UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR
DE INGENIEROS DE TELECOMUNICACIÓN

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

TRABAJO FIN DE MÁSTER

DESIGN AND IMPLEMENTATION OF AN ABR VIDEO STREAMING SIMULATION MODULE FOR NS-3. ANALYSIS AND COMPARISON OF ABR VIDEO STREAMING ALGORITHMS OVER VARIOUS MOBILE NETWORK SCENARIOS.

XINXIN LIU

JUNIO 2021

ERICSSON

TRABAJO DE FIN DE MÁSTER

Título:	Diseño e implementación de un módulo de ABR video streaming para NS-3. Análisis y comparación de algoritmos de ABR video streaming sobre varios escenarios de redes móviles.
Título (inglés):	Design and implementation of an ABR video streaming simulation module for NS-3. Analysis and comparison of ABR video streaming algorithms over various mobile network scenarios.
Autor:	Xinxin Liu
Tutor:	Marcus Ihlar (Ericsson AB)
Ponente:	Carlos Mariano Lentisco Sanchez (ETSIT-UPM)
Departamento:	Departamento de Ingeniería de Sistemas Telemáticos

MIEMBROS DEL TRIBUNAL CALIFICADOR

Presidente:	—
Vocal:	—
Secretario:	—
Suplente:	—

FECHA DE LECTURA:

CALIFICACIÓN:

UNIVERSIDAD POLITÉCNICA DE MADRID

ESCUELA TÉCNICA SUPERIOR DE
INGENIEROS DE TELECOMUNICACIÓN

Departamento de Ingeniería de Sistemas Telemáticos

TRABAJO FIN DE MÁSTER

DESIGN AND IMPLEMENTATION OF AN ABR VIDEO STREAMING SIMULATION MODULE FOR NS-3. ANALYSIS AND COMPARISON OF ABR VIDEO STREAMING ALGORITHMS OVER VARIOUS MOBILE NETWORK SCENARIOS.

Xinxin Liu

Junio 2021

Resumen

El streaming de vídeo con tasa de bits adaptativa se está convirtiendo en la técnica más utilizada para las plataformas de vídeo en línea. Con la pandemia mundial *COVID-19*, el streaming de vídeo se ha convertido en una de las principales fuentes de entretenimiento durante los confinamientos. De hecho, más de la mitad de la cuota de tráfico de la red se utiliza hoy en día para streaming de vídeo [7].

El objetivo de este Trabajo Fín de Máster es construir un framework en *ns-3*, implementado en *C++*, para probar algoritmos de adaptación de vídeo y comparar algunas implementaciones sobre diferentes escenarios de red. El primer paso es estudiar *ns-3*, familiarizarse con algunos módulos de *ns-3* y construir varios escenarios de red *LTE*. El segundo paso es construir un módulo que pueda simular servidores y clientes de vídeo *ABR*, estudiar algunos enfoques de los algoritmos de adaptación de la tasa de bits de vídeo e implementar dichos algoritmos, incluyendo soluciones basadas en el ancho de banda, en el buffer y algoritmos híbridos. Por último, podemos comparar y evaluar el rendimiento de diferentes algoritmos *ABR* en escenarios con condiciones variables con diferentes métricas objetivas de *QoE*.

//// Resultados

Este proyecto se ha llevado a cabo con la cátedra Ericsson-UPM en software y sistemas.

Palabras clave: DASH, ABR, ns-3, streaming de video por HTTP, simulación, QoE

Abstract

Adaptive bitrate video streaming is becoming the most used technique for online video platforms. With the *COVID-19* worldwide pandemic, video streaming has become one of the primary sources of entertainment during the shutdown. In fact, more than half of the network traffic share today is used by video streaming [7].

The objective of this Master's Thesis is to build a framework in *ns-3*, implemented in *C++*, for testing video adaptation algorithms and to compare some implementations over different network scenarios. The first step is to study *ns-3*, familiarize with some *ns-3* modules, and build various LTE network scenarios. The second step is to build a module that can simulate *ABR* video servers and clients, study some approaches of video bitrate adaptation algorithms and implement those algorithms, including throughput based, buffer based and hybrid solutions. Finally we can compare and evaluate the performance of different *ABR* algorithms on scenarios with varying conditions with different objective *QoE* metrics.

//// Resultados

This project has been carried out with the Ericsson-UPM scholarship in software and systems.

Keywords: DASH, ABR, ns-3, HTTP video streaming, simulation, QoE

Acknowledgements

Contents

List of Figures

List of Tables

Glossary

3GPP - 3rd Generation Partnership Project

ABR - Adaptive BitRate

BOLA - Buffer Occupancy based Lyapunov Algorithm

CDN - Content Delivery Network

CPU - Central Processing Unit

DASH - Dynamic Adaptive Streaming over HTTP

DRM - Digital Rights Management

e-NodeB - enhanced Node B

EPC - Evolved Packet Core

EPS - Evolved Packet System

GSM - Global System for Mobile communications

HDS - HTTP Dynamic Streaming

HLS - HTTP Live Streaming

HSS - Home Subscriber Server

HTTP - HyperText Transfer Protocol

IEC - International Electrotechnical Commision

IETF - Internet Engineering Task Force

IIS - Internet Information Services

IP - Internet Protocol

ISO - International Organization for Standarization

ITU-T - International Telecomunication Union - Telecomunication standarization

LENA - LTE-EPC Network simulAtor

LTE - Long Term Evolution

MIMO - Multiple Input Multiple Output

MME - Mobility Management Entity

MMS - Multimedia Message Service

MPEG - Moving Picture Experts Group

MPD - Media Presentation Description

MSS - Microsoft Smooth Streaming

NAT - Network Address Translation

NR - New Radio

ns-3 - network simulator 3

OFDMA - Orthogonal Frequency Division Multiple Access

OSMF - Open Source Media Framework

PCRF - Policy Charging and Rule Function

PGW - Packet data network GateWay

QoE - Quality of Experience

QoS - Quality of Service

RB - Resource Block

RE - Resource Element

SC-FDMA - Single-Carrier Frequency Division Multiple Access

SGW - Serving GateWay

UE - User Equipment

UHD - Ultra High Definition

UMTS - Universal Mobile Telecomunications System

URL - Universal Resource Locators

XML - eXtensible Markup Language

Chapter 1 Introduction

1.1 Context

There is no doubt about the importance of online video streaming. According to Sandvine [7], in 2020, 57% of the global internet traffic was used by video streaming. Moreover, one of the key predictions made by Cisco in 2018 [8] stated that by year 2022, video traffic will make up 82% of all *IP* traffic.

Consequently, many challenges arise. Due to the growth in the number and diversity of connected video-capable devices, and the increasing bandwidth and higher quality content available, the client and the server need to adapt the video content to the network and the devices. The technique of taking account the varying network conditions and computing resources of the user device to choose the adequate quality level is denominated as *Adaptive BitRate (ABR)*. Adaptation may be performed by monitoring different parameters such as estimated bandwidth, client's buffer level, CPU load or screen size.

