Coordinated Control and Spectrum Management for 5G Heterogeneous Radio Access Networks

Grant Agreement No. : 671639
Call: H2020-ICT-2014-2

Deliverable D2.4
Final Release of the SDK

Version:	1.0
Due date:	28.2.2018
Delivered date:	22.3.2018
Dissemination level:	PU

The project is co-funded by

Authors

Roberto Riggio (CREATE-NET, Editor)

Navid Nikaein (EURECOM)

Laurent Louf (Thales)

Akhila Rao (SICS)

Shahab Shariat (EURECOM)

Chia-Yu Chang (EURECOM)

Fang-Chun Kuo (Travelping)

Coordinator

Dr. Tao Chen

VTT Technical Research Centre of Finland Ltd

Tietotie 3

02150, Espoo

Finland

Email: tao.chen@vtt.fi

Disclaimer

Acknowledgement

This report is funded under the EC H2020 5G-PPP project COHERENT, Grant Agreement No. 671639.

Executive summary

This deliverable accounts for the final public release of the COHERENT Software Development Kit (SDK). The SDK fully supports both Wi-Fi based and LTE-based Radio Access Networks (RANs). The entire code has been released under different open-source licenses.

Two coherent SDKs have been released: the 5G-EmPOWER SDK and the FlexRAN SDK. The former is being promoted through the Twitter channel (https://twitter.com/5gempower) and a Google Analytics account has been created in order to track website access statistics. The later is also being advertised across within the OpenAirInterface users, which has shown an increase of 10 active members per month. Moreover, both platforms are also currently used within few other H2020 projects, namely H2020-SESAME, H2020-5G-ESSENCE, 5G-Picture, SliceNet among the others) as well as the DA2GC (Direct Air to Ground Communications) EIT Digital project.

The full documentation of the COHERENT SDK for both the Central Controller and Coordinator (C3) and Real-Time Controller (RTC) has been made available on respective websites and linked from the COHERENT official web portal. The documentation includes step-by-step instructions on how to setup a fully programmable heterogeneous Wi-Fi and LTE network, starting from open source software components and off-the-shelf devices. Tutorials on how to implement basic control and coordination applications on top of the programmable controller are also available.

Table of contents

List of abbreviations

AP	Access Point
BSR	Buffer Status Report
C3	Central Controller and Coordinator
CMI	Control Module Interface
CQI	Channel Quality Indicator
LVAP	Light Virtual Access Point
LVNF	Light Virtual Network Function
MAC	Multiple Access Control
MCS	Modulation and Coding Schemes
OAI	OpenAirInterface
RAN	Radio Access Network
RRU	Remote Radio Unit
RT	Radio Transceiver
RTC	Real-Time Controller
SDK	Software Development Kit

1. Introduction

The report accounts for the final version of the COHERENT Software Development Kit (SDK) which has been released to the general public on June 12th, 2017 (in conjunction with the EuCNC 2017) and July 11th, 2017 (in conjunction with the ITU workshop[1]). Two SDKs have been released as follows:

1. 5G-EmPOWER SDK: providing full support for Wi-Fi and LTE based RANs (non realtime);

2. FlexRAN SDK: providing full support of LTE based RAN (realtime and non-realtime).

The capabilities of the COHERENT SDK have been presented at EuCNC 2017 where a load-aware SDN-based handover between two CommAgility small cells has been demonstrated. Moreover, RAN sharing/slicing coordinated by a radio resource management (RRM) control app has been demonstrated at the ITU workshop 2017. In addition, new network apps have been developed to support spectrum management, multicasting, mobility management, and traffic steering.

A reference implementation of both the COHERENT Central Controller and Coordinator (C3) and the COHERENT SDK is made available by 5G-EmPOWER[2]. 5G-EmPOWER is a Multi-access Edge Computing Operating System supporting lightweight computing virtualization solutions and heterogeneous radio access technologies. The 5G-EmPOWER platform currently supports both open (srsLTE) and commercial (CommAgility) small cells as well as any Wi-Fi Access Point that can run the OpenWRT operating system. The entire 5G-EmPOWER platform has been released under a permissive APACHE 2.0 Licence for academic use[3]. FlexRAN[4] is a Flexible and Programmable Platform for Software-Defined Radio Access Networks that allows fine-grain monitoring, control and coordination of the underlying RAN following the SDN principles. It currently supports OpenAirInterface[5] and is released under MIT license, subject to be changed in future release.

The 5G-EmPOWER based implementation of the COHERENT SDK is being advertised on Twitter and a Google Analytics account has been created in order to track its usage. Both Twitter and Google Analytics show a steady increase in the number of visitors and users. More specifically, the Twitter account currently has 46 followers while Google Analytics shows an average of 50 unique users per week in the last 30 Months. The FlexRAN implementation of the COHERENT SDK is also being advertised across different communities, namely on-going H2020 projects as well as OpenAirInterface user base, which has shown an increase of 10 active members per month.

The full documentation of the COHERENT SDK has been made available on both the 5G-EmPOWER GitHub wiki[6] and FlexRAN gitlab wiki[7]. The documentation includes step-by-step instructions on how to setup a programmable RAN starting from open source software and off-the-shelf components as well as coding tutorials and developers' resources. We decided not to duplicate such information in this deliverable in that it could become either obsolete or incorrect by the time the deliverable is public. The Wiki represents the single point of entry for the COHERENT SDK.

In Section 2 we introduce the COHERENT Wi-Fi and LTE programming abstractions. Section 3 reports on the proof-of-concept C3 and RTC reference software implementations. Section 4 describes the available tutorials. Finally, Section 5 concludes the deliverable providing a possible development roadmap of the COHERENT SDK beyond the end of the COHERENT project.

[1] http://www.itu.int/en/ITU-T/Workshops-and-Seminars/201707/Pages/Programme.aspx
[2] http://empower.create-net.org/
[3] https://github.com/5g-empower/
[4] https://gitlab.eurecom.fr/flexran
[5] http://www.openairinterface.org/
[6] https://github.com/5g-empower/5g-empower.github.io/wiki
[7] https://gitlab.eurecom.fr/flexran/flexran/wikis/home

2. Programming Abstractions

This section reports on the high-level programming abstractions supported by the COHERENT SDK. The section will first describe the Wi-Fi and the LTE programming abstractions separately. Then the converged RAN slicing abstractions will be introduced. Table 1 and 2 provide an overview of the currently available programming abstractions and their level of support in the COHERENT SDK through the 5G-EmPOWER and the FlexRAN platforms.

Table 1 Abstractions supported by 5G-EmPOWER.

Abstraction	Wi-Fi	LTE	RAT Agnostic	Notes
Channel occupancy/ Physical Resource Blocks Utilization	Yes	Yes	Yes	-
Network Graph • Traffic-matrix (Wi-Fi, LTE) • RSSI Map (Wi-Fi) • Rate control statistics (Wi-Fi) • RSRP/RSRQ Map (LTE)	Yes	Yes	Yes	-
Handover	Yes	Yes	Yes	LTE handover is supported only on CommAgility small cells.
Transmission Policy	Yes	No	No	-
Slicing	Yes	Yes	No	LTE slicing is supported only on open small cells (OAI and srsLTE).

Table 2 Abstractions supported by FlexRAN

Abstraction	LTE	RAT Agnostic	Notes
RLC buffer Map and trigger events PHR Map and trigger events BSR Map and trigger events RB Map and trigger events CQI Map and trigger events HARQ Map and trigger events RSRP/RSRQ Map and trigger events KPI statistics	Yes	No	Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice Per eNodeB, UE, Slice
Network Graph (neo4j)	Yes	Yes	Only network topology
Handover	Yes	No	Adjust X2 HO paramters
Slicing	Yes	No	Slice control Enforcing RRM policy
Spectrum management	Yes	No	Support of phantom-cells

2.1 Wi-Fi Programming Abstractions

In this section, the Wi-Fi programming abstractions are briefly summarized. The reader can refer to the relevant publications listed therein for a more in-depth discussion.

2.1.1 Light Virtual Access Point (LVAP)

The LVAP abstraction provides a high-level interface for wireless clients' state management. The implementation of such an interface handles all the technology-dependent details such as association, authentication, handover, and resource management. A client attempting to join the network will trigger the creation of a per-client virtual access point (the LVAP) which becomes a potential candidate AP for the client to perform an association. Similarly, each AP will host as many LVAPs as the number of wireless clients that are currently under its control. Removing a LVAP from an AP and instantiating it on another AP effectively results in a handover.

The original LVAP abstraction was already presented in [1]. However, during the course of the COHERENT project the abstraction has been extended in order to support:

1. Wireless link-diversity [2, 3]. This allows clients to be attached to more than one Wi-Fi AP in the uplink direction. This also required to extend the LVAP concept in order to allow chaining with LVNFs.
2. Multi-channel [4]. The original LVAP concept requires all Wi-Fi APs to be tuned on the same channel severely limiting spatial re-use. With this extension, the native Wi-Fi Channel Switching Announcement feature is used in order to migrate LVAPs across different channels.
3. The Shared LVAP concept [5, 6, 7, 8]. This extension allows LVAPs to be shared among multiple wireless clients, as opposed to the original mechanism where one LVAP is created for each client. This has been done in order to support multicast applications in that multicast frames must share the same BSSID. The handover API remains unchanged. However, when shared LVAPs are used the actual handover is not seamless and the standard Wi-Fi handover is used.

