NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCE
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc Thesis

Software-Defined Networking Enabled Technologies for Heterogeneous Radio Access Networks

Christos V. Kitsanelis

Supervisors: **Alonistioti Athanasia,** Associate Professor
Barmpounakis Sokratis, Postdoctoral Research Associate

ATHENS
JULY 2019

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνολογίες Δικτύωσης Βασισμένης στο Λογισμικό για Ετερογενή Δίκτυα Ασύρματης Πρόσβασης

Χρήστος Β. Κιτσανέλης

Επιβλέποντες: **Αλωνιστιώτη Αθανασία,** Αναπληρώτρια Καθηγήτρια
Μπαρμπουνάκης Σωκράτης, Μεταδιδακτορικός Ερευνητής

ΑΘΗΝΑ
ΙΟΥΛΙΟΣ 2019

BSc THESIS

Software-Defined Networking Enabled Technologies for Heterogeneous Radio Access Networks

Christos V. Kitsanelis
S.N.: 1115201200065

Supervisors: **Alonistioti Athanasia,** Associate Professor
Barmpounakis Sokratis, Postdoctoral Research Associate

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Τεχνολογίες Δικτύωσης Βασισμένης στο Λογισμικό για Ετερογενή Δίκτυα Ασύρματης Πρόσβασης

Χρήστος Β. Κιτσανέλης
Α.Μ.: 1115201200065

Επιβλέποντες: **Αλωνιστιώτη Αθανασία,** Αναπληρώτρια Καθηγήτρια
Μπαρμπουνάκης Σωκράτης, Μεταδιδακτορικός Ερευνητής

ABSTRACT

The term Internet of Things is used to describe a network of physical devices that are connected to each other through the internet for the exchange of data and information. It is an emerging technology with real world applications in numerous fields such as smart cities, agriculture and healthcare. With the number of IoT devices expected to reach over 18 billion by 2022, the management and control of such a large network will be no easy ordeal. Software-Defined Networking provides us with a solution to this issue, as it allows the configuration and monitoring of the network performance to be implemented programmatically through a logical software component, bypassing the need to alter existing hardware architecture. The aim of this thesis is to present an in depth view of both of these technologies whilst also evaluating the integration of SDN in a heterogeneous environment which includes LTE and IoT devices in a real life scenario. In order to achieve this task, we will be using the NS-3 tool, an open source discrete-event network simulator targeted primarily for research use. We will be creating a topology with a script written in C++ using the LoRaWAN, LTE and OpenFlow 1.3 modules provided by the NS-3 community. After conducting a series of experiments based on diverse configurations and comparing the results we receive, we will be able to determine the effectiveness of incorporating these technologies.

SUBJECT AREA: Communication Networks

KEYWORDS: Internet of Things, SDN, LTE, LoRaWAN, OpenFlow, NS-3

ΠΕΡΙΛΗΨΗ

Ο όρος Διαδίκτυο των Πραγμάτων χρησιμοποιείται για να περιγράψει ένα δίκτυο από συσκευές περιορισμένων δυνατοτήτων που είναι συνδεδεμένες μεταξύ τους μέσω του διαδικτύου με σκοπό την ανταλλαγή δεδομένων και πληροφοριών. Είναι μια αναπτυσσόμενη τεχνολογία με πραγματικές εφαρμογές σε πολυάριθμα πεδία όπως των έξυπνων πόλεων, της γεωργίας και της ιατρικής περίθαλψης. Με τον αριθμό τέτοιων συσκευών να υπολογίζεται ότι θα ξεπεράσει τα 18 δισεκατομμύρια μέχρι το 2022, η διαχείριση και ο έλεγχος τόσο μεγάλου δικτύου δε θα είναι εύκολη υπόθεση. Η δικτύωση βασισμένη στο λογισμικό μας προσφέρει μια λύση σε αυτό το πρόβλημα, καθώς επιτρέπει η ρύθμιση και επίβλεψη της δικτυακής δραστηριότητας να υλοποιείται προγραμματιστικά μέσω ενός λογικού στοιχείου λογισμικού, διαπερνώντας την ανάγκη για τροποποίηση προϋπάρχουσας αρχιτεκτονικής υλικού. Σκοπός της συγκεκριμένης πτυχιακής είναι η παρουσίαση αυτών των τεχνολογιών καθώς και η αξιολόγηση της ένταξης των τεχνολογιών δικτύωσης βασισμένης στο λογισμικό σε ένα ετερογενές περιβάλλον που περιλαμβάνει συσκευές LTE και IoT σε ένα πραγματικό σενάριο. Για την επίτευξη του σκοπού μας, θα χρησιμοποιηθεί το εργαλείο NS-3, ένα πρόγραμμα ελεύθερου λογισμικού προσομοιώσεων δικτύου διακριτών καταστάσεων που χρησιμοποιείται για ερευνητικούς σκοπούς. Θα δημιουργηθεί μία τοπολογία μέσω ενός προγράμματος που έχει γραφτεί σε C++ με τις μονάδες LoRaWan, LTE και OpenFlow 1.3 που έχουν διαμοιραστεί από την κοινότητα του NS-3. Μετά από μια σειρά πειραμάτων με ποικίλες παραμέτρους και συγκρίνοντας τα αποτελέσματα που παράγονται, θα καταφέρουμε να αποφανθούμε για την αποτελεσματικότητα της ενσωμάτωσης αυτών των τεχνολογιών.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Δίκτυα Επικοινωνιών

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: Διαδίκτυο των Πραγμάτων, SDN, LTE, LoRaWAN, OpenFlow, NS-3

To my family and friends, who have supported me throughout my studies.

ACKNOWLEDGMENTS

I would like to express my gratitude to Prof. Athanasia Alonistioti for her guidance and for giving me the chance to work on such an interesting topic.

I would also like to thank Dr. Sokratis Barmpounakis for his continuous support on this project. His networking expertise and positive attitude were instrumental in the completion of this thesis. I sincerely hope that I will be able to collaborate with him again in the future.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview

The world of communication has witnessed a great transformation over the past few decades. The foundation of the communication industry can be traced back to fixed land-line phones, however, as time progressed and the number of subscribers increased, telecommunications providers began to develop innovative technologies to satisfy the constantly evolving user demands. Following the introduction of first generation cellular technology, every generation thereafter has offered advanced technological capabilities and improved data rates.

The first generation (1G) of cellular networks began in the early 1980's and was introduced as an analog communication system that provided limited voice services. All of the standards in 1G used frequency modulation techniques for voice signals.

Mobile communication networks received their first major upgrade when they moved on to the second generation (2G). 2G was introduced in the late 1980's and was based on a digital signaling technique that improved spectrum efficiency and the speed at which data could be transferred. The most popular 2G wireless technology to oversee these changes was Global Systems for Mobile Communications (GSM), which also brought about SMS messaging. In order to serve multiple subscribers, GSM adopted Time Division Multiple Access (TDMA) technology, as it breaks down data transmission into fragments and transmits each fragment in a short burst, assigning each fragment a time slot. The evolved forms of GSM that followed, namely General Packet Radio Service (GPRS) and Enhanced Data Rates for GSM Evolution (EDGE), introduced the concept of packet switching in mobile networks, paving the way for the next generation.

The third generation (3G) symbolized another major progression for mobile wireless technology and set the standards for most of the wireless technology we have come to expect. As the core of its network architecture, 3G utilised the Universal Mobile Telecommunication System (UMTS), which was developed by the Third Generation Partnership Project (3GPP) and built on the GSM standard that was established with 2G. in order to fulfill the users’ ever increasing demand for data and service quality and therefore By increasing the data transmission rates and the data capacity, networks where now able to support multimedia applications.

The proliferation of sophisticated user platforms, such as smart-phones and tablets, and new bandwidth-intensive mobile applications further fueled the users desire for bandwidth and quality. This led to the next evolutionary leap towards the fourth generation (4G) of networks, which made mobile networks provide a true wireless broadband service to its customers, by transferring to an all-IP architecture. With the enhanced options offered by 4G, new use cases with diverse service requirements came into play in a plethora of sectors.

A summary of the differences between the generations of networks is presented in Table 1.

4G remains until today the standard for mobile high-speed wireless communication. LTE is a 4G wireless broadband technology developed by the 3GPP as the next step in the progression of mobile internet. Unlike its predecessors, LTE consists of two layers. The first is an IP connectivity layer, which handles all data, voice, video and messaging traffic, while the second handles the overall communication procedure. Furthermore, as a by-product of GSM and UMTS technologies, the majority of operators were able to upgrade infrastructure from past generations to support LTE technology [1].

LTE adopted orthogonal frequency division multiplexing (OFDM) as multiple access technology, in order to efficiently support wideband transmission. In addition, spectral efficiency was hugely improved with the use of multiple-input multiple-output (MIMO) techniques. Additional benefits of LTE include improved mobility, a dependable level of security, reduced latency and an improved experience for all users. LTE also has the capability to power cloud-based applications as live streaming and high-performance imaging.

Research and innovation have advanced the capabilities of mobile and wireless technology to the standard experienced today while also establishing a platform for 5G and beyond. 5G is currently under development and promises significantly faster data speeds, higher connection density as well as much lower latency, among other improvements. Some of the plans for 5G include device-to-device communication, better battery consumption, and improved overall wireless coverage. 5G has the potential to enable fundamentally new applications, industries, and business models and dramatically improve quality of life around the world.

These revolutionary networking capabilities brought forth with them an exciting new prospect, the idea of Internet of Things (IoT). The IoT term represents a general concept for the ability of network devices to sense and collect data from around the world, and then share that data across the Internet where it can be processed and utilized for various interesting purposes.

Table 1: Comparison of Mobile Generations

	1G	***2G***	***3G***	***4G***	***5G***
Deployment	1980-1990	1990-2001	2001-2009	Current	Future
Data Bandwidth	2 Kbps	64 Kbps	2 Mbps	200 Mbps to 1 Gbps	1 Gbps and higher
Technology	Analog	Digital	CDMA 2000, UMTS, EDGE	Wi-Max, Wi-Fi, LTE	WWWW Unified IP
Core Network	PSTN	PSTN	Packet N/W	Internet	Internet
Multiplexing	FDMA	TDMA/CDMA	CDMA	CDMA	CDMA
Switching	Circuit	Circuit/Packet	Packet	All Packet	All Packet
Primary Service	Analog Phone Calls	Digital Phone Calls and Messaging	High quality audio, video and data	All-IP Service (including Voice Messages)	Dynamic Information Access
Key differentiator	Mobility	Secure, Mass adoption	Better Internet experience	Faster Broadband Internet, Lower Latency	Better speed and coverage, Lower Latency
Weakness	Poor spectral efficiency, major security issue	Limited data rates, difficult to support demand for internet and e-mail	Real performance fail to match type, failure of WAP for internet access	Battery use is more, Required complicated and expensive hardware	Yet to be deployed

The main strength of IoT is the positive effects it could have on the daily life of both private citizens and business users. The possibilities are almost endless as it promises applications in a great range of fields. Possibly the most discussed area of application is that of smart cities, where intelligent systems will improve public infrastructures and services, reduce traffic congestion and provide a better public transportation experience. IoT will even play a part in humanity’s fight for the conservation of the environment, by allowing easy monitoring and management of resources. In the context of businesses, developments in the fields of automation, manufacturing and logistics could provide better economic returns and safer working conditions [2].

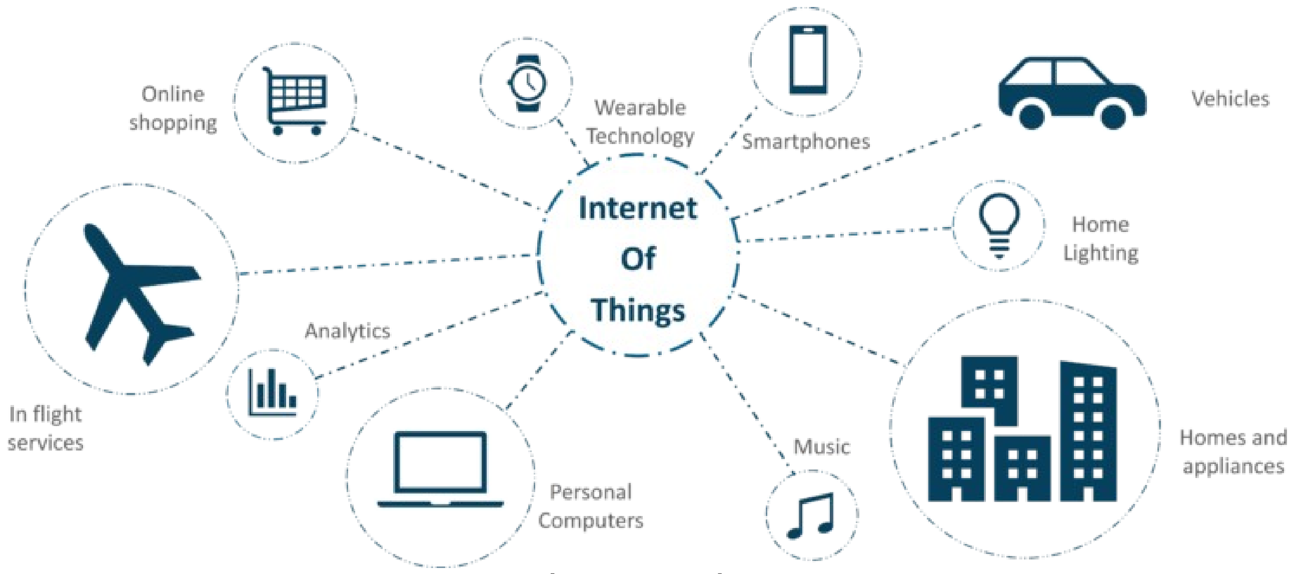


Figure 1: IoT Idea

It has become apparent that current networks have already began integrating multiple radio access technologies, which constitute a heterogeneous network. This drastic increase in network complexity has brought about many difficulties in computer network management. Providers must not only manage the ever expanding volume of data, but further ensure that customer service requests are being adequately fulfilled by the network, meeting the respective Quality of Service (QoS) requirements. Even more so in the case of multimedia applications, which take up a lot of bandwidth, heterogeneous networks are presented with the challenge to manage their respective load capacity. Software-Defined Networking (SDN) has been proposed as the solution to the load balancing issues between the Radio Access Networks (RAN).

SDN approaches were introduced as early as the mid-1990s, but just recently became a well-established industry standard. The aim of SDN is to deliver open interfaces that enable software development capable of controlling the connectivity of the devices included in a network, as well as the examination and modification of network traffic. By separating the control from the individual network devices, applications and network services are able to treat the network as a virtual entity. SDN offers flexible, dynamic, and programmable functionality of network systems, as well as many other advantages such as centralized control, reduced complexity, better user experience, and a dramatic decrease in network systems and equipment costs.