The *Dynamic Adaptive Streaming over HTTP (DASH)* is one of the standards that implements adaptive bitrate video streaming and was developed by the *Moving Picture Experts Group (MPEG)* [17]. *MPEG-DASH* enables provisioning and delivering media using existing *HTTP*-delivery networks supports dynamic adaptation with seamless switching. By using *HTTP*, the player will not have firewall problems. The quality selection relays on the client thus providing better scalability, and there is no need to have session at the server.

The *MPEG-DASH* standard was published in 2012 and revised in 2019 by the *International Organization for Standardization (ISO) / International Electrotechnical Commission (IEC)* as *MPEG-DASH ISO/IEC 23009-1:2019* [14]. In addition, the *3rd Generation Partnership Project (3GPP)* defines the use of *DASH* as the standard continuous delivering of multimedia content in mobile networks, specifically in 4G such as *LTE* and 5G networks.

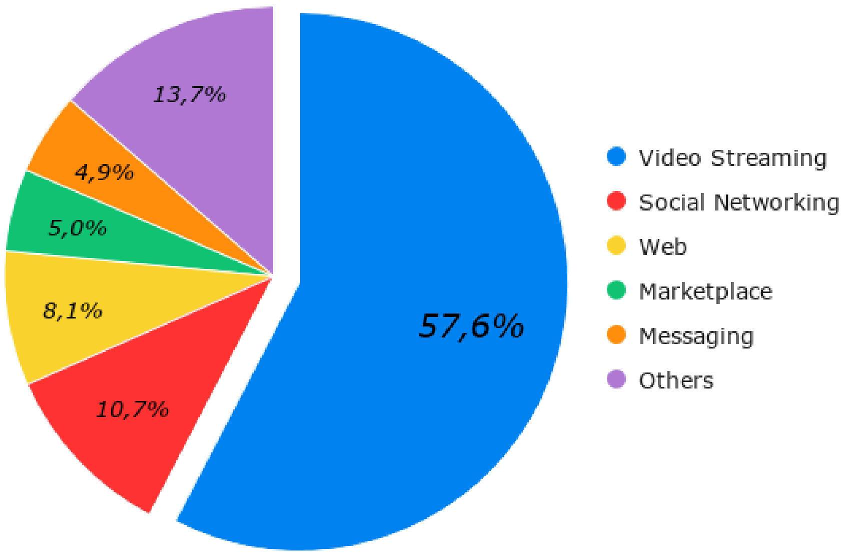

Figure 1.1: *Global application category total traffic share during COVID-19 lockdown. Source: Sandvine [7]*

DASH divides the media file into small chunks or segments. and defines the *Media Presentation Description (MPD)*, which is an XML manifest file that contains the *Universal Resource Locators (URL)* of the segments. Different qualities are defined as representations, the *MPD* contains information for each representation such as the codec, bandwidth, resolution or framerate.

However, the DASH Standard [14] only defines the data formats for the media reproduction and do not provide the adaptation algorithm. The *DASH Industry Forum* [10] provides an open source *MPEG-DASH* player implemented in *JavaScript* with different adaptation algorithms. Similarly, *hls.js* is an implementation of a *HTTP Live Streaming*[1] client.

The adaptation algorithms needs to be tested in different scenarios (they can be simulated) and be tweaked to provide the maximum perceived quality by the users. Also, there are algorithms that perform better in some specific scenarios and worse in others. The adaptation algorithm is the responsible for avoiding problems that may have a negative impact on the *Quality of Experience (QoE)*. One problem is that, the algorithm can overestimate the bandwidth and it would cause a pause in the reproduction because all the

[1]HTTP Live Streaming is a HTTP-based adaptive bitrate streaming protocol developed by Apple Inc. [4]

segments in the buffer are emptied. The algorithm can also underestimate the bandwidth, the video player requests media segments with inferior quality than the quality at which the bandwidth available of the network can allow. Lastly, the algorithm should avoid constant bitrate switches result of bandwidth fluctuations, and provide a smooth and seamless video watching experience.

The *ns-3* simulator is an open-source and extensible discrete-event network simulator. The extensible nature of this tool allows us to develop a new module for *ns-3* mimicking the behaviour of *ABR* clients and servers. With this new module, *ns-3* will be able to simulate diverse mobile network scenarios and test the performance of adaptation algorithms.

1.2 Objectives

The objectives of this thesis is to build a framework for testing *ABR* adaptation algorithms, and implement some adaptation algorithms and compare them in various mobile network scenarios with different objective *QoE* metrics. In order to achieve the proposed objectives, the following steps will be proposed:

1. Study and understand *ns-3* and basic modules such as the core module, the internet module, applications module, *LENA* module among others. Build basic *LTE* scenarios tweak radio parameters, and output results.
2. Design a new module in ns-3 that simulates behaviours of *ABR* clients and servers. Study and implement existing adaptation algorithms.
3. Implement objective *QoE* and *QoS* metrics. Build new *LTE* scenarios and compare the performances of the implemented adaptation algorithms.

1.3 Structure of the thesis

Chapter 1. Presents the context, the motivations and the objectives of this thesis.

Chapter 2. The State of the Art.

Chapter 3. dddd

Chapter 4. dddd

Chapter 5. dddd

Chapter 2 State of the Art

This chapter will introduce the main concepts and tools that will be used during the development of the project. The section 2.1 will explain the different methods of content distribution over *HTTP* and different types and implementations of adaptive streaming. The section 2.2 will make a introduction to the *DASH* standard, different types of adaptation algorithms and *QoE* and *QoS* metrics. The section 2.3 will describe basic architecture and fundamentals of 4G LTE, such as the radio interface, propagation loss model, fading model, antenna model, etc.

2.1 ABR Video Streaming

There are three ways of media delivery over *HTTP*. The first method is by **file download**, the media file is downloaded in its entirety in a local hard disk and then it can be played. The second method is called **progressive download**, this method is similar to the file download, but instead the download starts from the beginning and the media starts playing once enough data are playable. However, these two methods have disadvantages like waste of bandwidth or *DRM* issues and also requiring a reliable transmission. The last method is called **streaming**, contrary to the former two, the file itseft is not stored locally, smaller chunks of video are sent from the server and the client needs a data buffer to store the data that is being downloaded. The client plays the multimedia content from the buffer, and when the session is closed the data are deleted.

Streaming media also comes with some challenges. There are a lot of network variability and a big heterogeneity in video capable devices. Therefore, to overcome these shortcomings, *Adaptive bitrate streaming (ABR)* was created.

