Alma Mater Studiorum · Università di Bologna

SCHOOL OF ENGINEERING AND ARCHITECTURE

Department of Electrical, Electronic and Information Engineering
DEI
"Guglielmo Marconi"

Second Cycle Degree
in
Telecommunications Engineering

Software Defined Radio for NB-IoT

Master Thesis in Radio Systems

Candidate:
Michele Paffetti

Supervisor:
Chiar.mo Prof. Ing.
Roberto Verdone
Co-Supervisor:
Prof. Ing. Raymond Knopp

Academic Year 2016/2017
Session II

Alla mia Famiglia

Abstract

The next generation of mobile radio systems is expected to providing wireless connectivity for a wide range of new applications and services involving not only people but also machines and objects. Within few years, billions of low-cost and low-complexity devices and sensors will be connected to the Internet, forming a converged ecosystem called Internet of Things (IoT). As a result, in 2016, 3GPP standardizes NB-IoT, the new narrowband radio technology developed for the IoT market. Massive connectivity, reduced UE complexity, coverage extension and deployment flexibility are the targets for this new radio interface, which also ensures harmonious coexistence with current GSM, GPRS and LTE systems. In parallel, the rise of open-source software combined with Software Defined Radio (SDR) solutions has completely changed radio systems engineering in the late years. These platforms provide testbed for experimental analysis and prototype development enabling researchers to test, validate and assess the performance of new technologies for wireless networks. This thesis focuses on developing the NB-IoT's protocol stack on the EURECOM's open-source software platform OpenAirInterface (OAI). First part of this work aims to implement NB-IoT's Radio Resource Control functionalities on OAI. After an introduction to the platform architecture, a new RRC layer code structure and related interfaces are defined, along with a new approach for Signalling Radio Bearers management. A deep analysis on System Information scheduling is conducted and a subframe-based transmission scheme is then proposed. The last part of this thesis addresses the implementation of a multi-vendor platform interface based on Small Cell Forum's Functional Application Platform Interface (FAPI) standard. A configurable and dynamically loadable Interface Module (IF-Module) is designed between OAI's MAC and PHY layers. Primitives and related code structures are presented as well as corresponding Data and Configuration's procedures. Finally, the convergence of both NB-IoT and FAPI requirements lead to re-design PHY layer mechanisms for which a downlink transmission scheme is proposed.
This work constitutes a precious starting point to ensure OpenAirInterface interoperability, flexibility and IoT capabilities to face the next generation of mobile radio technology.

This Master thesis project is conducted in collaboration with the Communication Systems Department of EURECOM, a France research institute based in Sophia Antipolis.

Acknowledgement

Vorrei sfruttare queste righe per fare dei ringraziamenti che credo riassumano un po' quello che sono oggi.

Innanzitutto, se sono arrivato fin qui, è perchè mi è stato concesso da Dio, in cui credo. Pertanto, oggi come ogni giorno lo ringrazio per donarmi questa vita piena di amore, bellezza, fatica, errori e felicità.

Un sentito ringraziamento va al Professor Roberto Verdone, relatore di questa tesi. Ringrazio il professore per aver riposto in me una grande fiducia, dandomi l'opportunità di svolgere questo progetto in un contesto di altissimo livello nel campo delle telecomunicazioni. Il tirocinio ad EURECOM mi ha reso più consapevole dei miei mezzi e sarà un esperienza che porterò sempre con me.
Ringrazio dal profondo del cuore la mia famiglia, a cui dedico questa tesi. Ringrazio i miei genitori, per i sacrifici che hanno fatto e fanno ogni giorno per me; mi hanno dato l'opportunità di arrivare dove sono oggi senza farmi mancare niente. Rigrazio i miei fratelli, Chiara e Gabriele, che riempiono le mie giornate tristi con sorrisi e amore; spero di poter essere sempre per loro un esempio da seguire. Ringrazio poi la mia ragazza Flora per tutte quei giorni, quelle settimane, quei mesi che ha sopportato la mia lontananza. Sono convinto che nessun'altra avrebbe fatto lo stesso in tutti questi anni e questo dimostra quanto lei tenga a me. Spero di poter ripagare con il mio amore tutto quello che mi ha dato e continua a darmi ogni giorno.
Ringrazio gli amici di una vita, quelli della palla a spicchi e quelli dell'Agazzi Cleb che seppur in questi ultimi anni ci vediamo veramente poco, sono sempre lì, pronti ad accogliermi per organizzare qualche cena o intraprendere insieme una nuova avventura. Un ringraziamento speciale va al mio amico «Relly», ormai ho perso il conto da quanti anni ci conosciamo. Lui c'è sempre quando «gna fare du sgassate» e in questo ultimo periodo ha pazientemente aspettato il mio ritorno in pista (gasse a manetta ora). Ringrazio poi il nostro (mio e di Albe) amico di merende PowerCorte, che in questi due anni ha fatto passi da gigante grazie alla mia assidua presenza; mi ha insegnato che a volte nella vita bisogna essere un po' più «scialli». Infine, ma non per importanza, ringrazio i compagni di viaggio Magjar; seppur per molti di noi le strade si sono divise, so che ognuno potrà sempre contare sull'aiuto degli altri. Questa nostra amicizia è per me qualcosa di veramente importante che porterò sempre nel cuore.

I would hereby like to express my gratitude and appreciation to Prof. Raymond Knopp, my supervisor at EURECOM, for having followed me during my Internship and put me in contact with the OSA reality. Moreover, I express my gratitude to anyone in EURECOM who helped and made me possible to successfully complete my Internship: Prof. Navid Nikaein, for his precious advices on approach OAI's RRC code; Younes Khadraoui, who was the first person to welcome me on my arrival; Cedric Roux, who endured me every time I asked him help on programming; Shahab Shariat, for our talks on the world of research. Last, but not least, I would like to thank my Internship companion Nick Ho, in the three months we worked side by side he taught me much and we had a lot of fun together.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

Long Term Evolution (LTE) is the standard name indicating the mobile system technology developed by the 3rd Generation Partnership Project (3GPP) and released for the first time in 2008. LTE evolved from the previous quite old 3GPP system known as Universal Mobile Telecommunication System (UMTS) with the aim of increasing cellular network capabilities towards achieving the fourth generation (4G) of mobile systems. Among all the early targets and requirements, the most essential aspects for LTE have been a strong demand for higher data rates, in addition to top Quality of Service (QoS), low latency, scalability and cost reduction thanks to considerably low design complexity. Nowadays, LTE is the most prevalent and successful mobile communication technology worldwide. It has reached a maturity level that not only addresses enhanced functionalities but also the support for new use cases [1], becoming an essential piece for the next generation of mobile networks, i.e 5G.

The fifth-generation (5G) of mobile radio systems will be driven by an increase in mobile traffic demand and it will provide wireless connectivity for a wide range of new applications and services involving ultra-reliability, Enhanced Mobile Broadband (eMBB) and Massive Machine Type Communication (mMTC). The latter, will play a fundamental role in the Internet of Things (IoT) scenario in which billions of low-cost and low-complexity devices and sensors will be connected to the Internet. For this reason, 3GPP has taken evolutionary steps on both the network and device side of its cellular radio technology to meet the new connectivity requirements of the emerging massive IoT segment. As a result, a new family of LPWA technologies has been standardized including Extended Coverage GSM (EC-GSM), LTE-M (or eMTC) and Narrowband Internet of Things (NB-IoT). The latter, is the new 3GPP solution addressing ultra-low-end IoT applications and constitutes the radio interface on which this Master thesis focuses.

NB-IoT shows cost, deployment flexibility and deep coverage advantages over other IoT solutions by scaling down to extreme device simplicity and allowing harmonious coexistence with current cellular technologies. These performance objectives have led to a new system design for both Access and Core network parts. In particular, the radio protocol architecture has been primarily revolutionized at Physical (PHY) and Radio Resource Control (RRC) layer, with minor changes also for Packet Data Convergence Protocol (PDCP), Radio Link Control (RLC) and Medium Access Control (MAC).

Parallel to this, the rise of modern wireless transmissions based on Software Defined Radio (SDR) solutions has completely changed radio systems engineering. Many radio components are implemented in software and can be reconfigured on-the-fly, thus allowing a single piece of hardware for multi-standard, multi-band and multi-functional wireless systems [12]. Such paradigm change has converged the cellular systems from slow-moving proprietary and expensive hardware platforms towards open-source ecosystems with several benefits in terms of cost, time and flexibility. Moreover, the telecommunication industry has been transformed in

recent years by the emergence of open-source mobile communication software. A number of companies, universities and research institutes have started developing open-source projects combining open software with SDR solutions, to make trial environments to test the newest generation of mobile communication architecture and equipment.

This is the case of OpenAirInterface (OAI), an open-source platform developed by the EURECOM research institute as an open and flexible framework for implementing standard-compliant experimental mobile radio systems. OAI is considered one of the first open-source SDR implementation of LTE, spanning the full protocol stack both in E-UTRAN and EPC. It can be used to build and customize an eNodeB base station and core network on commodity PC and connect commercial UEs to test and monitor the network and mobile devices in real time. EURECOM believes that OpenAirInterface can be instrumental for the development of key 5G technologies [13]. In particular, its latest research directions are addressing IoT requirements for future design of cellular networks by bringing NB-IoT and LTE-M functionalities into OpenAirInterface. This is the context in which this Master project fits, by contributing to the development of NB-IoT protocol stack on the OAI eNodeB.

In addition, a key target for future mobile radio networks is to achieve interoperability. A multi-vendor, virtualized and heterogeneous network is the vision that mobile operators have of their next generation deployments [14]. In this scenario, the standardization of a common architecture in which parts are interchangeable and ensure the latest hardware and software innovation with minimum barriers, is fundamental. The goal of interoperability has been central to the Small Cell Forum's work which has standardized multi-vendor Application Platform Interfaces (APIs) such as the Functional API (FAPI) and Network Functional API (nFAPI) to ensure an unified framework which allows small cell from different vendors to work together seamlessly. This approach has not escaped to the research community in EURECOM which has quickly started to bring FAPI/nFAPI functionalities into OpenAirInterface. Also in this case, open software solutions like CISCO's "open-nFAPI" [15] have proven to be a useful tool for supporting the MAC/PHY split foreseen by the Small Cell Forum's standards.

1.1 Objectives

This Master thesis project is conducted in collaboration with the Communication Systems Department of EURECOM, a France research institute based in Sophia Antipolis.

The main focus of this work is to develop the Narrowband Internet of Things (NB-IoT) protocol stack over the open-source platform OpenAirInterface (OAI). In particular, the major contributions are the design and implementation of Radio Resource Control (RRC) layer, along with the adoption of a multi-vendor platform interface compatible with Small Cell Forum's FAPI standard.

NB-IoT is the new 3GPP radio technology developed for the Internet of Things (IoT), targeted for massive connectivity, reduced UE complexity, coverage extension and deployment flexibility. To the best of our knowledge, no free open-source project like OpenAirInterface is addressing the implementation of a NB-IoT compatible eNodeB base station. Bringing IoT functionalities into this open platform not only provides a testbed for prototype validation and performance evaluation but accelerates innovation for the next generation of mobile radio systems (5G).

The starting point of this thesis is the investigation and analysis of OAI software, providing insights on procedures and mechanisms of RRC layer code along with some knowledges on PHY. The top-level approach adopted on describing layers processes, primitives classification and data structures, constitutes a valuable tool to address the lack of currently update documentation on OpenAirInterface's protocol.
To meet NB-IoT requirements, RRC functionalities and related procedures at eNodeB side of

OpenAirInterface are re-designed. The original layer's working principles are not modified but we account for different message types, reduced functionalities and underlying Information Elements (IEs) introduced by the new technology. This leads to outline a new RRC code structure and re-define primitives for configuration and data exchange with other protocol entities.
For this project, NB-IoT bearer management represents one challenging task. The 3GPP standard introduces a new Signalling Radio Bearer (SRB), called SRB1bis, established implicitly with the legacy SRB1 but utilized before security activation, i.e. bypassing PDCP. The developed approach exploits Logical Channel Identity (LCID) to differentiate between SRB1bis and SRB1 allowing related PDCP layer configuration. Moreover, key point in this case is the received UE message after which SRB1bis translates into SRB1 without requiring additional configurations from eNodeB.
In addition, the "NPDCCH less" approach introduced by NB-IoT for System Information (SI) scheduling revolutionizes the radio resource mapping. Some considerations on required upgrades for OAI's MAC scheduler are given, justifying the need of new frame checking algorithms and new input parameters. Furthermore, a possible scheduling approach is proposed which optimizes the number of subframes available for those radio frames dedicated to SI transmissions.

Finally, the implementation of a multi-vendor platform interface based on FAPI standard is addressed. The flexibility enabled by this solution is achieved by the definition of a suitable split point between OAI's MAC and PHY layer along with related messages and procedures. To achieve this, a configurable and dynamically loadable Interface Module is developed, which provides FAPI-like primitives for both data and configurations. However, the introduction of such Module compliant with NB-IoT specifications requires a rethink of PHY layer mechanisms of which a new design is provided with specific details on downlink transmission.

1.2 Outline

The thesis work is structured as follow:

Chapter 2 provides some backgrounds on different topics. It introduces the IoT concept and related 3GPP cellular solutions. It describes the Software Defined Radio paradigm and gives insight into OpenAirInterface, the reference open-source platform of this thesis. For the latter, both software and hardware perspective is given. A brief survey on ASN.1 description language is also available, useful for clearer understanding of addressed topics. Finally, the chapter provides an in-depth analysis of NB-IoT radio technology, focusing on major changes introduced by the standard at PHY and RRC layer.

Chapter 3 describes the multi-vendor platform interfaces FAPI and nFAPI released by Small Cell Forum, focusing on downlink FAPI procedures related to NB-IoT.

The next two chapters describe the given open-source system and related improvements. Chapter 4 addresses the initial characterization, providing a general overview of OpenAirInterface software architecture with details on RRC and PHY layer code. Chapter 5 reports all the improvements and followed approaches to develop the NB-IoT protocol stack, along with the introduction of a FAPI-compliant interface between PHY and MAC.

Finally, Chapter 6 summarizes major considerations on the developed work and proposes future works.

Chapter 2

Background

This Master thesis is written assuming the reader has knowledge of the field of mobile radio systems. In particular, an adequate background on the 3GPP Long Term Evolution (LTE) standard is required to fully understand topics addressed in this and the following chapters. If that is not the case, [16] and [17] are recommended as valid support to fill knowledge gaps on LTE technology during the reading, in particular about Radio Resource Control (RRC) layer and some Physical (PHY) layer procedures.

2.1 Towards IoT

2.1.1 Introduction

Each generation of mobile communication, from the first (1G) introduced in the 1980s, to the forth (4G) launched in recent years, has had a significant impact on the way people and businesses operate. The fifth-generation (5G) of mobile radio systems is expected to extend its capabilities far beyond those of previous ones, providing wireless connectivity for a wide range of new applications and services involving not only people but also machines, objects and more general devices. Although the requirements for 5G are still being finalized both in the ITU and 3GPP, there is a preliminary agreement regarding the three main use cases the technology must support [1] as illustrated in Figure 2.1:

- Enhanced Mobile Broadband (eMBB): used as a general term referring to the extended support of conventional MBB technology through improved data rates, capacity and coverage.
- Ultra-Reliable Low Latency Communication (URLLC): an emerging sector of critical applications such as infrastructure protection, remote surgery or intelligent transport systems (ITSs) in which low latency and reliability together with zero mobility interruption gab are of highest importance.
- Massive Machine Type Communication (mMTC): referring to the envisioned scenario with billions of low-cost and low-complexity connected devices and sensors.

In particular, mMTC is expected to play a fundamental role within future 5G systems enabling a complete implementation of the Internet of Things (IoT) concept. The term "IoT" refers to information networks where objects ("things") from diverse environments are mutually connected into a single large-scale ecosystem based on Internet Protocol (IP) [18]. The development of IoT will allow smart machines and devices to interact with other objects, things, environment, infrastructure and humans through the Internet. In this "networked society", every person and every industry will be empowered to reach their full potential [19] exploiting machine-to-machine (M2M) and machine-to-person communications on a massive

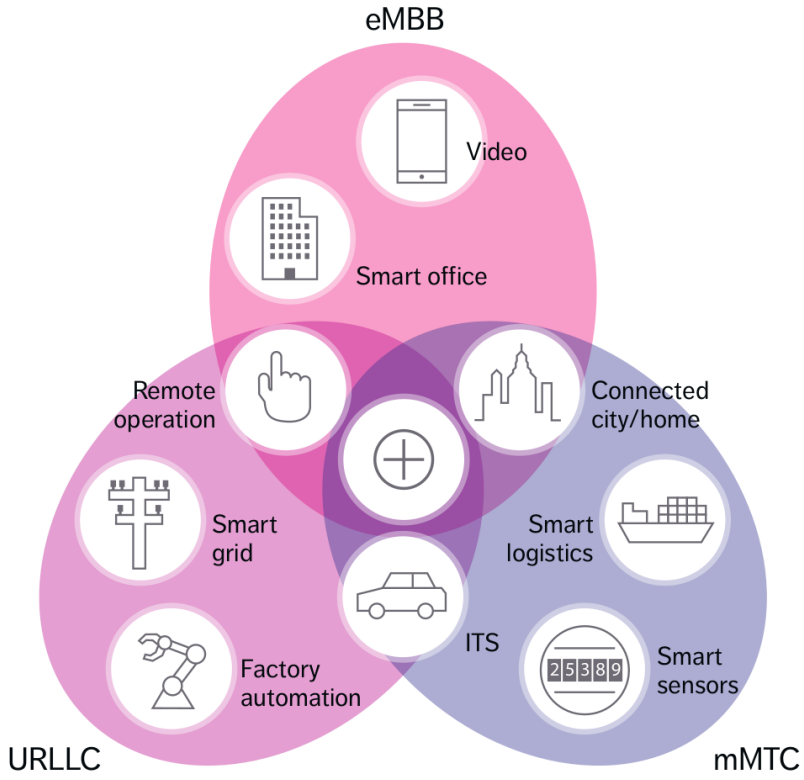

Figure 2.1: The three main 5G use cases and examples of associated applications [1].

scale. In the incoming years, it is expected that IoT will shift from vision to reality. According to [20], of the 29 billion connected devices predicted to exist by 2020, 18 billion will be IoT devices. The potential applications for the Internet of Things run into the millions, with a huge variety of requirements regarding cost, battery life, coverage, connectivity and performance (throughput and capacity). Some devices, such as temperature sensors, will be required to only send a few messages per day while others, for example, may need to transmit a video stream to guide surgeons during emergency operations. Differences on requirements, applications, nature and complexity of IoT equipments will directly affect the type of access technologies needed. According to [19], a large share of IoT devices will be served by short-range radio Local Area Network (LAN) technologies such as Wi-fi, Bluetooth and Zigbee, operating in unlicensed spectrum. However, these technologies are designed with limited QoS and security requirements being mainly applicable for home or indoor environment. On the other hand, the most attractive solution for IoT connectivity will be enabled by wide-area coverage technologies for which currently there are two alternatives:

- Unlicensed Low-Power Wide Area Network (LPWAN): proprietary radio technology solutions designed to address long-range and low-power communications for MTC applications. Most prominent standard in this sector are LoRaWAN [21] and Sigfox [22].
- Cellular technologies: 3GPP solutions like GSM, LTE and future 5G that are being rapidly evolved with new functionalities and new radio access technologies. These, are specifically tailored for emerging LPWA applications and able to addressing everything from Massive to Critical IoT use cases.

2.1.2 LTE Evolution towards IoT: eMTC and NB-IoT

Future 5G networks will not be based on one specific radio-access technology. Rather, they will be realized by a portfolio of access and connectivity solutions [23]. As the most prevalent mobile communication technology worldwide, LTE constitutes an essential piece of the final

5G puzzle [1]. As such, its latest releases are intended to meet requirements and address the relevant use cases expected in the 5G era. In particular, in Rel.12 and mainly in Rel.13, 3GPP has taken evolutionary steps on both the network and device side to meet the new connectivity requirements of the emerging Massive IoT segment [19]. Key improvement areas address reduction of device cost, enhanced UE coverage, longer battery life and support for massive number of IoT connections. As a result, in the late years a new family of LPWA technologies have been standardized including Extended Coverage GSM (EC-GSM), LTE-M (or eMTC) and Narrowband Internet of Things (NB-IoT).

EC-GSM, is the result of 3GPP initiative to further improve GSM, boosting coverage up to 20dB with respect to GPRS on the 900 MHz band [19]. However, the flat and flexible architecture of LTE as well as its more efficient signalling, higher performance and lower operating costs have moved the attention to other two solutions: LTE-M and NB-IoT.
LTE-M and NB-IoT have been introduced with 3GPP LTE Release 13 as a suit of complementary narrowband LTE IoT technologies both optimized for low complexity, low power and high density device deployment. They offer similar improvements with regard to coverage enhancement, battery life, signalling efficiency and scalability, but address slightly different demands in terms of flexibility and performance [1].
LTE-M or eMTC, is the pure LTE solution brought into 3GPP Rel.13 as a first step in addressing mMTC capability over LTE. It continues the optimization already done in Rel. 12 (UE Cat-0) with the introduction of the new UE category: Cat-M1. It brings new power-saving functionalities and mechanisms for substantially reduced device cost and extend coverage. Moreover, it can deliver up to 1 Mbps of throughput utilizing just 1.4 MHz of bandwidth.
In addition, 3GPP Rel.13 introduces Narrowband-IoT (NB-IoT), a new radio technology addressing ultra-low-end IoT applications. It is tightly connected with LTE providing higher deployment flexibility with cost advantages over LTE-M. It scales down to extreme simplicity on devices with self-contained carrier of 200 KHz bandwidth and ultra-low-throughput (about 200 kbps). NB-IoT constitutes the main focus of this Master thesis project and a deeper analysis of its standard features will be presented in section 2.5.

Figure 2.2 shows a comparison of 3GPP solutions starting from the first LTE Rel. 8 (Cat-1), up to the latest access technologies for addressing IoT market.

	LTE-Cat 1	LTE-Cat 0	LTE-Cat M 1	NB-IoT	EC-GSM-IoT
Deployment	In-band LTE	In-Band LTE	In-band LTE	In-Band LTE Guard-Band LTE Standalone	In-band GSM
Downlink	OFDMA [15 kHz]	OFDMA [15 KHz]	OFDMA [15 kHz]	OFDMA [15kHz]	TDMA/FDMA
Uplink	SC-FDMA [15 kHz]	SC-FDMA [15 kHz]	SC-FDMA [15 kHz]	Single Tone [15/3.75 kHz]	TDMA/FDMA
Peak Rate	DL:10 Mbps UL: 5 Mbps	DL: 1 Mbps UL: 1 Mbps	DL: 1 Mbps UL: 1 Mbps	UL: 250 kbps DL: 20 kbps	UL: 70/240 kbps DL: 70/240 kbps
UE receiver BW	20 MHz	20 MHz	1.4 MHz	200 kHz	200 kHz
Duplex Mode	Full-Duplex	Half-Duplex	Half-Duplex	Half-Duplex	Half-Duplex
Max UE transmit power	23 dBm	23 dBm	23 or 20 dBm	23 dBm	33 dBm or 23 dBm
Power saving	PSM, eDRX	PSM, eDRX	PSM, eDRX	PSM, eDRX	PSM, eDRX

Higher throughput, lower latency, full mobility

Figure 2.2: 3GPP technology evolution for the Internet of Things [2].

Beyond 3GPP Rel.13, there is a rich roadmap of LTE IoT technology inventions that will deliver many further enhancements to meet tomorrow's massive IoT connectivity needs [24]. The completion of Release 14 will bring new capabilities such as single-cell multicast, enhanced reference signals for positioning applications, larger channel bandwidth for LTE-M (up to 5 MHz) and support for lower NB-IoT power classes [1].

2.2 Software Defined Radio

As every aspect of our lives becomes connected via smart devices, new digital business opportunities arise and evolve, as well communication technologies require further interoperable flexible systems. In the future, smart technology software will be adapted to meet changing needs and in this scenario a fundamental role will be played by the concept of Software Defined Radio (SDR).
The term SDR refers to a radio communication system where components that have been typically implemented in hardware are instead replaced by means of software-defined functions on a general-purpose computer or embedded system such as a System-on-Chip (SoC). The development of SDR systems overcomes the disadvantages of radio hardware that are generally expensive and heterogeneous, allowing to develop programmable platforms by means of new class of technologies such as Field Programmable Radio Frequency (FPRF). In an ideal software-defined radio system, the entire radio function runs on a General-Purpose (GPP) or Digital Signal (DSP) Processor to achieve a full programmability of the Radio Frequency (RF) part, and only requires analog-to-digital and digital-to-analog conversions, power amplifiers and antennas [25]. As a result, users can enable the radio to support different wireless communication protocols by simply configuring the waveform software without necessarily throwing away the hardware design. Figure 2.3 illustrates the ideal SDR block diagram.

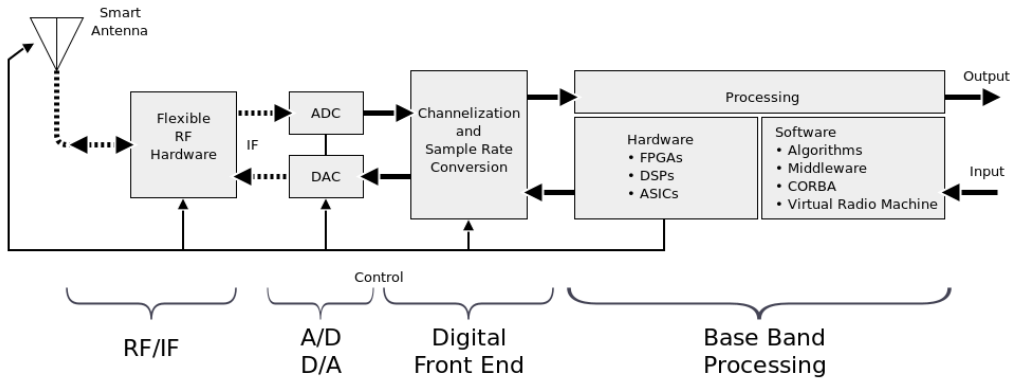

Figure 2.3: Block Diagram of an ideal Software Defined Radio system [3].

In recent years, SDR has grown to the point where it can be incorporated into different systems such as Wi-Fi routers, TVs, laptops and wireless sensor technologies. Not least, the tremendous flexibility and cost effectiveness of SDR made it a key implementation technology for the infrastructure component of cellular systems moving from ossified and expensive hardware platforms towards open-source software architectures. As SDRs have become more commonplace, many companies and organizations have developed hardware front-ends, such as Universal Software Radio Peripheral (USRP), and software toolkit like the open-source GNU Radio, to help the software-defined radios development. This make realistic to researchers and communities of hackers outside the industry, to access high-end computing power to prototype their own experimental code developing many of the most relevant mobile radio standards as open-source solutions. For instance, three of the most relevant are:

- OpenBTS [26]: a Unix application that uses a software radio to implement the GSM air interface for direct communication with standard 2G GSM handset.
- Amarisoft [27]: a fully software-based LTE Base Station which allows to build a real 4G eNodeB using a standard PC and a low cost software radio front end.
- OpenAirInterface [25]: an open-source software-based implementation of the LTE system spanning the full protocol stack of 3GPP standard both in E-UTRAN and EPC.

2.3 OpenAirInterface

This section gives an introduction of OpenAirInterface and describes it as the reference SDR platform for the work of this Master thesis project. Detailed aspects on the software architecture will be then deepened in chapter 4.

2.3.1 Introduction

The vast majority of current generation of Radio Access Networks (RANs) are based on proprietary hardware (HW) and software (SW) components that stifle innovation and increase the cost for operators to deploy and maintain new services and applications in an ever-changing fast paced cellular network [25]. For this reason, radio systems are converging from a slow-moving proprietary and expensive HW and SW platforms towards an open cellular ecosystem. In this context, the emergence of industrial solutions based on free open-source software running on general purpose processors can greatly simplify network access, improve innovation speed and accelerate time-to-market for introduction of new services.

In 1998, EURECOM, a Franch based research institute, launched an experimental research activity for the development of next-generation of wireless communications systems. Through the years, the initiative evolved towards a free, open and flexible software framework for the implementation of standard-compliant experimental radio systems that, in 2009, goes under the name of "OpenAirInterface (OAI)" [25]. Parallel to this, to ensure openness, transparency and access to all, in 2014 OpenAirInterface has been placed under the responsibility of a French independent non-profit organization called OpenAirInterface Software Alliance (OSA), with its headquarters in EURECOM.