SDN is viewed as a key enabler for next-generation wireless networks and has the potential to fulfill some of the key requirements of IoT applications. Huge amounts of data will be generated, which will need to be processed and delivered in an efficient manner. SDN can help distribute and control the traffic flows and allocate network resources according to Quality of Service requirements. Additionally, SDN is capable of freeing RAN networks from the traditional protocol constraints. This would enable networks to serve multiple applications over their time, while also allowing an increase in the number of devices or network capabilities, without requiring an update in the firmware. Employing the SDN architecture would also pave the way for the concept of Network Function Virtualization (NFV) to enter into large scale IoT networks. Aside from empowering IoT networks with greater computing resources at the controller, NFV would allow the devices to alter their functions in real-time depending on application specific requirements [3].

1.2 Research Methodology

There is an enormous amount of literature in existence for all of the technologies involved in this thesis. The official documentation papers from the 3GPP, the LoRa Alliance and the Open Networking Foundation (ONF) contributed significantly to the research work. Additional research was done with the following methods: literature study of white papers, company perspective and journals, discussions with the supervisors and experimentation with the NS-3 tool. A substantial amount of attention had to be given to our simulation, as multiple issues arose because two of the three modules have not been officially included in the NS-3 supported module list and therefore present their own limitations.

1.3 Thesis Outline

The thesis is organized accordingly, with Chapter 2 giving the general synopsis of the LTE features and its emphasis on the basic structure and features. The purpose of Chapter 3 is to present an introductory technical overview of LoRa and LoRaWAN. Chapter 4 examines the architecture and working methodology of Software-Defined Networking and the OpenFlow protocol. Chapter 5 introduces the NS-3 simulation tool and some of its most important utilities. Continuing on, we showcase how LTE, OpenFlow and LoRaWAN have been modeled in NS-3. In chapter 6 we exhibit the topology of our experiment and investigate the possibility to control the network from a central node by updating flows rather than touching individual devices in the network. Lastly, in chapter 7 we draw conclusions from the results of our experiment and discuss future work potential.

2. LONG-TERM EVOLUTION

Cellular operators compete with traditional broadband operators by offering mobile broadband access and IP services such as rich multimedia (e.g., video-on-demand, music streaming) to mobile devices like laptops, smart-phones and other advanced handsets. They offer these services through access networks such as High-Speed Packet Access (HSPA), Evolution-Data Optimized (EV-DO) and Long-Term Evolution (LTE). These access networks deliver performance comparable to Asymmetric Digital Subscriber Line (ADSL) services, but with the added benefits of mobility and ubiquitous coverage. Furthermore, they offer mobile operators significantly improved data speeds, short latency and increased capacity.

LTE started with the third Generation Partnership Project (3GPP) release 8 and continued in release 10 with the objective of meeting the increasing performance requirements of mobile broadband [4]. Some key features of release 8 include: high spectral efficiency,very low latency, support of variable bandwidth, simple protocol architecture, and support for Self-Organizing Network (SON) operation. Release 10, otherwise known as LTE Advanced is a fourth generation (4G) specification that provides enhanced peak data rates to support advanced services and applications (100 Mb/s for high mobility and 1 Gb/s for low mobility). Table 2 explains some of the new features introduced with every new 3GPP Release.

In contrast to the circuit-switched model of previous cellular systems, LTE has been designed to support only packet-switched services. It aims to provide seamless Internet Protocol (IP) connectivity between User Equipment (UE) and the Packet Data Network (PDN), without any disruption to the end users' applications during mobility.

The LTE protocol architecture is made up of two planes: the User Plane (UP), which provides functions such as formatting user traffic between User Equipment and the Evolved Universal Terrestrial Radio Access Network (E-UTRAN); and the Control Plane (CP), which supports functions used for control purposes such as network authentication.

There are only two types of nodes in the UP, Base Stations and Gateways, in contrast to previous hierarchical networks which had four types (Node B, RNC, SGSN, GGSN). The gateway consists of two logical UP entities, Serving Gateway and Packet Data Network Gateway (PDN-GW), collectively called the SAE-GW. This flat architecture with less involved nodes reduces latencies and improves performance.

While the term LTE encompasses the evolution of the Universal Mobile Telecommunications

System (UMTS) radio access through the E-UTRAN, it is accompanied by an evolution of the non-radio aspects under the term “System Architecture Evolution” (SAE), which includes the Evolved Packet Core (EPC) network. The SAE architecture and concepts have been designed for efficient support of mass-market usage of any IP-based service and is based on an evolution of the existing GSM/WCDMA core network, with simplified operations and smooth, cost-efficient deployment. Together LTE and SAE comprise the Evolved Packet System (EPS).

Table 2: Evolution of 3GPP Releases

Release	***Functional Freeze***	***Main Radio Features of the Release***
Rel-99	March 2000	3G UMTS incorporating WCDMA radio access
Rel-4	March 2001	UMTS all-IP Core Network
Rel-5	June 2002	IMS and HSDPA
Rel-6	March 2005	HSUPA, MBMS, IMS enhancements
Rel-7	December 2007	Improvements in QoS & latency, VoIP, HSPA+, EDGE Evolution
Rel-8	December 2008	Introduction of LTE, SAE, OFDMA, MIMO, Dual Cell HSDPA
Rel-9	December 2009	WiMAX/LTE/UMTS interoperability, Dual Cell HSDPA with MIMO, Dual Cell HSUPA, LTE HeNB
Rel-10	March 2011	LTE-Advanced (4G), Multi-Cell HSDPA
Rel-11	September 2012	Heterogeneous Networks, Coordinated Multipoint, Advanced IP Interconnection of Services
Rel-12	September 2014	Enhanced Small Cells, 3D channel modeling, MTC-UE Cat, eMBMS enhancements
Rel-13	December 2015	LTE-U / LTE-LAA, LTE-M, Full Dimension MIMO LTE-M Cat, Indoor positioning
Rel-14	March 2017	Energy Efficiency, Location Services, Mission Critical Data over LTE, MBPS, CBS
Rel-15	March 2019	New Radio, Support for 5G Vehicle-to-x service, IP Multimedia Core Network Subsystem

EPS uses the concept of EPS bearers to route IP traffic from a gateway in the PDN to the UE. A bearer is an IP packet flow with a defined QoS between the gateway and the UE. The

E-UTRAN and EPC together set up and release bearers as required by applications.

A default EPS bearer is established when UE connects to a PDN and it remains established throughout the lifetime of the connection, to provide the UE with always-on IP connectivity. Any additional EPS bearer that is established for the same PDN connection is referred to as a dedicated bearer.

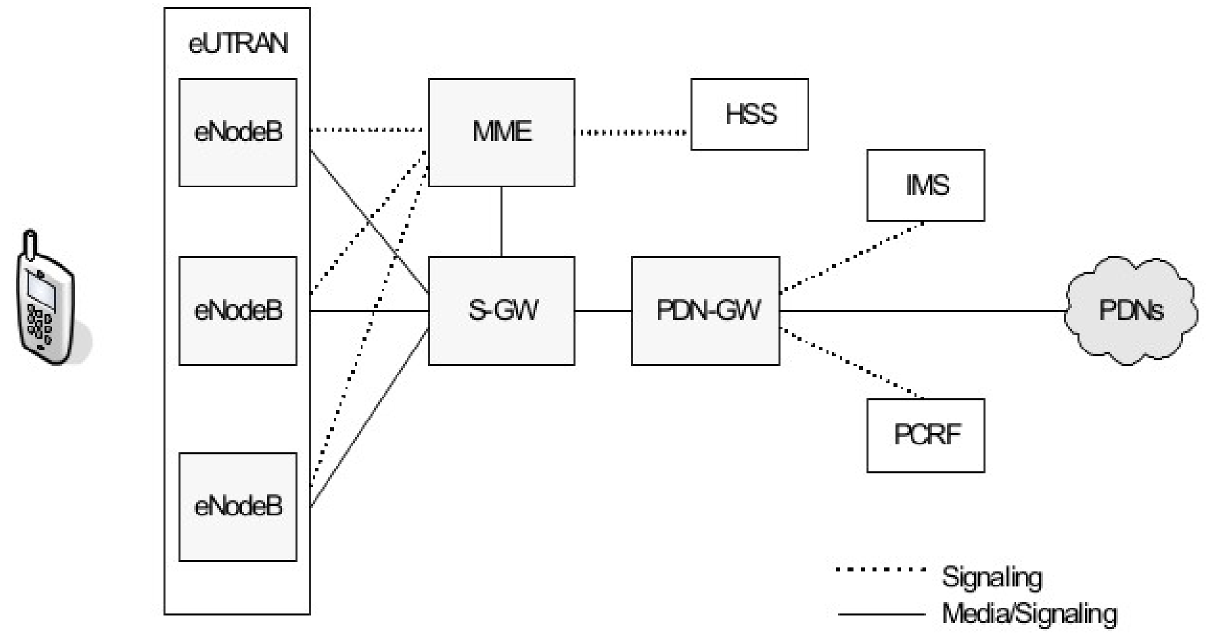


Figure 2: LTE High-Level Network Architecture

2.1 Network Architecture

At a high level, the network is comprised of the Control Network (called EPC in SAE) and the Radio Access Network E-UTRAN. While the Control Network consists of many logical nodes, the Radio Access Network is made up of essentially just one node, the evolved NodeB (eNodeB), which connects to the UE. Each of these network elements is interconnected by means of interfaces that are standardized in order to allow multi-vendor interoperability.

2.1.1 Core Network

The EPC is composed of several functional entities that are responsible for the overall control of the UE and bearers establishment. The core network is formed from the following nodes:

• Mobility Management Entity (MME): The MME is the control node that processes the signaling between the UE and the CN. The protocols running between the UE and the CN are known as the Non Access Stratum (NAS) protocols. Its main function is UE location management, as it tracks and maintains the current location of the UE. In addition, MME handles functions related to bearer management, which involves signaling procedures used to set up packet data context and negotiate associated parameters like the QoS.

• Packet Data Network Gateway (P-GW): The P-GW connects UE to external PDNs and acts as the UE default router to the Internet. It is also responsible for IP address allocation for the UE, as well as QoS enforcement and flow-based charging according to rules from the PCRF. By using the the Traffic Flow Templates (TFTs), it is able to filter downlink user packets into different QoS based bearers. It also serves as the mobility anchor to support seamless mobility of the UE between LTE and trusted non-3GPP networks [5].

• Serving Gateway (S-GW): All user IP packets are transferred through the S-GW, which serves as the local mobility anchor for the data bearers when the UE moves between eNodeBs. It also serves as the mobility anchor for interworking with other 3GPP technologies such as 2G/GSM and 3G/UMTS. It retains the information about the bearers when the UE is in the idle state and temporarily buffers downlink data while the MME initiates paging of the UE to reestablish the bearers. In addition, the S-GW performs some administrative functions involving lawful interception and accounting of user data.

• Policy and Charging Resource Function (PCRF): The PCRF is responsible for policy control decision making, as well as for controlling the flow-based charging functionalities in the Policy Control Enforcement Function (PCEF), which resides in the P-GW. The PCRF provides the QoS authorization that decides how a certain data flow will be treated in the PCEF and ensures that this is in accordance with the user's subscription profile. It is a server usually located with other core network elements in the operator switching centers.

• Home Subscription Server (HSS): The HSS functions as the database for storing users subscription data such as QoS profile and any access restrictions for roaming. It also holds information about the PDNs to which the user can connect to. In addition, the HSS holds dynamic information such as the identity of the MME to which the user is currently attached or registered. Additionally, the HSS may house the authentication center (AUC), which generates the vectors for authentication and security keys.

2.1.2 Radio Access Network

The access network of LTE, E-UTRAN, usually consists of a network of eNodeBs. For normal user traffic there is no centralized controller in E-UTRAN. Due to the absence of a network controller, it is said to have a flat architecture. This structure reduces system complexity and cost and allows better performance over the radio interface. One consequence of the lack of a centralized controller node is that, as the UE moves, the network must transfer all information related to a UE, that is, the UE context, together with any buffered data, from one eNodeB to another. Mechanisms are therefore needed to avoid data loss during handover.

As shown in figure 2, the eNodeBs are interconnected with each other by means of an interface known as “X2” and to the EPC by means of the S1 interface — more specifically, to the MME by means of the S1-MME interface and to the S-GW by means of the S1-U interface [6].

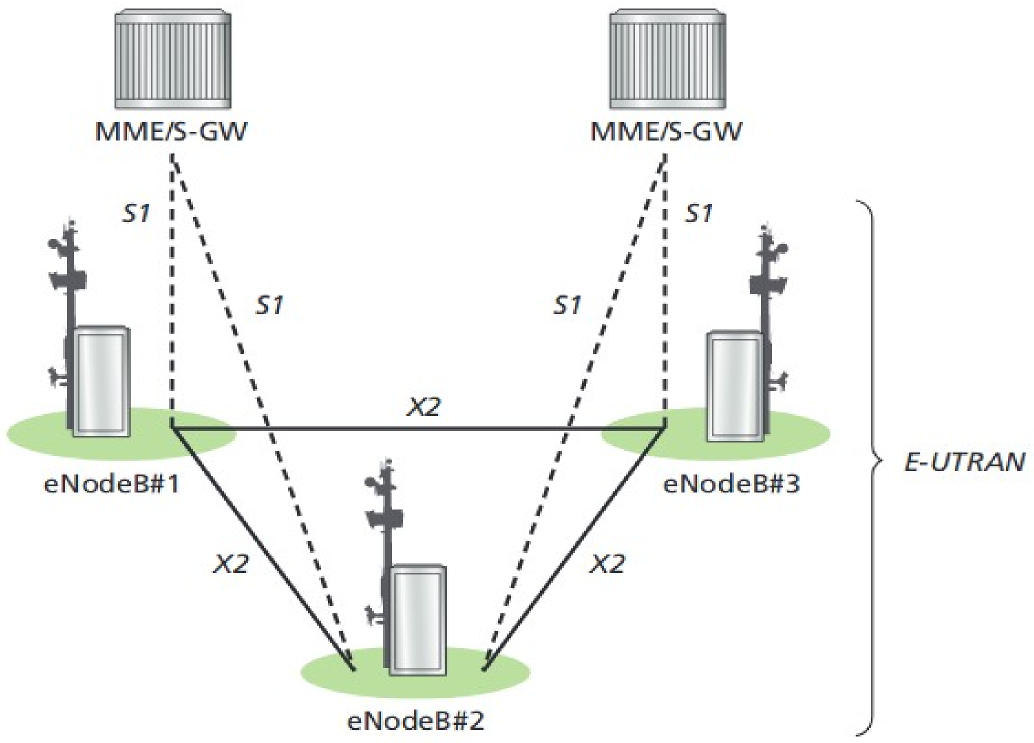


Figure 3: E-UTRAN Network Architecture

The E-UTRAN is responsible for all radio-related functions, which can be summarized briefly as:

- Radio Resource Management: This covers all functions related to the radio bearers, such as radio bearer control, radio admission control, radio mobility control, scheduling and dynamic allocation of resources to UEs in both uplink and downlink.