The basic idea of *Adaptive bitrate streaming* is to adapt the media content for the user by monitoring different parameters like estimated bandwidth, buffer level or *CPU load*, see

Figure 2.1. There are many propietary adaptive streaming solutions:

- ***Apple HTTP Live Streaming (HLS)***: *HTTP Live Streaming HLS* is an implementation of an *ABR* protocol over *HTTP* developed by Apple [4] as part of the QuickTime software and the mobile operating system *iOS*. *HLS* supports live streaming and video on demand. *HLS* is proposed in 2009 as a standard to the *IETF* [16].
- ***Microsoft Smooth Streaming (MSS)***: *Smooth Streaming* is part of *Internet Information Services (IIS) Media Services* for delivering media over *HTTP* [20]. Their *MSS* technology was used for several sports events such a the Beijing Summer Olympic Games in 2008 and the 2010 Winter Olymplics in Vancouver [21].
- ***Adobe HTTP Dynamic Streaming (HDS)***: *HTTP Dynamic Streaming* is the implementation of adaptive streaming by Adobe. *HDS* enables high-quality, network efficient HTTP streaming for media delivery that is tightly integrated with Adobe software [3]. The solution is based in using *Open Source Media Framework (OSMF)* and Adobe Flash Player.

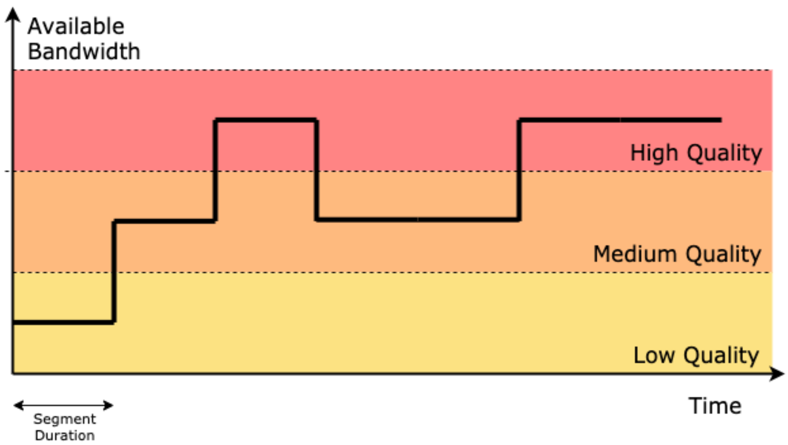

Figure 2.1: *Evolution of segment quality with time*

But there was no official standarization for adaptive video delivery over HTTP. For that reason, a new international stadard called *MPEG-DASH* was developed and published.

2.2 Dynamic Adaptive Streaming over HTTP

DASH was developed from January 2009 to March 2010 and published in April 2012. The most recent revision of the standarization was released in 2019 as *MPEG-DASH*

ISO/IEC 23009-1:2019 [14]. *Moving Picture Experts Group* from *ISO/IEC* and the *3GPP* collaborated on the *DASH* standard. The *3rd Generation Partnership Project* defined the use of *DASH* as the standard of digital media delivery in mobile networks (4G *LTE*, 5G) in [2].

The objective of *DASH* was to create a unique standard that replaces the propietary solutions from Microsoft, Apple and Adobe. Also, it will offer the interoperability and the convergence needed for the expansion of large-scale video streaming solutions. Also, the *DASH Industry Forum (DASH-IF)* was created to promote and help the expansion of *DASH*. Microsoft, Apple, Netflix, Qualcomm, Ericsson and Samsung are some of the companies members of the *DASH-IF*.

One of the biggest advantages of *DASH* is that the video streaming is over *HTTP* version 1.1 protocol (*HTTP/1.1*). The use of *HTTP* means that reusing existing internet infrastructure and media content distribution tecniques using *CDN (Content Delivery Networks)* can be done. Another convenience of using *DASH* is that due to using *HTTP* encapsulation, problems with passing through firewalls and the *Network Address Translation (NAT)* are not existent.

All the control of the media content delivery is located in the *DASH* client side. The standard does not define any web delivery mechanism nor the bitrate adaptation algorithm. What *DASH* does define in [14] are:

- ***The Media Presentation Description (MPD) File Format***: The *MPD* file uses the *eXtensible Markup Language (XML)* and contains the specifications of the media content and the *URL* of the segments in the *HTTP* video servers.
- **Segment format**: *DASH* defines the characteristics of the necessary codifications and the way that the media content is divided in small fragments called *segments*.

The Figure 2.2 presents a simple *DASH* architecture. The video and audio content are processed and stored on an *HTTP* server. To access the content, the client sends *HTTP* requests to the server. But first, the client needs to download the *MPD* file, normally through *HTTP*. The client then does the parsing of the *MPD*, extract information such as the duration of a segment, media types or resolutions. Finally, the *DASH* client chooses the adequate quality and starts the streaming of the content using *HTTP GET* request to fetch the segments.

The *DASH* client stores the segments in a buffer and consumes the content. It con-

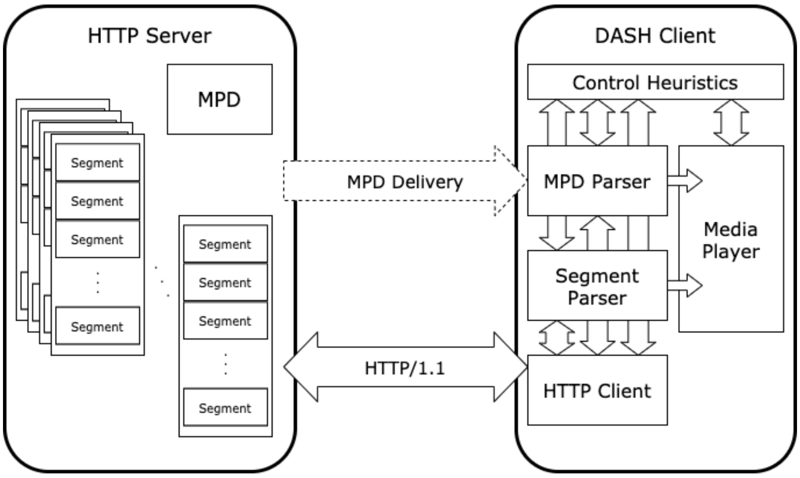

Figure 2.2: *DASH client-server architecture. Source: MPEG [25]*

tinues to fetch new segments and by monitoring network variables it will decide which quality (higher or lower bitrate) to request next to avoid problems like buffer underflow and maintain at least a set number of segments in the buffer.

2.2.1 MPD

The *MPD* file is an *XML* document that describes the characteristics of the different media components that composes the media content (e.g. video, audio, subtitles).

The structure of the *MPD* is hierarchical as illustrated in Figure 2.3. The media content is divided in a sequence of **periods**, each period has a starting time and a duration. In a period, the set of encoded versions of the media content is consistent, that is, the same bitrates, languages and so on.

Each period consists of one or multiple **adaptation sets**. A collection of interchangeable encoded versions of one or more media content components is referred to as an adaptation set. For instance, and adaptation set may contain the different bitrates of the video component of the same multimedia content and another adaptation set may contain the different bitrates of the audio component of the same multimedia content.

