UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola d'Enginyeria de Telecomunicació
i Aeroespacial de Castelldefels

TREBALL FINAL DE GRAU

TFG TITLE: Applicability of 5G NR and Wi-Fi technologies in the aerospace industry sector (industry 4.0)

DEGREE: Grau en Enginyeria d'Aeronavegació

AUTHOR: Antoni Gallardo Llorca

ADVISORS: Sandra Lagén Morancho
Lorenza Giupponi

SUPERVISOR: David Pérez Díaz de Cerio

DATE: February 7, 2020

Títol: Applicability of 5G NR and Wi-Fi technologies in the aerospace industry sector (industry 4.0)

Autor: Antoni Gallardo Llorca

Directores: Sandra Lagén Morancho
Lorenza Giupponi

Supervisor: David Pérez Díaz de Cerio

Data: February 7, 2020

Resum

En aquest treball s'examina si algunes de les tecnologies de l'anomenada Indústria 4.0 es poden aplicar en determinats escenaris de la indústria aeronàutica des d'un punt de vista de telecomunicacions. Es planteja implantar un escenari, en el que diverses eines connectades mitjançant una xarxa sense fils, ajuden als treballadors d'un hangar de manteniment de l'aeroport de Barcelona El Prat-Josep Tarradelles a realitzar les seves operacions. Per determinar si aquestes tecnologies són vàlides, es realitza una simulació amb un simulador anomenat ns-3, que simula totes les capes de protocol de diversos estàndards de comunicacions sense fils. Aquest simulador, dona una sèrie de resultats de rendiment d'aquesta xarxa que han de correspondre amb els requeriments de rendiment de les èines utilitzades. Es realitza una primera simulació amb un escenari creat per fer una sèrie de proves al simulador, seguint els paràmetres publicats per 3GPP de l'estàndard 5G, per assegurar que els resultats que se n'obtenen són coherents i acceptables. Seguidament, es realitza una simulació amb múltiples variacions del mencionat escenari amb el mateix estandard, de l'hangar de manteniment. La simulació de l'escenari es planteja aproximant les mides d'aquest i la quantitat de personal que hi treballa. Es crea amb un nombre determinat d'èines connectades, i es comprova el rendiment de la seva xarxa variant diferents paràmetres dels enllaços terminals d'aquesta (enllaços sense fils entre les estacions base i les èines intel·ligents). Finalment, es recullen totes les dades de les configuracions, s'escull la configuració de xarxa i la variació que permet obtenir el màxim rendiment, i es comprova si la mencionada configuració compleix amb els requeriments dels aparells connectats. En alguns casos, s'ha replantejat ser menys exigent amb els requeriments, ja que en aquests, la configuració que dona més rendiment no compleix amb alguns dels requeriments. En aquest treball, també es comprova que l'actual standard de 3GPP, el 4G-Advanced, no permet complir els requeriments de la xarxa sense fils simulada.

Title : Applicability of 5G NR and Wi-Fi technologies in the aerospace industry sector (industry 4.0)

Author: Antoni Gallardo Llorca

Advisors: Sandra Lagén Morancho
Lorenza Giupponi

Supervisor: David Pérez Díaz de Cerio

Date: February 7, 2020

Overview

In this project, some of the Industry 4.0 technologies are evaluated in order to verify if they can be applied in aerospace industry scenarios. It is proposed to implement an scenario in which various wireless-connected tools assist workers in a maintenance hangar at Barcelona El Prat-Josep Tarradellas airport. To validate these technologies on the mentioned scenario, a simulation is performed by a network simulator called ns-3, which can simulate all the protocol stack of different wireless communication standards. This simulator provides a set of performance indicators the must comply with the performance requirements of the wireless-connected tools. A first simulation is performed with an scenario created to run a series of tests, following the parameters defined by 3GPP for the 5th Generation New Radio (5G-NR) standard, to ensure that the obtained results are consistent and acceptable. Then, a simulation based on the modeling of the commented realistic maintenance scenario with multiple variations according to the standard specification is performed. The scenario is simulated by approximating its dimensions and the number of workers working there, and so, the amount of wireless-connected devices. A number of connected devices is proposed, and the simulation is performed to obtain network performances results of multiple variations of some of the deployment parameters and technical features parameters of 5G-NR that affect the wireless links (links between base stations and wireless-connected devices) and so the devices performances. Lastly, performance data from all the simulation variations is collected, and the variation which offers best performances for every simulation configuration is chosen to determine whether the devices requirements are met or not. In some cases, some configurations do not comply with the requirements. This project also compares performances of the current 3GPP standard, 4th Generation Long Term Evolution Advanced (4G-LTE) Advanced, with 5G-NR, which shows that 4G-LTE specification can not satisfy the simulated network requirements.

Tot està per fer
i tot és possible

CONTENTS

LIST OF FIGURES

LIST OF TABLES

CHAPTER 1. INTRODUCTION

1.1. Context

This project basically aims at evaluating different radio access technologies within an Industry 4.0 aerospace scenario. Accordingly, in this section we introduce the context, which includes Industry 4.0, the aerospace industry, and communications.

1.1.1. Industry 4.0

In order to clearly define what industry is, it is necessary to go to the uses and needs of the world population. As time has been passing by, and technology has been evolving, needs of population have been changing, and new opportunities of business have been appearing. The industry is what can be said as the gap between these needs that population have, and the ideas to satisfy these. For example, 100 years ago, was almost impossible to keep in touch with someone who needed to go to live abroad. Nowadays, it is almost impossible to be disconnected from the big amount of information that technology has made available almost for the whole humanity. It is obvious that something has changed.

In order to understand a little bit more the concept of the industry, it is important to start reviewing its evolution and what is its direction. Basically, the evolution of the industry can be associated with its cost reductions to the production of goods and services to humanity, by means of evolving technologically. Since the first steps of the industry to nowadays, the way the industry and its philosophy is understood has changed a lot. Let's take a look at how has been evolving. The evolution of the industry is classified in four big steps (see figure 1.1):

1. **Mechanization** Introduction of steam powered machines to power factory facilities.
2. **Electrification** Introduction of electricity and electrical machines in the industrial environment.
3. **Digitization and automation** Digitization and automatic control of manufacturing machines and equipment.
4. **Connectivity** Introduction of new communication technologies to industries and implementation of other technologies to make the industry, connected and intelligent. These set of connectivity and computing technologies are called, Industry 4.0 (I4.0)

Nowadays, it can be said that industry is considered to be in its third step. Poorly talking, a current factory can be summarised in a bunch of machines, which all of them are programmed to make simple and repetitive tasks, each of them, is held in the same position, all of them, being pieces of a manufacturing chain line. The rest of processes of the industry, are done basically, manually by workers that follow many steps from a given instructions or manual. That, doesn't seem to be somehow, obsolete, but this model of manufacturing starts getting noncompetitive thinking about, development of 3rd world countries [1] like for example, China and India. Even less competitive, if, for example, start realising that current markets are quickly changing and

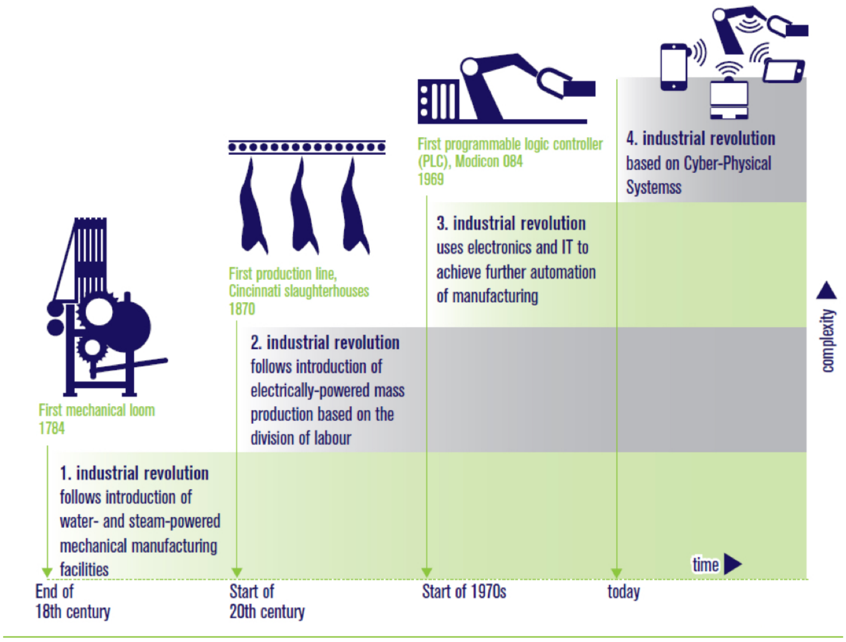


Figure 1.1: 4 steps of the industry evolution [1].

evolving, a fixed, single focused chain line can compete with that?

The I4.0 concept, brings new opportunities to implement new technologies, as cloud computing or Internet of Things (IoT) that have never been implemented in the industry sector, in order to alleviate the limitations presented so far current technologies in the industry. These technologies are aimed to increase productivity by reducing costs increasing even more level of automation, reducing the human intervention, and using new wireless network technologies to increase connectivity and modularity.

1.1.2. Aerospace industry

Taking into account the definition commented in section 1.1.1., the industry provides to people, the services and products they need in order to make life easier. When talking about the aviation industry, a general idea about what it really provides to people, is transport services. Poorly talking, a bunch of companies and other public organisations, provide a service that consists in transporting people through long distances by air. With a general idea, what people understands as aerospace industry, is just that medium and long distance air transport services. Well, that is right though, but there are many things else to be considered. The companies that provide these air services, also need to be provided by services which let these to operate and also to provide services. Let's take some simple examples. Airlines need to be provided with the air vehicles they use to offer their services. Also, these companies need facilities to operate their air vehicles. And these air vehicles need to be provided with energy and maintenance, in order to ensure their operations can be carried out the safer, the better. Other services that these kind of companies need, are for example, services to enable many air vehicles to be flying close together safely and in a great quantity. The list of examples is large so, but the main idea, is that there are other 'industries' behind this 'known industry' that have to be considered, as their services condition other services.

When considering this project, the **aerospace industry sectors** commented, are the ones dedicated to provide service to airlines. To be more precised, **services provided to aircraft**.
These commented services are the following:

- **Aircraft manufacturing** The process which consists in the assembly of an aircraft.
- **Air operations** The operations carried out to move an aircraft from one point to another.
- **Aircraft maintenance** The activities of maintaining the aircraft in good conditions to perform air operations.

The mentioned activities are meant to be carried out mostly by people. For example, **maintenance**. Maintenance operations can be briefly defined as **inspecting** the current conditions of the aircraft. Mainly, this is done to verify the integrity of the aircraft. If there are some components in not enough good conditions, these have to be **repaired** or **replaced**.

The procedures of assembling and disassembling aircraft components are long, complex and require very qualified and experienced workers. These commented works have very strict safety requirements. Taking into account these, and that this sector has always been away from automation and other industry advanced technologies to improve safety and productivity and reduce workers workload, how can new and future industry technologies improve this sector?

1.1.3. Communications

When talking about communications, it can be said that are the type of technologies that have revolutionised the world. They have become part of our lives without even realise it. Thanks to the discovery of the electromagnetism, nowadays, it is possible to transmit any kind of message throughout the world. It all started with the invention of the telegraph, passing by the invention of the telephone, as the example of bidirectional communication, to the radio and television. Both are good examples of broadcast communications. The most relevant invention, and the one which was more innovative and revolutionary, still nowadays, and promises to change people's lives, is without any doubt, the Internet.

Internet, colloquially defined as a packet-switching network of computers, also has been developing trough out the years. From the beginnings, when there was the development of ARPANET (a network that connected the most important American research centres and universities), to nowadays, when a new bunch of new and cheap computing machines have appeared and wireless connectivity has provided connectivity to a wide range of devices far away from these conventional machines. Let's take a look of some of the most relevant thinks to consider about this evolution [10]:

- **1957, Establishment of ARPA from the USA Defense Department.** That was when the Soviet Union launched its first satellite, The Sputnik. The US Department of Defense as an answer, created the Advanced Research Projects Agency (ARPA); an organisation to innovate in new and interesting technologies to dominate the world influence over its main competitor, the Soviet Union. One of the departments of ARPA, founded a research and development unit to design a computer network to connect American universities and research centres to improve their communications.
- **1969, Establishment of ARPANET.** That was a new concept of network brought by Paul Baran. This concept was based on a new architecture, in a non-centralised and scalable network, in which fragments of information are able to travel through different paths of the network to their destination. Notice this concept is something pretty close of the current Internet architecture. This idea was meant to be for avoiding this network to collapse in case of a nuclear attack.
- **1973, ARPANET establishes international communications** A File Transfer Protocol (FTP) is developed in order to be able to transfer files form one machine to an other. This protocol featured a new concept of communication, which featured the possiblity to treat computers as network nodes not as terminals, similar as is done in Internet Protocol (IP).
- **1980-1990, ARPANET becomes commercial.** As the network is able to serve many more users, the Defense Department sets up MILNET. A network for military use only. ARPANET continues to grow, adding new and more capable networks to allow connection to even more users.
- **1989, The World Wide Web (WWW).** Proposed by Tim Berners Lee. The concept consisted in a kind of documents composed by multiple universal protocols. These documents were a new concept of presenting information to users, specially because they were added with multimedia capabilities, as sound playing, images, and hyperlinks. These, let easy access to other Web pages. Finally, in 1990, the ARPANET project ended by letting the way to a more capable and powerful set of networks, as known as the Internet.
- **1995, Internet access via cellular connectivity.** In 1995, some important mobile operators announce to offer internet access services via cellular connectivity. Later on, the General Packet Radio Service (GPRS) protocol would offer this possibility via the Second Generation GSM networks. In which before GPRS standard was implemented, digital connectivity and the possibility to communicate via Short Text Service (SMS) messages or Multi-Media Message Services (MMS) were implemented.
- **1997, IEEE 802.11 Standard [11].** 1997 was an important year. The year when the Institute of Electrical and Electronics Engineers (IEEE) released its first 802.11 standard, commonly known as its brand, Wi-Fi. This standard introduced the possibility to create Wireless Local Area Networks (WLAN), in which compatible devices can have wireless internet connectivity in a small range of coverage.
- **2008, First specifications release of 3GPP 4G Long-Term Evolution (4G-LTE).** Definition of the current mobile cellular standard (4th generation), which is the predecessor of the future 5th generation standard. 4G-LTE standard is standardized by 3rd Generation Partnership Project (3GPP) and offers a good data rate broadband service to mobile devices while reducing also the latency compared to its predecessor.
- **2008, The concept of Internet of Things (IoT) [12].** In these dates, the Internet Protocol of Small Objects (IPSO) was created. This alliance introduced the concept of implement-

ing internet connectivity over small and low-complexity devices.

- **2017, First specifications release of 3GPP 5G New Radio (5G-NR)[13].** This new standard, defined by 3GPP, promises an increase in data-rates, and also, new wireless services to meet new connected devices requirements.

As it can be seen on the mentioned points of the history of the telecommunications, this sector tended to offer wireless internet access to computing devices, and the implementation of internet connectivity to other devices that have never had internet before. The emergence of these new concepts introduces new possibilities to apply new technologies of communications and computing, also to other sectors that have never had this level of connectivity. On example of this last statement is the industry sector. As mentioned in section 1.1.1., the 4th industrial revolution bases its main feature on connectivity. This new technology, and other technologies of the mentioned I4.0, can offer new market opportunities to the aerospace sector. For example, for maintenance operations. As mentioned in section 1.1.2., maintenance operations are largely human-dependant. That is why there is a good opportunity to implement I4.0 technologies to make these tasks less human-dependent.

These wireless networks of smart objects like these mentioned smart tools and other new connected machines, need new connectivity standards which need to be adapted to their connectivity requirements. For instance, smart-sensor networks, do not need as much bandwidth as for example a home computer does. On the other hand, these types of networks need the possibility to connect a large number of devices as well, and keeping a low energy consumption rate. The new-coming 3GPP standard, 5G, is not only focused on provide cellular connectivity to mobile devices, as would its predecessor. This standard, promises new services to cover all the mentioned requirements of the new networks. These services can be summarised as follows:

- **High data-rate service** Providing a down-link maximum throughput of 20 GB/s and 100 Mbps at 90% of cases. This service would be the continuation of the main service offered by standard LTE. A substantial improvement in data rate performance.
- **Very-low latency and very reliable service** Implementing mechanisms to increase redundancy and limiting packet size to ensure reduced latency values. These service aims to provide service to certain machines which require very high precision, as remote-controlled robots or vehicles.
- **A service to provide connectivity to a large number of simple devices.** These devices, are referred to as networks of numerous devices that require just small data-rates.

If one wants to implement any wireless standard, in order to verify if the requirements of the connected devices are met with the specifications, a validation method is required.

1.2. The project

As mentioned in sections 1.1.1. and 1.1.2., the I4.0 concept offers new possibilities to implement new and never-implemented technologies to the current industry to make a step further over the current limitations of it. As the way to work in the aerospace industry sector can be, in some ways, rudimentary, implementing some I4.0 technologies can help to improve performance in certain operations, also improving safety and reducing time needs. In this project, an aerospace maintenance scenario has been selected to study. The key question this project tries to answer is: *How can such scenario be improved by implementing I4.0 technologies?*

1.2.1. Objectives

The objectives of this project can be summarised in the following points:

- **Looking for a good I4.0 possible application to aerospace industry in which implementing certain technologies can help to improve its operations**. For that, a brief review of I4.0 is presented in chapter 2 to understand a little deeper these technologies. Having understood and reviewed I4.0 technologies, in chapter 3, some possible ideas of I4.0 technologies applications in the aerospace sector are gathered and reviewed. Having considered this, is concluded that the best found application can be **maintenance operations**. In section 3.1., the maintenance aerospace sector is reviewed, and some implementations of I4.0 is are commented in section 3.2..
- **Reviewing a suitable wireless standard to be implemented in the commented scenario.** The most suitable standard to implement in the planned scenario, is considered to be 5G-NR. As mentioned back in section 1.1.3., this standard raises services focused to serve requirements of I4.0 technologies. To simulate this specification, it is important to have a good review of it. The standard specifications and the objective requirements are reviewed in section 4.1.2..
- **Creating a maintenance simulated scenario in order to evaluate if 5G-NR standard can satisfy devices requirements.** In order to create a maintenance simulated scenario, the following steps are performed:
    - **Look for devices which can be used in aerospace maintenance.** The idea is inspired by the reviews of the example of application of section 3.3.2.. The devices proposed for this scenario, are the following:
        - **A smart tool.** The concept of this tool, is a remote-controlled tool in which its main requirements, are to be wireless, and to have the minimum amounts of latency as possible.
        - **An augmented-reality device.** This device should be able to transmit video and receive a good amount of information, its main requirement, is to be able to transmit and receive great amounts of data rate.

      These devices are chosen in purpose in such a way they have very different applications and thus, with very different requirements.
    - **Find a way to approximate the requirements of the mentioned tools.** As this is a supposition of an implementation which has never been done, requirements of the scenario devices are similar to other similar industry devices. This approximation is reviewed in section 7.2.1..

- **Design a close-to-reality scenario.** Which describes an implementation of the smart devices in a possible real situation. At this case, the chosen scenario is a maintenance hangar in Barcelona. A detailed description of this scenario is reviewed in section 7.2.1..
- **Simulate the maintenance implemented network with ns-3 simulator.** To obtain its performances. Different deployment configurations for the scenario are simulated, and the ones that give the best performances are chosen. The whole results evaluation process is commented in section 7.4..
- **Compare the results of the current-implemented wireless standard with an other wireless standard.** At this case, the standard in which this evaluation is carried on, is LTE-advanced. The mentioned comparison is reviewed in section 7.5..
- **Evaluate if the selected wireless standards can give service to the device.** Comparing the proposed requirements with the real performances of the network. This is done in section 7.4.

1.2.2. Methodology

This section reviews the performed steps to arrive to the conclusion, on whether the mentioned I4.0 can be implemented in the proposed scenario or not. It describes from how the scenario is designed, through the steps for getting the results, to finally, how the best configurations are chosen and how the feasibility of the whole project is analysed. In this section, the Test Scenario simulation is not mentioned, rather only the steps to get the results that determine the final conclusions of the project are described. The test simulation so, is not necessary to get the mentioned results. The following steps, are the ones followed to reach the results wanted to obtain:

1. **An scenario to evaluate the I4.0 technologies is found.** Is the first step to be carried out. Taking inspiration on the example of I4.0 implementation commented in section 3.3.2., the preffered scenario to simulate, is a maintenance hangar. More details of this scenario, in section 7.2.1.. Also, the reviewed sections should give an idea of what kind of devices can be simulated for the evaluation.
2. **The simulation scenario is modelled.** Is the step between the real scenario, and the simulated scenario. For the Real-Based scenario simulation, it is modeled following the following steps:
    - **The sizes of the scenario are determined.** Basically, delimits the area in which the simulated devices are spread, so is important to set some size limits in the simulation. In the mentioned scenario case, the references taken to determine the size of the scenario, are the sizes of the hangar, as it is obvious. The easiest way and the one carried on to determine these sizes, is by finding the hangar in Google Earth, and using the rule tool to determine approximated sizes. The detailed review of this step is shown in section 7.2.1.2..

- **The amount of devices to serve is determined.** By searching information on the current activity carried out in the real scenario. In the case of the chosen scenario, information is searched on how many aircrafts can be maintained at the same time, and the amount of people who performs operations at the same moment. See section 7.2.1. for details. Finally, a number of devices for every kind wanted to simulate, is decided.
- **The layout of the devices is determined.** The position on every device is determined. In the hangar scenario, as the devices are wireless-connected handheld devices, it is decided that they would be spread all around the scenario randomly as real workers would. It is important to mention that in some scenarios, a fixed position is preferred.

3. **The requirements of the connected devices are determined.** The requirements of every kind of devices are searched. If the devices wanted to simulate are not found, these requirements can be approximated with similar devices. At this project simulation, the chosen device requirements are approximated as shown in table 7.2 of section 7.2.1.1. following the requirements of table 2.1.

4. **The specifications of the implemented wireless standard are reviewed.** As these standard specifications are inputs in the simulator. As in this simulation case, there are, for instance, many possible bands and bandwidth values, many configurations of different values are specified to be evaluated in the simulation. As for example, NR offers many services with different protocol specifications, it is important to have many variations to find the most suitable one.

5. **All the inputs of the simulator are introduced.** By modifying the code of the simulation examples (see section 6.2.2.), the following data is introduced:

   - **Positions of devices.** The layout of all devices is introduced by inserting the position on every device. The smart objects, in this case, are spread randomly across the whole area, so the sizes of the scenario, determine the range of the random position values.
   - **Simulation parameters specified by the standard.** As for example frequency and bandwidth. These parameters are changed on every variation of the simulation. For example, if bands of 6 GHz and 28 GHz are chosen from the available bands of the simulator, there are so, two variations of the simulation. One that uses the 6 GHz band, and the other one that uses the 28 GHz band.

6. **The simulation is performed.** All the decided simulation variations are simulated. The average throughput and average latency are obtained as a performance indicators on every variation of the simulation. The performance results on every variation are annotated to then, be analized. Is worth mentioning that this step, and step 5, are performed multiple times, as multiple variations are decided to be simulated.

7. **The variations that give best performances on every deployment are selected.** The results obtained on the previous steps are compared, and the configurations that give the best performances are selected, and so, annotated.

8. **The final results are compared with the predefined requirements.** In order to determine if these requirements are successfully met. That gives an idea on which devices can be successfully implemented and which of them not. Also, it gives an idea on whether the wireless standard can be implemented or not, among other conclusions.

1.2.3. Contributions

In this section, the different contents that have contributed information are detailed and explained in the following paragraphs.

1. **Multiple technologies and other industrial areas have been reviewed** Not many details are shown in this section, due to all of them have been reviewed in a more complete way, in other sections. These technologies are worth to mention.

    - **Radio access technologies.** As this project is about implementing wireless technologies and evaluating them, some of the most important ones have been reviewed and detailed. In particular:
        - **3GPP 5G-NR** The most relevant for this project, due to it is the one that better serves the I4.0 requirements, and so, the one implemented in the performed simulations. A whole review is shown in this project, both with technical specifications, applications and industrial services. See section 4.2..
        - **3GPP 4G-LTE** It is also important to be reviewed due to it is the predecessor to 5G-NR. Also, a section of the simulator is dedicated to compare performances between this standard and 5G.
        - **Standards 802.11** Less relevant, but also important to mention, because is a standard that works on unlicensed spectrum. Future generations of other standards like 3GPP, pretend also to work with unlicensed bands [6]. The four most recent releases have been reviewed in section 4.4..
    - **I4.0 technologies.** As the simulated scenario is about implementing some of these technologies, for instance IoT, some of the I4.0 environment must be reviewed and thus, some aerospace industrial applications and examples are reviewed.
    - **Aerospace industry.** As the simulation implementation is done in an aerospace maintenance scenario, a good review of the current aerospace maintenance and an analysis of what can be improved with I4.0 technologies is very important. A simple review of some aspects on the aerospace industry and aerospace maintenance are listed in chapter 3.

2. **ns-3 review and familiarisation** As in this project, the ns-3 simulator is the main key to determine if the smart-device network can be implemented as the planned design, a full review of the mentioned simulation is detailed in section 6.2.1., which is important when considering this work for further implementations in other projects. At this review, some of the test scenarios are reviewed, and also, it is explained how are modified in order to get the necessary results both presented simulation scenarios. In order to test the capabilities

and the usability of the ns-3 simulator, a test simulation is planned before testing performances on the real-based scenario. This test simulation is based on a small scenario in which deployments are varied depending on the number of user devices, the number of base stations, the throughput and latency requirements and the transmit direction. Also, the rest of parameters which are left unmodified, are mentioned and commented. Then, the performances results are obtained and analized in order to check if the results are as expected. Finally, the results of this simulation, and the real-based simulation are compared. When finishing every simulation, this project also proposes a way to show the results. It is a rudimentary procedure, but if preferable, is also possible to create a function or procedure which collects the results and displays them automatically for example. At this case, as mentioned in section 6.2.2.3., the procedure consists of annotating the results and entering them into a MATLAB script which produces the corresponding plots. These mentioned reviews show general capabilities on the reviewed software and shows some of its possibilities.

3. **A proposed real-based simulation example.** The idea is to propose a simulation example comparable to a real scenario. In this project, the example proposed to simulate is a deployment of smart tools in a maintenance hangar of one of the principal airlines in Spain. The main ideas for this scenario, are the following:

   - **Two different types of devices with different requirements.** The selected ones are: smart tools and augmented reality (AR) headsets. AR headsets require a good data rate, while smart tools require a low latency.
   - **A comparable size with a real scenario.** As mentioned in section 1.2.2., the simulated scenario is sized based on approximated real values.
   - **Hypothesising the number of devices to which the service is given.** For that, realistic values are considered, since there are not real values available.

   Summarising the mentioned, this project is done with hypothetical and approximated data. If it is possible to use actual maintenance data and requirements of real devices, (as for example, requirements shown on a data sheet of a commercial device) are taken for a future-close implementation, the results of the simulation could be reliable as a first implementation study.

4. **Comparison between two same deployments with different wireless standards.** It is important to look for the most suitable wireless standard for the certain application, also, it is worth to do a comparison between releases of a determined standard. This is a good way to evaluate if the advantages that provide new standard releases worth the implementation. As for example, at section 7.5., LTE-Advanced performances are compared with performances provided with 5G-NR, both, releases of 3GPP. As can be seen in the mentioned section, 5G-LTE offers much better performances without no doubt. Actually, at the same section, it is proved that LTE-Advanced does not satisfy the requirements at all. Therefore, it has been determined that is better to focus on implementing systems following the NR standard.

CHAPTER 2. INDUSTRY 4.0

2.1. Industry 4.0 review

One of the concepts that better define what the Industry 4.0 (I4.0) is, is the evolution of the current manufacturing model, from digitization, to one step further, smart [14]. Smart industries introduce a whole new concept of what a current industry is. Thanks to new emerging technologies, new services will be available to change how products are manufactured, delivered, and also, how the relationship with costumers will also be improved. The industry will overcome its current limitations of it, like for example, poor flexibility and a narrow application range, to be more competitive, specially, compared to new emerging countries which their industries, are currently more profitable [1].

The Industry 4.0 can be described in 3 blocks [14]:

1. Physical resources.
2. Network resources.
3. Data resources.

See them in sections 2.1.1., 2.1.2., 2.1.3. respectively.

2.1.1. Physical resources

This term, Physical resources, stand for all manufacturing equipment involved in the manufacturing process. The two main disadvantages of the conventional industry physical resources are that physical resources are barely configurable, and machines have short range of functions. For that, there are two solutions for this problem.

- Separating manufacturing equipment into **independent manufacturing blocks**. These blocks, would add the possibility to the manufacturing line to be configurable, cooperative and to adapt better to changing manufacturing demand.
- Making **manufacturing blocks, multi-use**. There are many possibilities of use for some hardware, to make it more versatile just by improving their software.

2.1.2. Network resources

All connected devices rely in a network which provides them with the connectivity requirements they need. Two groups of networks may be deployed within an industry [14]:

1. The **field bus** technology integrates all conventional wired network technologies. A good example of a field bus, would be Industrial Ethernet, which connects all manufacturing facilities with services as for example, cloud.

2. **Wireless networks**, which connect devices that must have wireless network connectivity, for example, a network of connected sensors. An example of a wireless network standard is 5G-NR; a standard adapted to provide service to a great range of wireless devices.

2.1.2.1. Cloud technology

Cloud is the technology that provides network services all across the smart factory and outside it. These services can be sumarized in three basic services [1] which are commented below.

1. **Information access** It can store or process information on real time from any place and device from the network, and let this information to be accessed from all around the factory and outside.
2. **Storage capacity** It as storage service mainly for data storage from sensors, manufacturing tools and other connected equipment to be processed.
3. **Cloud computing** Tets the access to processing resources for collected data analysis

2.1.3. Data resources

The industry network can provide a lot of data from many places, devices, manufacturing machines and tools, data from sensors... This whole amount of data, is actually useful if processed using algorithms of **machine learning** or **data mining**. These algorithms can get useful knowledge in many different areas.

2.1.3.1. Big data

Processing data can provide useful information in a bunch of different applications. In case of the industry, big data can be implemented for example, in manufacturing, maintenance and product optimization [14].

- **Big data in manufacturing.** Processing all the production data of the factory, can provide information to optimize manufacturing processes.
- **Big data in maintenance.** Big data can offer opportunities in maintenance, like for example, failure predictions or active maintenance, which helps to improve efficiency by flexible maintenance time intervals. That helps to reduce downtime, and increases production.
- **Big data in product optimization.** Product offering can be optimized using big data by for example, two different ways.
  - **Product distribution.** Obtaining data from distribution and selling equipment. Logistics, storage and selling processes can be optimized.
  - **Product design.** Obtaining and processing data from the use of the sold product itself in order to improve software updates and future product generations.

2.2. Industry 4.0 device requirements

In this section, some of the most common possible industry smart devices are shown. The data shown in table 2.1, shows the minimum performance in which the network has to provide service in order to make the work as expected.