Besides these major extensions, the LVAP abstraction and its implementation have been significantly overhauled during the COHERENT project. Some minor but relevant extensions are the support for the 802.11n version of the Wi-Fi standard, the implementation of a two-phase commit protocol for the LVAP migration aimed at improving the handover reliability (in particular in multi-channels setups), and in general several data-path performance improvements.

2.1.2 Light Virtual Network Function (LVNF)

The LVNF is a generalization of the LVAP abstraction. However, unlike the LVAP, the LVNF abstraction allows arbitrary packet processing blocks to be exposed to the developer through a high-level declarative interface. LVNFs are instantiated starting from templates called Images.

Each network function corresponds to an Image and consists of a Click script together with some additional information such as the number of input/output ports, and the list of Click handlers[8] exposed by the Image. Handlers are used in order to manipulate the internal state of the LVNF. For example, in the case of a Firewall LVNF, specific handlers shall be defined in order to allow the network operator to add/remove firewall rules. In the ETSI terminology, the handlers essentially provide the interface for the Operator's Element Management Systems (EMS).
The extension has been introduced [3], in order to support arbitrary packet processing on LVAP traffic. So far, its main use has been to support the removal of duplicate traffic when wireless uplink diversity is enabled. However several other applications can also be envisioned [9].

[8] Handlers are access points through which users can interact with elements in a running Click router or with the router as a whole.

2.1.3 Transmission Policy (TXP)

The TXP specifies the range of parameters that radio access network elements can use for their communication with the wireless clients. For example, in the case of a Wi-Fi network such parameters include: the set of valid Modulation and Coding Schemes (MCSs), the RTS/CTS threshold, the ACK policy, and the multicast policy.

Although an early version of the TXP was already introduced in [2], such version has been completely revisited in order to support wireless multicast transmission policies as well as differentiation between 11g and 11n TXPs.

2.1.4 Virtual Port (VP)

LVNFs do not define which kind of traffic they should process. The VP abstraction allows developers to define the logical sequence of LVNFs that a certain portion of the flow-space must traverse. Each Virtual Port, is associated with one, and only one, LVAP or LVNF and is mapped to a physical network interface (e.g. a port of an OpenFlow switch).

Developers need not to know the physical port id and the data path id. Instead, they can perform LVAP/LVNF chaining by using their Virtual Ports. The translation from Virtual Ports into forwarding instructions is performed by the Intent Compiler and by the Path Computation Engine.

The virtual port abstraction has been introduced during the COHERENT project in order to allow LVAP/LVNF chaining [3, 9].

2.2 LTE Programming Abstractions

This section reports on the high-level programming abstractions supported by the COHERENT SDK to allow control apps to be easily implemented for an LTE network.

To control and manage the BS user-plane (UP) actions, the RAN Runtime API is introduced to provide a set of functions that constitute the southbound APIs. These functions allow the control-plane (CP) to interact with the UP in five ways:

1. To get and set configurations like the UL/DL bandwidth of a cell;
2. To request and obtain statistics like transmission queue sizes of UEs and signal-to-interference and noise ratio (SINR) measurements of cells;
3. To issue commands to apply control decisions (e.g., calls for applying MAC scheduling decisions, performing handovers, activating secondary component carriers);
4. To obtain event notifications like UE attachments and random-access attempts; and
5. To perform a dynamic placement of control functions to the master controller or the agent (e.g. centralized scheduling at the master controller or local scheduling at the agent-side).

These API calls create a separation layer between the LTE RAN and runtime system on top and can be invoked either by the C3 or RTC through the FlexRAN control protocol or directly from the runtime if control for some operation has been delegated to it. Table 3 provides a list of some exemplary RAN-specific API calls. Note that the runtime is in charge of retrieving the cell and user related information from the underlying eNodeB such as cell bandwidth, and user Reference Signal Receive Power (RSRP) and Reference Signal Receive Quality (RSRQ) through the API calls, and can trigger events when a state changes, e.g.UE attachment or the radio link failure timer is expired. In addition, such API calls may be related to the NFs, resources, UEs etc. belonging to a particular slice [10]. It can be seen that different types of network applications, ranging from pure network monitoring to network control and optimization, can be implemented using the provided abstractions. We introduce below some applications on top of FlexRAN controller. Specific Control Modules are introduced to get abstracted information and support control applications. These control modules are

designed in an extendable manner and they are completely agnostic to the underlying access infrastructure.

Table 3 LTE-specific API calls

API	Target	Direction	Example	Applications
Configuration (synchronous)	eNodeB, UE, Slice	Centralized/Edge → Agent	UL/DL cell bandwidth, reconfigure Data Radio Bearer (DRB), Measurements	Monitoring, Reconfiguration, Self-Organizing Networks (SON)
Statistic, Measurement, Metering (Asynchronous)	List of eNodeB, UE, Slice	Agent → Centralized/Edge	Channel Quality Indicator (CQI) measurements, SINR measurements, RSRP / RSRQ / UL/DL performance	Monitoring, Optimization, SON
Commands (synchronous)	Agent	Centralized/Edge → Agent	Scheduling decisions, Admission control Handover (HO) initiation	Realtime Control, SON
Event Trigger	Master	Agent → Centralized/Edge	TTI, UE attach/detach, Scheduling request, Slice created/destroyed	Monitoring, Control actions
Control delegation	Agent	Centralized/Edge → Agent	Update DL/UL scheduling, Update HO algorithm	Programmability, Multi-service

2.2.1 Low-level Primitives

These are a set of RAN-specific APIs that can be called to retrieve raw information, such as CQI, PHR on per user, base station, and slice identifier. They represent the instantaneous state of the underlying network and are the basis of the RAN abstractions both at the FlexRAN runtime and controller levels. This means that a hierarchical abstraction can be implemented allowing to pre-process the raw data to reduce the signalling on the control plane toward the high-level controller.

2.2.2 Statistics Manager

This entity manages the statistics of the underlying RAN on per cell, user, and logical channels. Such statistics can be requested in one-shot, periodical, continuous, or event-driven. All the statistics received over time are stored in a radio information base (RIB) at the controller, which can be then reused for further processing and analysis.

2.2.3 RRC measurement

RRC measurements can be obtained from every UE in the network and the trigger of RRC measurement is flexible. A Reconfiguration message can be sent to the UE to trigger the RRC

measurement in three modes: (1) **event driven** where measurements are sent when the requested event occurs, (2) **periodical** allowing UE to periodically transmit the RRC measurement report to the controller via the base station, and (3) **triggering** where base station is being triggered using a predefined threshold defined for some parameters within the measurement report.

2.2.4 Control Delegation and Policy Reconfiguration

The control delegation feature in FlexRAN allows a control app or the high-level controller to delegate the control-decision to the runtime, effectively reprogramming the RAN functions. Each runtime owns a local VNFs repository that can be used to make the local decisions and also dynamically update and change the behaviour of underlying base station without the intervention of the high-level controller. This is done through a set of control modules that interact with the underlying protocol stack through the RAN-specific APIs.
The policy reconfiguration feature, on the other hand, allows to flexibly reconfigure the RAN parameters. These parameters act as a public API that the controller can modify and can either refer to a single value or a sequence of values (e.g., an array). The available parameters depend on the VSF implementation (e.g., two scheduler implementations could have different sets of parameters based on their functionality).

Both control delegation and policy reconfiguration of FlexRAN protocol messages indicate the control modules to be modified using YAML syntax, as shown in the figure below.

```
<Control Module Name>
- <VNF name 1>:
  behaviour: <callback name>
  parameters:
    parameter<i>: val
- <VNF name2>:
  behaviour: <callback name>
  parameters:
    parameter<j>: [val1, val2, val3]
```

2.2.5 High-Level Primitives

These high-level primitives allow to (1) create/update/destroy a virtual BS on the top of physical RAN infrastructure so that a slice owner (operator, service provider) can control their service, and (2) change the behaviour or parameters of the underlying RAN to meet different slice requirements (inter-slice user-control plane processing and resource partitioning).

2.3 Converged Abstractions

In this section, we will present the converged abstractions developed within the COHERENT project. These include the abstractions for RAN slicing and the network graph abstractions.

2.3.1 Converged RAN Slicing Abstractions

In this section, we present a converged abstraction for Radio Access Network (RAN) slices, with focus on LTE and Wi-Fi which are the main wireless technologies used in the COHERENT demonstration. We also give pointers to the work on network slicing inside 3GPP. We focus on 3GPP because it is one of the main standardization targets of the COHERENT project, and its work may

constitute a good framework for the definition of a converged programming abstraction. Here we will concentrate on the abstractions required for the definition and fundamental operation of network slices.

2.3.2 Network slicing in 3GPP

Network slicing is recognized as one of the main concepts for implementing the flexibility required by future 5G networks in terms of service support, while keeping the complexity manageable. Since network slices are a fundamental tool, COHERENT puts effort in demonstrating how network slices in the RAN can be defined and instantiated. In 3GPP context, network slices can cover both the access and core network.