An adaptation set contains a set of **representations**. A representation describe an enconded alternative of the same media component, the alternatives can vary by bitrate, resolution, framerates, codec, sampling rate or other characteristics.

Each representation consists of one or multiple **segments**. A segment is the media

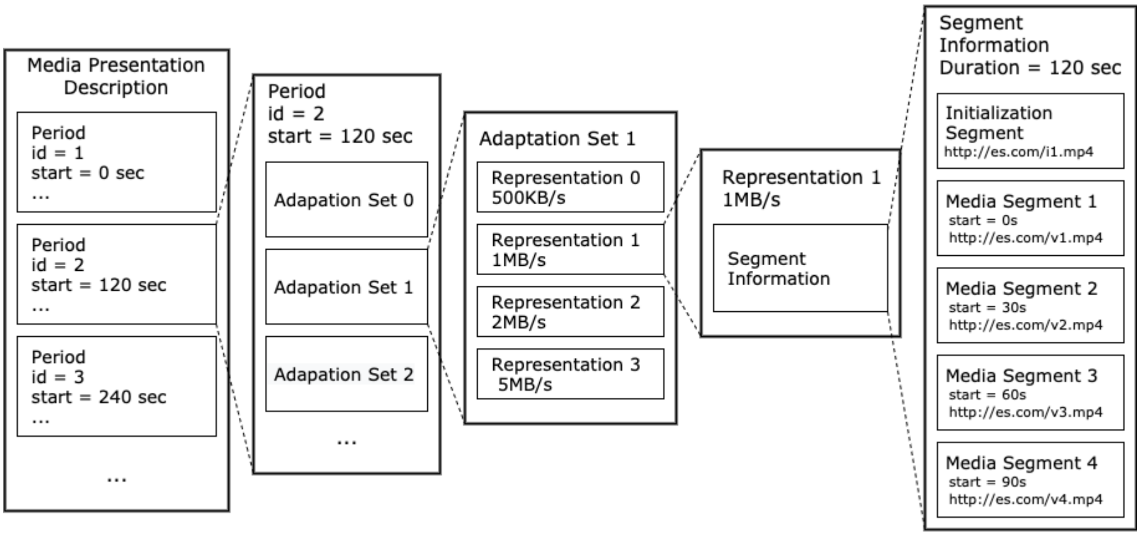

Figure 2.3: *The MPD hierarchical data model. Source: MPEG [25]*

stream chunks in temporal sequence. Each segment has a *URI*, the client will use this *URI* to make *HTTP GET* requests to the video server.

2.2.2 Adaptation Algorithms

In a video streaming service, there are a number of factors like the download bandwidth, delay or packet losses that can produced undesirable effects on the client such as buffer underflow, rebuffering and interruptions that lead to bad playback experience, thus, a bad Quality of Experience. To solve these problems, the ABR video streaming clients uses different adaptation algorithms to give a higher QoE.

An adaptation algorithm is a technique used in a multimedia streaming service to adjust the video quality in real-time according to different parameters. Some of the parameters are:

- **Client device**: The screen resolution, CPU capabilities, Buffer size, etc.
- **Network**: Type of access network (Mobile, Fixed), available bandwidth, etc.

The following subsections will explain different types of adaptation algorithms and the algorithms implemented for this thesis in *ns-3*.

2.2.2.1 Bandwidth throughput based algorithms

This group of algorithms uses estimations of bandwidth throughput as the main rule to select the qualities of the multimedia content for the client. The main difference between algorithms of this kind is the bandwidth estimation method and how the estimation relates to the qualities.

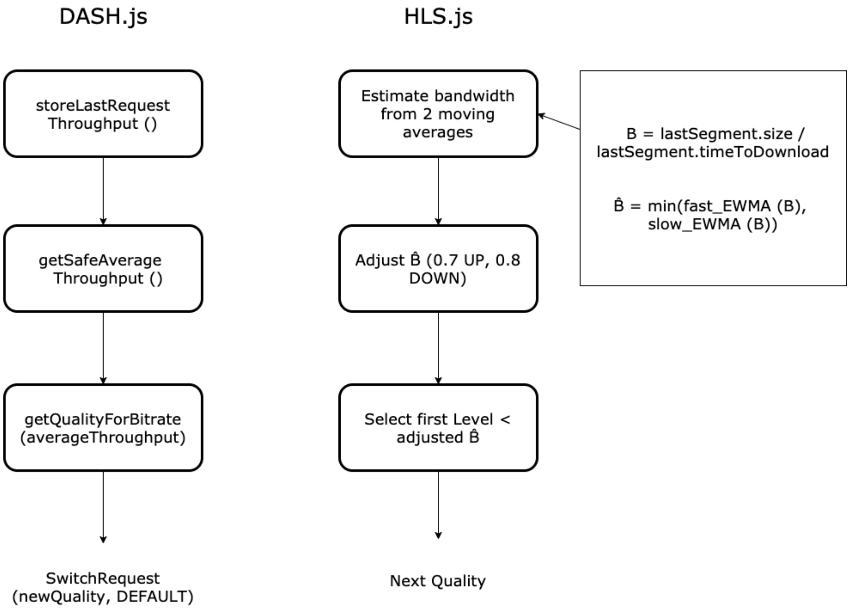

Figure 2.4: *Bandwidth based algorithms. Source: [13]*

- **HLS.js** [9]. The algorithm is called Bandwidth estimation.

  The algorithm processes two EWMA (Exponentially Weighted Moving Averages) and chooses the minimum of the two as the bandwidth estimation. Then the bandwidth estimation is multiplied by a factor to reduce oscilation. And finally it selects the first quality with a bitrate less than the adjusted bandwidth estimation.

- **DASH.js** [11]. The Throughput Rule.

  This algorithm is basically the same as the Bandwidth estimation from HLS.js. It computes the average throughput, and uses an safety factor to avoid oscilations. And then chooses the quality based on the safe average and creates a new *SwitchRequest*.

2.2.2.2 Buffer based algorithms

This group of algorithms uses buffer occupancy information to try to choose the highest level of bitrate for the multimedia content. These algorithms are usually used to avoid buffer underflow.

- **BOLA** [26]. Buffer Occupancy based Lyapunov Algorithm.

  The BOLA adaptation algorithm uses the Lyapunov optimization to make decisions. This is an utility theory and it is configurable with a tradeoff parameter to choose between rebuffering potential and bitrate maximization.

