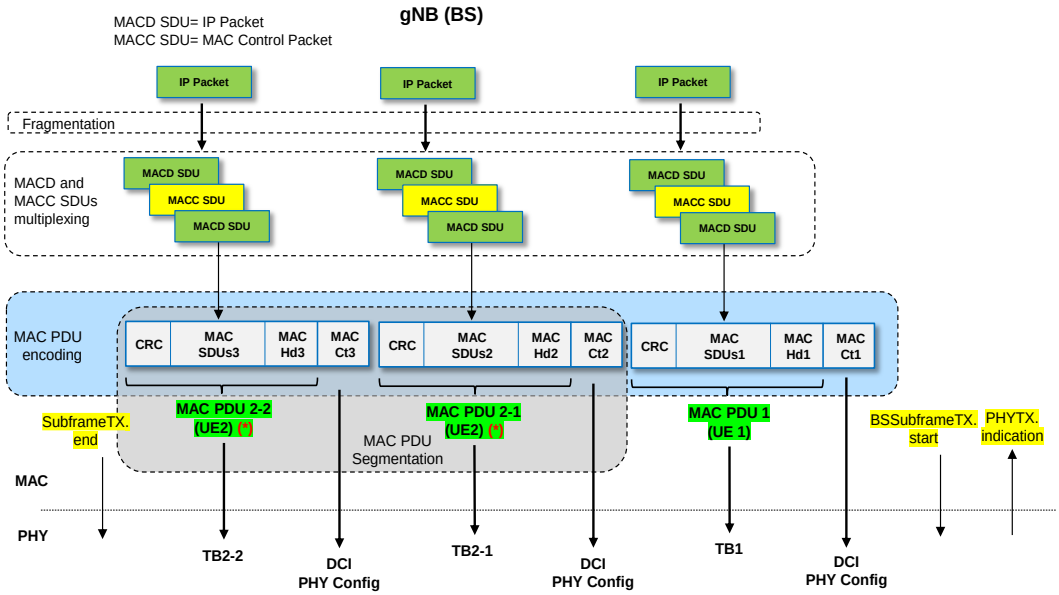

Figure 15. gNB MAC PDU Fragmentation, Multiplexing, Encoding, and Segmentation, and the MAC PDUs with prepended MAC Cts dispatch to the PHY layer. In addition, the communication sequence of the control messages can be seen.

Table 4. Contents of MAC Ct field.

MAC Ct Type	Parameters	Description
MAC Ct (BS TX and UE RX) Contained into MacPDU class	MAC5GR DA (UE)	For the MAC layer, address size is 4 bits. In the reception (RX) there is a need to adjust the No of bits between PHY and MAC
	RBstart	RB ID start of allocation
	NumRBs	Number of contiguous RBs allocated
	AverageSNR	Average SNR for the used RBs. Parameter that exists only on reception (RX)
	DL MCS	DL MCS used in the initialization, when there´s no Rx Metric for all the UEs
	MIMOConf	MIMO configuration. MIMO/SISO.
MAC Ct (UE TX and BS RX) Contained into MacPDU class	MAC5GR DA (BS)	For the MAC layer, address size is 4 bits. In the reception (RX) there is a need to adjust the No of bits between PHY and MAC
	RBstart	RB ID start of allocation
	NumRBs	Number of contiguous RBs allocated
	AverageSNR	Average SNR for the used RBs. Parameter that exists only on reception (RX)
	UL MCS	UL MCS used in the initialization, when there´s no Rx Metric for all the UEs
	TPC	UL Transmission Power Control (TPC) in dBm
	MIMOConf	MIMO configuration. MIMO/SISO.

Table 5. Message queues between MAC and PHY.