Table 2.1: Requirements of frequent industrial devices [6]

Industrial devices	Latency (ms)	Availability (%)	Throughput (bps)	Number
Industrial robot	< 1	> 99.9999	10^3	> 100
Mobile robot	< 1	> 99.9999	10^6	> 100
Sensor	~ 100	> 99.99	10^3	> 200
Head mounted display	< 10	> 99.9999	$10^6 - 10^9$	> 50
Handheld terminal	< 10	> 99.9999	$10^3 - 10^6$	> 50
Automated guided vehicle	< 10	> 99.9999	10^6	> 10
Security camera	~ 100	> 99.99	$10^6 - 10^9$	> 10

CHAPTER 3. INDUSTRY 4.0 IN AEROSPACE

Industry 4.0 technologies have a very important application range, including the aerospace and aeronautics environment. Even if the aerospace scenery is very dependable to human resources, I4.0 technologies can possibly be implemented in a bunch of different situations to improve safety, efficiency and reduce costs. This section shows some applications of I4.0 in the aerospace sector in the following enumeration:

1. **Aerospace smart manufacturing** This scenario describes how current aerospace manufacturing environment is, and how can be improved with I4.0 technologies. Manual working with smart tools with among other things, help to increase productivity.
2. **Optimization of airline operations** It is also possible to implement I4.0 technologies among the usage of aircrafts to reduce costs of airlines and emissions, and increase time efficiency. A good way to make airlines more profitable and safer.
3. **Optimization of maintenance operations** Aircraft maintenance is a complicated activity. Current air crafts are complex, and technologies implemented to perform maintenance activities are simple and rudimentary. At this section, current aerospace manufacturing is described, and how technologies like augmented reality and additive manufacturing can help to make this activity more reliable, safer, simpler, and cheaper.

3.1. Current aerospace maintenance

As aerospace engineering evolved, aircrafts have become larger and more complex by the implementation of new technologies such new advances in electronics mechanics, and manufacturing materials. Nowadays, aircraft maintenance had become that complex that there is a compulsory formation degree to certificate workers of this activity. Thus, airworthiness compulsory regulate maintenance with programs to be carried out by aircraft operators. Each operator, also have their own maintenance procedures depending on aircraft equipment.

In order to understand possible applications of I4.0 to aerospace maintenance and the following subsections, some of the most relevant concepts of maintenance are reviewed.

- **Maintenance, Repair and Overhaul (MRO).** MRO describes all procedures to: (1) Operations of inspection, fixing, or replacement of broken or damaged pieces from an aircraft; (2) Replenishment of gases, lubricants and other fluids, and replacement of consumables like sealants and coatings. All of them, to ensure Airworthiness Directives, in order to keep aircrafts in operational conditions to fly. These activities are very important to ensure safety on flight operations and are strictly regulated by aviation organizations like Federal Aviation Administration (FAA) or European Union Aviation Safety Agency (EASA). They certificate companies that carry out this type of activities.
- **Shell model.** The shell model describes the four most influential factors on aviation safety. To ensure it, all of them must be totally coordinated.

- **Software.** All conceptual resources, as would be regulations, instructions, organization, information...
- **Hardware.** All physical properties of the company. Like aircrafts, buildings, materials...
- **Environment.** Both concepts, natural environment, or political environment are influent factors on aviation safety.
- **Liveware.** People encharged to run the company. Engineers, pilots, maintenance workers...

- **Inspections** Aircraft maintenance inspections can be classified in four levels depending on how exhaustive they are. Also, maintenance intervals are programmed taking into account the following variables:
  - **Flight hours** To measure aircraft fatigue in general.
  - **Number of take off and landing cycles.** To measure the amount of peak load situations.
  - **Amount of time** To measure when to replace components which degrade over time.

  The four levels of an inspection are:
  - **Check A**: A quick general integrity check of the aircraft. Its carried out approximately every three hundred cycles.
  - **Check B**: A more exhaustive inspection of the aircraft. This includes checks of A with more deep engine, structural and control surfaces checks. Is carried every two thousand flight hours and takes around one to four days.
  - **Check C**: An exhaustive check of the aircraft. Many important parts like engines are totally disassembled for a close inspection. The inspection is done every three thousand five hundred flight hours and takes to do it around eight to fifteen days.
  - **Check D**: The most exhaustive inspection of the aircraft. Every structural component of the aircraft is totally disassembled and closely inspected. This inspection is carried out every twenty thousand flight hours approximately, and takes around two months to finish it. After the inspection, a three hour flight test is required.

3.2. IIoT in the aerospace industry

IIoT is the I4.0 technology that better can improve operation conditions and increase productivity and efficiency. As operation conditions relay at most on human workers, IIoT's main aim, is to improve manufacturing performance and interconnect manufacturing plants providing information across all resources of the company. Two examples of use of IIoT could be for example:

- **Precision procedures** As for example, disassembly and assembly procedures. Both, in the maintenance case, and in the maintenance case. Using technologies as augmented reality and an image recognition algorithm, by scanning an aircraft is possible to get information from assembly or disassembly procedures. These can be sent to a determinated smart tool to proceed with the necessary manufacturing step. As there are more than 1.100 different tools and up to 400.000 bolts and screws [15], manufacturing process is simplified by finding automatically the most necessary information.

- **Location Tracking** Keeping all tools accurately located all the time, can help to increase manufacturing or maintenance security and safety, and keep track of workers productivity. This information may be used to ensure that workers use the right tools and do not use the ones who are not qualified to have access to, so production is controlled and safety across manufacturing or maintenance plants can be achieved. Also, using data analysis, it is possible to find new ways to optimize productivity.

3.3. Optimization of airline operations

I4.0 technologies can also deliver services to airlines aiming to reduce operation costs. Airlines operations to be improved by I4.0 can be classified in:

- **Maintenance operations** Industry 4.0 can generally contribute to aerospace maintenance in two different ways.
  - By collecting real-time data from sensors spread through the aircraft to provide information about its technical conditions. This information is processed and analyzed to determine if a certain aircraft is susceptible for a possible failure or maintenance need and thus, reduces the amount of work to maintenance workers making them able to reduce complex diagnostic procedures.
  - By assisting maintenance workers on their operations in order to reduce their workload, increase their efficiency and reduce the possibility of them to make mistakes, increasing safety.
- **Flight operations** Similar to maintenance operations, data from sensors across the aircraft, and specially, data from sensors in the engines, is processed to improve engine performance, reduce noise and so, reduce fuel consumption. Also, other knowledge can be obtained by processing this data to optimize other parameters in flight operations.

3.3.1. Assisted aerospace maintenance

Industry 4.0 can provide new opportunities to change how maintenance operations are done. Implementing I4.0 technologies artificial manufacturing, can improve such important things like workers productivity and safety, but also, can provide other advantages over traditional manufacturing, like reducing needs of having all replacement parts stored next to maintenance facilities, in case of addition manufacturing, or manufacturing parts with better physical properties. More detailed advantages of maintenance 4.0 are:

- Using IoT and network technologies 2.1. for Interconnecting people, smart tools, machines with all the maintenance facilities and company resources for better information flow and thus, better organisation.
- Using these technologies to obtain information from maintenance facilities and aircraft sensors to process it for taking decisions and optimizing procedures using technologies like big data and machine learning.

- Having virtualized information and CAD models of all aircraft components and having the possibility to modify them dynamically to introduce improved parts on the aircraft without having to replace all assembling/disassembling manuals and components catalog. Definition of the concept virtual twin.
- Collaboration between workers and machines to visualize clear and interactive information of maintenance procedures and automatize repetitive, dangerous and tiring tasks.

3.3.2. Aerospace maintenance using IoT example: Augmented Reality

Technologies of industry 4.0 can change the ways in which maintenance is done. For example, I4.0 technologies add the possibility to have virtual copies of aircraft models (technically known as Virtual Twins). Assembling and disassembling manuals can be done using these models and updating them is a much more easier and dynamic tasks. Providing on-demand and updated information to maintenance workers, for example, using Augmented Reality (AR). Also, depending on certain type of replacement parts, some of them can be manufactured using the technique of additive manufacturing (AM), which allows to have spare parts without the need of them to be ordered within advance. It also, allows to have smaller replacements storehouses.

It is important to mention that the AR technology, is equally important than other I4.0 technologies like IoT, and Big data. AR technology depends on network support, it provides information to other industry resources, and receives information of them.

The AR technique consists in combining real objects with computer-generated images. Devices commonly used for AR, could be **tablets and smartphones**, or **augmented reality lens**. In case of tablets or smartphones, the camera-recorded image is displayed on screen with virtual objects interacting with recorded images. With augmented reality lens, these virtual objects display is done by projecting them over the lens.

Real and virtual objects must interact. The interaction is done by the recognition of marks distributed over visible surface of the real object or directly, recognition of real and unmodified objects comparing recorded images to stored image on recording devices. As the operator sees the images on real time, also, image processing and display of virtual objects **must be also real time**. So low latency is a compulsory requirement in AM.

CHAPTER 4. RADIO ACCESS TECHNOLOGIES

4.1. 5G NR

The term 5G, stands for fifth generation of mobile communications. This new standard, brings new possibilities of communication and enhanced communication services never brought before by any previous 3GPP standard (except for an evolution of 4G, which has small applications in industry environments). Specially, services never brought to environments like oil refineries, construction, energy generation, mining or manufacturing industries [16], like automotive or aerospace industry. All of them, having different requirements and possibilities on the implementation of the mentioned standard.

4.1.1. 5G NR Key Services

The previous generation of mobile communications, 4G, specified high data rates and better coverage to deliver good services to mobile communications. 5G, will continue improving data rates and coverage delivery, but its services will be extended to provide other services to environments much further away from mobile communications [13]. As mentioned, services that fullfill industrial requirements. The main three services that this new standard will deliver are [17]:

- **Enhanced mobile broad band (eMBB).** A service which provides high data-rates to devices which need, for example, to transmit or recieve images. 5G will extend its coverage and improve data rates. The improved data rate will be achieved using aggressive modulations as for example, 256-QAM, using new PHY techniques, like massive MIMO, exploring higher frequencies, over 6 GHz and including the millimetre-wave spectrum up to 100 GHz bands, where the amount of available bandwidth is much larger, and using wider channel bandwidths of, for example, 400 MHz.
- **Ultra-reliable low-latency communications (URLLC).** Previous 3GPP standards are not able to meet reliability and low-latency requirements for, for example, remote controlled machines like robotic arms or vehicles. For that, 5G offers the URLLC service which will be able to achieve very low latencies and very high reliability. Low latency requirements will be satisfied using solutions like reducing duration of transmission intervals, and high reliability will be satisfied combining more robust modulations and codings with mechanisms to improve transmission robustness, and adding redundancy.
- **Massive machine-type communications (mMTC).** mMTC aims to provide communication services to small, simple, low cost, and low consumption devices in very large quantities, for example, wireless sensor networks. Lots of devices that require great coverage, but small data rates and short transmission times. mMTC is able to provide service to more than 1 million of devices per Km^2 [17] and great battery life, thanks to discontinuous transmissions and long sleep mode times.

4.1.2. 5G NR Specifications Review

NR has to deal with requirements in many more areas apart from broadband mobile communications. For instance, the main one taken into account at this project: smart industry communications. In this section, the specification of 5G NR is reviewed [13] by means of the most important features and functionalities.

- **Carrier Frequency** (F_c)**:** NR considers wide bandwidth operation in a wide range of centre carrier frequencies ranging **from 1 GHz to 52.6 GHz** and under diverse spectrum sharing paradigms, such as licensed, unlicensed and shared spectrum. It is interesting to achieve higher frequencies (also know as the millimeter-wave (mmWave) region), specially where a large amount of bandwidth is available and necessary to achieve certain requirements, specially the ones which require **very low latency values** or **high data rate requirements**. For low-complexity devices that do not require high data rates and very low latency values, frequencies of the order of 1 GHz are considered. Also, as relatively low frequencies have better propagation, these sort of devices do not require more complex and more expensive antennas as would with devices that work in the mmWave spectrum.
- **Bandwidth** (**BW**)**:** As the amount of available BW is different when comparing low frequencies with mmWave frequencies, NR specifies different BW spaces depending on some (F_c) ranges. Table 4.1 shows the mentioned BW spaces size.

Table 4.1: Available BW spaces depending on F_c

Frequency Band	Subcarrier Spacing	Bandwidth space
0.45-6 GHz.	15-30-60 KHz.	50-100-200 MHz.
24-52.6 GHz.	60-120 KHz.	200-400 MHz.

- **Duplexing Schemes:** Generally speaking, NR can use mainly two duplexing schemes:
  - **Time-division duplex (TDD)**. More probable to be used in **higher frequency bands**.
  - **Frequency-division duplex (FDD)**. More probable to be used in **lower frequency bands.**
- **Channel Coding:** For channel coding in NR, a low-density parity check (LDPC) is used for data transmissions in which big data rates are required to transmit. To be more accurate, of the order of Gbps. For control channels instead, Polar Coding is used. In order to add more redundancy, a hybrid automatic request (Hybrid ARQ) is used in a similar way than with the LTE case.
- **Flexible Frame Structure and Numerologies:** Unlike LTE, NR supports multiple frame structures, also known as numerologies. All of them are referenced in table 4.2 with some important data related to them. Note that the LTE frame structure (see table 4.4) is a particular case of NR, when ignoring the slot concept: it corresponds to numerology $\mu = 0$ of NR.

Table 4.2: NR Numerologies Table

	$\mu = 0$	$\mu = 1$	$\mu = 2$	$\mu = 3$	$\mu = 4$
Frame length	10 ms				
Number of subframes / frame	10				
Subframe length	1 ms				
Number of slots / subframe	1	2	4	8	16
Slot length	1000 us	500 us	250 us	125 us	62.5 us
Number of OFDM symbols / slot	14				
OFDM symbol length	66.67 us	33.33 us	16.67 us	8.33 us	4.17 us
Resource Block BW	0.18 MHz	0.36 MHz	0.72 MHz	1.44 MHz	2.88 MHz

4.1.3. 5G Target performances

NR performances can be summarised and classified taking into account the three main NR services (or use cases):

- **Enhanced Mobile Broad Band (eMBB)**
  Performances wanted to achieve of this service can be summarised in table 4.3.

  Table 4.3: eMBB performances

	Down-link	**Up-link**
Maximum performance	20 GB/s	10 GB/s
Performance in 95% of cases	100 Mbps	50 Mbps
Latency	1 ms	

  The maximum-throughput performances will be achieved by implementing massive and multi-user MIMO.
- **Massive Machine-Type Communications (mMTC)**
  Deployments to offer this service use NB-IoT technology with a maximum inter-site distance of 1732 m. Device density is to be achieved, 1,000,000 devices per Km^2. Maximum battery live to be achieved, around 10 years or more.
- **Ultra-Reliable Low-Latency Communications (URLLC)**
  Performances standing out for this service, are to be 99.999% of reliability with less than 1 ms of latency.

4.2. Industrial 5G

5G has lots of applications and a better extended range of services. To understand how this services can be delivered, is important to review definitions and classifications of industry requirements. Requirements can be classified in three groups. Operational requirements, functional requirements, and performance requirements. **Operational requirements** describe how operation conditions of a certain application must be. For example, how simple a network to configure must be, how easy must be to manage, etc. Examples of functional requirements would be, Security, functional safety, authentication, etc. [17]. Performance requirements, can be said that are the most important ones, and are reviewed in what follows.

4.2.1. Performance requirements to industry

Performance requirements describe how services of a certain device or system are served. These are divided between dependability and security. Dependability also can be considered an operational requirement, and is divided into five requirements. Security describes how protected must the network be in order that their performance requirements do not get affected.

- **Dependability.** This requirement is described as if certain device fails, how much can affect its failures to the ability to work as intended and when intended of other devices. Is divided between the following requirements.
  - **Availability.** Can be considered a system to be available if is meeting desired served requirements like latency and throughput (Quality of service requirements). Availability may be quantified with the percentage of time any device or system is available.
  - **Reliability.** Reliability can be described as the amount of interrupted time a system is available. More interruptions, means the mentioned system is less reliable.
  - **Maintainability.** Maintainability is the ability that a certain device or system has to maintain or return to its desired operation state fulfilling its performance requirements.
  - **Safety.** A certain device or system can be considered to be safe, if do not compromise physical integrity both for the environment and workers or users.
  - **Integrity.** If data transmitted to any device of the system is transmitted and received without packet loss, corrupted and contaminated information, this system assures integrity.
- **Security.** Security consists in avoiding attacks to the network that compromise performance requirements. It was not a problem at all in wired networks isolated from internet, but as connectivity increased and future industrial networks will rely on connected and wireless technologies, these networks have become more vulnerable to attacks. These attacks can be classified depending on their proximity to the target network:
  - **Local attacks**
  - **Remote attacks**

And also, classified depending if they are logical of physical. Industrial networks must ensure security, specially to ensure that safety requirements are met, without compromising scalability, energy efficiency and low latency.

- **Quality of Service (QoS).** Quality of service is defined as the capacity of a certain service to be served within certain performance requirements. The most important QoS requirements can be:
  - **Throughput**
  - **Latency**
  - **Jitter**

  Throughput and latency, which are the most relevant performance indicators, are reviewed in section 5.3.1..

4.3. 4G LTE

4G, or Long-Term Evolution (LTE) is the last evolution of 3GPP standards before 5G. The main motivation for this standard was to increase its data rate up to 100 Mbps than its predecessor, increase its spectral efficiency, up to four times more, and reduce its latency, up to two or three times more. Also, keeping reducing its complexity and costs.

4.3.1. 4G LTE Specifications review

It is important to have a little bit of review of this standard, for the reason that 5G is an evolution of that previous standard, and shares some of its specifications.

- **Carrier frequency:** There are different available frequency bands, basically in sub 6 GHz frequency range, from 450 MHz to 3.4 GHz. The current ones that are used by deployed LTE networks are: 450 MHz, 470 MHz, 698 MHz, 700 MHz, 800 MHz, 1500 MHz, 1700 MHz, 1800 MHz, 2.1 GHz, 2.3 GHz and 3.4 GHz.
- **Bandwidth Specifications:** The following enumeration describes the main relevant things that summarise bandwidth specifications on LTE.
  - LTE Maximum Bandwidth of 20 MHz. LTE targets to make bandwidth flexible from 1.25 to 20 MHz. Actually, the different supported bandwidths are 1.25 MHz, 2.5 MHz, 5 MHz, 10 MHz, 15 MHz and 20 MHz.
  - Fixed resource block width of 180 KHz. Up-link resource blocks must be allocated continuously. When commenting about down-link resource blocks, these can be allocated in different frequencies and so either continuously or not, up to the scheduler implementation.
- **Channel coding:** Channel coding used on LTE for transmitting user data is Turbo coding, which determines what packets are corrupted by checking a certain value obtained with an operation with a bit which does not carry information, but the result of the operation. This bit is called soft bit, and if the computed result when received the information block

does not match with the soft bit, the information is considered to be corrupted. Turbo coding so, increases transmission robustness. For the channel encoding, LTE uses Parallel Concatenated Convolution Coding (PCCC), as is it also used with WCDMA and HSPA.

- **Frame structure:** LTE uses a fixed frame structure and OFDM configuration. The specifications of the LTE frame structure are shown in table 4.4

Table 4.4: LTE Frame configuration Table

Frame length	10 ms
Number of subframes / frame	10
Subframe length	1 ms
Number of OFDM symbols / subframe	14
OFDM symbol length	66.67 us
Resource Block BW	0.18 MHz

4.3.2. 4G Target performances

LTE general target performances can be summarised on the following points:

- **Throughput**: 100 Mbps of maximum down-link throughput, and 50 Mbps of maximum up-link throughput. Both of them, if propagation conditions are optimal and other factors that can affect transmissions are minimal.
- **Data-rate**: The maximum data-rate to be achieved with LTE-Advanced, is 100 Mbps in the high mobility case, and 1 Gbps in case of low mobility.
- **Round-Trip Delay Time** (**RTT**): maximum of 10 ms combined with an access delay of less than 300 ms.
- **Spectral efficiency**: increment up to 4 times more than HSPA combined with latency reduction.

4.4. IEEE Standards

The standard 802.11 is a very important standard to mention, because it is one of the standards to be used in the unlicensed spectrum, and also, the standard that mainly let internet networks to become wireless, which was something that really revolutionised how everything is connected to internet. As it is very important for the Intelligent Industry to have a wireless-edge communication standard to industrial networks, in other words, a wireless communication to the smart factory. Over the time, IEEE has been releasing versions of 802.11, the most important releases are displayed on following timeline, which shows the evolution of these standards until the current implemented and almost implemented standards, which can be said to be respectively standards 802.11n, and 802.11ac. These, are only mentioned and not reviewed. This is because simulations are not implemented following their specifications.

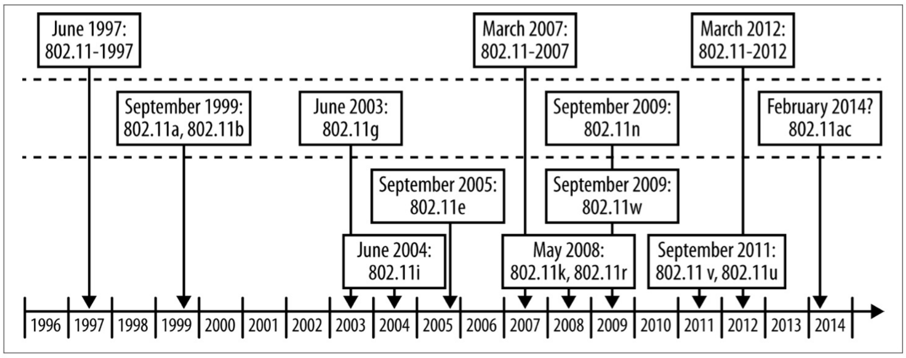


Figure 4.1: Timeline of other 802.11 releases [2]

The ones that are more relevant to take into account, are the last or future releases, which help to understand the transition of conventional and current wireless unlicensed spectrum networks, in other words, conventional Wi-Fi, to the wireless networks developed focusing more to IoT. For example, one of the requirements mentioned in bibliography reference [2], is the increment in throughput, which also high throughput is one of the services to be supplied to IoT. Nowadays, as mentioned, the two current and more modern implemented 802.11 protocols, are the 802.11n, the protocol released before 802.11ac and that same protocol, the 802.11ac.

The mentioned most relevant IEEE 802.11 standards commented are the following:

1. 802.11ac
2. 802.11ad
3. 802.11ax
4. 802.11ay

To summarise, the most relevant specifications of each mentioned standard, are collected on the Table 4.5.

Table 4.5: IEEE Most relevant 802.11 Standards table [2] [7] [8] [9]

Standard	**802.11n**	**802.11ac**	**802.11ad**	**802.11ax**	**802.11ay**
Channel Spacing	20 MHz, 40 MHz	80 MHz, 160 MHz (a)	2.16 GHz	80 MHz, 160 MHz (a)	From 2.16 GHz to 8.64 GHz (b)
Carrier Frecuency	2.4 GHz, 5 GHz	5 GHz	60 GHz	From 1 to 6 GHz	60 GHz
Maximum Speed	Up to 450 Mbps (c)	Up to 2.6 Gbps (d)	Up to 7 Gbps	(e)	Up to 100 Gbps
Modulation	64-QAM (f)	256-QAM (g)	64-QAM (h)	1024-QAM (i)	64-QAM (f)
# Spacial streams	4	8	8	8	8

(a) All channel sizes: 20 MHz, 40 MHz, 80 MHz, 80+80 MHz and 160 MHz.
(b) All channel sizes: 2.16 GHz, 4.32 GHz, 6.48 GHz and 8.64 GHz.
(c) With 3 spacial streams.
(d) Second implementation wave.
(e) four times more average throughput per user.
(f) This modulations including BPSK, QPSK, and 16-QAM.
(g) This modulation including 64-QAM, and modulations of (f).
(h) This modulation including BPSK and $\pi/2$-BPSK.
(i) This modulation including modulations of (g).

The parameters compared at this table, are focused mainly on the physical protocol layer, which is the most relevant one for the project.

CHAPTER 5. NETWORK SIMULATOR

5.1. Introduction

For the validation of NR (or other still not fully implemented standards) over planned scenarios, it is important to obtain acceptable results on the behaviour of the implementation of this technology following new technical specifications of 3GPP (other organizations in case of external protocols to 3GPP). In particular, in this project, focus is given to evaluate NR end-to-end (E2E) performance in I4.0 scenarios, and to determine whether it satisfies I4.0 requirements.

To validate the NR technology, there is no public validation methods or network simulators at all. Experiment results published by 3GPP are not reproducible or their results are not detailed in terms of performance and used models or assumptions. There are though, multiple link-level simulators for NR. However, to get good simulation results, is important to simulate all protocol stack layers to, not just evaluating PHY and MAC layers, but evaluating how PHY and MAC 5G-specified affect the other protocol stacks. Also, it is important for validating coexistence between other network technologies. Evaluating just PHY and MAC layers, as many simulators do, would mean that it has been taking assumptions of other layers behaviour. For some validations, results obtained in this way may not be acceptable.

Accordingly, for the NR evaluation in this project, it is decided to use the widely-known open source ns-3 network simulator with an extension to be able to support NR specifications, developed by CTTC, as presented in [3] and detailed next.

5.2. ns-3

ns-3 is a research-oriented, discrete-event network simulator, written in C++ with Python bindings. It has been under continuous development since 2005, courtesy of funding from the US NSF, INRIA in France, and several other public and private organizations. ns-3, and its predecessor ns-2 are the most frequently cited tools used in computer network research (based on a recent survey of journal and conference papers published in 2016 in the IEEE and ACM Digital Libraries). Both ns-3 and ns-2 are licensed under the GNU General Purpose License, version 2. This open source license is used for many software projects, including the Linux kernel.

ns-3 is a complex, multi-author piece of software that has undergone 29 software releases since 2008, and has been used for thousands of research papers and projects. ns-3 is a powerful tool that by one measure (academic citations) is already the leading packet simulation tool for 3GPP oriented network simulations. In particular, the ns-3 LTE module (developed and maintained by CTTC) appears to be the most popular packet-level simulator in use in terms of citations in publications found from digital library searches. The simulator is characterized by high fidelity implementations of the standard, especially from MAC (Medium Access Control) to APP (Application), and by a PHY (Physical) layer abstraction. In 2019, the CTTC released the first version of the first open source ns-3 NR simulator, based on an extension of ns-3. The model is a fork of the ns-3 LTE module and it mainly focuses on refactoring the PHY and MAC layers of LTE codes in order to provide a standard-compliant implementation of Release-15 NR. The RRC (Radio Resource Control) and upper layers, still rely, as of today, on the LTE implementation, as much as the EPC (Evolved Packet Core), which makes the available NR model a non-standalone implementation.

5.3. NR simulation overview

The ns-3 NR module [3] is the one selected for the E2E evaluation in this project. This simulator is programmed to simulate different communication and network devices including all layers of the NR communication protocol stack. Also, it lets the user design its own deployment layout of Base Stations (BSs, also known as gNBs in NR) and users (UEs), includes a variety of traffic application types, and offers the possibility of configuring different NR parameters, like the centre carrier frequency, bandwidth, numerology.

A general structure of E2E performed simulations is represented in Figure 5.1.

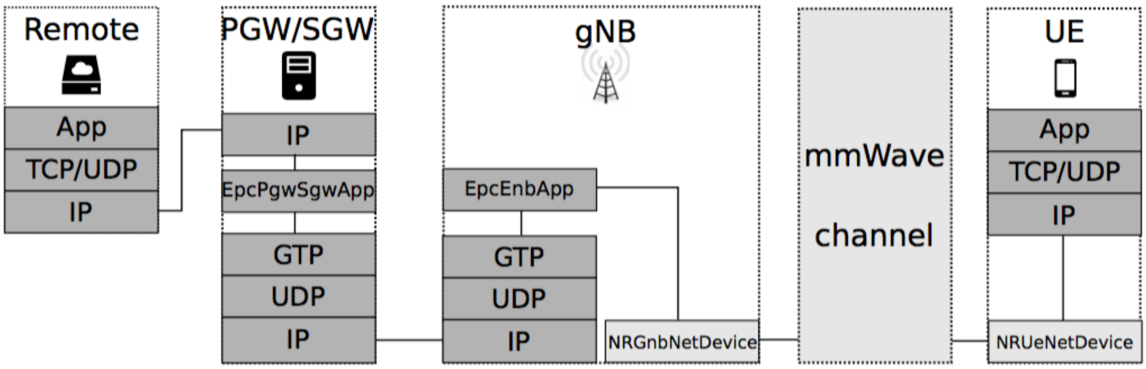


Figure 5.1: Simulation structure overview [3].

The simulated transmission process, consists basically about simulating all layers in the protocol stack of the communication devices, from a remote host to a mobile user (UE). Here is the simulated process of figure 5.1 more detailed and explained:

- For the down-link transmission, one or multiple remote hosts, send the information to transmit to the Service gateway(SGW)/Packet data network gateway (PGW). This information can be sent using multiple possible protocols.
- The information to transmit is encapsulated using General packet radio service Tunneling protocol (GPRS, GTP) and sent to the next-generation node B (gNB or base station) via IP.

- The packet received by the gNB is unencapsulated and transmitted to the UE by the Radio Access Network, through the NR radio access technology.
- If packet is received successfully at the UE (NRUeNetDevice), then it is sent to higher protocol stack layers and finally decoded.

For the up-link, the same process is performed but inverted. That is, from the UE to the remote host.

As key performance indicators (KPIs) that can be extracted from the simulator, in this project, two main KPIs are considered. Throughput and latency.

More detailed information about these KPIs, is provided in Section 5.3.1..

5.3.1. Performance indicators

As the main objective of the experimental part of this project is to evaluate some of the current and future wireless standards over aerospace industry requirements, some performance indicators are needed to have an idea if the simulated device deployments and configurations on every wireless device, are suitable enough to achieve their requirements, and also, to compare which deployment or configuration is the best among the others. As ns-3 is used to evaluate simulated networks performances, everything on this project is based around accomplishing performance requirements.

The most relevant performance indicators can be the ones that ensure QoS delivery, which also, affect to the rest of performance requirements. These performance requirements are commented in section 4.2.1.. From the QoS Requirements, this project focuses in:

- **Throughput:** The throughput performance indicator stands for the amount of data that is delivered per unit of time, from one point of the network to another. It cannot be longer than the generated traffic load, and at the same time it is upper limited by the system capacity. Figure 5.4 shows the relations between load, capacity, and the real amount of throughput transmitted.
- **Latency:** In the network context, latency stands for the time that it takes for a packet to travel between two places. For example, the end-to-end latency is measured as the time in which a packet travels from the application layer of one device, to the application layer of another device.

In order to understand all simulation results of this project, it is necessary to review how these performance indicators vary depending on certain situations or parameters of the devices and other concepts.

Figure 5.2 shows the most relevant layers of the NR protocol stack. Actually, they correspond to the layers of the devices, gNB (BS) and UE devices, shown on figure 5.1. The explanations on section 5.3.1.2., are taken into account figure 5.2

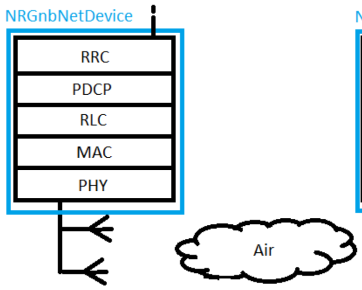


Figure 5.2: Protocol stack simplified picture.