The OSA mission is to build a community of academic hackers and major industries that are embracing the usage of open source for the development of the next generation of mobile radio systems. As professor Raymond Knopp, co-founder of the OSA, says:
"We would become a very strong voice and maybe even a game changer in the 3GPP world and we will bring a real impact from the work that we are doing here in EURECOM into 3GPP systems"
The OSA primary future objective is to provide an open-source reference implementation which follows the 3GPP standardization process and the evolutionary path from 4G towards 5G systems. This has lead to the establishment of six strategic work areas, including also the Internet of Things, on which the alliance is currently investing and which are reported in Figure 2.4.

2.3.2 Platform Architecture

OpenAirInterface is one of the first and most complete open-source software-based implementation of the 4th generation of mobile radio systems, namely Long Term Evolution (LTE). It spans the full protocol stack of the 3GPP standard both in Evolved Universal Terrestrial Radio Access Network (E-UTRAN) and Evolved Packet Core (EPC). OAI currently includes a standard-compliant implementation of a subset of LTE Release 10 for UE, eNodeB, Mobility Management Entity (MME), Home Subscriber Server (HSS), Serving Gateway (SGW) and Packet Data Network Gateway (PGW) [28]. It can be used to build and customize an LTE base station and core network on a Personal Computer (PC) and connect commercial UEs to test and monitor the network and the mobile device in real time. Moreover, OAI provides a rich development environment with a range of build-in tools such as protocol analyser, emulation options and logging systems for all protocol stack layers and channels [5].

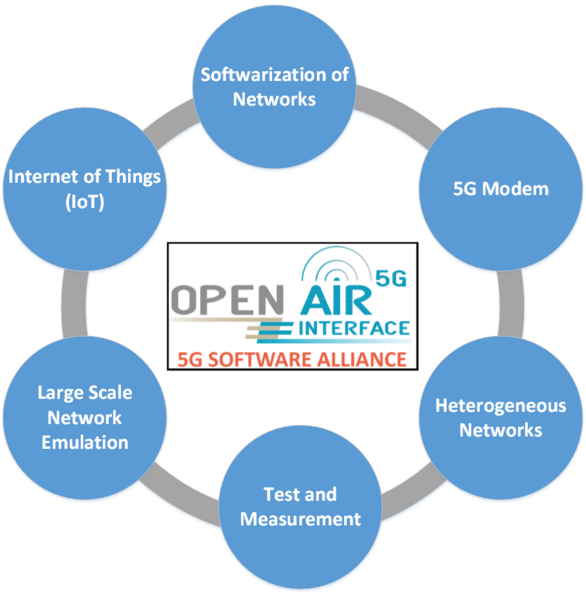

Figure 2.4: OSA strategic areas [4].

Software

At date, the entire OAI software is hold in the web-based Git repository manager GitLab, with dedicated wiki and issue tracking features. The OSA's EPC software is known as *openairCN* while the access network software goes under the name of *openairinterface5G*. OpenAirInterface is written in standard C language and released as free software under the terms of version 3 of the GNU General Public License (GPLv3). It works on several real-time Linux variants optimized for x86 architecture providing UE, eNodeB and core-network functionalities. Figure 2.5 shows the implemented LTE protocol software architecture of OpenAirInterface.

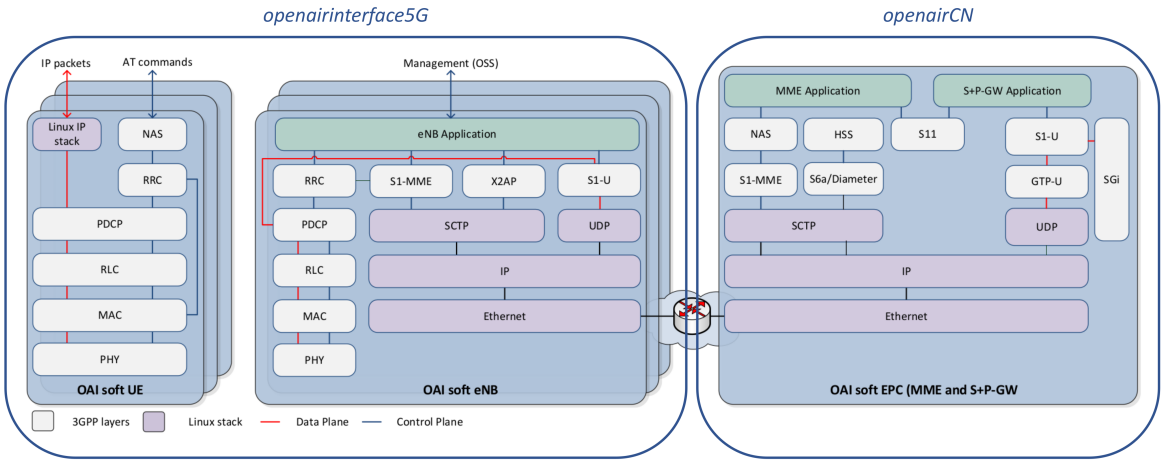

Figure 2.5: OpenAirInterface LTE software stack [5].

Hardware

OpenAirInterface is based on a PC hosted software radio front end architecture in which the transceiver functionalities are provided by a software-defined Radio Frequency (RF) front end connected to a host PC for processing. In particular, OAI is designed to be agnostic to the hardware RF platforms, allowing to be interfaced with 3rd party SDR RF solutions without significant effort [25]. For instance, OAI supports the recent Universal Software

Radio Peripheral (USRP) systems via USRP Hardware Driver (UHD) interface, LimeSDR as well as some proprietary HW platforms from industrial partners. Figure 2.6 shows the OpenAirInterface platform architecture for both RAN and EPC.

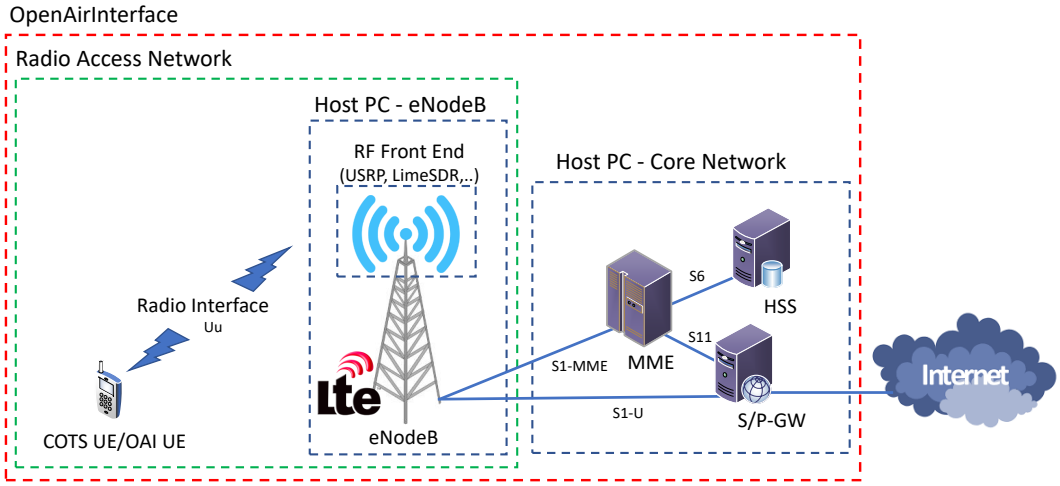

Figure 2.6: OpenAirInterface platform architecture.

Additionally, the OAI platform can be used in several different configuration involving also commercial components to varying degrees [5] as listed below.

- Commercial UE ↔ OAI eNB + Commercial EPC.
- Commercial UE ↔ OAI eNB + OAI EPC.
- Commercial UE ↔ Commercial eNB + OAI EPC.
- OAI UE ↔ Commercial eNB + OAI EPC.
- OAI UE ↔ Commercial eNB + Commercial EPC.
- OAI UE ↔ OAI eNB + Commercial EPC.
- OAI UE ↔ OAI eNB + OAI EPC.

2.3.3 The Alliance Project: Narrowband Internet of Things

On April 2017, at the Beijing University of Posts and Telecommunications (BUPT) in Beijing, People's Republic of China, the 3rd OpenAirInterface workshop took place. The workshop has brought together users and developers of OpenAirInterface from both academia and industry with the aim of sharing the latest developments from the community to the community [25]. Furthermore, the four-day workshop officially marks the start of the OSA's "Project 5" aiming for the implementation of the 3GPP standard NB-IoT on OpenAirInterface platform. The project is directly lead by the EURECOM institute in collaboration with national French industrial and universities including the University of Bologna (UNIBO). Indeed, this Master thesis project benefits from the participation of the School of Engineering and Architecture of Bologna under the OSA's "Project 5" for the study and the implementation of the NB-IoT protocol stack on OpenAirInterface.

2.4 Abstract Syntax Notation

Abstract Syntax Notation One (ASN.1) is an interface description language for defining data structures that can be serialized and de-serialized in a standard, cross-platform way [29]. Because the language is both human-readable and machine readable, is extensively used in telecom industry, computer networking and to build up security protocols. Standardization bodies, like 3GPP, define data structures in ASN.1 modules, which are generally a section of a broader standard document or specification. These modules can be automatically turned into libraries for processing of their data structures by an ASN.1 compiler. Moreover, this description language is closely associated with a set of encoding rules that specify how to present the ASN.1 data structures as a series of bytes during information exchange among communication systems. Among the others, the Packet Encoding Rules (PER, unaligned: UPER, canonical: CPER) is the reference Recommendation utilized by many open-source platforms including OpenAirInterface. In OAI software for instance, a specific directory is defined for containing the necessary scripts and Makefiles to generate all the LTE configuration data structures based on the ASN.1 source code. In particular, an *exctract_asn1_from_spec.pl* script is used for extracting the ".asn" files from the text version of 3GPP specifications and turn it into ".c" and ".h" C-based files that can be directly used in the OpenAirInterface code. Figure 2.7 shows a general ASN.1 extraction process.

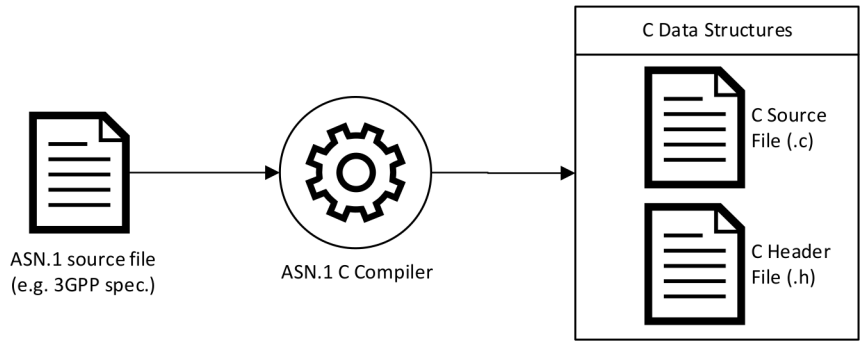

Figure 2.7: ASN.1 extraction process.

At runtime, the communication between two entities takes place through the usage of a UPER encoding script for transmission while a UPER-based decoder is used whenever data are received. This is exemplified in Figure 2.8.

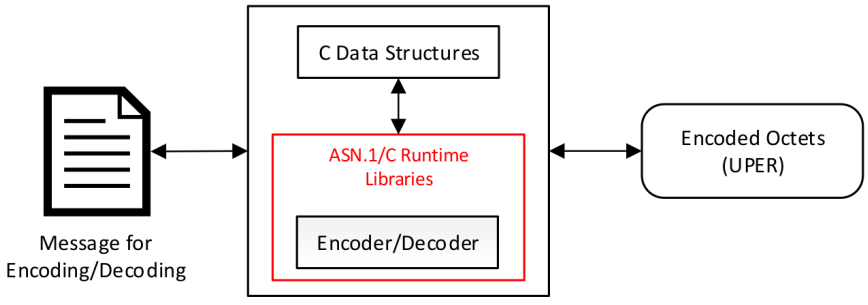

Figure 2.8: Runtime UPER encoding-decoding process.

2.5 Narrowband Internet of Things (NB-IoT)

On June 2016 the 3GPP completed the standardization of NB-IoT, a new LPWAN technology for the Internet of Things. NB-IoT is part of 3GPP Rel.13 specification (LTE Advance Pro) introduced as a new independent radio interface optimized for machine type traffic [6]. It is not fully backward compatible with existing 3GPP devices [7] though it ensures harmonious coexistence with current GSM, GPRS and LTE systems. To address key IoT requirements such as Machine Type Communication (MTC) the NB-IoT standard targets are derived:

- Massive connectivity
- Reduced UE complexity and costs
- Improved power efficiency
- Coverage Extension
- Deployment Flexibility

To fulfil these requisites, many advanced and even basic features of LTE Release 8 and 9 are not supported [10] while, at the same time, performance objectives led to a new system's design for both Access and Core Network at various levels.

First, this section gives an overview on NB-IoT network and radio protocol architecture. Afterwards, details on Physical and Radio Resource Control layers are discussed focusing on those aspects that have been investigated during this Master thesis project.

2.5.1 Network Architecture

2.5.1.1 Core Network

NB-IoT it is not just an IoT platform and software upgrade but also the roll-out of new entities in the Core Network, called Cellular IoT (CIoT) Evolved Packet System (EPS), shown in Figure 2.9.

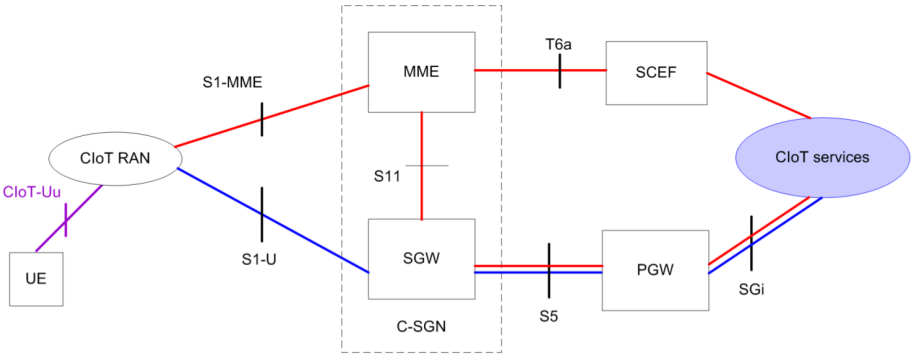

Figure 2.9: Core Network for the NB-IoT data transmission and reception. In red, the Control Plane CIoT EPS optimisation is indicated while in blue the User Plane CIoT EPS optimisation [6].

The new NB-IoT Core Network offers two major optimizations [30]:

Control Plane CIoT EPS Optimization (Solution 2): is the basic and mandatory solution to support infrequent small data transmission (IP data, non-IP data and SMS) over Signalling Radio Bearers avoiding the establishment of Data Bearers to save battery life. Non-IP data may be routed bi-directionally via the new network entity Service Capability Exposure Function (SCEF) or via SGW/PGW for IP data too.

User Plane CIoT EPS Optimization (solution 18): is the optional solution to support infrequent small data transmission (IP data and SMS) with the setup of up to 2 Data Radio Bearer for connection-oriented sessions.

A NB-IoT UE does not transfer data using both solution at the same time since both will be never configured by the network together. Instead, selection of which solution to be used is done between UE and network at Non-Access Stratum (NAS) level.

2.5.1.2 Access Network

From architecture view point, the NB-IoT Access Network has no difference to LTE as shown in Figure 2.10. The same S1 interface is used for connecting eNodeB to the MME and SGW and there is still an X2 interface between eNodeBs although no handover procedure is defined. Moreover, NB-IoT uses same frequency bands numbers as LTE but with a restricted set of 14 bands [31] most of which are in sub-GHz range, reflecting the need to leverage better coverage performance of UHF frequencies.

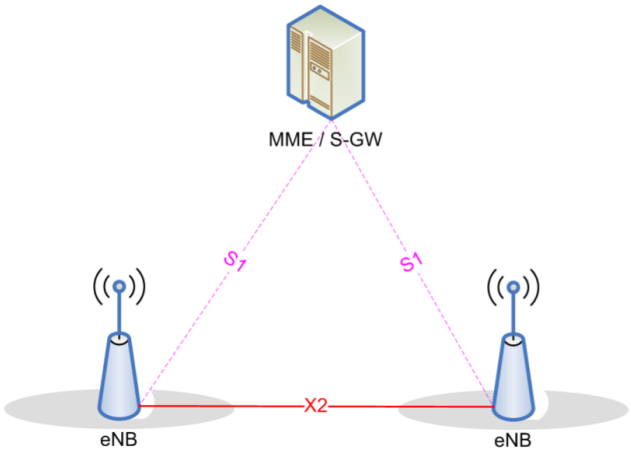

Figure 2.10: Network architecture towards the air-interface [6].

2.5.2 NB-IoT Radio Protocol Architecture

To build the NB-IoT protocol layers, 3GPP started with the LTE protocol specification, reduced it to a minimum and optimized it as needed for NB-IoT. This way, the proven structures and procedures are re-used while overhead from unused LTE features is prevented. Consequently, the NB-IoT technology can be regarded as a new air interface also from the protocol stack view-point, while being built on a well established framework [6]. Figure 2.11 shows the E-UTRAN protocol stack architecture for NB-IoT highlighting Radio Bearers, logical, transport and physical channels, as well as the main protocol layers functionalities and data paths for both user- and control-plane.

As in legacy LTE, each PDCP entity is associated either to control or user data flows being assigned to Signalling or Data Radio Bearer respectively. However, the introduction of a new Radio Bearer, SRB1bis, grants to bypass PDCP before security activation or whenever CIoT Control Plane solution is adopted by the core network. Moreover, PDCP SDU maximum size is reduced from 8188 to 1600 octects and only short size 7 bits Sequence Number (SN) is allowed instead of the 15 or 18 bits available. As for RLC layer, NB-IoT foresees to work only on Transparent and Acknowledged Mode with the possibility of choosing whether or not configure Status Report functionalities for reception failure directly from RRC layer. Moreover, NB-IoT reduced set of resources is posing challenges on radio resource management

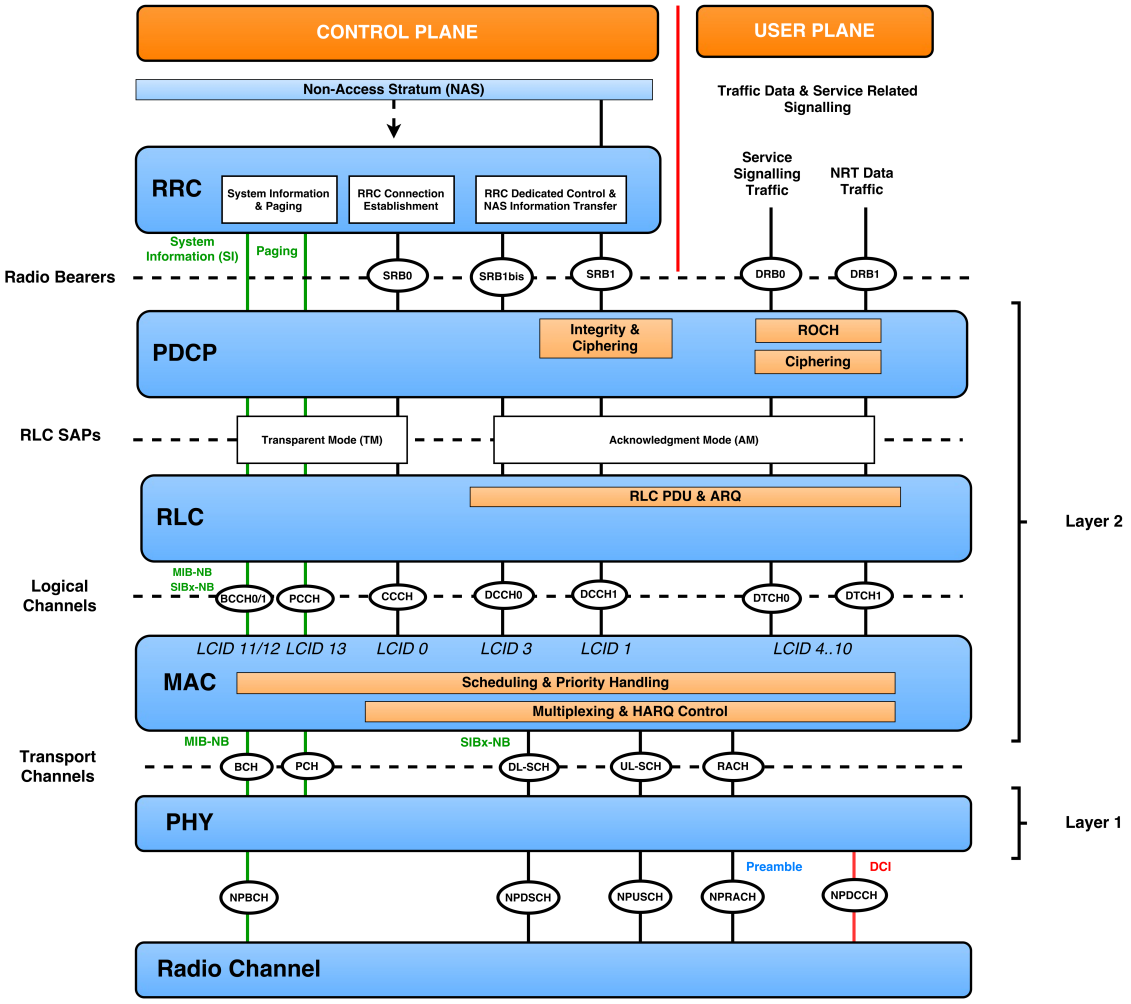

Figure 2.11: NB-IoT radio protocol stack architecture.

and assignment, in particular on MAC scheduler design which is considered by the scientific community one of the fundamental block of this new radio technology. Nevertheless, the most important modifications have been introduced in the PHY and RRC layer which will be discussed in the following subsections.

2.5.3 PHY

NB-IoT technology works on a 180 kHz bandwidth, i.e. one LTE PRB, applying a Type-B Half-Duplex Frequency Division Duplexing (FDD) mode with an additional subframe between every UL to DL switch [8]. The NB-IoT cell can be logically seen as a parallel cell to LTE having its own Narrowband Physical Cell Identity (NCellID), synchronization and physical signals. Therefore, deployment scenarios foresee three possible operating modes:

Stand Alone: the NB-IoT carrier is allocated irrespectively form the LTE band. Possible solution is the deployment over a GSM frequency carrier of 200 kHz bandwidth.

In-Band: the NB-IoT carrier is allocated within the LTE cell utilizing same resource blocks. However, the 100 kHz UE search raster implies that for in-band deployment a NB-IoT carrier can be placed only in certain PRBs [7]. Moreover, there is no support of this solution over LTE bands with 1.4 MHz bandwidth.

Guard Band: the NB-IoT carrier is allocated over the unused resource blocks within the LTE carrier's guard bands.

The NB-IoT operating modes are shown in Figure 2.12:

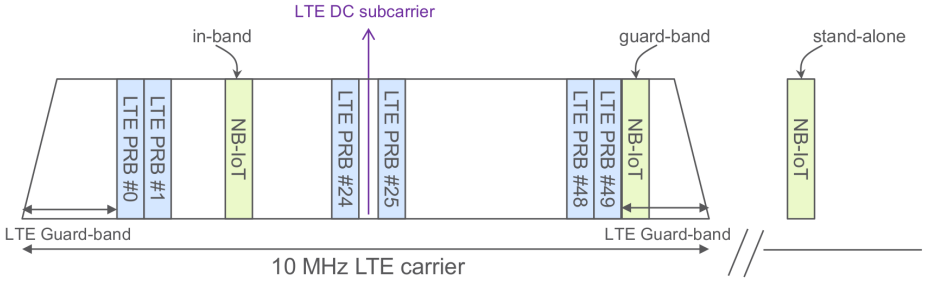

Figure 2.12: Examples of NB-IoT stand-alone deployment and LTE in-band and guard-band deployments in the downlink [7].

Furthermore, to cope with different radio conditions, NB-IoT technology aims on 20 dB link budget higher than LTE Rel.12, reaching up to 164 dB of Maximum Coupling Loss (MCL) [7]. Coverage extension is achieved allowing large number of repetitions, up to 2048 in downlink and up to 128 for uplink. These are managed by MME that configures up to 3 Coverage Enhancement (CE) levels to set the number of times a message should be repeated depending upon UE location.

2.5.3.1 Downlink

In the Downlink, Orthogonal Frequency Division Multiplexing (OFDM) scheme is adopted using a 15 kHz subcarrier spacing with 7 OFDMA symbols bounded into 1 slot of 0.5 ms. Slots are summed up into subframes and radio frames in the same way as for LTE as reported by Figure 2.13 in which the NB-IoT downlink resource grid is for size of one single PRB.

Also in this case, a Resource Element (RE) is defined as one subcarrier in one OFDMA symbol as indicated in Figure 2.13 by one square. However, the RE channel mapping varies depending if In-Band, Guard Band or Stand Alone operating mode is deployed. In addition, NB-IoT introduces the concept of Hyper-System Frame Number (HSFN) which counts legacy LTE System Frame periods incrementing by 1 each time the SFN wrap around (i.e. every 1024 frames). This new time basis results useful for energy saving mechanism such as Power Saving Mode (PSM) or extended Discontinuous Reception (eDRX) on which long interval of time are considered.

For the Downlink, NB-IoT defines three physical signals:

- Narrowband Reference Signal (NRS): transmitted every NB-IoT downlink subframe.
- Narrowband Primary Synchronization Signal (NPSS): transmitted in subframe #5.
- Narrowband Secondary Synchronization Signal (NSSS): transmitted in subframe #9 of even radio frames.

and three physical channels:

- Narrownband Physical Broadcast Channel (NPBCH): transmitted in subframe #0.
- Narrowband Physical Downlink Control Channel (NPDCCH): transmitted in remaining subframes.
- Narrowband Physical Downlink Shared Channel (NPDSCH): transmitted in remaining subframes previously scheduled through NPDCCH.

These, are primarily multiplexed in time generating the NB-IoT downlink subframe structure reported in Figure 2.14 for both odd and even frames.

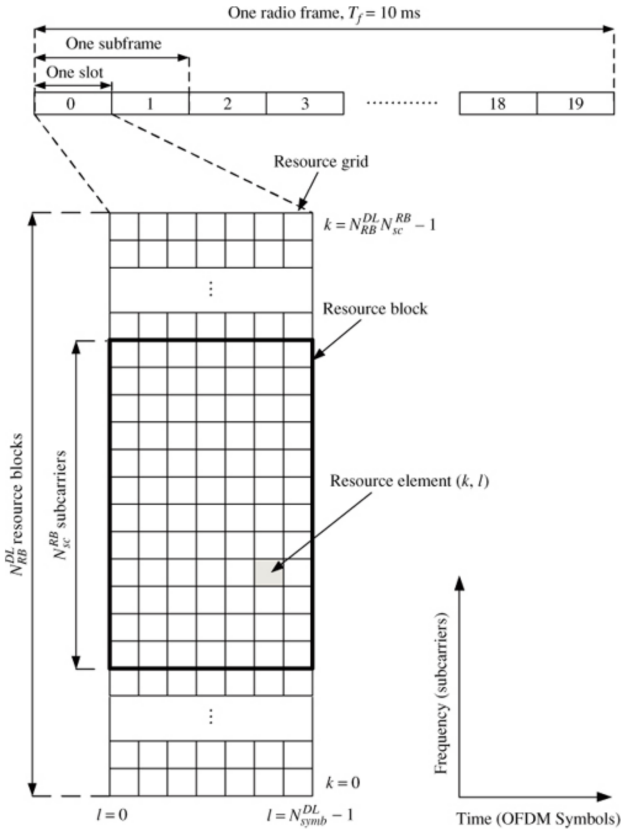

Figure 2.13: LTE Downlink Resource Grid [8].

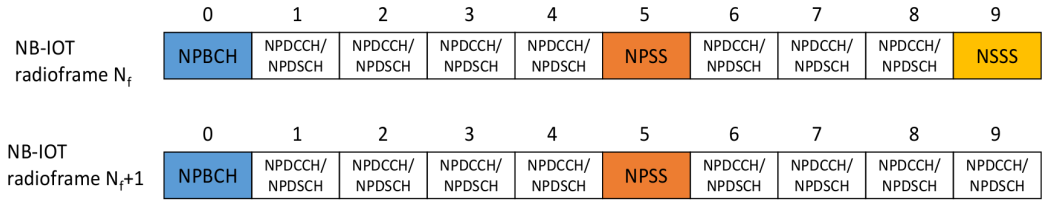

Figure 2.14: NB-IoT Downlink Subframe [9].

As can be noticed, there are less downlink channels than for LTE. For instance, Physical Multicast Channel (PMCH) is not included since no eMBMS service is defined for NB-IoT.