Looking at the activity in 3GPP concerning network slicing, TR 28.801 [11] is a 3GPP technical report which investigates and makes recommendations on management and orchestration of network slices. Aligned with ETSI NFV view, capitalization also on the work about management and orchestration of VNFs,, 3GPP TR 28.801 defines a network slice instance as "a set of network functions and the resources for these network functions which are arranged and configured, forming a complete logical network to meet certain network characteristics". The concept of network slice spans three layers: the service layer, the network slice instance layer and the resource layer. TR 28.801 focuses on issues stemming by the application of the slicing concept to wireless networks, for instance impact on slice management when sharing network slices among multiple parties, relationship between network slices and evolved Self Organizing Network (SON) capabilities, etc. In the following we recall the fundamental phases of the life-cycle of a network slice. For further information, the reader shall refer to TR 28.801 [11].

Figure 1 shows a block diagram description of the life-cycle of a network slice instance as in TR 28.801. Before the instantiation of a network slice, there is a preparation phase in which the network environment is prepared, the network slice template (i.e. the definition of the network slice through its main features) is defined. In the instantiation, configuration and activation phase, the network slice instance is instantiated and configured (meaning that physical/infrastructural resources and network functions are reserved, instantiated, and configured) starting from the information contained in the network slice template; the network slice is also activated, meaning that the traffic is directed through it. During the run-time phase, typical activities of network slice management are done: the network slice is able to report parameters about its status which is supervised by the network slice manager/orchestrator, which can take decisions and actions about the modification of certain network slice parameters for the optimization of some performance criteria. The final decommissioning phase contains all the activities necessary for deleting a network slice, like termination of network functions, reconfigurations for freeing dedicated or common resources, etc.

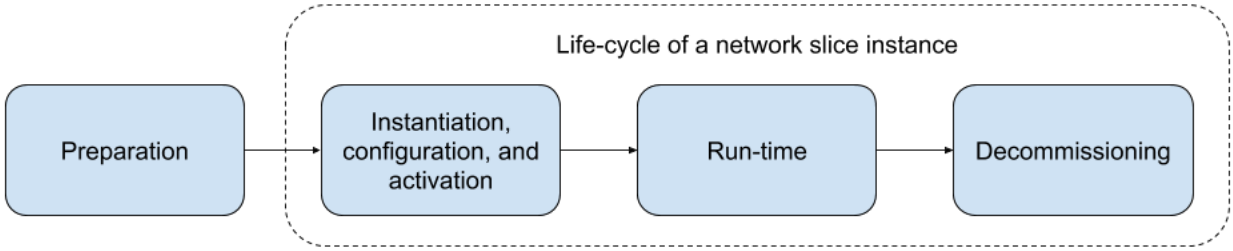

Figure 1 Network Slice Instance Life-cycle as in TR 28.801 [11]

2.3.3 COHERENT slice descriptor

The 3GPP framework is useful for setting the general steps of network slice creation and management. However, we stress here that COHERENT is not trying to comply with one given definition or solution of 3GPP technical work, but rather to take inspiration from it. The following work is hence related to COHERENT needs especially in view of the final demonstration.

In the context of COHERENT, the preparation phase includes, for instance, preparation of the physical infrastructure, basic configuration of eNodeBs and APs, especially of parameters which are not configurable on the fly, not enough flexible, like standard or hardware constraints (which should be exposed by the virtualized RAN) or regulatory constraints for spectrum. Examples of these parameters may be the change from FDD to TDD or vice versa which impact on the definition of the control / reporting cycles and are typically not configurable on the fly, the change of frequency carrier, if the RF is not sufficiently flexible to enable a reconfiguration on the fly, etc..

As recalled in Section 2.3.2, we need a way to define a network slice. In the context of COHERENT we call this tool a network slice descriptor, which is substantially equivalent to the network slice template in 3GPP. For sake of completeness we report here also the definition of a similar concept given by NGMN, the Network Slice Blueprint [10]:

“A complete description of the structure, configuration and the plans/work flows for how to instantiate and control the Network Slice Instance during its life cycle. A Network Slice Blueprint enables the instantiation of a Network Slice, which provides certain network characteristics (e.g. ultra-low latency, ultra-reliability, value-added services for enterprises, etc.). A Network Slice Blueprint refers to required physical and logical resources and/or to Sub-network Blueprint(s).”

The COHERENT slice descriptor is a first step towards the NGMN network slice blueprint, and, through its parameters, defines a converged RAN slice abstraction, since it is used for instantiating slices both for LTE and Wi-Fi. It is defined in the preparation phase of the network slice life-cycle and it is used for defining and partially configuring the network slice. Certain parameters can also describe capabilities for the slice management during run-time.

The parameters of the descriptor are: ***Slice ID***

It is a unique identifier of the RAN slice. In LTE it is mapped to the PLMN ID. In Wi-Fi it is mapped to the tenant ID, i.e. the SSID.

RAN elements ID list

It is a list of RAN nodes belonging to the slice (it defines the spatial location of the slice) with the corresponding supported technologies.

If RAN nodes are LTE eNodeBs, the RAN element ID maps to global Cell IDs, or the eNodeB unique identifiers, like eNodeB ID in 3GPP standard, or target area identity (i.e. all the eNodeBs covered by an MME).

If RAN nodes are Wi-Fi APs, it maps to AP ID (i.e. MAC address of AP).

Available frequency carriers and bands

It is a list of available frequency carriers and bands for each RAN node, plus a default value for initial instantiation.

Services ID list

It is the list of identifiers of the services which the RAN slice can support, with corresponding KPIs.

This list indicates the traffic types that the slice supports and can be used to configure or to choose the type of scheduler that will be used inside the slice.

In case of LTE, the profiles of the traffic types corresponding to their ID are mapped to the set of QCI LTE can support (e.g. Public Safety QCI). In New Radio (i.e. 5G radio in 3GPP terminology), the mechanism is different, since session IDs can be used for mapping traffic to slices.

In case of Wi-Fi: the profiles of the traffic types corresponding to their ID are mapped to possibly preconfigured sets of constraints on Wi-Fi main parameters: back-off, transmission opportunities, MCS selection, DCF parameters, etc.

Slice management policy

A slice management policy is defined as a set of parameters which identify how the slice should behave with respect to other slices.

Currently, the slice management policy may contain part of or all the following parameters:

- Priority,
- Guaranteed data rate,
- Guaranteed resources,
- Max/min delay,
- Packet loss,
- Life time.

Priority: indicates the level of priority among slices. The main idea here is that resource and performance isolation is guaranteed for slices with equal priority. In case of different priorities, isolation may be broken, in the sense that slices with lower priority can see their guaranteed resources be allocated (totally or in part, according to a special repartition agreement) to slices with higher priority. This concept is used in particular for enforcing high priority to slices supporting public safety applications. It is to be used in crisis situation and should not be used in normal situation.

Guaranteed data rate (e.g. in Mb/s): it is the minimum reserved cumulative data rate for the whole slice. The data rate is calculated over correctly received packets, not sent packets. The data rate is an average one, for instance measured in an interval of time, depending on the capability of the infrastructure. The way in which this guaranteed data rate is calculated depends on the technology of the RAN nodes. In run-time, for instance, in LTE, the HARQ ACK/NACK or ARQ reports/ Buffer Status Report (BSR) messages could be used to understand if packets have been correctly received and hence estimate if the target guaranteed data rate can be supported with the currently available resources.

Guaranteed resources (e.g. in Hz): it is the minimum physical band reserved to the slice. For Wi-Fi, this metric maps to minimum percentage of airtime reserved to the whole slice. For LTE, it maps to a specific set or number of physical resource blocks and subframes dedicated to the whole slice. As an example, the exact set of subframes and PRBs can be specified by a 2-dimensional mask (or two 1-D masks, one for time and another for frequency), which is then repeated in time. Notice that here a correct reception is not assumed, but only the availability of spectral resources is guaranteed.

Max/min delay: it is the maximum/minimum guaranteed delay over the RAN slice. For all the technologies it is important to define the exact interface at which the latency is measured (MAC sublayer, network layer, etc). In LTE: the hardware must be able to time stamp the packets, in order to

evaluate the latency, which could not be the case for all devices. Certain settings, like the RLC mode, the number of transmissions for HARQ, load of the system, etc. influence the value of the latency. This parameter could be useful for slices supporting real-time services or critical communication services.

Packet loss: it is the maximum allowed packet loss for the whole slice. In Wi-Fi it is mapped to frame error rate, while in LTE it could be mapped to the MAC PDU error rate. The use of this parameter depends on the possibility of retrieving this information from the RAN nodes.

Life time: is it the time interval during which the policy is applied. The end of the life time of the policy can also be specified upon an event, e.g. until a new policy is applied or the slice is released. In Wi-Fi, this metric can be equivalently mapped to the number of frames of the time interval. In LTE, it can be mapped into a number of frames or subframes.

Notice that the slice management policy could be used to fill certain information about the service profiles supported by the slice. If these profiles are defined independently, a validation check should be done between them and the slice management policy, for trying to spot and to avoid possible inconsistencies.

Number of supported users

It is the typical number of supported users, or a maximum one, depending on the use case.

It is a way to understand how much resources are needed by the slice (e.g. small cell case), especially at the instantiation phase.

Multicast/broadcast support

It states if the slice supports multicast/broadcast services (yes/no), and if yes, the corresponding policy format is dependent on the applied RAT for the moment..

In Wi-Fi, this parameter is mapped to multicast, while in LTE it is mapped to eMBMS. This field is currently not supported in the COHERENT SDK.