5.3.1.1. Numerology and time slots

The NR radio resource grid is divided into **resource blocks**. Every resource block is composed of 12 subcarriers in frequency domain (which determine a **resource block width**) and a **time slot** in time domain. The amount of data that can be fed into one resource block, in ideal conditions, is the same, but NR introduces the concept of the numerologies, which enable changing the subcarrier spacing, and so the distribution of the resource blocks within the NR time/frequency grid. A representation of this distribution can be better understood by observing figure 5.3.

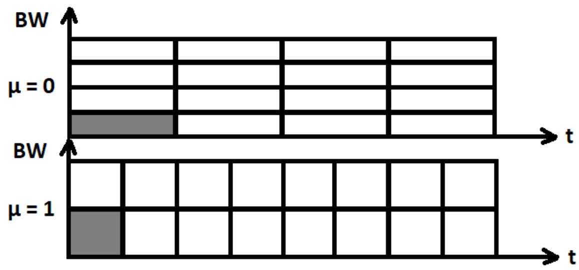


Figure 5.3: Representation on how numerology represents different frequency and time spaces.

As can be seen in the figure, every resource block is represented by a rectangle. The top figure represents numerology 0, and the bottom figure, numerology 1. The rectangle area is the same on both numerologies ($\mu = 0$ and $\mu = 1$). However, the resource block width in frequency and the time slot in time, vary with the numerology. The height of the rectangle represents the resource block width in frequency, and the width, represents the time slot length.

The difference between the two numerologies, is how the area is distributed over the NR time/frequency grid and how the grid of resource blocks is formed. For example, the time slot occupied by each resource block on the top figure, doubles the amount of time occupied by a slot of the bottom figure, and the amount of bandwidth occupied by each resource block on the top figure is half the amount of bandwidth occupied by a resource block of the bottom figure. When working with high F_c, because there is a good amount of bandwidth available, it is interesting to work with numerologies and resource blocks like the ones of the bottom figure, which take more bandwidth but take less time to be transmitted. And for situations in which there is not much frequency available, much better to work with resource blocks like the top figure. Based on that, the numerology, affects the latency. See how it is affected in section 5.3.1.2..

5.3.1.2. Contributions to latency increase

To better understand how latency occurs, it is important to review the most important delays that occur between packet transmissions from an application layer of a certain device to another application layer of another device.

The most important delays are:

- **Buffering Delay** The amount of time that packets spend waiting in the RLC buffers before the MAC layer indicates that such RLC buffer is scheduled for transmission.
- **Processing Delay** The amount of time that MAC and PHY layers need to process a packet, i.e., the time in between the point in which the packet is sent from RLC to MAC until the point in which such packet is in the air. In the ns-3 simulator, it is assumed that, as the scheduler works on a slot basis, the processing delays are proportional to the slot length (and so, numerology-dependent).
- **Transmission Delay** The time that it takes to transmit a packet through the wireless medium environment (i.e., over the air). The transmission time depends on the numerology, as explained in section 5.3.1.1..
- **RLC Reordering Time** At the receiver node, packets that come from the MAC layer are stored at the RLC layer which is in charge of reordering the packets before delivering them to the PDCP layer. The amount of time that it takes for a packet while waiting for the reordering, is called RLC reordering time.
- **Retransmissions** In case of data failures through the wireless medium, either due to bad propagation conditions (deep fading) or due to interference situations, the packet has to be retransmited. This incurs additional delays, associated to the retransmission(s) of the packet.

5.3.1.3. Contributions to throughput reduction

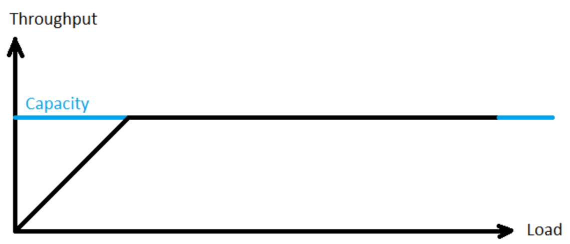


Figure 5.4: Representation on how the capacity limits transmission throughput.

At the beginning of section 5.3.1., throughput and capacity are mentioned to be related to each other. It can be considered that the relationship between throughput, traffic load and capacity, can be represented by the following Figure 5.4. As shown in Figure 5.4, throughput value follows the traffic load value until the system reaches its maximum capacity. Then, throughput is maintained at the maximum value even if the load increases, as the capacity is a system limiting factor and can not be surpassed. There are many factors that affect this capacity limit, the most relevant to understand the results of the simulation are the following:

- **Amount of available BW:** As commented in section 5.3.1.1., resource blocks take a specific amount of bandwidth and time each of them. If the same amount of data must be transmitted and the total BW available is reduced, more resource blocks must be assigned occupying more time slots. Consequently, taking more time to transmit the same amount of data, and thus, reducing the amount of throughput that is possible to be transmitted.
- **F_c value:** Free-space propagation gets better when F_c is lower. In other words, increasing F_c, makes the signal propagation worse. Therefore, in general, increasing F_c, reduces the useful signal power, and so increases the probability of errors, and so increases the number of retransmissions, and ends up with a throughput reduction. At the same time, reducing F_c increases the amount of interferences that are observed between different devices transmissions (due to the better propagation conditions). That could mean more probability of errors, which increases the amount of retransmitted packets. As a result of this, channel is occupied more time, which also increases the amount of transmissions that can also interfere other devices, and avoid other resources to be assigned at the same channel. For these reasons, also at lower F_c, the same amount of data takes more time to be transmitted.
- **Number of BSs (BS deployment):** If the number of BSs is increased, that could result in more transmissions that may interfere themselves. Therefore, the amount of data that can be transmitted in the same amount of time is reduced in case that multiple neighbour BSs transmit simultaneously, i.e., when the network load increases in dense deployments.

CHAPTER 6. TEST SIMULATION

6.1. Introduction

This project focuses on evaluating wireless communication standards for multiple devices that have diverse requirements and different data to transmit and receive. The easiest and more accurate way to determine whether a standard can meet the requirements of each type of device, is by simulating the factory environment, where there are multiple devices that need to fulfill their traffic requirements and performance targets. The simulation method used, is the NR ns-3 network simulator [3].

6.2. Software review and modifications

This section reviews how all the simulation is performed, and how results are taken and how are visualised to understand them and to get conclusions. The methods used to obtain the results of this project are the easiest to perform, not sometimes the most suitable, but are still sufficient to get wanted results. For modeling both simulation scenarios (this scenario and the scenario of chapter 7) , a certain example is modified. See 6.2.1. for more detailed information on the baseline examples, and then Section 6.2.2. for the code that has been modified for this project based on the baseline examples. For results and scenarios visualization, data obtained from simulations is displayed using some MATLAB scripts, which are described in section 6.2.2.3..

6.2.1. Relevant baseline examples

As mentioned in section 5.3., ns-3 has the advantage of setting new deployments or modifying existing examples with the aim of modeling a deployment as similar as the real scenario as possible. For that, in this section the two most relevant baseline examples are explained. Start commenting the simplest baseline example: ”cttc-3gpp-channel-simple-ran.cc”.

- **cttc-3gpp-channel-simple-ran.cc**
  This example is used mainly to test if the ns-3 parent directory and the NR module are installed and working correctly. The example consists in a single NR base station transmitting to a single mobile user, as shown in figure 6.1.

Figure 6.1: cttc-3gpp-channel-simple-ran.cc layout [4].

Some parameters of the transmission can be changed by introducing them in the $'Run'$ command followed by "−−". It is important to be able to change them, in order to ensure that they fullfill every requirement, depending of the simulated service, deployment parameters, and also, it provides a practical way to change them quickly. This is a simple example, so the parameters you can introduce this way are **Numerology**, **Packet size**, and **transmission direction** (**up-link or down-link**). The instructions on the code that initialize these variables are, in lines 57 of the code listed in the appendices which also contain their default values. See section C.1.1.. Depending on the variables that the user wants to evaluate quickly after any simulation, in this case, both performance indicators, i.e., throughput and latency, it is interesting to have a look on the code instructions of lines from 129 to 133. These lines show some variables on console after simulation is ended. For example, if only throughput and latency are needed, the instructions that show more variables can be deleted in order to have a clean output of the results after simulation ends.

- **cttc-3gpp-channel-nums.cc**
  Another relevant example to talk about is "cttc-3gpp-channel-nums.cc". This example not only simulates the RAN layer, but also simulates all E2E layers of the communication network. This case, on its simplest configuration, radio transmission consists of a NR BS transmitting to 1 UE, but multiple UEs can be deployed. See figure 6.2 which shows a representation of the configuration with 1 BS and 2 UEs.

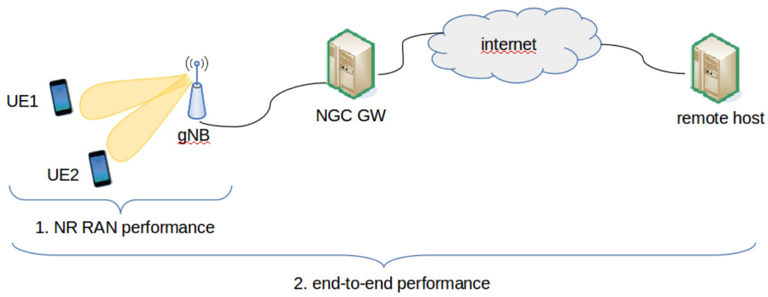


Figure 6.2: cttc-3gpp-channel-nums.cc layout [4].

As for the simple ran example, it is also possible to change certain parameters. As this example is much more complete, more parameters can be modified. This time is possible to change how many UEs can be associated to a BS for example, providing different layouts. The simplest configuration is the default value shown in Appendix C.1.2. on code lines 126 to 129, is 1 UE per BS. Also, the rest of default values can be modified with the console instruction ” – –”. All of them are summarised on table 6.1.

Variable	Default Value	Code Lines
Frequency	28 GHz	74-77
Bandwidth	200 MHz	79-82
Numerology	2	84-87
UDP interval	0.01 s	89-92
UDP Packet Size	150000 Bytes	94-97
MCS Fixed	True	111-114
Fixed MCS Value	28	116-119
BS number	1	121-124
Number of UEs per BSs	1 UE/BS	126-129
Transmission power	1	142-145
Sim Tag	”default”	147-150
Output Directory	Root folder	152-155

Table 6.1: The most relevant transmission parameters with their default values of the example cttc-3gpp-channel-nums.cc

Frequency and bandwidth are mainly the most important parameters to modify. These parameters are used to ensure that a service can be provided within possibilities of the network factory to provide certain bandwidths or frequencies. At the simulation, certain frequencies for certain services give better performances than others, so is important to ensure that the network uses frequencies and bandwidths that satisfy service requirements as much as possible.

Other parameters like UDP Interval and UDP packet size are also important, particularly to simulate traffic loads of each user. Both parameters are mentioned at the same time because data rate generated at the application layer is obtained from these two. UDP interval is the time separation between packets, and packet size is the actual amount of bytes of every packet (at application layer, without taking into account lower-layer packet headers). The data rate of generated packets at application layer is given by:

$$Datarate = \frac{PacketSize \times 8bits + 224bits}{UDPInterval}, \tag{6.1}$$

where $PacketSize$ and $UDPInterval$ are the mentioned parameters, in bytes and seconds, respectively. The 224 bits are the header size, and the 8 bits are used to convert packet size from bytes to bits. In order to introduce data rate requirements on the simulator, a fixed packet size is used, and a UDP interval is computed using equation (6.1) to provide a specific application data rate based on the traffic type.

Traffic congestion and coverage loss are factors that directly affect to the real data rate provided to every individual UE, so being able to set this parameter and being able to know the real data rate served, is used to check if data rate requirements are met. On the first simulation, for example, one of the requirements set, is, an average delivered throughput of 95 % of the actual generated data rate. These results are mentioned in table 6.4 on section 6.4.

For different layouts, this example has an algorithm to deploy more UEs and BSs if desired. Depending on the amount of BSs and and UEs per BS, their layout is spread following a certain pattern programmed in lines 295 to 332.

Finally, this example also has the possibility to save in a file the simulation performance results, containing the average throughput and the average latency results of the UEs, which are the most relevant results and the ones used to take conclusions on all deployments. It is also useful to have transmission data from every UE, in case that the average throughput or average latency, do not get an expected result. Some UE's bad performances may make vary the average performance of the rest of the deployment in some cases, so it is interesting to have performance data from each and every UE to BS connection. An example of a saved file, is listed in the appendix D.

6.2.2. Modifications over the cttc-3gpp-channel-nums example code.

As mentioned previously in section 6.2.1., cttc-3gpp-channel-nums.cc is a perfect code to modify to get the desired results of the planned deployments, as it has the possibility to have a complete E2E evaluation, and also, the possibility to set enough performance requirements to have a good evaluation for deployments in which only one carrier frequency and one bandwidth are simulated at the same time.

Modifications over the example code, can be classified depending on the parts of the code where they have been carried out to get simulation results. These are the main modified parts:

- **Default variables declaration** Variables commented on table 6.1 Basically, what it is done at this part of the code, is that some of the default values are changed, and some unused variables declarations deleted. Also, some needed variables are added. The details of the modification are explained in section 6.2.2.1.
- **BS and UE layout creation** The algorithm commented on section 6.2.1. allocates BSs and UEs depending on two variables that account for the number of BSs and number of UEs per BS. This algorithm is removed to allocate a new algorithm. The new one commented in section 6.2.2.2., generates certain positions for the BSs, and allocates randomly the UEs with in a certain space and number previously introduced. This new algorithm is the one used for the two scenarios. The one in section 6.3.1. and the one of section 7
- **Display of simulation results and BSs/UEs positions.** The original version of cttc-3gpp-channel-nums.cc does not display the average throughput and average latency, instead, these figures are directly stored on the mentioned file D. A modification is introduced to

display these values on the console after ending the simulation, in order to perform tests more easily, and to get the desired values to later on, use them to make results plots with Matlab scripts B. Generating a results file for every simulation configuration is something to be considered messy and unpractical in case of having to organize data to be compared. Also, in order to ensure that the obtained results are coherent, another modification is introduced to display UEs and BSs positions on the console for every configuration. For that, it is ensured that the UEs have the desired positions and these positions are held during all variations on every scenario. More information in section 6.2.2.3.

- **Uplink-traffic.** Another relevant difference between the cttc-3gpp-channel-nums example and the cttc-3gpp-simple-ran example, is that the later has a parameter to change the data flow direction. As there are also requirements for up-link performances, it is also important to evaluate not only performances on the down-link but also in the up-link direction. cttc-3gpp-channel-nums, as mentioned, does not have this option, so a modification is introduced to make this example suitable to analize up-link performances as well. This mentioned modification is explained in section 6.2.2.4.

6.2.2.1. Default variables declaration and modifications

Basically, as the new algorithm of UEs allocation is completely independent of BSs, the variable which associates a determinate number of UEs per BS, see lines 126 of C.1.2., $ueNumPergNB$, is converted to $ueNum$, which stands for the number of UEs on the desired deployment. See this change in C.1.3. in lines 132 to 135. Another variable that is eliminated, is $singleUeTopology$, as in our case, number of UEs are between 10 to 20 and even 60 in case of the second simulation, so a single UE topology variable is no longer needed.

6.2.2.2. BS and UE layout creation

The example cttc-3gpp-channel-nums, was intended to simulate multiple configurations in an easy way, specially, to simulate many layout configurations with only three parameters. This original code, has the possibility to be configurated from, as the example 1, just one UE which connects to one BS, to $(AsmanyBSandUEsasdesired)$*, spread in a certain way as described on lines 275 to 332 of code C.1.2..

As layouts wanted to simulate are totally differently distributed, this fragment of code is replaced with the one of lines 289 to 333 of the modified code C.1.3.. This code is modified according layout requirements described in section 6.3.1., which consists in two BS configurations, and from 10 to 20 UEs spread randomly with in certain boundaries. For the randomly generated coordinates, the function $rand()$ is used, which generates integer random coordinates for the users but these coordinates are maintained through different simulation variations while maintaining the same number of UEs. Another thing worth mentioning, is that all the simulation layouts have two simplifications between the planned scenario and the one which is simulated. The differences stand for, the UE positions are integers, and the BS positions are moved 0.5m to the positive x axis, and 0.5m to the y axis, in order not to make them too close to certain UEs. As this is software makes theoretical computations, if one or some distances between a BS and a UE reach 0, could result in frequent computations errors which are not convenient. This is a common limitation when working on random points and these points can not get certain values between all possible values that the randomly-generated variable can get, so it is a simple but effective solution.

6.2.2.3. Results and deployments visualization (Matlab code)

Matlab codes are used for simulation information and results displayment. To be more accurate, mentioned codes are used to:

- Generate the deployment scenario overview, either to show how BSs are deployed, to show how UEs are deployed, or to display both UEs and BSs deployed at the same time.
- Generate results plots for better understanding and to compare the information provided by the simulations.

Further details on Matlab codes are not relevant at all, but are included in Appendix B. Code on section B.1.1. is the one that generates plots of BS and UEs layouts. Code on section B.1.2. is the one that generates the results plots.

6.2.2.4. Switching to up-link traffic

$cttc-3gpp-channel-nums$ does simulate down-link traffic by default, as mentioned at the beginning of section 6.2.2.. For that reason, some lines of its code must be changed, so a different example from this one can be created, that instead of simulating down-link traffic, simulates up-link traffic. In other examples like the mentioned $cttc-3gpp-simple-ran$, an up-link variant is not necessary because it has a native up-link traffic support. The modifications carried on to enable this example as an up-link example, are the following:

$serverApps.Add(dlPacketSinkHelper.Install(ueNodes.Get(0)));$,
$UdpClientHelperdlClient(ueIpIface.GetAddress(j),dlPort);$,
$clientApps.Add(dlClient.Install(remoteHost));$,
$dlpf.localPortStart = dlPort;$ and
$dlpf.localPortEnd = dlPort;$

Are replaced by:

$serverApps.Add(dlPacketSinkHelper.Install(remoteHost));$,
$UdpClientHelperdlClient(internetIpIfaces.GetAddress(1),dlPort);$,
$clientApps.Add(dlClient.Install(ueNodes.Get(j)));$,
$dlpf.remotePortStart = dlPort;$,
$dlpf.remotePortEnd = dlPort;$,

The replaced code lines can be found on the listed code C.1.4.. These lines are 389, 393, 411, 417 and 419 respectively.

6.3. End-to-End Evaluation

To have a simple example to start getting some reliable results it is necessary to start step by step, understanding how the simulator works, how to modify its codes to generate desired examples and also to understand its results. We focus on assessing the impact on the end-to-end performance of specific parameters to see how the communication is affected by changing them, and also, the whole layout is a very simplified layout of a factory, which is easy to simulate but enough to provide good results.

6.3.1. Simulation Scenario

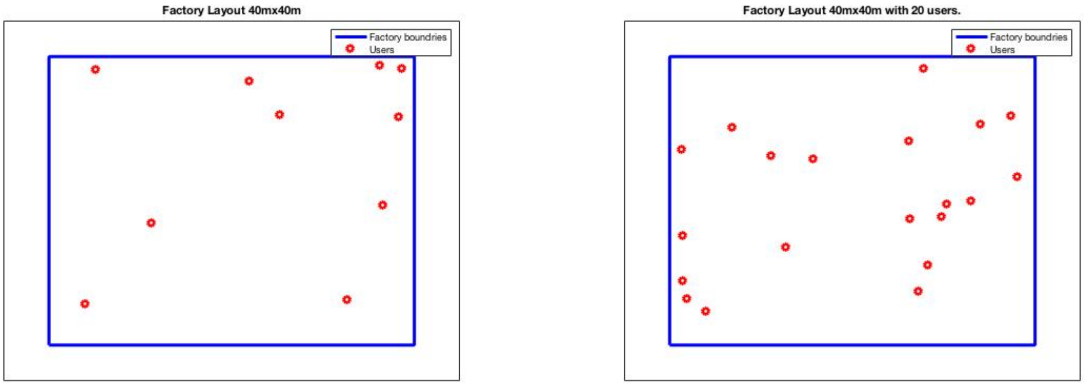
The following figures show how the simulation example is. The factory consists in a facility sized 40m x 40m. The deployment scenario consists on a specific number of UEs spread randomly around the factory, and a specific number of BSs to serve the UEs. Four variations are considered for the factory layout, depending on the mentioned parameters.

- **Number of UEs**, 10 or 20 distributed randomly within the factory (see figure 6.3).
- **Number of BSs**, 2 or 4 placed in fixed positions as shown on figure 6.4

Different scenarios for the traffic are considered:

- **transmit direction (download/upload)**, scenarios where each UE needs to download or upload a certain amount of traffic. For downloading, the UEs receive data packets from the remote host through the BSs. For uploading, the UEs transmit data packets to the BSs, to finally reach the remote host.
- **traffic load (0.8 and 16 Mbps)**: the amount of data traffic the UEs need to download/upload vary from 0.8 Mbps up to 16 Mbps.

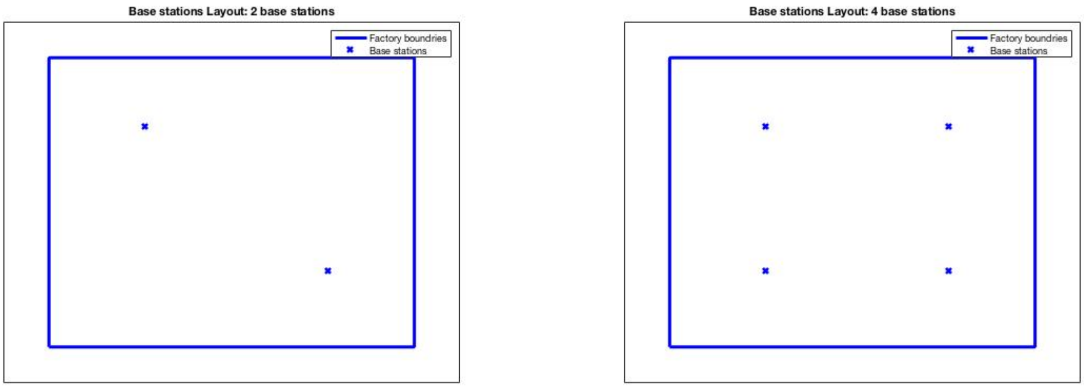
For simulating the load, two parameters are taken into account: 1) Inter Packet Arrival Time (IPAT) (in seconds), which is the amount of time in between different packet generation, and 2) packet size (in bytes), which is the amount of data on every packet. The total load is the division of the packet size and the IPAT value. The simulation is done with a constant IPAT value of 0.01s, and a varying packet size.



(a) 10 UEs factory layout

(b) 20 UEs factory layout

Figure 6.3: Factory layout: UEs deployment example



(a) 2 BSs layout

(b) 4 BSs layout

Figure 6.4: Factory layout: BSs deployment example

Doing this simulation, as mentioned before, the capabilities of the network must be proved in order to verify that is capable to handle all the devices without compromising the requirements of the system. The two key performance indicators that are evaluated are:

- the **average E2E throughput** that every UE is capable to transmit to/receive from the network, and
- the **average E2E latency**, also measured for every UE, which corresponds to the average amount of time that takes to transmit/receive a packet.

Small sensors for example, do not require a lot of information to transmit, but in some cases, a noticeable delay on its data transmission is not convenient, not like for example, images from a security camera, that some seconds of delay will not affect anything at all, but a lot of data traffic must be transmitted. Depending on the results of the simulation, selection of certain parameters will be done in order to make their data transmission as much efficient as possible.

It is also important to mention that, in a network, frequencies and bandwidths must be assigned intelligently, due to the propagation of electromagnetic fields is different depending on the carrier frequency, and also the amount of information that can be transmitted depends on the available resources, and correspondingly on the bandwidth. The available bandwidth also varies depending on the carrier frequency that is used. For example, there is much more bandwidth available at 28 GHz or 60 GHz than at 2.4 GHz frequency, which is important to know, because there are many devices like, for example, the ones that transmit images, they need a lot of throughput, and thus, more bandwidth is required if keeping their latency low is desirable.

This end-to-end evaluation shows how throughput and latency change depending on the network deployment (number of BSs and the related parameters: frequency and bandwidth) for different traffic requirements in terms of number of UEs, generated traffic load per UE, and transmit direction. The simulation configurations depending on all the parameters are illustrated on the following tables.

Table 6.2: Configurations for traffic requirements.

number of UEs	**traffic load (Mbps)**	**transmit direction**
10	0.8	Down-link
20	0.8	Down-link
10	16	Down-link
20	16	Down-link
10	0.8	Up-link
20	0.8	Up-link
10	16	Up-link
20	16	Up-link

Table 6.3: Configurations for network deployment.

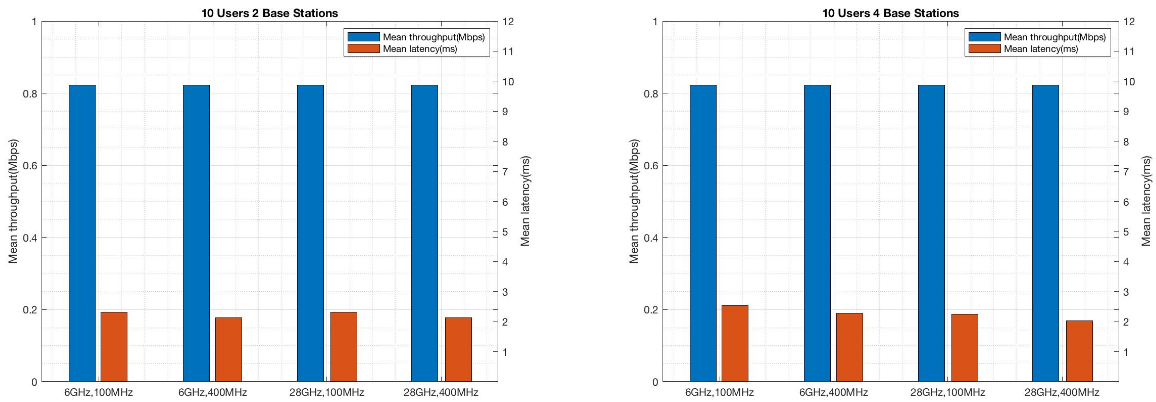
number of BSs	**carrier frequency (GHz)**	**bandwidth (MHz)**
2	6	100
2	6	400
2	28	100
2	28	400
4	6	100
4	6	400
4	28	100
4	28	400

6.3.2. Simulation Results

All the simulation results are collected on the next figures. The exact numbers taken from the simulation, in which all results plots are based of, are shown on the tables in the appendix A

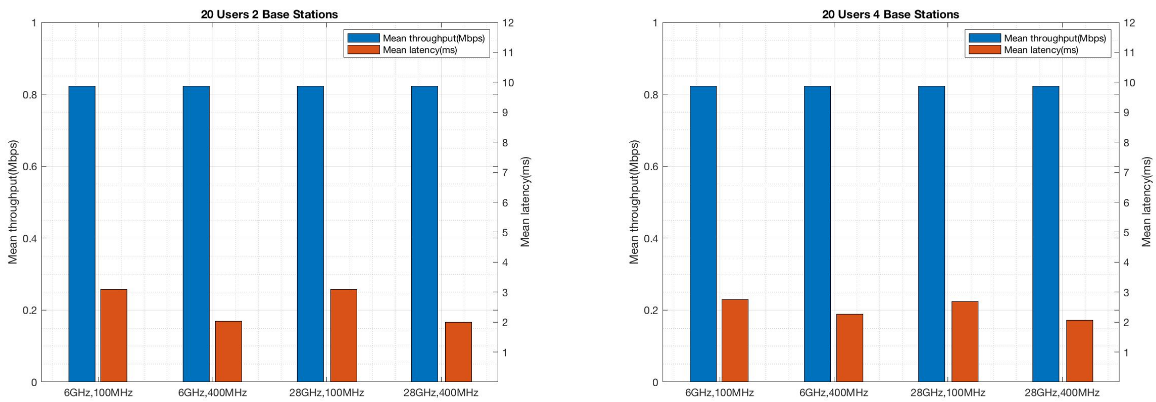
Figure 6.5 and figure 6.6 show the results for low data traffic (0.8 Mbps) in the down-link direction, in case of 10 UEs and 20 UEs, respectively. Figure 6.7 and figure 6.8 display the results for high data traffic (16 Mbps) in the down-link direction, in case of 10 UEs and 20 UEs, respectively.

Then, for the up-link, figure 6.9 and figure 6.10 show the results for low data traffic, (0.8 Mbps), in up-link direction, of 10 UEs and 20 UEs respectively, and on figure 6.11 and figure 6.12 can be seen the plotted results of high data traffic in the up-link direction. Also, respectively 10 UEs and 20 UEs.



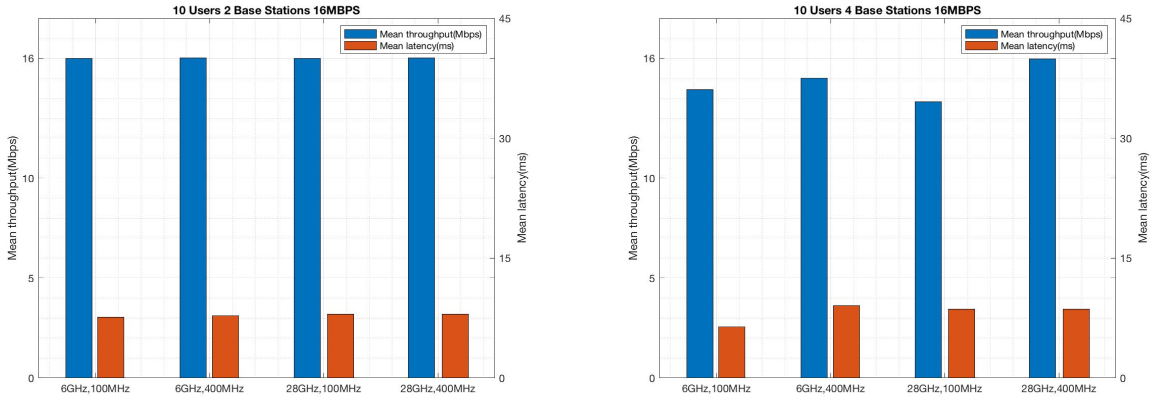
(a) 10 UEs, 2 BSs. (b) 10 UEs, 4 BSs.

Figure 6.5: 10 UEs downloading at 0.8 Mbps each one.