2.5.3.2 Uplink

In the UL, Single Carrier Frequency Division Multiple Access (SC-FDMA) with 3.75 kHz or 15 kHz subcarrier spacing is applied. For 15 kHz, same resource grid of Figure 2.13 applies while for 3.75 kHz the modified structure in Figure 2.15 for 2 ms slot is used. Again there are 7 OFDM symbols within a single slot.

For the Uplink, NB-IoT defines two physical channels:

- Narrowband Physical Uplink Shared Channel (NPUSCH)
- Narrowband Physical Random Access Channel (NPRACH)

and one reference signal:

- Demodulation Reference Signal (DMRS)

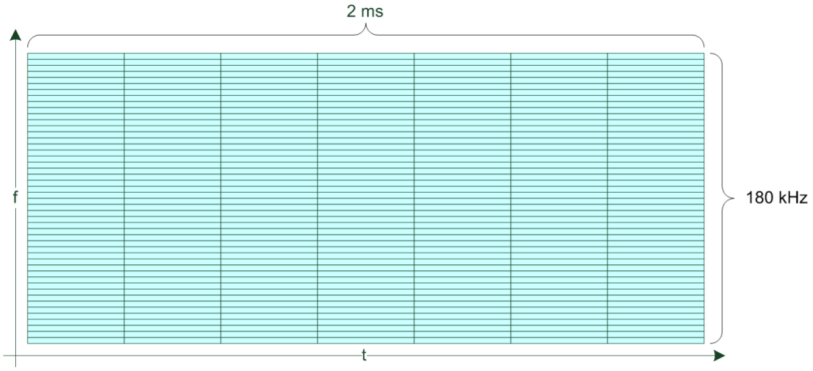

Figure 2.15: NB-IoT resource grid for 3.75 kHz subcarrier spacing. There are 48 subcarriers for the 180 kHz bandwidth [6].

Except for RACH transmission, all data are transmitted over NPUSCH including also the Uplink Control Indication (UCI) since there is no equivalent to the LTE PUCCH for NB-IoT [6]. Over NPUSCH, the smallest unit to mapping a transport block is called Resource Unit (RU) whose definition depends on subcarrier spacing and NPUSCH Format used as described in [8] chapter 10.1.2.3. Moreover, the new NPRACH implements pseudo-random single-tone frequency hopping for preamble transmission as specified in [8] chapter 10.1.6.

For the sake of completeness, Table 2.1 summarizes the major changes introduced at physical layer by NB-IoT in comparison to legacy LTE Rel.9. Some of them have been discussed in this subsection, others are reported to give a wider look to this new radio technology.

	LTE Rel.9	**NB-IoT (LTE Rel.13)**
System Bandwidth	1.4 MHz, 3 MHz, 5 MHz, 10 MHz, 15 MHz, 20 MHz	180 kHz (200 kHz)
Duplexing Mode	Full Duplex FDD/TDD	Half-Duplex FDD
Throughput	DL: 150 Mbps, UL: 50 Mbps	DL/UL (tilde) 200 Kbps
Operating Mode	LTE Licensed Spectrum	In-Band, Guard Band, Stand Alone
Maximum Coupling Loss (MCL)	$\sim$ 145 dB	$\sim$ 164 dB
Carrier Spacing	DL/UL: 15 kHz	DL: 15 kHz, UL: 15kHz or 3.75 kHz
Transmission Mode	TM1-TM9	TM1/TM2 (1 antenna or 2 antenna)
Synchronization Signal	PSS/SSS	NPSS/NSSS
Detection of UL and DL channel	DL: CRS, UL: DMRS	DL: NRS, UL: DMRS
Downlink Channel	PDSCH	NPDSCH
	QPSK, 16 QAM, 64 QAM	QPSK
	1/3 Turbo Coding	1/3 Tail biting convolutional coding
	1 subframe for Transport Block	1 or more subframes for Transport Block
Control Channel	PDCCH	NPDCCH
	Use 1 to 3 OFDM symbols of the first slot of the same PDSCH subframe	Use an entire NB-IoT DL subframe
	DCI Format: 0, 1, 1A, 2, 2A, 3, 3A, ...	DCI Format: N0 (UL), N1 (DL), N2 (DL)
Uplink Channel	PUSCH	NPUSCH
	15 kHz sub-carrier spacing	15 kHz or 3.75 kHz sub-carrier spacing
	1/3 Turbo Coding	1/3 Turbo Coding
	Use 1 subframe for transmission	Resource allocation based on Resource Unit (RU)
	UL-SCH and UCI over same subframe	UL-SCH without UCI
Low Power Consumption Technique	DRX	PSM, eDRX

Table 2.1: PHY layer features of legacy LTE Rel.9 and NB-IoT in comparison.

2.5.4 RRC

Similarly to the physical layer, NB-IoT introduces drastic changes at RRC layer too, with the establishment of new messages and dedicated Information Element (IE). Being a non backward-compatible variant of E-UTRAN, some of the normal LTE functionalities and corresponding procedures are not supported in NB-IoT while others equally apply. These, are partially summarized in Table 2.2 with reference to 3GPP Rel.13 specification available at [10]. However, with further releases, new RRC capabilities have been introduced for NB-IoT but not reported here since out of the scope of this Master thesis.

Not Supported	**Supported**
Connected Mode Mobility (Handover and measurement reporting)	System Information
Inter-RAT cell reselection or inter-RAT mobility in connected mode	Connection Control
Relay Node (RN)	DL Information Transfer
Dual Connectivity (DC)	UL Information Transfer
Carrier Aggregation (CA)	UE Capability Transfer
MBMS (Multimedia Broadcast Multicast Service)	General Error Handling
Self-configuration and self-optimisation	
Measurement configuration and reporting	
Sidelink (including direct communication and direct discovery)	
Real time services (including emergency call)	

Table 2.2: RRC features and procedures not supported or supported by NB-IoT [10].

Additionally, the usage of different CIoT solutions for data transfer (see subsubsection 2.5.1.1) introduced some exceptions on the RRC Connection Control procedures that can be applied depending on whether user- (UP) or control-plane (CP) optimization is adopted. This is shown in Table 2.3.

Connection Control Procedures	**CP**	**UP**
Paging	✓	✓
RRC connection establishment	✓	✓
RRC connection resume		✓
Initial Security Activation		✓
RRC connection reconfiguration		✓
RRC connection re-establishment		✓
RRC connection release	✓	✓
RRC conection release requested by upper layers	✓	✓
Radio resource configuration	✓	✓
Radio link failure related actions	✓	✓
UE actions upon leaving RRC_CONNECTED	✓	✓

Table 2.3: RRC Connection Control procedures for different CIoT EPS Optimization.

Since no handover to LTE or different Radio Access Technology (RAT) is supported, the NB-IoT RRC state transition model simplifies to the one shows in Figure 2.16.

As in LTE, there are only two states, RRC_IDLE and RRC_CONNECTED, without other associated UTRA or GSM states. RRC_CONNECTED to RRC_IDLE transition occurs whenever Radio Link Failure or cell selection/re-selection is required. In the latter case, the UE has first to release the connection, selects another "suitable" cell [32] and establishes a new communication.

When a NB-IoT UE camps on a cell, it follows the same principle as for LTE: it first acquires

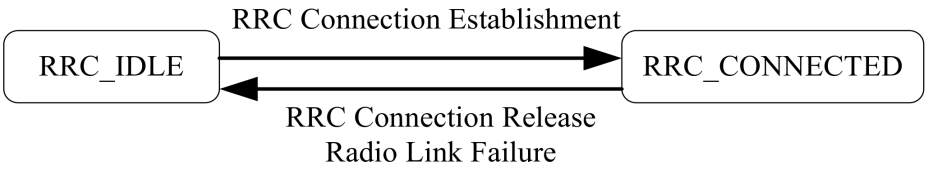

Figure 2.16: Model of RRC states and their transitions.

Physical Cell ID (NCellID), time slot and frame synchronization through NPSS and NSSS. Then, the UE reads associated System Information Blocks (SIBs) and starts the Random Access Procedure to establish an RRC Connection and register within the core network. The NB-IoT cell access flow with corresponding messages is reported in Figure 2.17. To be noticed that a NB-IoT (-NB) version for System Information as well for RRC messages is defined, while Random Access procedure has the same message flow as LTE although with different carried parameters.

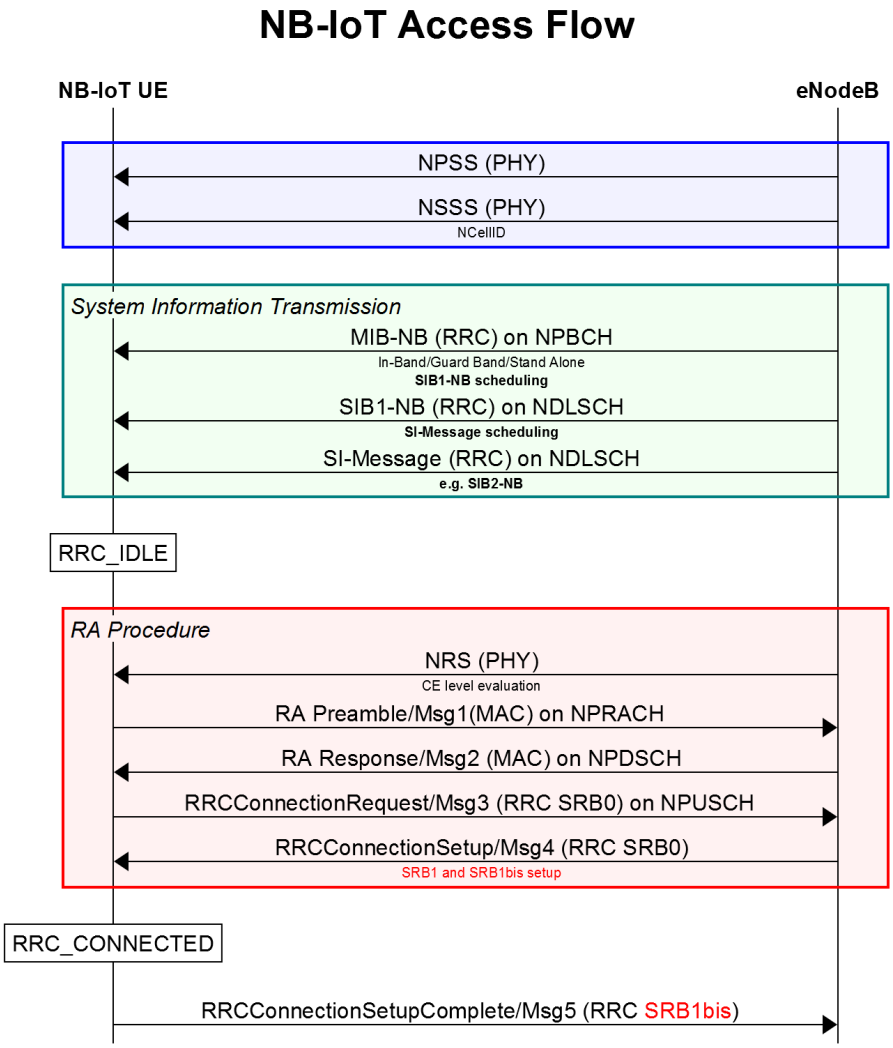

Figure 2.17: NB-IoT cell access flow.

Overall, NB-IoT introduced further changes on UE Capabilities, Access Barring, Establishment Causes, Radio Link Failure and other procedures involving RRC layer at different levels. In the remaining part of this section, the new Radio Bearer and System Information design for NB-IoT are discussed, since of primary interest for this Master thesis and for the work

presented in the following chapters.

2.5.4.1 Radio Bearers

Similarly to LTE, NB-IoT radio bearers are classified as Data Radio Bearers (DRBs) and Signalling Radio Bearers (SRBs). In particular, the latter are partly re-used from legacy LTE systems by NB-IoT which defines three [10]:

- SRB0: is for carrying common RRC messages transmitted using CCCH logical channel.
- SRB1: is for carrying dedicated RRC messages (which may include a piggybacked NAS message) as well NAS messages, all using DCCH logical channel.
- SRB1bis: is for carrying dedicated RRC messages (which may include a piggybacked NAS message) as well NAS messages prior the activation of security, all using DCCH logical channel.

As can be noticed, there is no SRB2 defined but, in addition, a new Signalling Radio Bearer 1bis (SRB1bis) is introduced as previously highlighted in Figure 2.17. SRB1bis is implicitly established with SRB1 using the same configuration but no PDCP entity [10]. This because, it takes the role of SRB1 until security is activated and then is not used anymore [6]. SRB1bis distinguishes from SRB1 by only the Logical Channel Identity (LCID) that is equal to 3 instead of 1. Moreover, for NB-IoT UEs that only supports the mandatory Control Plane CIoT EPS optimization (see subsubsection 2.5.1.1) SRB1bis is always used since there is no security activation in this mode.

As for Data Radio Bearers, instead of the 8 allowed by LTE, NB-IoT has reduced the number. To keep the complexity low, a NB-IoT UE supports 0,1 or only up to 2 DRBs simultaneously, depending on its capability. These, are configured together with Signalling Radio Bearers through *RRCConnectionSetup-NB* or *RRCConnectionReconfiguration-NB* messages. Furthermore, A NB-IoT UE that only supports the Control Plane CIoT EPS optimisation (see subsubsection 2.5.1.1) does not need to support any DRBs and associated procedures.

2.5.4.2 System Information

NB-IoT defines a reduced set of System Information Blocks, indicated with suffix "-NB", with similar functionalities as LTE but modified Information Elements (IEs). These, are shown in Table 2.4 with reference to 3GPP Rel.13. Further SIBs have been introduced in NB-IoT Rel.14 and later versions available at [10] but not reported here since out of the scope of this Master thesis.

System Information Block	**Content**
MasterInformationBlock-NB (MIB-NB)	Essential information required to receive further System Information
SystemInformationBlockType1-NB (SIB1-NB)	Cell access and selection, information for other SIB scheduling
SystemInformationBlockType2-NB (SIB2-NB)	Radio resource configuration information
SystemInformationBlockType3-NB (SIB3-NB)	Cell re-selection information for intra-frequency, inter-frequency
SystemInformationBlockType4-NB (SIB4-NB)	Neighbouring cell related information relevant for intra-frequency cell re-selection
SystemInformationBlockType5-NB (SIB5-NB)	Neighbouring cell related information relevant for inter-frequency cell re-selection
SystemInformationBlockType14-NB (SIB14-NB)	Access Barring parameters
SystemInformationBlockType16-NB (SIB16-NB)	Information related to GPS time and Coordinated Universal Time (UTC)

Table 2.4: NB-IoT System Information Blocks for 3GPP Rel.13.

NB-IoT UE exclusively use these SIBs and ignore those from LTE, even in the case of in-band operation. It is always mandatory for a UE to have a valid version of MIB-NB, SIB1-NB and SIB2-NB through SIB5-NB [10] while other SIBs are needed only if their functionalities

are required. Moreover, System information acquisition and change procedure is only applied in the RRC_IDLE state, therefore, NB-IoT UEs are not expected reading SIB information while being in the RRC_CONNECTED state [6]. Furthermore, to cause minimum UE battery consumption, the physical layer imposes a limit of 680 bits to the maximum size a SIB and SI message can take [33].

2.5.4.3 System Information Scheduling

In terms of sequence of decoding, NB-IoT takes a similar approach to LTE. It starts first with MIB-NB, then gets SIB1-NB and finally extracts SI-Messages containing SIB2-NB and other Information Blocks. However, the most outstanding difference between the two radio technologies is that for NB-IoT there is no Downlink Control Indication (DCI) associated to SI or SIBs. All necessary information to acquire System Informations are notified to UE over MIB-NB first and SIB1-NB later as highlighted in bold in previous Figure 2.17. This defines a "NPDCCH-less" approach in which scheduling parameters are fixed instead of being dynamically indicated on NPDCCH. Moreover, NB-IoT SI messages are transmitted discontinuously to take advantages of time diversity since spreading out transmission across time can improve performance at very low Signal-to-Noise Ratio (SNR) conditions [34]. In the following, scheduling procedure for Narrowband Master Information Block, System Information Block 1 and System Information Messages is discussed since constitutes the starting point for the work presented in subsubsection 5.1.5.2.

MIB-NB Scheduling The Narrowband Master Information Block (MIB-NB) is the fundamental message needed to acquire essential information from the cell on which the UE is camped. Therefore, its scheduling results in the most frequent transmissions and repetitions by the eNodeB. MIB-NB contains 34 bits and is transmitted over the NPBCH channel using a fixed schedule with a periodicity of 640 ms (64 radio frames) over which repetitions are made. The first transmission of the MIB-NB is scheduled in subframe #0 of radio frames for which $SFNmod64 = 0$ and repetitions are made in subframe #0 of all the next consecutive radio frames. Moreover, due to baseband processing, the MIB-NB transmission is arranged in 8 independent decodable blocks of 80 ms duration each. The first block is transmitted in the first subframe of the starting radio frame and repeated in subframe #0 of the next 7 consecutive radio frames, respectively. In the subframe #0 of the following radio frames the same procedure is applied for MIB-NB Block #2 (BL2) and all the others blocks. This process is continued until the whole Narrowband Master Information Block is transmitted. Figure 2.18 shows the MIB-NB scheduling highlighting frames, subframes numbers and decodable blocks delivery.

SIB1-NB Scheduling The Narrowband System Information Block Type1 (SIB1-NB) is transmitted over the NPDSCH channel using a fixed schedule with periodicity of 2560 ms (256 radio frames) over which 4, 8 or 16 equally spaced repetitions are made. SIB1-NB transmission occurs in subframe #4 of every other frame in 16 continuous frames [10]. Four possible Transport Block Sizes (TBSs) of 208, 328, 440 and 680 bits are defined and SIB1-NB content may only be changed on each modification period, which has a length of 4096 radio frames, i.e. 40.96 seconds. The SIB1-NB starting radio frame and the number of repetitions are derived from the eNodeB Narrowband Physical Cell Identity (NCellID) and from the *schedulingInfoSIB1* parameter delivered in the Narrowband Master Information Block. Figure 2.19 reports the evaluation procedure for SIB1-NB repetitions and starting radio frame through Table 16.4.1.3-3 and Table 16.4.1.3-4 as specified in [33]. Moreover, Figure 2.20 shows the resulting SIB1-NB scheduled transmission in subframe #4 with starting radio frame #16 and repetition number 4.

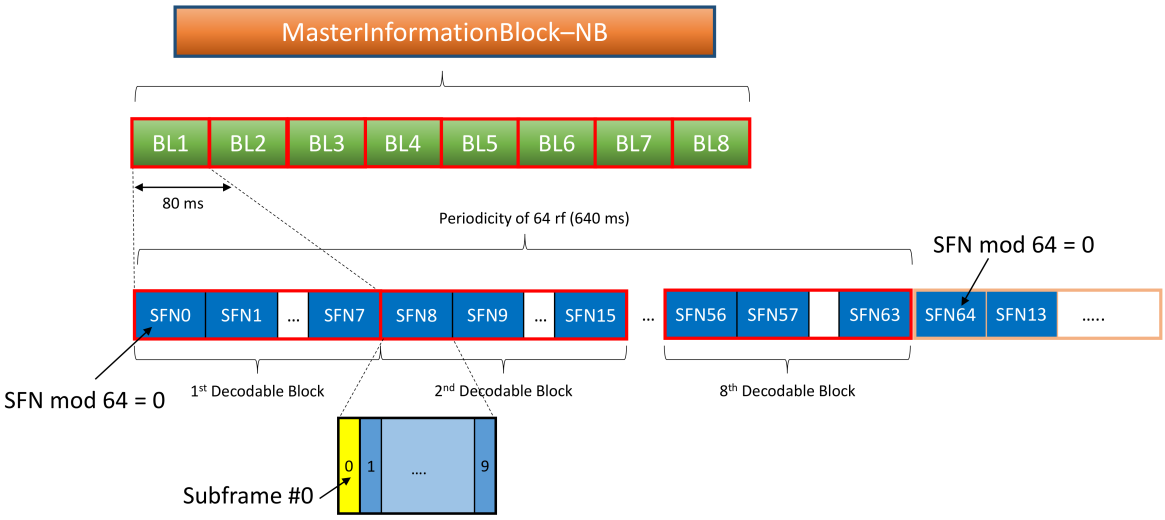

Figure 2.18: MIB-NB scheduling.

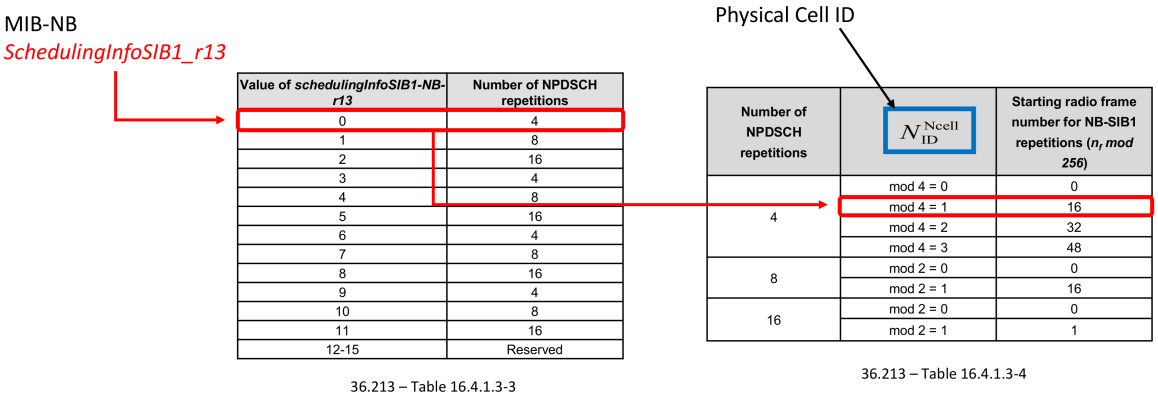

Figure 2.19: Evaluation procedure for SIB1-NB repetitions and starting radio frame.

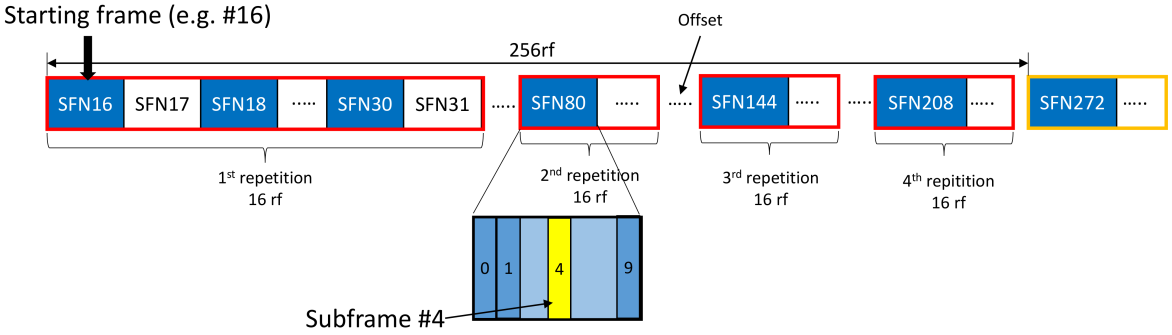

Figure 2.20: SIB1-NB scheduling example. Starting radio frame #16 and repetition number 4.

Since repetitions are equally spaced and in each of them SIB1-NB is delivered in every other frame within 16 continuous ones, transmissions may occur in odd or even frames depending if the starting radio frame is an odd or even number, respectively. Moreover, given the repetitions characteristics, the following formula can be used to calculate the repetition offset in radio frames (rf):

$$Offset = \frac{256rf - (16 \cdot n\_repetitions)}{n\_repetitions} \tag{2.1}$$

with $n\_repetitions$ given by Table 16.4.1.3-3 shown in Figure 2.19.

SI-Message Scheduling Narrowband System Information Blocks (SIBs) other than SIB1-NB and with same scheduling requirements are mapped and carried in System Information Messages (SI-Messages) with the mapping procedure that is flexibly configurable by the *schedulingInfoList* parameter carried by SIB1-NB. As for LTE system, also in NB-IoT the *SystemInformationBlockType2-NB* is always mapped in the SI-Message that corresponds to the first entry of the scheduling list of SIB1-NB. As previously mentioned, the SI-Messages scheduling is "NPDCCH less": all the necessary scheduling information for the message acquisition by the UE like timing, Transport Block Size (TBS) and repetition patterns are directly indicated by SIB1-NB parameters instead of being dynamically specified in a DCI. SI-Messages are transmitted within periodically occurring time domain windows, referred as SI-Windows, using scheduling information provided in SIB1-NB. Each SI-Message is associated with its own SI-Window, and SI-Windows of different SI-Messages do not overlap [10] such that the UE receives only one single information message within one SI-Window. The window length and its possible starting offset are common to all SI-Messages and are configurable through the SIB1-NB's *si_ windowLength* and *si_ RadioFrameOffset* parameters respectively. Moreover, within the SI-Window the corresponding SI-Message is transmitted over a number of consecutive downlink subframes depending on the TBS. For instance, a TBS of 56 or 120 bits requires only 2 subframes while other TBSs are transmitted over 8 subframes.

Table 2.5 summarizes the most relevant parameters provided by SIB1-NB for SI-Messages scheduling together with a brief description of their functionalities.

SIB1-NB parameter	**Description**
si_ WindowLength	Absolute length in milliseconds of the SI-Window. It is common to all SI-Messages.
si_ RadioFrameOffset	Offset in number of radio frames to calculate the starting of the SI-Window. It is common to all SI-Messages.
schedulingInfoList	List of SI-Messages carried by,the SIB1-NB.
si_ Periodicity	Periodicity of the SI-Message,in radio frames.
si_ RepetitionPattern	Starting radio frames within,the SI-Window in which an SI-Message transmission occurs.
sib_ MappingInfo	List of the SIBs mapped in the SI-Message.
si_ TB	Indicates the Transport Block Size in number of bits used to broadcast the SI-Message and the corresponding number of consecutive downlink subframes required.
downlinkBitmap	Indicates the allowed subframe configuration for downlink transmission. If not present, all subframes are valid except for those carrying NPSS/NSSS/NPBCH/SIB1-NB.

Table 2.5: Relevant SIB1-NB parameters for SI-Message scheduling.

As reported in [10] and based on parameters of Table 2.5, every SI-Window starts in subframe #0 in the radio frame that satisfies the following formula:

$$(HSFN \cdot 1024 + SFN) mod T = FLOOR(\frac{x}{10}) + Offset \tag{2.2}$$

where T is the *si_ Periodicity*, $Offset$ corresponds to *si_ RadioFrameOffset* and x is an integer value determined from the formula:

$$x = (n-1) \cdot w \tag{2.3}$$

with w the *si_ WindowLength* and n a number corresponding to the order of entry of the concerned SI-Message in the *schedulingInfoList* of SIB1-NB.

After determining the SI-window start, a NB-IoT UE accumulates SI-Message transmissions on Downlink Shared Channel (DLSCH) until the end of the window, starting from the radio

frames as provided by *si_RepetitionPattern* and in subframes as provided by *downlinkBitmap*, if defined, until a successful decoding of the message. In the following, a simple transmission example of two SI-Messages (SI_1 and SI_2) is shown in Figure 2.21 with the corresponding parameters setup reported in Table 2.6.

SI_1 parameter	**Value**
si_WindowLength	160 ms
si_RadioFrameOffset	0
si_Periodicity	64 RF
si_RepetitionPattern	Every2ndRF
sib_MappingInfo	sibType3_NB, sibType4_NB, sibType5_NB
SI_2 parameter	**Value**
si_WindowLength	160 ms
si_RadioFrameOffset	0
si_Periodicity	128 RF
si_RepetitionPattern	Every8thRF
sib_MappingInfo	sibType14_NB, sibType16_NB

Table 2.6: SIB1-NB parameters setup for SI_1 and SI_2 message scheduling of Figure 2.21.

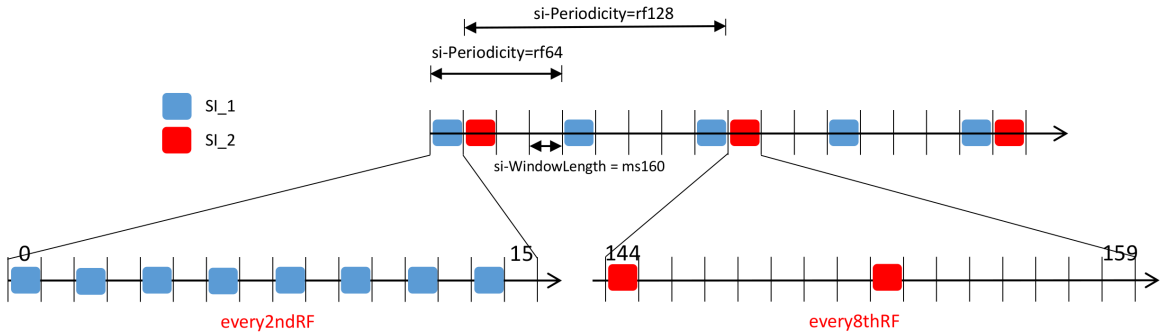

Figure 2.21: SI-Message scheduling example.