Message Queue	Direction	Function
mqMacToPhy_Data	MAC to PHY	Data
mqMacToPhy_Ctrl	MAC to PHY	Control
mqPhyToMac_Data	PHY to MAC	Data
mqPhyToMac_Ctrl	PHY to MAC	Control

The data transmission at the PHY level occurs on a subframe basis. The control queue from the MAC to the PHY carries messages to configure the PHY for the next subframe. In the opposite direction, the messages report the medium status on the last subframe, including Channel State Information (CSI) and Spectrum Sensing Measurement (SSM) reports.

At the MAC side, the Protocol Control module manages the information flow between the MAC and PHY. It is written in C++ and is responsible for the protocol implementation. At the PHY side, the block that handles the information flow between MAC and PHY is written in Python.

5.2 MAC – L3 Integration Details

One of the key concerns about MAC layer operation was how to forward application packets into the 5G-RANGE network. A series of approaches for dealing with Linux IP stack were studied and the Linux TUN kernel virtual network device driver was chosen.

5.2.1 L3 Interface Implementation

The network operation of Tun Interface is exemplified in Figure 16, which describes a downlink transmission and reception procedures with IP packets captured from Linux Network IP Layer.

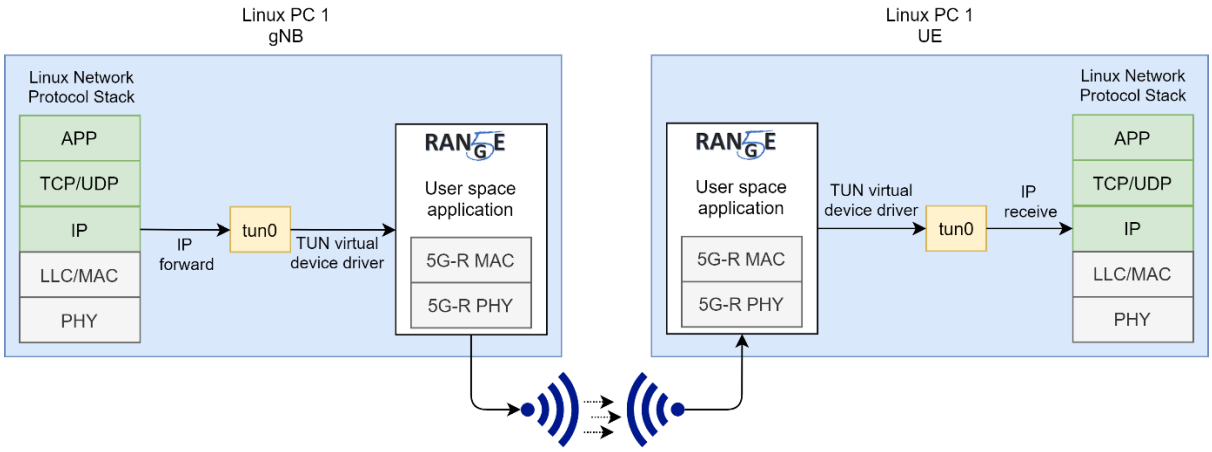

Figure 16. Demonstration of downlink procedure with Network IP layer packets captured by TUN virtual device.

At gNB transmission side, Linux IP layer can receive IP packets from local applications, running in "APP" layer, or from remote network applications running in other machines. In case of remote applications, the packets arrive through a physical device (e.g. Ethernet device) and are decapsulated in lower layers, PHY and LLC/MAC before arriving at IP layer. In both cases (local or remote applications), IP layer decides where to forward the packet based on its destination IP address and local Linux routing table.

Tun Interface virtual device is represented in Figure 16 by its default name, "*tun0*". This is a virtual device created by TUN driver to perform packet capture at IP layer level. This device can be set with IP address, Maximum Transmission Unit (MTU) and other parameters like a physical device interface, so that the L3 layer is able to forward IP packets to this virtual device. But unlikely a physical device, *tun0* will forward these IP packets to a user-space application that uses TUN interface API to receive these IP packets. This application is represented by 5G-RANGE user space application, containing 5G-RANGE protocol stack: 5G-RANGE (5G-R) MAC and 5G-R PHY.