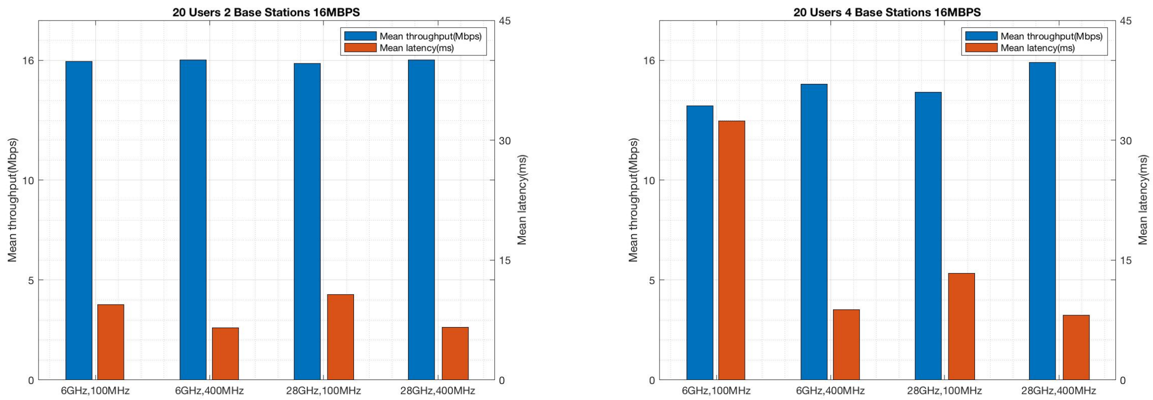
(a) 20 UEs, 2 BSs. (b) 20 UEs, 4 BSs.

Figure 6.6: 20 UEs downloading at 0.8 Mbps each one.



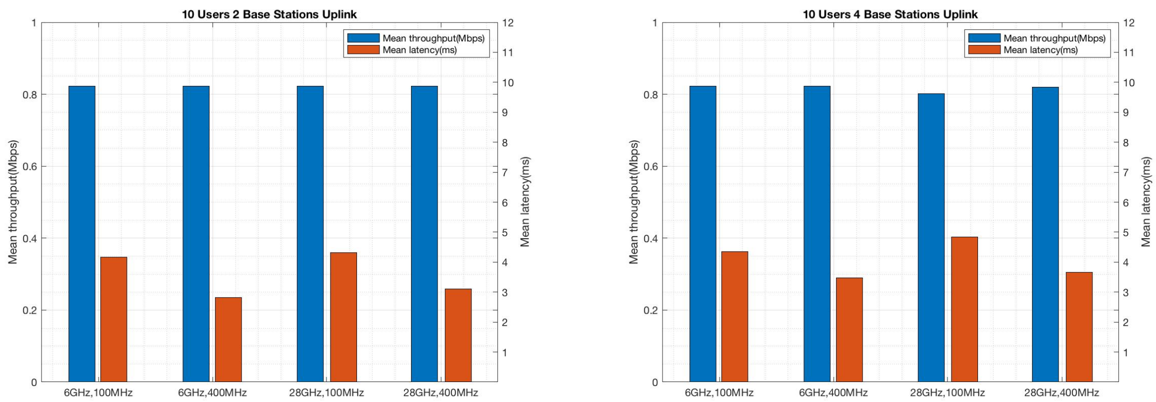
(a) 10 UEs, 2 BSs (b) 10 UEs, 4 BSs

Figure 6.7: 10 UEs downloading at 16 Mbps



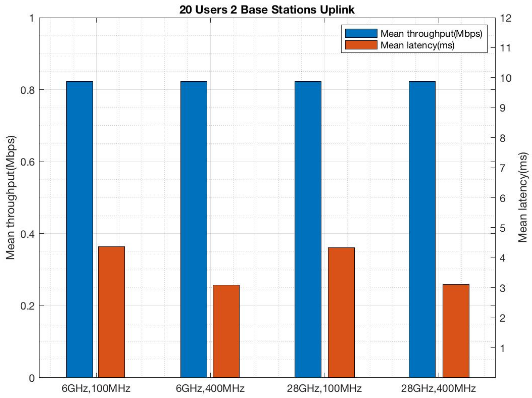
(a) 20 UEs, 2 BSs. (b) 20 UEs, 4 BSs.

Figure 6.8: 20 UEs downloading at 16 Mbps.

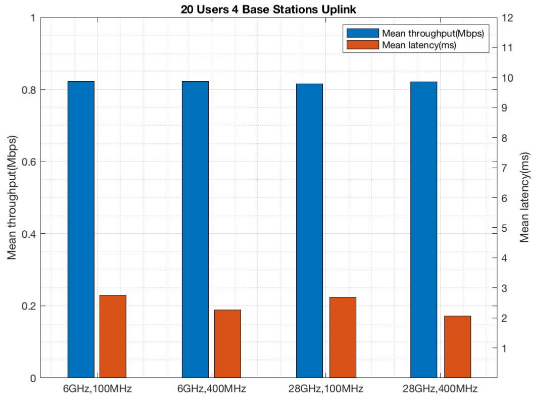


(a) 10 UEs, 2 BSs. (b) 10 UEs, 4 BSs.

Figure 6.9: 10 UEs uploading at 0.8 Mbps.

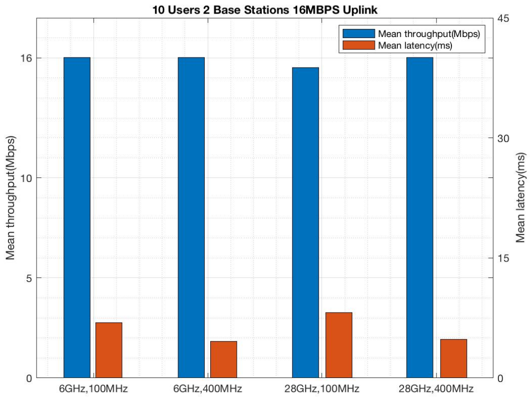


(a) 20 UEs, 2 BSs.

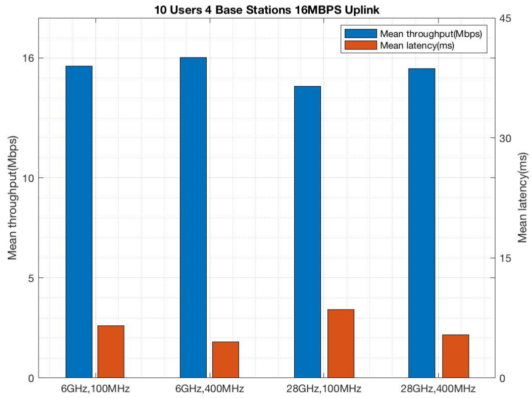


(b) 20 UEs, 4 BSs.

Figure 6.10: 20 UEs uploading at 0.8 Mbps

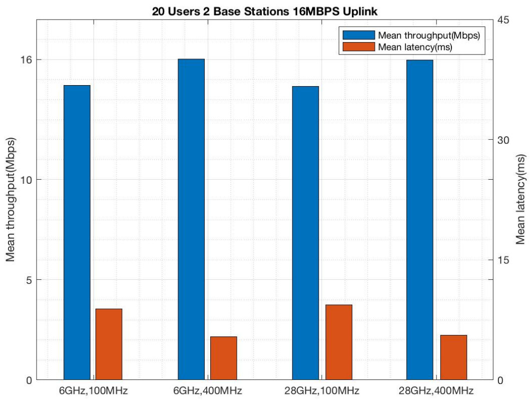


(a) 10 UEs, 2 BSs

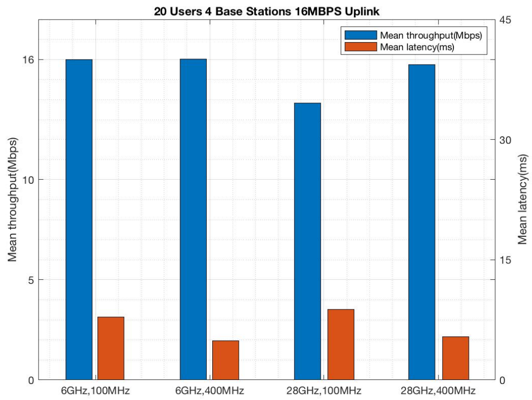


(b) 10 UEs, 4 BSs

Figure 6.11: 10 UEs uploading at 16 Mbps



(a) 20 UEs, 2 BSs.



(b) 20 UEs, 4 BSs.

Figure 6.12: 20 UEs uploading at 16 Mbps.

6.3.3. Comments on the results

Once results have been presented, it is interesting to take into account the following observations in order to understand better many conclusions.

In figures 6.5, 6.6, 6.8, 6.9, 6.10, 6.11 and 6.12, focusing on the mean latency, it can be seen that in all cases, when increasing the bandwidth from 100 MHz to 400 MHz, the latency is reduced. Also, in some cases, when all the traffic load could not be delivered with 100 MHz bandwidth, the throughput is increased and the data traffic can go through the channel in the 400 MHz bandwidth case. It must be taken into account that the available resources for receiving/transmitting the packets can be distributed either in frequency- or time-domains. For example, it is desirable to spread the transmission in more bandwidth, so that less time will take to receive/transmit a packet because there are more subcarriers containing information. One of the disadvantages of using more bandwidth is that less bandwidth is left for other transmissions, so the transmission are more susceptible to interference and also, the transmission power per resource block decreases when the bandwidth is increased, so transmission can be even worse in some cases.

A different effect is observed though, in figure 6.7 for the case of 4 BSs. Therein, stepping from 100 MHz to 400 MHz of bandwidth, makes an increment of the mean latency to transmit a packet. This is counter-intuitive, because there is better performance in all situations when increasing the bandwidth. But this case is a particular case. By having a close look at the concrete simulations, it is observed that, with 100 MHz, there is one UE that is not served because it has very bad propagation conditions, and even with retransmissions it cannot manage to receive any packet. As there is no data reception, the performance of this UE does not affect the average latency. Instead, the UE receives data, when increasing the bandwidth up to 400 MHz, for which that UE starts receiving some information because the code block size is larger and the block error rate for decoding the packets and combining the retransmissions is reduced. But, due to the poor channel conditions that this UE experiences, the amount of throughput is low, and the latency shoots up that much, that makes a noticeable increment on the average latency. All in all, it is a matter of the latency statistics over the received packets, because in this case it seems it is better having worse average latency than not serving the UE.

The frequency is a parameter that affects the transmission. Propagation is better, as long as frequency is lower, so if working on two different frequencies, makes a difference depending on which value is chosen. For example, in figure 6.8(b), can be observed that increasing the frequency, reduces latency and increases throughput. This is something unexpected, because there are worse propagation conditions at higher carrier frequencies. However, not only the useful signal power is degraded when the frequency increases, but also the interfering signal's power. Consequently, the whole performance is improved in situation of figure 6.8(b), because interferences are significantly reduced and thus, the SNR increased. This attenuation effect is not convenient in all situations. For example, if we focus on figure 6.12(b), as the frequency is increased, the latency is slightly increased and throughput is also, slightly reduced. At this situation, transmitting at a higher frequency, attenuates signal too much, does not help too, the fact that transmission of UEs is less powerful.

Between up-link and down-link (see comparisons between figures 6.5 and 6.9, 6.6 and 6.10, 6.7 and 6.11 and 6.8 between 6.12), under the same number of BSs, UEs, bandwidth, frequency, and traffic load, a larger latency is observed in the up-link case. This is because when downloading, BSs do not have to request access to proceed. Instead, in NR up-link, by following a dynamic scheduling, there is a message exchange in between the UE and the serving BS, in which the UE requests access, the BS assigns resources to transmit, and the UE finally transmits on the scheduled resources. The situation of having to ask for access to upload traffic, is what generally makes up-link transmissions slower in terms of latency, as observed in the results.

Observing the difference between BSs configurations (6.4 and 6.6), there are many things to comment about. Getting the best performance, doesn't depend only on increasing the number of BSs, it also depends on the number of UEs, generated traffic load and also if the transmission is either an up-link or down-link. For example, in figures 6.6 and 6.10, performances are better when having more BSs in case of having many UEs. As there are more BSs, traffic is distributed between the BSs, and we can see a small improvement on latency. Something totally opposed happens when these UEs demand high amount of data traffic. Better results are got, if 2 BSs are used instead of 4. When having 4 BSs and a high load, the amount of interference between all transmissions are that high, that more packets get corrupted, and consequently the latency increases because retransmissions are needed to complete a packet successful reception. This last situation gets better, when transmitting in small bandwidth. The specific power increment when reducing bandwidth helps to increase the SINR (Signal-to-Interference-plus-Noise Ratio) and also, keeping higher frequencies help, due to the worse propagation conditions. Focusing on up-link transmissions, on the situation of having low network load and a small number of UEs, latency is increased if 4 BSs are deployed (see figure 6.9), something totally opposed happens on down-link transmissions. See also, if the number of UEs is increased, a larger number of deployed BSs help in improving the system performance, see figures 6.10 and 6.12.

6.4. Simulation conclusions

Taking into account the results and comments mentioned before, now we focus on extracting meaningful conclusions.

First, is determined which is the optimal configuration for each combination of requirements (i.e., number of UEs, traffic load, and transmit direction). The optimal network configuration refers to the number of BSs, bandwidth and carrier frequency. To determine the optimal configuration, is used a two step approach:

1. The one that leads to the best throughput and latency system performance.
2. If different configurations provide the same performance, is selected the optimal one as the one that is cheaper from the network deployment perspective.

On the first approach, different performance parameters are taken into account to decide if a certain configuration is more optimal than others, depending on their throughput requirements, due to are considered to offer different services. for 0.8 Mbps will be considered preferable the best latency, and for 16 Mbps, is considered preferable the biggest delivered throughput. An example of second approach would be, using lower bandwidth preferable than using a larger bandwidth; using less BSs is preferable as well; and finally is assumed that lower carrier frequencies are more expensive than higher frequencies due to the lack of available spectrum therein, so higher carrier frequencies are preferable.

As a second important output of the results, is to determine whether a configuration is acceptable or not, by considering some target thresholds on the expected throughput and latency performance.The threshold for throughput is a 95 % over the required one (in the IP layer) and the one for latency is 3ms, in case of down-link and 10 ms in case of up-link (see 7th column of Table 6.4). Threshold % is calculated based on the amount of throughput obtained by the simulation, and the required one in the IP layer to be delivered. The required one, taking into account an IPAT of 0.01s, which means 100 packets per second are generated, headers take 22.4 kbps, so it is 0.8224 Mbps in case of transmitting 0.8 Mbps, and 16.0224 Mbps in case of transmitting 16 Mbps. This means, that if the received throughput is, for example 16.0224 Mbps, 100% of required throughput is delivered. Last two columns of Table 6.4 show if commented requirements are met. For more information about performances of optimal configurations, see Table A.9.

Table 6.4: Results table

Configuration			Optimal deployment				Target met	
DL/UL	**Load**	**UEs**	**BSs**	**BW**	F_c	**Latency target**	**Throughput**	**Latency**
Down-link	0.8 Mbps	10	4	400	28	3ms	✓	✓
Down-link	0.8 Mbps	20	2	400	28	3ms	✓	✓
Down-link	16 Mbps	10	2	400	6	10ms	✓	✓
Down-link	16 Mbps	20	2	400	6	10ms	✓	✓
Up-link	0.8 Mbps	10	2	400	6	3ms	✓	✓
Up-link	0.8 Mbps	20	4	400	6	3ms	✓	✓
Up-link	16 Mbps	10	4	400	6	10ms	✓	✓
Up-link	16 Mbps	20	4	400	6	10ms	✓	✓

As it can be observed in Table 6.4 and Appendix Table A.9, taking into account commented requirements, all optimal configurations met the mentioned requirements.

CHAPTER 7. REAL-BASED DEPLOYMENT SIMULATION

7.1. Introduction

Once the Test Simulation is done, it is time to analyse a more close-to-reality case. This time, NR technologies are evaluated over a simulated scenario which consists in a aircraft maintenance hangar. The workers of this hangar use a determinated set of smart and connected tools which are similar to other wireless-connected devices of the smart industry or other fields in which these smart technologies are also implementable.

The mentioned hangar is the Iberia Maintenance hangar located in Barcelona-El Prat Josep Tarradellas Airport, shown in Figure 7.1. Approximate sizes of the building are taken in order to size simulation boundaries, and connectivity of some possible smart tools is evaluated in order to ensure that all tools can work within a certain amount of bandwidth and carrier frequencies.

Figure 7.1: Iberia hangar. Source: [5]

7.2. End-to-End Evaluation

Aerospace maintenance, is a conventional and tiring task that needs to satisfy strictly safety and time requirements, which if not satisfied, could result in compromising safety and significant amounts of losses for the airline or the maintenance centre. As current aerospace industry relies mainly on human sources, technologies that help to reduce workers workload, can help to improve safety and thus, cost of maintenance operations.

In this section, a model of the technologies mentioned on section 3.3.1. is proposed. Their connectivity requirements are evaluated with ns-3 in order to get some clues to know if connectivity requirements of these technologies can be successfully achieved in the proposed scenario.

7.2.1. Proposed scenario

As mentioned in 7.1., the proposed scenario is the Iberia maintenance hangar located in Barcelona-El Prat Josep Tarradellas Airport. This hangar is one of the most important maintenance centres all around Spain, being the only one in the whole country that could allocate an Airbus A380 when opened in 2010 [5]. As being such an important maintenance centre, the airlines in which their maintenance operations are performed in this hangar, represent a high percentage of flights that operate at this airport. Therefore, it is of a paramount importance to streamline maintenance operations at the best.

The capacity of this hangar can be approximated by one Airbus A380, two Airbus A330 aircrafts, and five Airbus A320 aircrafts. According to Iberia, which is the company that owns this facility, a total of 60 people work on every aircraft [18]. Taking into account all of this, a simulated scenario can be represented by a maintenance of an A380, in which 60 people are working . It can be assumed that all of these maintenance crew people use a smart tool, and only 20 of them are using augmented reality glasses, and one aircraft is currently on service. The requirements of these devices can be modeled as in Table 1 of document [6]. This table, table 2.1, is also listed in section 2.2.. Then, the requirements of the mentioned scenario, are summarised on table 7.1.

7.2.1.1. Device Requirements

Table 7.1: Requirements of simulation devices based on table 2.1

Scenario device	Equivalent industrial device	Latency (ms)	Throughput(bps)	Number of devices
Smart tool	**Industrial robot**	< 1	$\sim 10^3$	60
AR set	**Head mounted display**	< 10	$\sim 10^6$ - 10^9	20

Table 7.2: Specified requirements values used in the simulation

Device	Latency	Throughput
Smart tool	1 ms	10 Kbps
VR Headset	10 ms	10 Mbps

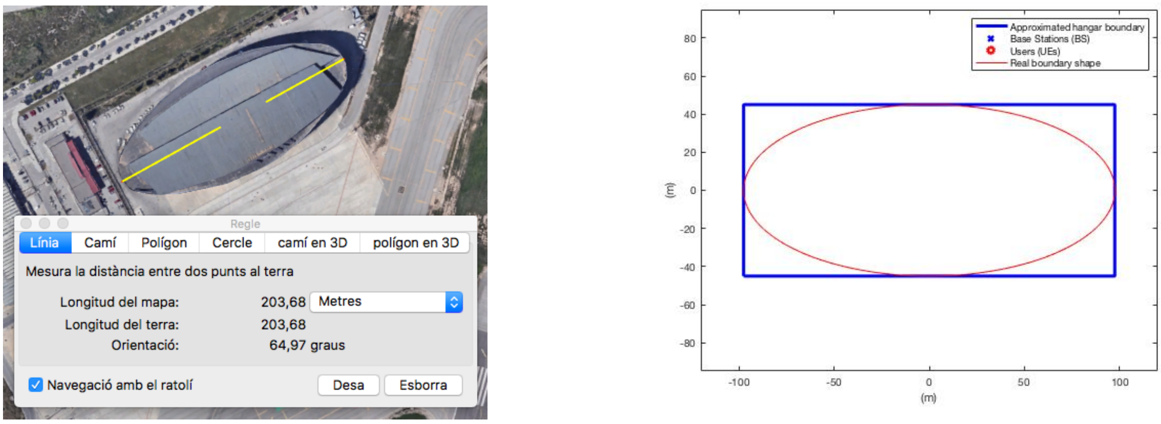
Notice that the requirements of the smart tools used in this scenario, are assumed to be similar as generic industrial smart devices. The assumptions are listed in first and second columns of table 7.1. The smart tool is considered to be a connected machine which tightens certain type of bolts detected by the AR set, which also recognises all other parts of the aircraft. In fact, similar devices mentioned in section 3.3.2., which are considered the most relevant involved in aerospace maintenance. Also, other devices must be considered, but the mentioned ones for this simulation are mobile devices. These are the ones that can only relay on wireless technology. The rest of them, can be considered in a fixed position, so can be connected using other type of communication technologies.

These devices are supposed to stay connected while they are inside the hangar building. This oval-shaped building can be approximated as a rectangular shape (see Figure 7.2(b)), in which results would not change that much. Separation distances between BSs and UEs have a similar magnitude order in a rectangular approximation.

As the simulator needs specific requirements values, values used in the simulator are selected according the magnitudes of the mentioned table 7.1. The selected values, are displayed in table 7.2 which are inputs of the simulator. Latency is selected as the one displayed in table 7.1, and throughput is selected as one order of magnitude above the minimum of the displayed throughput value.

7.2.1.2. Scenario modelling

Size references of the actual building are taken using Google Earth rule tool (see Figure 7.2(a)), which has the possibility to calculate straight distances over a map. As it is very difficult to get the plans of the hangar, this rule tool is used to get an approximation of the hangar sizes needed for the simulation.



(a) Rule tool of Google Earth

(b) Simulation 2 Scenario scheme

Figure 7.2: Simulation 2 Scenario

The measured sizes using the Google Earth Rule Tool 7.2(a) and thus considered for the simulation, are listed on table 7.3, also with BS positions taking into account that point $(0,0)$ is the hangar centre. Devices distribution is commented in section 7.2.1.3.

Table 7.3: Scenario size and BS positions.

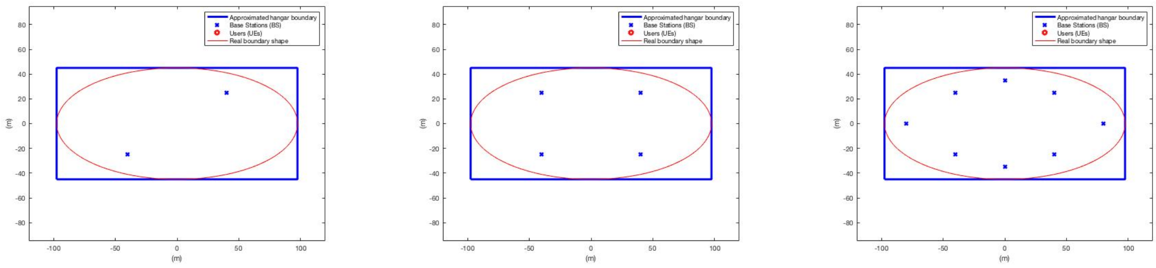
	X in metres	Y in metres
Scenario Size	195	90
BS #1 position	-40	-25
BS #2 position	40	-25
BS #3 position	40	25
BS #4 position	-40	25
BS #5 position	-80	0
BS #6 position	0	-35
BS #7 position	80	0
BS #8 position	0	35

The main different variations of the scenarios depend on the different configurations wanted to try, as this time, there are different devices on the same scenario. The main goal of this simulation, is to find the most suitable BS configuration, carrier frequency, Bandwidth that better satisfies requirements of table 7.1 for every kind of device. Also, as these devices are working over licensed spectrum, is important to use as less bandwidth as possible without scarifying performance. So as the test simulation, the most suitable configuration will be the one that uses less resources and satisfies requirements at the same time.

7.2.1.3. Overall simulation configuration

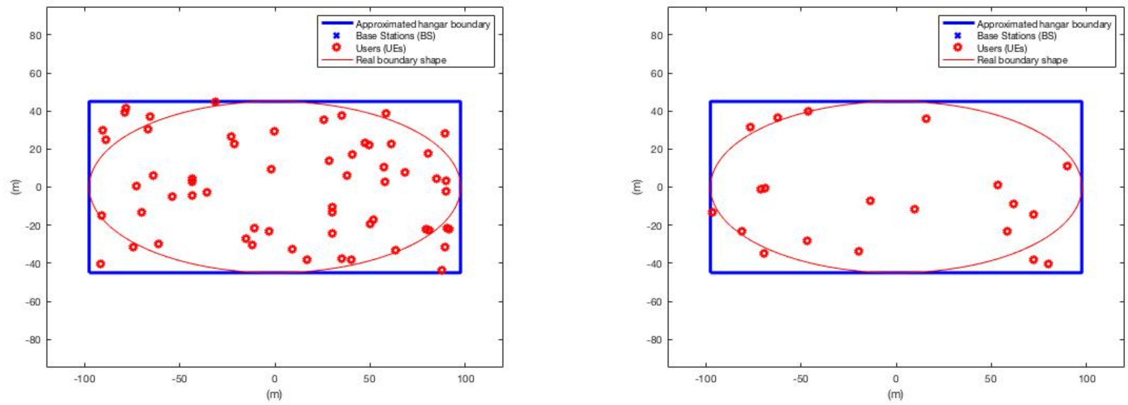
BSs configurations consists in three different configurations. The different variations are tried with 2, 4 or 8 BS. As in this simulation, the scenario is much bigger and also there is a need for connecting more devices. The three different BS layout are shown on figure 7.3 and positions on every BS on table 7.3. As performances in some variations of 4 BS are better than in 8 BS, the 2 BS configuration is kept as done in Simulation 1 to prove if 4 BS configuration is the most optimal one, or if there is a possibility to improve performances with a configuration of less Bs.

UEs are spread randomly all across the simulation scenario, the same way as in the test simulation, as mentioned in section 7.2.1.2.. See Figure 7.4. The two different carrier frequencies configuration of this simulation, are 6 GHz and 28 GHz, its all about try which of them is more suitable for depending on the type of device, so a relatively low one and one higher are chosen. Also, is proved if increasing bandwidth is worth over the performance increase, so performances between 100 MHz and 400 MHz are compared. It is important to remark that the bandwidth increase is never chosen unless it is strictly necessary to achieve the requirements.



(a) 2 BS layout (b) 4 BS layout (c) 8 BS layout

Figure 7.3: Available BS layouts



(a) Smart tools layout (b) AR glasses layout

Figure 7.4: Available BS layouts

Summarising all mentioned, all the parameters in which different variations are taken, are shown on table 7.4 with their different possible chosen values.

Table 7.4: Variations on simulation parameters

# of devices	20 AR headsets	60 Smart tools	-
# of BS	2	4	8
Carrier frequency	6 GHz	28 GHz	-
Bandwidth	100 MHz	400 MHz	-
Data traffic direction	Up-Link	Down-Link	-

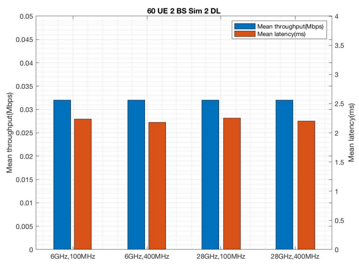
7.3. Simulation results

All the simulation results will be displayed as ordered on table 7.5. For each case, the different Fc and BW parameters are varied within the corresponding plot.

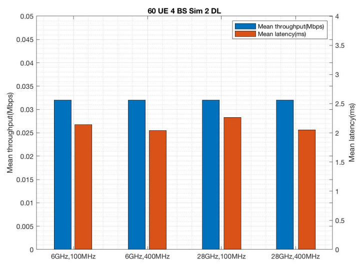
Table 7.5: Real-Based deployment simulation variations

Config. #	# of devices	# of BS	Link Direction
1	60	2	Down-link
2	60	4	Down-link
3	60	8	Down-link
4	60	2	Up-link
5	60	4	Up-link
6	60	8	Up-link
7	20	2	Down-link
8	20	4	Down-link
9	20	8	Down-link
10	20	2	Up-link
11	20	4	Up-link
12	20	8	Up-link

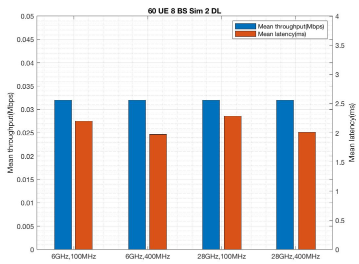
7.3.1. Results Plots



(a) 2 BS, Down-link A.10

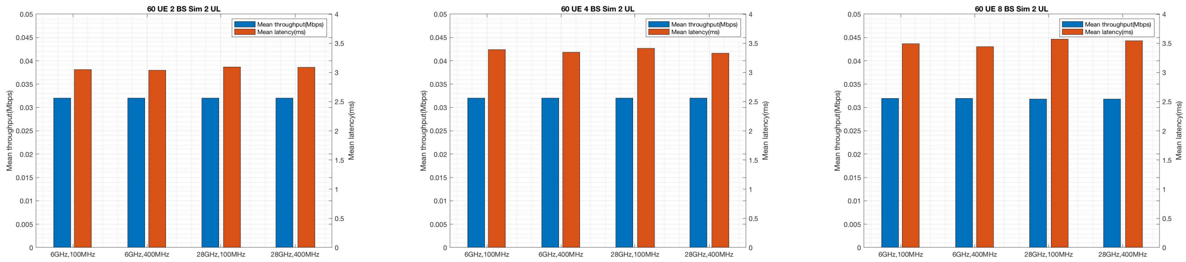


(b) 4 BS, Down-link A.11



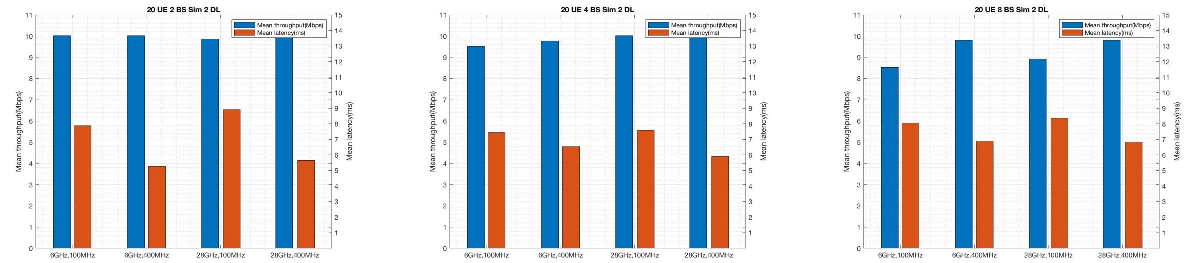
(c) 8 BS, Down-link A.12

Figure 7.5: Smart tools down-link configurations results.



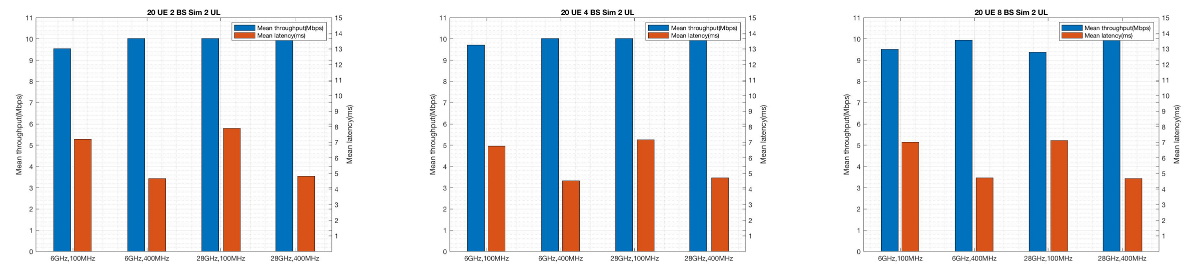
(a) 2 BS, Up-link A.13 (b) 4 BS, Up-link A.14 (c) 8 BS, Up-link A.15

Figure 7.6: Smart tools up-link configurations results.