In NB-IoT, various possible configurations of System Information delivery poses challenges on efficient combination of SI-Message transmission together with SIB1-NB and MIB-NB. This is also dependent on messages' TBS which affects the number of downlink subframes needed. For instance, block sizes greater than 120 bits require 8 subframes for SI-Message delivery and if the SI transmission frame coincides with the *SystemInformationBlock1-NB* one, there will be one subframe less (subframe #4). As a consequence, an UE might be unable to completely receive a SI-Message in one single frame but should continue to receive remaining parts in other radio frames following the *si_RepetitionPattern*. This and other aspects on System Information scheduling will be resumed in subsubsection 5.1.5.2.

Chapter 3

Small Cell Deployment and Functional API

3.1 Introduction

Today, mobile networks have shifted from being predominantly voice networks to mainly transporting data with an unabated growing of consumer demand for "always-on", high speed and low-latency services. This scenario poses many challenges to mobile network operators, the first of which is how supply the enormous data capacity required and how to ensure economic sustainability at the same time. It is widely accepted that in order to achieve higher capacity and increased speeds in a real world deployment, many more cell sites are required and this approach has not escaped to operators that have been quick to endorse small cells deployment in residential, urban and rural areas. There is no formal definition of the term "small cell" but it mostly refers to low-powered radio access nodes that operate in licensed and/or unlicensed spectrum, with a coverage range of 10 meters up to hundreds of meters. Nevertheless, small cell technology has evolved considerably in the late years to provide longer range, higher capacity designs and Quality of Service (QoS) management while maintaining scalability and cost-effectiveness. For this reason, sometimes the term "small cell" may imply femtocells, picocells and microcells concepts [35]. The need to accelerate small cell adoption to change the shape of mobile networks and maximize the potential of mobile services [14] resulted in the creation of the non-profit organization "Small Cell Forum" in 2017. It is not a standards organization but partner with organizations like 3GPP, ETSI, GSMA and others that inform and determine standards development [14]. The Forum's work is concerned with the multiple ways in which licensed small cells can be deployed across residential, urban and rural networks by providing operators with all the information they need to successfully launch a small cell technology in their own systems. To this end, with more than fifty documents, the Small Cell Forum Release Program provides a combination of practical working guides, case studies and technical "how-to" publications to ease the roll-out of small cells.

3.2 Functional and Network Functional API

Among the different initiatives within the small cell industry, the Small Cell Forum have driven the standardization of two common Application Platform Interfaces (APIs) presented in the Release 9 document available at [9]. The first goes under the name of Functional Application Platform Interface (FAPI) and is the internal interface between the MAC and PHY protocol layers within a small cell ecosystem in which the eNodeB is considered as a single element. The second is an extension of the FAPI called Network Functional Application Platform Interface (nFAPI) where the eNodeB functionalities reside partially in a Virtual Network Function (VNF) and partially in a Physical Network Function (PNF). Also in the latter, the MAC-PHY interface has been identified as a suitable split point towards a converged approach

to virtualization in which a packet switched IP network is used to support communication between the VNF and PNF entity. Figure 3.1 shows the FAPI and nFAPI architecture and the possible relationship between the two.

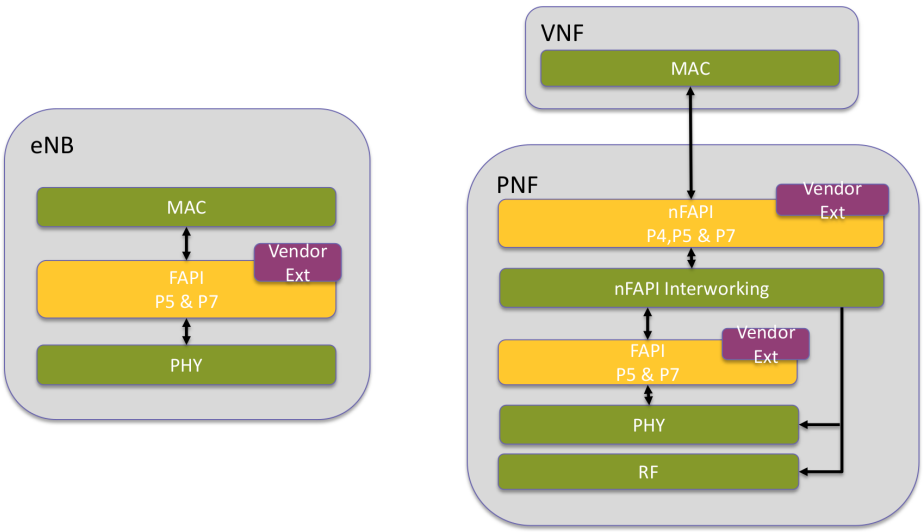

Figure 3.1: FAPI vs nFAPI architecture [9].

The FAPI and nFAPI standardization aims to provide a common architecture over a scalable ecosystem in which parts are interchangeable, ensuring that system vendors can take advantage of the least hardware and software innovation with minimum barriers to entry and the least amount of custom re-engineering [9].

The Release 9 specification addresses the usage of commercial small cells both for today's 4th and for future 5th generation of mobile radio networks. The document [9] foresees the adoption of a 3GPP eNodeB compliant with Rel.8 up to Rel.13 standard. The latter includes also the new radio technology developed for the IoT market, NB-IoT, for which an entire chapter addressing major use cases is dedicated in the Forum's document. This has been the reference starting point in this Master project for the development of a FAPI compliant MAC-PHY interface on OpenAirInterface platform as will be presented in section 5.2.

Moreover, to accelerate the adoption of Small Cell Forum's standards, in 2017 CISCO company started an open-source project called "open-nFAPI" [15]. Open-nFAPI is the implementation of nFAPI, aiming to provide an open interface between LTE layer 1 and layer 2 to allow for interoperability between the PNF and VNF. This project rose interests in the research communities as a valuable tool for bring FAPI/nFAPI functionalities on their systems and to align with the multi-vendor platform requirements for the next generation of small cell deployments. For this reason, open-nFAPI has been reused with some extent also in OpenAirInterface as a valuable tool for supporting the MAC-PHY functional split introduced during this Master thesis.

3.3 FAPI Procedures and Messages for NB-IoT

The purpose of this section is to give a general overview of the major Narrowband-IoT FAPI procedures and messages specified in [9] that have been subject of study during this Master thesis.

The reference architectures presented in Figure 3.1 foresee the usage of several APIs for both FAPI and nFAPI use cases:

- P4: is the Network Monitor Mode (NMM) interface that is typically used by PNF for scanning neighbouring LTE or E-UTRAN cells.
- P5: is the PHY mode control interface supporting different configuration procedures for the physical layer.
- P7: is the data path interface controlling and managing DL and UL data transfer between PHY and higher protocol stack layers.

In FAPI standard, both control- and data-plane information is passed through the PHY-MAC API (L1 API). Interactions with upper protocol layers (L2/L3) are managed by a PHY Control Entity for configurations and by the MAC layer for data-plane messages exchange. As a result, the FAPI procedures are split into two groups, namely, Configuration Procedures and Subframe Procedures taking place over P5 and P7 respectively. Configuration procedures handle the management of the PHY layer and are expected to occur infrequently, while Subframe procedures determine the structure of each 1 ms subframe and operate with a 1 ms periodicity [9]. Figure 3.2 provides an example on how the different L2/L3 protocol layers interact with the L1 API through the P5 and P7 interfaces.

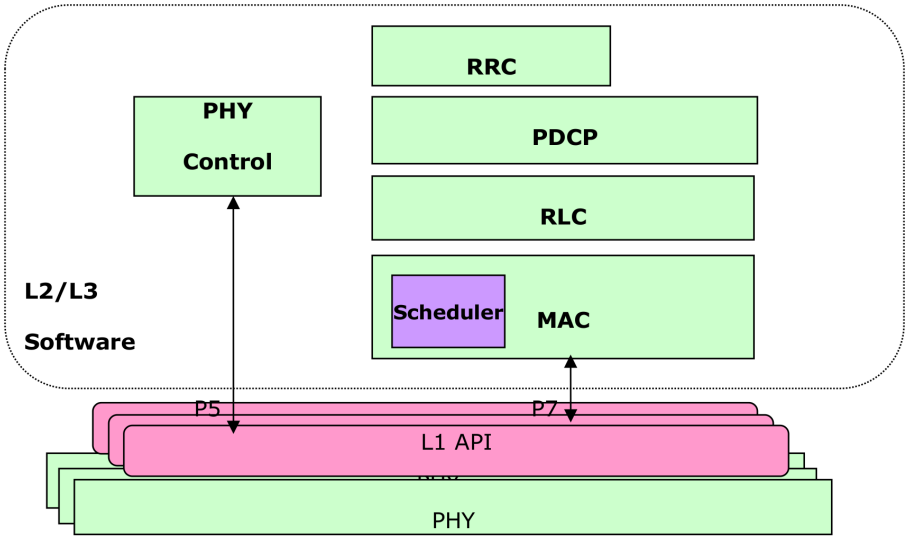

Figure 3.2: L1 API interactions [9].

3.3.1 P5 Configuration Procedure

In Configuration Procedures, the Physical layer is presented as a Finite State Machine (FSM) which can move in IDLE, CONFIGURED and RUNNING states through a set of messages received over the P5 interface. Figure 3.3 shows the PHY states and the corresponding messages for state transition.

Each configuration message is characterized by a specific collection of Type-Length-Value (TLV) which specify the different carried elements. In particular, relevant for this Master thesis work is the *CONFIG.request* message which allows the L2/L3 software to configure the PHY layer at initialization time bearing all the necessary NB-IoT Information Elements (IE). Implementation details of this and other configuration messages on the OpenAirInterface platform will be given in section 5.2.

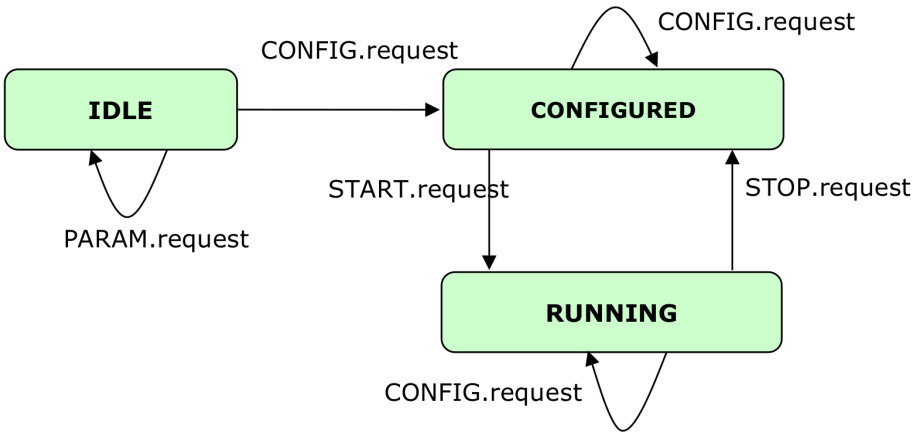

Figure 3.3: PHY layer state transactions on L1 API configuration messages [9].

3.3.2 NB-IoT Downlink Subframe Procedure

The Subframe Procedure has two purposes. On one hand, it is used to control the Downlink and Uplink frame structures by maintaining System Frame Number (SFN) and Sub-Frame (SF) synchronization between the L2/L3 software and PHY layer. On the other hand, it allows the subframe data transport through P7 interface.

Since part of this thesis project addresses the implementation of a FAPI-like interface for downlink data communication within NB-IoT, a brief description of downlink Subframe Procedures over Narrowband BCH and DLSCH are introduced in the following. For further details, refers to the official Small Cell Forum FAPI/nFAPI specification available at [9].

3.3.2.1 NB-IoT BCH Procedure

The Narrowband Broadcast Channel (NBCH) is used to transmit the Narrowband Master Information Block (MIB-NB) to the UE. The corresponding FAPI procedure is reported in Figure 3.4.

Whenever a BCH transmission is initiated, the L2/L3 software should provide the following information messages to the PHY layer:

- *DL_ CONFIG.request*: including a N-BCH PDU indicating all the necessary control information to enabling the Physical layer to transmit the MAC PDU carried by the following *TX.request* message.
- *TX.request*: carrying the MAC PDU in which the MIB-NB message is included.

Following the MIB-NB scheduled transmission already presented in subsubsection 2.5.4.3, the L2/L3 layer should provide the N-BCH PDU to the PHY only in SF #0 for each radio frame when $SFN mod 64 = 0$, i.e. every 640 ms. Then, the PHY manages the MIB-NB sub-blocks transmission and repetitions as reported in Figure 3.4. Both *DL_ CONFIG.request* and *TX.request* messages are transmitted over the same subframe at the start of MIB-NB period.

3.3.2.2 NB-IoT DLSCH Procedure

The DLSCH is the transport channel used for delivering UE-specific data and System Information from the eNodeB. The corresponding FAPI procedure is shown in Figure 3.5.

To transmit a DLSCH PDU the L2/L3 software must provide the following information messages to the PHY layer:

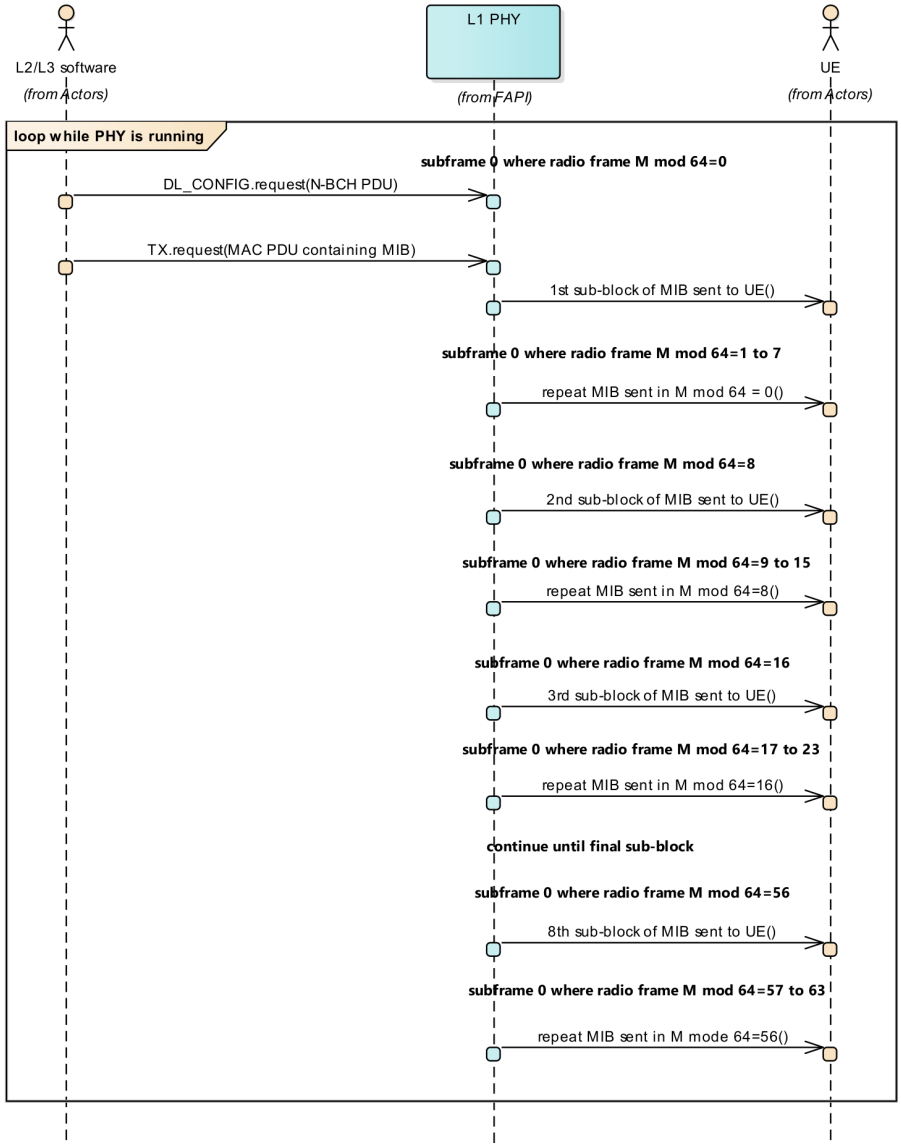

Figure 3.4: NB-IoT FAPI BCH procedure [9].

- *DL_ CONFIG.request*: including a NPDCCH DCI PDU which specifies control information instructing the PHY for both DCI and the following DL subframe transmission. In particular, it specifies the number of times the control channel is repeated and the delay between the final DCI transmission and start of NDLSCH transmission [9].

At the required DL subframe specified by the NPDCCH PDU, the L2/L3 software provides:

- *DL_ CONFIG.request*: including a NDLSCH PDU indicating to the PHY layer further control information for the following MAC PDU transmission.
- *TX.request:* carrying the MAC PDU containing the data to be delivered.

Whenever the HARQ process is enabled for the current MAC PDU, at the relevant UL subframe for ACK/NACK reception, the L2/L3 software provides:

- *UL_ CONFIG.request*: including a NULSCH PDU instructing the PHY for ACK/NACK reception.

Finally, PHY layer will return the ACK/NACK response through a *NB-HARQ.indication* message.

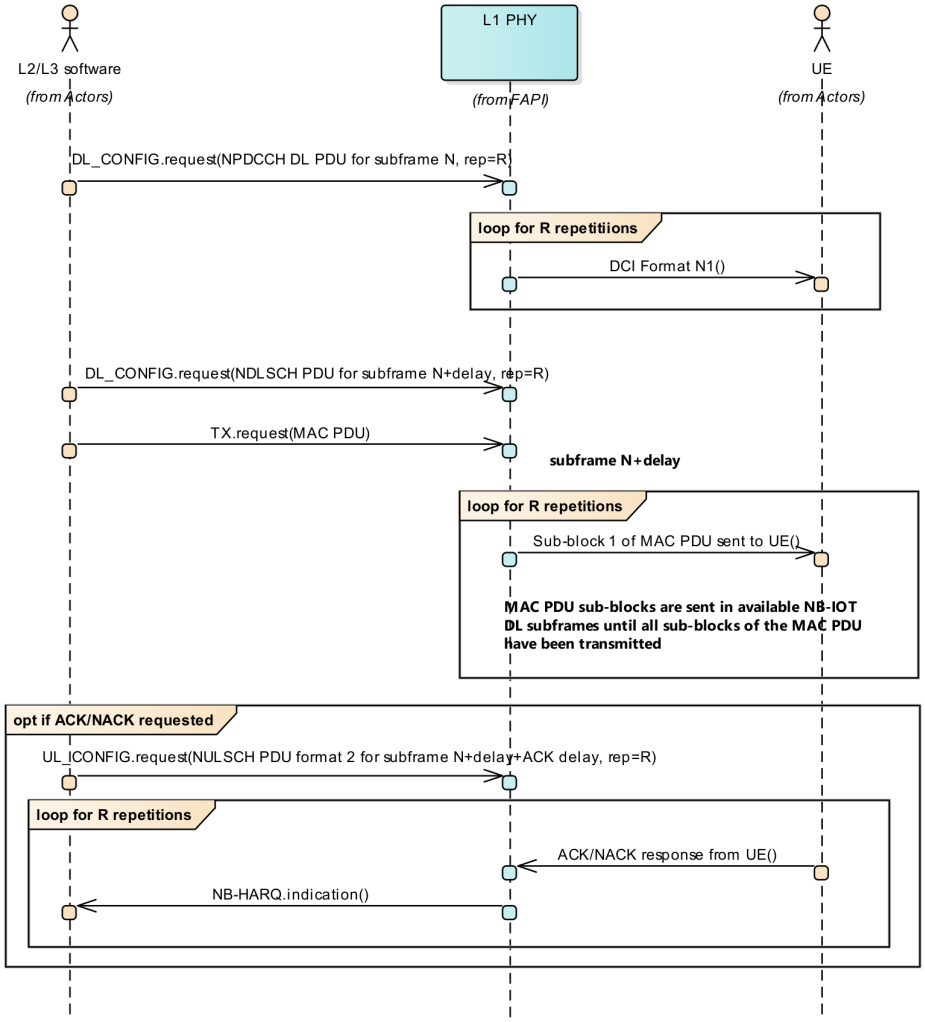

Figure 3.5: NB-IoT FAPI DLSCH procedure [9].

Chapter 4

Initial System Characterization

This chapter introduces the general RAN software architecture of OpenAirInterface, namely *openairinterface5G*. As a preliminary view, general features currently in force over the OAI platform are presented with details for the RRC protocol layer. Then, the software architecture is discussed starting from directories organization up to the description of main running threads and tasks. Afterwards, a top level view of RRC and PHY layer eNodeB code is given, with the former explained with much details since constitutes the starting point of this Master thesis project.

4.1 OpenAirInterface Features

As an open-source project, OpenAirInterface is a community effort and therefore, its implemented features are constantly evolving. As a reference starting point, Table 4.1 and Table 4.2 present a list of all consolidated LTE characteristics of OAI eNodeB's physical layer (PHY) and E-UTRAN. Moreover, Table 4.3 describes the initial status of the eNodeB's Radio Resource Control (RRC) layer before the start of Narrowband IoT (NB-IoT) protocol implementation.

Physical Layer (PHY): LTE Release 8.6 compliant, with a subset of Release 10	
Duplexing mode	FDD and TDD configurations 1(experimental) and 3
Bandwidth	5,10 and 20 MHz
Transmission mode	Stable: 1 (SISO) and 2,4,5,6 (MIMO 2x2) Experimental: 7 (MIMO 2x2)
Number of Antennas	2
CQI/PMI reporting	Aperiodic, Feedback mode 3-0 and 3-1
PRACH preamble	Format 0
DL Channels	PSS, SSS, PBCH, PCFICH, PHICH, PDCCH, PDSCH, PMCH
UL Channels	PRACH, PUSCH, PUCCH (format 1/1a/1b), SRS, DRS
HARQ support	UL and DL
Expected Throughput Downlink	• 5 MHz, 25 PRBS/ MCS 28 = 16 Mbit/s (measured with COTS UE Cat 3/4) • 10 MHz, 25 PRBS/MCS 28 = 34 Mbit/s (measured with COTS UE Cat 3/4)
Expected Throughput Uplink	• 5 MHz, 20 PRBs / MCS 20 = 9 Mbit/s (measured with COTS UE Cat 3/4) • 10 MHz, 45 PRBs / MCS 20 = 17 Mbit/s (measured with COTS UE Cat 3/4)

Table 4.1: OpenAirInterface eNodeB PHY features [5] [11].

E-UTRAN Protocol Stack: LTE Release 8.6 compliant, with a subset of Release 10	
Protocol Layers	MAC, RLC, PDCP and RRC
Broadcast and Multicast service (eMBMS)	PDCP, RLC, MAC, RRC and corresponding channels (MCH, MCCH, MTCH)
MAC Scheduler	Priority-based with dynamic MCS selection
Core Network Interfaces	S1AP and GTP-U
IP protocol support	IPv4 and IPv6
Integrity check and encryption using the Advanced Encryption Standard (AES) algorithm	
RRC measurement and measurement gap	
Fully reconfigurable protocol stack	

Table 4.2: OpenAirInterface E-UTRAN features [5] [11].

Radio Resource Control (RRC): LTE Release 8.6 compliant, with a subset of Release 10	
3GPP specification	TS 36.331 Rel.9.2.0
Radio Bearers	SRB0, SRB1 and SRB2
Connection Control procedures	• RRC Connection Establishment • Initial Security Activation (missed failure management) • RRC Connection Reconfiguration • RRC Connection Re-establishment (only rejection) • RRC Connection Release • Radio Resource Configuration • Radio Link Failure
System Information Messages	MIB, SIB1, SIB2, SIB3 and SIB13 (Rel.10)
RRC Messages	• RRCConnectionRequest (UE) • RRCConnectionSetup • RRCConnectionSetupComplete (UE) • RRCConnectionReject • ULInformationTransfer (UE) • DLInformationTransfer • UECapabilityEnquiry • UECapabilityInformation • SecurityModeCommand • SecurityModeComplete (UE) • RRCConnectionRelease • RRCConnectionReconfiguration • RRCConnectionReestablishmentRequest (UE) • RRCConnectionReestablishmentReject

Table 4.3: OpenAirInterface eNodeB and UE RRC features [5] [11].

4.2 Openairinterface5G directories

The *openairinterface5G* software is distributed on the OpenAirInterface GitLab repository available at [28]. The customization can be performed by editing different files at various levels of the OAI RAN code. Figure 4.1 shows the relevant working directories within the *openairinterface5G* project with also a brief description of their contents.

Both *openairinterface5G* and *openairCN*, make use of the open-source cross-platform CMake for managing the code build process. Therefore, the software updating results relatively simple by including the new file paths in the CMakeLists text file inside the cmake_targets directory. A top-level build script, *./build_ oai*, located in openairinterface5g/cmake_targets is used to start the lte-softmodem compilation. Currently, the tool is developed to build the eNodeB for different hardware platforms, 3GPP releases, standalone or with S1 interface, unitary simulations, and system simulation [28]. The compilation creates executables in openairinter-

```
openairInterface5G
├── cmake_targets → directory for building all the targets through
│                   a CMakeLists.txt file, contains "mother"
│                   build_oai script.
├── openair1 → basic DSP routines for implementing subset of
│   │          3GPP LTE specifications under x86 architecture,
│   │          channel simulation and PHY abstraction
│   │          software.
│   ├── PHY → contains all the signal processing related
│   │         for the physical layer (used in real-time) and
│   │         simulations.
│   ├── SCHED → schedules the differen PHY functions according
│   │           to the nodes role.
│   └── ...
├── openair2
│   ├── LAYER2 → 3GPP LTE Rel.10 RLC/MAC/PDCP implementation.
│   │   ├── PDCP
│   │   ├── RLC
│   │   └── MAC
│   ├── RRC
│   └── X2AP
├── openair3 → 3GPP LTE Rel.10 implementation of S1-C, S1-U
│              (S1AP, GTPV1-U, SCTP) and NAS interfaces for
│              both ENB and UE.
└── targets → hardware specific code (drivers, tools, etc),
    │         lte-softmodem, oaisim.
    ├── RT
    ├── PROJECTS → ".config" files for different band and hardware
    │              configuration.
    └── ...
```

Figure 4.1: *openairinterface5G* main directories and content description.

face5g/targets/bin which allows to run the eNodeB code with different hardware and band configurations from input files located in targets/PROJECTS/GENERIC-LTE-EPC/CONF/.

4.3 Software Architecture

As already mentioned, the OpenAirInterface RAN software foresees a full implementation of the LTE Access Stratum (AS) protocol (i.e. eNodeB and UE). The code is written in standard C and all the OAI protocol stack, including the PHY layer, runs entirely on a PC in a Linux-based operating system with low-latency kernel and Real Time Application Interface (RTAI) for addressing strict timing constraints operations. OpenAirInterface exploits a multi-thread parallel processing architecture as presented in Figure 4.2.