• Header Compression: This helps to ensure efficient use of the radio interface by compressing the IP packet headers that could otherwise represent a significant overhead.

• Security: Security-related network functions ensure that all data sent over the radio interface is encrypted.

• Connectivity to the EPC: This consists of the signaling toward MME and the bearer path toward the S-GW.

On the network side, all of these functions reside in the eNodeBs, each of which can be responsible for managing multiple cells. Unlike some of the previous second and third generation technologies, LTE integrates the radio controller function into the eNodeB. This allows tight interaction between the different protocol layers of the E-UTRAN, thus reducing latency and improving efficiency.

2.2 Protocol Architecture

An overview of the protocol stack for the UP and CP can be viewed at Figure 3 [7].

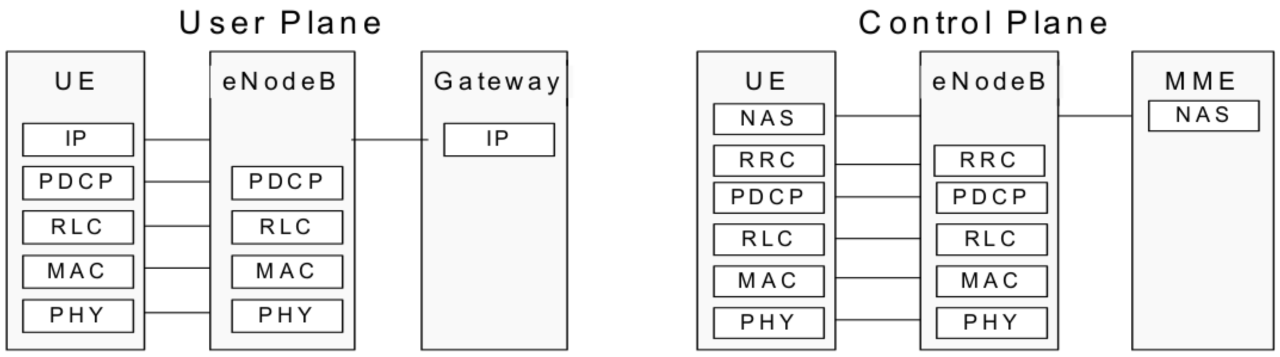


Figure 4: E-UTRAN Protocol Stack

The topmost layer in the CP is the Non-Access Stratum (NAS) [8], which consists of two separate protocols that are carried on direct signaling transport between the UE and the MME. The content of the NAS layer protocols is not visible to the eNodeB, and the eNodeB is not involved in these transactions by any other means, besides transporting the messages and providing some additional transport layer indications along with the messages in some cases. The NAS layer protocols are:

• EPS Mobility Management (EMM): This protocol is responsible for handling the UE mobility. It includes functions for attaching to and detaching from the network, and performing location updating in between. Note that the handovers in connected mode are handled by the lower layer protocols, but the EMM layer does include functions for re-activating the UE from idle mode. The UE initiated case is called Service Request, while Paging represents the network initiated case. Authentication and protecting the UE identity,are also part of the EMM layer, as well as the control of NAS layer security functions, encryption and integrity protection.

• EPS Session Management (ESM): This protocol may be used to handle the bearer management between the UE and MME, and it is used in addition for E-UTRAN bearer management procedures.

The radio interface protocols are:

• Radio Resource Control (RRC): This protocol is in control of the radio resource usage. Some of the main services and functions of this sub-layer include: mobility control, QoS management functions, radio bearer control, connection management and measurement control.

• Packet Data Convergence Protocol (PDCP): Implements functions such as robust packet header compression and decompression, ciphering and deciphering, transfer of user data. The main functions of the PDCP for the CP involve ciphering and integrity protection and transfer of CP data.

• Radio Link Control (RLC): The RLC protocol is responsible for the transportation of the PDCP-PDUs. It can work in 3 different modes depending on the reliability provided. Depending on this mode it can provide: ARQ error correction, segmentation/concatenation of PDUs, reordering for in-sequence delivery, duplicate detection,

• Medium Access Control (MAC): The MAC sublayer offers a set of logical channels to the RLC sublayer that it multiplexes into the physical layer transport channels. It also manages the Hybrid ARQ error correction, handles the prioritization of the logical channels for the same UE and the dynamic scheduling between UEs.

• Physical Layer (PHY): The function of the PHY sub-layer is to provide data transport services on physical channels to the upper RLC and MAC sub-layers [9].

Regarding the UP, below the end user IP, the protocol structure is very similar to the CP. This highlights the fact that the whole system is designed for generic packet data transport.

3. INTERNET OF THINGS

It is difficult to give a specific definition for the Internet of Things. The term first appeared about 20 years ago at the Massachusetts Institute of Technology, when the Auto-ID Labs were studying networked radio frequency identification (RFID) and emerging sensing technologies. These labs would end up designing the prototype architecture for IoT. With the passage of time and a further grasp of the possibilities presented, the concept of IoT has surpassed the scope of simple RFID technologies. Nowadays, IoT allows millions of devices to be connected, measured and monitored to automate processes and operations and support better decision making.

As interest in the field of Internet of Things rose drastically, the need to standardize the communication protocols and techniques became apparent. IoT applications require technologies that can offer low power consumption, low cost and low complexity devices that are able to communicate wirelessly over very long distances.

The world of Internet of Things consists of different technological layers that all play a critical role in the route from connecting IoT devices to creating applications that serve a predefined purpose. These include industry-grade IoT projects and consumer applications. The four tiered layered architecture can be viewed in figure 5 [10].

The bottom layer, called the sensing layer, consists of sensor connected IoT devices. These are small, memory-constrained, often battery-operated electronics devices with onboard sensors and actuators. Their main capabilities include collecting and communicating data. They can either function as standalone sensing devices or be embedded as part of a bigger machinery for sensing and control.

The second layer is responsible for the secure transmission of the sensor data towards the upper layer. IoT gateways are adopted to achieve this task. These gateways are critical components of the IoT ecosystem, as they offer data encryption and connectivity aggregation. Typically, IoT gateways are equipped with multiple communication capabilities to communicate with the IoT devices on one end and an IP connection for the opposite side.

The third layer stores all of the data relayed by the gateways in a server. These servers accept, store and process data for analysis and decision making. Multiple data science and analytics techniques are employed to make sense of the data and enable corrective action.

The application layer is considered as the top layer of conventional IoT architecture. This layer constitutes the front end of the whole IoT architecture. By exploiting the processed data from the previous layer, personalized based services can be provided according to user relevant needs.

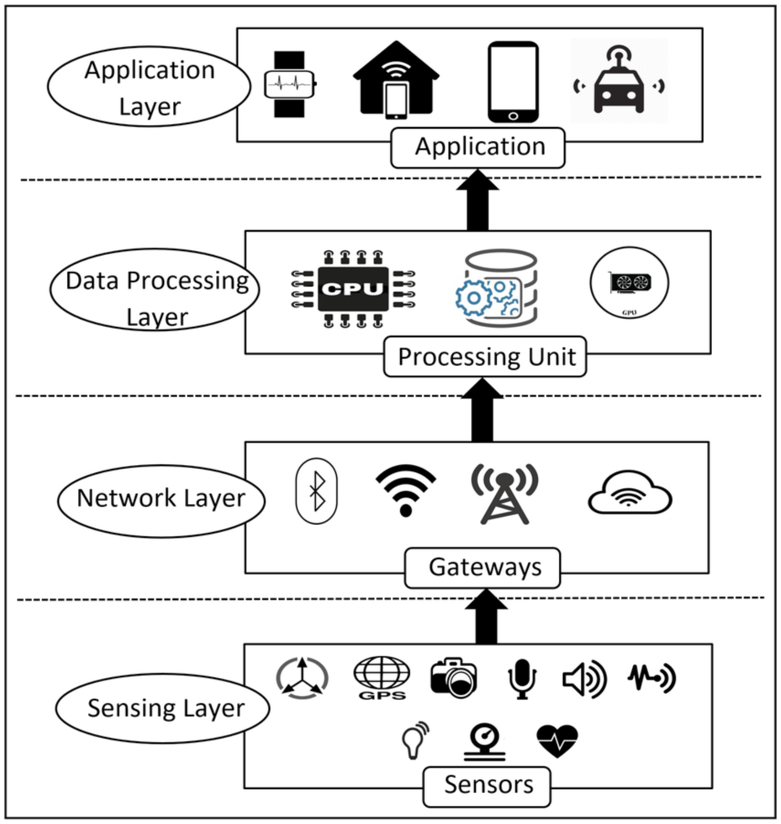


Figure 5: IoT Architecture Overview

Current short-range radio technologies are not adapted for scenarios that require long range transmission, whilst cellular technologies can provide larger coverage, however they consume excessive amounts of device energy. Therefore, IoT applications' requirements have driven the emergence of new wireless communication protocols and corresponding hardware.

Possibly the most popular IoT implementation is called LoRa, which can be split into two parts, LoRa modulation and LoRaWAN. The former is a protocol for the physical layer in the OSI model developed by Semtech. The latter is a network architecture defined by the LoRa Alliance, a non-profit technology consortium committed to enabling large scale deployment of IoT through the development of the LoRaWAN open standard [11].

3.1 Low Power Wide Area Networks

Low Power Wide Area Networks (LPWAN) are increasingly gaining popularity in industrial and research communities because of their power efficient, long range and low cost communication characteristics. Globally, they are becoming the preferred connectivity mode for low cost, low power assets, particularly in hard-to-reach locations such as rural areas, underground locations or at the margins of mobile cellular coverage.

Development of LPWAN started in 2011, when SigFox first introduced the technology in 2012 as an alternative to IoT connectivity. Since then, there have been significant developments in LPWAN technology making it more efficient, scalable, and flexible.

LPWAN technologies are designed for a wide area coverage and an excellent signal propagation to hard-to-reach indoor places. This allows the end devices to connect to the base stations at a distance ranging from a few to tens of kilometers depending on their deployment environment. With an exception of a few LPWAN technologies, most use Sub-GHz band, which offers robust and reliable communication at low power budgets.

A major factor contributed to the rise of LPWAN is its low cost. Non-cellular LPWAN require limited infrastructure development and operate on unlicensed spectrum, providing an excellent alternative to the cellular network. In addition, the advances in the hardware design and the simplicity of the end-devices makes LPWANs economically viable.

To accommodate as many connected devices as possible, efficient exploitation of diversity in channel, time, space, and hardware is vital. Due to low-power and inexpensive nature of the end devices, much of this is achieved by cooperation from more powerful components in LPWAN such as base stations and backend systems. LPWAN technologies employ multi-channel and multi-antenna communication to parallelize transmissions to and from the devices, while also allowing adaptive transmission power control and channel selection to ensure reliable transmissions. The extent to which adaptive channel selection and modulation is possible depends on the underlying LPWA technology.

Several LPWAN technologies are already present in the market [12]. SigFox plans to offer global coverage by means of a single operator network, with roll-outs in different countries performed by a number of member companies. Narrow Band IoT (NB-IoT) is being offered by telecommunication companies as an IoT communication alternative to sub-GHz LPWAN technologies. As NB-IoT operates in licensed spectrum, it offers higher traffic reliability compared to other sub-GHz technologies. Unlike SigFox and NB-IoT, LoRaWAN offers the possibility for private network deployments and easy integration with a number of world-wide network platforms. Due to this and its open access specifications, LoRaWAN drew the research community’s attention since its first appearance in the market.

There is a lot of activity in the IoT sector comparing LPWAN options both from a technical comparison but also from a business model perspective. LPWAN networks are being deployed now because there is a strong business case to support immediate deployment and the cost to deploy the network in unlicensed bands requires much less capital than even a 3G software upgrade. Many factors should be considered when choosing the appropriate LPWAN technology for an IoT application including quality of service, battery life, latency, scalability, coverage, range, deployment, and cost.

Table 3: Comparison of LPWAN Technologies

Feature	***LoRaWAN***	***Narrow-Band***	***LTE Cat-1***	***LTE Cat-M***	***NB-LTE***
Modulation	SS Chirp	UNB/GFSK/BPSK	OFDMA	OFDMA	OFDMA
Rx Bandwidth	500–125KHz	100 Hz	20 MHz	20 – 1.4 MHz	200 KHz
Data Rate	290 bps – 50 Kbps	100 bps	10 Mbps	200 Kbps – 1 Mbps	20 Kbps
Messages/day	Unlimited	140 per day	Unlimited	Unlimited	Unlimited
Output Power	20 dBm	20 dBm	23 – 46 dBm	23 – 30 dBm	20 dBm
Link Budget	154 dB	151 dB	130 dB	146 dB	150 dB
Battery Life	105 months	90 months		18 months	
Power Efficiency	Very High	Very High	Low	Medium	Med High
Interference Immunity	Very High	Low	Medium	Medium	Low
Coexistence	Yes	No	Yes	Yes	No
Security	Yes	No	Yes	Yes	Yes
Mobility / Localization	Both	Limited Mob No Loc	Mobility	Mobility	Limited Mob No Loc

3.2 LoRa Modulation

LoRa is the physical layer protocol, typically used in conjunction with the LoRaWAN MAC-layer protocol. Unlike the LoRaWAN protocol, which is open source, the LoRa protocol is a proprietary protocol developed by Semtech. Because of this, detailed documentation about the design is not readily available. However, some pieces of information about the protocol have been revealed by Semtech documents and subsequently the modulation has been reverse engineered to a point where the implementation of the protocol is considered well understood. LoRa is a PHY layer implementation and is agnostic to higher-layer implementations. This allows LoRa to coexist and interoperate with existing network architectures.

Many legacy wireless systems use frequency shifting keying (FSK) modulation as the physical layer because it is a very efficient modulation for achieving low power. LoRa utilizes a spread spectrum technique called Chirp Spread Spectrum (CSS), which maintains the same low power characteristics as FSK modulation but significantly increases the communication range. In CSS a chirp is used to spread data over a wider spectrum than it would normally be capable of. A chirp is a signal of linearly varying frequency and fixed duration. The frequency is uniformly distributed over a larger bandwidth to provide resistance to frequency selected noise. These chirps are often referred to as up-chirps, if they are continuously increasing in frequency, or down-chirps if they are continuously decreasing.