2.3.4 Converged Handover Abstractions

SDN-based handover in Wi-Fi networks was already supported at the beginning of the project using the LVAP abstraction. Such abstraction has been extended in order to support multi-channel deployments and multicasting. SDN-based LTE handover has been introduced in the third reporting period and has been demonstrated at EuCNC using CommAgility's small cells. This type of handover requires X2 handover support at the small cell and at the EPC sides.

The actual API calls for the Wi-Fi and the LTE handover are quite similar. For example, performing a handover in a Wi-Fi network is a simple matter of assigning a new value to the *cells* property of an LVAP object:

```
lvap.cells = cells
```

Here *cells* is a list of Wi-Fi cells available in the network. Notice how one AP can have one or more cells. Cells can be filtered using the Network graph API:

```
lvap.cells = self.cells().filterByAddr(lvap.addr).sortByRssi().first()
```

This call will fetch all the cells available in the network, will filter the ones that have measurements for a specific LVAP, and will sort the list by decreasing RSSI. Then the first entry in the list is taken. The resulting list can be also empty, in which case no handover is performed.

Similarly, performing a handover in a LTE network is a simple matter of assigning a new value to the *cells* property of an UE object:

```
ue.cells = cells
```

Here the variable *cells* is a list of LTE cells available in the network. Notice how one eNodeB can have multiple sectors and each sector can have multiple cells (i.e. carriers).

If a device is both a Wi-Fi station and an LTE UE it is possible for the programmers to navigate from one to another, for example the following lines fetch a UE object instance from an LVAP instance and vice-versa:

```
ue = lvap.ue
lvap = ue.lvap
```

The mapping between the two instances is transparently kept by the controller which monitors all the relevant S1AP procedures.

Notice that when a mapping between an LTE UE and a Wi-Fi station is found, all the Wi-Fi traffic is tunnelled trough the S1 interface to the LTE EPC.

2.3.5 Converged Network Graph Abstractions

Network Graph database can not only provide a new set of graph-based low-complexity primitives (e.g. merge) to operate on the network information whether it is raw or processed, but also provides visualization of logical network topology with different level of granularity and geographical areas. It can also be used for big data applications for large scale scenarios.

2.3.6 Link performance abstractions

This section presents the network state representation using link performance abstractions for Wi-Fi and LTE RAN that are performed by the network information function (NIF) [12]. The implementation of the Wi-Fi and LTE NIFs are described in more detail in [13] can be found online[9].

A Wi-Fi/LTE RAN consists of association links from users to corresponding associated APs/eNodeBs and candidate links from users to candidate APs/eNodeBs. Instances of the NIF operate locally on each AP/eNodeB as well as on the controller. This distributed approach to aggregation of network state information introduces low overhead compared to an approach of sending raw metrics directly to a centralized unit for processing.

The performance of an association link is represented using a RAT agnostic abstraction: measured MAC layer throughput. A probabilistic abstraction of this throughput captures the underlying variation in its performance. The probabilistic abstraction is represented as an empirical cumulative distribution function (ECDF). The probability of throughput being less than a required threshold is reported as the QoS violation probability with the aim of maintaining this below a defined tolerance level. The estimated performance on a candidate link is represented using a RAT agnostic abstraction:

[9] https://github.com/nigsics/aquamet

attainable MAC layer throughput. The estimate of attainable throughput is obtained using an estimation model that is specific to each RAT [13]. A probabilistic abstraction of this estimate is captured in the attainable throughput ECDF, and the probability of attainable throughput being less than a required threshold. This is reported as an estimate of QoS violation probability on candidate links.

This abstraction over candidate links is used as input to RTC/C3 for decision making (eg. user-AP association). Mobility management using such an agnostic abstraction of the current and potential performance combines the process of performing associations based on physical layer parameters (eg. RSSI/RSRP/RSRQ) and load balancing by presenting a single abstraction which captures the performance as seen by the user application. Our approach of applying probabilistic attainable throughput to do interface selection and make mobility management decisions has been evaluated in detail in our technical paper that has been submitted and is under peer-review [14]. In performed evaluations, our approach reduces the number of handovers between interfaces by 35 times along with reducing performance violations by 20% and 116%, compared to the state-of-the-art and naïve baselines (still under use in e.g. enterprise Wi-Fi) respectively.

Figures 2 and 3 show block diagrams of the input metrics used and the output abstractions obtained in a Wi-Fi and LTE NIF. The NIFs take as input RAT specific (Wi-Fi or LTE) and RAN specific (eNodeBs or APs with different capabilities and current status) metrics along with user/service defined requirements and provides a unified abstraction that represents current and potential link performance as an agnostic, comparable, and probabilistic abstraction.

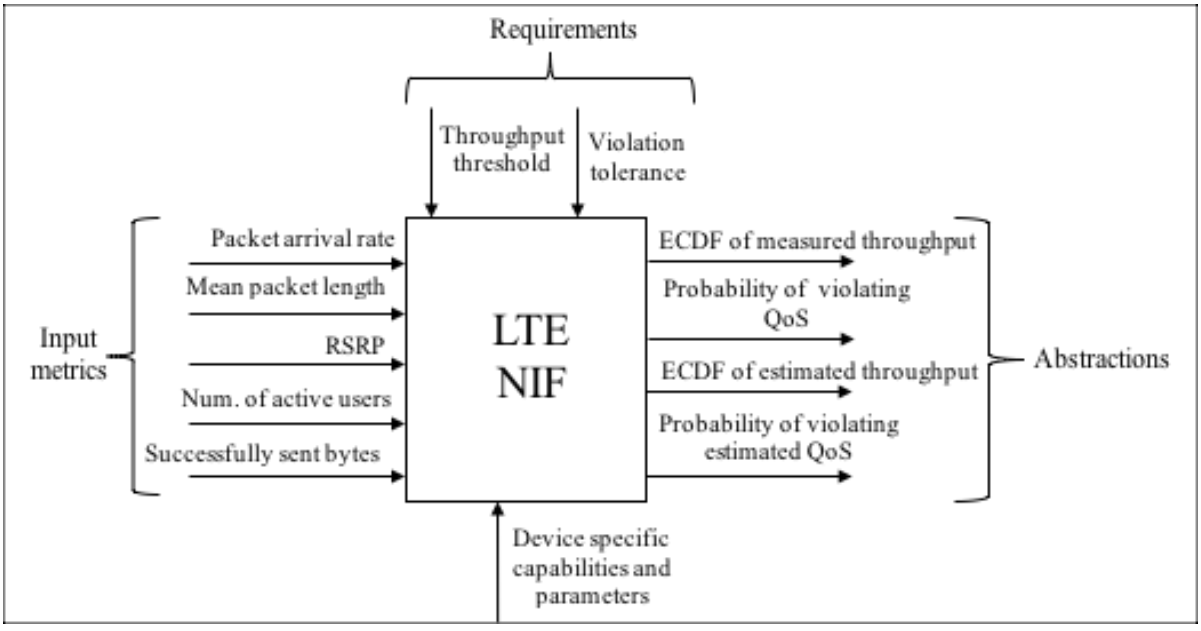

Figure 2 Block Diagram representation of LTE NIF

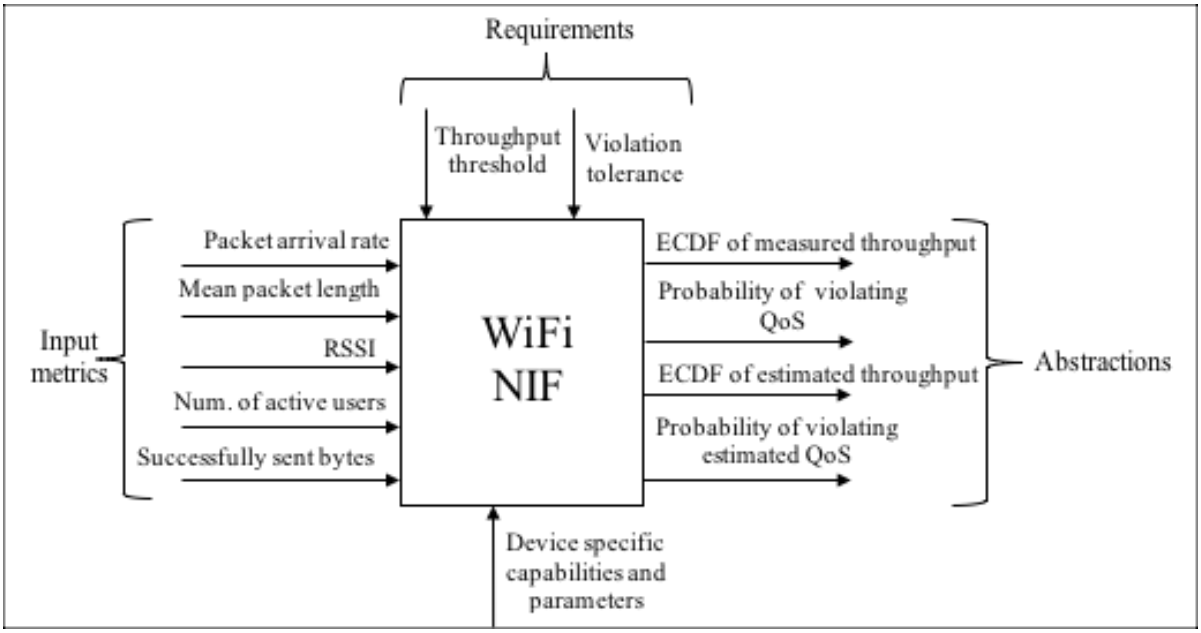

Figure 3 Block Diagram representation of Wi-Fi NIF

2.3.6.1 Summary of input metrics and output abstractions

Tables 4 and 5 summarize the input metrics and output abstractions respectively. F_λ is the ECDF of attainable throughput on a candidate link. $P(\lambda > \rho_{th}) > tolerance$ represents the condition for QoS. Similarly, we have F_ρ and $P(\rho \leq \rho_{th}) \leq tolerance$ for throughput on an associated link.