At UE reception side, all the process is basically the same for transmission, but mirrored. Application receives samples, decode them, extract MAC PDU and writes the IP packet received to Tun Interface. Then, *tun0* device delivers packet for receiver-side IP layer treatment.

As mentioned earlier, for Linux network stack, Tun Interface appears as a device with the following configurations available:

- IP address and subnet mask: the IP address of Tun Interface can be set with a simple "*ifconfig*" Linux terminal call, like any other network device. The subnet mask can also be set with IP address to allow the Linux IP network layer to proceed with forwarding packets to Tun Interface according to IP address and subnet configured. Specific routes can also be set to perform packet forwarding to Tun Interface
- Maximum Transmission Unit (MTU): the MTU of Tun Interface can also be set and this is a crucial point of help for the MAC layer. This is because Linux IP layer itself fragment packet if necessary so that the IP packets size do not extrapolate the MTU number of bytes. Then at reception, Linux IP layer reassembly the IP packets. This L3 functionality allied to setting MTU as the maximum SDU size that MAC and PHY layers can work with removes from the MAC layer these tasks of fragment and reassembly user data SDUs (IP packets).
- Tx Queue Length: the number of frames enqueued for the interface can also be changed, allowing stress tests to be executed. The queue treatment is described in section 2.7.

Other configurations about the Tun device can be changed. The parameters above detailed were listed because they were used in MAC implementation. The MTU parameter can be set in the configuration file. When allocating Tun Interface, MAC implementation calls "*ifconfig*" to set interface MTU and to bring up Tun device. After the MAC stating process, IP address of Tun interface can be changed using Linux terminal.

5.2.2 Tun Interface Implementation

To use Tun Interface in a user-space implementation, it is required from Linux driver to open a file descriptor at the system directory "*/dev/net/tun*". Then, an interface requirement object, of type "*struct ifreq*" must be instantiated with interface flags (the flag to set the interface as a Tun interface is "*IFF_TUN*") and interface name can also be chosen here. After setting interface requirement, a call to system API *ioctl()* is made to link the new virtual interface to the file descriptor declared early and to the interface requirement object configured. All these operations are made into the function *CoreTunInterface::allocTunInterface()*.

After allocating Tun Interface and linking the virtual interface to the file descriptor, read and write operations from and to the interface are trivial using *read()* and *write()* methods from system library *unistd.h*. Such operations are performed by the functions *CoreTunInterface::readTunInterface()* and *CoreTunInterface::writeTunInterface()*, respectively. Since reading from the file descriptor is a blocking operation, *select()* function call was used to verify if there is new information at the interface.

5.3 Core integration details

The integration of the network layer developments (in particular the 5G core and the UE/GW) is particularly challenging, because it includes prototypes and technologies that operate at different layers but also because not all the components are software developments.

To simplify the integration process a stepwise approach has been followed, 1) starting with a remote EU-BR integration of the main components, 2) then a local validation through a general testbed, including all the equipment (this was done in Madrid and a demonstration of this local validation was performed during the interim review) and 3) a remote final integration of the PoC network layer components at INATEL premises.

The final purpose of the whole integration in the first two steps is to be able to validate the PHY-NET layer protocol combination by means of the scenario shown in the figure below that includes end to end communications of voice and data (and therefore with different delay and throughput requirements) through the whole set of PHY and network layer developments (in particular CPE equipment, UE/GW and 5G core developments). Since the PHY-MAC layer integration is also taking place as described in section 5.1, the whole integration in the final demonstration would then be straightforward.