Next to bandwidth, the spreading factor is a second important parameter of the LoRa modulation as it provides the flexibility of trading range for data rate. The spreading factor can range from seven to twelve and in conjunction with coding rates dictate the achievable data rates. The nominal bit rate can be calculated as follows:

$$R_b = SF \cdot \frac{\frac{4}{4+CR}}{\frac{2^{SF}}{BW}} \cdot 1000$$

where SF is the spreading factor, CR is the code rate and BW is the bandwidth in KHz. The data rate in this formula is expressed in bps.

As the spreading factor increases the symbol rate is lowered, thereby trading reduced data rate for increased range. The different spreading factors are orthogonal. This means that a LoRa gateway can receive multiple transmissions on different spreading factors simultaneously. Note that SF bits are mapped per LoRa symbol.

In order to further increase the robustness of the LoRa modulation, additional techniques such as forward error correction (FEC) and diagonal interleaving are employed. Hamming codes, a simple linear block code algorithm, is used for the FEC. Additionally, LoRa radios operate in the sub-GHz unlicensed bands as they provide a good trade-off between available unlicensed spectrum and reduced path loss. The combination of these design considerations results in a high link budget at the expense of data rate, which means that LoRa transmissions can still be received successfully even though they are below the noise floor.

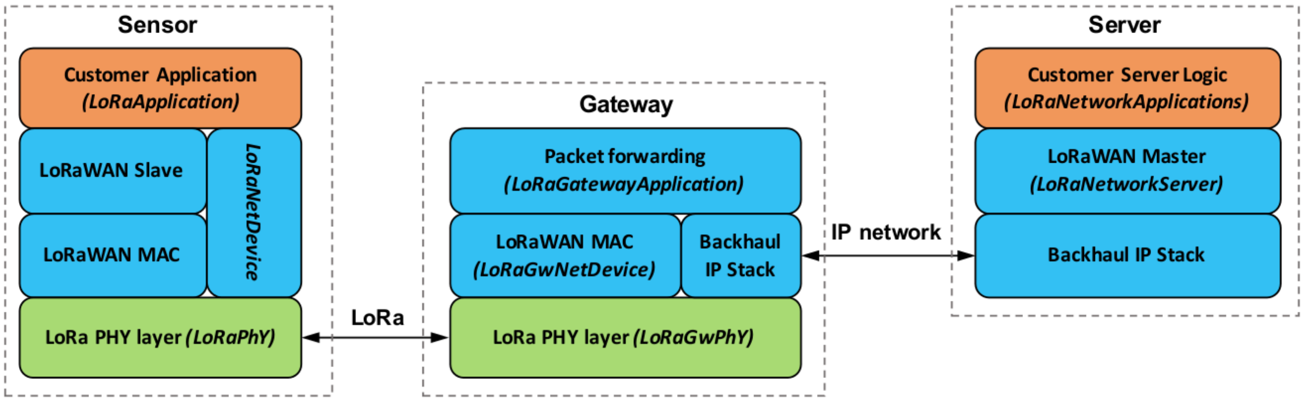

Sensor
Customer Application (LoRaApplication)
LoRaWAN Slave
LoRaWAN MAC
LoRaNetDevice
LoRa PHY layer (LoRaPhY)
LoRa
Gateway
Packet forwarding (LoRaGatewayApplication)
LoRaWAN MAC (LoRaGwNetDevice)
Backhaul IP Stack
LoRa PHY layer (LoRaGwPhY)
IP network
Server
Customer Server Logic (LoRaNetworkApplications)
LoRaWAN Master (LoRaNetworkServer)
Backhaul IP Stack

Figure 6: LoRa Network Structure

3.3 LoRaWAN

LoRaWAN is an open source LPWAN communication protocol standardized by the LoRa Alliance, that runs on top of the LoRa physical layer. It is designed from the bottom up to optimize the battery lifetime of a node, the capacity, quality of service and security of the network, while keeping network structures and management simple.

Most existing networks utilize a mesh network architecture, where the end nodes share data between them to increase the communication range of the network. This, however, increases the network complexity, reduces network capacity and quickly depletes the battery life of the nodes, as they have to stay awake to await for packets.

A LoRaWAN network consists of at least one network server, gateways and end devices. End devices are LoRa embedded sensors that produce data to be transferred to a network server. All of these entities are deployed in a star-of-stars topology, in which messages between end devices and a central network server are relayed by gateways. Each end device is not associated with one specific gateway. Instead, data transmitted by a device will typically be received by multiple gateways. The gateways are connected to the network server via standard IP connections and act as a transparent bridge, simply converting RF packets to IP packets and vice versa. The wireless communication takes advantage of the long range characteristics of the LoRa physical layer, allowing a single-hop link between the end devices and one or more gateways. All modes are capable of bi-directional communication, and there is support for multicast addressing groups to make efficient use of spectrum during tasks which require mass distribution messages.

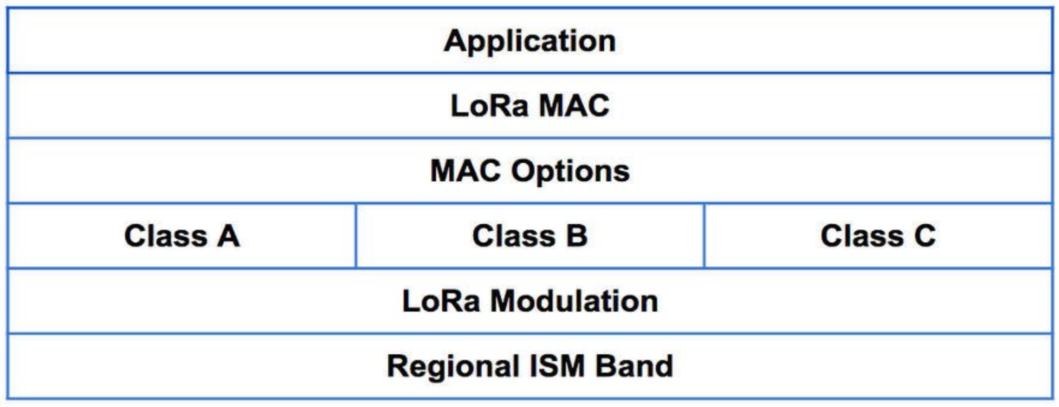


Figure 7: LoRaWAN Node Architecture

In addition to frequency hopping, all communication packets between end devices and gateways also include a variable Data rate setting. An Adaptive Data Rate (ADR) mechanism is built into LoRaWAN to dynamically manage an end device's data rate and transmit power, thus increasing the packet delivery ratio. A device can allow the network server to manage these parameters, by setting the uplink ADR bit in its uplink communication packets. If a device wants to set the transmit parameters itself, it does so by making use of the ADR mechanism that resides at the end device side. For this reason, both parts of the ADR mechanism run asynchronously at the network server and at the end node.

The LoRaWAN standard defines three classes of end devices to address the different requirements presented by an application. These are classified as class A, B and C [13]. The device classes trade off network downlink communication latency versus battery lifetime. In a control or actuator-type application, the downlink latency is an important factor.

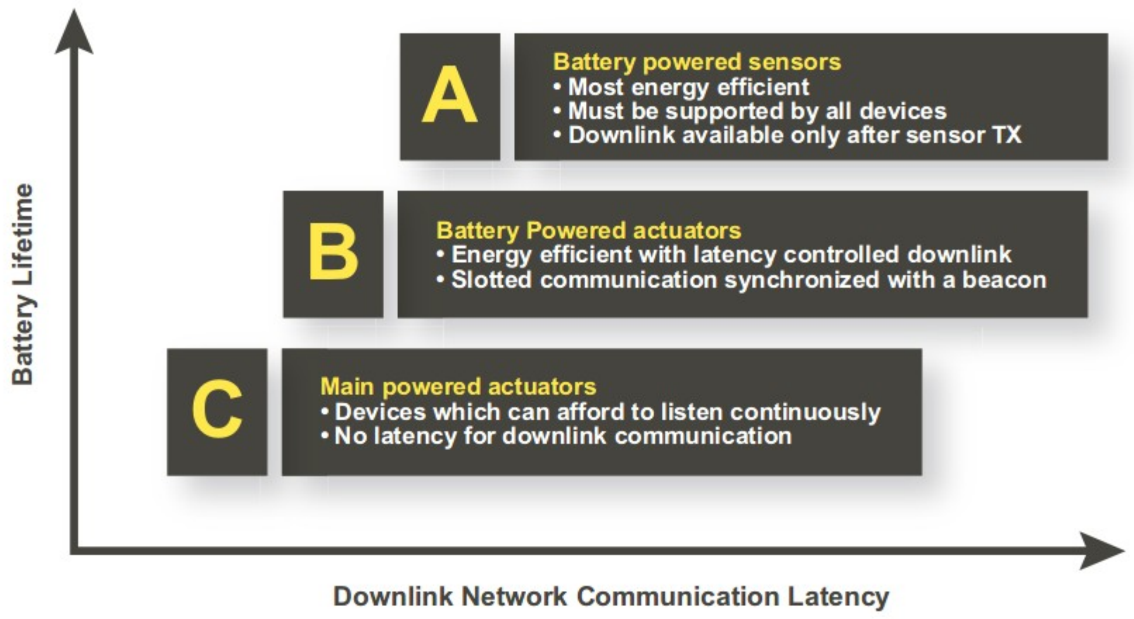

A
Battery powered sensors
• Most energy efficient
• Must be supported by all devices
• Downlink available only after sensor TX
B
Battery Powered actuators
• Energy efficient with latency controlled downlink
• Slotted communication synchronized with a beacon
C
Main powered actuators
• Devices which can afford to listen continuously
• No latency for downlink communication
Battery Lifetime
Downlink Network Communication Latency

Figure 8: LoRaWAN End Device Classes

Class A is the default class and must be supported by an end devices, in order for it to join a LoRaWAN network. These devices enable bi-directional communications whereby each uplink transmission is followed by two short downlink receive windows, during which they listen for possible downlink traffic. Because downlink transmission is not possible outside of those two receive windows, at any other time the downlink communication will have to be buffered at the network server until the next scheduled uplink. Class A is the lowest power operating mode as there is no network requirement for periodic wake-ups, while still allowing uplink communication at any time.

To increase the downlink possibilities, Class B end devices will open additional receive windows at scheduled times. In order for the end-device to open its receive window at the scheduled time, it receives a time-synchronized beacon from the gateway. Class B devices consume more power compared to Class A devices, as they need to open more receive windows, even if those windows might not be used for downlink traffic at all.

Class C further decreases downlink latency by keeping the receive windows continuously open, except when the devices are transmitting.

For battery powered devices, temporary mode switching between classes A and C is possible and is useful for intermittent tasks such as firmware over-the-air updates.

4. SOFTWARE-DEFINED NETWORKING

Software-Defined Networking has been designed to enable more agile and cost-effective networks. In the SDN architecture, the control and data planes are decoupled, network intelligence and state are logically centralized, and the underlying network infrastructure is abstracted from the applications. By centralizing network intelligence, decision-making is facilitated based on a global (or domain) view of the network, as opposed to today's networks, which are built on an autonomous system view where nodes are unaware of the overall state of the network [14] .

The Open Networking Foundation (ONF) is taking the lead in SDN advancement and architecture standardization with the OpenFlow protocol [15]. Figure 9 depicts a logical view of the SDN architecture as defined by the ONF.

4.1 SDN Architecture

This SDN architecture consists of three distinct layers that are accessible through open Application Program Interfaces (APIs):

• The Application Layer consists of a variety of applications, which can be developed to take advantage of the provided network information to provide end-to-end solutions for businesses and data centers.

• The Control Layer provides the consolidated control functionality that supervises the network behavior through use of an SDN controller. The SDN controller is a logical entity in charge of relaying instructions from the top layer down to the networking components. Furthermore, it is responsible of providing the SDN applications with an abstract view of the network for decision making purposes.

• The Infrastructure Layer is composed of the networking devices which are in charge of the forwarding and data processing capabilities for the network.

The SDN architecture APIs are referred to as Northbound and Southbound interfaces. The connection between the controller and applications is defined as a Northbound interface and typically provides abstract network views and enables direct expression of network behavior and requirements. On the other hand, Southbound interface is the connection between the controller and the physical networking hardware and facilitates efficient control over the network and enables the controller to dynamically make changes according to real-time demands and needs. Because SDN is a virtualized architecture, these elements do not have to be physically located in the same place.

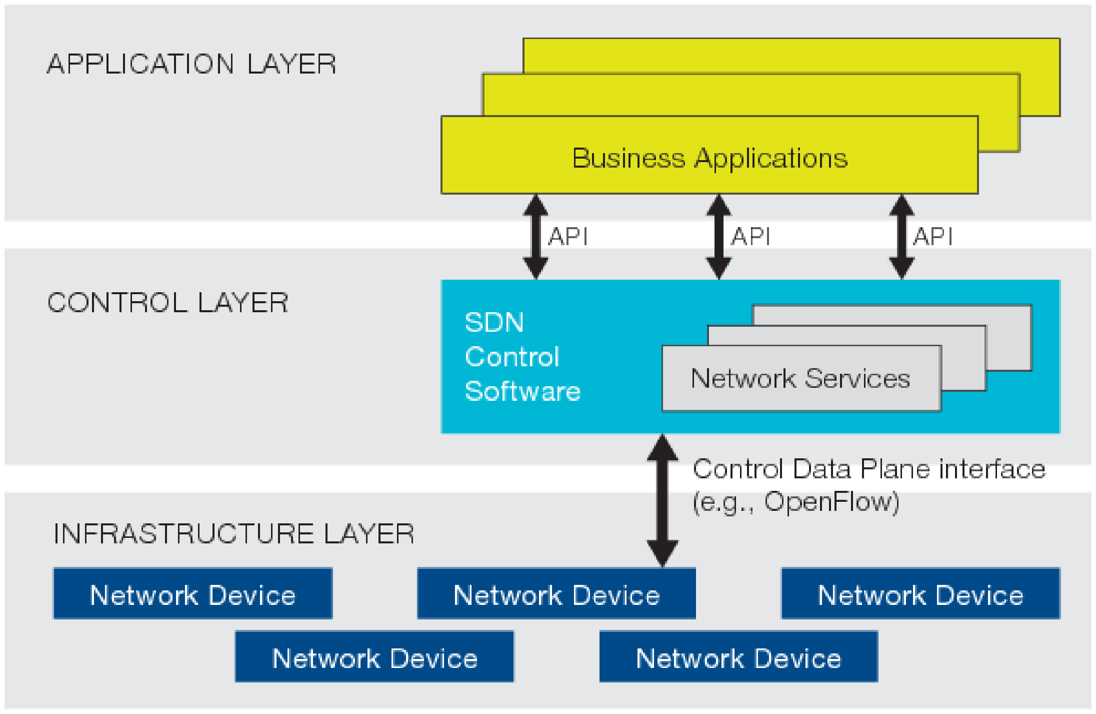


Figure 9: ONF / SDN Architecture

Network intelligence is centralized at the control layer, in a software-based controller. The controller can maintain a network global view, thus providing real-time information and fast optimized routing. By centralizing the network state, SDN provides the flexibility to configure, manage, secure, and optimize network resources via dynamic, automated SDN programs. Such programmability enables network configuration to be automated, influenced by the rapid adoption of the cloud. By providing open APIs for applications to interact with the network, SDN networks can achieve unprecedented innovation and differentiation [16].