Table 4 Summary of input metrics

Metric	**RAT** measured	**Location of measurement**
Packet arrival rate	Wi-Fi/LTE	AP/eNodeB/backhaul-switch
Mean packet length	Wi-Fi/LTE	AP/eNodeB/backhaul-switch
RSSI	Wi-Fi	Client/UE
RSRP	LTE	Client/UE
Num. of active users	Wi-Fi/LTE	AP/eNodeB
Successfully sent bytes	Wi-Fi/LTE	AP/eNodeB

Table 5 Summary of RAT agnostic abstractions

Abstraction	RAT supported
F_λ	Wi-Fi/LTE
$P(\lambda > \rho_{th}) > tolerance$	Wi-Fi/LTE
F_ρ	Wi-Fi/LTE
$P(\rho \leq \rho_{th}) \leq tolerance$	Wi-Fi/LTE

3. Software Architecture

3.1 Overview

In this section, we will first briefly recall the general COHERENT architecture and terminology. Then we will introduce the 5G-EmPOWER platform and the OpenAirInterface RTC which are two reference implementations of the COHERENT C3 and RTC components respectively.

The SDK described in this deliverable provides the APIs used for implementing control and coordination tasks on top of the C3 and of the RTC. An SDK is typically defined as a set of software development tools that facilitate the creation of software components. In the specific case of COHERENT, the COHERENT SDK provides programmers with a high-level interface to their Software Defined Mobile Networks as opposed to the very low-level interfaces exposed by single network elements (i.e. the southbound interface).

We refer the reader to COHERENT D2.2 [12] for the system architecture, the complete description of the COHERENT semantic model for Wi-Fi and LTE networks, and the associated functional components and interfaces. The COHERENT architecture is depicted in Figure 4.

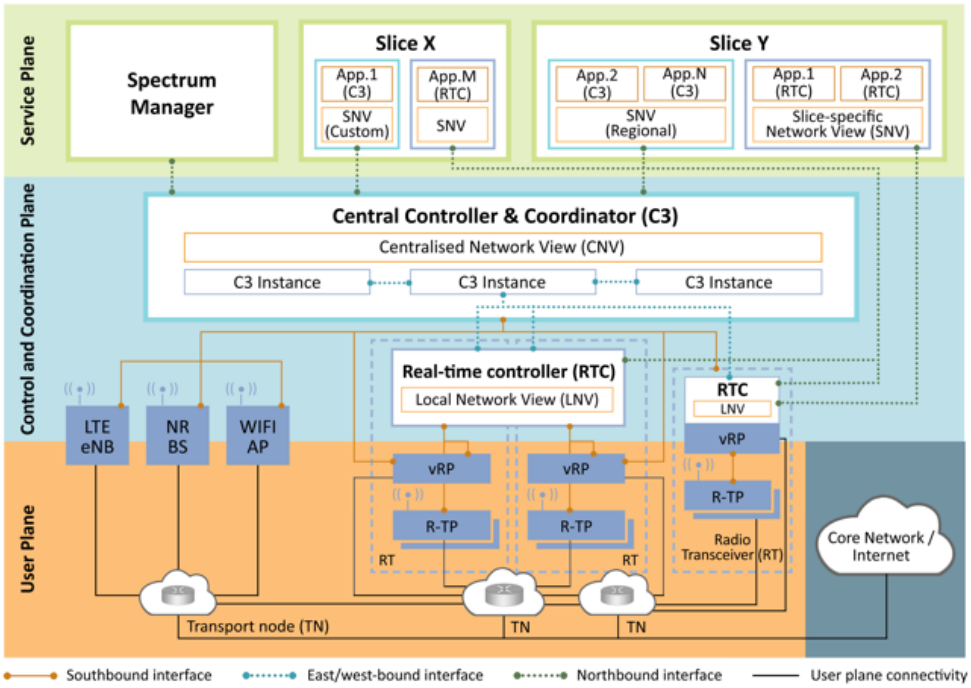

Figure 4 The COHERENT Architecture.

The following components are defined in [12]. For clarity reasons they are recapped in this report:

- **Radio Transmission Point (R-TP):** R-TP is a radio access point implementing full or partial RAN node functions while rest of functions are offloaded to and handled by the vRP. An R-TP may include control plane functions.

- **Virtual Radio Processing (vRP):** vRP is a computing platform allowing for centralised processing of full or partial RAN node functions (including the data plane and the control plane) offloaded from one R-TP or multiple R-TPs. A vRP may include control plane functions.

- **Radio Transceiver (RT):** RT is a logical radio access entity with full RAN node functions, which is the flexible combination of R-TP, vRP and RTC functions. A set of RTs is forming a

radio access network which is coordinated and controlled by C3. Some examples of RTs include LTE eNodeB in cellular networks or Wi-Fi APs in WLANs. An RT could be composed by one vRP (virtual device) and one or more R-TPs (physical devices). For example, in the Cloud-RAN architecture the R-TP coincides with the RRH, while the vRP coincides with the BBU pool. In some particular cases like D2D, RT could be an UE acting as a relay node.

- **Transport Node (TN):** TN is the entity located between RTs and core network. A set of TNs is forming a backhaul/fronthaul network whose data plane can be configured by the C3. A network switch is an example of Transport Node.

- **Real-Time Controller (RTC):** A logical entity in charge of local or region-wide control, targeting at real-time control operations, e.g., MAC scheduling. It has local network view. It could run directly on one RT or on a virtualised platform and receives monitoring information gathered from one RT or multiple RTs. It can delegate control functionality to the RTC on RTs. RTC communicates with an RTC agent(s) on one or multiple RTs.

- **Central Controller and Coordinator (C3):** A logical centralised entity in charge of logical centralised network control and coordination among entities in RAN based on centralised network view. C3 could be implemented with distributed physical control instances sharing network information with each other. Sharing network information among C3 instance creates the logically centralised network.

- **Network Slice:** A network slice is defined as a collection of specific network services and RAT configurations, which are aggregated together for some particular use cases or business applications. A network slice can span all domains of the network: software programs running on cloud nodes, specific configurations of the transport network, a dedicated radio access configuration, as well as settings of the 5G devices. Different network slices contain different network applications and configuration settings.

3.2 The 5G-EmPOWER Platform

3.2.1 Overview

5G-EmPOWER is a Multi-access Edge Computing Operating System (MEC-OS) supporting lightweight computing virtualization and heterogeneous radio access technologies. A high-level view of the 5G-EmPOWER system architecture is depicted in Figure 5. The architecture is logically divided into three layers. The bottom layer consists of the physical and virtualised resources composing the data-plane. The second layer runs 5G-EmPOWER OS which controls the physical and virtual resources in the data-plane. The third layer deals with the actual slices.

5G-EmPOWER is a reference implementation of the C3, but it also targets Lightweight NFV Management and Orchestration and Multi-access Edge Computing. The details of these features are out of the scope of this document and are thus omitted.

The entire software stack has been released under a permissive APACHE 2.0 for academic use and is available at the official 5G-EmPOWER website: http://empower.create-net.org/. Documentation and tutorials are available in the associated GitHub project: https://github.com/5g-empower/5g-empower.github.io/wiki. Commercial use of the platform is regulated by the APACHE 2.0 License.

The communication between the 5G-EmPOWER MEC-OS and the data-plane device happens over a persistent TCP connection. The definition of the protocol and of the various messages is too verbose for this report and has been made available online under the same licence as the rest of the platform: https://github.com/5g-empower/5g-empower.github.io/wiki/EmPOWER-Protocol.

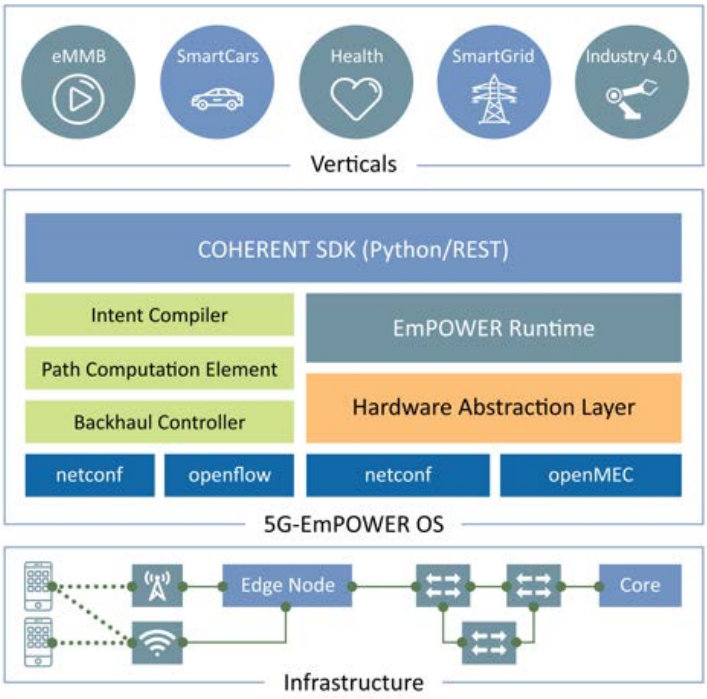

Figure 5 The 5G-EmPOWER System Architecture.