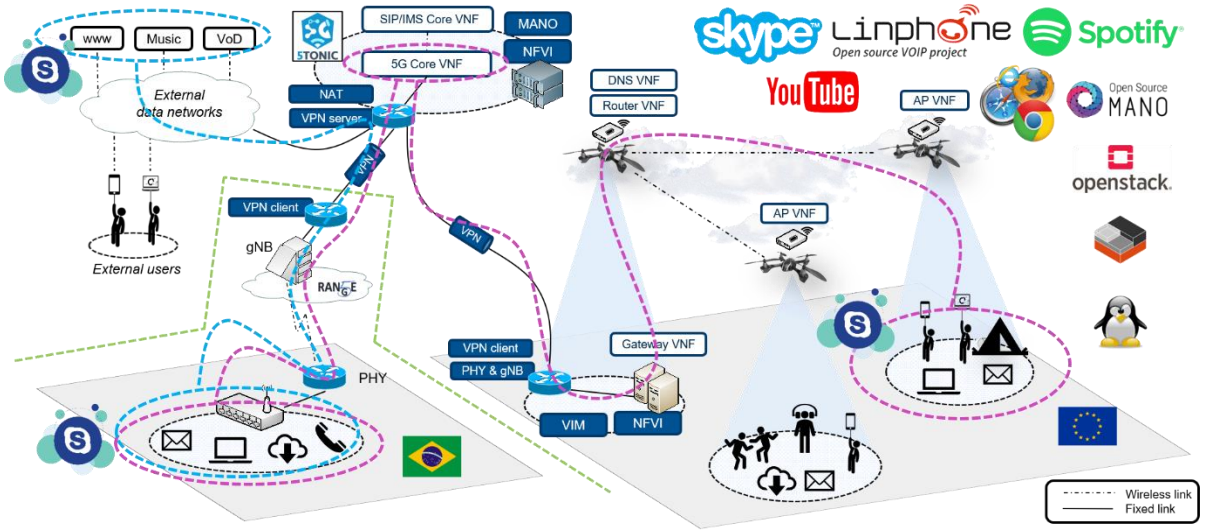

Figure 17. Validation testbed for PHY-NET layer integration

In this scenario two remote areas (BR left, EU right) share voice and data communications (pink line) and there is also a connection with the public Internet (blue line) for commercial services validation (Youtube, Spotify and Skype). This scenario includes the whole deployment of the network layer components described in section 4. Next figure shows the protocol combination validated in this scenario, with the CPE (PHY layer equipment), 5G Core and Gateway.