4.2 OpenFlow Protocol

The OpenFlow protocol is the first and most well-known southbound interface designed for SDN networks, providing high-performance and granular traffic control across multiple network devices from different vendors [17]. OpenFlow uses the concept of flows to identify network traffic based on predefined match rules that can be statically or dynamically programmed by the SDN controller. Both the network infrastructure devices and the SDN controller need to implement OpenFlow specifications so that they can communicate through the use of OpenFlow messages. The current specification covers the components and the basic functions of the switch, and the OpenFlow protocol to manage an OpenFlow switch from a remote OpenFlow controller. Figure 3 shows the main components of an OpenFlow switch [18].

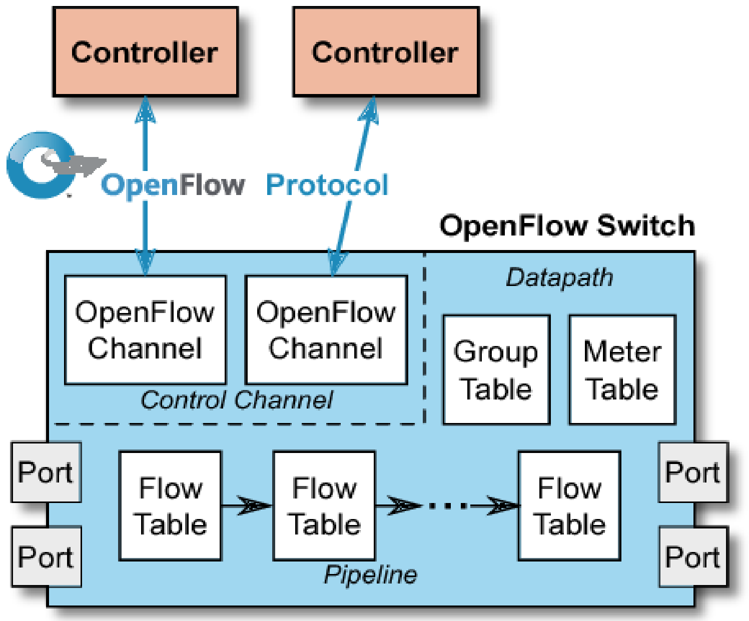
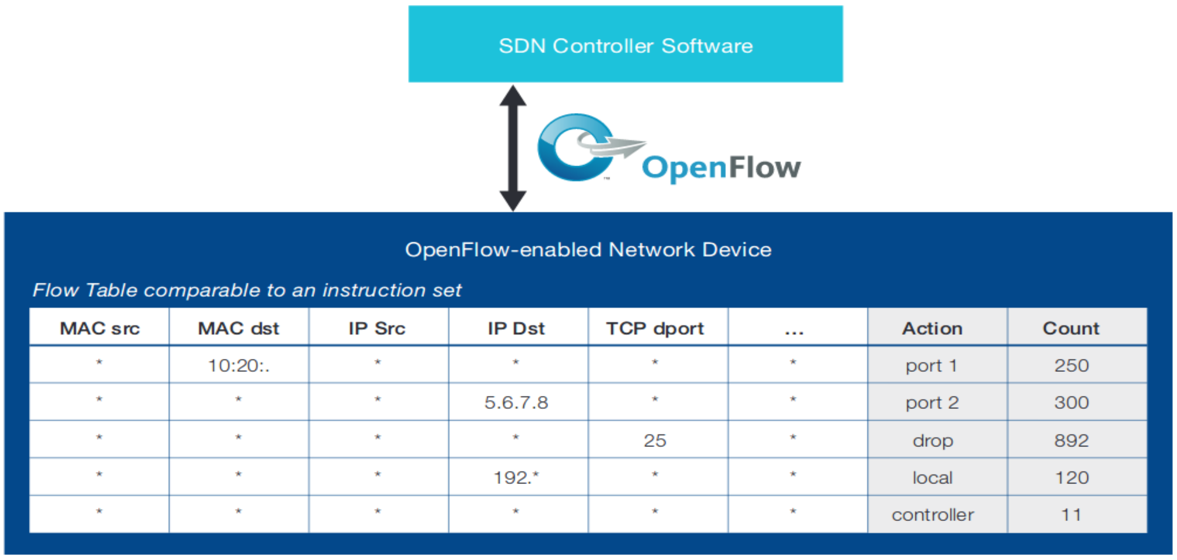


Figure 10: OpenFlow Switch Architecture

The OpenFlow channel is the interface that connects each OpenFlow switch to an OpenFlow controller. Through this interface, the controller configures and manages the switch, receives events and sends packets out to the network. The control channel of the switch may support a single OpenFlow channel with a single controller, or multiple OpenFlow channels enabling multiple controllers to share management of the switch. The OpenFlow channel is usually encrypted using Transport Layer Security (TLS), but may be run directly over Transmission Control Protocol (TCP). All OpenFlow channel messages must be formatted according to the OpenFlow protocol.

The OpenFlow switch datapath consists of a pipeline of one or more flow tables, which perform packet lookups and forwarding, based on flow entries configured by the controller. Each flow entry consists of OpenFlow eXtensible Match (OXM) Type-Length-Value (TLV) fields to identify the flow, counters, and a set of instructions to apply to matching packets. The matching starts at the first pipeline flow table, following order of priority, and may continue to next tables. If a matching entry is found, the instructions associated with the specific flow entry are applied. Instructions can modify the pipeline processing, sending the packet to one of the following tables, or can contain actions that describe packet forwarding, packet modification, and group table processing. The most common action is the output action, which forwards the packet to an output port. Another available action is the group action, which directs the packets to another switch datapath element that is called group table. Groups represent sets of actions for flooding, as well as more complex forwarding semantics. The switch can also send unmatched packets to the controller, or simply drop the packet.


SDN Controller Software
OpenFlow
OpenFlow-enabled Network Device
Flow Table comparable to an instruction set

MAC src	MAC dst	IP Src	IP Dst	TCP dport	...	Action	Count
*	10:20:.	*	*	*	*	port 1	250
*	*	*	5.6.7.8	*	*	port 2	300
*	*	*	*	25	*	drop	892
*	*	*	192.*	*	*	local	120
*	*	*	*	*	*	controller	11

Figure 11: OpenFlow Flow Table Example

Another OpenFlow switch datapath element is the meter table, consisted of meter entries defining per-flow meters. Per-flow meters enable OpenFlow to implement rate-limiting, a simple QoS operation constraining a set of flows to a chosen bandwidth. A switch can also optionally have one or more queues attached to a specific output port, and in many cases, those two features can be combined to implement complex work conserving QoS frameworks.

Extended details on the protocol operation can be found on the official OpenFlow Switch specifications [19].

5. NS-3 SIMULATOR

The basic idea of computer simulation is to use virtual models to study the behaviour of a real-world system, which might otherwise be complex enough to model. Even though the price of hardware has decreased, researchers cannot expect to receive adequate funding to acquire all of the necessary equipment. So instead of relying on hardware in the initial stages of research, simulation tools are used to produce data, and if the results prove promising, then actual hardware based implementations can follow to verify their assumptions.

Network simulators can be classified into continuous simulators and discrete event simulators. Continuous simulators continuously track the systems response based on a set of predefined conditions. On the other hand, discrete event simulators model the working of a system as a discrete sequence of events in time. Therefore, its internal state will only be changed when responding to some event happening in the simulation universe.

NS-3 is a free and open source, discrete event network simulator licensed under the GNU GPLv2 license and developed by its community of users. NS-3 provides models of different aspects of a network and provides an engine for users to simulate complex networks. NS-3 is not limited to Internet Systems as users are able to also model non-Internet-based systems. While there are other options for network simulators in the market, what differentiates NS3 is the fact that it follows a modular logic. It is designed as a set of libraries to be combined together, but also allows integration with external software libraries, which can further expand the capabilities of the simulations. The necessary tools for a user to execute NS-3 simulations are presented in Table 4. Detailed documentation is available through the main NS-3 web site [20].

Table 4: NS-3 Prerequisites

Prerequisite	***Package***
C++ compiler	clang++ or g++ (g++ version 4.9 or greater)
Python	Python3 version 3.4 or greater
Git	Any recent version (to access NS-3 source code)
tar	Any recent version (to unpack an NS-3 release)
bunzip2	Any recent version (to uncompress an NS-3 release)

While the models and functions that are specific to a network or protocol are implemented in the module's classes, the topology and models that are to be used in a specific NS3 simulation are described via a C++ or Python program. The typical steps in a simulation are the following:

1) The nodes (i.e. devices) that will be present in the network are instantiated using the Node class. If we wish to also simulate a node's position or movement inside the network, a MobilityModel can be associated with each Node.

2) The desired protocol stack is installed on each of the nodes, so it can intercommunicate with other nodes and interpret packets belonging to the specific protocol. This is usually accomplished with the helper classes.

3) Links between different nodes need to be created. Usually, this is done by “subscribing” nodes' PHY layers to the same channel object. The Channel class also provides methods for managing communication subnetwork objects.

4) Applications are installed on specified Nodes, so as to generate activity to be simulated in the network. The Application class provides methods for managing the representations of user-level applications in simulations.

5) When the simulation begins, the Simulator class goes through the events and executes the corresponding function calls. If trace sources where installed in the simulation, then during the simulation, data will be saved to specified data structures or files.

6) After the simulation is completed, the data that was produced can be analyzed and visualized.

5.1 LTE Module

Development of the module began by the LENA project in 2010 at the Google Summer of Code. Initially, it was not possible to simulate a complete LTE network, as only a basic implementation of LTE devices was provided [21]. The module was quickly included in the official NS-3 module list, as improvements arrived over the years. The complete API documentation for the LTE module can be found at the Centre Tecnològic de Telecomunicacions de Catalunya web page [22].

The overall architecture of the LENA simulation model is comprised of two main components: LTE model and EPC model. An overview of the LTE module is depicted in figure 10.

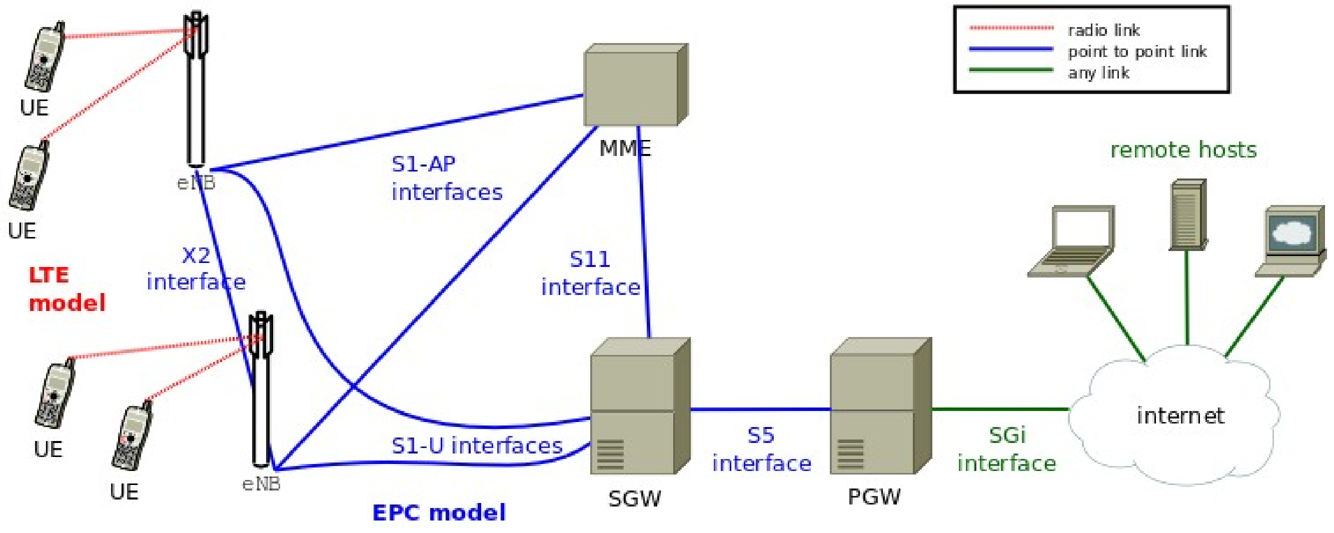


Figure 12: LTE Module Overview

The LTE model includes the complete Radio Protocol stack (RRC, PDCP, RLC, MAC, PHY). These entities reside entirely within the UE nodes and the eNodeBs. LENA LTE model has been designed to support the evaluation of radio resource management, packet scheduling, inter-cell interference coordination and dynamic spectrum access. To allow a correct evaluation of these aspects, the following requirements had to be considered during the modeling process:

- At the radio level, granularity of the model should be at least that of the Resource Block (RB), which is the fundamental unit for resource allocation.
- The simulator should scale up to tens of eNodeBs and hundreds of UEs.
- Within the simulation, it should be possible to configure different cells so that they can use different carrier frequencies and system bandwidths.
- To be as close as possible to real-world implementations, the simulator should support the MAC Scheduler API.
- The LTE simulation model should contain its own implementation of the API.
- The model is to be used to simulate the transmission of IP packets by the upper layers.

The EPC model includes core network interfaces, protocols and entities. These entities and protocols reside within the SGW, PGW and MME nodes,and partially within the eNodeBs. The main objective of the EPC model is to provide means for end-to-end IP connectivity simulation over the LTE model. To this aim, it supports the interconnection of multiple UEs to the Internet, via a radio access network of multiple eNodeBs connected to the core network.