It is possible to distinguish between real-time (Worker, PRACH and Master) and non-real-time (RRC, S1AP, X2AP, GTP, Application) threads and each of them is possibly supplied with a queue defining a task. The eNodeB Application processes the configuration files, initializes the eNodeB modem and the protocol stack layers, and starts the OAI threads. The timing is managed by the Master thread that generates and maintains the System Frame Number (SFN) and the Subframe (SF) numerology allowing transmitter and receiver to operate in synchronized mode. The Worker thread constitutes the main routine for transmission and reception between the eNodeB and UE while the exchange of messages among the OAI protocol

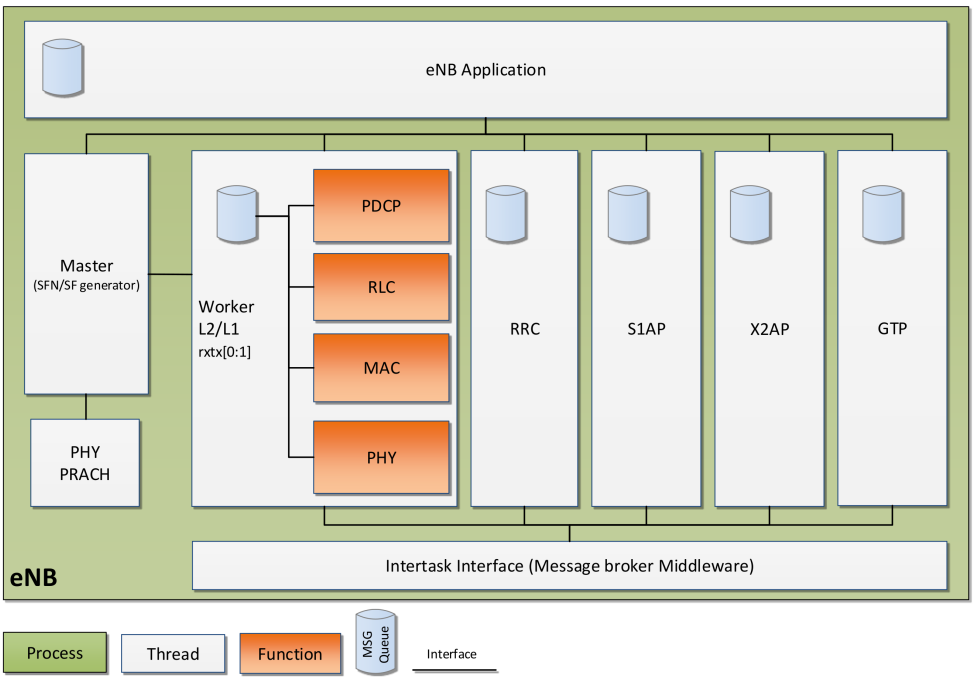

Figure 4.2: OpenAirInterface RAN software architecture.

stack takes place through an intermediary program module called Inter Task Interface (ITTI). However, the communication protocols used on OpenaAirInterface are sometimes heterogeneous and in some cases implemented through simple function calls among the different parts. In addition, the architectural framework presented in Figure 4.2 supports a flexible functional split over Ethernet Fronthaul (FH) for C-RAN architecture solutions on the OpenAirInterface platform. This allows a separation of the eNodeB baseband processing into small functional components placed either at Remote Radio Units (RRUs) or Baseband Units (BBUs) [36].

4.3.1 RRC eNodeB Software Architecture

This subsection describes the RRC layer code structure of OAI's eNodeB from a general view point. Among the possible ways to explain it, to ease the programming approach and make understandable the implemented procedures, a logical representation has been preferred as reported in Figure 4.3.

The RRC task, namely *rrc_enb_task*, is performed through a Finite State Machine (FSM) which, from a logical view point, plays two fundamental roles:

- Message Handler/Generator: processes the received UL messages over CCCH or DCCH logical channels and, based on their contents, may generate DL ones.
- ASN.1 UPER Encoder/Decoder: encode/decode messages for transmission/reception procedures in ASN.1 UPER standard encoding rules (see section 2.4).

The state machine is first initialized then triggered by different protocol entities and, consequently, may transmit data to lower layers through RRC specific interfaces. The latter ones, are not considered in this introductory chapter but a full description will be given in subsection 5.1.3. On the contrary, initialization and main procedure of RRC are discussed in the following.

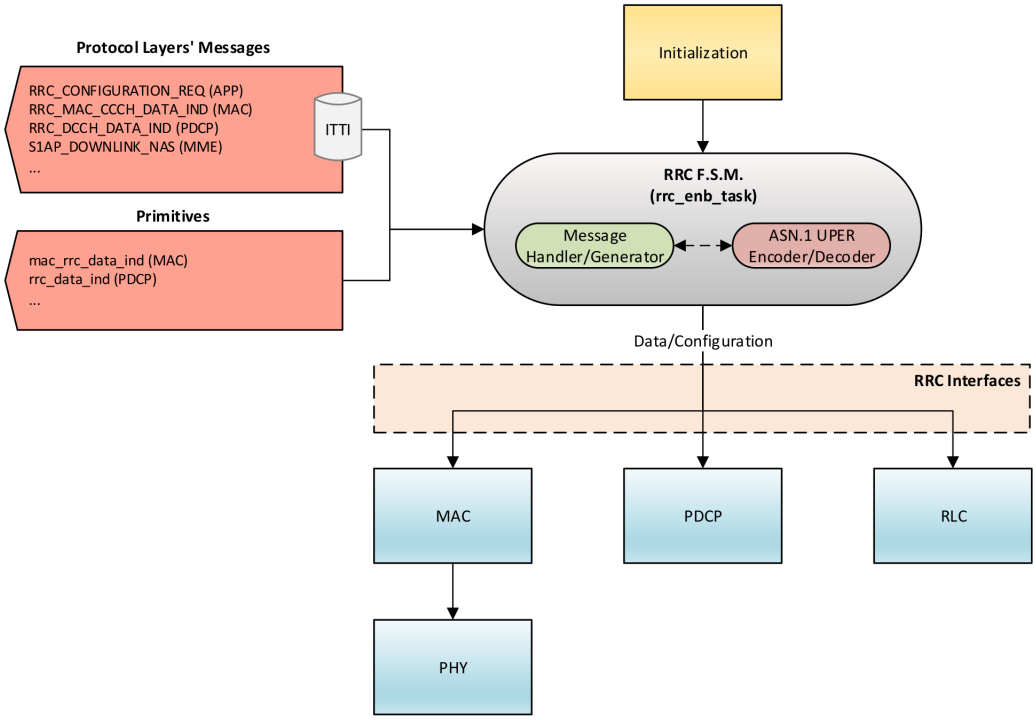

Figure 4.3: OpenAirInterface RRC logical representation.

4.3.1.1 Initialization

The initialization procedure for the OAI RRC layer is shown in Figure 4.4. The process is started by the "main" function in *lte-softmodem.c* file that also constitutes the entry point of the overall OpenAirInterface system. At first, the *rrc_enb_task* is created through the ITTI primitive called *create_tasks*, together with other tasks. Then, RRC parameters are initialized by means of the *l2_init* function that first configures the MAC and consequently the RRC. In particular, the *rrc_init_global_param* registers RRC with the lower RLC entity while *openair_rrc_top_init* allocates memory for the Radio Resource Control's instance, namely *eNB_RRC_INST*. The latter is the C-structure carrying all the relevant information for UE context management, Radio Bearer setup, System Information's buffer and layer configurations.

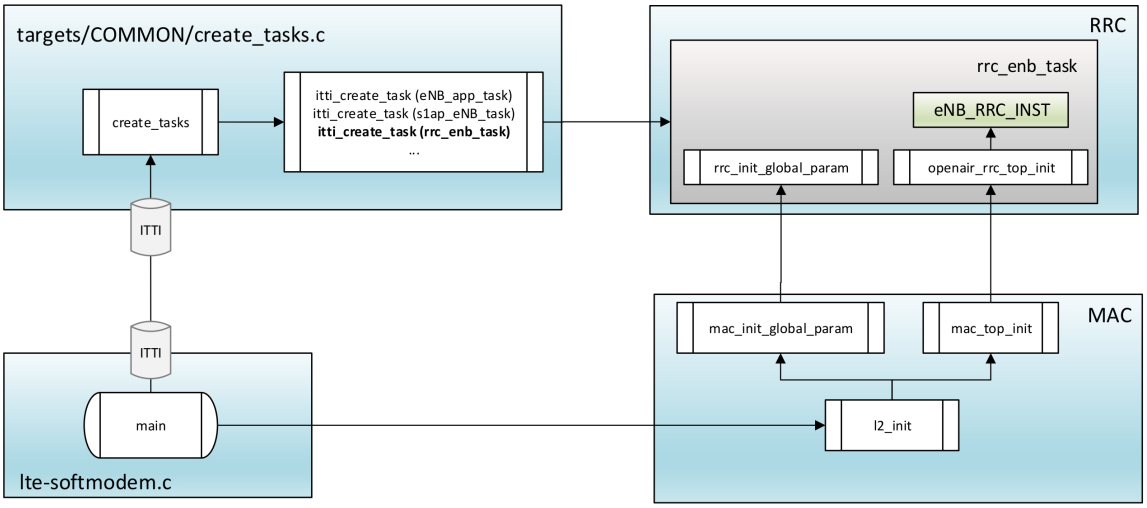

Figure 4.4: OpenAirInterface RRC initialization procedure.

4.3.1.2 RRC Main Procedures

The RRC layer is an event-based state machine with events generated by different protocol entities as ITTI messages or primitives. Figure 4.3 shows some examples. Based on input message type, different logical procedures can be initiated which involve various parts of OAI RRC code. For the sake of simplicity, it is decided to group them in three:

Configuration procedures: triggered by the eNodeB Application (APP) layer through the RRC_CONFIGURATION_REQ message.

Decoding/Encoding procedures: triggered by MAC or PDCP through RRC_MAC_CCCH_DATA_IND and RRC_DCCH_DATA_IND respectively.

S1 AP procedures: triggered by MME messages over the S1 AP protocol interface. For Instance, S1AP_DOWNLINK_NAS and S1AP_INITIAL_CONTEXT_SETUP_REQ. These procedures will be no more considered further since out of the scope of this thesis.

Configuration Procedure On OpenAirInterface, the RRC configuration aims to initialize the basic Signalling Radio Bearers (i.e. SRB0 and SRB1) buffers, create hash tables for storing the UE context and triggers System Information transmission.

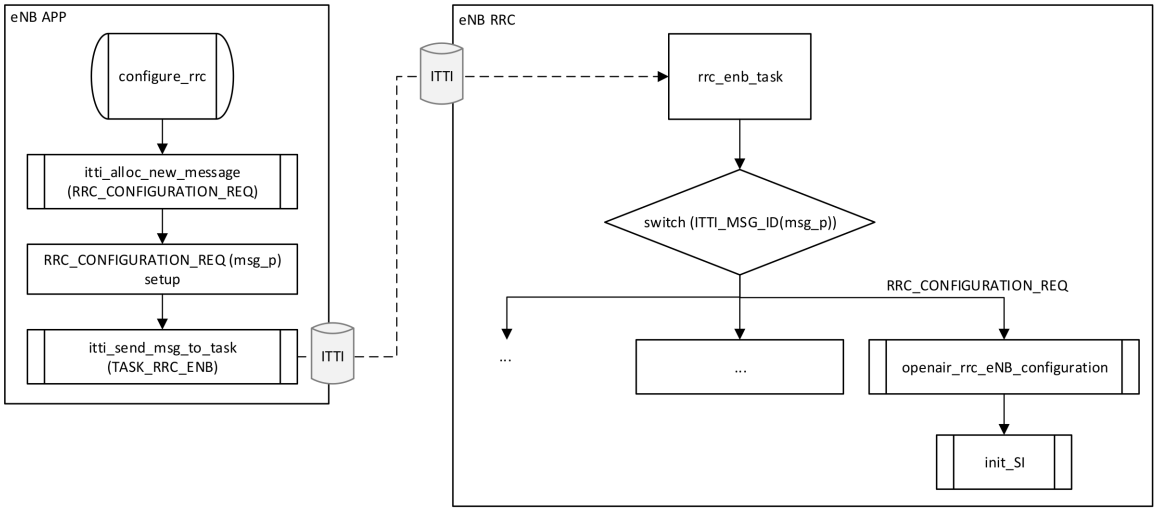

Figure 4.5: OpenAirInterface RRC configuration procedure.

As shown in Figure 4.5, the entire process is initiated by the *configure_rrc* function at eNodeB application layer that fills the RRC_CONFIGURATION_REQ message and sends it through ITTI interface. The RRC configuration is then managed by the *openair_rrc_eNB_configuration* primitive which sets major flags for Multicast service (eMBMS) and, among other things, recalls *init_SI*. The latter is the function that first generates and encodes System Information messages, then triggers PHY and MAC layer configuration.

Decoding/Encoding Procedure The OAI software provides a full set of routines for RRC message encoding/decoding, process and generation. These can be categorized as follow:

- rrc_eNB_decode_xxx: are the entry routines to decode and parsing uplink CCCH and DCCH messages. Examples are: *rrc_eNB_decode_ccch* and *rrc_eNB_decode_dcch*.
- rrc_eNB_process_xxx: set of functions in charge of processing the received RRC messages from the UE and extracts relevant Information Elements (IEs). Examples are: *rrc_eNB_process_RRCConnectionSetupComplete*, *rrc_eNB_process_RRCConnectionReconfigurationComplete* etc.

- rrc_eNB_generate_xxx: set of functions to generate and encode RRC messages before being transmitted through the downlink path. Examples are: *rrc_eNB_generate_RRCConnectionSetup*, *rrc_eNB_generate_RRCConnectionReject* etc.
- others: functions for UE context management, handover and measurement reporting within the RRC instance.

The general decoding/encoding procedure is presented in Figure 4.6. Green boxes represent the ASN.1 UPER Encoding/Decoding capabilities of the RRC FSM while the blue rectangle groups message handling and generating functions.

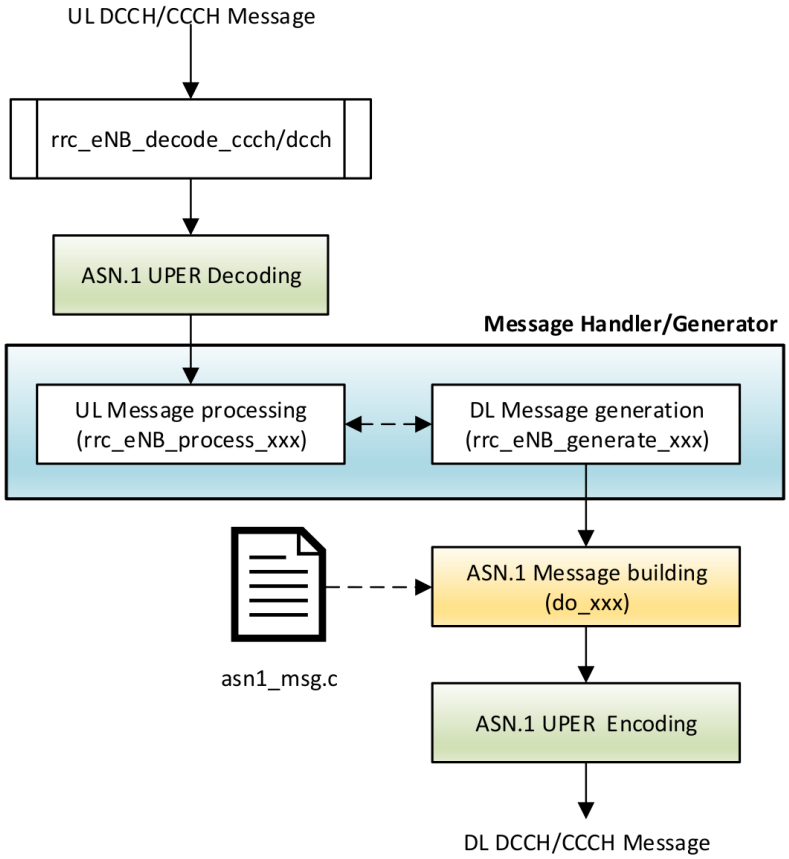

Figure 4.6: OpenAirInterface general decoding/encoding procedure.

A decoding procedure can be triggered either by MAC or PDCP whenever the eNodeB receives uplink messages over Command Control Channel (CCCH) or Dedicated Control Channel (DCCH) respectively. In this case, the usage of ITTI interface is optional since data transfer may also occur through some primitives, namely *mac_rrc_data_ind* and *rrc_data_ind* as reported in Figure 4.3. Depending on the logical channel, two specific functions are activated: *rrc_eNB_decode_ccch* and *rrc_eNB_decode_dcch*. These are used by RRC to start the ASN.1 decoding process following UPER rules and, based on message content, to invoke the suitable processing function. If a new downlink message needs to be generated, a set of message-specific routines, commonly referred as do_xxx, build the message contents based on ASN.1 notation (yellow box). These are located under the RRC/LITE/MESSAGES directory in the *asn1_msg.c* source file. Finally, before transmission the message is encoded with the same UPER rules as before.

4.3.1.3 RRC Message Sequence

For a comprehensive view, this subsection briefly describes the RRC message sequence of OpenAirInterface shown in Figure 4.7. For each message, transport or logical channel is indicated with corresponding radio bearers and LCID. Moreover, relevant carried information is reported below corresponding transmissions and important actions on both UE and eNodeB side are highlighted.

As a first step, eNodeB broadcasts System Information (SI) messages including SIB1, MIB and an SI-Message. The latter delivers only the mandatory SIB2, SIB3 and optionally SIB13 since other SIBs types are not supported by OAI (see Table 4.3). After the SI acquisition, the UE moves in RRC_IDLE state and starts the Random Access (RA) procedure. In this phase, MAC layer messages (Preamble and RA Response) are exchanged before the first RRC Connection Request message is received by the eNodeB. After UE Context creation, the eNodeB generate following RRC Connection Setup message through its specific *do_RRCConnectionSetup* function. Then, the UE changes in RRC_CONNECTED state and activates SRB1 with configuration carried in the setup message. The RA procedure concludes with the reception of an RRC Connection Setup Complete message by the eNodeB which triggers the UE context update over the RRC instance. Afterwards, the context is managed over the S1 AP protocol by the MME which triggers the RRC Security Activation procedure and the subsequent UE capability acquisition. Like in the previous case, any RRC message is generated starting from its specific ASN.1 function, i.e. "do_xxx". The last part of Figure 4.7 shows the "default" RRC Connection Reconfiguration procedure that brings to SRB2 and DRB1 setup, PHY and MAC's configurations and NAS information exchanged between the eNodeB and UE. A successful reconfiguration is followed by a UE context status update in RRC_RECONFIGURED and a setup of bearer flags within the RRC instance. Finally, a "dedicated" RRC Connection Reconfiguration procedure might be initiated by the S1 AP protocol to configure further DRBs and dedicated NAS information for the UE.

4.3.2 PHY eNodeB Software Architecture

This subsection gives a brief description of the eNodeB PHY's software architecture of OpenAirInterface. This will result useful as a preliminary introduction for the work presented in subsection 5.2.3 related to the design of a new Physical layer for Narrowband-IoT based on Functional Application Platform Interface (FAPI) standard.

The OAI PHY layer provides the LTE functionalities for encoding, modulation, channel estimation and reference signals. It implements all the uplink and downlink physical channels and oversees the transmission (TX) and reception (RX) procedures as well as the power control algorithm. In OpenAirInterface, physical layer TX and RX procedures run either over one single thread, called "eNB_thread_single" or two identical parallel threads, namely "eNB_thread_rxtx". In both cases, the *rxtx* is the fundamental function for the activation of physical layer mechanisms for transmission and reception. The *rxtx* flow chart is reported in Figure 4.8 and described below.

At first, *rxtx* initiates a common reception procedure on current subframe (n) by awakening the single RX Random-Access thread over the PRACH channel, namely *do_prach*. Then, the *phy_procedure_eNB_uespec_RX* function activates UE specific reception mechanisms to process messages and signals received over PUCCH and PUSCH channels. Finally, the transmission procedure for subframe n+4 is initiated and the *phy_procedures_eNB_TX* function is triggered. This will manage the encoding, modulation and scrambling over the PDSCH, PDCCH, PHICH and PCFICH channel as well as the transmission over RF front end.

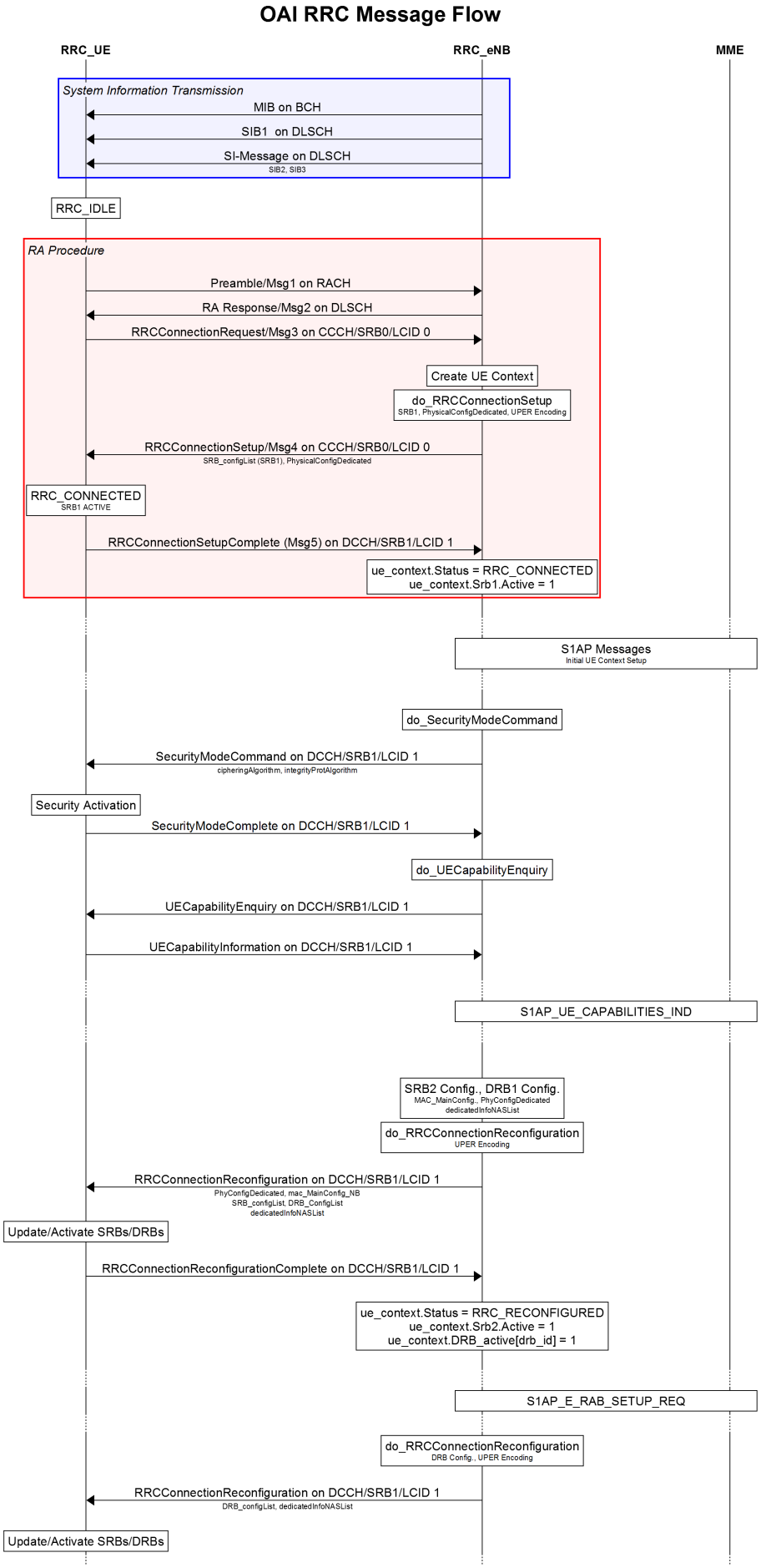

Figure 4.7: OpenAirInterface RRC message flow.

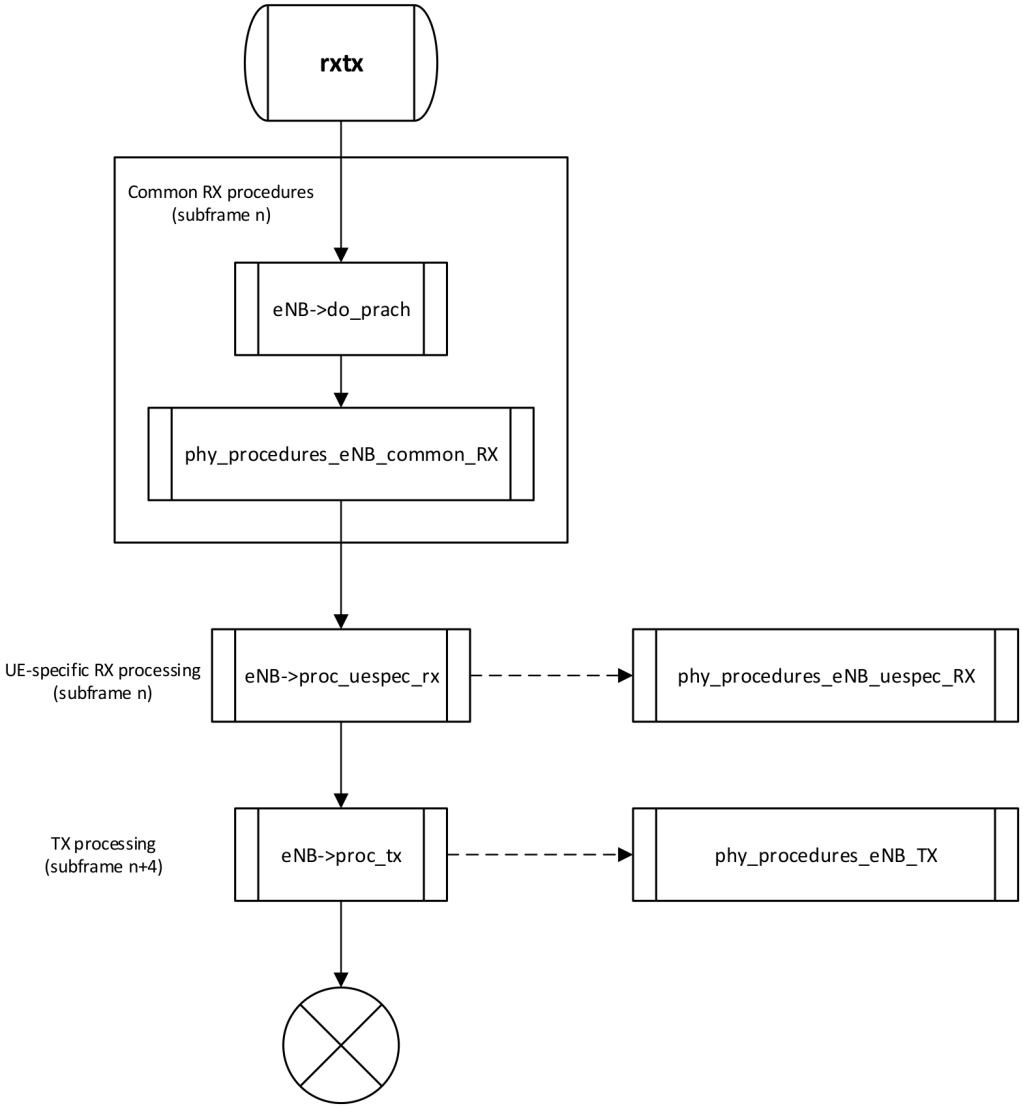

Figure 4.8: OpenAirInterface *rxtx* flow chart.

Chapter 5

NB-IoT Software Implementation on OpenAirInterface

This chapter describes the contribution of this master thesis to address the implementation of NB-IoT protocol stack over OpenAirInterface platform. At first, Radio Resource Control layer's improvements are discussed by outlining the new RRC Finite State Machine, a different SRBs management, RRC interfaces and System Information scheduling. Afterwards, the implementation of FAPI/nFAPI standard for the MAC-PHY interface is described by introducing a new module (IF-Module) and then focusing to the PHY procedures developed to support this new approach. Overall, only those parts of the OpenAirInterface code related to eNodeB have been considered during this master project, in line with the OSA purposes to introduce a clear distinction between eNodeB and UE side implementation.

5.1 NB-IoT RRC Layer Implementation

As discussed in section 2.5, the control-plane of NB-IoT introduces some major changes with respect to legacy LTE systems. It reduces the RRC functionalities supported, proposes a new Signalling Radio Bearer (i.e. SRB1bis), foresees a different SI-Message scheduling as well as different Information Elements (IEs) carried by RRC messages.

This section is organized as follow. First, the RRC instance and the UE context within the OpenAirInterface software are presented. Then, the RRC Finite State Machine is described as well as its protocol interfaces. Finally, the new Signalling Radio Bearers management and System Information scheduling are discussed.

5.1.1 RRC Instance and UE Context

First step of the project was the definition of a new eNodeB RRC instance, namely *eNB_RRC_INST_NB*, targeted for the NB-IoT technology. This has been declared in a new *defs_nb_iot.h* header file in the openair2/RRC/LITE directory to allow a first separation between the NB-IoT RRC module and the LTE one. Figure 5.1 shows the RRC instance taken from the OpenAirInterface software.

eNB_RRC_INST_NB is a C-structure grouping the current carried data over System Information Blocks and Signalling Radio Bearer but it also accommodates other RAN, PLMN parameters and variables as well as configurations got from the eNodeB Application Layer. However, because the reduced set of RRC functionalities for the NB-IoT standard, those parameters related to Multimedia Broadcasting services, localization, measurement reports, handover and Contention Based Access (CBA) of previous implementation have been removed. Moreover, in this preliminary phase it was observed that some parts of the already existing

```
typedef struct eNB_RRC_INST_NB_s {

  //carried data
  rrc_eNB_carrier_data_NB_t          carrier[MAX_NUM_CCs];

  uid_allocator_NB_t                    uid_allocator;

  //tree key search by rnti
  RB_HEAD(rrc_ue_tree_NB_s, rrc_eNB_ue_context_NB_s)     rrc_ue_head;
  uint8_t                              Nb_ue;
  hash_table_t                         *initial_id2_s1ap_ids;
  hash_table_t                         *s1ap_id2_s1ap_ids;

  //RRC configuration
  RrcConfigurationReq configuration;

  // other PLMN parameters
  int mcc; // Mobile country code
  int mnc; // Mobile network code
  int mnc_digit_length; // number of mnc digits

  // other RAN parameters
  int srb1_timer_poll_retransmit;
  int srb1_max_retx_threshold;
  int srb1_timer_reordering;
  int srb1_timer_status_prohibit;

} eNB_RRC_INST_NB;
```

Figure 5.1: RRC instance for NB-IoT within OpenAirInterface eNodeB software.

code were no more used since belonging to old OAI features, therefore these too have neither been considered in the development of the new RRC instance.