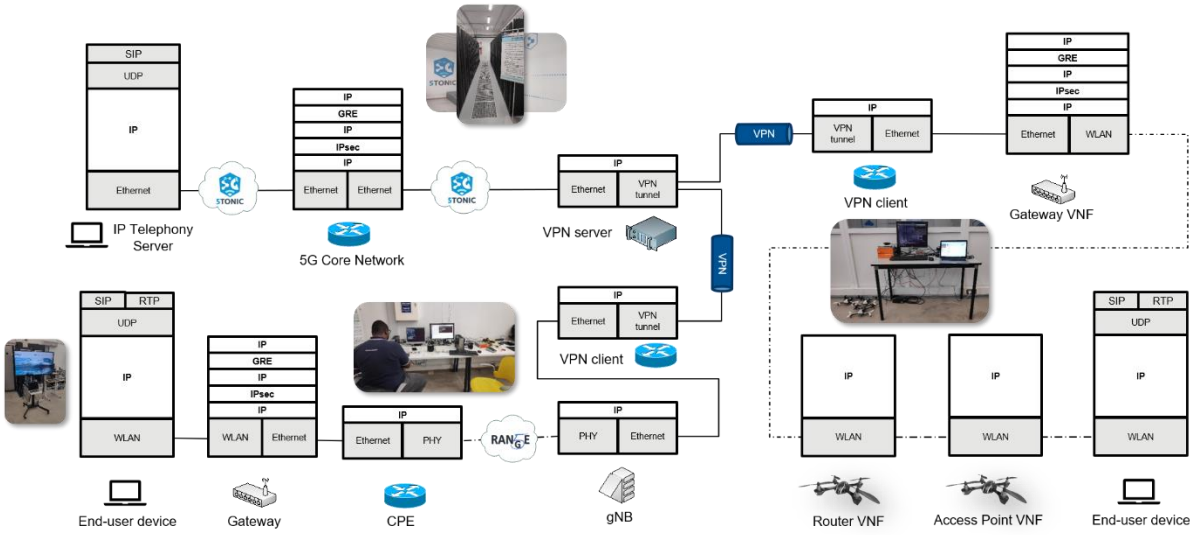

Figure 18. PHY-NET layer integration. Protocol scenario.

The third step includes a set of autoconfiguration scripts that will be used at INATEL site to integrate the 5G core and UE/GW software that is running at UC3M premises during the first two phases to facilitate the final PoC integration and validation.

5.3.1 Remote integration

As a first step, the remote area at Brasil was deployed (specifically at INATEL premises), using the physical-layer prototypes that were available there. Once the remote area was ready, it was interconnected with the 5TONIC laboratory using a VPN service.

This represented a very convenient way to work, as all the components of this remote area could be accessed via SSH as if they were physically at 5TONIC in Madrid (the utilization of the VPN service simplified all the integration activities).

After that, the transoceanic connection was characterized in order to verify that it was feasible to operate remotely for initial compliance testing and future measurement and experiments. We obtained measurements of the available bandwidth from a VM (at 5TONIC data centre) to the VPN client (located in Brazil), in both directions, besides RTT measurements. These tests have been going on for 30 days (a measurement of 600 seconds every hour).

In the following, it can be appreciated that the link is reasonably stable, with average throughput values of 15 Mbps (5TONIC -> BR) and 4 Mbps (BR -> 5TONIC). The RTT is also stable, with an average value of 200 ms. These results guarantee that it is possible to operate remotely, and it is also possible to perform the experiments that include multimedia services such as video on demand, VoIP calls, or video conferences. Moreover, these experiments can be programmed at any time due to the stability of the link.

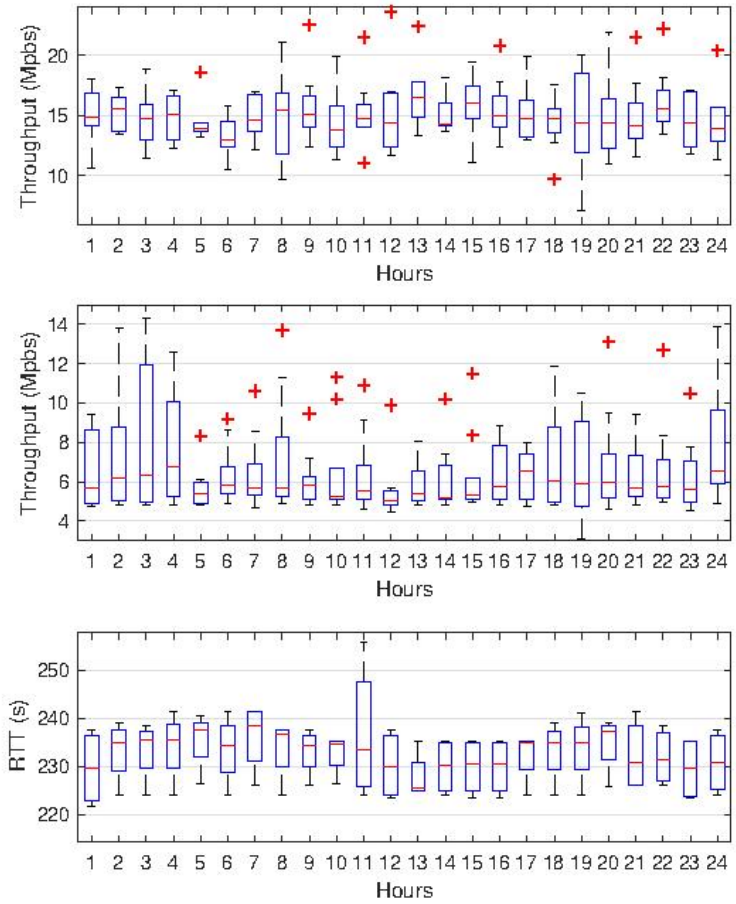

Figure 19. Transoceanic link characterization (UC3M-INATEL)