(a) 2 BS, Down-link A.16 (b) 4 BS, Down-link A.17 (c) 8 BS, Down-link A.18

Figure 7.7: AR Headsets down-link configurations results.



(a) 2 BS, Up-link A.19 (b) 4 BS, Up-link A.20 (c) 8 BS, Up-link A.21

Figure 7.8: AR headsets up-link configurations results.

7.4. Conclusions and comments on the results

The main objective of this section, is to select the most suitable configuration that satisfies at most, the requirements of each of the deployments. For that, the following criteria have been followed:

- The most suitable configuration for **smart tools deployments**, is the one that better satisfies **latency requirements**. So first, independently of throughput, the best latency configuration is wanted, then, if found many configurations that all meet this first preferable requirement with the same latency, is selected the configuration that also gets the best throughput delivery.
- For the case of **AR headsets**, the most suitable configuration is considered to be the one that gets the best throughput performance. At first, without taking into account latency performance. In case all throughput is delivered in many configurations, latency requirement is also considered and configuration with better latency performance is selected.

In case to stay between two configurations and the better one has just slightly better first preferable requirement performance but performance on the second preferable requirement is much better, will be discussed with more detail which one is preferred.

Simulation results shown on plots of section 7.3. are explained and detailed at this section. These plots are based on numeric results of tables on section A.2. on the appendices. Results are divided into two main sections. Deployments comparison of smart tools, and deployments comparison of AR headsets. Also, these subsections are divided between traffic direction.

7.4.1. Results comments of smart tools deployments

The total transmitted throughput per user of this simulation is 0.032 Mbps for smart tools. That corresponds to 10 Kbps of transmitted data throughput plus 22.4 Kbps of data headers. As requirements of smart tools shown on table 7.1 show a determined required throughput. Is assumed that this is the required data throughput plus an extra order of magnitude, without including headers.

7.4.1.1. Down-link smart tools deployments

The first thing to be commented, is that when frequency is increased, from 6 GHz to 28 GHz, latency performances are decreased on all deployments of 2 BS, 4 BS and 8 BS. Also with both bandwidth configurations of 100 MHz and 400 MHz. Throughput performance is maintained in the mentioned configurations on its maximum value of 0.032 Mbps.

When increasing BW from 100 MHz to 400 MHz, latency performance is increased but not so much. It has to be taken into account that BW is incremented four times, but it may not worth for the amount of latency performance increment. For example, when commenting about 2 BS and 6 GHz of carrier frequency deployment A.10, when switching from 100 MHz to 400 MHz of BW, latency decreases from 2.23641 ms to 2.17851 ms. This is just a 2.59 % of latency decrease, over the initial latency value when having 100 MHz. Anyway, following the previous criteria, as latency requirement of 1 ms is not met, is preferable to select the configuration that has better latency performance.
Comparing generally latency performance between deployments of 2 BS, 4 BS and 8 BS, the

configuration that gives worst latency performance, is the one with 2 BS, followed by both configurations 4 and 8 BS. The configuration that gives best latency performance over the three mentioned deployments, is 8 BS.

Summarising all, the **configuration** that gives the **best performance** of all mentioned down-link configurations A.2.1., is **400 MHz of BW, 6 GHz of F$_c$ and 8 BS.**

7.4.1.2. Up-link smart tools deployments

As happens on down-link 7.4.1.1., when increasing Fc from 6 to 28 GHz, latency performance decreases. This time, not all deployments decrease their latency performances. Deployment of 4 BS and 400 MHz A.14 increase a little bit its latency performance, so generally, without taking into account this exceptional case, can be commented that an Fc increase from 6 GHz to 28 GHz, decreases latency performances.

As happens on down-link cases, when increasing BW from 100 MHz to 400 MHz, latency performance in all deployments increase a little bit.

When commenting about throughput performances, this time not all throughput is delivered in most of the cases on deployments of 4 BS A.14 and in all 8 BS deployments A.15. Deployments on 2 BS A.13, deliver all throughput. This is a clue to guess that configurations that deliver the best performances are the ones that have 2 BS.

It is also important to notice, that the latency requirement of 1 ms is not met in any of the cases, so the up-link requirement should be for example, 4 ms, due to even the worst-performing case does not reach 4 ms. A requirement of 4 ms might be considered viable, since 4ms is in the order of 1ms of the initial requirements even if in table 7.2 is considered to not to take into account latencies of more than 1 ms.

As mentioned before, for choosing the configuration that better satisfies this deployment, is taken into account minimum latency configurations that may offer better throughput performances. Casually, configurations that offer the best latency performance, also offer the best throughput performance. So, no doubt of it, 2 BS configurations are the best ones for this deployment. The configuration of these, and the preferable one for this deployment, is **400 MHz of BW, 6 GHz of F$_c$ and 2 BS**.

7.4.2. Results comments of AR headsets deployments

The total transmitted throughput per user of this simulation is 10.0224 Mbps for AR headsets. This amount corresponds to 10 Mbps of the assumed required throughput 7.2.1.1. of 10^6 bps, the minimum throughput requirements plus one extra order of magnitude, and 0.0224 Mbps of headers.

7.4.2.1. Down-link AR headsets deployments

At this case, unlike the previous one, there are more combinations that when increasing frequency from 6 GHz to 28 GHz, performances are improved. The case of 4 BS A.17 for example, with 100 MHz of BW, throughput performance is increased to deliver the whole amount of throughput in both 100 MHz and 400 MHz of BW cases. In the 400 MHz case, latency performance is improved, not like the 100 MHz case, which latency performance is reduced. Something similar happens on the 8 BS case. Throughput delivery is better on both 100 MHz and 400 MHz cases with 28 GHz of F_c, even if in the best case, not all throughput is delivered.

On the 400 MHz case, latency performance is increased a little bit, but the change is almost imperceptible. Anyway, increasing BW helps to increase all performances indicators in all cases, so in order to select the best configuration of this deployment, 400 MHz of BW is chosen.
As there are a few configurations that deliver the best throughput performance, it is much easier to compare them. This is because configurations that deliver all throughput have very different latency performances. An example of this could be configuration of 4 BS. See table A.17. Both configurations of 28 GHz of F_c deliver all throughput, but the configuration of 400 MHz of BW, is more than a second and a half faster than the same configuration that uses 100 MHz of BW.
For selecting the most suitable configuration though, none of these example configuration are the best one. The configuration of **400 MHz of BW, 6 GHz of F_c and 2 BS** of deployment A.16 has the best performances.

7.4.2.2. Up-link VR headsets deployments

When increasing frequency from 6 GHz to 28 GHz, throughput performances are increased in all cases except for 8 BS and 100 MHz of deployment A.21. Throughput delivery is generally better with 28 GHz, so for combining simultaneously up-link and down-link performances on VR headsets, can be associated 6 GHz for down-link, and 28 GHz for up-link, which reduces the possibility of collision between opposite traffic directions.

This statement would not be acceptable if latency performances are considered to be more relevant than throughput delivery. Generally, the 6 GHz Fc cases offer better performances, although as mentioned above, it is preferred to have full delivery of throughput over better latency.

The deployment that lets to have **the best performances, is 4 BS, 400 MHz of BW, and 6 GHz, of Fc.** If taken into account that different transmission directions can be associated to different frequencies, latency performance is slightly reduced, but still can be found good combinations. If is this the case, the most suitable combination is the same 4 BS and 400 MHz of BW, but changing 6 GHz of Fc by 28 GHz. This other possible combination must be evaluated to see if having up-link and down-link traffic separated, among other possible advantages that may have, is worth over having better latency performance. Latency performance is not as relevant as throughput performance, so if, for other reasons, the combination that allows better throughput and latency performances is discarded, this is a good possible combination.

7.4.3. Optimal deployments configurations

Summarising the mentioned simulations results of section 7.4., the configurations that give the best performances are shown in the following table.

Table 7.6: Optimal configurations table.

Deployment	**# of BS**	**BW**	**F$_c$**	**Throughput**	**Latency**
Smart Tool DL	8	400 MHz	6 GHz	0.03200 Mbps	1.97035 ms
Smart Tool UL	2	400 MHz	6 GHz	0.03200 Mbps	3.03693 ms
AR Headset DL	2	400 MHz	6 GHz	10.02240 Mbps	5.26826 ms
AR Headset UL	4	400 MHz	6 GHz	10.02240 Mbps	4.52869 ms

Taking into account the optimal configurations, the following table shows what Latency and data-rate requirements are met.

Table 7.7: Real-Based Simulation compliance with the requirements.

Deployment	**Thr. Req.**	**Lat. Req.**	**Thr. met**	**Lat. met**
Smart Tool DL	0.03200 Mbps	1 ms	✓	✗
Smart Tool UL	0.03200 Mbps	1 ms	✓	✗
AR Headset DL	10.02240 Mbps	10 ms	✓	✓
AR Headset UL	10.02240 Mbps	10 ms	✓	✓

At section 7.4., configurations that give the best performance for each deployment were found.

7.5. NR deployment vs. LTE deployment

Among other objectives, this project aims to also compare different radio technologies to I4.0 possible scenarios. It is relevant to compare the mentioned communication standards technologies, both, to see which of them is the most suitable for the simulated application, and to see if current and new-coming standards fit within the requirements that the I4.0 demands. As this case, the size of the scenario is in the order of hundreds of meters, so Wi-Fi IEEE 802.11 standard would not be suitable at all. The technology which is the closest one to NR in terms of range and flexibility, is its predecessor, the LTE-Advanced standard. Its range is suitable for the implementation of the simulation scenario, and also, can be accurately simulated using the NR-adapted scenarios used in previous simulations.

NR examples can be adapted to simulate LTE-Advanced specifications. A good approximation to these specs can be done by fitting the following parameters to the simulated scenario:

Table 7.8: LTE-Advanced specific simulation parameters

F_c	BW	Numerology
6 GHz	20 MHz	0

The comparison is done by picking the best configurations obtained by analysing the simulation results, the ones mentioned in table 7.6 and simulating them using LTE-Advanced specifications of table 7.8. Also, in order to be able to compare up-link and down-link configurations in a more fair way, the same number of 4 BS on each deployment is implemented in all 4 deployments.

7.5.1. LTE-Advanced scenarios, conclusions and results

Results are shown as a comparison between performances of the best combination of every deployment in the NR cases, next to the results obtained with LTE-Advanced. Table 7.9 show the mentioned results:

Table 7.9: NR Deployment VS LTE-Advanced performances

	NR		NR Performances		LTE-Adv. Performances	
Deployment	F_c (**GHz**)	**BW** (**MHz**)	**Throughput** (**Mbps**)	**Latency** (**ms**)	**Throughput** (**Mbps**)	**Latency** (**ms**)
S. Tool DL	6	400	0.03200	2.04178	0.03194	5.88281
S. Tool UL	6	400	0.03199	3.34642	0.016338	10.27610
AR HS DL	6	400	9.78019	6.53813	4.47667	179.41300
AR HS UL	6	400	10.02240	4.52869	2.01840	244.86900
# of BS: 4						

Observing the results of the table, the following conclusions about this comparison are taken.

- Latency can not be considered satisfied in any case without exception. For example, it is obvious that if 1 ms latency requirement it is not satisfied even with NR, can not be satisfied with LTE-Advanced. Latency on both up-link and down-link smart tools scenarios, is around 3 times slower than NR latency performances. In case of smart tools down-link, the major contributor for this delay increase, is slot size. As numerology used in LTE is 0 instead of the used in NR, which is 2, the slot time is bigger and thus, transmission and packet processing take more time due to they virtually depend on slot size. In the smart tools Up-link case, can be seen that latency is around 10ms. As RLC Timer is set to be 10ms, can be assumed it is the main cause of the latency increase. As there are packet losses, the RLC layer "waits" (Increasing latency) in order to ensure all packets at the same period of time are received.

  In the AR Headsets cases, things get even worse, with around 30 times more latency in the down-link and around 50 times slower in the up-link case. When load overcomes capacity, the amount of transmission time increases, because of the bottle neck caused by it, and the amount of data to be transmitted takes more time increasing latency.

- Throughput delivery is not satisfied in any case, except for the smart tools down-link case, which can be considered an exception. Almost all throughput is delivered. Commenting about the rest of the deployments, throughput performance is reduced critically. For example, comparing both NR and LTE-Advanced on smart tools up-link and AR headsets down-link deployments, LTE-Advanced can only deliver around half of throughput delivered with NR. This situation gets even worse with the deployment of AR Headsets up-link, in which the LTE-Advanced throughput delivered is around 20% the NR delivered throughput. As capacity of the system is reduced, on certain transmissions in which buffers reach their limits, information that can not be stored on them is lost, causing these bad throughput delivery results.

CHAPTER 8. CONCLUSIONS

a What is the purpose of this project simulation?

The aim of this project, is about doing a first evaluation on the implementation of a last generation wireless communications standard in order to create a wireless network of multiple smart devices. Actually, the mentioned network implementation aimed to find an implementation of a connected smart-device network in an aeronautical field. A suitable possible implementation of some kind of Industry 4.0 devices and technologies in the aeronautical field is found to be the **aerospace maintenance**, by means of **smart and connected tools** to assist the workers.

In order to evaluate how implementable is a wireless network of a smart maintenance tools using a last-generation wireless communication standard, a simulated maintenance scenario has been created based on approximated data about the usage, needs, and assuming that these devices have similar known requirements than other I4.0 smart devices. The requirements of the selected devices are very different, because the selected tools offer different services, and so, have different requirements. A suitable wireless protocol which offers connectivity to a bunch of devices with a wide range of uses and requirements, is **5G-NR**. If the devices services need to be provided according to their requirements, is it possible to successfully connect them by the implementation of NR?

b What the ns-3 simulator is chosen for?

In order to have an idea of whether it is possible to implement the mentioned wireless protocol in the proposed scenario or not, it is necessary to have performance data of the network to be compared with requirements of devices connected to the mentioned network. That is why communications of the smart devices in the network have been simulated, and so, all of the layers of the protocol stack of the 5G specification. It is necessary to know if specifications of the NR standard can offer the necessary performance to satisfy the requirements of this smart industry network.

For that, the ns-3 network simulator has been chosen due to multiple reasons. The first one is that there is not enough information provided by the experiments performed by 3GPP, and thus, these experiments can not be replicable. The second one is that there is no public simulator that simulates all protocol layers except for the ns-3, and the others simulations assume certain behaviours of specific protocol layers that may introduce errors in the results of the simulation. Also, the ns-3 simulator has no restriction in the creation of any kind of wireless network layout, and can simulate a good range of wireless protocols.

Using ns-3, the commented scenario has been successfully simulated and the wanted performance data have been successfully obtained.

c How was the ns-3 set up for the purpose?

The ns-3 has many simulation examples in which one of them has been adapted to the network scenario of interest to simulate. Also, a test scenario has been simulated in order to test how ns-3 works and familiarize with it, to understand performance data obtained, and to test what parameters can be changed in order to adapt the examples.

The **Test Scenario** consists on a smart-factory sized 40 m by 40 m, in which the number of connected devices can be either 10 or 20 spread randomly across the building. Every device needs to have an available data-rate and latency depending on their use mode: 0.8 Mbps or 16 Mbps, and 3 ms or 10 ms, respectively, in both up-link and down-link traffic directions. If a configuration of 2 or 4 base stations are available, a bandwidth of 100 MHz to 400 MHz, and a band of 6 or 28 Ghz can be used, which combination satisfies the best the requirements of the devices? Is it possible to satisfy the devices requirements?

The results of the simulation show that all the initial requirements have been successfully satisfied by selecting the best combination of number of BS, F_c and BW. Also, the two main ideas obtained by interpreting the results of all combinations are the following:

- The first thing to comment, is that increasing the bandwidth from 100 MHz to 400 MHz, in low-traffic configurations, helps to reduce latency, due to resource blocks spread within the bigger amount of BW. Because of this, the time needed to transmit the same amount of data is shorter. Also, as the mount of time is reduced, buffers are less congested, so it is also a sign that the amount of transmitted throughput is greater.
- The second thing to comment is that, in case of having many deployed BS, if F_C is increased, network performances improve. This is because if propagation of electromagnetic waves is worse, less interferences happen and less packets have to be retransmitted. With the same conditions, if reducing the amount of BW, also, performances are improved because of SINR increase. This is because, if BW is reduced, and the total transmitted power is kept the same, the transmitted power per resource block increases.

d How was the Real-Based Scenario set?

The simulation modeling of the maintenance scenario is done according to the following assumptions. The simulated maintenance hangar consists on a plant sized 90 m by 195 m. This hangar has two types of connected smart tools. The first type, consists of 60 robotic smart tools that require a data rate of 10 Kbps and less than 1 ms of latency both, in up-link transit direction and in down-link. The second type consists on 20 Augmented Reality (AR) headsets that require a data rate of 10 Mbps and a latency of 10 ms. Also, in both up-link and down-link directions.

In order to satisfy the mentioned device requirements, the maintenance hangar has available 3 types of BS configurations: 2, 4 and 8 BS. It has also available two bands, 6 GHz and 28 GHz, and 100 MHz or 400 MHz of BW.

Interpreting the simulation results, the following conclusions have been reached. When commenting about performance of smart robotic tools, in down-link transit direction, the BW configuration that gives the best performance is 400 MHz. Due to the size of this deployment, using 6 GHz of band gives better performance. In case of using 28 GHz, performances decrease due to signal attenuation at high frequencies. Using 8 BS configuration, services provided to devices give better performances than using the 4 BS or 2 BS configurations. This is because as being a low-data-rate configuration, the amount of transmitting time is lower enough, that interferences do not affect the mentioned configuration at all. Also, as BW that gives better performances is 400 MHz, transmitted specific power is lower, which also helps to reduce interferences. In case of this same configuration but with up-link traffic direction, smart devices transmit to too many BS, so the amount of interferences does not make the 8 BS configuration the most suitable.

When commenting about AR headsets, using the 8 BS configuration, generates too many interferences due to collisions caused by the high data-rate. This causes that many packets have to be re transmitted, and so, buffers overload causing that not all throughput is able to be delivered. When commenting about up-link traffic direction, 4 BS is the best BS configurations. If performances are compared with the other configurations, with 8 BS, traffic received by each BS would be reduced, but the amount of interferences does not compensate that. By the other hand, with the 2 BS configuration, interferences are reduced but traffic saturation in every BS reduces performances too much, and it does not compensate less interferences.

When commenting about requirements achievement, it can be stated that **all requirements are met, except for the latency requirements of the smart robotic tools**. As the latency requirements of these tools are too strict (1 ms end-to-end), NR may not be able to satisfy them. Another reason for this statement, is that numerology, which can also reduce latency if increased, can not be increased because higher numerologies are not available in sub 6 GHz bands. By increasing the frequency, higher numerologies can be adopted, but when frequency increases, signal attenuation decreases performances. The numerology increment does not compensate the loss of performances due to higher frequencies.

In the case of the AR devices, the latency performance is good enough that more AR devices can be introduced in the network, without compromising requirements achievement.

Given these results, it has also been interesting to compare network performances with another standard. That is why, the same deployment has been implemented using specifications of the predecessor of 5G-NR, i.e., 4G-LTE-Advanced. The performance of this network implementation was much worse that would be impossible to implement it and meet, at least, requirements in one configuration.

BIBLIOGRAPHY

[1] V. Fontaine. Implementation of the industry 4.0 in the aeronautical industry in belgium, 2016-2017. xiii, 1, 2, 11, 12

[2] Matthew S Gast. *802.11 ac: a survival guide: Wi-Fi at gigabit and beyond*. " O'Reilly Media, Inc.", 2013. xiii, xv, 25

[3] Natale Patriciello, Sandra Lagen, Biljana Bojovic, and Lorenza Giupponi. An E2E simulator for 5G NR networks. *Simulation Modelling Practice and Theory (SIMPAT), vol.96, 101933*, 96:101933, Nov. 2019. xiii, 27, 28, 33

[4] Natale Patriciello, Sandra Lagen, Biljana Bojovic, and Lorenza Giupponi. *NR Module*, Feb., 2019. `https://5g-lena.cttc.es`. xiii, 34

[5] La Vanguardia. *Iberia inaugura en Barcelona un gran hangar de mantenimiento*, 2010 (accessed Sept. 3, 2019). `https://www.lavanguardia.com/economia/20101018/54025904371/iberia-inaugura-en-barcelona-un-gran-hangar-de-mantenimiento.html`. xiii, 49, 50

[6] Sandra Lagen and Lorenza Giupponi. From NR to 6G in Unlicensed Spectrum: the RAT for Wireless Private Networks in Industry 4.0. *6G Summit*, Mar. 2019. xv, 9, 13, 50

[7] Thomas Nitsche, Carlos Cordeiro, Adriana B Flores, Edward W Knightly, Eldad Perahia, and Joerg Widmer. IEEE 802.11 ad: directional 60 GHz communication for multi-Gigabit-per-second Wi-Fi. *IEEE Communications Magazine*, 52(12):132–141, 2014. xv, 25

[8] Der-Jiunn Deng, Ying-Pei Lin, Xun Yang, Jun Zhu, Yun-Bo Li, Jun Luo, and Kwang-Cheng Chen. IEEE 802.11 ax: highly efficient WLANs for intelligent information infrastructure. *IEEE Communications Magazine*, 55(12):52–59, 2017. xv, 25

[9] Yasaman Ghasempour, Claudio RCM da Silva, Carlos Cordeiro, and Edward W Knightly. IEEE 802.11 ay: Next-generation 60 GHz communication for 100 Gb/s Wi-Fi. *IEEE Communications Magazine*, 55(12):186–192, 2017. xv, 25

[10] Raphael Cohen-Almagor. Internet history. In *Moral, ethical, and social dilemmas in the age of technology: Theories and practice*, pages 19–39. IGI Global, 2013. 3

[11] Bob O'hara and Al Petrick. *IEEE 802.11 handbook: a designer's companion*. IEEE Standards Association, 2005. 4

[12] Priya Suresh, J Vijay Daniel, Velusamy Parthasarathy, and RH Aswathy. A state of the art review on the internet of things (iot) history, technology and fields of deployment. In *2014 International conference on science engineering and management research (ICSEMR)*, pages 1–8. IEEE, 2014. 4

[13] Stefan Parkvall, Erik Dahlman, Anders Furuskar, and Mattias Frenne. Nr: The new 5g radio access technology. *IEEE Communications Standards Magazine*, 1(4):24–30, 2017. 5, 19, 20

[14] Baotong Chen, Jiafu Wan, Lei Shu, Peng Li, Mithun Mukherjee, and Boxing Yin. Smart factory of industry 4.0: Key technologies, application case, and challenges. *IEEE Access*, 6:6505–6519, 2017. 11, 12

[15] B. Pedersen O.K. Lonkeu PwC V.Bonneau B. Copigneaux, IDATE; L. Probst. *Industry 4.0 in Aeronautics: IoT applications*. Digital Transformation Monitor, 2017. 16

[16] Mehmet Yavuz. How will 5g transform industrial iot. *Qualcomm Technologies, Inc.: San Diego, CA, USA*, 2018. 19

[17] 5G ACIA. *5G for Connected Industries and Automation*. "ZVEI - German Electrical and Electronic Manufacturer's Assosiation", 2018. 19, 22

[18] Iberia. *Estrenamos hangar de mantenimiento en Barcelona*, 2010 (accessed Sept. 3, 2019). http://megustavolar.iberia.com/2010/11/estrenamos-hangar-de-mantenimiento-en-barcelona/. 50

APPENDICES

APPENDIX A. RESULTS TABLES

A.1. Test Simulation results tables

A.1.1. Down-link tables

Table A.1: 0.8 Mbps, 10 UEs, down-link

BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)
2	100	6	0.8224	2.32018
2	100	28	0.8224	2.318
2	400	6	0.8224	2.12857
2	400	28	0.8224	2.13033
4	100	6	0.8224	2.53515
4	100	28	0.8224	2.24756
4	400	6	0.8224	2.27312
4	400	28	0.8224	2.02625

Table A.2: 0.8 Mbps, 20 UEs, down-link

BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)
2	100	6	0.8224	3.09586
2	100	28	0.8224	3.09586
2	400	6	0.8224	2.02318
2	400	28	0.8224	1.9988
4	100	6	0.8224	2.75848
4	100	28	0.8224	2.68077
4	400	6	0.8224	2.26033
4	400	28	0.8224	2.06793

Table A.3: 16 Mbps, 10 UEs, down-link

BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)
2	100	6	15.9957	7.60387
2	100	28	15.9957	7.95655
2	400	6	16.0224	7.80717
2	400	28	16.0224	8.0083
4	100	6	14.4202	6.37681
4	100	28	13.8327	8.60714
4	400	6	15.0076	9.04554
4	400	28	15.969	8.63974

Table A.4: 16 Mbps, 20 UEs, down-link

BSs	**BW (MHz)**	**f_c (GHz)**	**Throughput (Mbps)**	**Latency (ms)**
2	100	6	15.9423	9.41495
2	100	28	15.8488	10.7194
2	400	6	16.0224	6.50742
2	400	28	16.0224	6.59823
4	100	6	13.7259	32.3868
4	100	28	14.3935	13.3414
4	400	6	14.794	8.81826
4	400	28	15.8889	8.09946

A.1.2. Up-link tables

Table A.5: 0.8 Mbps, 10 UEs, up-link

BSs	**BW (MHz)**	**f_c (GHz)**	**Throughput (Mbps)**	**Latency (ms)**
2	100	6	0.8224	4.17172
2	100	28	0.8224	4.31612
2	400	6	0.8224	2.81134
2	400	28	0.8224	3.1032
4	100	6	0.8224	4.35493
4	100	28	0.80184	4.83971
4	400	6	0.8224	3.47435
4	400	28	0.819659	3.65634

Table A.6: 0.8 Mbps, 20 UEs, up-link

BSs	**BW (MHz)**	**f_c (GHz)**	**Throughput (Mbps)**	**Latency (ms)**
2	100	6	0.8224	4.36151
2	100	28	0.8224	4.32861
2	400	6	0.8224	3.09583
2	400	28	0.8224	3.10615
4	100	6	0.8224	4.27009
4	100	28	0.816232	4.50513
4	400	6	0.8224	3.07766
4	400	28	0.821029	3.3523

Table A.7: 16 Mbps, 10 UEs, up-link

BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)
2	100	6	16.0224	6.89481
2	100	28	15.515	8.19572
2	400	6	16.0224	4.60452
2	400	28	16.0224	4.85473
4	100	6	15.5951	6.55078
4	100	28	14.5804	8.54893
4	400	6	16.0224	4.53004
4	400	28	15.4616	5.3852

Table A.8: 16 Mbps, 20 UEs, up-link

BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)
2	100	6	14.7006	8.84397
2	100	28	14.6471	9.37372
2	400	6	16.009	5.42205
2	400	28	15.969	5.60708
4	100	6	15.9957	7.83706
4	100	28	13.8193	8.79726
4	400	6	16.0224	4.90191
4	400	28	15.7287	5.41714

A.1.3. Optimal configurations.

Table A.9: Optimal deployments table

	Optimal Depl.			Optimal Performances		Targets	
Config.	BSs	BW (MHz)	f_c (GHz)	Throughput (Mbps)	Latency (ms)	% Thr.	Latency
# 1	4	400	28	0.8224	2.02625	100	3ms
# 2	2	400	28	0.8224	1.9988	100	3ms
# 3	2	400	6	16.0224	7.80717	100	10ms
# 4	2	400	6	16.0224	6.50742	100	10ms
# 5	2	400	6	0.8224	2.81134	100	3ms
# 6	4	400	6	0.8224	3.07766	100	3ms
# 7	4	400	6	16.0224	4.53004	100	10ms
# 8	4	400	6	16.0224	4.90191	100	10ms

A.2. Real-Based Simulation results tables

In what follows, for each cell of the tables, which corresponds to a specific configuration of carrier frequency (F_c) and channel bandwidth (BW), the attained throughput (in Mbps) and the end-to-end latency (in ms) are shown.

A.2.1. Smart tools tables

Table A.10: 2 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03200 Mbps 2.23641 ms	0.03200 Mbps 2.25394 ms
BW **400 MHz**	0.03200 Mbps 2.17851 ms	0.03200 Mbps 2.20457 ms

Table A.11: 4 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03200 Mbps 2.14288 ms	0.03200 Mbps 2.26608 ms
BW **400 MHz**	0.03200 Mbps 2.04178 ms	0.03200 Mbps 2.04983 ms

Table A.12: 8 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03200 Mbps 2.20000 ms	0.03200 Mbps 2.22866 ms
BW **400 MHz**	0.03200 Mbps 1.97035 ms	0.03200 Mbps 2.00954 ms

Table A.13: 2 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03200 Mbps 3.04998 ms	0.03200 Mbps 3.09422 ms
BW **400 MHz**	0.03200 Mbps 3.03693 ms	0.03200 Mbps 3.09097 ms

Table A.14: 4 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03199 Mbps 3.39163 ms	0.03198 Mbps 3.41585 ms
BW **400 MHz**	0.03119 Mbps 3.34642 ms	0.03200 Mbps 3.33030 ms

Table A.15: 8 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	0.03191 Mbps 3.48971 ms	0.03178 Mbps 3.57210 ms
BW **400 MHz**	0.03193 Mbps 3.44295 ms	0.03178 Mbps 3.54405 ms

A.2.2. AR headsets tables

Table A.16: 2 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	10.01400 Mbps 7.88477 ms	9.87206 Mbps 8.90277 ms
BW **400 MHz**	10.02240 Mbps 5.26826 ms	9.96394 Mbps 5.63873 ms

Table A.17: 4 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	9.51293 Mbps 7.44127 ms	10.0224 Mbps 7.57638 ms
BW **400 MHz**	9.78019 Mbps 6.53813 ms	10.0224 Mbps 5.90303 ms

Table A.18: 8 BS Down-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	8.51904 Mbps 8.04580 ms	8.92829 Mbps 8.35427 ms
BW **400 MHz**	9.80525 Mbps 6.89191 ms	9.79690 Mbps 6.83562 ms

Table A.19: 2 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	9.53798 Mbps 7.21573 ms	10.02240 Mbps 7.88931 ms
BW **400 MHz**	10.02240 Mbps 4.68048 ms	10.02240 Mbps 4.82592 ms

Table A.20: 4 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	9.71338 Mbps 6.75818 ms	10.02240 Mbps 7.16266 ms
BW **400 MHz**	10.02240 Mbps 4.52869 ms	10.02240 Mbps 4.72679 ms

Table A.21: 8 BS Up-link

	F_c **6 GHz**	F_c **28 GHz**
BW **100 MHz**	9.50458 Mbps 7.01897 ms	9.37094 Mbps 7.11226 ms
BW **400 MHz**	9.93888 Mbps 4.72509 ms	10.00570 Mbps 4.67687 ms

APPENDIX B. MATLAB CODES

B.1. MATLAB Codes

Codes mentioned in section 6.2.2.3. used for plotting results and describing scenarios are listed in this section. It is only worth to have a look to two of them, due to the rest of them are based on these codes with some modifications.

B.1.1. Layout figures generation

Figures that describe network layout, are generated with the following MATLAB code. Each plotted combination is called 'Experiment'. Experiments from 1 to 4, describe all factory layout, From 2 BS and 10 Users, to 4 BS to 20 users. The rest of combinations show deployments of only BS, or only UEs.