The *defs_nb_iot.h* file contains also two data structures for UE context management within the OpenAirInterface eNodeB. They are called *rrc_eNB_ue_context_NB_t* and *eNB_RRC_UE_NB_t* respectively and are shown in Figure 5.2. As in previous case, major changes on their definition were related to the absence of measurement reports capability, the missing SRB2 and the introduction of SRB1bis whose implementation details will be given in subsection 5.1.4.

```
typedef struct rrc_eNB_ue_context_NB_s {

  /* Tree related data */
  RB_ENTRY(rrc_eNB_ue_context_NB_s) entries;

  rnti_t          ue_id_rnti;

  ue_uid_t        local_uid;

  struct eNB_RRC_UE_NB_s    ue_context;

} rrc_eNB_ue_context_NB_t;
```

```
typedef struct eNB_RRC_UE_NB_s {
  uint8_t                                 primaryCC_id;

  SRB_ToAddModList_NB_r13_t*                    SRB_configList;
  SRB_ToAddModList_NB_r13_t*                    SRB_configList2[...];
  DRB_ToAddModList_NB_r13_t*                    DRB_configList;
  DRB_ToAddModList_NB_r13_t*                    DRB_configList2[...];
  uint8_t                                       DRB_active[2];

  struct PhysicalConfigDedicated_NB_r13*    physicalConfigDedicated_NB;
  MAC_MainConfig_NB_r13_t*                  mac_MainConfig_NB;

  SRB_INFO_NB                                  SI;
  SRB_INFO_NB                                  Srb0;
  SRB_INFO_TABLE_ENTRY_NB                      Srb1;
  SRB_INFO_TABLE_ENTRY_NB                      Srb1bis;

  uint8_t kenb[32];

  e_CipheringAlgorithm_r12                              ciphering_algorithm;
  e_SecurityAlgorithmConfig__integrityProtAlgorithm integrity_algorithm;

  uint8_t                                 Status;
  rnti_t                                  rnti;
  uint64_t                                random_ue_identity;
    ...                                     ...                          ...
```

Figure 5.2: UE context data structures from OpenAirInterface code.

5.1.2 NB-IoT RRC State Machine

This section presents the Narrowband-IoT version of the RRC Finite State Machine, namely *rrc_enb_task_NB*, resuming from what introduced in subsection 4.3.1. The RRC task and associated functions for message processing and generation have been defined in the new *rrc_eNB_nb_iot.c* file in openair2/RRC/LITE working directory.

In this phase of the project no changes in the already existing layer's working principles were needed. Instead, it resulted necessary to introduce all Information Elements, message types and data structures defined for NB-IoT. Moreover, those functions related to measurement reports, handover and multicast services have been deleted for the NB-IoT software since not implemented by the standard. The *rrc_enb_task_NB* logical scheme is shown in Figure 5.3 and its major functionalities are described here below.

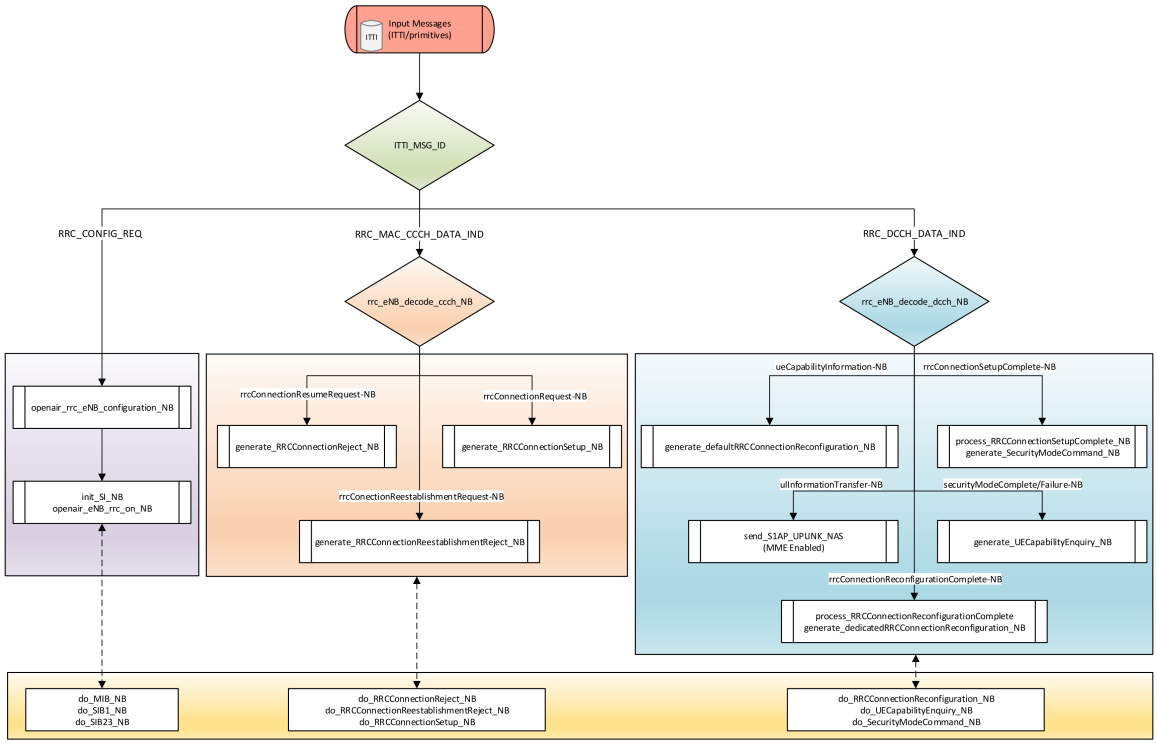

Figure 5.3: OpenAirInterface *rrc_enb_task_NB*.

As discussed in subsection 4.3.1, the RRC's state machine is triggered by inputs received from other protocol layers by means of ITTI interface or primitives. The task is performed through a "while" loop waiting for an uplink message that, once received, is processed by a "switch-case" based on its identification (ITTI_MSG_ID). This triggers the corresponding RRC procedure involving different functions. For instance, an input message from eNodeB Application layer (APP) triggers the RRC configuration procedure (purple box) by activating *openair_rrc_eNB_configuration_NB* and consequently initializes System Information and Radio Bearer buffers (see subsubsection 4.3.1.2). More important, an uplink message over CCCH from MAC layer or over DCCH from PDCP, triggers *rrc_eNB_decode_ccch_NB* or *rrc_eNB_decode_dcch_NB* respectively. Decoding functions apply on their turn a "switch-case" statement (orange or blue box) to distinguish among the received RRC's CCCH/DCCH information type and call the corresponding primitives for processing the NB-IoT messages and eventually generate new ones. In the latter case, a full set of routines for building message contents based on ASN.1 notation has been redefined for NB-IoT (yellow box). For instance, *do_RRCConnectionSetup_NB* or *do_RRCConnectionReject_NB*.
During the definition of the NB-IoT RRC's state machine, it was noticed that there were no primitives allowing the generation of Master Information Block message based on ASN.1 notation. This was because in OpenAirInterface LTE software the broadcasting transmission over PBCH channel was always managed directly at PHY layer bypassing RRC. Therefore, in *asn1_msg_nb_iot.c* file was introduced the new *do_MIB_NB* function for Narrowband Master Information Block's generation which is triggered by the *init_SI_NB* primitive at con-

figuration time. Finally, common functions to UE and eNodeB, like *openair_rrc_top_init_eNB_NB*, have been redefined to address initialization and memory allocation for eNodeB side only.

5.1.3 NB-IoT RRC Interfaces

The OpenAirInterface platform foresees a relatively large set of primitives for the exchange of user- and control-plane information among different Layer 2 (L2) protocol entities. In particular, given its role, the Radio Resource Control is the layer with the largest amount, capable of direct communication with PDCP, RLC and MAC. Therefore, one important step during this Master thesis involves the redefinition of the OAI interfaces between the new NB-IoT RRC module and the other protocol layers. Like in previous cases, there was no worth on changing the internal logic of each function but, for each of them, the feasibility and the compliance with the NB-IoT standard's characteristics were considered.

Figure 5.4 shows the full set of interfaces for OpenAirInterface L2 protocols. In the following, the ones related to RRC will be described, however, it should be clear that many are correlated with others from different layers. Thus, some further investigations have been conducted also in PDCP and RLC layers.

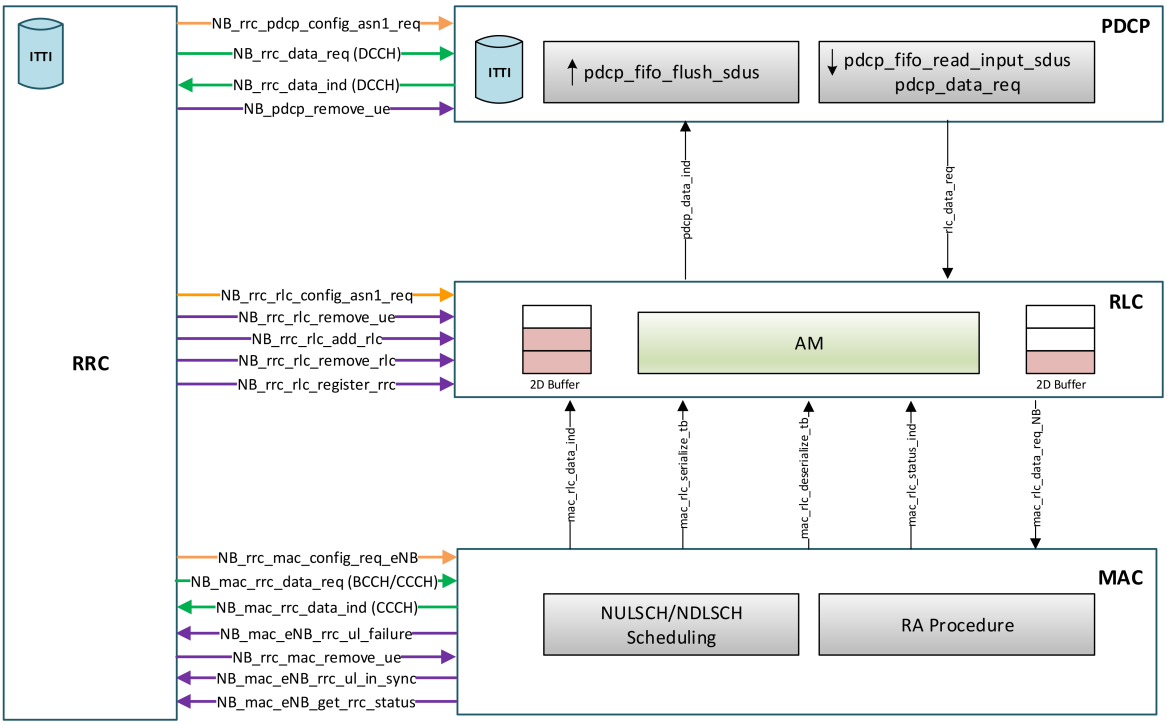

Figure 5.4: OpenAirInterface RRC interfaces for NB-IoT.

As shown in Figure 5.4, on OpenAirInterface the full set of RRC primitives for NB-IoT can be grouped in three major categories:

- Configuration Interfaces (in orange)
- Data Interfaces (in green)
- Operational Interfaces (in purple)

5.1.3.1 Configuration Interfaces

Configuration Interfaces is the set of primitives used by the RRC layer to configure other L2 protocol entities, i.e. PDCP, RLC and MAC. They have almost the common naming part "NB_xxx_config_asn1_req" to indicate that are NB-IoT dedicated, their logic direction ("_req") and the compliance with the transport of ASN.1 variables. In the following, a brief description of the Configuration Interfaces is given, highlighting the major changes introduced for NB-IoT implementation purposes.

NB_rrc_pdcp_config_asn1_req: this primitive allows the Radio Resource Control to configure or reconfigure the PDCP layer by establish, modify or release SRBs and DRBs carried in the input lists. As already mentioned, NB-IoT introduces the usage of the Signalling Radio Bearer 1bis bypassing PDCP layer before security activation. Therefore, this primitive allows the PDCP's configuration only for DRBs and SRB1 by imposing initial checking on Logical Channel Identity (LCID). At the same time, it was noticed that OpenAirInterface not managed the case in which RRC Security Activation procedure fails. In this case, as reported in [10] ch. 5.3.4.3 , the UE applies neither integrity protection nor ciphering but continues using the configuration prior reception of the Security Mode Command, i.e. the UE does not activate security. Therefore, it was necessary to introduce a further value for the *security_mode* variable in input to the primitive (i.e. *security_mode = -1*) to deactivate ciphering or integrity protection algorithms at PDCP configuration time. This allows to skip any usage of security for PDCP PDU through Data Interfaces, like *NB_pdcp_data_ind* and *NB_rlc_data_req*, in case of Security Activation Failure. An example of Security Activation Failure is reported in Figure 5.6 as will be discussed later.

NB_rrc_rlc_config_asn1_req: this primitive configures or removes RLC entities associated to SRBs and DRBs carried in the input lists according to their LCID. As mentioned in subsection 2.5.2, NB-IoT allows RLC layer to work only in Acknowledged Mode (RLC-AM) over DCCH and DTCH, while Transparent Mode (RLC-TM) is used over PCCH, BCCH and CCCH as in legacy LTE. Thus, it was necessary to remove all conditions for setup of RLC entities in Unacknowledged Mode (RLC-UM) which were mainly adopted for eMBMS services over the no more used Multicast Traffic Channel (MTCH).

NB_rrc_mac_config_req_eNB: this primitive is used whenever RRC initiates a configuration or reconfiguration procedure. It instructs the MAC layer about common radio resource and logical channel configurations, number of antenna ports utilized, cyclic prefix type and others. Moreover, at initialization time it triggers specific functions for directly configure the PHY layer. For instance, *NB_phy_config_mib_eNB* and *NB_phy_config_sib2_eNB* (see subsection 5.2.1). Different from previous implementation, the primitive's suffix "_eNB" indicates that the new interface addresses only the eNodeB side of the OpenAirInterface platform.

5.1.3.2 Data Interfaces

Data Interfaces includes those primitives aiming for data transport within the OpenAirInterface protocol stack. They are indicated with the two common suffixes "_req" and "_ind" to specify the direction "from" or "to" the RRC layer. From Figure 5.4, it can be observed that there is no direct data communication between RLC and RRC but there is between RRC and MAC or PDCP. This is because common control information over CCCH, BCCH and PCCH goes through PDCP and RLC transparently and from MAC directly reaches RRC. On the contrary, user-specific control data over DCCH needs PDCP and RLC entities establishment.

NB_rrc_data_req: this primitive is used whenever a default or dedicated RRC Connection Reconfiguration procedure is initiated or RRC messages as Security Mode Command or UE Capability Enquiry are transmitted over DCCH . It allows to specify the

PDCP Transmission Mode over DRBs and SRBs that, in case of SRB1bis, should be set to "PDCP_TRANSMISSION_MODE_TRANSPARENT" (i.e. bypass PDCP) based on previous considerations.

NB_rrc_data_ind: this primitive is triggered directly by the PDCP layer whenever data over SRBs should be transmitted to RRC. It initiates the decoding message procedure over DCCH channel by generating the RRC_DCCH_DATA_IND message in the case of ITTI communication interface or directly recalling the *rrc_eNB_decode_dcch* function.

NB_mac_rrc_data_req_eNB: this data interface is mostly used by the MAC scheduler whenever a transmission over BCCH or CCCH occurs. It is adopted to set the SDU length and fill the corresponding channel buffer with the PDU. In OpenAirInterface LTE software, this interface directly implements System Information messages scheduling by checking the current frame's and subframe's number. However, as discussed in subsubsection 2.5.4.3, the SI transmission for NB-IoT results completely different and rather complex from the LTE one, therefore, it was decided to implement auxiliary functions for the SI scheduling decision within this primitive. For instance, *is_SIB1_NB* and *is_SIB23_NB* are used to identify frames for SIB1-NB and SIB23-NB transmission respectively. In the definition of *NB_mac_rrc_data_req_eNB* two facts have been considered. First, the introduction of a FAPI standard in the MAC-PHY interface which assumes that repetitions over multiple subframes are managed at PHY layer (see subsection 3.3.2). Second, the lack of an NB-IoT scheduler which was not completed during the time of this Master thesis project. Therefore, it was only possible to imagine the behaviour of the MAC layer jointly with FAPI specifications and consequently design this interface doing some assumptions. These aspects will be resumed during this and the next sections.

NB_mac_rrc_data_ind: this primitive is used for data reception over CCCH, for instance, whenever an RRC Connection Request (i.e. Msg3) message is received. It initiates the decoding procedure over SRB0 by generating the RRC_MAC_CCCH_DATA_IND message or directly triggering the *rrc_eNB_decode_ccch* function.

5.1.3.3 Operational Interfaces

Operational Interfaces refer to those primitives aiming for UE context management, synchronization and registration among the different layer entities within the OAI platform. Major changes only concerned the introduction of NB-IoT Information Elements while the internal logic was left the same as the already existing one. For this reason, no description of this functions will be given in this chapter but refer directly to the OpenAirInterface code available at [37].

5.1.4 NB-IoT Signalling Radio Bearers Management

The NB-IoT radio interface introduces a new Signalling Radio Bearer over the DCCH logical channel, called SRB1bis, used for the transmission of RRC and NAS messages prior the security activation. It is established implicitly with SRB1 adopting same configurations but different LCID (i.e. 3) and, most relevant, without PDCP entity. Moreover, in NB-IoT the low-priority Radio Bearer SRB2 is not applicable so that RRC messages carrying dedicated NAS information are only transmitted over SRB1 also after any reconfiguration procedures. Given this, the new RRC module required a different SRBs management for the NB-IoT case.

In this section, the Radio Bearer utilization over OpenAirInterface platform is first introduced, then, the new approach for SRBs will be discussed through a description of an RRC message flow for NB-IoT.

5.1.4.1 Signalling Radio Bearers Lists

On OpenAirInterface, Radio Bearers are handled in the *eNB_RRC_UE_NB_s* data structure within the RRC UE's context (see Figure 5.2). In particular, they are organized in two kinds of lists: one for Data and the other for Signalling Radio Bearers, namely *DRB_configList* and *SRB_configList* respectively. For each kind, two lists are then defined as reported in Figure 5.5.

```
SRB_ToAddModList_NB_r13_t*          SRB_configList;
SRB_ToAddModList_NB_r13_t*          SRB_configList2[RRC_TRANSACTION_IDENTIFIER_NUMBER];
DRB_ToAddModList_NB_r13_t*          DRB_configList;
DRB_ToAddModList_NB_r13_t*          DRB_configList2[RRC_TRANSACTION_IDENTIFIER_NUMBER];
```

Figure 5.5: OpenAirInterface Signalling Radio Bearers lists.

This implementation choice is linked to the Transaction Identifier (TI) concept. Each couple of RRC message, transmitted and received by the eNodeB, together with the RRC message type define a Transaction. To correlate any RRC procedure with the corresponding UE in RRC_CONNECTED state, the eNodeB assign a Transaction Identifier. This allows to validate and identify the UE RRC's response message also whenever more than one of the same type are received by the eNodeB at the same time. As a result, the "_configList2" is used to store the Radio Bearer configuration related to a given kind of RRC message for different transactions (up to 3) while the "_configList" refers only to the current RRC message managed by the eNodeB. Therefore, whenever a new RRC message is transmitted in downlink, the first list (_configList) is used for setup the new SRB/DRB's configurations while the second (_configList2) is used to store and retrieve them once the RRC message is received from the UE.

5.1.4.2 RRC Message Flow for NB-IoT

The introduction of SRB1bis over the new RRC module resulted more complicated than planned on. 3GPP specifications [10] were not clear about the "implicit" establishment of SRB1bis with SRB1 and which was the message that should mark the transition from one to the other. After further investigations, [38] stated that SRB1bis ("no PDCP" or PDCP-TM) could be used for carrying Security Mode Command (SMC) and Security Mode Failure messages and, after receiving the SMC and performing security activation, the UE shall use the SRB1 for Security Mode Complete.

As regard for the Bearers lists, the same approach previously described in Figure 5.5 was maintained. However, the new NB-IoT data type, SRB_ToAddMod_NB_r13, does not include the legacy *srb_identity* Information Element to distinguish among different Signalling Radio Bearers. Therefore, the LCID was the only way to differentiate SRB1bis from SRB1 that is when "PDCP-TM/no PDCP" or "full/normal PDCP" is considered respectively. Over the OpenAirInterface software the following identity was applied:

- SRB1bis: LCID = DCCH0 (3) → PDCP-TM/no PDCP
- SRB1: LCID = DCCH1 (1) → full/normal PDCP

Figure 5.6 shows the message sequence chart for the new Narrowband-IoT RRC module. Both RRC messages and relevant RRC interfaces (in dashed blue lines) are reported. For each message, logical channel type, channel identity and corresponding SRB are specified as well as relevant carried information. Moreover, important elements for Bearers management are highlighted in red and relevant procedures are point out through coloured boxes (green, yellow or blue). A description of Figure 5.6 is then given below.

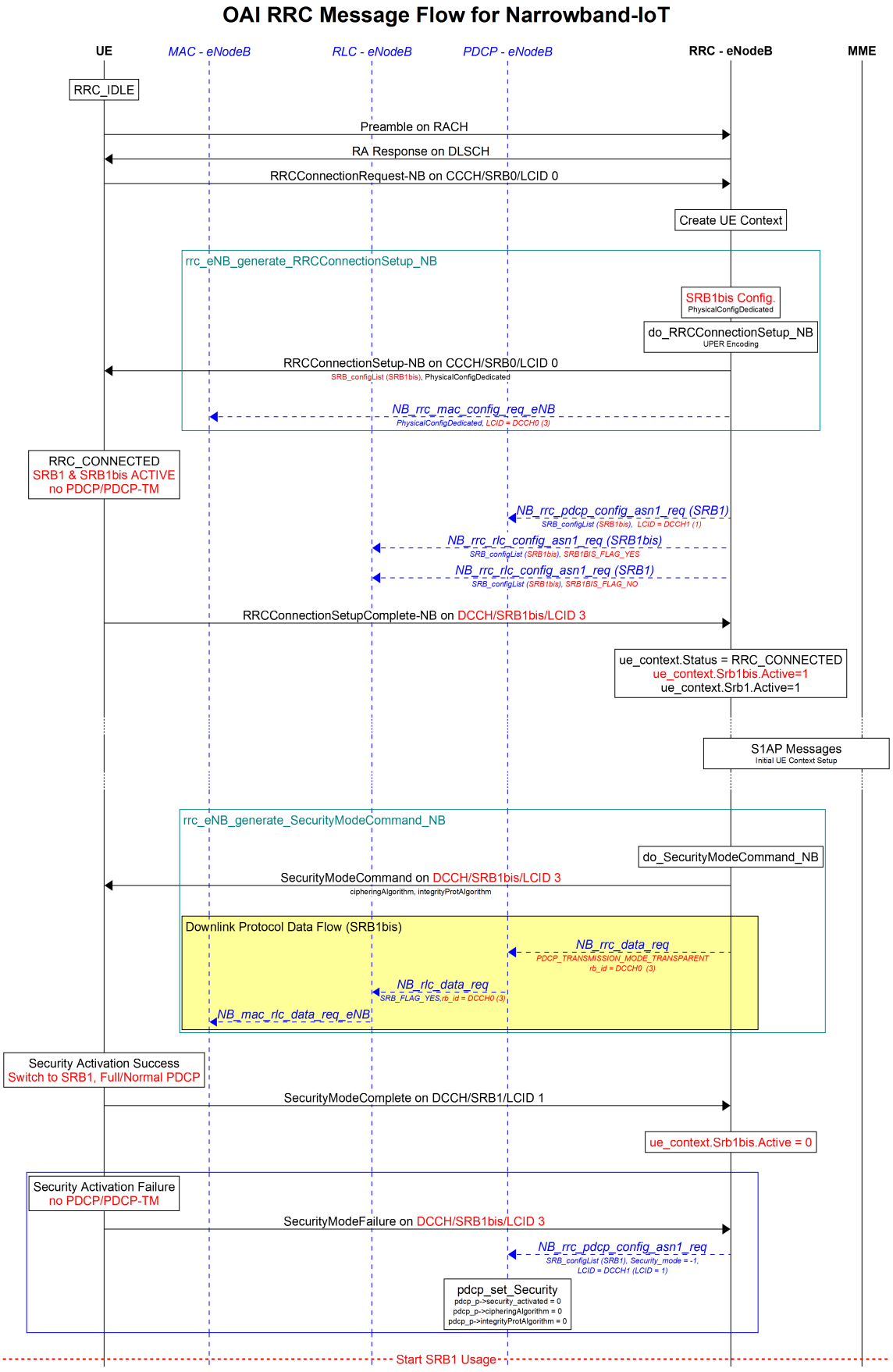

Figure 5.6: OpenAirInterface RRC Message Flow for Narrowband-IoT, part 1.

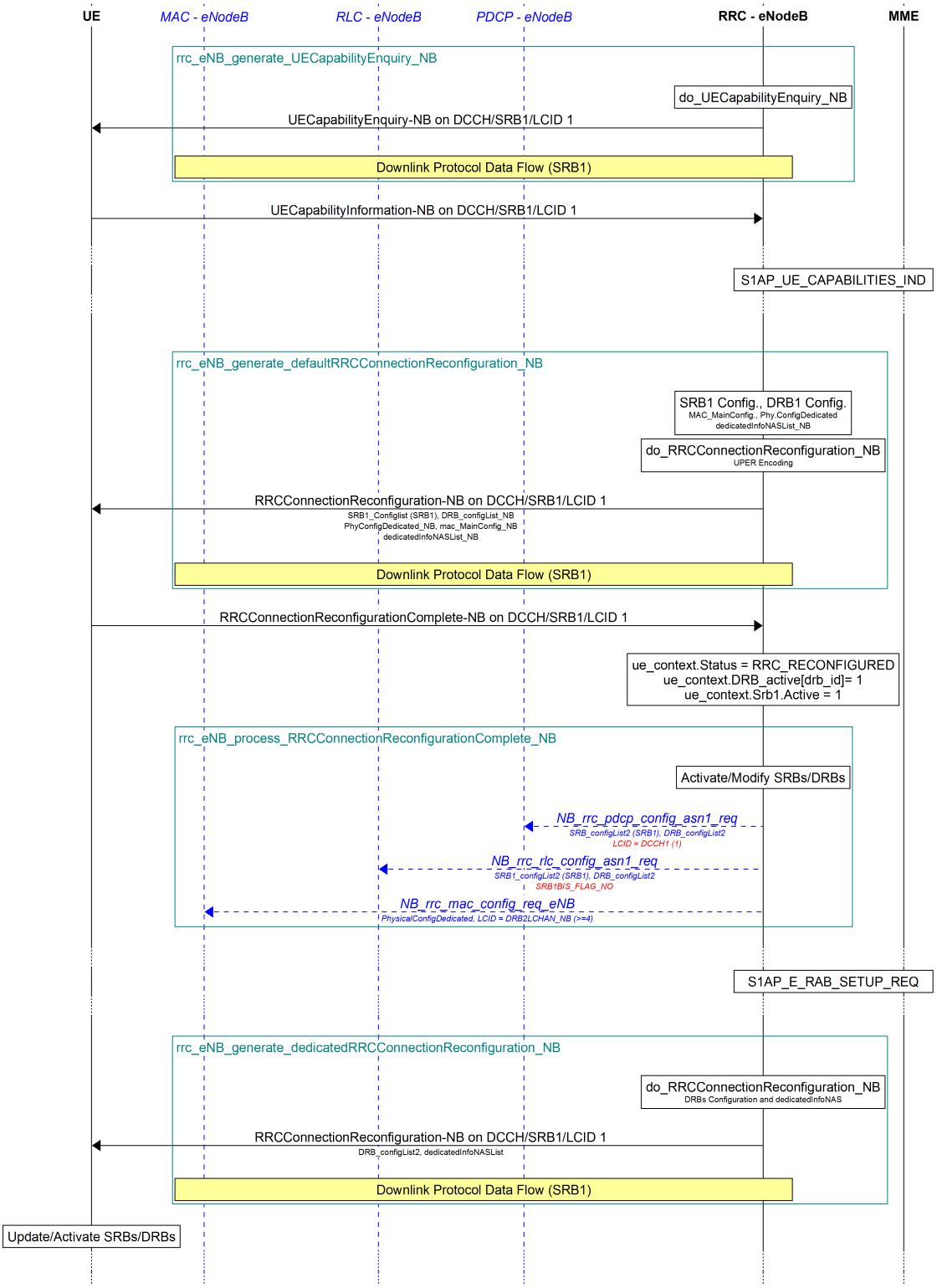

Figure 5.6: OpenAirInterface RRC Message Flow for Narrowband-IoT, part 2.