Once the interconnection was ready, we deployed and activated the different network-layer components at 5TONIC that were needed to support Internet access from the remote area in Brazil. This included the 5G core network VNF and a NAT function. With this, we verified that we could access Internet services from the remote area at INATEL.

Finally, we deployed a second remote area at 5TONIC, using the available NFVI of small-size drones and the MANO platform, which was used to deploy all the VNFs shown in section 4.3.

5.3.2 Local integration and demonstration

After the remote integration, the PHY equipment was brought to 5TONIC for local integration and validation. Due to the effort invested in the remote integration phase, the local integration was really simple and in less that 2h time the whole scenario as depicted in the figure was deployed.

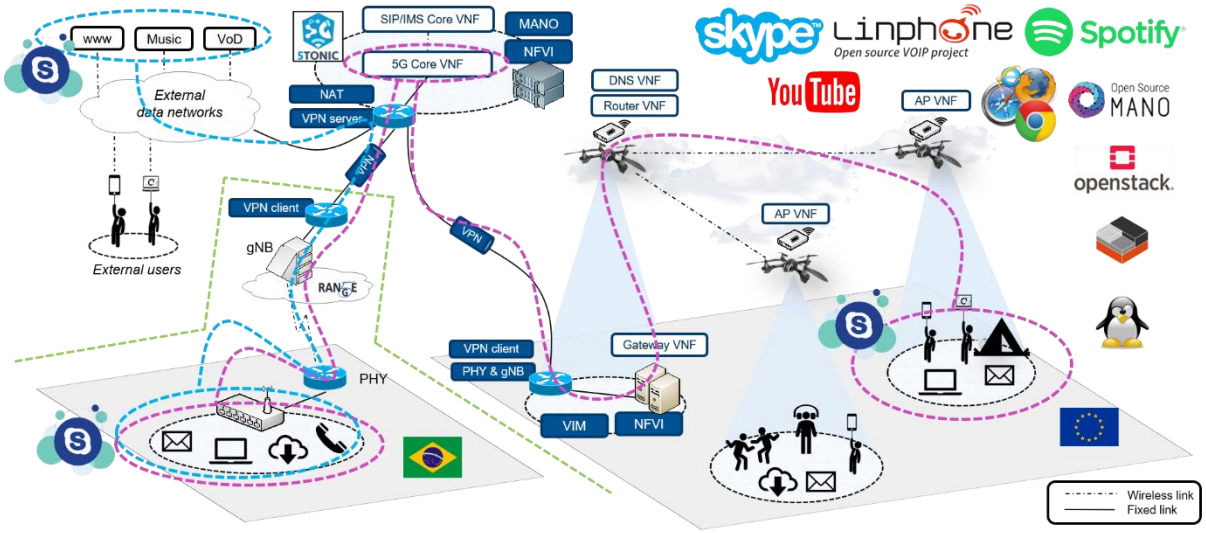

Figure 20. Validation testbed for PHY-NET layer integration

This scenario is similar to the one remotely deployed in Brazil but with all the equipment physically interconnected together in the same location. As it was seen in the demo, it allowed the provisioning of VoIP services (both based on standard SIP and Skype) and real-time video and audio streaming through Youtube and Spotify applications respectively. A joint article has been sent to the IEEE Communications Magazine.