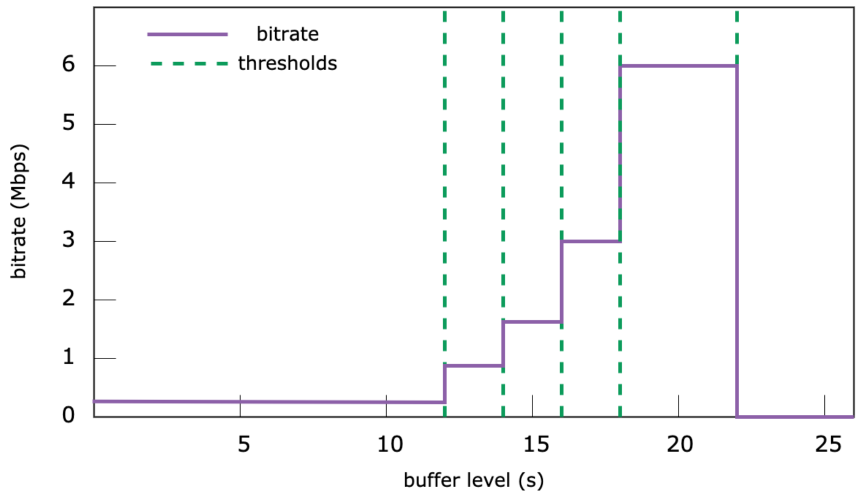

Figure 2.5: *BOLA's bitrate choice as function of buffer level. Source: [26]*

BOLA tries to maximize $V_n + \gamma S_n$. where:

- V_n is the bitrate utility.
- S_n is the playback smoothness.
- γ is the tradeoff weight parameter.

2.2.2.3 Control theory based or hybrid algorithms

This class of algorithms uses a combination of throughput estimation and buffer occupancy and tries to maximize the bitrate selection with decision-taking indicators calculated making use of control theory or stochastic optimal control equations.

2.2.3 QoS & QoE Metrics

The *Quality of Service (QoS)* is defined by the *ITU-T* in the document P.10/G.100 [19] as "The totality of characteristics of a telecommunications service that bear on its ability to satisfy stated and implied needs of the user of the service". And the *Quality of Experience (QoE)* is defined as "The degree of delight or annoyance of the user of an application or service".

The standard *ISO/IEC 23009* defines a list of parameters for *Quality of Service (QoS)* and *Quality of Experience (QoE)* for the adaptation algorithms to base on. There parameters is also used to evaluate the overall quality in the multimedia distribution service.

Some of the metrics defined in [2] and [14] are as follows:

- **Average Throughput**: This is a *QoE* metric that defines a list in which the average Throughput observed in the client during a measuring period.
- **Initial Playout Delay**: This is a *QoE* metric that represents the initial delay in the reproduction of the media content.
- **Representation Switch Events**: This is a *QoS* metric for measuring the number of representation switch events of the multimedia content.
- **Buffer Level**: This is a *QoS* metric that monitors the level of occupancy of the buffer during the reproduction of the multimedia content.

2.3 LTE

Long Term Evolution (LTE) was first introduced in 2008 in the Release 8 of the *3GPP* specification [1]. The objective of *LTE* was to migrate the *3GPP* systems into a optimized system based on packet switching (all *IP*), with greater bitrates, lower latency y multiple radio access technologies support.

2.3.1 History

The first mobile phone call was made in 1973 [15]. New generations of mobile networks are developed almost every decade. The first generation 1G launched years later, but it was only capable of doing voice calls. In 1991, the second generation 2G *(GSM)* of mobile networks was introduced. *GSM* provided improved wireless capabilities and introduced by the first time multimedia content with *Multimedia Message Service (MMS)*. But it was the third generation 3G, launched in 2001, that enabled new internet-driven services such as video conferencing and streaming. Later in 2009, the *LTE* 4G standard was commercially deployed. With theorical download bandwidth of almost 100Mbps made high-quality streaming into reality. 5G technologies improves in bandwidth even more and brings video streaming in *UHD* and more.

The consumption of multimedia content on mobile networks is becoming increasingly relevant with the rise of bandwidth and ease of access. This section will provide a brief introduction to the basic concepts of mobile networks, their architecture and fundamentals.

2.3.2 Architecture

The design of the *LTE* architecture was done from the ground up. The goal was to build a flat, all *IP* architecture using packet-switching, well structured (separation of control plane and user plane) and with few elements.

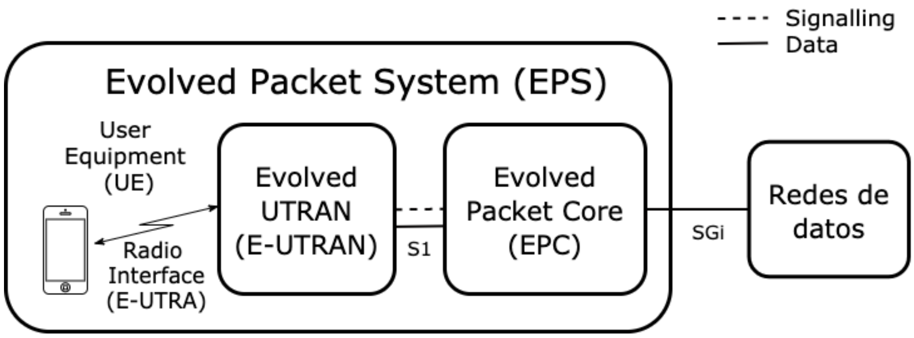

Figure 2.6: *LTE Architecture*

The *Evolved Packet System (EPS)* is constituted by the following elements:

- ***User Equipment (UE)***: An *UE* is any device used by an end user to communicate in a mobile network.
- ***Evolved UMTS Terrestial Radio Access Network (E-UTRAN)***: The only elements in the *E-UTRAN* are the *e-NodeB*. An ***enhanced Node B (e-NodeB)*** works as a base station and a controller.
- ***Evolved Packet Core (EPC)***: The *EPC* is made up of a network of gateways,

control servers, and databases linked by a *IP* backbone. The main elements of the *EPC* are:

- ***Mobility Management Entity (MME)***: The *MME* is a server used for managing the signalling of the operation.
- ***Serving Gateway (SGW)***: The *SGW* is the gateway used for communicating the access network *E-UTRAN* and the *PGW*.
- ***Packet Data Network Gateway (PGW)***: The *PGW* is the gateway for the traffic between the core network and external packet data networks.
- ***Home Subscriber Server (HSS)***: The *HSS* is a database containing information about the *EPC* network users.
- ***Policy Charging and Rule Function (PCRF)***: The *PCRF* is used for *QoS*, policy and charging management.