The following design choices have been made for the EPC model:

• The end-to-end connections between the UEs and the remote hosts can be either IPv4 or IPv6. However, the networks between the core network elements (MME, SGWs and PGWs) are IPv4-only.

• The SGW and PGW functional entities are implemented in different nodes.

• The MME functional entities is implemented as a network node.

• The scenarios with inter-SGW mobility are not of interest. But several SGW nodes may be present in simulations scenarios.

• The EPC model must be able to simulate the end-to-end performance of realistic applications. Hence, any regular NS-3 application working on top of TCP or UDP, should be possible to use with the EPC model.

• Another requirement is the possibility of simulating network topologies with the presence of multiple eNodeBs, some of which might be equipped with a backhaul connection with limited capabilities.

• A single UE should be able to use different applications with varying QoS profiles. Hence, multiple EPS bearers should be supported for each UE.

• The main goal is to accurately model the EPC data plane. The EPC control plane is implemented in a simplified way. Necessary control plane interactions between the nodes of the core network are realized by implementing control protocols/messages among them.

• The focus of the EPC model is on simulations of active users in ECM connected mode. Thus, the functionality that is relevant only for ECM idle mode (i.e. tracking area update and paging) is not modeled at all.

• The model should allow the possibility to perform an X2-based handover between two eNodeBs.

Figure 12 and 13 showcase the LTE-EPC data plane and control plane protocol stack respectively, as it has been implemented in LENA [23].

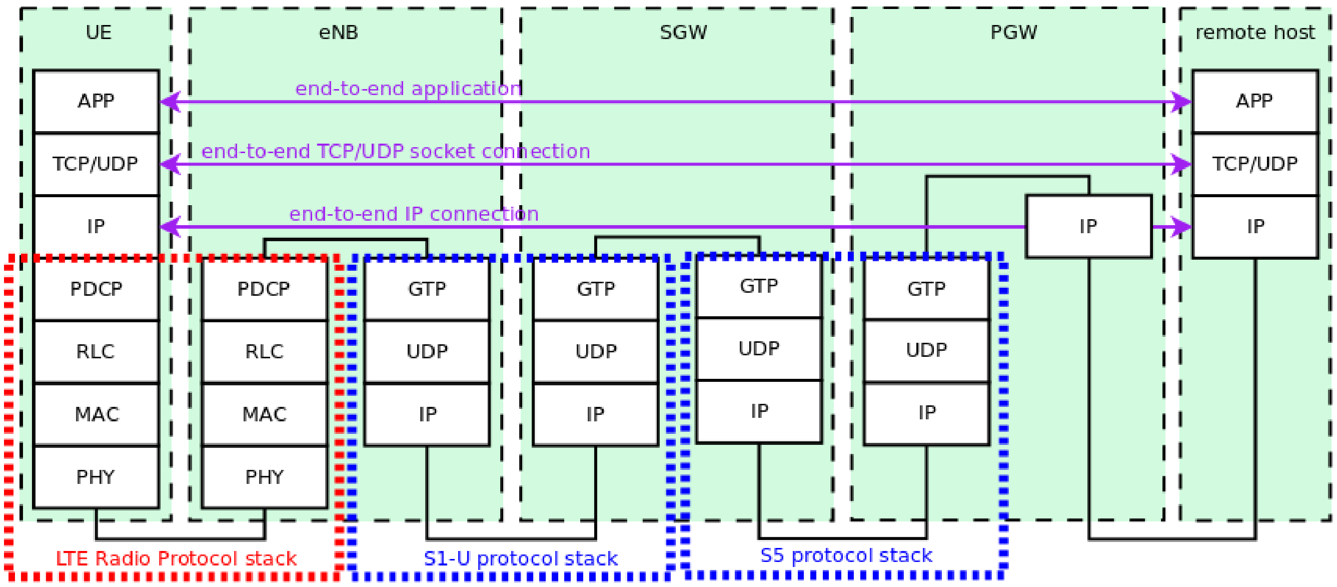


Figure 13: LTE Module Data Plane Protocol Stack

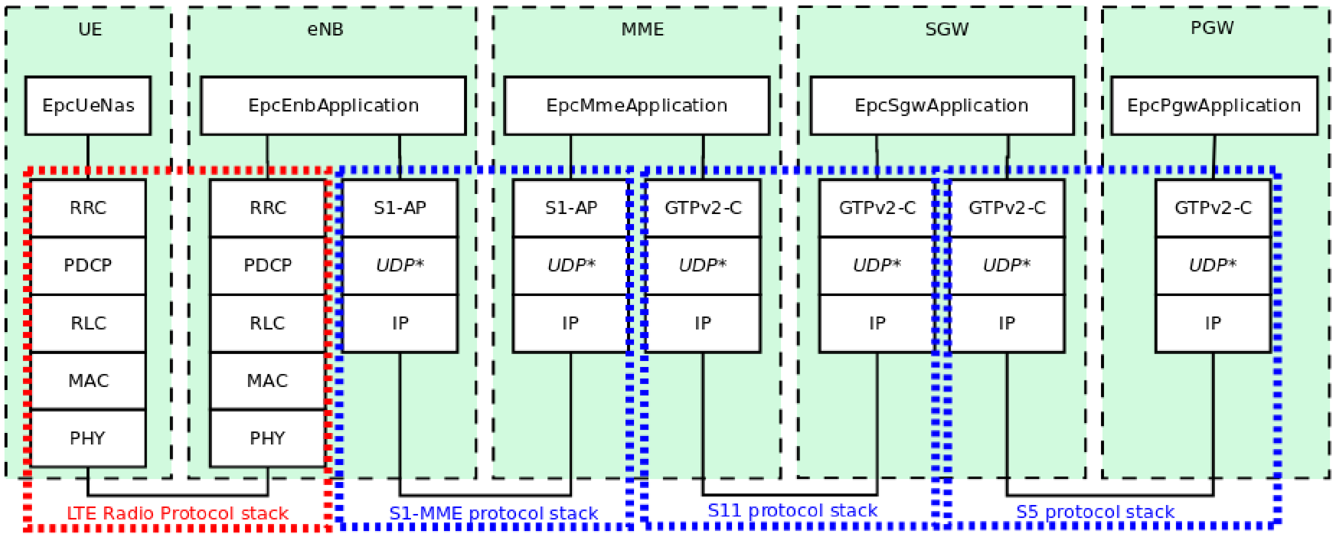


Figure 14: LTE Module Control Plane Protocol Stack

5.2 LoRaWAN Module

The LoRaWAN NS-3 module that was used in the context of this thesis is an external module, developed by a student in the university of Padova for their Masters' thesis in Telecommunication Engineering [24]. This module is comprised by a number of classes that work together to describe the behavior of LoRa EDs and GWs from the PHY to the Application layer. Additional classes were created to expand the modules' features and to model aspects of the system like losses caused by buildings, correlated shadowing, interference and duty cycle limitations. The work described in this thesis is focused on the ED and GW implementations, since it’s at this level that the LoRa modulation is employed. The NS was developed in a very simplistic way, as it was not considered a priority for the objectives of his work.

The set of classes required to simulate the protocol stack on a device can be seen in Figure 15. The source code for the LoRaWAN module can be found in SIGNET Labs' Github [25].

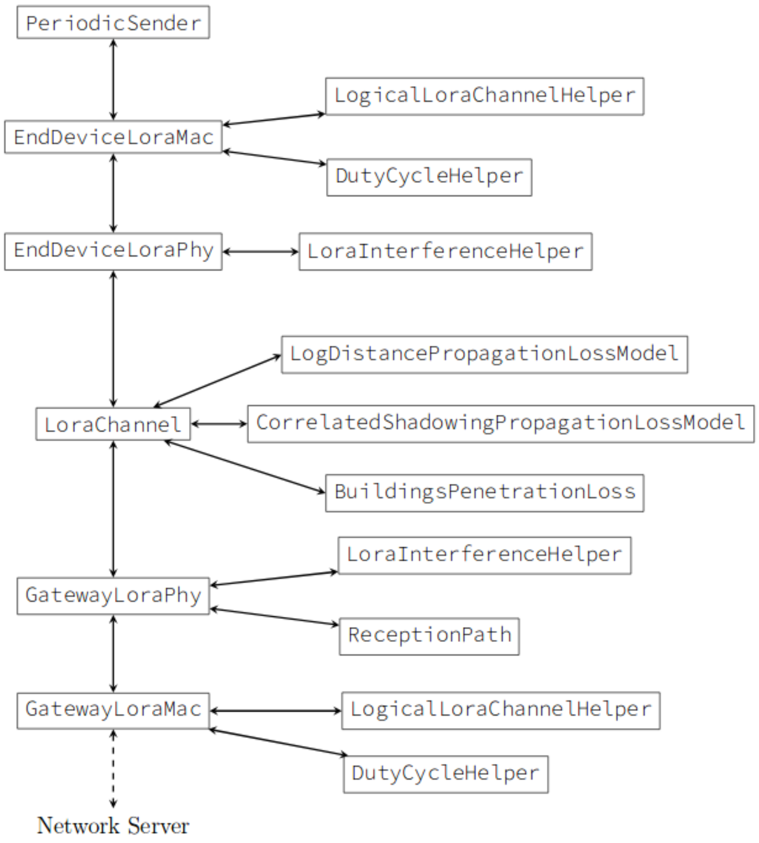


Figure 15: LoRaWAN Module Stack

The module provides an application layer class called Periodic Sender, which is installed on EDs and creates zero-filled packets of randomized or user-set payload size. The application transmission period m_interval determines the data transmission frequency from each ED. It should be noted that at the application level, transmission simply means that the packet is forwarded down to the LoRa MAC layer.

The LoraMac class models the MAC layer of a LoRaWAN device. This class is used to keep track of the available network channels and to account for the duty cycle limitations demanded by the regulations: messages arriving from the application layer should not be sent if this would mean breaking the duty cycle rule, and instead should queue them and send them at a more appropriate time. There are two subclasses EndDeviceLoraMac and GatewayLoraMac that implement behaviors specific to an ED and a GW, respectively.

The EndDeviceLoraMac defines a device’s class and influences the behavior of the ED. Since this class controls the state of the underlying PHY layer, it is also required to correctly handle waking up the radio from sleep when a receive window needs to be opened, or to continuously listen. Unfortunately, the current implementation only supports Class A devices.

The GatewayLoraMac class differs from EndDeviceLoraMac in that it implements a simpler, forward-only mac layer. These classes are also responsible for the interpretation of MAC commands. LoRa packets, with their specific structure, are implemented as extensions of the basic packet class.

The LoraPhy class models the physical layer of a LoRa device. This is the class that simulates the behavior of the SX1272 chip in EDs and SX1301 chip in GWs. When the device has to send a message, this class’s role is to take the packet from the MAC layer and deliver it to the channel class. Furthermore, it decides whether a packet obtained from the channel is correctly received, based on its power and the experienced device interference. At any time, the chip can be in one of these four states:

Table 5: LoRaPhy States

State	***Action***
TX	Transmitting packet
RC	Receiving packet
IDLE	Waiting for packet
SLEEP	Low power mode

Just like previously, there are two subclasses of the LoraPhy class. When a packet arrives at an EndDeviceLoraPhy, an instance of the Event class is created, which computes performs interference computations. If the device is in the IDLE state, then a reception event is scheduled, which will check whether the packet was destroyed by interference. Based on this function's result, the device will either forward the correctly received packet up the stack or will not, before going into SLEEP mode in either case.

The LoraPhy classes are also the location where packet tags are applied. The LoraTag is built to contain information about the spreading factor that is used by a packet. This way, whenever a packet is lost at the PHY layer due to interference the packet tag is changed, so that the simulation script knows what happened to every single packet in the simulation.

The LoraChannel class models the wireless channel that is shared by all devices in the LoRaWAN environment. This class is responsible of taking packets that a PHY layer wants to transmit and delivering them to a set of LoraPhy objects. Interconnection between registered PHYs is done via two methods: Send and Receive. When a PHY needs to send a message in the channel, it can do so by calling the Send function with parameters such as the spreading factor of the message,the PHY-layer packet itself, the duration, transmission power and channel number. Upon calling of this method, the channel goes through the list of PHYs, and schedules a Receive event for those nodes that are registered as listening to that communication’s channel.

During a simulation, only one instance of LoraChannel is created and all PHY devices are connected to it. It’s important to notice that LoraChannel is completely unaware of issues such as sensitivity and interference. It is only in charge of computing the received power and delay at the location of every other device in the network.

The LoraNetDevice class models the network card. This NetDevice can be attached to an ED, whose applications can then use the card to send data to other LoRa devices. This class is essentially used to hold together all the LoRa objects that need to be aggregated to a node, namely LoraPhy and LoraMac. The module uses the NetDevice as an encapsulating class, and only leverages a generic Send version that is adapted to handle the underlying MAC layer. Support for concepts such as multicast and IPv6 addresses is not provided up till now.

5.3 OFSwitch13 Module

The OpenFlow 1.3 module for NS-3, also known as the OFSwitch13 module, was designed to enhance NS-3 with proper SDN functionalities, by providing OpenFlow version 1.3 support . It should be noted that NS-3 already has a module that supports simulations with OpenFlow switches, however its implementation relies on an outdated OpenFlow protocol. Figure 10 depicts an overview of the OFSwitch13 module. In this new model, the controller and switch communicate through standard NS-3 devices and channels. Any desired network control logic can be implemented with the new controller application interface. The source code for the OFSwitch13 module is publicly available [26].