3.2.2 Multi-tenancy and RAN slicing

The 5G-EmPOWER MEC-OS supports multi-tenancy in the form of RAN slicing. Users can use either the web-based dashboard or a REST interface in order to perform CRUD (Create, Retrieve, Update, and Delete) operations on a network slice. The technical details about how slicing is performed will be provided by WP5 deliverable D5.2 [15]. In this section, we will report on the slicing interface exposed by the 5G-EmPOWER OS.

```
version: 1.0

WTPs:
  - name: WTP1
    description: human readable description
    hwaddr: 00:00:00:00:00:01
    location:
      latitude: 46.0702531
      longitude: 11.1216386
    interfaces:
      - interface_id: 1
        hwaddr: aa:bb:cc:dd:ee:ff
        available_airtime: 50
        channel: 11
        band: HT20
- name: WTP2
    description: human readable description
    hwaddr: 00:00:00:00:00:02
    location:
      latitude: 46.0702531
      longitude: 11.1216386
    interfaces:
      - interface_id: 1
        hwaddr: aa:bb:cc:dd:ee:ff
        available_airtime: 30
        channel: 36
        band: HT20
```

Figure 6 Wi-Fi Resource Manifest.

The 5G-EmPOWER slicing API follows the slicing abstractions presented in Section 2.3. In this section, we briefly report on the YAML descriptors used to instantiate new Wi-Fi / LTE network slices.

Figure 6 reports the resource manifest advertised by the 5G-EmPOWER OS for a network composed of two Wi-Fi APs. Notice how, the acronym WTP used in the manifest means Wireless Transmission Point and is the term used by 5G-EmPOWER to refer to standard Wi-Fi APs.

Each of the WTP advertised in this manifest comes with the following set of properties:

1. The physical address of the device.
2. A human readable description of the device.
3. The geographical position of the device.
4. The list of wireless interfaces available on the device together with their operating channel, band, and physical hardware address.
5. The percentage of downlink airtime that is currently not allocated to any slice.

Figure 7 reports the resource manifest advertised by the 5G-EmPOWER OS for a network composed of one LTE eNodeB. Notice how the acronym VBS used in the manifest means Virtual Base Station and is the term used by 5G-EmPOWER to refer to standard LTE eNodeB.

```
version: 1.0

VBSes:
  - name: VBS1
    description: human readable description
    hwaddr: 00:00:00:00:20:E1
    location:
      latitude: 46.0702531
      longitude: 11.1216386
    cells:
      - cell_id: 10
        prbs: 6
        available_prbs: 3
        frequency: 2660 # MHZ
      - cell_id: 20
        prbs: 25
        available_prbs: 20
        frequency: 2660 # MHZ
```

Figure 7 LTE Resource Manifest.

Each of the VBS advertised in this manifest comes with the following set of properties:

1. The physical address of the device.
2. A human readable description of the device.
3. The geographical position of the device.
4. The list of cells available on the device together with their operating frequency, and total number of physical resource blocks.
5. The number of physical resource blocks that are currently not allocated to any slice.

The above resource manifests essentially advertise the resources that are available in the physical infrastructure. The slice owner can then either use the web-based management dashboard exposed by the 5G-EmPOWER OS or interface directly with the 5G-EmPOWER REST. The latter option is particularly useful if the administrative commands are originating from a network service orchestration platform.

Figure 8 reports a slice creation descriptor built using the information contained in the previous manifest. As it can be noticed the descriptor includes the network SSID and PLMNID. The descriptor also specifies the use of three RAN devices: one LTE eNodeB and two Wi-Fi APs. For each device, a certain amount of resources are requested. The 5G-EmPOWER OS checks if the request made by the user can be accepted according to the available resources.

```
version: 1.0

Name: TrentoOfficeNetwork
Description: Human-readable description

PLMNID: 22293
SSID: TrentoOffice

VBSes:
  - name: VBS1
    requested_resources:
      - cell_id: 10
        resource_blocks: 6

WTPs:
  - name: WTP1
    requested_resources:
      - interface_id: 1
        airtime: 10
  - name: WTP2
    requested_resources:
      - interface_id: 1
        airtime: 10
```

Figure 8 Slice creation descriptor.

3.2.3 Supported platforms

The 5G-EmPOWER platform supports four types of data-plane devices:

1. **Wireless Termination Points (WTPs).** Each WTP consists of two components: one OpenVSwitch instance implementing the backhaul data; and one Agent implementing the 802.11 data–path. The Agent is implemented using the Click Modular Router. Click is a framework for writing multi–purpose packet processing engines and is being used to implement just the WTPs/Wireless Clients frame exchange, while all the decision logic is implemented at the EmPOWER Runtime. Communications between the Agent and the Runtime take place over a persistent TCP connection. Any Wi-Fi capable of running the OpenWRT OS can become an EmPOWER WTP. Instruction on how to configure a WTP can be found online[10].
2. **Click Packet Processors (CPPs**). These nodes essentially combine an OpenFlow switch with a general purpose x86 CPU. As the name suggests, CPPs leverage on multiple instances of the Click Modular Router in order to implement packet processing. Each CPP includes an OpenVSwitch instance, one or more Click instance, and one Agent. The latter is in charge of monitoring the status of each Click instance as well as of handling the requests coming from the controller. Any x86 machine capable of running Linux can become a CPP. Instructions on how to configure a CPP can be found online[11].
3. **Virtual Base Stations (VBSes).** These are essentially LTE eNodeBs. The 5G-EmPOWER platform currently supports both open and commercial LTE small cells. In order to communicate with the EmPOWER Runtime, the small cells must implement the EmPOWER Agent. Such Agent, which is part of the EmPOWER platform, has a modular architecture allowing it to be adapted to different platforms (both open source and commercial). Instruction on how to configure a VBS can be found online[12].
4. **Transport Nodes (TN).** These are the entities located between RTs and core network. A set of TNs is forming a backhaul/fronthaul network whose data plane can be configured by the 5G-EmPOWER. An OpenFlow switch is an example of Transport Node.

[10] https://github.com/5g-empower/5g-empower.github.io/wiki/Setting-up-the-WTP
[11] https://github.com/5g-empower/5g-empower.github.io/wiki/Setting-up-the-CPP
[12] https://github.com/5g-empower/5g-empower.github.io/wiki/Setting-up-the-VBS

3.3 FlexRAN and OpenAirInterface Platforms

3.3.1 Overview

FlexRAN platform is made up of two main components: the FlexRAN service and control plane and FlexRAN application plane. The FlexRAN service and control planes follow a hierarchical design and is composed of a RTC that is connected to a number of underlying RAN runtimes, one for each RAN module (e.g. one for monolithic 4G eNodeB, or multiple for a disaggregated 4G and 5G). The control and data plane separation is provided by RAN runtime environment which acts as an abstraction layer with RAN module on one side and RTC and control apps on the other side. The FlexRAN protocol facilitates the communication between the realtime controller and the RAN runtime embedded in runtime environment. RAN control applications can be developed both on top of the RAN runtime and RTC SDK allowing to monitor, control and coordinate the state of RAN infrastructure. Such applications could vary from soft realtime applications including monitoring that obtain statistics reporting to more sophisticated distributed applications that modify the state of the RAN in realtime (e.g. MAC scheduler). All the produced RAN data and APIs are open to be consumed by 3rd parties. The communication between the FlexRAN controller (RTC) and the data-plane device happens over the FlexRAN protocol. The protocols and messages are defined based on google protobuf message serialization[13]. A RAN API[14] which acts as the Southbound APIs interface for the agent separates the control plane from the data plane of the RAN through technology agnostic/abstracted design.

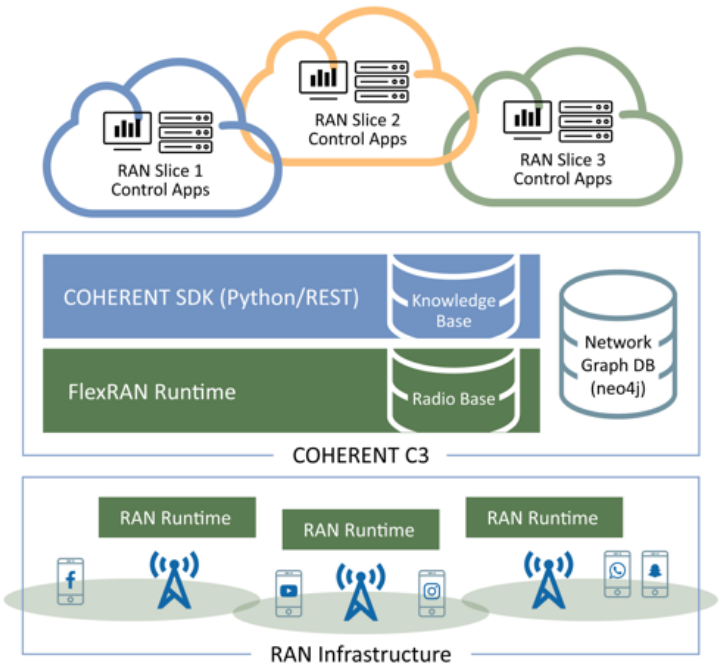

Figure 9 FlexRAN OAI software Architecure

3.3.2 Multi-tenancy and RAN slicing

FlexRAN implements a set of slicing APIs following the slicing abstractions presented in Section 2.3. In this section, we briefly report on the YAML descriptors used to instantiate new virtual LTE slices. Currently, FlexRAN only supports different level of isolation and sharing of radio resources without providing functional isolations among different slices.