Listing B.1: Code for generating layout figures.

```
%------------------------SCRIPT 2 FOR GENERATING LAYOUT FIGURES----------
close all; clear all;
%-------------Parameters--------------

Exp = -1;%Experiment number
%<<Experiment>> -3=4EB, -2=2EB, -1=20UES, 0=10UES, 1=Exp1, 2=Exp2, 3=Exp3, 4=Exp4
Guar = false;%Automatic save function
Close = false;%Automatic figure closure

%----------------------------------

%-------------Coordinates-------------

x2 = [0 40 40 0 0]; %Factory Boundary coordinates
y2 = [0 0 40 40 0];

if(Exp == -3)
    x1 = [10.5 10.5 30.5 30.5]; %4 Base station layout configuration
    y1 = [30.5 10.5 30.5 10.5];
    Title = 'Base stations Layout: 4 base stations';
end

if(Exp == -2)
    x1 = [10.5 30.5]; %2 Base Stations configuration
```

```
    y1 = [30.5 10.5];
    Title = 'Base stations Layout: 2 base stations';
end

if(Exp == -1) %Factory layout with 20 users.
    %x1 = [];
    %y1 = [];
    Title = ('Factory Layout 40mx40m with 20 users.');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40];
end


if(Exp == 0) %Factory layout with 10 users.
    %x1 = [];
    %y1 = [];
    Title = ('Factory Layout 40mx40m');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40];
end

if(Exp == 1)
    x1 = [10.5 30.5]; %Base stations coordinates
    y1 = [30.5 10.5];
    Title = ('2 Base Stations, 10 users Factory Layout');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40];
end

if(Exp == 2)
    x1 = [10.5 10.5 30.5 30.5]; %Base stations coordinates
    y1 = [30.5 10.5 30.5 10.5];
    Title = ('4 Base Stations, 10 users Factory Layout');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand
```

```
*40 rand*40 rand*40 rand*40];
end

if(Exp == 3)
    x1 = [10.5 30.5]; %Base stations coordinates
    y1 = [30.5 10.5];
    Title = ('2 Base Stations, 20 users Factory Layout');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40];

end

if(Exp == 4)
    x1 = [10.5 10.5 30.5 30.5]; %Base stations coordinates
    y1 = [30.5 10.5 30.5 10.5];
    Title = ('4 Base Stations, 20 users Factory Layout');
    x3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40]; %Users coordinates
    y3 = [rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40 rand*40];
end


%-------------- Plots ---------------------------

plot(x2,y2,'b','LineWidth',3) %Factory boundary plot
xlim([-5 45]) %Boundary limits
ylim([-5 45])
hold on;

if(Exp <= -2)
    scatter(x1,y1,'b','x','LineWidth',3)%Base stations plot
    legend('Factory boundries','Base stations'); %Plot without base stations (Requ.)
elseif(Exp >= -1 && Exp <= 0)
    scatter(x3,y3,'r','o','Linewidth',3)%Users plot
    legend('Factory boundries','Users')
```

```
elseif(Exp >= 1)
    scatter(x1,y1,'b','x','LineWidth',3)%Base stations plot
    hold on;
    scatter(x3,y3,'r','o','Linewidth',3)%Users plot
    legend('Factory boundries','Base stations','Users')
end
    title(Title)
    set(gca,'xtick',[]) %Eliminate axis coordinates
    set(gca,'ytick',[])
hold off;

%
   %---------------------------------------------------------------------------------
```

B.1.2. Results plots

Results from simulations are used in this code to generate their plots. Specifically, this one plots all combinations of users and base stations of both 0.8 Mbps and 16 Mbps. On the code, each mentioned combination is called 'experiment number'.

Listing B.2: Code for generating simulation results plots.

```
%------------------------SCRIPT FOR PLOTTING SIMULATION RESULTS
   ----------
clear all;
close all;
%-------------Parameters--------------

Exp = 8;%Experiment Number
Guar = false;%Save
Close = false;%Automatic figure close

%-------------Data-------------------
%-------------0.8MBPS-----------------
if(Exp == 1)
    Title = '10 Users 2 Base Stations';
    Title1 = '10Users2BaseStations';
%     T = [0.819659 0.8224 0.819659 0.8224];
%     L = [4.27262 2.27024 4.35476 2.27202];
    T = [0.8224 0.8224 0.8224 0.8224];
    L = [2.32018 2.12857 2.318 2.13033];

end

```

```
if(Exp == 2)
    Title = '10 Users 4 Base Stations';
    Title1 = '10Users4BaseStations';
%     T = [0.8224 0.8224 0.8224 0.8224];
%     L = [2.71137 2.40214 2.37172 2.15544];
    T = [0.8224 0.8224 0.8224 0.8224];
    L = [2.53515 2.27312 2.24756 2.02625];
end

if(Exp == 3)
    Title = '20 Users 2 Base Stations';
    Title1 = '20Users2BaseStations';
%     T = [0.811435 0.8224 0.811435 0.8224];
%     L = [10.0552 2.15977 10.0905 2.13631];
    T = [0.8224 0.8224 0.8224 0.8224];
    L = [3.09586 2.02318 3.09586 1.99881];
end

if(Exp == 4)
    Title = '20 Users 4 Base Stations';
    Title1= '20Users4BaseStations';
%     T = [0.816917 0.8224 0.817603 0.8224];
%     L = [6.60033 2.39224 6.01237 2.19955];
    T = [0.8224 0.8224 0.8224 0.8224];
    L = [2.75848 2.26033 2.68077 2.06793];
end

%--16MBPS--

% if(Exp == 5)
%     Title = '10 Users 2 Base Stations 16MBPS';
%     T = [15.949 15.8312 15.949 15.8312];
%     L = [6.24643 10.7583 6.45368 10.9566];
% end

if(Exp == 5)
    Title = '10 Users 2 Base Stations 16MBPS';
    Title1= '10Users2BaseStations16MBPS';
    T = [15.9957 16.0224 15.9957 16.0224];
    L = [7.60387 7.80717 7.95655 8.00833];
end

% if(Exp == 6)
%     Title = '10 Users 4 Base Stations 16MBPS';
%     T = [15.949 15.0284 15.9383 15.6385];
```

```
%      L = [7.03776 26.4369 6.98634 17.2662];
% end

if(Exp == 6)
    Title = '10 Users 4 Base Stations 16MBPS';
    Title1= '10Users4BaseStations16MBPS';
    T = [14.4202 15.0076 13.8327 15.969];
    L = [6.37681 9.04554 8.60714 8.63974];
end


% if(Exp == 7)
%      Title = '20 Users 2 Base Stations 16MBPS';
%      T = [15.949 15.9329 15.9061 15.9383];
%      L = [6.02823 6.71698 7.783 6.5728];
% end

if(Exp == 7)
    Title = '20 Users 2 Base Stations 16MBPS';
    Title1 = '20Users2BaseStations16MBPS';
    T = [15.9423 16.0224 15.8488 16.0224];
    L = [9.41495 6.50742 10.7194 6.59823];
end


% if(Exp == 8)
%      Title = '20 Users 4 Base Stations 16MBPS';
%      T = [13.9634 12.9781 14.7234 15.8473];
%      L = [43.2567 41.2076 27.1267 10.3398];
% end

if(Exp == 8)
    Title = '20 Users 4 Base Stations 16MBPS';
    Title1 = '20Users4BaseStations16MBPS';
    T = [13.7259 14.794 14.3935 15.8889];
    L = [32.3868 8.81826 13.3414 8.09946];
end

%-------------------------------------
%'6GHz,100MHz';'6GHz,400MHz'; '28GHz,100MHz'; '28GHz,400MHz'

l = length(T);
%--------Max values-----
MaxT = max(T);
MaxD = max(L);
```

```
MaxAb = max([MaxT MaxD]);
%---------------------------

y = [T',L']; %Data matrix
%---------------------Plot function-------------------
  hAx1=plotyy(1:l,[y(:,1) nan(l,1)],1:l,[nan(l,1) y(:,2)],@bar,@bar);
%-----------------------------------------------------------------


%---------------------Format Plot----------------------------------
  xlim([0.5 l+0.5])%X axis limit

 if(Exp <= 4)
    ylim(hAx1(1),[0 1])%Y axis limit, Throughput
    ylim(hAx1(2),[0 12])%Y axis limit, Latency
    yticks([0 0.2 0.4 0.6 0.8 1])
    yticks(hAx1(2),[1 2 3 4 5 6 7 8 9 10 11 12])
 end

 if(Exp > 4)
    ylim(hAx1(1),[0 18])%Y axis limit, Throughput
    ylim(hAx1(2),[0 45])%Y axis limit, Latency
    yticks([0 5 10 16])
    yticks(hAx1(2),[0 15 30 45])
 end

columnname = {'6GHz,100MHz';'6GHz,400MHz'; '28GHz,100MHz'; '28GHz,400MHz'};%Column names
set(gca,'xticklabel',columnname) %Column names
ylabel(hAx1(1), 'Mean throughput(Mbps)');%Axis name Throughput
ylabel(hAx1(2), 'Mean latency(ms)');%Axis name Latency
title(Title)
legend('Mean throughput(Mbps)','Mean latency(ms)')
%set(gca,'ytick',linspace(0,0.25,1));
grid on;
grid minor;

%------------------------------------------------------------------

if(Guar == true)
    saveas(gcf,Title1,'jpg') %Save
```

```
end
if(Close == true)
    close all; %Figure close
end

% -----------------------------------------------------------------
```

APPENDIX C. NS-3 CODES

C.1. Examples of ns-3 simulations

C.1.1. cttc-3gpp-channel-simple-ran code

```



/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 *   Copyright (c) 2017 Centre Tecnologic de Telecomunicacions de Catalunya (CTTC)
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License version 2 as
 *   published by the Free Software Foundation;
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA
 *
 *
 *   Author: Biljana Bojovic <bbojovic@cttc.es>
 */

#include "ns3/core-module.h"
#include "ns3/network-module.h"
#include "ns3/mobility-module.h"
```

```
#include "ns3/config-store.h"
#include "ns3/mmwave-helper.h"
#include <ns3/buildings-helper.h>
#include "ns3/log.h"
#include <ns3/buildings-module.h>
#include "ns3/mmwave-point-to-point-epc-helper.h"
#include "ns3/network-module.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/internet-module.h"
#include "ns3/eps-bearer-tag.h"

using namespace ns3;

/**
 * \ingroup examples
 * \file cttc-3gpp-channel-simple-ran.cc
 * \brief Simple RAN
 *
 * This example describes how to setup a simulation using the 3
   GPP channel model
 * from TR 38.900. This example consists of a simple topology
   of 1 UE and 1 gNb,
 * and only NR RAN part is simulated. A packet is created and
   directly sent to
 * gNb device by SendPacket function. Then several functions
   are connected to
 * PDCP and RLC traces and the delay is printed.
 */


/**
 * \brief Global variable used to configure the numerology. It
   is accessible as "--numerology" from CommandLine.
 */
static ns3::GlobalValue g_numerology ("numerology",
                                      "The default 3GPP NR 
                                         numerology to be used"
                                         ,
                                      ns3::UintegerValue (0),
                                      ns3::MakeUintegerChecker<
                                         uint32_t>());

/**
 * \brief Global variable used to configure the bandwidth for
   packet size. This value is expressed in bytes. It is
```

```
    accessible as "--packetSize" from CommandLine.
 */

static ns3::GlobalValue g_udpInterval ("packetSize",
                                        "packet␣size␣in␣bytes",
                                        ns3::UintegerValue (1000),
                                        ns3::MakeUintegerChecker<uint32_t>());

/**
 * \brief Global boolean variable used to configure whether the UE performs the uplink traffic. It is accessible as "--isUplink" from CommandLine.
 */
static ns3::GlobalValue g_isUplink ("isUplink",
                                    "whether␣to␣perform␣uplink",
                                    ns3::BooleanValue (false),
                                    ns3::MakeBooleanChecker());


/**
 * Function creates a single packet and directly calls the function send
 * of a device to send the packet to the destination address.
 * @param device Device that will send the packet to the destination address.
 * @param addr Destination address for a packet.
 */
static void SendPacket (Ptr<NetDevice> device, Address& addr)
{
  UintegerValue uintegerValue;
  GlobalValue::GetValueByName("packetSize", uintegerValue); // use optional NLOS equation
  uint16_t packetSize = uintegerValue.Get();

  Ptr<Packet> pkt = Create<Packet> (packetSize);
  Ipv4Header ipv4Header;
  ipv4Header.SetProtocol(UdpL4Protocol::PROT_NUMBER);
  pkt->AddHeader(ipv4Header);
  EpsBearerTag tag (1, 1);
  pkt->AddPacketTag (tag);
  device->Send (pkt, addr, Ipv4L3Protocol::PROT_NUMBER);
}

```

```
/**
 * Function that prints out PDCP delay. This function is designed as a callback
 * for PDCP trace source.
 * @param path The path that matches the trace source
 * @param rnti RNTI of UE
 * @param lcid logical channel id
 * @param bytes PDCP PDU size in bytes
 * @param pdcpDelay PDCP delay
 */
void
RxPdcpPDU (std::string path, uint16_t rnti, uint8_t lcid, uint32_t bytes, uint64_t pdcpDelay)
{
  std::cout<<"\n Packet PDCP delay:"<<pdcpDelay<<"\n";
}

/**
 * Function that prints out RLC statistics, such as RNTI, lcId, RLC PDU size,
 * delay. This function is designed as a callback
 * for RLC trace source.
 * @param path The path that matches the trace source
 * @param rnti RNTI of UE
 * @param lcid logical channel id
 * @param bytes RLC PDU size in bytes
 * @param rlcDelay RLC PDU delay
 */
void
RxRlcPDU (std::string path, uint16_t rnti, uint8_t lcid, uint32_t bytes, uint64_t rlcDelay)
{
  std::cout<<"\n\n Data received at RLC layer at:"<<Simulator::Now()<<std::endl;
  std::cout<<"\n rnti:"<<rnti<<std::endl;
  std::cout<<"\n lcid:"<<(unsigned)lcid<<std::endl;
  std::cout<<"\n bytes :"<< bytes<<std::endl;
  std::cout<<"\n delay :"<< rlcDelay<<std::endl;
}

/**
 * Function that connects PDCP and RLC traces to the corresponding trace sources.
 */
void
```

```
ConnectPdcpRlcTraces ()
{
  Config::Connect ("/NodeList/1/DeviceList/0/LteUeRrc/DataRadioBearerMap/1/LtePdcp/RxPDU",
                      MakeCallback (&RxPdcpPDU));

  Config::Connect ("/NodeList/1/DeviceList/0/LteUeRrc/DataRadioBearerMap/1/LteRlc/RxPDU",
                      MakeCallback (&RxRlcPDU));
}

/**
 * Function that connects UL PDCP and RLC traces to the corresponding trace sources.
 */
void
ConnectUlPdcpRlcTraces ()
{
  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/UeMap/*/DataRadioBearerMap/*/LtePdcp/RxPDU",
                      MakeCallback (&RxPdcpPDU));

  Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/UeMap/*/DataRadioBearerMap/*/LteRlc/RxPDU",
                      MakeCallback (&RxRlcPDU));
}

int
main (int argc, char *argv[])
{

  CommandLine cmd;
  cmd.Parse (argc, argv);
  ConfigStore inputConfig;
  inputConfig.ConfigureDefaults ();
  // parse again so you can override input file default values via command line
  cmd.Parse (argc, argv);
  Time sendPacketTime = Seconds(0.4);

  UintegerValue uintegerValue;
  GlobalValue::GetValueByName("numerology", uintegerValue); // numerology to use
  uint16_t numerology = uintegerValue.Get();

```

```
BooleanValue boolValue;
GlobalValue::GetValueByName("isUplink", boolValue); // use uplink
bool uplink = boolValue.Get();

Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Frequency", DoubleValue(28e9));
Config::SetDefault ("ns3::MmWavePhyMacCommon::CenterFreq", DoubleValue(28e9));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Bandwidth", DoubleValue(400e6));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Numerology", UintegerValue(numerology));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Shadowing", BooleanValue(false));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::ChannelCondition", StringValue("l"));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("UMi-StreetCanyon"));

Config::SetDefault("ns3::MmWaveMacSchedulerNs3::FixedMcsDl", BooleanValue (true));
Config::SetDefault("ns3::MmWaveMacSchedulerNs3::McsDefaultDl", UintegerValue (28));

Ptr<MmWaveHelper> mmWaveHelper = CreateObject<MmWaveHelper>();
mmWaveHelper->SetAttribute ("PathlossModel", StringValue ("ns3::MmWave3gppPropagationLossModel"));
mmWaveHelper->SetAttribute ("ChannelModel", StringValue ("ns3::MmWave3gppChannel"));
Ptr<MmWavePointToPointEpcHelper> epcHelper = CreateObject<MmWavePointToPointEpcHelper> ();
mmWaveHelper->SetEpcHelper (epcHelper);

Ptr<Node> ueNode = CreateObject<Node> ();
Ptr<Node> gNbNode = CreateObject<Node> ();

MobilityHelper mobility;
mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.Install (gNbNode);
mobility.Install (ueNode);
gNbNode->GetObject<MobilityModel>()->SetPosition (Vector(0.0, 0.0, 10));
```

```
  ueNode->GetObject<MobilityModel> ()->SetPosition (Vector (0,
     10 , 1.5));

  NetDeviceContainer enbNetDev = mmWaveHelper->InstallEnbDevice
      (gNbNode);
  NetDeviceContainer ueNetDev = mmWaveHelper->InstallUeDevice (
     ueNode);

  InternetStackHelper internet;
  internet.Install (ueNode);
  Ipv4InterfaceContainer ueIpIface;
  ueIpIface = epcHelper->AssignUeIpv4Address (
     NetDeviceContainer (ueNetDev));

  if (uplink)
    Simulator::Schedule (sendPacketTime, &SendPacket, ueNetDev.
       Get(0), enbNetDev.Get(0)->GetAddress());
  else
    Simulator::Schedule (sendPacketTime, &SendPacket, enbNetDev
       .Get(0), ueNetDev.Get(0)->GetAddress());

  // attach UEs to the closest eNB
  mmWaveHelper->AttachToClosestEnb (ueNetDev, enbNetDev);

  if (uplink)
    {
      std::cout<<"\n␣Sending␣data␣in␣uplink."<<std::endl;
      Simulator::Schedule(Seconds(0.2), &ConnectUlPdcpRlcTraces
         );
    }
  else
    {
      std::cout<<"\n␣Sending␣data␣in␣downlink."<<std::endl;
      Simulator::Schedule(Seconds(0.2), &ConnectPdcpRlcTraces);
    }

  mmWaveHelper->EnableTraces();

  Simulator::Stop (Seconds (1));
  Simulator::Run ();
  Simulator::Destroy ();
}
```

C.1.2. cttc-3gpp-channel-nums code

This is the most relevant code of the simulation. All of the results obtained from this simulations, are based on this original code.

```

/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil;
   -*- */
/*
 *   Copyright (c) 2017 Centre Tecnologic de Telecomunicacions
     de Catalunya (CTTC)
 *
 *   This program is free software; you can redistribute it and
     /or modify
 *   it under the terms of the GNU General Public License
     version 2 as
 *   published by the Free Software Foundation;
 *
 *   This program is distributed in the hope that it will be
     useful,
 *   but WITHOUT ANY WARRANTY; without even the implied
     warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See
     the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public
     License
 *   along with this program; if not, write to the Free
     Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA
     02111-1307  USA
 *
 *
 *   Author:  Biljana Bojovic <bbojovic@cttc.es>

*/

#include "ns3/core-module.h"
#include "ns3/config-store.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/log.h"
```

```
#include "ns3/point-to-point-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mmwave-helper.h"
#include "ns3/mmwave-point-to-point-epc-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/log.h"


/**
 * \file cttc-3gpp-channel-nums.cc
 * \ingroup examples
 * \brief Simple topology numerologies example.
 *
 * This example allows users to configure the numerology and test the end-to-end
 * performance for different numerologies. In the following figure we illustrate the simulation setup.
 *
 * For example, UDP interval can be configured by setting
 * "--udpInterval=0.001". The numerology can be toggled by the argument,
 * e.g. "--numerology=1". Additionally, in this example two arguments
 * are added "bandwidth" and "frequency". The modulation scheme of
 * this example is in test mode, and it is fixed to 28.
 *
 * By default, the program uses the 3GPP channel model, without shadowing and with
 * line of sight ('l') option. The program runs for 0.4 seconds and one single packet
 * is to be transmitted. The packet size can be configured by using  the
 * following parameter: "--packetSize=1000".
 *
 * This simulation prints the output to the terminal and also to the file which
 * is named by default "cttc-3gpp-channel-nums-fdm-output" and which is by
 * default placed in the root directory of the project.
 *
 * To run the simulation with the default configuration one shall run the
 * following in the command line:
 *
```

```
 * ./waf --run cttc-3gpp-channel-nums
 *
 */

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("3gppChannelNumerologiesExample");

static ns3::GlobalValue g_frequency("frequency",
                                    "The␣system␣frequency",
                                     ns3::DoubleValue(28e9),
                                     ns3::MakeDoubleChecker<double>(6e9,100e9));//!< Global variable used to configure the frequency. It is accessible as "--frequency" from CommandLine.

static ns3::GlobalValue g_bandwidth("bandwidth",
                                    "The␣system␣bandwidth",
                                     ns3::DoubleValue(200e6),
                                     ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure the bandwidth. It is accessible as "--bandwidth" from CommandLine.

static ns3::GlobalValue g_numerology ("numerology",
                                      "The␣default␣3GPP␣NR␣numerology␣to␣be␣used",
                                      ns3::UintegerValue (2),
                                      ns3::MakeUintegerChecker<uint32_t>());//!< Global variable used to configure the numerology. It is accessible as "--numerology" from CommandLine.

```

```
static ns3::GlobalValue g_udpInterval ("udpInterval",
                                       "Udp interval for UDP application packet arrival, in seconds",
                                       ns3::DoubleValue (0.01),
                                       ns3::MakeDoubleChecker<double> ());//!< Global variable used to configure the UDP packet interval. It is accessible as "--udpInterval" from CommandLine.

static ns3::GlobalValue g_udpPacketSize ("udpPacketSize",
                                         "Udp packet size in bytes",
                                         ns3::UintegerValue (150000),
                                         ns3::MakeUintegerChecker<uint32_t> ()); //!< Global variable used to configure the UDP packet size. It is accessible as "--udpPacketSize" from CommandLine.

static ns3::GlobalValue g_udpRate ("udpFullBuffer",
                                   "Whether to set the full buffer traffic; if this parameter is set then the udpInterval parameter"
                                   "will be neglected.",
                                   ns3::BooleanValue (true),
                                   ns3::MakeBooleanChecker ()); //!< Global variable used to configure whether the traffic is the full buffer traffic. It is accessible as "--udpFullBuffer" from CommandLine.

```

```
static ns3::GlobalValue g_singleUeTopology ("singleUeTopology",
                                             "When true the example uses a single UE topology, when false use topology with variable number of UEs"
                                             "will be neglected.",
                                             ns3::BooleanValue (false),
                                             ns3::MakeBooleanChecker ()); //!< Global variable used to configure whether topology is with single of various number of UEs. It is accessible as "--singleUeTopology" from CommandLine.

static ns3::GlobalValue g_useFixedMcs ("useFixedMcs",
                                        "Whether to use fixed mcs, normally used for testing purposes",
                                        ns3::BooleanValue (true),
                                        ns3::MakeBooleanChecker ()); //!< Global variable used to configure whether to use fixed MCS. It is accessible as "--useFixedMcs" from CommandLine.

static ns3::GlobalValue g_fixedMcs ("fixedMcs",
```

```
                                   "The MCS that will be used in this example",
                                   ns3::UintegerValue (28),
                                   ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure fixed MCS. It is accessible as "--fixedMcs" from CommandLine.

static ns3::GlobalValue g_gNbNum ("gNbNum",
                                  "The number of gNbs in multiple-ue topology",
                                  ns3::UintegerValue (1),
                                  ns3::MakeUintegerChecker<uint32_t>());//!< Global variable used to configure the number of gNbs in multi-UE topology. It is accessible as "--gNbNum" from CommandLine.

static ns3::GlobalValue g_ueNumPergNb ("ueNumPergNb",
                                  "The number of UE per gNb in multiple-ue topology",
                                 ns3::UintegerValue (1),
                                 ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the number of UEs in multi-UE topology. It is accessible as "--ueNumPergNb" from CommandLine.

static ns3::GlobalValue g_cellScan ("cellScan",
                                   "Use beam search method to determine beamforming vector, the default is long-term covariance matrix method"
                                   "true to use cell scanning method, false to use the default power method.",
```

```
                                      ns3::BooleanValue (false),
                                      ns3::MakeBooleanChecker()); //!< Global variable used to configure whether to use Beam Search of Long-Term Cov. matrix for beamforming. It is accessible as "--cellScan" from CommandLine.

static ns3::GlobalValue g_beamSearchAngleStep ("beamSearchAngleStep",
                                                "Beam search angle step for beam search method",
                                                ns3::DoubleValue (10),
                                                ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure beam search angle step in the case that beam search method is used. It is accessible as "--beamSearchAngleStep" from CommandLine.

static ns3::GlobalValue g_txPower ("txPower",
                                   "Tx power",
                                   ns3::DoubleValue (1),
                                   ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure gNb TX power.
```

```
                                           It is accessible as "--
                                           txPower" from
                                           CommandLine.

static ns3::GlobalValue g_simTag ("simTag",
                                  "tag␣to␣be␣appended␣to␣output
                                     ␣filenames␣to␣distinguish␣
                                     simulation␣campaigns",
                                  ns3::StringValue ("default"),
                                  ns3::MakeStringChecker ());
                                     //!< Global variable used
                                     to configure simulation
                                     output tag that helps
                                     distinguishing different
                                     simulation campaigns. It
                                     is accessible as "--simTag
                                     " from CommandLine.

static ns3::GlobalValue g_outputDir ("outputDir",
                                     "directory␣where␣to␣store␣
                                        simulation␣results",
                                     ns3::StringValue ("./"),
                                     ns3::MakeStringChecker ())
                                        ;  //!< Global variable
                                         used to configure
                                        simulation output
                                        folder. It is
                                        accessible as "--
                                        outputDir" from
                                        CommandLine.

int
main (int argc, char *argv[])
{

    CommandLine cmd;
    cmd.Parse (argc, argv);
    ConfigStore inputConfig;
    inputConfig.ConfigureDefaults ();
    // parse again so you can override input file default
       values via command line
    cmd.Parse (argc, argv);

  // enable logging or not
  bool logging = false;
```

```
  if(logging)
    {
      LogComponentEnable ("MmWave3gppPropagationLossModel",
         LOG_LEVEL_ALL);
      LogComponentEnable ("
         MmWave3gppBuildingsPropagationLossModel",
         LOG_LEVEL_ALL);
      LogComponentEnable ("MmWave3gppChannel", LOG_LEVEL_ALL);
      LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
      LogComponentEnable ("LtePdcp", LOG_LEVEL_INFO);

    }

  // set simulation time and mobility
  double simTime = 1; // seconds
  double udpAppStartTime = 0.4; //seconds
  //double speed = 1; // 1 m/s for walking UT.

  // parse the command line options
  BooleanValue booleanValue;
  StringValue stringValue;
  IntegerValue integerValue;
  UintegerValue uintegerValue;
  DoubleValue doubleValue;
  GlobalValue::GetValueByName("numerology", uintegerValue); //
     use optional NLOS equation
  uint16_t numerology = uintegerValue.Get();
  GlobalValue::GetValueByName("fixedMcs", uintegerValue); //
     use optional NLOS equation
  uint16_t fixedMcs = uintegerValue.Get();
  GlobalValue::GetValueByName("gNbNum", uintegerValue); // use
     optional NLOS equation
  uint16_t gNbNum = uintegerValue.Get();
  GlobalValue::GetValueByName("ueNumPergNb", uintegerValue); //
      use optional NLOS equation
  uint16_t ueNumPergNb = uintegerValue.Get();
  GlobalValue::GetValueByName("udpInterval", doubleValue); //
     use optional NLOS equation
  double udpInterval = doubleValue.Get();
  GlobalValue::GetValueByName("udpPacketSize", uintegerValue);
     // use optional NLOS equation
  uint32_t udpPacketSize = uintegerValue.Get();
  GlobalValue::GetValueByName("frequency", doubleValue); //
  double frequency = doubleValue.Get();
```

```cpp
GlobalValue::GetValueByName("udpFullBuffer", booleanValue); //
bool udpFullBuffer = booleanValue.Get();
GlobalValue::GetValueByName("singleUeTopology", booleanValue); //
bool singleUeTopology = booleanValue.Get();
GlobalValue::GetValueByName("bandwidth", doubleValue); //
double bandwidth = doubleValue.Get();
GlobalValue::GetValueByName("cellScan", booleanValue); //
bool cellScan = booleanValue.Get();
GlobalValue::GetValueByName("useFixedMcs", booleanValue); //
bool useFixedMcs = booleanValue.Get();
GlobalValue::GetValueByName("beamSearchAngleStep", doubleValue); // use optional NLOS equation
double beamSearchAngleStep = doubleValue.Get();
GlobalValue::GetValueByName("txPower", doubleValue); // use optional NLOS equation
double txPower = doubleValue.Get();
GlobalValue::GetValueByName ("simTag", stringValue);
std::string simTag = stringValue.Get ();
GlobalValue::GetValueByName ("outputDir", stringValue);
std::string outputDir = stringValue.Get ();

// attributes that can be set for this channel model
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Frequency", DoubleValue(frequency));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::ChannelCondition", StringValue("l"));

if (singleUeTopology)
  {
  //Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("UMi-StreetCanyon"));
    Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("RMa"));
  }
else
  {
    Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("InH-OfficeOpen"));
  }

Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Shadowing", BooleanValue(false));