The first relevant implementation aspect concerns SRB1bis and SRB1 setup. As already mention, the two Bearers have exactly same configuration, therefore, during *rrc_ eNB_ generate_ RRCConnectionSetup_ NB* procedure only one of the two, SRB1bis, is configured and transmitted inside the *SRB_ configList* data structure over RRC Connection Setup. Once received the message, the UE updates its RRC status and configures its protocol layer entities to go through SRB1bis bypassing PDCP ("no PDCP/PDCP-TM"). To be noticed that at the same time UE is also configured for reception over SRB1 since implicitly established together with

SRB1bis.
Meantime, the eNodeB instructs PDCP and RLC layers through the corresponding interfaces. In particular, it was decided to immediately setup a PDCP entity for SRB1 using the same configuration list of SRB1bis but setting a LCID = 1 as if it were SRB1. Moreover, RLC entities are configured for both SRBs to activate the RLC Acknowledged mode (RLC-AM). Once the RRC Connection Setup Complete message is received without errors, the eNodeB's RRC module considers both SRB1bis and SRB1 to be properly established over the UE context.
Afterwards, the Security Activation procedure starts with the transmission of Security Mode Command (SMC) over SRB1bis, carrying Cyphering and Integrity protection algorithms. For the sake of completeness, Figure 5.6 shows Data Interfaces (i.e *NB_rrc_data_req*, *NB_rlc_data_req* and *NB_mac_rlc_data_req_eNB*) for the downlink transmission in the yellow box. These highlight both LCID and PDCP Transmission Mode, that in the case of SRB1bis is "TRANSPARENT". Upon reception of Security Mode Command, the UE activates security and turns SRB1bis into SRB1 i.e. changes from "no PDCP" to "normal PDCP". Consequently, UE transmits the Security Mode Complete message over SRB1. In this the case, the eNodeB deactivates SRB1bis in the stored UE context and communication over SRB1 starts. However, the security activation procedure might fail and in this case the UE should continue using the configuration used prior the reception of the SMC, as reported in [10] chapter 5.3.4.3. That means, the UE must use SRB1bis for Security Mode Failure. Beyond that, all subsequent RRC messages are transmitted over SRB1 in RLC-AM mode but no SRB2 is activated.
SRB1 configurations can be modified and Data Radio Bearers might be setup through reconfiguration procedure (i.e. *RRCConnectionReconfiguration-NB*). Upon the reception of RRC Connection Reconfiguration Complete message, the eNodeB updates the RRC status in the UE context, activates Radio Bearers and reconfigures the RLC, PDCP and MAC protocol entities through the corresponding interfaces. Finally, dedicated reconfiguration (i.e. *rrc_eNB_generate_dedicatedRRCConnectionReconfiguration_NB*) might be initiated by the S1 AP protocol for establish only up to two DRBs for NB-IoT.

5.1.5 NB-IoT System Information Message Transmission

This subsection discusses about NB-IoT System Information (SI) scheduling and transmission over OpenAirInterface platform. At first, considerations on OAI MAC scheduler are given. Then, some implementation aspects along with an exemplified time diagram are presented as a possible solution for SI delivery.

As explained in subsubsection 2.5.4.3, System Information scheduling in NB-IoT presents deep differences with respect to legacy LTE. Primarily, the reduced set of frequency resources requires in most cases the transmission of SI-Messages over multiple subframes depending on TBS. Moreover, the absence of NPDCCH indication for SI implies a fixed and ordered decoding of MIB-NB and SIB1-NB before getting any other System Information Blocks. Furthermore, development of SI-messages scheduling was made difficult by two other circumstances. From one hand, the downlink NB-IoT MAC's scheduler not completed during this Master project, on the other, the introduction of FAPI standard within the MAC-PHY interface (see section 5.2).

5.1.5.1 OpenAirInterface MAC scheduler

The OpenAirInterface LTE software foresees a subframe-based MAC scheduler, namely *eNB_dlsch_ulsch_scheduler*, for messages delivery over ULSCH and DLSCH channel in both LTE TDD and FDD duplexing mode. Figure 5.7 shows the simplified scheduler scheme for FDD case only where for each subframe the corresponding triggered functions are indicated.

As highlighted in Figure 5.7, the SI scheduling is managed by the *schedule_SI* function that, due to implementation choice, is retrieved only every subframe #5 for each radio frame, providing both SIB1 and SI-Messages transmission. Internally, *schedule_SI* exploits the MAC-

RRC interface *mac_rrc_data_req* whose NB-IoT version has been presented in subsubsection 5.1.3.2. This primitive aims to fill the BCCH's PDU buffer and get the SDU length of the SI message to be delivered from RRC. In particular, since in LTE one single subframe is sufficient for any SI transmission, the *mac_rrc_data_req* only checks if the current radio frame is SIB1 or SI-Message dedicated applying very simple formulas.

Therefore, a first improvement to implement NB-IoT was made on the frame checking algorithms for SIB1-NB and Narrowband SI-Messages transmission in *schedule_SI*. Based on considerations of subsubsection 2.5.4.3, the new *NB_mac_rrc_data_req_eNB* interface was defined introducing two ad-hoc functions, namely *is_SIB1_NB* and *is_SIB23_NB* to check whether a frame is SIB1-NB and/or SI-Message dedicated. This has required additional input parameters like HSFN, Physical Cell Identity and *schedulingInfoSIB1* not previously considered. Moreover, the new primitive was designed for addressing eNodeB side only and the Master Information Block's handling, missing in the previous implementation, was also introduced. For the latter case, it was assumed that, whenever a MIB-NB PDU is requested from RRC, a proper "flag" should be set by the scheduler through a new *schedule_MIB* function. For implementation details of *NB_mac_rrc_data_req_eNB* and the related *is_SIB1_NB* and *is_SIB23_NB* algorithms, refer to OAI code available at [37].
It is worth to point out that, also for NB-IoT case, it was decided to transmit only one single SI-Message containing both the mandatory SIB2-NB and SIB3-NB at the same time, while other System Information Blocks have not been considered.

As a result, Figure 5.8 shows a possible NB-IoT version of the subframe-based MAC scheduler in which the relevant functions for MIB-NB, SIB1-NB and SI scheduling are highlighted. In this case, the *schedule_SI* function is triggered two times: one for SIB1-NB transmission on subframe #4, and one for SI-Message transmission whose starting subframe was assumed #1. Moreover, *schedule_MIB* was introduced on subframe #0 to trigger the MIB-NB delivery. All these functions exploit the *NB_mac_rrc_data_req_eNB* interface to filling the MAC BCCH buffer getting SI-Message, SIB1-NB or MIB-NB's PDU from RRC.

5.1.5.2 System Information Scheduling for NB-IoT

Since a NB-IoT MAC's scheduler was not completed during this Master thesis, this phase of the project has been more oriented to find possible scheduling solutions combining the different system information's requirements (see subsubsection 2.5.4.3).

Figure 5.9 proposes a subframe-based transmission scheme in which MIB-NB, SIB1-NB and an SI-Message, carrying both SIB2-NB and SIB3-NB (namely SIB23-NB), are transmitted over a period of 256 radio frames (i.e. SIB1-NB periodicity). PHY layer NB-IoT's signals, NSSS and PSSS, are also reported since relevant from the subframe occupation view point. Different colours are used to distinguish transmissions, repetitions, periodicity or parameters from different messages. SIB1-NB starting frames (in dark and light green) and associated offset (in orange) are highlighted over the entire period.

With reference to subsubsection 2.5.4.3, the parameters setting for SIB1-NB and SIB23-NB scheduling shown in Figure 5.9 is reported in Table 5.1.

Thus, based on Table 5.1, parameters reported in Table 5.2 were calculated.

The scheduling scheme presented in Figure 5.9 considers a Transport Block of 680 bits for SI, that is the maximum allowed by NB-IoT, but shows the possibility of transmit the 8 subframes required in one single frame. To this aim, the introduction of *si_radioFrameOffset = 1* is fundamental. It allows the SI-Window's starting radio frame to shift with respect to SIB1-NB transmissions which happen in all even frames, and respect to NSSS transmissions (in purple) which fall in even frames too. Since the *si_repetitionPattern* is always an even number, the System Information is delivered in every odd radio frames within the SI-Window following its

SIB1-NB Parameters	**Value**
schedulingInfoSIB1	1, 4, 7 or 10
Physical Cell Identity	0
SIB23-NB Parameters	**Value**
si_ windowLength	160 ms
si_ periodicity	64 rf (radio frames)
si_ repetitionPattern	Every2ndRF
si_ TB	680 bits
si_ radioFrameOffset	1
downlinkBitmap	Not Present

Table 5.1: SIB1-NB and SIB23-NB scheduling parameters for Figure 5.9.

SIB1-NB Parameters	**Value**	**Note**
Number of NPDSCH Repetitions	8	Based on Table 16.4.1.3-3 in [33] and indicated by index "i" in Figure 5.9
SIB1-NB Starting Radio Frame	0	Based on Table 16.4.1.3-4 in [33] and indicated in dark green in Figure 5.9
Offset for SIB1-NB repetitions	16 rf	See Equation 2.1 (radio frames)
SIB23-NB Parameters	**Value**	**Note**
Required subframes for SI-Message	8	As reported in [10] for TBS grater that 120 bits
SI-Window starting Radio Frame	1	Based on Equation 2.2 setting: • n = 0, since no mapping information for SIB2-NB in the first SI-Message as reported in [10]. • w = 16, T = 64 and Offset = 1.
SI-Window starting Subframe number	0	As reported in [10] chapter 5.2.3a

Table 5.2: Calculated SIB1-NB and SIB23-NB parameters for Figure 5.9.

repetition pattern, i.e. every2ndRF. As a result, SI transmission starts at subframe #1 and all 8 subframes available can be used to deliver SIB23-NB. This is highlighted in dark and light blue in Figure 5.9 and is in line with the previous supposed MAC scheduler presented in Figure 5.8.

Overall, different scheduling solutions are possible with the same advantages in terms of SI transmission just discussed. For instance, it is possible to increase the SIB1-NB repetitions up to 16 by varying the *schedulingInfoSIB1* parameter carried by the MIB-NB (see Table 16.4.1.3-3 in [33]). In this way, an UE should be able to acquire the SIB1-NB in less amount of time and consequently starts the decoding of SI-Messages earlier. However, this implies a reduction of radio resources available for other transmissions that can be compensated by a proper trade-off among *si_ windowLength*, *si_ peridicity* and *si_ repetitionPattern* parameters.

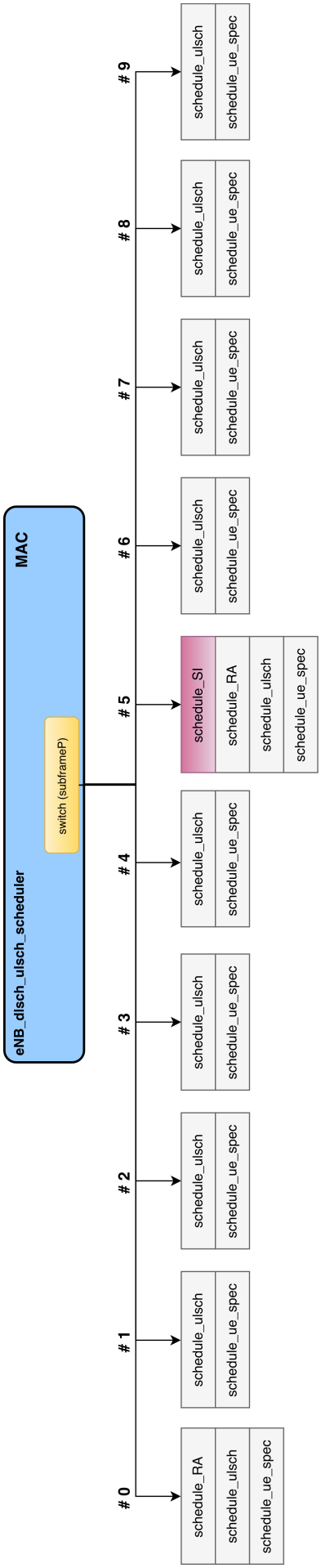

Figure 5.7: OpenAirInterface LTE MAC scheduler scheme for FDD case.

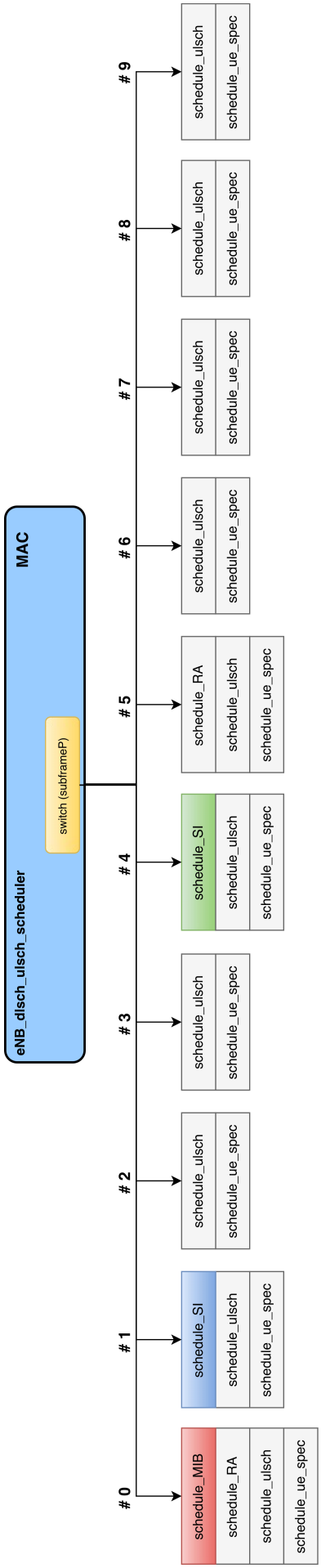

Figure 5.8: OpenAirInterface supposed NB-IoT scheduler scheme.

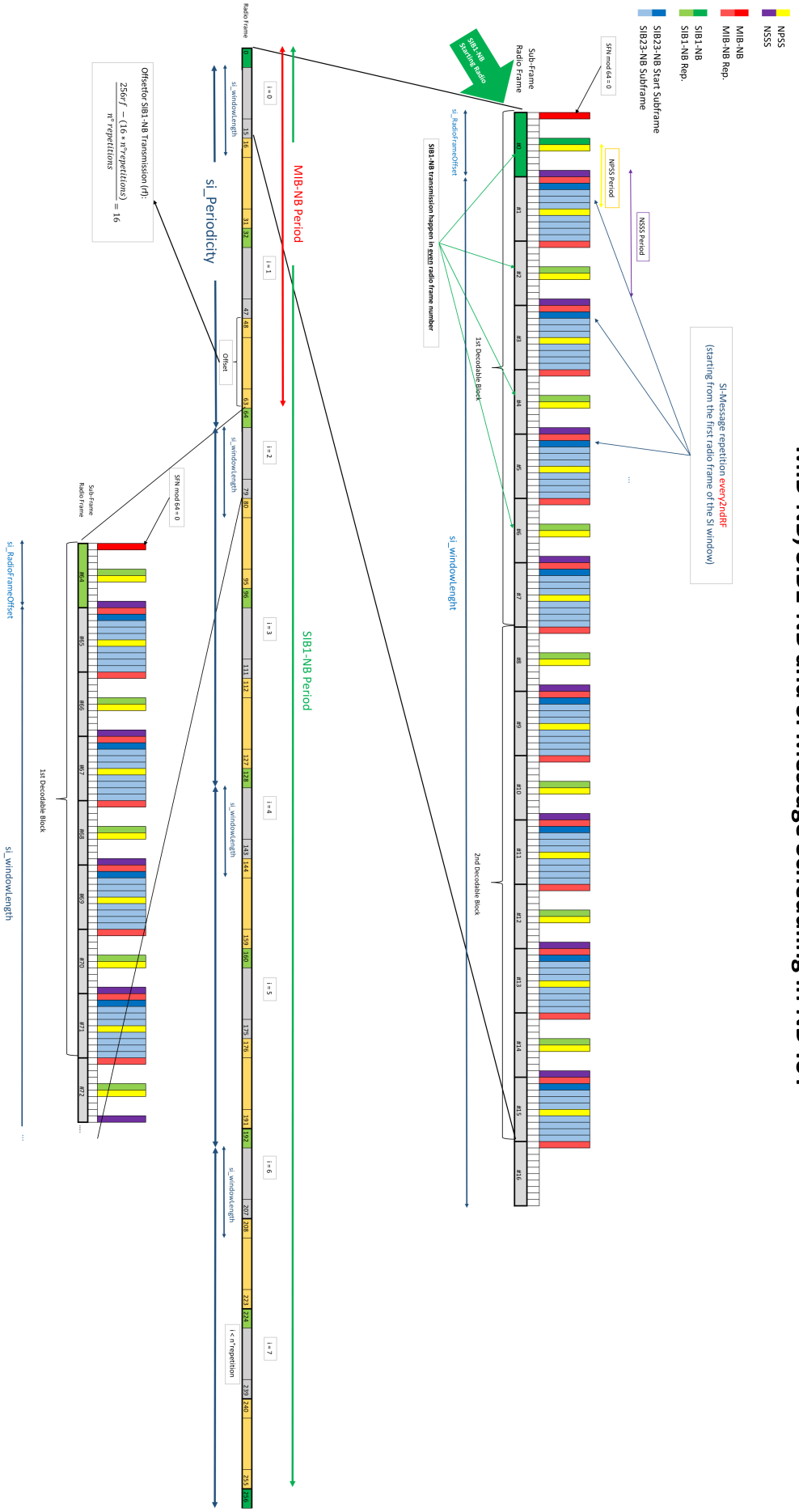

Figure 5.9: Proposed System Information scheduling for NB-IoT.

5.2 A FAPI-Like Approach for OpenAirInterface

As introduced in chapter 3, the definition of a common Application Platform Interface (API) ensures that system vendors can take advantages of the latest innovation in silicon and software with minimum barriers and least amount of investments [9]. The establishment of such scalable ecosystem also for future 5G deployments has not escaped to the OSA which has quickly started adopting FAPI over the *openairinterface5G* software for both LTE and NB-IoT standard.

In this chapter, the design of a Functional API (FAPI) within OAI's MAC and PHY layer for NB-IoT is discussed. First, the implementation of an Interface-Module (IF-Module) and related primitives over the OpenAirInterface platform are described. Both FAPI's configuration and data transport aspects are considered (see section 3.3). Finally, a new PHY layer's approach for downlink transmission is presented.

5.2.1 IF-Module and Related Procedures

The first step towards FAPI adoption for NB-IoT, was the definition of a configurable and dynamically loadable Interface Module (IF-Module) between MAC (L2) and PHY (L1) layers. The module, whose data structure is called *IF_Module_t*, is instantiated at eNodeB initialization time and registered by both L1 and L2. It provides the transport mechanism for uplink, downlink and configuration data in a FAPI-like approach.
However, instead of replicating the large set of FAPI messages for both P5 and P7 interfaces, these have been grouped in only three primitives listed below and shown in Figure 5.10.

- UL_indication: enables all uplink information received in one Transmission Time Interval (TTI) to move from PHY to MAC layer. Comprises different FAPI P7 messages including *RACH.indication*, *RX_ULSCH.indication*, *HARQ.indication* and others [9]. It provides Preambles, ULSCH SDU, ACK/NACK etc.
- schedule_response: enables scheduling results to be transmitted from MAC to PHY layer. Comprises different downlink FAPI P7 messages like *DL/UL_CONFIG.request*, *TX.request* and *HI_DCI0.request* [9]. It provides DLSCH, RACH, BCH or DCI SDUs.
- PHY_config_req: enables common and UE-specific PHY layer configuration procedure. Comprises the FAPI P5/P7 messages *CONFIG.request* and *UE_CONFIG.request* [9].

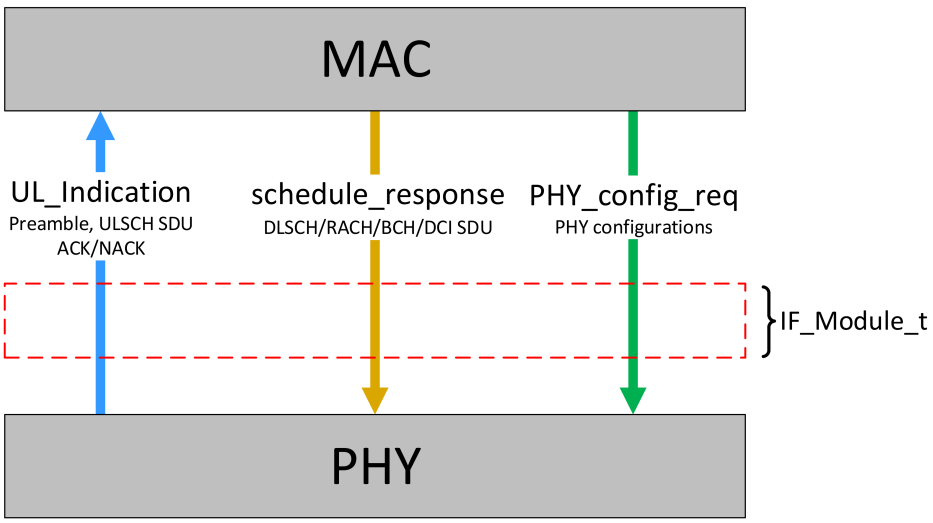

Figure 5.10: IF-Module FAPI-like messages.

Specifically, the three designed primitives are used to carrying fundamental PHY and MAC layer information stored in three different data structures, namely PHY_Config_t, UL_IND_t and Sched_Rsp_t (see Figure 5.11 and Figure 5.12). The latter ones are defined in the *IF_Module_nb_iot.h* file held in openair2/PHY_INTERFACE directory, and contain FAPI-compliant parameters defined from [9].

Following FAPI specifications, OpenAirInterface implements both P5-like configuration procedure and P7-like subframe/data procedure which are reported in Figure 5.11 and Figure 5.12 respectively.

5.2.1.1 Configuration procedure

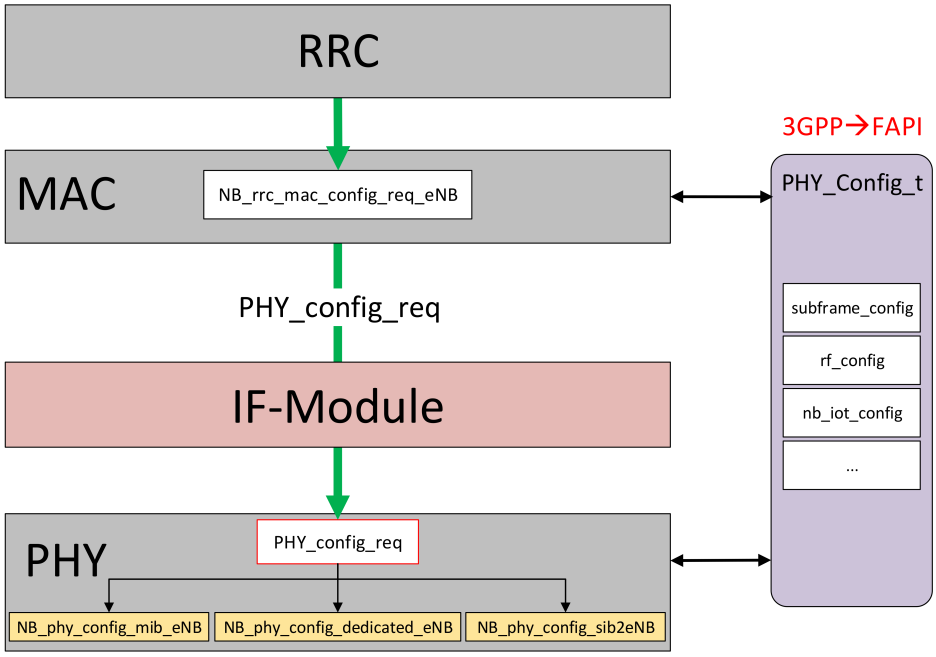

Figure 5.11: FAPI P5-like configuration procedure between OpenAirInterface MAC and PHY layers.

As introduced in section 3.3, FAPI standard foresees a set of configuration procedures that move the L1 through different states. However, as reported in Figure 5.11, no state machine at physical layer was introduced but RRC statically configure PHY through *NB_rrc_mac_config_req_eNB* interface. Before FAPI adoption, this primitive directly triggered the PHY layer's functions *phy_config_sib1_eNB*, *phy_config_sib2_eNB* and *phy_config_dedicated_eNB* (yellow boxes) based on input parameters. After the IF-Module introduction, the RRC layer should first fill the newly defined PHY_Config_t structure through *NB_rrc_mac_config_req_eNB* interface, and then activates the PHY_config_req message to transmit the stored data. The latter causes *NB_phy_config_mib_eNB*, *NB_phy_config_sib2_eNB* or *NB_phy_config_dedicated_eNB*'s activation that directly configures the physical layer. For further implementation details refer to [37]. As shown in Figure 5.11, PHY_Config_t structure carries parameters related to subframe, radio frequency and specific NB-IoT configuration. In this phase of the project, an important step was related on mapping the 3GPP compliant variables got from *NB_rrc_mac_config_req_eNB* interface, into FAPI ones over the PHY_Config_t structure. During this, it was noticed that a group of OAI's parameters were not considered by the FAPI standard for different logics. First, Small Cell Forum's releases only focus on eNodeB side of the Access Network, therefore all the RRC UE related information were not reported. Moreover, some parameters were only for MAC layer configuration and did not require to be transmitted to PHY layer. Last, but most relevant, there was a

difference in the UE context management used by OpenAirInterface and the one foreseen in FAPI standard. In OAI, once an UE context is established at eNodeB, all the user-specific information is maintained over that until UE's connection release. On the contrary, FAPI UE-specific parameters are maintained in a semi-static configuration at PHY layer being sent periodically over a specific *UE_CONFIG.request* message and deleted immediately after. As a result, most of FAPI NB-IoT's specific TLVs over the PHY_Config_t data structure does not hold UE-Specific configurations but only common one.

5.2.1.2 Data procedure

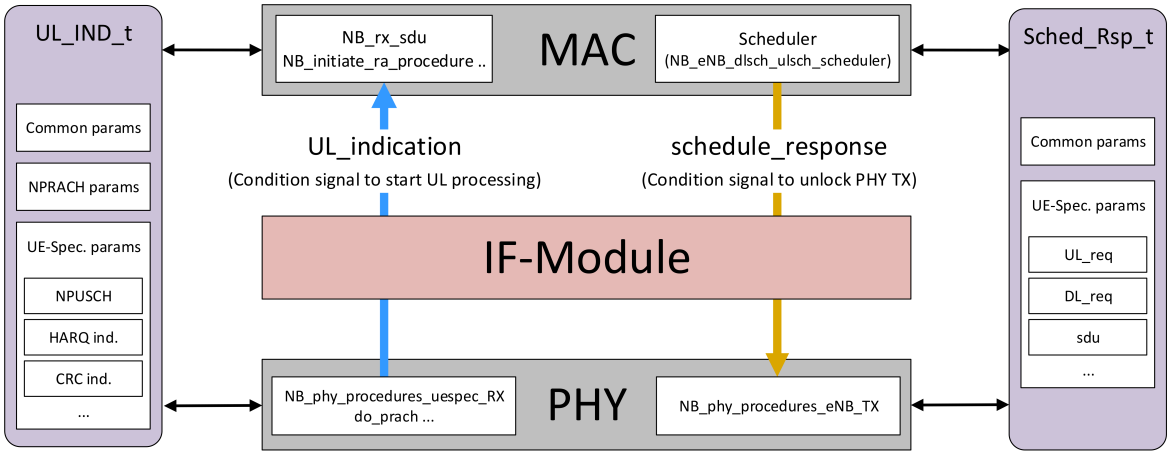

Figure 5.12: FAPI P7-like subframe/data procedure between OpenAirInterface MAC and PHY layers.