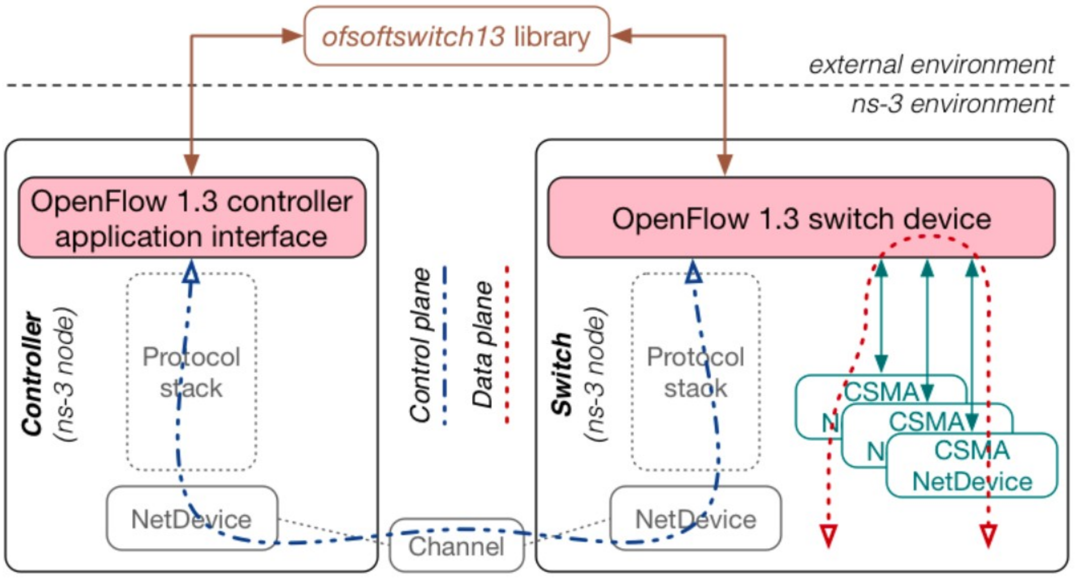


Figure 16: OFSwitch13 Module Architecture

i

The OpenFlow 1.3 switch device (hereinafter referred to as OFSwitch13Device) can be used to interconnect NS-3 nodes using the existing CSMA network devices and channels. Figure 11 shows the internal structure of an OFSwitch13Device. The switch device takes a collection of OFSwitch13Port as input/output ports, each one of which will be associated with an underlying NS-3 NetDevice. The OpenFlow switch datapath implementation (flow tables, group table,and meter table) is provided by the ofsoftswitch13 library. For this reason, packets received at a port are sent to the library for OpenFlow pipeline processing before being forwarded to the correct output port. OpenFlow messages received from the controller are also sent to the library for datapath configuration.

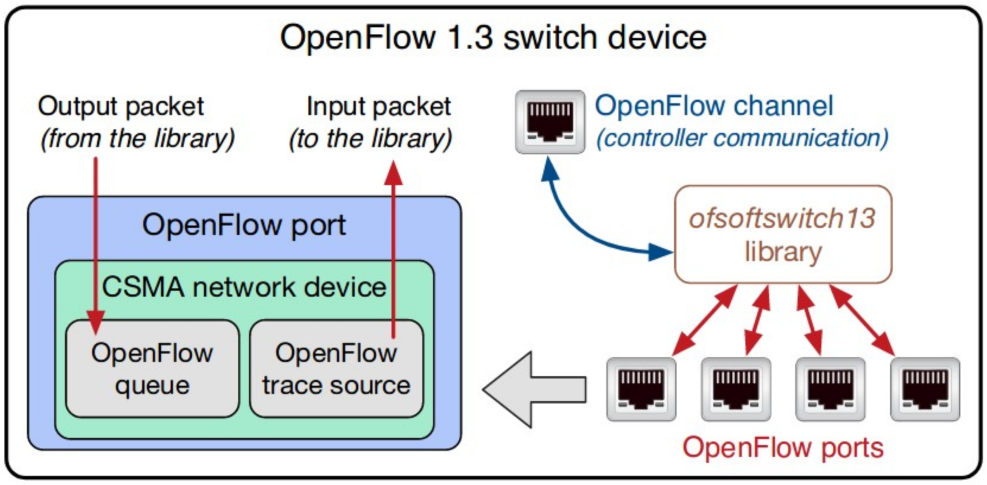

OpenFlow 1.3 switch device
Output packet (from the library)
Input packet (to the library)
OpenFlow channel (controller communication)
ofsoftswitch13 library
OpenFlow port
CSMA network device
OpenFlow queue
OpenFlow trace source
OpenFlow ports

Figure 17: OFSwitch13Device Internal Structure

When a packet is successfully received by the OFSwitch13Device, the receive callback in the NetDevice is invoked. This callback reads the whole packet (including headers), as the headers are necessary for the OpenFlow pipeline processing. Packets that do not exceed the CPU processing capacity defined by the OFSwitch13Device, are sent to the pipeline. In the case the capacity has been exceeded, then the packet is dropped.

Based on the fact that real OpenFlow implementations use sophisticated search algorithms for packet matching in the flow tables, the module uses the concept of virtual Ternary Content-Addressable Memory (TCAM) to estimate the average flow table search time. This times equals to $K * \log_2(n)$, where K is the TCAM Delay attribute set in OFSwitch13Device, and n is the current number of entries on pipeline flow tables.

Packets returning from the ofsoftswitch13 library, are sent to the OpenFlow queue provided by the module. Each port can have multiple queues attached to it and each queue can have its own configuration. The output scheduling algorithm will always serve the higher priority queues first.

The OpenFlow 1.3 controller application interface (referred to as OFSwitch13Controller) provides the necessary functionalities for controller implementation. It can handle a collection of OpenFlow switches, as illustrated in Figure 12. For constructing OpenFlow configuration messages and sending them to the switches, the controller interface relies on the dpctl utility provided by the ofsoftswitch13library. With a simple command-line syntax, this utility can be used to add flows to the pipeline, query for switch features and status,and change other configurations.

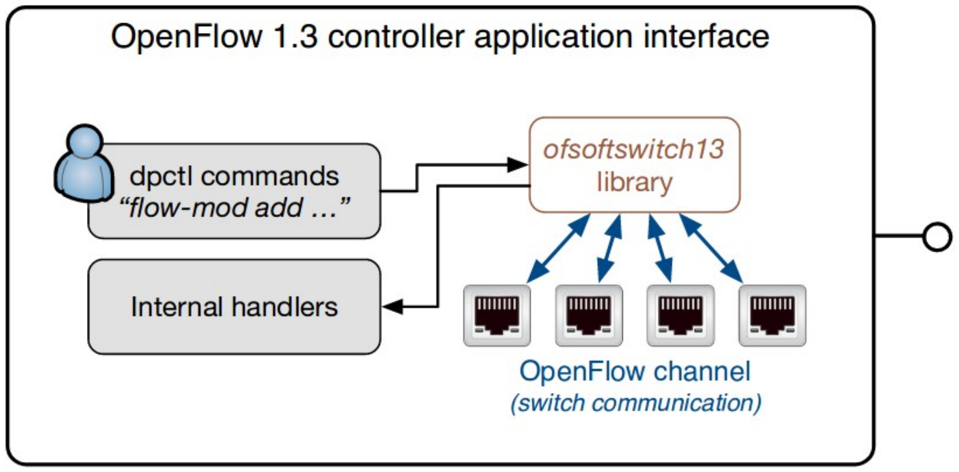


Figure 18: OFSwitch13Controller Internal Structure

The OFSwitch13Controller provides a collection of internal handlers that are capable of dealing with different types of OpenFlow messages coming from the switches. Some handlers can be overridden to implement desired control logic, however others must behave as already implemented.

Switches and OpenFlow controllers are connected by the OpenFlow channel. In the OFSwitch13 module, the controller interface can manage switch devices remotely over a separate dedicated network . It is possible to use standard NS-3 protocol stack, channels and devices to create the OpenFlow channel connections using a single shared channel or individual links between the controller interface and each switch device. This model provides realistic control plane connections, including communication delay and, optionally, error models. The module also allows users to integrate an external OpenFlow controller running on the local machine to the simulated environment, with the NS-3 TapBridge module [27].

5. CASE STUDY

A case study scenario is used to demonstrate how some of the available OpenFlow features can be used to improve network management and user experience. A hypothesis was drafted based on the assumption that in a Smart City scenario divided into different RAN spaces, IoT traffic management could be optimized over a shared SDN infrastructure.

Towards demonstrating the desired features in a way as realistic as possible, the selected scenario involves heterogeneous radio access networks, i.e., on the one side an IoT network, comprising numerous LoRa devices along with the respective gateways and application servers, while on the other side an LTE network comprised of twenty UEs connected to the same eNodeB and the corresponding PGW and SGW. The LoRa Network Server and the LTE PGW are connected via CSMA connections to an OpenFlow enabled switch. Two additional nodes are connected to the switch, which will act like remote servers.

Figure 19 demonstrates the network topology that was created for this case study scenario.

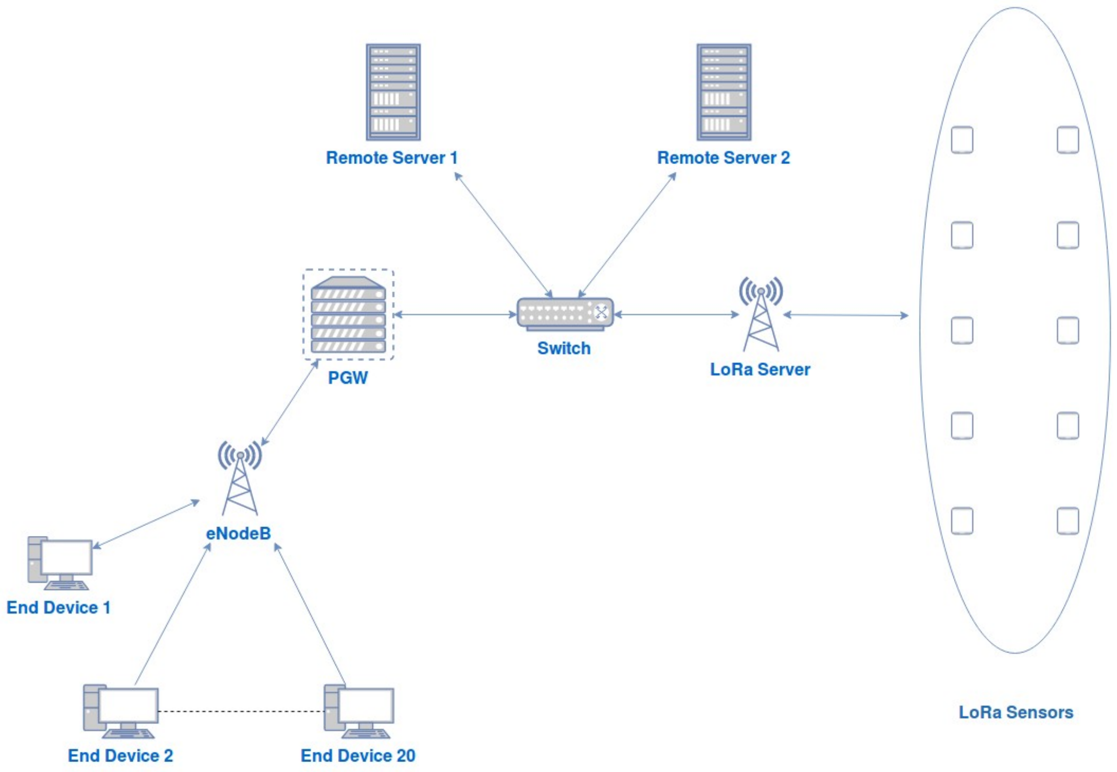


Figure 19: Simulation Topology

Before continuing on, it should be noted that initially we planned to have a more complex test case to study. We were going to evaluate the advantages of integrating SDN features into the network, incorporating diverse types of IoT radio technologies as well, such as NB-IoT -besides LoRa-; however, the NB-IOT NS-3 module has not yet been fully implemented. Moreover, we wished to examine the network management through different controller logics. These included packet forwarding based on packet size and ED source address. Because of the limited options supplied by the dpctl utility, which is used to create flow mod cases in the OpenFlow module, we were only able to implement control logic based on simulation time and data congestion.

The resources available to run simulations are constrained by small memory and limited CPU speed. Thus, simulations of smaller duration were run in iterations to ascertain the expected network behaviour. To balance between a satisfactory data sample size and a realistic duration time, a simulation time of 90 seconds was chosen.

Simulating a large LoRaWAN network to understand the impact of the network parameters is a challenge by itself. To circumvent this issue, we decided to simulate a smaller network of 300 LoRa end devices, which were positioned in a rectangle area around the Network Server to provide fair distance. The Periodic Sender application was installed on all of the end devices to simulate the traffic in the IoT environment. Due to the time limitations of the simulation, data had to be sent much more frequently than it potentially would in a real-life scenario. In our case, each end device sends one packet every ten seconds.

Table 6: Simulation Parameters

LoRa EDs	300
LoRa GWs	2
LTE UEs	20
LTE eNodeB	1
Remote Servers	2
Transmission Frequency	10s
Simulation Time	90s
Switch Connection	CSMA
Mobility Model	Constant Position

Packets produced by the sensors will be forwarded to the network server through their corresponding gateway. The LoRa NS-3 module was created to work autonomously, therefore we had to add the ability for the network server to forward the incoming messages to an LTE UE.

Due to the routing protocols, packets exiting the LoRa network are sent to the OpenFlow switch. The packet information is cross-checked with the table instructions to follow the correct datapath. Our initial table consists of the following entries:

Table 7: OpenFlow Table Entries

EtherType	***In Port***	***IP Destination***	***Out Port***
0x0806/0x0800	1	13.1.1.5	4
0x0806/0x0800	1	13.1.1.4	3
0x0806/0x0800	2	13.1.1.5	4
0x0806/0x0800	3	7.0.0.0/8	1
0x0806/0x0800	4	7.0.0.0/8	1

To give a better understanding of the process, we will use an example of a packet exiting the LoRa network. This packet is received by the switch in the OpenFlow port 4, which is the connection between the switch and the network server, and is destined to be sent to the node with an 10.0.0.2 IP. Therefore, this packet will be sent out of the OpenFlow port 1, which is connected with the LTE PGW. From here, the LTE protocol is able to pass the packet to the correct eNodeB, that will forward it to the correct UE.

It is unrealistic to assume that in a real-life scenario, data would only be transferred one way. For this reason, at t_0=40 seconds, all of the LTE UEs begin transmitting packets towards the second Remote Server. Of course, these packets will also have to be intercepted by the OpenFlow switch and follow the table instructions.

At t_1=60 seconds, specific LTE-EPC links begin to experience data congestion. Consequently, two new entries are added to the OpenFlow table to route packets between the LoRa network server and the first Remote Server.

Figures 20 and 21 exhibit the downlink and uplink throughput of the LTE-EPC with different different amounts of data being sent from the UEs.