The following figure shows the most relevant results obtain from the different tests. The top-left graph shows the traffic of a video call with three users using Skype. The first user is connected to the residential domain (BR area), the second user is connected to the SUAV domain (EU area), and the third user is connected to an external network (in the previous figure a Skype icon is locating each one of the three users). The traffic is captured in the residential domain user laptop. Around the second 65, all the call participants switch on the video camera. As can be appreciated, the transmitted traffic (0.5 Mbps) is half of the received traffic (1 Mbps). The user experience with the audio and video quality was satisfactory.

The top right graph shows the traffic of a video call with two users using the Bria software (SIP based). The first user is connected to the residential domain (BR area), while the second user is connected to the SUAV domain (EU area). The traffic is captured in the residential domain user laptop. Like in the previous experiment, both users switch on the video camera around the second 70. The average transmitted and received traffic is around 2 Mbps with the video-enabled. The bottom left graph shows the traffic received when playing a 4K quality video on the YouTube platform using the PHY/MAC equipment and without the equipment (used in the residential domain).

Similarly, the bottom right graph shows the traffic received when playing audio using Spotify. The two traffics streams (using PHY equipment or not) are different since both Youtube and Spotify use video/audio prefetching and the moment and quantity of retrieved video and audio depends on the network status at a particular moment (in any case the user experience was fine in both cases). In the bottom left graph, it can be appreciated in the black line how around second 60 a channel fading was emulated in the PHY equipment (using an attenuator) and later on how the signal is recovered.

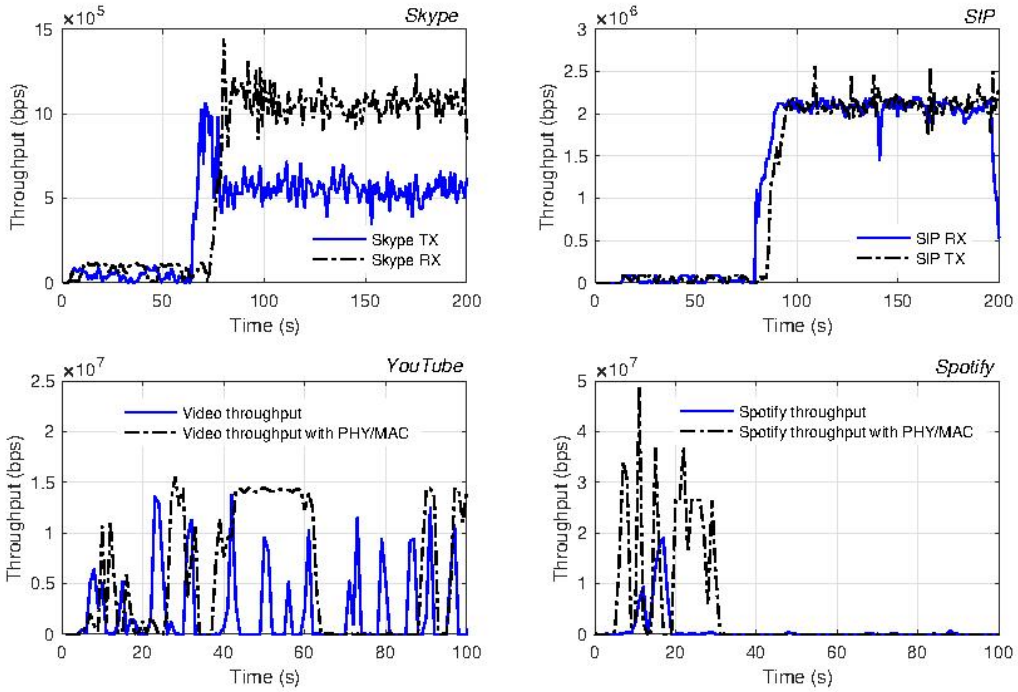

Figure 21. Throughput measurements performed in the test

Beyond the obtained numbers, the most relevant result in this case for the project is the successful integration of the PHY and the MAC layer.