Similar approach as for configuration is applied for data procedures. The Sched_Rsp_t structure is filled by the MAC scheduler with common and UE-specific information using FAPI parameters. Then, the *schedule_response* message is activated on a subframe-base pace, working as a condition signal to unlock the PHY access to Sched_Rsp_t contents. Once received, *schedule_response* triggers the downlink transmission procedure, i.e. *NB_phy_procedures_eNB_TX*.
Likewise, the UL_IND_t is filled by L1 with common's, NPRACH and UE-specific's parameters from corresponding procedures. Then, the UL_indication message is activated allowing MAC layer to access and process data over the shared structure and consequently trigger the scheduler, i.e. *NB_eNB_dlsch_ulsch_scheduler*. For further details on functions implementation refer to OAI code available at [37].

5.2.2 IF-Module Initialization

The IF-module is agnostic to L1 and L2 implementation details, being only the transport mechanism which provides suitable interfaces. Its data structure, IF_Module_t, is just constituted by function pointers that are associated to the corresponding messages at initialization time. To this aim, two recording functions are used, namely *IF_Module_init_L2* and *IF_Module_init_L1*, for the north and the south bound of the module respectively. They are triggered to mapping the UL_indication, schedule_response and PHY_config_req's implementations to the module function pointers. Figure 5.13 shows an example of the IF-Module's recording procedure while Figure 5.14 reports the module's data structure from the OpenAirInterface code.

So far, the IF-Module just provides a function mapping mechanism since it is assumed that both PHY and MAC layer are running over the same host. However, the module has been

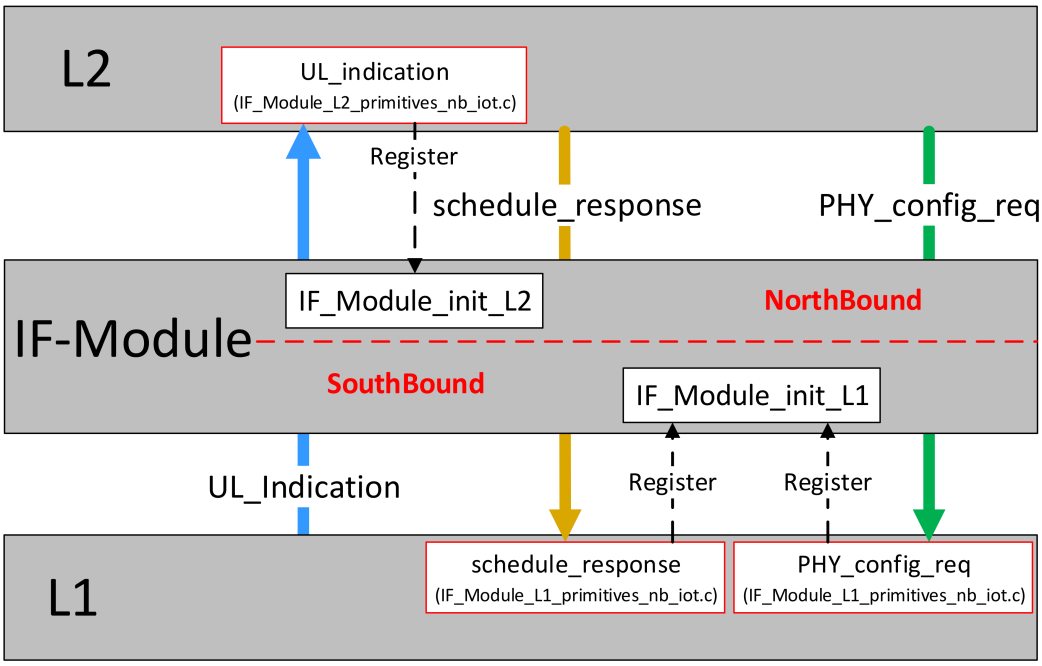

Figure 5.13: IF-Module recording procedure.

```
typedef struct IF_Module_s{

    //Function pointers
    void (*UL_indication)(UL_IND_t *UL_INFO);
    void (*schedule_response)(Sched_Rsp_t *Sched_INFO);
    void (*PHY_config_req)(PHY_Config_t* config_INFO);

}IF_Module_t;
```

Figure 5.14: IF-Module data structure from OpenAirInterface code.

thought to provide also UDP or nFAPI transport mechanism for future implementations in which PHY and higher layer parameters run over separate machines.

5.2.3 PHY Procedures for FAPI Approach

The introduction of a Functional API (FAPI) within the OAI software also affected the original PHY layer procedure in both transmission and reception part. In this subsection, a first overview on the new physical layer processes for NB-IoT and FAPI purposes is given. Then, the design for the downlink procedure is presented, since constitutes part of the final work of this Master's thesis project.

With reference to subsection 4.3.2, the adoption of NB-IoT and FAPI standards has lead to the design of a new *NB_rxtx* function whose flow diagram is reported in Figure 5.15 and described here below.

As a preliminary step, it is assumed that only one single physical thread is running to perform both Reception (RX) and Transmission (TX). As in original implementation, RX refers to the current (n) subframe while TX to the future (n+4) one. After memory allocation of both UL_INFO_t and Sched_Rsp_t data structures, the *NB_rxtx* function starts common and UE-specific reception procedures. In this case, it first wakes up the NPRACH thread and then, after filling UL_INFO_t, it activates the UL_indication message triggering MAC layer's processing and scheduler. The entire transmission procedure is directly managed over

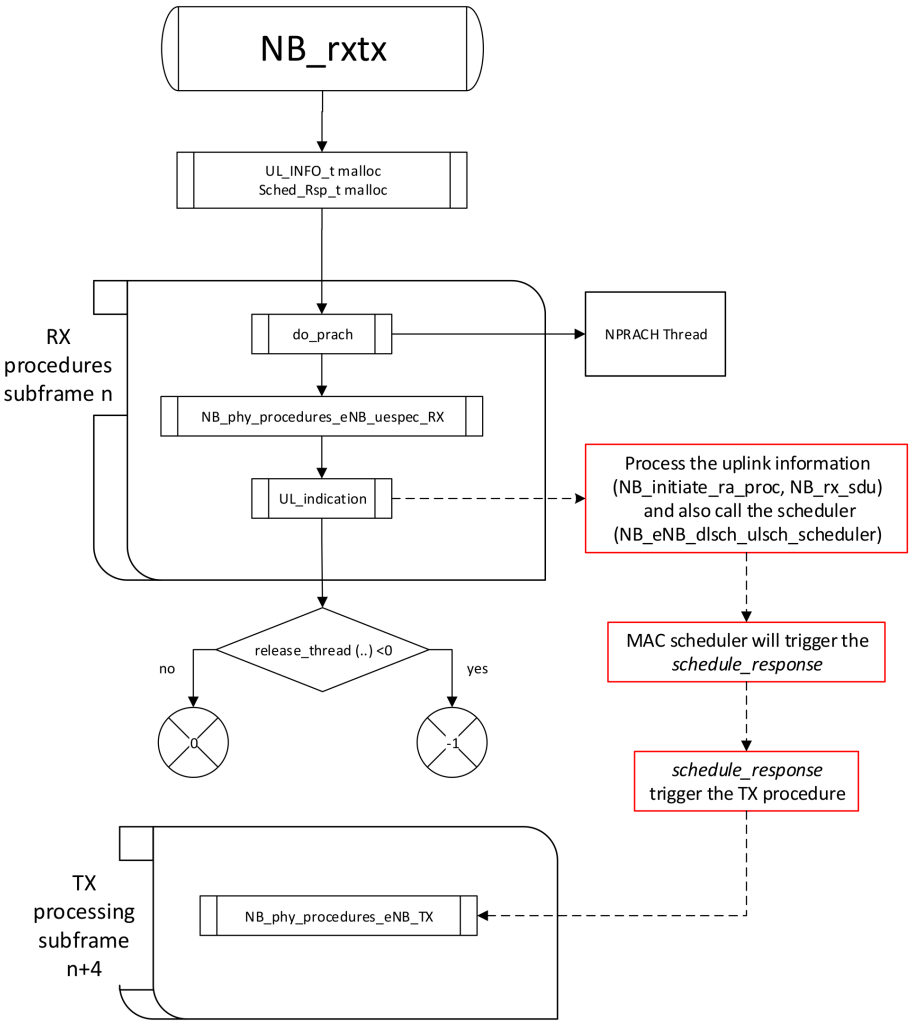

Figure 5.15: *NB_rxtx* flow chart.

the *schedule_response* function that first parses and stores the Sched_Rsp_t's parameters in corresponding PHY layer's data structures (see green boxes in Figure 5.16), and then triggers the *NB_phy_procedure_eNB_TX*. This is also compliant with what shown in Figure 5.12.

5.2.3.1 PHY Transmission procedure

In this subsection, the design of the new physical layer transmission procedure (i.e. *NB_phy_procedure_eNB_TX*) is described from a logical view point. Important assumptions on scheduler functionalities have been introduced to combine both NB-IoT and FAPI requirements paying particular attention to system information delivery.

Figure 5.16 shows the logical steps of *NB_phy_procedure_eNB_TX*. Relevant PHY's variables at stake with corresponding data structures are reported in green boxes while checking conditions are highlighted in blue and purple. Moreover, all triggered functions are depicted with corresponding transmitted messages. The entire procedure is then described here below while for further details on functions implementation refer to OpenAirInterface code available at [37].

In each subframe, the transmission buffer for each antenna port is flushed before generating Narrowband Primary (NPSS) and Secondary (NSSS) Synchronization Signals. After, transmission of System Information messages starts by applying frame and subframe's checks (light blue boxes).

Whenever subframe #0 occurs, *generate_ npbch* function is activated to transmit the Master Information Block. Following FAPI approach, the NBCH PDU is provided by higher layers over the *npbch* structure only at the beginning of MIB-NB period (i.e. every 640 ms). Therefore, for each subframe #0, PHY checks whether a new MIB-NB PDU is present. If not, MIB-NB's sub-blocks transmission and repetition is performed (see subsubsection 3.3.2.1). Similar approach is for SIB1-NB delivery, in which both frame and subframe are checked exploiting *is_ SIB1_ NB* function. If a SIB1-NB transmission occurs, the *npdsch_ procedure* is triggered with the SIB1-NB's PDU got from the *ndlsch_ SIB1* data structure.
For SI-Message delivery, the implementation resulted more complicated. At first, the availability of empty subframes not occupied by NSSS, NPSS, MIB-NB and SIB1-NB transmission is checked. Then, a strong assumption has been introduced. As discussed in subsubsection 3.3.2.2 about FAPI, it seems that also for SI transmissions over NDLSCH the PHY layer should manage subframe repetitions. However, no FAPI message can instruct L1 about fundamental System Information's parameters like *si_ WindowLength*, *si_ Periodicity* or *si_ RepetitionPattern* (see Table 2.5). Therefore, it has been necessary to assume that the MAC manages transmissions and repetitions by triggering the PHY layer in every subframe on which a SI delivery occurs. This is realized through the Sched_Rsp_t's content (see Figure 5.12). In the latter, a new System Information SDU is carried whenever an SI-Message transmission starts, while no SDU is included when PHY layer subframe repetitions should be performed (see Figure 5.9 with reference to dark and light blue subframes). Fortunately, this approach was approved by the Small Cell Forum itself that kindly confirmed that System Information messages and higher layer repetitions described in [10] section 5.2.3a, are expected to be handled by the MAC.
Afterwards, Random Access Response (RAR) and UE-Specific transmissions are initiated by checking for available subframes (purple boxes) and verifying if *ndlsch_ ra* and *ndlsch[UE_ id]* data structure have been previously filled by the *schedule_ response* message. In this case, repetitions are managed directly at PHY layer as reported by FAPI specifications.
Finally, *generate_ dci_ top_ NB* is triggered to transmit DCI messages stored over the physical layer *DCI_ pdu* data structure.

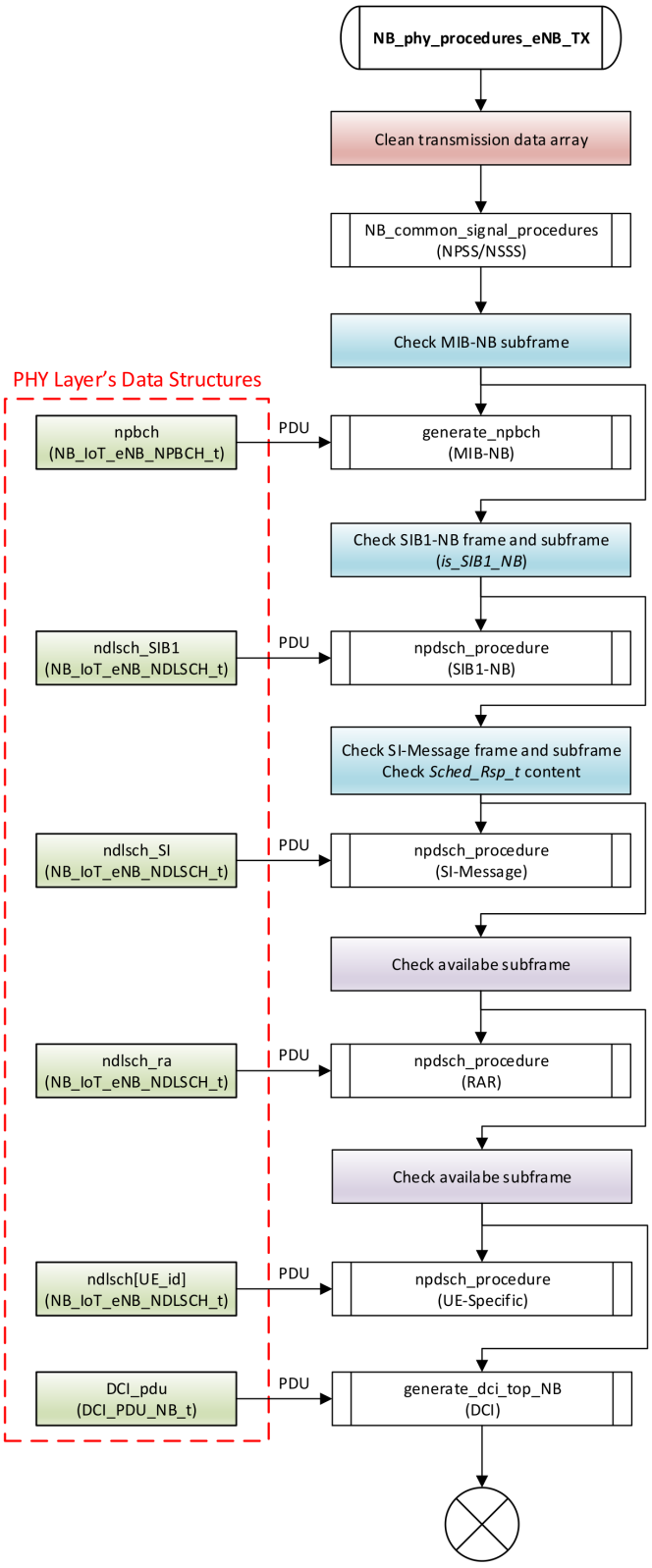

Figure 5.16: *NB_phy_procedure_eNB_TX*'s logical steps with relevant PHY's data structures.

Chapter 6

Conclusions and Future Works

This Master thesis addresses the implementation of Narrowband Internet of Things (NB-IoT) protocol stack on OpenAirInterface (OAI), focusing on Radio Resource Control (RRC) layer and on developing a multi-vendor platform interface following the Functional Application Platform Interface (FAPI) standard.

The first part of this work presents OpenAirInterface as a flexible platform towards an open LTE ecosystem. The software architecture is discussed to logically describe and classify major RRC procedures and primitives as well as give an insight on the basic PHY mechanism. In this sense, the thesis represents a valid tool for any user that is willing to approach software programming on OAI platform.
To bring NB-IoT functionalities onto OpenAirInterface, a new RRC layer state machine is defined ensuring compliance with new data structures and Information Elements carried by the protocol messages. A set of RRC interfaces for configuration, data exchange and context management is proposed taking in consideration NB-IoT requirements on System Information scheduling, PHY layer procedures and bearer management. As for the latter, a new approach for Signalling Radio Bearers (SRBs) is introduced taking into account the new NB-IoT bearer, SRB1bis. The Logical Channel Identity (LCID) is used to differentiate between SRB1bis and SRB1 allowing to bypass PDCP configuration for the first but not for the second. Further investigations have lead to identify the suitable RRC message after which SRB1bis is released and the layer starts communicate over SRB1 only.
Furthermore, a deeper analysis on NB-IoT System Information (SI) scheduling is conducted. The absence of NPDCCH indication for SI transmissions implies a fixed and ordered decoding of MIB-NB and SIB1-NB before getting any further SIBs. This requires new scheduling parameters and a different approach for resource mapping at subframe level. To this purpose, a subframe-based transmission scheme is proposed, which optimizes radio frame utilization considering the worst case of maximum SI-Message size.

Finally, previously developed aspects are fitted into the functional split introduced on OpenAirInterface following FAPI standard. A configurable and dynamically loadable Interface Module (IF-Module) is designed between OAI's MAC and PHY layers to ensure multi-vendor interoperability. Both P7 and P5 procedures are supported but instead of replicating the large set of FAPI messages, the Module is equipped with only three basic primitives to provide both data and configuration capabilities. Moreover, introducing such functional split along with NB-IoT requisites, has required to re-design PHY layer procedures of OAI, for which a downlink transmission scheme is proposed.

The lack of experimental results in this six-month thesis work is justified by the complexity of the context in which it fits. Developing a new 3GPP standard into an open-source platform is generally addressed as long-term project by research communities. It takes time, especially in the case of NB-IoT which revolutionizes many aspects of LTE protocol stack. In parallel,

the convergence of a FAPI approach within OpenAirInterface made the achievement of such results more challenging.
Nevertheless, the scenario on which this project takes place is of scientific relevance. It is shown how open-source solutions, like OAI, break the boundary of proprietary and closed systems by providing researchers with an environment in which they can rapidly prototype and test latests mobile radio technologies, NB-IoT for instance. This approach is transforming the telecommunication industry by providing flexible platforms which boosts innovation and revolutionizes the future of mobile networks.

6.1 Future Work

This Master thesis opens a large number of possible extensions, prospectives and future works.

As for Radio Resource Control, OpenAirInterface still lacks from some basic messages and related procedures, which are included also by NB-IoT: for instance, *Paging*, which is not implemented at all, and *RRCConnectionReestablishment* which is directly rejected by OAI eNodeB without implementing any related management procedure.
Moreover, 3GPP Rel.13 introduces the new RRC Suspend/Resume procedure to re-establish an RRC connection by reducing the amount of signalling required. Bringing this feature into OpenAirInterface will make it faster on handover processes or Radio Link Failures recovery for instance, ensuring less bit over the air and low latency.
As this work mainly focuses on the radio software of OpenAirInterface, i.e. *openairinterface5G*, there is room for new projects to upgrade OAI core network (*openairCN*) to support Cellular IoT (CIoT) optimizations for infrequent small data transmission.
Overall, the implementation of NB-IoT on OpenAirInterface platform still requires the development of a MAC scheduler, and PHY layer encoding, modulation and scrambling procedures. These are expected to be integrated before the end of 2017 by the research groups and industries currently working on the project.

Abbreviations

3GPP	3rd Generation Partnership Project.
API	Application Platform Interface.
APP	Application.
BBU	Baseband Unit.
CCCH	Command Control Channel.
CIoT	Cellular IoT.
DCCH	Dedicated Control Channel.
DCI	Downlink Control Indication.
DL	Downlink.
DLSCH	Downlink Shared Channel.
DMRS	Demodulation Reference Signal.
DRB	Data Radio Bearer.
EC-GSM	Extended Coverage GSM.
eMBB	Enhanced Mobile Broadband.
eMBMS	Evolved Multimedia Broadcast Multicast Services.
eMTC	Enhanced Machine Type Communication.
EPC	Evolved Packet Core.
EPS	Evolved Packet System.
E-UTRAN	Evolved Universal Terrestrial Radio Access Network.
FAPI	Functional Application Platform Interface.
FDD	Frequency Division Duplexing.
FSM	Finite State Machine.
HSFN	Hyper-System Frame Number.
HSS	Home Subscriber Server.
IE	Information Element.
IoT	Internet of Things.
ITTI	Inter Task Interface.
ITU	International Telecommunication Union.

LCID	Logical Channel Identity.
LPWAN	Low-Power Wide Area Network.
LTE	Long Term Evolution.
MAC	Medium Access Control.
MCL	Maximum Coupling Loss.
MIB-NB	Narrowband Master Information Block.
MME	Mobility Management Entity.
mMTC	Massive Machine Type Communication.
MTC	Machine Type Communication.
MTCH	Multicast Traffic Channel.
NAS	Non-Access Stratum.
NBCH	Narrowband Broadcast Channel.
NB-IoT	Narrowband Internet of Things.
nFAPI	Network Functional Application Platform Interface.
NMM	Network Monitor Mode.
NPBCH	Narrownband Physical Broadcast Channel.
NPDCCH	Narrowband Physical Downlink Control Channel.
NPDSCH	Narrowband Physical Downlink Shared Channel.
NPRACH	Narrowband Physical Random Access Channel.
NPSS	Narrowband Primary Synchronization Signal.
NPUSCH	Narrowband Physical Uplink Shared Channel.
NRS	Narrowband Reference Signal.
NSSS	Narrowband Secondary Synchronization Signal.
OAI	OpenAirInterface.
OFDM	Orthogonal Frequency Division Multiplexing.
OSA	OpenAirInterface Software Alliance.
PDCP	Packet Data Convergence Protocol.
PGW	Packet Data Network Gateway.
PHY	Physical.
PMCH	Physical Multicast Channel.
PNF	Physical Network Function.
QoS	Quality of Service.
RAN	Radio Access Network.
RAT	Radio Access Technology.
RF	Radio Frequency.
RLC	Radio Link Control.
RRC	Radio Resource Control.
RRU	Remote Radio Unit.
SCEF	Service Capability Exposure Function.
SC-FDMA	Single Carrier Frequency Division Multiple Access.
SDR	Software Defined Radio.

SFN	System Frame Number.
SGW	Serving Gateway.
SI	System Information.
SIB	System Information Block.
SRB	Signalling Radio Bearer.
TBS	Transport Block Size.
TI	Transaction Identifier.
TLV	Type-Length-Value.
TTI	Transmission Time Interval.
UE	User Equipment.
UHF	Ultra High Frequency.
UL	Uplink.
URLLC	Ultra-Reliable Low Latency Communication.
USRP	Universal Software Radio Peripheral.
VNF	Virtual Network Function.

Bibliography

[1] "Evolving LTE to fit the 5G future," Ericsson, Technology Review, January 2017. [Online]. Available: https://www.ericsson.com/en/publications/ericsson-technology-review/archive/2017/evolving-lte-to-fit-the-5g-future

[2] NB-IoT Principle and Test. [Online]. Available: https://wenku.baidu.com/view/b752b510b42acfc789eb172ded630b1c59ee9bf4

[3] Software-Defined Radio. [Online]. Available: https://en.wikipedia.org/wiki/Software-defined_radio

[4] F. Kaltenberger, "Openairinterface 5g: Overview, installation, usage," 2016, OAI Workshop.

[5] F. Kaltenberger, R. Knopp, N. Nikaein, D. Nussbaum, L. Gauthier, and C. Bonnet, "Openairinterface: Open-source software radio solution for 5G," in European Conference on Networks and Communications (EUCNC), Paris, France, 2015.

[6] "Narrowband Internet of Things," Rohde & Schwartz, Whitepaper, 2016. [Online]. Available: https://www.rohde-schwarz.com/us/applications/narrowband-internet-of-things-white-paper_230854-314242.html

[7] Y.-P. E. Wang, X. Lin, A. Adhikary, A. Grövlen, Y. Sui, Y. Blankenship, J. Bergman, and H. S. Razaghi, "A primer on 3GPP Narrowband internet of things (NB-IoT)," arXiv preprint arXiv:1606.04171, 2016.

[8] TS 36.211 Evolved Universal Terrestrial Radio Access (E-UTRA): Physical channels and modulation, 3GPP, 2017, v14.2.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2425

[9] FAPI and nFAPI specifications, Small Cell Forum, 2017, rel. 9.0. [Online]. Available: http://scf.io/en/documents/082_-_nFAPI_and_FAPI_specifications.php

[10] TS 36.331 Evolved Universal Terrestrial Radio Access (E-UTRA): Radio Resource Control (RRC), 3GPP, 2017, v14.2.1. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2440

[11] F. Kaltenberger, "Openairinterface 5g: Overview, installation, usage," 2017, OAI Workshop.

[12] S. Mao, Y. Huang, Y. Li, P. Agrawal, and J. Tugnait, "Introducing software defined radio into undergraduate wireless engineering curriculum through a hands-on approach," in Proc. The 2013 ASEE Annual Conference, 2013, pp. 1–10.

[13] N. Nikaein, M. K. Marina, S. Manickam, A. Dawson, R. Knopp, and C. Bonnet, "Openairinterface: A flexible platform for 5g research," ACM SIGCOMM Computer Communication Review, vol. 44, no. 5, pp. 33–38, 2014.

[14] Driving connectivity for 10 years. [Online]. Available: http://www.smallcellforum.org/

[15] open-nFAPI. [Online]. Available: https://github.com/cisco/open-nFAPI

[16] S. Sesia, M. Baker, and I. Toufik, LTE-the UMTS long term evolution: from theory to practice. John Wiley & Sons, 2011.

[17] C. Johnson, Long term evolution in bullets, 2012.

[18] I. Jovović, I. Forenbacher, and M. Periša, "Massive machine-type communications: An overview and perspectives towards 5g," in The 3rd International Virtual Research Conference In Technical Disciplines, RCITD 2015, 2015.

[19] "Cellular networks for massive IoT," Ericsson, White Paper, January 2016. [Online]. Available: https://www.ericsson.com/assets/local/publications/white-papers/wp_iot.pdf

[20] "Ericsson Mobility Report," Ericsson, Tech. Rep., November 2016. [Online]. Available: https://www.ericsson.com/assets/local/mobility-report/documents/2016/ericsson-mobility-report-november-2016.pdf

[21] Lora. [Online]. Available: https://www.lora-alliance.org

[22] Sigfox. [Online]. Available: https://www.sigfox.com/en

[23] "5G radio access," Ericsson, White Paper, April 2016. [Online]. Available: https://www.ericsson.com/assets/local/publications/white-papers/wp-5g.pdf

[24] LTE IoT is starting to connect the massive IoT today, thanks to eMTC and NB-IoT. [Online]. Available: https://www.qualcomm.com/news/onq/2017/06/15/lte-iot-starting-connect-massive-iot-today-thanks-emtc-and-nb-iot

[25] Openairinterface: 5g software alliance for democratising wireless innovation. [Online]. Available: http://www.openairinterface.org

[26] OpenBTS. [Online]. Available: http://openbts.org/

[27] Amarisoft. [Online]. Available: https://www.amarisoft.com/

[28] Welcome to the openairinterface project. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g/wikis/home

[29] Abstract Syntax Notation One. [Online]. Available: https://en.wikipedia.org/wiki/Abstract_Syntax_Notation_One

[30] TR 23.720 Study on architecture enhancements for Cellular Internet of Things, 3GPP, 2016, v13.0.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2894

[31] TS 36.101 Evolved Universal Terrestrial Radio Access (E-UTRA): User Equipment (UE) radio transmission and reception, 3GPP, 2017, v14.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2411

[32] TS 36.304 Evolved Universal Terrestrial Radio Access (E-UTRA): User Equipment (UE) procedures in idle mode, 3GPP, 2017, v14.3.0. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2432

[33] TS 36.213 Evolved Universal Terrestrial Radio Access (E-UTRA): Physical layer procedures, 3GPP, 2017, v14.2.1. [Online]. Available: https://portal.3gpp.org/desktopmodules/Specifications/SpecificationDetails.aspx?specificationId=2427

[34] TSG-RAN WG1 NB-IoT Adhoc, 3GPP, 2016. [Online]. Available: http://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=678674

[35] Small cells, what's the big idea?, Small Cell Forum, 2014, rel. 7.0. [Online]. Available: http://scf.io/en/documents/030_-_Small_cells_big_ideas.php

[36] C.-Y. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, "Flexcran: A flexible functional split framework over ethernet fronthaul in cloud-ran."

[37] Openairinterface 5G Wireless Implementation. [Online]. Available: https://gitlab.eurecom.fr/oai/openairinterface5g

[38] TSG RAN2 NB-IOT Ad-hoc Meeting 2, 3GPP, 2016. [Online]. Available: http://portal.3gpp.org/ngppapp/CreateTdoc.aspx?mode=view&contributionId=700654