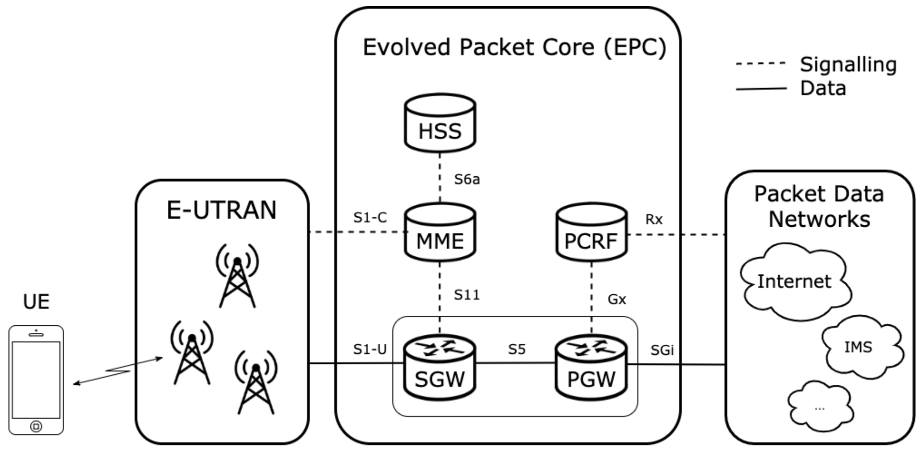

Figure 2.7: *Evolved Packet Core (EPC) Architecture*

2.3.3 OFDMA and SC-FDMA

The cellular communication systems needs to have a strategy for multiple access. In LTE, the *Orthogonal Frequency Division Multiple Access (OFDMA)* is used for downlink and the *Single- Carrier Frequency Division Multiple Access (SC-FDMA)* is used for uplink. Both are very similar, consisting in allocating each subscriber some portion of the subcarriers for certain amount of time.

In the Figure 2.8, a transmission structure of LTE is presented. The two dimentions of the plane are time and frequency. Two important concepts are defined as:

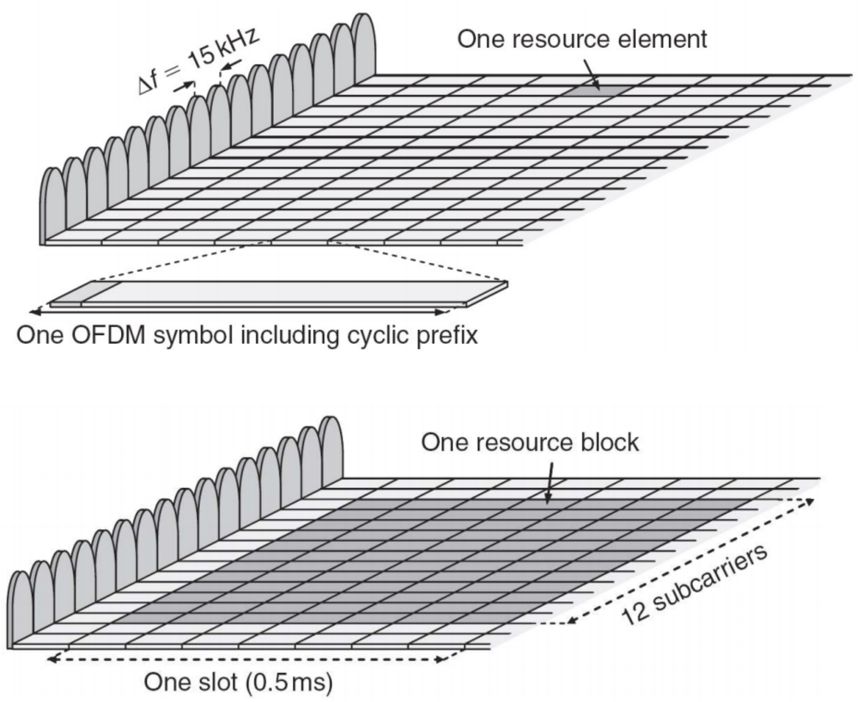

Figure 2.8: *LTE Time-Frequency Grid. Source: [28]*

- ***Resource Element (RE)***: A Resource Element is the basic element of resouce, it is defined as one subcarrier in a symbol period.
- ***Resource Block (RB)***: A Resource Block is composed by twelve subcarriers (180 kHz) in a time interval of 0.5 ms (7 OFDM symbols).

Users are assigned resources in resource blocks across a subframe, i.e., 12 subcarriers over $2 \times 7 = 14$ OFDM symbols for a total of 168 Resource Elements. Because some of the 168 resource components are utilized for various layer 1 and layer 2 control messages, not all of them can be used for data.

The number of Resource Blocks available for each channel bandwidth is given by the Table 2.1.

2.3.4 Wireless Fundamentals

Large-scale wireless networks, such as LTE, are fundamentally inefficient and prone to interference. Supporting mobility while also obtaining high levels of power efficiency, such

Bandwidth	1.4 MHz	3 MHz	5 MHz	10 MHz	15 MHz	20 MHz
Number of RBs available	6	15	25	50	75	100

Table 2.1: *Number of Resource Blocks against each channel bandwidth. Source: [27]*

as through directional antennas, can be really challenging. Base stations must be selectively installed but accommodate vast user populations in order to be cost-effective, resulting in a significant amount of self-interference. As a result, achieving high coverage, capacity, and dependability at low cost and used power is extremely difficult, if not impossible.

The following list highlights the main parameters affecting the received signal in a wireless system.

2.3.4.1 Propagation loss

The amount of transmitted power that actually reaches the receiver is the first visible difference between wired and wireless channels. The transmitted signal energy extends along a spherical wavefront if an isotropic antenna is utilized, hence the energy received at an antenna d distant is inversely proportional to the sphere surface area, $4\pi d^2$. However, in reality the propagation environment is not free space, we may also take into account other factors such as reflections.

2.3.4.2 Shadowing

Obstacles such as trees and buildings, as shown in Figure 2.9, may be situated between the transmitter and receiver, causing temporary signal degradation, whereas a temporary line-of-sight transmission path would result in abnormally high received power.

2.3.4.3 Fading loss

The fading effect is another aspect of wireless channels. Fading is generated by the receiving of multiple versions of the same signal (multipath), unlike path loss or shadowing, which are large-scale attenuation effects induced by distance or obstacles.

The reflections may arrive at very short intervals. For example, if there is local disper-