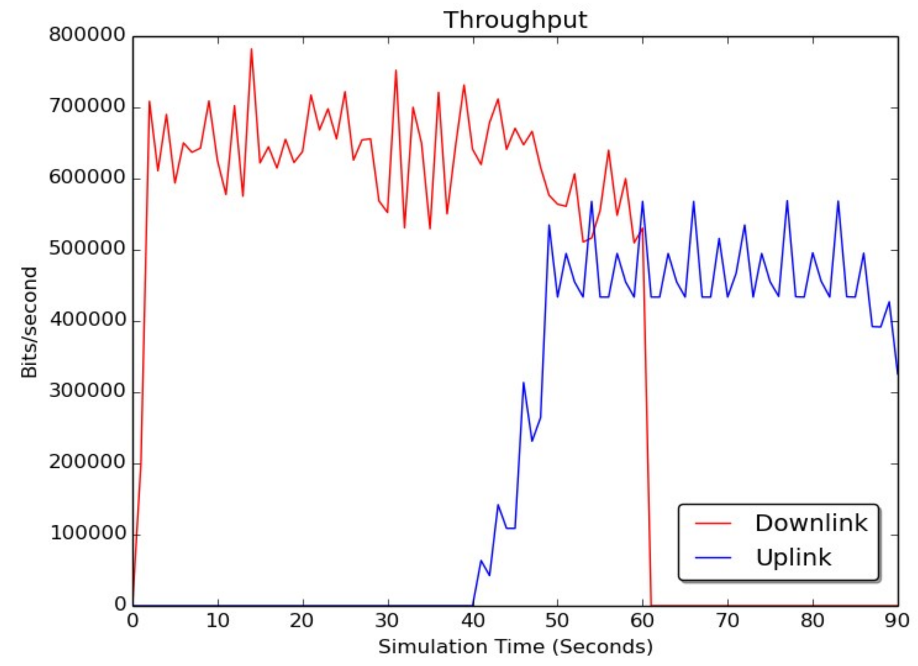


Figure 20: Throughput in LTE-EPC with lower uplink

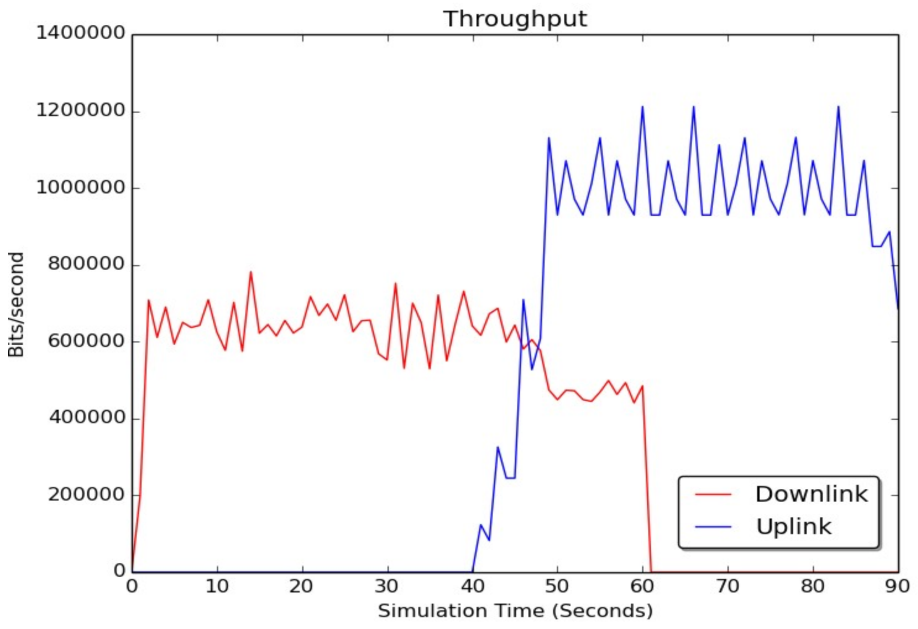


Figure 21: Throughput in LTE-EPC with higher uplink

The results we received from all of the simulations, adequately validated our initial assumptions. We did, however, expect to notice a more substantial drop in the downlink throughput in the LTE part of the network. We can assume that this was a result of two factors. Firstly, the LoRa network is comprised of much fewer end devices than it would have been in a real scenario, hence the overall amount of data sent every second is insufficient to cause massive data congestion. Secondly, as we have already mentioned, since LoRa networks are not expected to support such frequent data transmissions, a great number of packets do not reach the network server to be forwarded to the UEs.

Figure 22 depicts the throughput of the first Remote Server. As foreseen, packets begin arriving only after the OpenFlow instructions are modified. An interesting observation is the fact that the amount of data arriving here is higher than the data reaching the UE. This reveals that some packets did not reach the UEs, possibly because of congestion problems in the PGW.

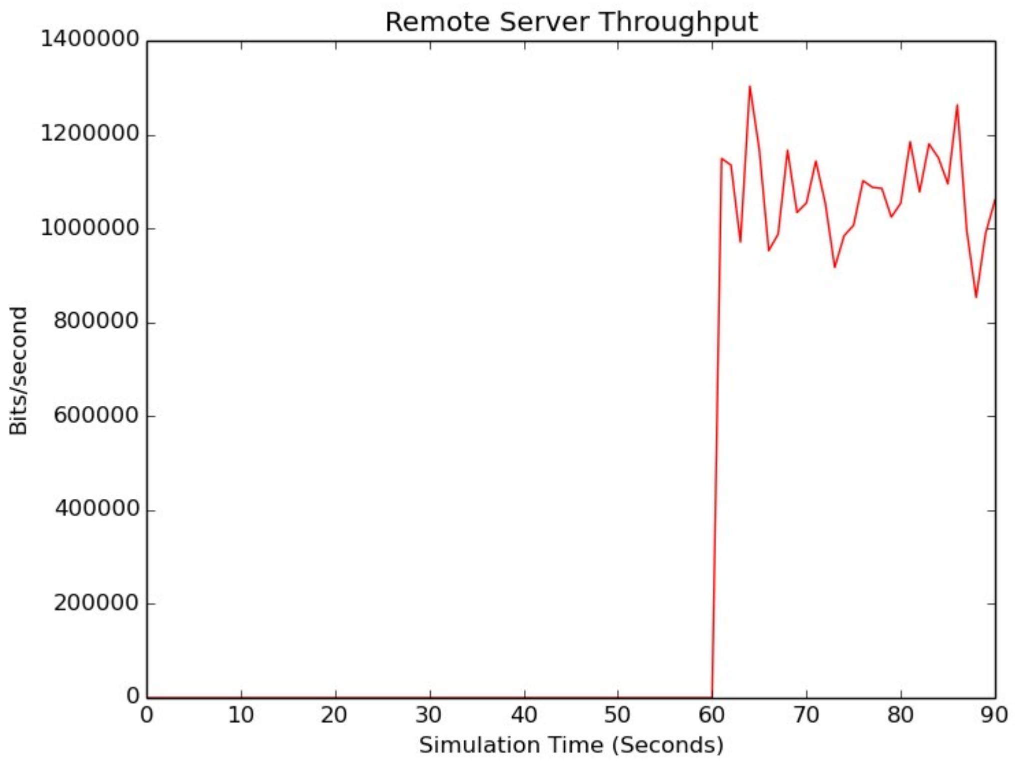


Figure 22: Remote Server 1 Throughput

Aside from the trace sources located in the LTE and OpenFlow modules that record data from signal events, the Flow Monitor module was used to track the packets exchanged by nodes during the simulation. This module installs probes in the nodes and divides packets based on the flow they belong to, where each flow is defined according to the probes' characteristics. This module allowed us to check the number of successful packet transmission from the LoRa EDs to the Network Server.

The results we received showed a 30% packet loss ratio. This was not deemed surprising, as at a constant device density, lower message inter-arrival times will cause more collisions between packets. Moreover, synchronisation in packet departure times could as well increase the probability of destructive interference events.

6. CONCLUSION

In this thesis — motivated by the vision that future internet will be comprised of a multitude of Radio Access Network technologies — we explored the use of the Software-Defined Networking paradigm in order to showcase the several advantages that network programmability brings in diverse radio access technologies and their respective applications, depending on the criticality and prioritization that a network administrator or business user would wish to provide. An overview of the proposed architectures and basic rule sets of each technology was provided. Following, we examined a simulated scenario in which SDN was in charge of forwarding data from an IoT environment towards users inside a 4G LTE network. The results we received confirmed our expectations that by manipulating the data forwarding behavior in a flexible and almost real-time manner, we are able to achieve better load balancing and resource management.

However, there are still many questions that require further investigation. In order to enable network automation and minimize the need for human intervention, the concepts of cognitive networking have been proposed. Cognitive radios have achieved a level of automation using Software defined radios to tune-in to the available frequency channels. Full automation that detects user service needs and then decides the selection of frequency bands based on the service requirements is an interesting future research topic. Furthermore, as the number of IoT devices continues to increase, automatic resource provision will become a key requirement for future networks. The ability to enable network nodes and network segments to cooperate in order to grow and shrink in capacity at real time demands further research. Some limitations might be present, however we can say with great certainty that the use of the concepts of SDN will continue, due to the many benefits it offers, even more so in contrast to previous proposals for programmable network architectures.

On the simulator front, it is evident that NS-3 is a powerful tool, which provides all the necessary provisions to be used in research work. Thus, improvements need to be made only in specific modules. Integrating the NS-3 energy system into the LoRaWAN module, would open up research into battery consumption and management of the devices connected in a LoRa environment. Additionally, a more thorough modeling of the communication link between the Network Server and the gateways, could allow research into congestion avoidance strategies. On the other hand, the OFSwitch13 module is already quite well implemented, with only a few minor features missing. Proper IPv6 support would enhance the research of SDN integration in IoT environments, as IPv4 research could prove cumbersome due to the massive amount of IoT devices expected to arrive in the following years.

ABBREVIATIONS – ACRONYMS

1G	1st Generation
2G	2nd Generation
3G	3rd Generation
3GPP	3rd Generation Partnership Project
4G	4th Generation
ACK	Acknowledgement
ADSL	Asymmetric Digital Subscriber Line
API	Application Programming Interface
ARP	Address Resolution Protocol
CP	Control Plane
CSS	Chirp Spread Spectrum
ED	End Device
EDGE	Enhanced Data Rates for GSM Evolution
EPC	Evolved Packet Core
EPS	Evolved Packet System
EV-DO	Evolution-Data Optimized
FSK	Frequency Shifting Keying
GPRS	General Packet Radio Service
GSM	Global Systems for Mobile Communications
GW	Gateway
HSPA	High-Speed Packet Access
IEEE	Institute of Electrical and Electronics Engineers
IoT	Internet of Things
IP	Internet Protocol
ITU	International Telecommunication Union
LTE	Long-Term Evolution
LPWAN	Low-Power Wide Area Network

MAC	Medium Access Control
NAS	Non Access Stratum
NB-IOT	Narrow Band IoT
NFV	Network Function Virtualization
NS	Network Server
OFDM	Orthogonal Frequency Division Multiplexing
ONF	Open Networking Foundation
PCEF	Policy Control Enforcement Function
PDCP	Packet Data Convergence Protocol
PDN	Packet Data Network
PGW	Packet Data Network Gateway
PHY	Physical Layer
QOS	Quality of Service
RAN	Radio Access Network
RFID	Radio Frequency Identification
RLC	Radio Link Control
RRC	Radio Resource Control
SAE	System Architecture Evolution
SDN	Software-Defined Network
SF	Spreading Factor
SGW	Serving Gateway
SON	Self-Organizing Network
TCP	Transmission Control Protocol
TDMA	Time Division Multiple Access
TFT	Traffic Flow Template
UE	User equipment
UDP	User Datagram Protocol
UL	Uplink
UMTS	Universal Mobile Telecommunications System
UP	User Plane

REFERENCES

[1] Mishra, Ajay K. Fundamentals of Cellular Network Planning and Optimization, 2G/2.5G/3G...Evolution of 4G, John Wiley and Sons, 2004.

[2] Evans D., The Internet of Things: How the Next Evolution of the Internet Is Changing Everything. Cisco IBSG, 2011.

[3] Khan, Sahrish & Shah, Munam & Khan, Omair & Wahab Ahmed, Abdul., Software Defined Network (SDN) Based Internet of Things (IoT): A Road Ahead, 2017.

[4] Dahlman E., 4G LTE/LTE-Advanced for Mobile Broadband, 1st edition, UK, 2011.

[5] 3GPP Technical Specification 23.402, Architecture enhancements for non-3GPP accesses (Release 8), www.3gpp.org.

[6] ETSI TS 136 300 V10.4.0, Evolved Universal Terrestrial Radio Access (E-UTRA) and Evolved Universal Terrestrial Radio Access Network (E- UTRAN): overall description, 2011.

[7] Evolved Packet System (EPS): An Overview of 3GPP's Network Evolution, Qualcomm White Paper, 2007.

[8] 3GPP Technical Specification 24.301, Non-Access-Stratum (NAS) protocol for Evolved Packet System (EPS); Stage 3 (Release 8), www.3gpp.org.

[9] Pierre Lescuyer and Thiery Lucidarme, Evolved Packet System: The LTE and SAE Evolution of 3G UMTS, UK, 2008.

[10] Darwish D., Improved Layered Architecture for Internet of Things, 2015.

[11] Lora-Alliance, Available at https://www.lora-alliance.org/.

[12] Sinha R., Wei Y., Hwang S.A survey on LPWA technology: LoRa and NB-IoT, 2017.

[13] N. Sornin, M. Luis, T. Eirich, T. Kramp, and O. Hersent, LoRaWAN Specifications, LoRa Alliance, 2015.

[14] Open Networking Foundation, Software-Defined Networking: The new norms for networks, White Paper, 2012.

[15] Open Networking Foundation, Software-Defined Networking: Definition, Available at https://www.opennetworking.org/sdn-definition/.

[16] Jarschel, M. & Zinner, T. & Hossfeld, T. & Tran-Gia, P. & Kellerer, W., Interfaces, Attributes, and Use Cases: A Compass for SDN, 2014.

[17] Open Networking Foundation, OpenFlow-Enabled Mobile and Wireless Networks, 2013.

[18] Open Networking Foundation, OpenFlow Management and Configuration Protocol, 2013.

[19] Open Networking Foundation, OpenFlow Switch Specification, 2015.

[20] NS-3 Documentation, Available at https://www.nsnam.org/documentation/.

[21] LTE module for NS-3, Available at https://www.nsnam.org/docs/models/html/lte.html.

[22] LENA Design Documentation. Available at

http://networks.cttc.es/mobile-networks/software-tools/lena/.

[23] Piro G., Baldo N., Miozzo M., An LTE module for the ns-3 network simulator. Proceedings of the 4th International ICST Conference on Simulation Tools and Techniques, 2011.

[24] Magrin, Davide., Network level performances of a LoRa system, 2016.

[25] LoRaWAN module for NS-3, Available at https://github.com/signetlabdei/lorawan.

[26] OpenFlow 1.3 module for NS-3, Available at http://www.lrc.ic.unicamp.br/ofswitch13/.

[27] Luciano Jerez Chaves, Islene Calciolari Garcia, Edmundo Roberto Mauro Madeira, OFSwitch13: Enhancing ns-3 with OpenFlow 1.3 support. In: Proceedings of the 8th Workshop on ns-3 (WNS3), ACM, 2016.