[13] https://gitlab.eurecom.fr/oai/openairinterface5g/tree/feature-68-enb-agent/openair2/ENB_APP/MESSAGES/V2
[14] https://gitlab.eurecom.fr/oai/openairinterface5g/blob/feature-68-enb-agent/openair2/ENB_APP/flexran_agent_ran_api.h

```
version: 1.0

enb_slices :
  - name: VBS1
    cell_id: val
    service_types: [“name1”, "name2", "name3"]
  - name: VBS2
    cell_id: val
    service_types: ["name2", "name3"]

service_policy :
    name1 :
        UL :
            requested_vrbg: val
            requested_rate: val
            requested_latency: val
            requested_priority: val
            requested_isolation:  val
        DL :
            requested_vrbg: val
            requested_rate: val
            requested_latency: val
            requested_priority: val
            requested_isolation:  val
```

Figure 10 LTE Slice and Resource template.

```
version: 1.0

BS1:
   node_function: "eNodeB_3GPP"
   eutra_band: val1
   downlink_frequency: val
   uplink_frequency_offset: val
   N_RB_DL: val

rrc:
 - trigger_measurment : val
  - x2_ho:
      parameters:
          ttt_ms                    : val
          hys                       : val
          ofn                       : val
          ocn                       : val
          ofs                       : val
          ocs                       : val
          off                       : val
          filter_coeff_rsrp         : val
          filter_coeff_rsrq         : val

mac:
- dl_scheduler:
    behaviour: <callback>
    parameters:
      n_active_slices: val
      slice_maxmcs: [val1, val2, val3, val4]
      slice_percentage: [val1, val2, val3, val4]
      slice_rb_map: [val1, val2, val3, val4]

- ul_scheduler:
    behaviour: <callback>
    parameters:
      n_active_slices_uplink: 1
      slice_maxmcs_uplink: [val1, val2, val3, val4]
      slice_percentage_uplink: [val1, val2, val3, val4]
      slice_rb_map_uplink: [val1, val2, val3, val4]
```

Figure 11 BS and Control Module Slice Descriptor.

It can be seen from the template that there are two cells, named VBS1 and VBS2, that offer different types and number of services (e.g. VBS1 has 3 slices of different types). Each service is described by

a set of policies that will be applied when a slice is created or updated, both in terms of radio resources, performance, priority, and isolation. Such a template will be then mapped to a BS and Slice descriptor that defines physical BS parameters, and the slice-specific resource allocation policy as shown in the yaml file below. Note that each segment of the descriptor is handled by the corresponding control module.

In addition, the FlexRAN SDK supports spectrum sharing across different BS by maintaining different information and policies in form of knowledge base retrieved from different sources, which are briefly described below:

- **General policies:** describing the available frequencies in each country with max transmission power among other parameters.
- **Sensing data:** describing the neighbouring operators and their respective operational frequencies.
- **Operator policies:** describing the available licensed frequency information (power mask, sensitivity) as well as the leasing conditions (such as min lease time, and cost).
- **LSA policies:** which is similar to operator policies but it is applied in the LSA bands.
- **Rules:** defines the criteria and cost functions to determine the best candidate frequencies to be shared.

As a result, the candidate frequency will be pushed to the underlying BS in form of a yaml file similar to Figure 11.

3.3.3 Supported platforms

The FlexRAN platform supports the following types of data-plane devices:

1. **Virtual RAN OAI (VRO)** The runtime system can provide multi virtual points (radio access) for same physical RAN which are then communicate with controller through FlexRAN protocol. These points can be emulated on top of RAN by runtime system to provide the virtual access here.

2. **Virtual Core OAI (VCO)** The runtime system in the core network (MME, S/P-GW, HSS) can communicate with an OVS switch controller. These can provide also the virtualization on top of every runtime system.

4. Tutorials

A set of tutorials have been prepared in order to guide new developers through the process of creating new applications using the COHERENT SDK. In this section we will first report on the available tutorials for the C3 and then on the tutorials for the RTC.

4.1 C3 Tutorials (Wi-Fi/LTE)

The COHERENT SDK allows to write new coordination applications as Python modules running on top of the 5G-EmPOWER MEC-OS. Network Apps can be loaded either at bootstrap or using the Controller REST interface without affecting the network operations. A command line utility for loading Network Apps is also available.

A tutorial for writing a simple “Hello, World!” network app can be found at the following address: https://github.com/5g-empower/5g-empower.github.io/wiki/Python-API-documentation. In this section, we will briefly report on some of the tutorials available on the Wi-Fi providing the relevant pointers. This is done due to the fact that tutorials are updated on a daily basis and any information provided in this report would be inevitably outdated after a few weeks.

4.1.1 Mobility Manager (Wi-Fi)

This tutorial illustrates how to implement a mobility management application for Enterprise Wi-Fi networks. The application takes into account the channel quality as exposed by the network graph in order to implement handover decisions. The tutorial is available at the following address: https://github.com/5g-empower/5g-empower.github.io/wiki/Tutorial-Mobility-Manager.

4.1.2 Mobility Manager (LTE)

This tutorial illustrates how to implement a mobility management application for LTE networks. Notice that this is the codebase used for the final project demonstration reported in WP2. The tutorial shows how to implement LTE handover decisions starting from the global network view exposed by the controller. The tutorial is available at the following address: https://github.com/5g-empower/5g-empower.github.io/wiki/Tutorial-Mobility-Manager-LTE. This tutorial requires a small cell which supports X2 handover. At the time of writing only the CMA’s small cells satisfy this feature.

4.1.3 Traffic Steering (Wi-Fi/LTE)

This tutorial illustrates how to implement a traffic steering application to redirect traffic from the Wi-Fi access network to the LTE network and vice-versa. The decision is taken starting from the overall channel condition as well as from the actual traffic demands originating from the clients. The tutorial is available at the following address: https://github.com/5g-empower/5g-empower.github.io/wiki/Tutorial-Mobility-Manager-LTE-Wi-Fi.

4.1.4 Multicasting (Wi-Fi)

This tutorial illustrates how to implement a joint multicast MCS selection and multicast group management scheme using the COHERENT SDK. The tutorial is available at the following address: https://github.com/5g-empower/5g-empower.github.io/wiki/Tutorial-Multicast.

4.2 RTC Tutorials (LTE)

This section includes short tutorials on the real-time control applications developed using the COHERENT SDK.

The control applications require and run on top of OAI[15] and the FlexRAN[16] platform described in Section 3.3. The RTC is configured by default to provide support for real-time applications as long as a Linux kernel >=3.14 is available. It provides also a REST northbound API for controlling the developed applications (for example the RRM app) using Pistache library.

We provided Store[17], an SDK environment for network application developers to write their application completely independent of and decoupled from FlexRAN controller with an SDK separation layer. This lets application developers to write their applications on their preferred language using these network apps. This is also a very good fit for RAT agnostic scenarios where the application developer does not need to care about underlying RAN technology and can write his/her applications using just the RAN statistics, data to create mathematical models and analysis.

SDK provides two modes of operation *test* and *sdk* mode. In test mode, input data are taken from pre-existent JSON files. In the repository, the interested reader/programmer can find sample data on eNodeBs' configuration and capabilities, UEs' configuration and capabilities and logical channels' configuration and capabilities. Moreover, the JSON files contain various statistics of parameters belonging to different layers, e.g. for MAC layer CQI and BSR statistics are available. These files can be modified during the execution of the apps to simulate varying situations, like UEs competing for the channel resource or channel conditions varying over time. In *SDK* mode, the application dynamically gets the same information from the FlexRAN platform which reports the parameters directly from a running cellular system on OAI. The applications are explained in following subsections. These applications are all written in Python.

4.2.1 RRM application

In addition to original 3GPP RRM design vision and also to the good definition from [16] "RRM refers to a group of mechanisms that are collectively responsible for efficiently utilizing RRUs within a Radio Access Technology (RAT) to provide services with an acceptable level of QoS", the RRM app SDK is enriched with the addition of radio resource management for multiple slices. A slice descriptor (Figure 10) is used to define multiple slices with different policies for proper radio resource management. The slice descriptor allows, for instance, to define a number of active slices per eNodeB, a percentage of resource blocks to be allocated to each slice, a special bitmap for resource allocation enforcing a predefined set of MCSes for scheduling decisions inside a given slice. A scheduler (remote or local) can be passed also to runtime to change the policy of scheduling on the fly both for downlink and uplink. The RRM app can get all of the PDCP layer statistics. It can also trigger RRC Measurements of the underlying RAN to get the handover measurement parameters. In the future, this application can also be extended to additional RRM functions like Power Control, Admission Control and Congestion Control and Dynamic Frequency/Channel Allocation, Interference Management for a specific slice, to create even a chain of RRM subcomponents/functions for that specific slice.

This application can be useful to a mobile network operator or any service provider (OTTs) for specifying RRM QoS/QoE policies. Different high level QoS/QoE components can be defined.