6 Conclusion and Results

The prototype described in this document, designed based on the requirements defined in WP 2 and on the specifications and techniques presented in WPs 3, 4 and 5, is a proof-of-concept that demonstrante the main functions and capabilities of the 5G-RANGE network. All main features of the 5G-RANGE network has been implemented and the PoC is capable of convering the Voice and Data Connectivity, Smart Farm and Wireless Backhauling use cases.

The development of all prototype layers using SDR approach allowed the researchers and developers to reuse functions of other phases in the 5G-RANGE project, mainly those developed for software simulation. The developed prototype also enables a flexible framework for further use in research and technology transfer.

The 5G-RANGE BS and UE will be used for demonstration purposes in Santa Rita do Sapucaí – Brazil, under real conditions. The main goal is to demonstrate that the 5G-RANGE Network can achieve the 50 km distance with data rates up to 100 Mbps under real channel conditions.

Therefore, it is necessary to evaluate the PoC before this demonstration. Testing has to be executed taking into account all essential requirements to assure that all functions of the PoC are operating properly and no unknown states has been introduced in the PoC during the integration process. This evaluation and analysis will be presented in D6.3.

References

[1] 5G-RANGE, “Deliverable 2.1 Application and Requirements Report – Version 2”, February 2019.

[2] 5G-RANGE, “Deliverable 2.2 Architecture, system and interface definitions of a 5G for Remote Area network – Version 2”, February 2019.

[3] 5G-RANGE, “Deliverable 6.1 Software-based system integration,” June 2019.

[4] A. Cassagne, B. Le Gal, C. Leroux, O. Aumage, D. Barthou, “An Efficient, Portable and Generic Library for Successive Cancellation Decoding of Polar Codes,” The 28th International Workshop on Language and Compilers for Parallel Computing (LCPC 2015), September 2015.

[5] I. A. Ali, U Wasenmuller, and N. Wehnn, “A high throughput architecture for a low complexity soft-output demapping algorithm,” Advances in Radio Science, vlo. 13, pp. 73-80, November 2015.

[6] 5G-RANGE, “Deliverable 3.2 Physical layer of the 5G-RANGE – Part II,” February 2019.

[7] M.Frigo and S. G. Johnaon, “FFTW: Fastest Fourier Transform in the West,” Astrophysics Source Code Library, record ascl:201.015, January 2012.

[8] 3GPP, “System Architecture for the 5G System; Stage 2,” 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects, 3GPP Technical Specification 23.501, version 16.2.0, September 20189.

[9] D. Farinacci, T. Li, S. Hanks, D. Meyer, and P. Traina, “Generic routing encapsulation (gre),” RFC 2784, RFC Editor, March 2000. http://www.rfc- editor.org/rfc/rfc2784.txt.

[10] S. Kent and K. Seo, “Security architecture for the internet protocol,” RFC 4301, RFC Editor, December 2005. http://www.rfc-editor.org/rfc/rfc4301.txt.

[11] Deliverable D5.2. Final version of the Network-level Architecture and Procedures. [Online]. Avilable: http://www.it.uc3m.es/fvalera/5grange/D5.2FinalVersionNetworkArchitecture.pdf

[12] Deliverable D5.3. Final report on Network-level mechanisms implementation. [Online]. Avilable: http://www.it.uc3m.es/fvalera/5grange/D5.3_%20Final_report_on_Network-level_mechanisms.pdf

[13] “5TONIC: an open research and innovation laboratory focusing on 5G technologies,” [Online]. Available: https://www.5tonic.org.

[14] B. Nogales et al., “Automated Deployment of an Internet Protocol Telephony Service on Unmanned Aerial Vehicles Using Network Functions Virtualization,” Journal of Visualized Experiments (153), e60425, doi:10.3791/60425 (2019).

[15] I. Vidal et al., "A Multi-site NFV Testbed for Experimentation with SUAV-based 5G Vertical Services," in IEEE Access, doi: 10.1109/ACCESS.2020.3001985.