```

```
Config::SetDefault ("ns3::MmWave3gppChannel::CellScan",
   BooleanValue(cellScan));
Config::SetDefault ("ns3::MmWave3gppChannel::
   BeamSearchAngleStep", DoubleValue(beamSearchAngleStep));

Config::SetDefault ("ns3::MmWavePhyMacCommon::CenterFreq",
   DoubleValue(frequency));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Bandwidth",
   DoubleValue(bandwidth));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Numerology",
   UintegerValue(numerology));

Config::SetDefault ("ns3::LteRlcUm::MaxTxBufferSize",
   UintegerValue(999999999));

Config::SetDefault("ns3::MmWaveMacSchedulerNs3::FixedMcsDl",
   BooleanValue (useFixedMcs));
Config::SetDefault("ns3::MmWaveMacSchedulerNs3::McsDefaultDl"
   , UintegerValue (fixedMcs));

//Config::SetDefault("ns3::MmWaveUeNetDevice::AntennaNum",
   UintegerValue (4));
//Config::SetDefault("ns3::MmWaveEnbNetDevice::AntennaNum",
   UintegerValue (16));

Config::SetDefault("ns3::MmWaveEnbPhy::TxPower", DoubleValue
   (txPower));

// setup the mmWave simulation
Ptr<MmWaveHelper> mmWaveHelper = CreateObject<MmWaveHelper>
   ();
mmWaveHelper->SetAttribute ("PathlossModel", StringValue ("
   ns3::MmWave3gppPropagationLossModel"));
mmWaveHelper->SetAttribute ("ChannelModel", StringValue ("ns3
   ::MmWave3gppChannel"));

Ptr<MmWavePointToPointEpcHelper> epcHelper = CreateObject<
   MmWavePointToPointEpcHelper> ();
mmWaveHelper->SetEpcHelper (epcHelper);
mmWaveHelper->Initialize();

// create base stations and mobile terminals
NodeContainer gNbNodes;
NodeContainer ueNodes;
MobilityHelper mobility;
```

```

  double gNbHeight = 10;
  double ueHeight = 1.5;

  if (singleUeTopology)
    {
      gNbNodes.Create (1);
      ueNodes.Create (1);
      mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
      mobility.Install (gNbNodes);
      mobility.Install (ueNodes);
      gNbNodes.Get (0)->GetObject<MobilityModel>()->SetPosition (Vector (0.0, 0.0, gNbHeight));
      ueNodes.Get (0)->GetObject<MobilityModel> ()->SetPosition (Vector (0.0, 30.0 , ueHeight));
    }
  else
    {
      gNbNodes.Create (gNbNum);
      ueNodes.Create (ueNumPergNb * gNbNum);

      MobilityHelper mobility;
      Ptr<ListPositionAllocator> apPositionAlloc = CreateObject<ListPositionAllocator> ();
      Ptr<ListPositionAllocator> staPositionAlloc = CreateObject<ListPositionAllocator> ();
      int32_t yValue = 0.0;

      for (uint32_t i = 1; i <= gNbNodes.GetN (); ++i)
        {
          // 2.0, -2.0, 6.0, -6.0, 10.0, -10.0, ....
          if (i % 2 != 0)
            {
              yValue = static_cast<int>(i) * 30;
            }
          else
            {
              yValue = -yValue;
            }

          apPositionAlloc->Add (Vector (0.0, yValue, gNbHeight));


```

```
            // 1.0, -1.0, 3.0, -3.0, 5.0, -5.0, ...
            double xValue = 0.0;
            for (uint32_t j = 1; j <= ueNumPergNb; ++j)
              {
                if (j % 2 != 0)
                  {
                    xValue = j;
                  }
                else
                  {
                    xValue = -xValue;
                  }

                if (yValue > 0)
                  {
                    staPositionAlloc->Add (Vector (xValue, 1,
                       ueHeight));
                  }
                else
                  {
                    staPositionAlloc->Add (Vector (xValue, -1,
                       ueHeight));
                  }
              }
        }

      mobility.SetMobilityModel ("ns3::
         ConstantPositionMobilityModel");
      mobility.SetPositionAllocator (apPositionAlloc);
      mobility.Install (gNbNodes);

      mobility.SetPositionAllocator (staPositionAlloc);
      mobility.Install (ueNodes);
    }

  // install mmWave net devices
  NetDeviceContainer enbNetDev = mmWaveHelper->InstallEnbDevice
      (gNbNodes);
  NetDeviceContainer ueNetDev = mmWaveHelper->InstallUeDevice (
     ueNodes);

  // create the internet and install the IP stack on the UEs
  // get SGW/PGW and create a single RemoteHost
  Ptr<Node> pgw = epcHelper->GetPgwNode ();
  NodeContainer remoteHostContainer;
```

```
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);

// connect a remoteHost to pgw. Setup routing too
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate
   ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (2500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.000)
   ));
NetDeviceContainer internetDevices = p2ph.Install (pgw,
   remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (
   internetDevices);
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
   ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<
   Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("
   7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);
internet.Install (ueNodes);
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address (
   NetDeviceContainer (ueNetDev));
// assign IP address to UEs, and install UDP downlink
   applications
uint16_t dlPort = 1234;
ApplicationContainer clientApps;
ApplicationContainer serverApps;

// Set the default gateway for the UEs
for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    Ptr<Ipv4StaticRouting> ueStaticRouting =
       ipv4RoutingHelper.GetStaticRouting (ueNodes.Get(j)->
       GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->
       GetUeDefaultGatewayAddress (), 1);
  }

UdpServerHelper dlPacketSinkHelper (dlPort);
```

```
serverApps.Add (dlPacketSinkHelper.Install (ueNodes.Get(0)));

for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    UdpClientHelper dlClient (ueIpIface.GetAddress (j),
       dlPort);
    dlClient.SetAttribute("PacketSize", UintegerValue(
       udpPacketSize));
    dlClient.SetAttribute ("MaxPackets", UintegerValue(0
       xFFFFFFFF));
    //dlClient.SetAttribute ("MaxPackets", UintegerValue
       (1000));

    if (udpFullBuffer)
      {
        double bitRate = 75000000; // 75 Mb/s will saturate
           the system of 20 MHz

        if (bandwidth > 20e6)
          {
            bitRate *=  bandwidth / 20e6;
          }
        udpInterval = static_cast<double> (udpPacketSize * 8)
            / bitRate ;
      }
    dlClient.SetAttribute ("Interval", TimeValue (Seconds(
       udpInterval)));
    clientApps.Add (dlClient.Install (remoteHost));

    Ptr<EpcTft> tft = Create<EpcTft> ();
    EpcTft::PacketFilter dlpf;
    dlpf.localPortStart = dlPort;
    dlpf.localPortEnd = dlPort;
    dlPort++;
    tft->Add (dlpf);

    enum EpsBearer::Qci q;
    q = EpsBearer::GBR_CONV_VOICE;
    EpsBearer bearer (q);
    mmWaveHelper->ActivateDedicatedEpsBearer(ueNetDev.Get(j),
        bearer, tft);
  }

// start server and client apps
serverApps.Start(Seconds(udpAppStartTime));
```

```
clientApps.Start(Seconds(udpAppStartTime));
serverApps.Stop(Seconds(simTime));
clientApps.Stop(Seconds(simTime));

// attach UEs to the closest eNB
mmWaveHelper->AttachToClosestEnb (ueNetDev, enbNetDev);

// enable the traces provided by the mmWave module
//mmWaveHelper->EnableTraces();


FlowMonitorHelper flowmonHelper;
NodeContainer endpointNodes;
endpointNodes.Add (remoteHost);
endpointNodes.Add (ueNodes);

Ptr<ns3::FlowMonitor> monitor = flowmonHelper.Install (
   endpointNodes);
monitor->SetAttribute ("DelayBinWidth", DoubleValue (0.001));
monitor->SetAttribute ("JitterBinWidth", DoubleValue (0.001))
   ;
monitor->SetAttribute ("PacketSizeBinWidth", DoubleValue (20)
   );


Simulator::Stop (Seconds (simTime));
Simulator::Run ();



// Print per-flow statistics
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<
   Ipv4FlowClassifier> (flowmonHelper.GetClassifier ());
FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats
    ();

double averageFlowThroughput = 0.0;
double averageFlowDelay = 0.0;

std::ofstream outFile;
std::string filename = outputDir + "/" + simTag;
outFile.open (filename.c_str (), std::ofstream::out | std::
   ofstream::app);
if (!outFile.is_open ())
```

```
    {
      NS_LOG_ERROR ("Can't open file " << filename);
      return 1;
    }
  outFile.setf (std::ios_base::fixed);

  for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator
      i = stats.begin (); i != stats.end (); ++i)
    {
      Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i
          ->first);
      std::stringstream protoStream;
      protoStream << (uint16_t) t.protocol;
      if (t.protocol == 6)
        {
          protoStream.str ("TCP");
        }
      if (t.protocol == 17)
        {
          protoStream.str ("UDP");
        }
      outFile << "Flow " << i->first << " (" << t.sourceAddress
           << ":" << t.sourcePort << " -> " << t.
          destinationAddress << ":" << t.destinationPort << ") 
          proto " << protoStream.str () << "\n";
      outFile << "  Tx Packets: " << i->second.txPackets << "\n
          ";
      outFile << "  Tx Bytes:   " << i->second.txBytes << "\n";
      outFile << "  TxOffered:  " << i->second.txBytes * 8.0 /
          (simTime - udpAppStartTime) / 1000 / 1000  << " Mbps\n
          ";
      outFile << "  Rx Bytes:   " << i->second.rxBytes << "\n";
      if (i->second.rxPackets > 0)
        {
          // Measure the duration of the flow from receiver's
              perspective
          double rxDuration = i->second.timeLastRxPacket.
              GetSeconds () - i->second.timeFirstTxPacket.
              GetSeconds ();

          averageFlowThroughput += i->second.rxBytes * 8.0 /
              rxDuration / 1000 / 1000;
          averageFlowDelay += 1000 * i->second.delaySum.
              GetSeconds () / i->second.rxPackets;

```

```
            outFile << "  Throughput: " << i->second.rxBytes * 8.0 / rxDuration / 1000 / 1000  << " Mbps\n";
            outFile << "  Mean delay:  " << 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets << " ms\n";
            //outFile << "  Mean upt:  " << i->second.uptSum / i->second.rxPackets / 1000/1000 << " Mbps \n";
            outFile << "  Mean jitter:  " << 1000 * i->second.jitterSum.GetSeconds () / i->second.rxPackets  << " ms\n";
          }
        else
          {
            outFile << "  Throughput:  0 Mbps\n";
            outFile << "  Mean delay:  0 ms\n";
            outFile << "  Mean upt:  0  Mbps \n";
            outFile << "  Mean jitter: 0 ms\n";
          }
        outFile << "  Rx Packets: " << i->second.rxPackets << "\n";
    }

  outFile << "\n\n  Mean flow throughput: " << averageFlowThroughput / stats.size() << "\n";
  outFile << "  Mean flow delay: " << averageFlowDelay / stats.size () << "\n";
  outFile.close ();

  Ptr<UdpClient> clientApp = clientApps.Get(0)->GetObject<UdpClient>();
  Ptr<UdpServer> serverApp = serverApps.Get(0)->GetObject<UdpServer>();
  std::cout<<"\n Total UDP throughput (bps):"<<(serverApp->GetReceived()*udpPacketSize*8)/(simTime-udpAppStartTime)<<std::endl;

  Simulator::Destroy ();
  return 0;
}
```

C.1.3. Modified $cttc-3gpp-channel-nums$ code for down-link configurations

This code is the one used to obtain down-link simulations on the simulation scenarios.

```
/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 *   Copyright (c) 2017 Centre Tecnologic de Telecomunicacions de Catalunya (CTTC)
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License version 2 as
 *   published by the Free Software Foundation;
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA
 *
 *
 *   Author:  Biljana Bojovic <bbojovic@cttc.es>

*/

#include "ns3/core-module.h"
#include "ns3/config-store.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/log.h"
#include "ns3/point-to-point-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mmwave-helper.h"
#include "ns3/mmwave-point-to-point-epc-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
```

```
#include "ns3/log.h"
//#include "<koolplot.h>
#include <string>
#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;


/**
 * \file cttc-3gpp-channel-nums.cc
 * \ingroup examples
 * \brief Simple topology numerologies example.
 *
 * This example allows users to configure the numerology and test the end-to-end
 * performance for different numerologies. In the following figure we illustrate the simulation setup.
 *
 * For example, UDP interval can be configured by setting
 * "--udpInterval=0.001". The numerology can be toggled by the argument,
 * e.g. "--numerology=1". Additionally, in this example two arguments
 * are added "bandwidth" and "frequency". The modulation scheme of
 * this example is in test mode, and it is fixed to 28.
 *
 * By default, the program uses the 3GPP channel model, without shadowing and with
 * line of sight ('l') option. The program runs for 0.4 seconds and one single packet
 * is to be transmitted. The packet size can be configured by using  the
 * following parameter: "--packetSize=1000".
 *
 * This simulation prints the output to the terminal and also to the file which
 * is named by default "cttc-3gpp-channel-nums-fdm-output" and which is by
 * default placed in the root directory of the project.
 *
 * To run the simulation with the default configuration one shall run the
```

```
 * following in the command line:
 *
 * ./waf --run cttc-3gpp-channel-nums
 *
 */

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("3gppChannelNumerologiesExample");

static ns3::GlobalValue g_frequency("frequency",
                                    "The␣system␣frequency",
                                     ns3::DoubleValue(6e9),
                                     ns3::MakeDoubleChecker<double>(6e9,100e9));//!< Global variable used to configure the frequency. It is accessible as "--frequency" from CommandLine.

static ns3::GlobalValue g_bandwidth("bandwidth",
                                    "The␣system␣bandwidth",
                                     ns3::DoubleValue(100e6),
                                     ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure the bandwidth. It is accessible as "--bandwidth" from CommandLine.

static ns3::GlobalValue g_numerology ("numerology",
                                      "The␣default␣3GPP␣NR␣numerology␣to␣be␣used",
                                      ns3::UintegerValue (2),
                                      ns3::MakeUintegerChecker<uint32_t>());//!< Global variable used to configure the numerology. It is accessible as "--numerology" from
```

```
CommandLine.

static ns3::GlobalValue g_udpInterval ("udpInterval",
                                       "Udp interval for UDP application packet arrival, in seconds",
                                       ns3::DoubleValue (0.01), //ns3::DoubleValue (0.00075),
                                       ns3::MakeDoubleChecker<double>());//!< Global variable used to configure the UDP packet interval. It is accessible as "--udpInterval" from CommandLine.

static ns3::GlobalValue g_udpPacketSize ("udpPacketSize",
                                         "Udp packet size in bytes",
                                         ns3::UintegerValue (1000), //(20000),
                                         ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the UDP packet size. It is accessible as "--udpPacketSize" from CommandLine.

static ns3::GlobalValue g_udpRate ("udpFullBuffer",
                                   "Whether to set the full buffer traffic; if this parameter is set then the udpInterval parameter"
                                   "will be neglected.",
                                   ns3::BooleanValue (false),
                                   ns3::MakeBooleanChecker()); //!< Global variable used to configure whether the traffic is the full buffer traffic. It is
```

```
                                                    accessible as "--
                                                    udpFullBuffer" from
                                                    CommandLine.

//static ns3::GlobalValue g_singleUeTopology ("singleUeTopology
   ",
                                                  // "When true the
                                                     example uses a
                                                     single UE
                                                     topology, //when
                                                     false use
                                                     topology with
                                                     variable number
                                                     of UEs"
                                                  // "will be
                                                     neglected.",
                                                   //ns3::BooleanValue
                                                       (false),
                                                   //ns3::
                                                      MakeBooleanChecker
                                                      ()); //!< Global
                                                       variable used
                                                      //to configure
                                                      whether topology
                                                       is with single
                                                      of //various
                                                      number of UEs.
                                                      It is accessible
                                                       as //"--
                                                      singleUeTopology
                                                      " from
                                                      CommandLine.

static ns3::GlobalValue g_useFixedMcs ("useFixedMcs",
                                       "Whether to use fixed
                                          mcs, normally used
                                          for testing purposes"
                                          ,
                                        ns3::BooleanValue (
                                           false),
                                        ns3::MakeBooleanChecker
                                           ()); //!< Global
                                           variable used to
                                           configure whether to
                                            use fixed MCS. It
```

```
                                                                   is accessible as "--useFixedMcs" from CommandLine.

static ns3::GlobalValue g_fixedMcs ("fixedMcs",
                                    "The MCS that will be used in this example",
                                    ns3::UintegerValue (28),
                                    ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure fixed MCS. It is accessible as "--fixedMcs" from CommandLine.

static ns3::GlobalValue g_gNbNum ("gNbNum",
                                  "The number of gNbs in multiple-ue topology",
                                  ns3::UintegerValue (2),
                                  ns3::MakeUintegerChecker<uint32_t>());//!< Global variable used to configure the number of gNbs in multi-UE topology. It is accessible as "--gNbNum" from CommandLine.

static ns3::GlobalValue g_ueNum ("ueNum",
                                 "The number of UEs in multiple-ue topology",
                                 ns3::UintegerValue (10),
                                 ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the number of UEs in multi-UE topology. It is accessible as "--ueNumPergNb" from CommandLine.

static ns3::GlobalValue g_cellScan ("cellScan",
                                    "Use beam search method to determine beamforming vector, the default is 
```

```
                                          long-term covariance matrix method"
                                          "true to use cell scanning method, false to use the default power method.",
                                          ns3::BooleanValue (true),
                                          ns3::MakeBooleanChecker()); //!< Global variable used to configure whether to use Beam Search of Long-Term Cov. matrix for beamforming. It is accessible as "--cellScan" from CommandLine.

static ns3::GlobalValue g_beamSearchAngleStep ("beamSearchAngleStep",
                                               "Beam search angle step for beam search method",
                                               ns3::DoubleValue (30),
                                               ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure beam search angle step in the case that beam search method is used. It is accessible as "--beamSearchAngleStep" from CommandLine.

static ns3::GlobalValue g_txPower ("txPower",
                                   "Tx power",
```

```cpp
                                    ns3::DoubleValue (4),
                                    ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure gNb TX power. It is accessible as "--txPower" from CommandLine.

static ns3::GlobalValue g_simTag ("simTag",
                                  "tag to be appended to output filenames to distinguish simulation campaigns",
                                  ns3::StringValue ("test-toni-ex-2"),
                                  ns3::MakeStringChecker ()); //!< Global variable used to configure simulation output tag that helps distinguishing different simulation campaigns. It is accessible as "--simTag" from CommandLine.

static ns3::GlobalValue g_outputDir ("outputDir",
                                     "directory where to store simulation results",
                                     ns3::StringValue ("./"),
                                     ns3::MakeStringChecker ()); //!< Global variable used to configure simulation output folder. It is accessible as "--outputDir" from CommandLine.

int
main (int argc, char *argv[])
{

    CommandLine cmd;
    cmd.Parse (argc, argv);
    ConfigStore inputConfig;
    inputConfig.ConfigureDefaults ();
```

```
    // parse again so you can override input file default values via command line
    cmd.Parse (argc, argv);

  // enable logging or not
  bool logging = false;
  if(logging)
    {
      LogComponentEnable ("MmWave3gppPropagationLossModel", LOG_LEVEL_ALL);
      LogComponentEnable ("MmWave3gppBuildingsPropagationLossModel", LOG_LEVEL_ALL);
      LogComponentEnable ("MmWave3gppChannel", LOG_LEVEL_ALL);
      LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
      LogComponentEnable ("LtePdcp", LOG_LEVEL_INFO);

    }

  // set simulation time and mobility
  double simTime = 1; // seconds
  double udpAppStartTime = 0.4; //seconds
  //double speed = 1; // 1 m/s for walking UT.

  // parse the command line options
  BooleanValue booleanValue;
  StringValue stringValue;
  IntegerValue integerValue;
  UintegerValue uintegerValue;
  DoubleValue doubleValue;
  GlobalValue::GetValueByName("numerology", uintegerValue); // use optional NLOS equation
  uint16_t numerology = uintegerValue.Get();
  GlobalValue::GetValueByName("fixedMcs", uintegerValue); // use optional NLOS equation
  uint16_t fixedMcs = uintegerValue.Get();
  GlobalValue::GetValueByName("gNbNum", uintegerValue); // use optional NLOS equation
  uint16_t gNbNum = uintegerValue.Get();
  GlobalValue::GetValueByName("ueNum", uintegerValue);
  uint16_t ueNum = uintegerValue.Get();
  GlobalValue::GetValueByName("udpInterval", doubleValue);
  double udpInterval = doubleValue.Get();
  GlobalValue::GetValueByName("udpPacketSize", uintegerValue);
```

```
uint32_t udpPacketSize = uintegerValue.Get();
GlobalValue::GetValueByName("frequency", doubleValue); //
double frequency = doubleValue.Get();
GlobalValue::GetValueByName("udpFullBuffer", booleanValue);
   //
bool udpFullBuffer = booleanValue.Get();
GlobalValue::GetValueByName("singleUeTopology", booleanValue)
   ; //
bool singleUeTopology = booleanValue.Get();
GlobalValue::GetValueByName("bandwidth", doubleValue); //
double bandwidth = doubleValue.Get();
GlobalValue::GetValueByName("cellScan", booleanValue); //
bool cellScan = booleanValue.Get();
GlobalValue::GetValueByName("useFixedMcs", booleanValue); //
bool useFixedMcs = booleanValue.Get();
GlobalValue::GetValueByName("beamSearchAngleStep",
   doubleValue); // use optional NLOS equation
double beamSearchAngleStep = doubleValue.Get();
GlobalValue::GetValueByName("txPower", doubleValue); // use
   optional NLOS equation
double txPower = doubleValue.Get();
GlobalValue::GetValueByName ("simTag", stringValue);
std::string simTag = stringValue.Get ();
GlobalValue::GetValueByName ("outputDir", stringValue);
std::string outputDir = stringValue.Get ();

// attributes that can be set for this channel model
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::
   Frequency", DoubleValue(frequency));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::
   ChannelCondition", StringValue("l"));

if (singleUeTopology)
  {
  //Config::SetDefault ("ns3::MmWave3gppPropagationLossModel
     ::Scenario", StringValue("UMi-StreetCanyon"));
    Config::SetDefault ("ns3::MmWave3gppPropagationLossModel
       ::Scenario", StringValue("RMa"));
  }
else
  {
    Config::SetDefault ("ns3::MmWave3gppPropagationLossModel
       ::Scenario", StringValue("InH-OfficeOpen"));
  }

```

```
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::
   Shadowing", BooleanValue(false));

Config::SetDefault ("ns3::MmWave3gppChannel::CellScan",
   BooleanValue(cellScan));
Config::SetDefault ("ns3::MmWave3gppChannel::
   BeamSearchAngleStep", DoubleValue(beamSearchAngleStep));

Config::SetDefault ("ns3::MmWavePhyMacCommon::CenterFreq",
   DoubleValue(frequency));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Bandwidth",
   DoubleValue(bandwidth));
Config::SetDefault ("ns3::MmWavePhyMacCommon::Numerology",
   UintegerValue(numerology));

Config::SetDefault ("ns3::LteRlcUm::MaxTxBufferSize",
   UintegerValue(999999999));

Config::SetDefault("ns3::MmWaveMacSchedulerNs3::FixedMcsDl",
   BooleanValue (useFixedMcs));
Config::SetDefault("ns3::MmWaveMacSchedulerNs3::McsDefaultDl"
   , UintegerValue (fixedMcs));

//Config::SetDefault("ns3::MmWaveUeNetDevice::AntennaNum",
   UintegerValue (4));
//Config::SetDefault("ns3::MmWaveEnbNetDevice::AntennaNum",
   UintegerValue (16));

Config::SetDefault("ns3::MmWaveEnbPhy::TxPower", DoubleValue
   (txPower));
Config::SetDefault ("ns3::MmWavePhyMacCommon::
   MacSchedulerType",
TypeIdValue (TypeId::LookupByName("ns3::
   MmWaveMacSchedulerTdmaRR"))); // l nia modificada
   --------------------------------
                                                          //
                                                            TdmaPF

                                                            OfdmaRR

// setup the mmWave simulation
Ptr<MmWaveHelper> mmWaveHelper = CreateObject<MmWaveHelper>
   ();
mmWaveHelper->SetAttribute ("PathlossModel", StringValue ("
   ns3::MmWave3gppPropagationLossModel"));
```

```
mmWaveHelper->SetAttribute ("ChannelModel", StringValue ("ns3::MmWave3gppChannel"));

Ptr<MmWavePointToPointEpcHelper> epcHelper = CreateObject<MmWavePointToPointEpcHelper> ();
mmWaveHelper->SetEpcHelper (epcHelper);
mmWaveHelper->Initialize();

// create base stations and mobile terminals
NodeContainer gNbNodes;
NodeContainer ueNodes;
MobilityHelper mobility;

double gNbHeight = 1.5;
double ueHeight = 1.5;

gNbNodes.Create (gNbNum);
ueNodes.Create (ueNum);

Ptr<ListPositionAllocator> apPositionAlloc = CreateObject<ListPositionAllocator> ();
Ptr<ListPositionAllocator> staPositionAlloc = CreateObject<ListPositionAllocator> ();

//------------------Base Station Creation -->gNb Num <---------------------------------------------------------------


if (gNbNum == 2)
    {
        //#1
        apPositionAlloc->Add (Vector (10.5, 30.5, gNbHeight));
        std::cout<<"\n gNB position:"<<Vector (10.5, 30.5, gNbHeight)<<std::endl;
        //#2
        apPositionAlloc->Add (Vector (30.5, 10.5, gNbHeight));
        std::cout<<"\n gNB position:"<<Vector (30.5, 10.5, gNbHeight)<<std::endl;
    }
else if (gNbNum == 4)
{
        //#1
```

```
          apPositionAlloc->Add (Vector (10.5, 30.5, gNbHeight));
          std::cout<<"\n gNB position:"<<Vector (10.5, 30.5, gNbHeight)<<std::endl;
          //#2
          apPositionAlloc->Add (Vector (10.5, 10.5, gNbHeight));
          std::cout<<"\n gNB position:"<<Vector (10.5, 10.5, gNbHeight)<<std::endl;
          //#3
          apPositionAlloc->Add (Vector (30.5, 30.5, gNbHeight));
          std::cout<<"\n gNB position:"<<Vector (30.5, 30.5, gNbHeight)<<std::endl;
          //#4
          apPositionAlloc->Add (Vector (30.5, 10.5, gNbHeight));
          std::cout<<"\n gNB position:"<<Vector (30.5, 10.5, gNbHeight)<<std::endl;
  }
          else
  {
          cout << "None of gNb number configuration that was introduced is abailable. Exercice 2 only stands for 2 or 4 gNb";
  }



  //----------------------------------------------Users position allocation algorithm --------

  float Xposition = {0};
  float Yposition = {0};
  for (uint32_t i = 1; i <= ueNum; ++i)
  {
      Xposition = rand() % 40 + 1;
      Yposition = rand() % 40 + 1;
      staPositionAlloc->Add (Vector (Xposition,Yposition, ueHeight));
      std::cout<<"\n UEs position:"<<Vector (Xposition, Yposition, ueHeight)<<std::endl;

  }

```

```
mobility.SetMobilityModel ("ns3::
   ConstantPositionMobilityModel");
mobility.SetPositionAllocator (apPositionAlloc);
mobility.Install (gNbNodes);

mobility.SetPositionAllocator (staPositionAlloc);
mobility.Install (ueNodes);


// install mmWave net devices
NetDeviceContainer enbNetDev = mmWaveHelper->InstallEnbDevice
    (gNbNodes);
NetDeviceContainer ueNetDev = mmWaveHelper->InstallUeDevice (
   ueNodes);

// create the internet and install the IP stack on the UEs
// get SGW/PGW and create a single RemoteHost
Ptr<Node> pgw = epcHelper->GetPgwNode ();
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);

// connect a remoteHost to pgw. Setup routing too
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate
   ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (2500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.000)
   ));
NetDeviceContainer internetDevices = p2ph.Install (pgw,
   remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (
   internetDevices);
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting =
   ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<
   Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("
   7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);
internet.Install (ueNodes);
```

```
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address (
    NetDeviceContainer (ueNetDev));
// assign IP address to UEs, and install UDP downlink
    applications
uint16_t dlPort = 1234;
ApplicationContainer clientApps;
ApplicationContainer serverApps;

// Set the default gateway for the UEs
for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    Ptr<Ipv4StaticRouting> ueStaticRouting =
        ipv4RoutingHelper.GetStaticRouting (ueNodes.Get(j)->
        GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->
        GetUeDefaultGatewayAddress (), 1);
  }

UdpServerHelper dlPacketSinkHelper (dlPort);
serverApps.Add (dlPacketSinkHelper.Install (ueNodes.Get(0)));

for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    UdpClientHelper dlClient (ueIpIface.GetAddress (j),
        dlPort);
    dlClient.SetAttribute("PacketSize", UintegerValue(
        udpPacketSize));
    dlClient.SetAttribute ("MaxPackets", UintegerValue(0
        xFFFFFFFF));
    //dlClient.SetAttribute ("MaxPackets", UintegerValue
        (1000));

    if (udpFullBuffer)
      {
        double bitRate = 75000000; // 75 Mb/s will saturate
            the system of 20 MHz

        if (bandwidth > 20e6)
          {
            bitRate *=  bandwidth / 20e6;
          }
        udpInterval = static_cast<double> (udpPacketSize * 8)
            / bitRate ;
      }
```

```
      dlClient.SetAttribute ("Interval", TimeValue (Seconds(udpInterval)));
      clientApps.Add (dlClient.Install (remoteHost));

      Ptr<EpcTft> tft = Create<EpcTft> ();
      EpcTft::PacketFilter dlpf;
      dlpf.localPortStart = dlPort;
      dlpf.localPortEnd = dlPort;
      dlPort++;
      tft->Add (dlpf);

      enum EpsBearer::Qci q;
      q = EpsBearer::GBR_CONV_VOICE;
      EpsBearer bearer (q);
      mmWaveHelper->ActivateDedicatedEpsBearer(ueNetDev.Get(j), bearer, tft);
    }

  // start server and client apps
  serverApps.Start(Seconds(udpAppStartTime));
  clientApps.Start(Seconds(udpAppStartTime));
  serverApps.Stop(Seconds(simTime));
  clientApps.Stop(Seconds(simTime));

  // attach UEs to the closest eNB
  mmWaveHelper->AttachToClosestEnb (ueNetDev, enbNetDev);

  // enable the traces provided by the mmWave module
  //mmWaveHelper->EnableTraces();


  FlowMonitorHelper flowmonHelper;
  NodeContainer endpointNodes;
  endpointNodes.Add (remoteHost);
  endpointNodes.Add (ueNodes);

  Ptr<ns3::FlowMonitor> monitor = flowmonHelper.Install (endpointNodes);
  monitor->SetAttribute ("DelayBinWidth", DoubleValue (0.001));
  monitor->SetAttribute ("JitterBinWidth", DoubleValue (0.001));
  monitor->SetAttribute ("PacketSizeBinWidth", DoubleValue (20));