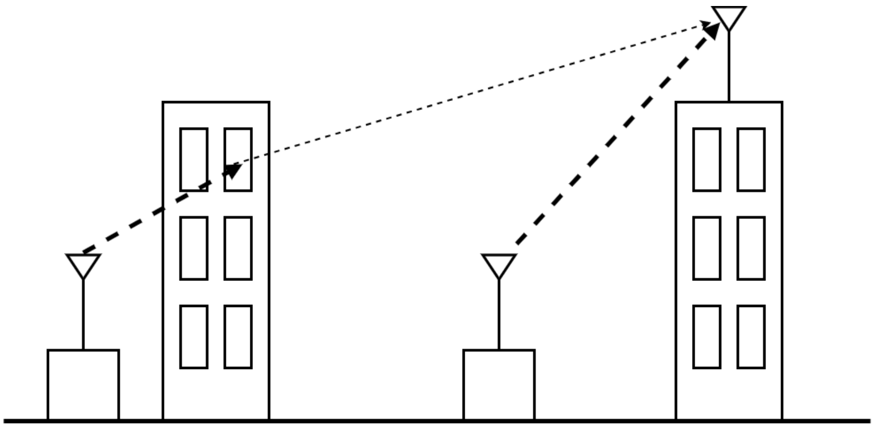

Figure 2.9: *Shadowing effect. Source: [22]*

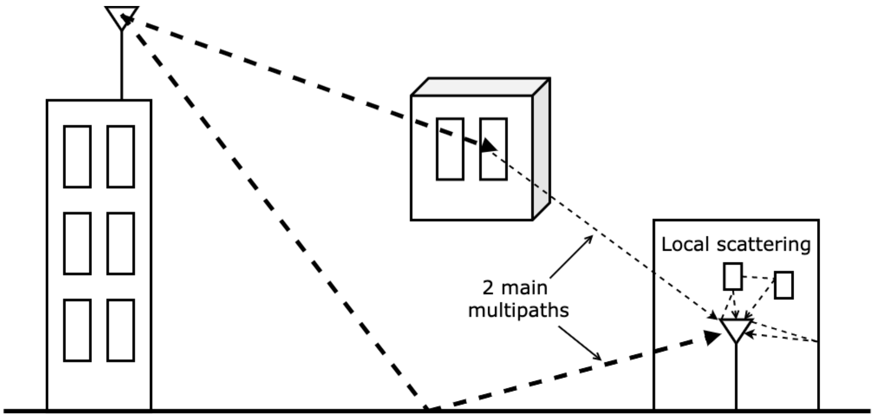

Figure 2.10: *Fading loss effect. Source: [22]*

sion around the receiver, or they may arrive at relatively longer intervals, for instance, if the transmitter and receiver are on multiple pathways. Figure 2.10 illustrates this.

2.3.5 Antennas & MIMO

An antenna is a device that uses electromagnetic waves to transmit or receive information. The transmitting antenna turns electrical currents into electromagnetic waves, and vice versa (receiving antenna).

Multiple Input, Multiple Output (MIMO) is a technique for increasing the capacity of a radio link by employing multiple transmitting and receiving antennas to take advantage of multipath propagation. MIMO has become a key component of wireless communication technologies such as LTE.

There are also special cases of MIMO:

- ***Multiple-input single-output (MISO)***: When there are multiple transmitting antennas and a single antenna.
- ***Single-input multiple-output (SIMO)***: When the transmitter has a single antenna and there are multiple receiving antennas.
- ***Single-input single-output (SISO)***: SISO is a radio system in which neither the transmitter nor the receiver has multiple antennas.

Another special type of MIMO is called *Multi User-Multiple Input Multiple Output*. Single-user SU-higher MIMO's per-user throughput is better suited to more sophisticated user devices with more antennas, whereas MU-MIMO is more practical for low-complexity mobile phones with a small number of reception antennas.

2.3.6 Physical Layer

LTE enb phy

UM buffer size Earfcn

Chapter 3 | Network Simulator 3

The *ns-3* simulator is an open, extensible discrete-event network simulator designed primarily for educational and network research purposes [23].

In summary, *ns-3* provides models of how packet data networks work and operate, as well as a simulation engine that allows users to run simulation experiments. To do research that are more difficult or impossible to do with real systems, to examine system behavior in a highly controlled, reproducible setting, and to understand about how networks work.

ns-3 is a collection of modules that can be used together as well as with other software libraries. This tool works mainly at the command line on *Linux* or *MacOS* and with *C++* and *Python* programming languages and development tools.

3.1 Getting Started

The prerequisites for the *ns-3* release version 3.32 are the following tools:

Prerequisite	Package/version
C++ compiler	`clang++` or `g++` (`g++` version 4.9 or greater)
Python	`python3` version >= 3.5
Git	any recent version
tar	any recent version
bunzip2	any recent version

Table 3.1: *Prerequisites for ns-3*

Start by downloading the source archive from nsnam or gitlab. Then build *ns-3* with `build.py`:

```
# Download from nsnam
$ cd
$ mkdir workspace
$ cd workspace
$ wget https://www.nsnam.org/release/ns-allinone-3.32.tar.bz2
$ tar xjf ns-allinone-3.32.tar.bz2
$ cd ns-allinone-3.32
# Building ns-3
$ ./build.py --enable-examples --enable-tests
# Running a script
# Create or copy a script to the scratch directory
$ cp examples/tutorial/first.cc scratch/myfirst.cc
$ ./waf --run scratch/myfirst
```

Listing 3.1: *Download and installation of ns-3*

3.2 ns-3 Concepts

This section will go over several networking concepts that have a specific meaning in ns-3.

Node: A `Node` in *ns-3* is the basic computing device abstraction. The `Node` class has methods for managing computing device representations in simulations.

Application: A *ns-3* application run on *ns-3* `Nodes`. An `Applicacion` is the basic abstraction for a user program that generates some simulated activity. The `Application` class provides functions for controlling the representations of the simulated version of user-level applications.

Channel: A `Channel` in *ns-3* is an abstraction of the basic communication subnetwork in which `Nodes` are connected in. It can be as simple as a wire or as complicated as a large Ethernet switch.

Net Device: A `NetDevice` in *ns-3* simulates a *Network Interface Card (NIC)* and the software controlling the *NIC*. A `NetDevice` is installed in a `Node` to allow it to communicate over `Channels` with other `Nodes` in the simulation.

Helpers: Helper objects are created to make some commun tasks easier. Such as connecting `NetDevices` to `Nodes`, `NetDevices` to `Channels`, assigning IP addresses, etc.

3.3 Logging Module

Message logging is a basic feature for large softwares, and *ns-3* is no different. *ns-3* offer a complete module for message logging with configurable verbosity levels. This means that logging functions of specific components can be enabled and other can be disabled completely.

There are different levels of log messages of ascending verbosity defined in *ns-3*:

- **`LOG_ERROR`**: For error messages (associated function: `NS_LOG_ERROR`).
- **`LOG_WARN`**: For warning messages (associated function: `NS_LOG_WARN`).
- **`LOG_DEBUG`**: For relatively rare, ad-hoc debugging messages (associated function: `NS_LOG_DEBUG`).
- **`LOG_INFO`**: For informational messages about program progress (associated function: `NS_LOG_INFO`).
- **`LOG_FUNCTION`**: For messages describing each function called (two associated function: `NS_LOG_FUNCTION` used for member functions, and `NS_LOG_FUNCTION_NOARGS`, used for static functions)).
- **`LOG_LOGIC`**: For messages describing logical flow within a function (associated function: `NS_LOG_LOGIC`).
- **`LOG_ALL`**: Log everything mentioned above (no associated function).

To enable all logs, it is as simple as modifying a shell variable. In the next example the logging for the class `UdpEchoClientApplication` and `UdpEchoServerApplication` is enabled with all levels, the time and the function prefixes:

```
$ export 'NS_LOG=UdpEchoClientApplication=level_all|prefix_func|
    prefix_time:UdpEchoServerApplication=level_all|prefix_func|
    prefix_time'
```

Listing 3.2: *Enabling logging in ns-3*

To disable logging simply type:

```
$ export NS_LOG=
```

Listing 3.3: Disabling logging in ns-3

For more information with the logging modure see [23].