This application can be used also for public safety applications, since it is possible to set slice priority values that guarantee resource pre-emption for critical services in case of heavily loaded systems.

[15] https://gitlab.eurecom.fr/oai/openairinterface5g/tree/develop
[16] https://gitlab.eurecom.fr/flexran/flexran-rtc/tree/develop-uplink
[17] https://gitlab.eurecom.fr/mosaic5g/store

This means that an operator can implement its own RRM functions to enforce its own policy to the underlying RAN network.

4.2.2 Spectrum Management Application

Spectrum Management Application deals about management of RRM spectrum aspects and its related policies. A mobile network operator could use this application to define mathematical policies (by means of weights and cost functions) in order to monitor and enforce a specific service level agreement (SLA). This application also includes the price modelling as well.

A formula of weight calculations that is based on cost is the following:

$$Total\_Weight = \frac{\sum_{i=1}^{N}(1 - param\_name_i / max\_param\_name_i))}{Sum\ Cost}$$

where the *Total Weight* is the sum of cost ratio calculated for every MVNO corresponding to every cost metric. *Param_name* is the name of the metric to use. In descriptors, these metrics are defined as *price*, *bandwidth*, *frequency_high*, *frequency_low*, *min_lease_time*, *duration_lease* and the *Sum cost* will be the total sum of all costs for that specific MVNO.

Different yaml file templates are provided for different policy types. These input files are *operator_policy*, *lsa_policy*, *general_policy*, *enb_assign*, *sensing_data*, *rules*.

Operator policy contains a list of offers. In simple words, some operator can share a spectrum with a specific rules defined by operator. Some parameters related to *operator_policy* template are 1) name of operator 2) minimum frequency 3) maximum frequency 4) busy, to indicate if the bandwidth is occupied by another operator 5) idle 6) power mask (for interference channels) 7) minimum lease time 8) maximum lease count 9) maximum time to live (time to leave this band when we detect) 10) price 11) sensing sensitivity (in dBm).

Parameters related to *lsa_policy* are similar to operator policy but without breakdown option “busy” and “idle”.

Parameters related to *general_policy* can be very high policy like specific to a country or government, which are provided by law, 1) minimum frequency 2) maximum frequency 3) frame type 4) maximum tx power.

Parameters related to *enb_assign* deal with grouping of eNodeB infrastructure: 1) group MVNO; 2) Cell id.

Parameters related to *sensing data* are related to detection of different eNodeB which are not connected to controller and can be seen even as neighbouring eNodeB.

Finally parameters related to *rules* are: 1) MVNO group; 2) pattern, specific pattern model for pricing and bandwidth usage for specific MVNO; 3) cost, a table in which we can set a weight of each parameter of offer (in line to default pattern) - when we use our own pattern, this values will be ignored; 4) operator_preference, preference for the spectrum owner, may prefer to have specific preference in some situations; 5) criteria, a table with limits for specific parameters, for example in parameter “price” we can set minimum, or maximum price for spectrum (in the same way we set limits for other parameters); 6) frequency_preference, to specify preferred bands (for example if we can transmit on 3.5 and 2.6 we would prefer 2.6).

4.2.3 Monitoring App

The AQuaMet monitoring application is implemented in Python on top of FlexRAN SDK[18]. The objective of AQuaMet is to observe the network state in a RAT-agnostic probabilistic framework and serve it as input to a control function in a heterogeneous network that requires the network to provide probabilistic guarantees on user QoS. Admission control, mobility management and radio interface selection are use-case control applications for AQuaMet monitoring.

Throughput at the user is the considered QoS metric. Throughput is observed in sliding windows on multi-RAT links in the network. A histogram of measured throughput captures the variability on traffic bearing links in the network in a RAT-agnostic manner. The QoS requirement at the user is given by the equation below. Here, ρ is the measured throughput, ρ_{th} is the required level of throughput for a user, and α is the tolerance level for throughput being less than the required level. When this condition is violated the user creates a trigger indicating that the QoS requirement has been violated.

$$P(\rho \leq \rho_{th}) \leq \alpha$$

When the serving link is not able to provide the service guarantee required by a user, candidate links need to be evaluated as target links for the handover. Attainable throughput is the throughput that is estimated to be achieved on a candidate link from the same or different RAT. This is obtained from base metrics of channel quality and network load along with estimation models specific to each RAT considered. Details of the estimation method and models are provided in Section 4.4.1 of deliverable D3.2 [13]. Attainable throughput not only evaluates the throughput expected to be achieved on a candidate link, but also the impact on throughput of users sharing that radio network. The base metrics required for the estimation of attainable throughput are also maintained in sliding windows to capture their variability under the dynamic network conditions. When a trigger is received from a user, a histogram of attainable throughput is evaluated on all candidate links to find a target base station (general for any RAT) that is estimated to satisfy the probabilistic QoS requirement of the triggered user and the users the target base station already serves. This condition is given by the equation below. Here, λ is the attainable throughput.

$$P(\lambda \leq \rho_{th}) \leq \alpha$$

This way association decisions can also be made with the objective of satisfying probabilistic guarantees on QoS. This application uses the statistics manager application to obtain the required base metrics. This RAT-agnostic AQuaMet monitoring application is implemented on top of the FlexRAN SDK which is also agnostic to the underlying network access technology.

[18] https://github.com/nigsics/aquamet

5. Conclusions

In this report, we accounted for the release of the final version of the COHERENT SDK. The SDK builds upon the 5G-EmPOWER platform developed by CREATE-NET and on the FlexRAN platform developed by EURECOM and is made available under different open-source licenses.

The platform is currently witnessing an increasing popularity as demonstrated by its adoption by other H2020 projects and by institutions not directly linked with the COHERENT project.

The substantial enhancements for the platform beyond the end of COHERENT are planned:

1. To add support for more LTE and Wi-Fi commercial platforms;
2. To allow users to deploy the complete COHERENT software stack using modern virtualization technologies, i.e. Containers;
3. To enlarge the user base of the COHERENT SDK by organizing tutorials, hands-on sessions, and summer schools.

References

[1] Lalith Suresh, Julius Schulz-Zander, Ruben Merz, Anja Feldmann, and Teresa Vazao. 2012. Towards Programmable Enterprise WLANS with Odin. In Proc. Of ACM HotSDN 2012, New York, NY, USA.

[2] Roberto Riggio, Mahesh K. Marina, Julius Schulz–Zander, Tinku Rasheed, Slawomir Kuklinski, "Programming Abstractions for Software–Defined Wireless Networks", in Network and Service Management, IEEE Transactions on, vol.12, no.2, pp.146-162, June 2015

[3] Roberto Riggio, Imen Grida Ben Yahia, Steven Latré, Tinku Rasheed, "Scylla: A Language for Virtual Network Functions Orchestration in Enterprise WLANs", in Proc. of IEEE NOMS 2016, Istabul, Turkey

[4] Estefania Coronado Calero; Roberto Riggio; José Villalón; Antonio Garrido" Wi–Balance: Channel–Aware User Association in Software–Defined Wi–Fi Networks", in Proc. Of IEEE IM 2018, Taipei, Taiwan.

[5] Estefania Coronado, Roberto Riggio, Jose Villalo and Antonio Garrido, "Programming Abstractions for Wireless Multicasting in Software–Defined Enterprise WLANs", in Proc. of IEEE IM 2017, Lisbon, Portugal

[6] Estefania Coronado Calero; Roberto Riggio; José Villalón; Antonio Garrido Efficient Real–time Contents Distribution for Multiple Multicast Groups in SDN–based WLANs", IEEE TNSM (to appear).

[7] Estefania Coronado Calero; Roberto Riggio; José Villalón; Antonio Garrido "SDN@Play: A Multicast Rate Adaptation Mechanism for IEEE 802.11 WLANs", in Proc. of IEEE CCNC 2017, Las Vegas, NV, USA.

[8] Estefania Coronado Calero; Roberto Riggio; José Villalón; Antonio Garrido "Joint Mobility Management and Multicast Rate Adaptation in Software–Defined Enterprise WLANs", IEEE TNSM (to appear).

[9] Julius Schulz-Zanderm, Stefan Schmid, James Kempf, Roberto Riggio, and Anja Feldmann "LegoFi the Wi-Fi Building Blocks! The Case for a Modular Wi-Fi Architecture", in Proc. of IEEE MobiArch 2016, New York, NY, USA

[10] NGMN, "Description of Network Slicing Concept", v1.0.8, 14th September 2016.

[11] TR 28.801, "Study for Management and Orchestration of Network Slicing for Next Generation Network (Rel 15)", v1.2.0, June 2017.

[12] D2.2, System Architecture and Abstractions for Mobile Networks, COHERENT project deliverable, 2016.

[13] D3.2, Final Report on Physical and MAC Layer Modelling and Abstraction, COHERENT project deliverable, 2017.

[14] A. Rao, R. Steinert. "Probabilistic Multi-RAT Performance Abstractions". Submitted for review in the third IFIP/IEEE International Workshop on Management of 5G Networks (5GMan 2018).

[15] D5.2, Final Specification and Implementation of the Algorithms for Programmable Radio Access Networks, COHERENT project deliverable, 2017.

[16] L. Wu and K. Sandrasegaran, "A Study on Radio Access Technology Selection Algorithms, Chapter 2, SpringerBriefs in Electrical and Computer Engineering