```

```
  Simulator::Stop (Seconds (simTime));
  Simulator::Run ();



  // Print per-flow statistics
  monitor->CheckForLostPackets ();
  Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmonHelper.GetClassifier ());
  FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats ();

  double averageFlowThroughput = 0.0;
  double averageFlowDelay = 0.0;

  std::ofstream outFile;
  std::string filename = outputDir + "/" + simTag;
  outFile.open (filename.c_str (), std::ofstream::out | std::ofstream::app);
  if (!outFile.is_open ())
    {
      NS_LOG_ERROR ("Can't open file " << filename);
      return 1;
    }
  outFile.setf (std::ios_base::fixed);

 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != stats.end (); ++i)
    {
      Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
      std::stringstream protoStream;
      protoStream << (uint16_t) t.protocol;
      if (t.protocol == 6)
        {
          protoStream.str ("TCP");
        }
      if (t.protocol == 17)
        {
          protoStream.str ("UDP");
        }
      outFile << "Flow " << i->first << " (" << t.sourceAddress << ":" << t.sourcePort << " -> " << t.destinationAddress << ":" << t.destinationPort << ") proto " << protoStream.str () << "\n";
```

```
    outFile << "  Tx Packets: " << i->second.txPackets << "\n";
    outFile << "  Tx Bytes:   " << i->second.txBytes << "\n";
    outFile << "  TxOffered:  " << i->second.txBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000  << " Mbps\n";
    outFile << "  Rx Bytes:   " << i->second.rxBytes << "\n";
    if (i->second.rxPackets > 0)
      {
        // Measure the duration of the flow from receiver's perspective
        //double rxDuration = i->second.timeLastRxPacket.GetSeconds () - i->second.timeFirstTxPacket.GetSeconds ();

        averageFlowThroughput += i->second.rxBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000;
        averageFlowDelay += 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets;

        std::cout<<"\n flow delay: " << 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets << " us" <<std::endl; //Sortida Retard mitj
        std::cout<<"\n flow throughput: " << i->second.rxBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000 << "mbps" <<std::endl; //Sortida Retard mitj

        outFile << "  Throughput: " << i->second.rxBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000 << " Mbps\n";
        outFile << "  Mean delay:  " << 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets << " ms\n";
        //outFile << "  Mean upt:  " << i->second.uptSum / i->second.rxPackets / 1000/1000 << " Mbps \n";
        outFile << "  Mean jitter:  " << 1000 * i->second.jitterSum.GetSeconds () / i->second.rxPackets  << " ms\n";
      }
    else
      {
        outFile << "  Throughput:  0 Mbps\n";
        outFile << "  Mean delay:  0 ms\n";
        outFile << "  Mean upt:  0  Mbps \n";
```

```
            outFile << "␣␣Mean␣jitter:␣0␣ms\n";
          }
        outFile << "␣␣Rx␣Packets:␣" << i->second.rxPackets << "\n";
    }

  outFile << "\n\n␣␣Mean␣flow␣throughput:␣" << averageFlowThroughput / stats.size() << "\n";

  std::cout<<"\n\n␣␣Mean␣flow␣throughput:␣" << averageFlowThroughput / stats.size() <<std::endl; // Sortida Throughput mitj

  outFile << "␣␣Mean␣flow␣delay:␣" << averageFlowDelay / stats.size () << "\n";

  std::cout<<"\n␣␣Mean␣flow␣delay:␣" << averageFlowDelay / stats.size () <<std::endl; //Sortida Retard mitj

  //--Print of the most important data for simulation--
  std::cout<<"\n␣␣-----------Simulation␣Configuration␣---------------";
  std::cout<<"\n\n␣␣Bandwidth:␣" << bandwidth <<std::endl;
  std::cout<<"\n␣␣Frecuency:␣" << frequency  <<std::endl;
  std::cout<<"\n\n␣␣Number␣of␣users␣(UES):␣" << ueNum  <<std::endl;
  std::cout<<"\n␣␣Number␣of␣base␣stations␣(gNb):␣" << gNbNum << std::endl;
  std::cout<<"\n\n␣␣-------------------------------------------------";

  outFile.close ();

  Ptr<UdpClient> clientApp = clientApps.Get(0)->GetObject<UdpClient>();
  Ptr<UdpServer> serverApp = serverApps.Get(0)->GetObject<UdpServer>();
  std::cout<<"\n␣Total␣UDP␣throughput␣(bps):"<<(serverApp->GetReceived()*udpPacketSize*8)/(simTime-udpAppStartTime)<<std::endl;

  Simulator::Destroy ();
  return 0;
}
```

C.1.4. Modified $cttc-3gpp-channel-nums$ code for up-link configurations

This code is the one used to obtain up-link simulations on the simulation scenarios.

```

/* -*-  Mode: C++; c-file-style: "gnu"; indent-tabs-mode:nil; -*- */
/*
 *   Copyright (c) 2017 Centre Tecnologic de Telecomunicacions de Catalunya (CTTC)
 *
 *   This program is free software; you can redistribute it and/or modify
 *   it under the terms of the GNU General Public License version 2 as
 *   published by the Free Software Foundation;
 *
 *   This program is distributed in the hope that it will be useful,
 *   but WITHOUT ANY WARRANTY; without even the implied warranty of
 *   MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.  See the
 *   GNU General Public License for more details.
 *
 *   You should have received a copy of the GNU General Public License
 *   along with this program; if not, write to the Free Software
 *   Foundation, Inc., 59 Temple Place, Suite 330, Boston, MA 02111-1307  USA
 *
 *
 *   Author:  Biljana Bojovic <bbojovic@cttc.es>

*/

#include "ns3/core-module.h"
#include "ns3/config-store.h"
#include "ns3/network-module.h"
#include "ns3/internet-module.h"
#include "ns3/internet-apps-module.h"
#include "ns3/applications-module.h"
#include "ns3/mobility-module.h"
#include "ns3/log.h"
```

```
#include "ns3/point-to-point-helper.h"
#include "ns3/flow-monitor-module.h"
#include "ns3/mmwave-helper.h"
#include "ns3/mmwave-point-to-point-epc-helper.h"
#include "ns3/ipv4-global-routing-helper.h"
#include "ns3/log.h"
//#include "<koolplot.h>
#include <string>
#include <cstdlib>
#include <ctime>
#include <iostream>

using namespace std;


/**
 * \file cttc-3gpp-channel-nums.cc
 * \ingroup examples
 * \brief Simple topology numerologies example.
 *
 * This example allows users to configure the numerology and test the end-to-end
 * performance for different numerologies. In the following figure we illustrate the simulation setup.
 *
 * For example, UDP interval can be configured by setting
 * "--udpInterval=0.001". The numerology can be toggled by the argument,
 * e.g. "--numerology=1". Additionally, in this example two arguments
 * are added "bandwidth" and "frequency". The modulation scheme of
 * this example is in test mode, and it is fixed to 28.
 *
 * By default, the program uses the 3GPP channel model, without shadowing and with
 * line of sight ('l') option. The program runs for 0.4 seconds and one single packet
 * is to be transmitted. The packet size can be configured by using  the
 * following parameter: "--packetSize=1000".
 *
 * This simulation prints the output to the terminal and also to the file which
```

```
 * is named by default "cttc-3gpp-channel-nums-fdm-output" and which is by
 * default placed in the root directory of the project.
 *
 * To run the simulation with the default configuration one shall run the
 * following in the command line:
 *
 * ./waf --run cttc-3gpp-channel-nums
 *
 */

using namespace ns3;

NS_LOG_COMPONENT_DEFINE ("3gppChannelNumerologiesExample");

static ns3::GlobalValue g_frequency("frequency",
                                    "The␣system␣frequency",
                                     ns3::DoubleValue(6e9),
                                     ns3::MakeDoubleChecker<double>(6e9,100e9));//!< Global variable used to configure the frequency. It is accessible as "--frequency" from CommandLine.

static ns3::GlobalValue g_bandwidth("bandwidth",
                                    "The␣system␣bandwidth",
                                     ns3::DoubleValue(100e6),
                                     ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure the bandwidth. It is accessible as "--bandwidth" from CommandLine.

static ns3::GlobalValue g_numerology ("numerology",
                                      "The␣default␣3GPP␣NR␣numerology␣to␣be␣used",
                                       ns3::UintegerValue (2),
```

```
                                    ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the numerology. It is accessible as "--numerology" from CommandLine.

static ns3::GlobalValue g_udpInterval ("udpInterval",
                                    "Udp interval for UDP application packet arrival, in seconds",
                                    ns3::DoubleValue (0.01), //ns3::DoubleValue (0.00075),
                                    ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure the UDP packet interval. It is accessible as "--udpInterval" from CommandLine.

static ns3::GlobalValue g_udpPacketSize ("udpPacketSize",
                                       "Udp packet size in bytes",
                                       ns3::UintegerValue (20000), //(20000),
                                       ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the UDP packet size. It is accessible as "--udpPacketSize" from CommandLine.

static ns3::GlobalValue g_udpRate ("udpFullBuffer",
                                 "Whether to set the full buffer traffic; if this parameter is set then the udpInterval parameter"
```

```
                                     "will be neglected.",
                                     ns3::BooleanValue (false),
                                     ns3::MakeBooleanChecker ()); //!< Global variable used to configure whether the traffic is the full buffer traffic. It is accessible as "--udpFullBuffer" from CommandLine.

//static ns3::GlobalValue g_singleUeTopology ("singleUeTopology",
                                            //"When true the example uses a single UE topology, //when false use topology with variable number of UEs"
                                            //"will be neglected.",
                                            //ns3::BooleanValue (false),
                                            //ns3::MakeBooleanChecker ()); //!< Global variable used //to configure whether topology is with single of //various number of UEs. It is accessible as //"--singleUeTopology" from CommandLine.

static ns3::GlobalValue g_useFixedMcs ("useFixedMcs",
                                       "Whether to use fixed mcs, normally used for testing purposes",
```

```
                                                 ns3::BooleanValue (false),
                                                 ns3::MakeBooleanChecker ()); //!< Global variable used to configure whether to use fixed MCS. It is accessible as "--useFixedMcs" from CommandLine.

static ns3::GlobalValue g_fixedMcs ("fixedMcs",
                                    "The MCS that will be used in this example",
                                    ns3::UintegerValue (28),
                                    ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure fixed MCS. It is accessible as "--fixedMcs" from CommandLine.

static ns3::GlobalValue g_gNbNum ("gNbNum",
                                  "The number of gNbs in multiple-ue topology",
                                  ns3::UintegerValue (2),
                                  ns3::MakeUintegerChecker<uint32_t>());//!< Global variable used to configure the number of gNbs in multi-UE topology. It is accessible as "--gNbNum" from CommandLine.

static ns3::GlobalValue g_ueNum ("ueNum",
                                 "The number of UEs in multiple-ue topology",
                                 ns3::UintegerValue (10),
                                 ns3::MakeUintegerChecker<uint32_t>()); //!< Global variable used to configure the number of UEs in multi-UE topology. It is accessible as "--
```

```
ueNumPergNb" from CommandLine.

static ns3::GlobalValue g_cellScan ("cellScan",
                                    "Use beam search method to determine beamforming vector, the default is long-term covariance matrix method"
                                    "true to use cell scanning method, false to use the default power method.",
                                    ns3::BooleanValue (true),
                                    ns3::MakeBooleanChecker()); //!< Global variable used to configure whether to use Beam Search of Long-Term Cov. matrix for beamforming. It is accessible as "--cellScan" from CommandLine.

static ns3::GlobalValue g_beamSearchAngleStep ("beamSearchAngleStep",
                                               "Beam search angle step for beam search method",
                                               ns3::DoubleValue (30),
                                               ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure beam search angle step in the case that beam search method is used. It is accessible
```

```
as "--beamSearchAngleStep" from CommandLine.

static ns3::GlobalValue g_txPower ("txPower",
                                   "Tx power",
                                   ns3::DoubleValue (4),
                                   ns3::MakeDoubleChecker<double>()); //!< Global variable used to configure gNb TX power. It is accessible as "--txPower" from CommandLine.

static ns3::GlobalValue g_simTag ("simTag",
                                  "tag to be appended to output filenames to distinguish simulation campaigns",
                                  ns3::StringValue ("test-toni-ex-2"),
                                  ns3::MakeStringChecker ()); //!< Global variable used to configure simulation output tag that helps distinguishing different simulation campaigns. It is accessible as "--simTag" from CommandLine.

static ns3::GlobalValue g_outputDir ("outputDir",
                                     "directory where to store simulation results",
                                     ns3::StringValue ("./"),
                                     ns3::MakeStringChecker ()); //!< Global variable used to configure simulation output folder. It is accessible as "--outputDir" from CommandLine.

int
```

```
main (int argc, char *argv[])
{

    CommandLine cmd;
    cmd.Parse (argc, argv);
    ConfigStore inputConfig;
    inputConfig.ConfigureDefaults ();
    // parse again so you can override input file default
       values via command line
    cmd.Parse (argc, argv);

  // enable logging or not
  bool logging = false;
  if(logging)
    {
      LogComponentEnable ("MmWave3gppPropagationLossModel",
         LOG_LEVEL_ALL);
      LogComponentEnable ("
         MmWave3gppBuildingsPropagationLossModel",
         LOG_LEVEL_ALL);
      LogComponentEnable ("MmWave3gppChannel", LOG_LEVEL_ALL);
      LogComponentEnable ("UdpClient", LOG_LEVEL_INFO);
      LogComponentEnable ("UdpServer", LOG_LEVEL_INFO);
      LogComponentEnable ("LtePdcp", LOG_LEVEL_INFO);

    }

  // set simulation time and mobility
  double simTime = 1; // seconds
  double udpAppStartTime = 0.4; //seconds
  //double speed = 1; // 1 m/s for walking UT.

  // parse the command line options
  BooleanValue booleanValue;
  StringValue stringValue;
  IntegerValue integerValue;
  UintegerValue uintegerValue;
  DoubleValue doubleValue;
  GlobalValue::GetValueByName("numerology", uintegerValue); //
     use optional NLOS equation
  uint16_t numerology = uintegerValue.Get();
  GlobalValue::GetValueByName("fixedMcs", uintegerValue); //
     use optional NLOS equation
  uint16_t fixedMcs = uintegerValue.Get();
```

```
GlobalValue::GetValueByName("gNbNum", uintegerValue); // use
    optional NLOS equation
uint16_t gNbNum = uintegerValue.Get();
GlobalValue::GetValueByName("ueNum", uintegerValue);
uint16_t ueNum = uintegerValue.Get();
GlobalValue::GetValueByName("udpInterval", doubleValue);
double udpInterval = doubleValue.Get();
GlobalValue::GetValueByName("udpPacketSize", uintegerValue);
uint32_t udpPacketSize = uintegerValue.Get();
GlobalValue::GetValueByName("frequency", doubleValue); //
double frequency = doubleValue.Get();
GlobalValue::GetValueByName("udpFullBuffer", booleanValue);
    //
bool udpFullBuffer = booleanValue.Get();
GlobalValue::GetValueByName("singleUeTopology", booleanValue)
    ; //
bool singleUeTopology = booleanValue.Get();
GlobalValue::GetValueByName("bandwidth", doubleValue); //
double bandwidth = doubleValue.Get();
GlobalValue::GetValueByName("cellScan", booleanValue); //
bool cellScan = booleanValue.Get();
GlobalValue::GetValueByName("useFixedMcs", booleanValue); //
bool useFixedMcs = booleanValue.Get();
GlobalValue::GetValueByName("beamSearchAngleStep",
    doubleValue); // use optional NLOS equation
double beamSearchAngleStep = doubleValue.Get();
GlobalValue::GetValueByName("txPower", doubleValue); // use
    optional NLOS equation
double txPower = doubleValue.Get();
GlobalValue::GetValueByName ("simTag", stringValue);
std::string simTag = stringValue.Get ();
GlobalValue::GetValueByName ("outputDir", stringValue);
std::string outputDir = stringValue.Get ();

// attributes that can be set for this channel model
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::
    Frequency", DoubleValue(frequency));
Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::
    ChannelCondition", StringValue("l"));

if (singleUeTopology)
  {
  //Config::SetDefault ("ns3::MmWave3gppPropagationLossModel
      ::Scenario", StringValue("UMi-StreetCanyon"));
```

```
      Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("RMa"));
    }
  else
    {
      Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Scenario", StringValue("InH-OfficeOpen"));
    }

  Config::SetDefault ("ns3::MmWave3gppPropagationLossModel::Shadowing", BooleanValue(false));

  Config::SetDefault ("ns3::MmWave3gppChannel::CellScan", BooleanValue(cellScan));
  Config::SetDefault ("ns3::MmWave3gppChannel::BeamSearchAngleStep", DoubleValue(beamSearchAngleStep));

  Config::SetDefault ("ns3::MmWavePhyMacCommon::CenterFreq", DoubleValue(frequency));
  Config::SetDefault ("ns3::MmWavePhyMacCommon::Bandwidth", DoubleValue(bandwidth));
  Config::SetDefault ("ns3::MmWavePhyMacCommon::Numerology", UintegerValue(numerology));

  Config::SetDefault ("ns3::LteRlcUm::MaxTxBufferSize", UintegerValue(999999999));

  Config::SetDefault("ns3::MmWaveMacSchedulerNs3::FixedMcsDl", BooleanValue (useFixedMcs));
  Config::SetDefault("ns3::MmWaveMacSchedulerNs3::McsDefaultDl", UintegerValue (fixedMcs));

  //Config::SetDefault("ns3::MmWaveUeNetDevice::AntennaNum", UintegerValue (4));
  //Config::SetDefault("ns3::MmWaveEnbNetDevice::AntennaNum", UintegerValue (16));

  Config::SetDefault("ns3::MmWaveEnbPhy::TxPower", DoubleValue(txPower));
  Config::SetDefault ("ns3::MmWavePhyMacCommon::MacSchedulerType",
  TypeIdValue (TypeId::LookupByName("ns3::MmWaveMacSchedulerTdmaRR"))); // scheduler ---------------------------------------------
```

```
// TdmaPF OfdmaRR

// setup the mmWave simulation
Ptr<MmWaveHelper> mmWaveHelper = CreateObject<MmWaveHelper> ();
mmWaveHelper->SetAttribute ("PathlossModel", StringValue ("ns3::MmWave3gppPropagationLossModel"));
mmWaveHelper->SetAttribute ("ChannelModel", StringValue ("ns3::MmWave3gppChannel"));

Ptr<MmWavePointToPointEpcHelper> epcHelper = CreateObject<MmWavePointToPointEpcHelper> ();
mmWaveHelper->SetEpcHelper (epcHelper);
mmWaveHelper->Initialize();

// create base stations and mobile terminals
NodeContainer gNbNodes;
NodeContainer ueNodes;
MobilityHelper mobility;

double gNbHeight = 1.5;
double ueHeight = 1.5;

gNbNodes.Create (gNbNum);
ueNodes.Create (ueNum);

Ptr<ListPositionAllocator> apPositionAlloc = CreateObject<ListPositionAllocator> ();
Ptr<ListPositionAllocator> staPositionAlloc = CreateObject<ListPositionAllocator> ();

//------------------Base stations creation-------------------------------------------------------------

if (gNbNum == 2)
    {
        //#1
        apPositionAlloc->Add (Vector (10.5, 30.5, gNbHeight));
        std::cout<<"\n gNB position:"<<Vector (10.5, 30.5, gNbHeight)<<std::endl;
```

```
        //#2
        apPositionAlloc->Add (Vector (30.5, 10.5, gNbHeight));
        std::cout<<"\n␣gNB␣position:"<<Vector (30.5, 10.5, gNbHeight)<<std::endl;
    }
else if (gNbNum == 4)
{
        //#1
        apPositionAlloc->Add (Vector (10.5, 30.5, gNbHeight));
        std::cout<<"\n␣gNB␣position:"<<Vector (10.5, 30.5, gNbHeight)<<std::endl;
        //#2
        apPositionAlloc->Add (Vector (10.5, 10.5, gNbHeight));
        std::cout<<"\n␣gNB␣position:"<<Vector (10.5, 10.5, gNbHeight)<<std::endl;
        //#3
        apPositionAlloc->Add (Vector (30.5, 30.5, gNbHeight));
        std::cout<<"\n␣gNB␣position:"<<Vector (30.5, 30.5, gNbHeight)<<std::endl;
        //#4
        apPositionAlloc->Add (Vector (30.5, 10.5, gNbHeight));
        std::cout<<"\n␣gNB␣position:"<<Vector (30.5, 10.5, gNbHeight)<<std::endl;
}
        else
{
        cout << "None␣of␣gNb␣number␣configuration␣that␣was␣introduced␣is␣abailable.␣Exercice␣2␣only␣stands␣for␣2␣or␣4␣gNb.";
}



//----------------------------------------------Users position allocation algorithm--------

float Xposition = {0};
float Yposition = {0};

for (uint32_t i = 1; i <= ueNum; ++i)
```

```
{
    Xposition = rand() % 40 + 1;
    Yposition = rand() % 40 + 1;
    staPositionAlloc->Add (Vector (Xposition,Yposition, ueHeight));
    std::cout<<"\n UEs position:"<<Vector (Xposition, Yposition, ueHeight)<<std::endl;

}



mobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
mobility.SetPositionAllocator (apPositionAlloc);
mobility.Install (gNbNodes);

mobility.SetPositionAllocator (staPositionAlloc);
mobility.Install (ueNodes);


// install mmWave net devices
NetDeviceContainer enbNetDev = mmWaveHelper->InstallEnbDevice (gNbNodes);
NetDeviceContainer ueNetDev = mmWaveHelper->InstallUeDevice (ueNodes);

// create the internet and install the IP stack on the UEs
// get SGW/PGW and create a single RemoteHost
Ptr<Node> pgw = epcHelper->GetPgwNode ();
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);

// connect a remoteHost to pgw. Setup routing too
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (2500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.000)));
NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
```

```
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting (remoteHost->GetObject<Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"), Ipv4Mask ("255.0.0.0"), 1);
internet.Install (ueNodes);
Ipv4InterfaceContainer ueIpIface;
ueIpIface = epcHelper->AssignUeIpv4Address (NetDeviceContainer (ueNetDev));
// assign IP address to UEs, and install UDP downlink applications
uint16_t dlPort = 1234;
ApplicationContainer clientApps;
ApplicationContainer serverApps;

// Set the default gateway for the UEs
for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    Ptr<Ipv4StaticRouting> ueStaticRouting = ipv4RoutingHelper.GetStaticRouting (ueNodes.Get(j)->GetObject<Ipv4> ());
    ueStaticRouting->SetDefaultRoute (epcHelper->GetUeDefaultGatewayAddress (), 1);
  }

UdpServerHelper dlPacketSinkHelper (dlPort);
//serverApps.Add (dlPacketSinkHelper.Install (ueNodes.Get(0))); //downlink to uplink ------------------------------------------------
serverApps.Add(dlPacketSinkHelper.Install(remoteHost)); // uplink

for (uint32_t j = 0; j < ueNodes.GetN(); ++j)
  {
    //UdpClientHelper dlClient (ueIpIface.GetAddress (j), dlPort); //downlink to uplink ---------------------------------------------
    UdpClientHelper dlClient (internetIpIfaces.GetAddress(1), dlPort);//uplink

```

```
    dlClient.SetAttribute("PacketSize", UintegerValue(
       udpPacketSize));
    dlClient.SetAttribute ("MaxPackets", UintegerValue(0
       xFFFFFFFF));
    //dlClient.SetAttribute ("MaxPackets", UintegerValue
       (1000));

    if (udpFullBuffer)
      {
        double bitRate = 75000000; // 75 Mb/s will saturate
           the system of 20 MHz

        if (bandwidth > 20e6)
          {
            bitRate *=  bandwidth / 20e6;
          }
        udpInterval = static_cast<double> (udpPacketSize * 8)
           / bitRate ;
      }
    dlClient.SetAttribute ("Interval", TimeValue (Seconds(
       udpInterval)));
    //clientApps.Add (dlClient.Install (remoteHost)); //
       downlink to uplink
       ---------------------------------------------
    clientApps.Add (dlClient.Install (ueNodes.Get(j))); //
       uplink

    Ptr<EpcTft> tft = Create<EpcTft> ();
    EpcTft::PacketFilter dlpf;
    //dlpf.localPortStart = dlPort; //downlink to uplink
       -----------------------------------------------
    dlpf.remotePortStart = dlPort;
    //dlpf.localPortEnd = dlPort; //downlink to uplink
       ------------------------------------------------
    dlpf.remotePortEnd = dlPort;
    dlPort++;
    tft->Add (dlpf);

    enum EpsBearer::Qci q;
    q = EpsBearer::GBR_CONV_VOICE;
    EpsBearer bearer (q);
    mmWaveHelper->ActivateDedicatedEpsBearer(ueNetDev.Get(j),
        bearer, tft);
  }

```

```
// start server and client apps
serverApps.Start(Seconds(udpAppStartTime));
clientApps.Start(Seconds(udpAppStartTime));
serverApps.Stop(Seconds(simTime));
clientApps.Stop(Seconds(simTime));

// attach UEs to the closest eNB
mmWaveHelper->AttachToClosestEnb (ueNetDev, enbNetDev);

// enable the traces provided by the mmWave module
//mmWaveHelper->EnableTraces();


FlowMonitorHelper flowmonHelper;
NodeContainer endpointNodes;
endpointNodes.Add (remoteHost);
endpointNodes.Add (ueNodes);

Ptr<ns3::FlowMonitor> monitor = flowmonHelper.Install (endpointNodes);
monitor->SetAttribute ("DelayBinWidth", DoubleValue (0.001));
monitor->SetAttribute ("JitterBinWidth", DoubleValue (0.001));
monitor->SetAttribute ("PacketSizeBinWidth", DoubleValue (20));


Simulator::Stop (Seconds (simTime));
Simulator::Run ();



// Print per-flow statistics
monitor->CheckForLostPackets ();
Ptr<Ipv4FlowClassifier> classifier = DynamicCast<Ipv4FlowClassifier> (flowmonHelper.GetClassifier ());
FlowMonitor::FlowStatsContainer stats = monitor->GetFlowStats ();

double averageFlowThroughput = 0.0;
double averageFlowDelay = 0.0;

std::ofstream outFile;
std::string filename = outputDir + "/" + simTag;
```

```
  outFile.open (filename.c_str (), std::ofstream::out | std::ofstream::app);
  if (!outFile.is_open ())
    {
      NS_LOG_ERROR ("Can't open file " << filename);
      return 1;
    }
  outFile.setf (std::ios_base::fixed);

 for (std::map<FlowId, FlowMonitor::FlowStats>::const_iterator i = stats.begin (); i != stats.end (); ++i)
    {
      Ipv4FlowClassifier::FiveTuple t = classifier->FindFlow (i->first);
      std::stringstream protoStream;
      protoStream << (uint16_t) t.protocol;
      if (t.protocol == 6)
        {
          protoStream.str ("TCP");
        }
      if (t.protocol == 17)
        {
          protoStream.str ("UDP");
        }
      outFile << "Flow " << i->first << " (" << t.sourceAddress << ":" << t.sourcePort << " -> " << t.destinationAddress << ":" << t.destinationPort << ") proto " << protoStream.str () << "\n";
      outFile << "  Tx Packets: " << i->second.txPackets << "\n";
      outFile << "  Tx Bytes:   " << i->second.txBytes << "\n";
      outFile << "  TxOffered:  " << i->second.txBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000  << " Mbps\n";
      outFile << "  Rx Bytes:   " << i->second.rxBytes << "\n";
      if (i->second.rxPackets > 0)
        {
          // Measure the duration of the flow from receiver's perspective
          //double rxDuration = i->second.timeLastRxPacket.GetSeconds () - i->second.timeFirstTxPacket.GetSeconds ();

          averageFlowThroughput += i->second.rxBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000;
```

```
        averageFlowDelay += 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets;

        std::cout<<"\n␣flow␣delay:␣" << 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets <<std::endl; //Sortida Retard mitj

        outFile << "␣␣Throughput:␣" << i->second.rxBytes * 8.0 / (simTime - udpAppStartTime) / 1000 / 1000 << "␣Mbps\n";
        outFile << "␣␣Mean␣delay:␣␣" << 1000 * i->second.delaySum.GetSeconds () / i->second.rxPackets << "␣ms\n";
        //outFile << "  Mean upt:  " << i->second.uptSum / i->second.rxPackets / 1000/1000 << " Mbps \n";
        outFile << "␣␣Mean␣jitter:␣␣" << 1000 * i->second.jitterSum.GetSeconds () / i->second.rxPackets  << "␣ms\n";
      }
    else
      {
        outFile << "␣␣Throughput:␣␣0␣Mbps\n";
        outFile << "␣␣Mean␣delay:␣␣0␣ms\n";
        outFile << "␣␣Mean␣upt:␣␣0␣␣Mbps␣\n";
        outFile << "␣␣Mean␣jitter:␣0␣ms\n";
      }
    outFile << "␣␣Rx␣Packets:␣" << i->second.rxPackets << "\n";
  }

outFile << "\n\n␣␣Mean␣flow␣throughput:␣" << averageFlowThroughput / stats.size() << "\n";

std::cout<<"\n\n␣␣Mean␣flow␣throughput:␣" << averageFlowThroughput / stats.size() <<std::endl; //Sortida Throughput mitj

outFile << "␣␣Mean␣flow␣delay:␣" << averageFlowDelay / stats.size () << "\n";

std::cout<<"\n␣␣Mean␣flow␣delay:␣" << averageFlowDelay / stats.size () <<std::endl; //Sortida Retard mitj

//--Print of the most important data for simulation--
```

```
  std::cout<<"\n  -----------Simulation Configuration ---------------";
  std::cout<<"\n\n  Bandwidth: " << bandwidth <<std::endl;
  std::cout<<"\n  Frecuency: " << frequency  <<std::endl;
  std::cout<<"\n\n  Number of users (UES): " << ueNum  <<std::endl;
  std::cout<<"\n  Number of base stations (gNb): " << gNbNum <<std::endl;
  std::cout<<"\n\n  --------------------------------------------------";

  outFile.close ();

  Ptr<UdpClient> clientApp = clientApps.Get(0)->GetObject<UdpClient>();
  Ptr<UdpServer> serverApp = serverApps.Get(0)->GetObject<UdpServer>();
  std::cout<<"\n Total UDP throughput (bps):"<<(serverApp->GetReceived()*udpPacketSize*8)/(simTime-udpAppStartTime)<<std::endl;

  Simulator::Destroy ();
  return 0;
}
```

APPENDIX D. GENERATED RESULTS DATA FILES

D.1. Output data file of results

Example of results output file of $cttc-3gpp-channel-nums$

D.1.1. cttc-3gpp-channel-nums output file.

```


Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 3 (1.0.0.2:49155 -> 7.0.0.4:1236) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.832344 Mbps
  Mean delay:  2.861307 ms
  Mean jitter:  0.029167 ms
  Rx Packets: 60
Flow 4 (1.0.0.2:49156 -> 7.0.0.5:1237) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833599 Mbps
```

```
  Mean delay:  1.967855 ms
  Mean jitter:  0.028571 ms
  Rx Packets: 60


  Mean flow throughput: 0.833310
  Mean flow delay: 2.173510
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 3 (1.0.0.2:49155 -> 7.0.0.4:1236) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.832344 Mbps
  Mean delay:  2.861307 ms
  Mean jitter:  0.029167 ms
  Rx Packets: 60
Flow 4 (1.0.0.2:49156 -> 7.0.0.5:1237) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833599 Mbps
  Mean delay:  1.967855 ms
  Mean jitter:  0.028571 ms
  Rx Packets: 60
```

```
  Mean flow throughput: 0.833310
  Mean flow delay: 2.173510
Flow 1 (1.0.0.2:49153 -> 7.0.0.2:1234) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 2 (1.0.0.2:49154 -> 7.0.0.3:1235) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833649 Mbps
  Mean delay:  1.932439 ms
  Mean jitter:  0.028869 ms
  Rx Packets: 60
Flow 3 (1.0.0.2:49155 -> 7.0.0.4:1236) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.832344 Mbps
  Mean delay:  2.861307 ms
  Mean jitter:  0.029167 ms
  Rx Packets: 60
Flow 4 (1.0.0.2:49156 -> 7.0.0.5:1237) proto UDP
  Tx Packets: 60
  Tx Bytes:   61680
  TxOffered:  0.822400 Mbps
  Rx Bytes:   61680
  Throughput: 0.833599 Mbps
  Mean delay:  1.967855 ms
  Mean jitter:  0.028571 ms
  Rx Packets: 60


  Mean flow throughput: 0.833310
  Mean flow delay: 2.173510
```