3.4 Command Line Arguments

There are local and global variables that can be changed in the command line without editing the scripts. An example of a command could be like this:

```
# To check help
$ ./waf --run "scratch/myfirst --PrintHelp"
# To check variables for PointToPointNetDevice
$ ./waf --run "scratch/myfirst --PrintAttributes=ns3::
    PointToPointNetDevice"
# We set the Datarate to 5Mbps
$ ./waf --run "scratch/myfirst --ns3::PointToPointNetDevice::DataRate=5
    Mbps"
```

Listing 3.4: Command line arguments

3.5 Tracing System

The main goal of the simulations is to extract and generate output, and *ns-3* offers two mechanisms for this. Also, since *ns-3* is a C++ software, using `std::cout` for output is also available.

3.5.1 ASCII Tracing

ns-3 includes helper function that encapsulates the low-level tracing system and guides you through the technicalities of establishing some simple packet traces. If you enable this feature, the output will be in ASCII files, hence the name.

To enable ASCII Tracing, right before the call to `Simulator::Run ()`, add the following lines of code:

```
AsciiTraceHelper ascii;
pointToPoint.EnableAsciiAll (ascii.CreateFileStream ("out.tr"));
```

Listing 3.5: *ASCII Tracing*

This will generate the output from `pointToPoint` to a file named `out.tr`.

3.5.2 PCAP Tracing

The ns-3 device helpers can also create `.pcap` trace files. The *pcap* file contains the packets captured during the simulation. *Wireshark* or *tcpdump* are programs capable of reading and visualizing *pcap* files.

To enable *pcap* tracing simply add:

```
pointToPoint.EnablePcapAll ("myfirst");
```

Listing 3.6: *PCAP Tracing*

This will create various `.pcap` files in the format `"myfirst-0-0.pcap"`, meaning the trace file for node 0 and device 0.

3.6 ns-3 Modules

REM

MIMO

LTE enb phy

UM buffer size

TCP new reno?

Lossmodel

Fading loss model

Earfcn

Antenna model

Building

Chapter 4 | ABR Module for ns-3

Chapter 5 | Simulations Scenarios and Results

Chapter 6 | Conclusions and Future Work

Bibliography

[1] 3GPP. 3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Technical Specifications and Technical Reports for a GERAN-based 3GPP system (Release 8). Technical Report 3GPP TS 41.101, 2009.

[2] 3GPP. *Transparent end-to-end Packet-switched Streaming Service (PSS); Progressive Download and Dynamic Adaptive Streaming over HTTP (3GP-DASH)*. 3GPP TS 26.247 v16.4.1, 2020.

[3] Adobe. Live Streaming. `https://www.adobe.com/es/products/hds-dynamic-streaming.html`.

[4] Apple. HTTP Live Streaming. `https://developer.apple.com/streaming`.

[5] Benny Bing. *Next-generation video coding and streaming*. Wiley, 1st edition, 2015.

[6] Julián Cabrera. *Apuntes de Sistemas y Servicios de Multimedia (MUIT)*. Universidad Politécnica de Madrid, 2020.

[7] Lyn Cantor. The global internet phenomena report covid-19 spotlight. Technical report, Sandvine, 2020.

[8] Cisco. Cisco predicts more ip traffic in the next five years than in the history of the internet. `https://newsroom.cisco.com/press-release-content?type=webcontent&articleId=1955935`, 11 2018.

[9] Dailymotion. HLS.js. `https://github.com/dailymotion/hls.js`.

[10] DASH-IF. DASH Industry Forum. `https://dashif.org/`.

[11] DASH-IF. dash.js. `https://github.com/Dash-Industry-Forum/dash.js`.

[12] Universidad Politécnica de Valencia. Vídeo adaptativo a través de MPEG-DASH. `https://www.comm.upv.es/es/dash/`.

[13] Streaming Media East. Adapting your player logic to your use case: how to use ABR to your advantage. `https://es.slideshare.net/EricaBeavers/abr-algorithms-2017-streaming-media-east`, 2017.

[14] International Organization for Standardization. *Information technology — Dynamic adaptive streaming over HTTP (DASH) - Part 1: Media presentation description and segment formats.* International Organization for Standardization, ISO/IEC 23009-1:2019(E) edition, 2019.

[15] William. E. Gibson. First Cellular Phone Call Was Made 45 Years Ago. `https://www.aarp.org/politics-society/history/info-2018/first-cell-phone-call.html`, 2018.

[16] IETF. RFC 8216 - HTTP Live Streaming - IETF Tools. `https://tools.ietf.org/html/rfc8216`, August 2017.

[17] ISO. MEPG-DASH. `http://www.iso.org/iso/home/standards.htm`.

[18] ITU-T. *Recommendation E.800 : Definitions of terms related to quality of service.* ITU, 2017.

[19] ITU-T. *Recommendation P.10/G.100: Vocabulary for performance, quality of service and quality of experience.* ITU, 2017.

[20] Microsoft. Silverlight. `https://www.microsoft.com/silverlight/smoothstreaming/`.

[21] Christopher Mueller. Microsoft smooth streaming. `https://bitmovin.com/microsoft-smooth-streaming-mss/`, 2015.

[22] Rias Muhamed, Arunabha Ghosh, Jun Zhang, and Jeffrey G Andrews. *Fundamentals of LTE.* Prentice Hall Communications Engineering and Emerging Technologies Series from Ted Rappaport. Pearson, 2010.

[23] ns 3. A Discrete-Event Network Simulator. `https://www.nsnam.org/`.

[24] Miguel Ángel Aguayo Ortuño. Contribución a los mecanismos de adaptación dinámica para servicios de distribución multimedia sobre redes móviles. December 2020.

[25] Iraj Sodagar. White paper on MPEG-DASH Standard. MPEG-DASH: The Standard for Multimedia Streaming Over Internet. Technical report, ISO/IEC, 2012.

[26] Kevin Spiteri, Rahul Urgaonkar, and Ramesh K. Sitaraman. Bola: Near-optimal bitrate adaptation for online videos. In *IEEE INFOCOM 2016 - The 35th Annual IEEE International Conference on Computer Communications*, pages 1–9, 2016.

[27] Techplayon. LTE FDD System Capacity and Throughput Calculation. `https://www.techplayon.com/lte-fdd-system-capacity-and-throughput-calculation/`.

[28] Luis Mendo Tomas. *Apuntes de Comunicaciones Móviles (GITST).* Universidad Politécnica de Madrid, 2018.

[29] Hubregt J. Visser. *Antenna theory and applications.* 2012.

Appendix A | Impact

A.1 Social Impact

A.2 Economic Impact

A.3 Ambiental Impact

A.4 Ethic Impact

Appendix B | Budget