FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Edge computing for VRU protection: empirical evaluation of 5G and 802.11 communication technologies

Afonso Azevedo



Mestrado Integrado em Engenharia Informática e Computação

Supervisor: Ana Aguiar

June 29, 2021

Edge computing for VRU protection: empirical evaluation of 5G and 802.11 communication technologies

Afonso Azevedo

Mestrado Integrado em Engenharia Informática e Computação

Approved in oral examination by the committee:

Chair: Prof. Rosaldo Rossetti
External Examiner: Prof. Andreas Kassler
Supervisor: Prof. Ana Aguiar

June 29, 2021

Abstract

Currently, intelligent self-driving cars have started to become more present on the road and commercially available to the public. To operate correctly and not be prone to collisions, these vehicles require mechanisms for timely detection of objects and vulnerable road users (VRUs), i.e. pedestrians and cyclists. The vehicles incorporate sensors such as radar, infrared and video. These sensors generate raw data which is critical for the autonomous driving process of the vehicle. However, the data must be transformed before it can be used in any decision making. The processing of said data is usually time and computationally expensive. Given this context, the objective consists of reducing the time needed for processing the vehicle sensor data to detect objects. By reducing this delay, the vehicle has more time to react to a situation on the road, allowing it to take measures earlier and improve the users' and the VRUs' safety.

This work aims to contribute to such a system by using high throughput network technologies (5G and 802.11ac), paired with edge-computing stations equipped with high-performance GPUs. The performance difference between the edge station's hardware and the commonly used hardware in self-driving vehicles could justify an offloading approach for video processing; if the added communication delay is lower than the performance gain. By following the offloading approach a means of communication must be defined between the vehicle or VRU and the edge computing station. However, nowadays multiple communication technologies are usually available in most situations; and the best technology changes depending on the situation.

The goal of this thesis is to design and implement a prototype of such a system, which includes a decision-making module to choose between the two different communication interfaces. This module uses previous records of registered throughput to predict the throughput in a given area. The module can change its decision depending on the situation at hand, more specifically, it can take in consideration its position and the data associated with said location. For this data to be available, a network monitoring system must exist and as such is part of the system design.

This work designs, implements and tests a network monitoring system, leading to the collection of data used by the decision-making module. This module was used in scenarios that aim to simulate the video processing offloading that would happen in self-driving vehicles. Its performance is also discussed in this work.

Keywords: Self-driving, edge-computing, networks, offloading, decision-making, 5G

CCS→Networks→Network performance evaluation→Network experimentation

Acknowledgements

First and foremost, I thank my parents for all the love, wisdom and support, without them the man creator of this work would not be possible. Although I may not say it very often I love you very dearly and value you immensely.

To my brother Rodrigo, for being there for me when I need him most.

I also thank my loving Sandra for all the love and support over the years, for helping me grow up and become a better man.

To dear Susana, for all the help, laughs, "disagreements", patience and support over this last 5 years. For the unwavering support during experiments and the fun chats in the lab.

To João and Gonçalo, my friends, for all the help, laughs and support over this five year journey, I learned a lot with you and without you, college wouldn't have been the same.

Also to Akil, João and Hugo, for all the support, knowledge and laughs we shared while working in the lab and doing our experiments.

I also thank the Instituto de Telecomunicações (UIDB/50008/2020) for having provided the needed environment and equipment for the development of my work.

And last but certainly not least, to my supervisor Professor Ana Aguiar for all the support, guidance, availability and patience that helped me develop and achieve a successful work.

Afonso Azevedo

"The way is lit. The path is clear.
We require only the strength to follow it."

The Ancestor

Contents

1 Introduction **1**
- 1.1 Context . . . 1
- 1.2 Motivation . . . 3
- 1.3 Objectives . . . 4
- 1.4 Document Structure . . . 4

2 Current approaches to Vehicular Communication **7**
- 2.1 Background on Relevant Technologies . . . 7
  - 2.1.1 Protocol in current use . . . 7
  - 2.1.2 5G and 4G Networks . . . 8
  - 2.1.3 Vehicular communication . . . 9
- 2.2 Related Work . . . 10
  - 2.2.1 Edge computing . . . 10
  - 2.2.2 Network Performance Metrics . . . 11
  - 2.2.3 Throughput prediction . . . 13
  - 2.2.4 Video Streaming . . . 15

3 Heterogeneous Wireless Access to Edge Computing Resources **17**
- 3.1 Defining the problem . . . 17
- 3.2 Use Case and System Design . . . 19
  - 3.2.1 Decision Making module . . . 20
  - 3.2.2 Network Monitoring module . . . 21
  - 3.2.3 Multi-RAT Forwarder Module . . . 22
  - 3.2.4 Edge Node . . . 23
  - 3.2.5 Minimal functionality . . . 23
- 3.3 Methodology . . . 24
- 3.4 Tools . . . 24
  - 3.4.1 JRTPLIB . . . 24
  - 3.4.2 Consumer and Producer . . . 25
  - 3.4.3 TEMS Pocket . . . 25

4 System Implementation **27**
- 4.1 Nomad Node . . . 27
  - 4.1.1 Central Manager . . . 27
- 4.2 5G Access Modem . . . 33
  - 4.2.1 Hardware and OS . . . 33
  - 4.2.2 App description - UI Component . . . 33
  - 4.2.3 App description - Communication with host . . . 34

4.3 Edge Station . . . 35
4.3.1 RTP and RTCP messages . . . 35
4.3.2 Hardware and OS . . . 38
4.3.3 Communications and FEUP network . . . 38
4.3.4 Decision making module . . . 38
4.4 History Station . . . 39
4.4.1 Main function . . . 39
4.4.2 Endpoints . . . 39
4.4.3 Database . . . 39
4.4.4 Hosting . . . 39
4.4.5 Populating the database . . . 40
4.5 Setting up the RATs . . . 40
4.5.1 WiFi Access Point . . . 40
4.5.2 5G gNodeB . . . 42

5 Experimental Setup **43**
5.1 WiFi Data Collection Experiment . . . 43
5.2 5G Data Collection Experiment . . . 46
5.3 Main Experiment . . . 47

6 Results and discussion **53**
6.1 Functional Validation . . . 53
6.2 Quality of Received Video . . . 54
6.3 Decision Making Validation . . . 55

7 Final Remarks **59**
7.1 Conclusion . . . 59
7.2 Future Work . . . 59
7.3 Contributions . . . 60

References **63**

List of Figures

1.1 Scenario and system model . . . 2
1.2 Local processing vs. computation offloading . . . 3

2.1 Example of dual plane architecture for 5G/4G integration, source: [11] . . . 9
2.2 Edge computing performance in [36] . . . 11
2.3 Influence of mobility metrics vs. network indicators, source: [22] . . . 12
2.4 Formula based predictor used in [25] . . . 13
2.5 Predictor performance relative to data used in [28] . . . 14
2.6 Prediction method performance in [37] . . . 15

3.1 Data exchange in computation offloading . . . 19
3.2 System network planes . . . 20
3.3 Hardware and software components of the system . . . 23

4.2 Example of command output . . . 29
4.3 Xiaomi MI 10 5G . . . 34
4.4 Labeled monitoring app screen . . . 35
4.5 Cables . . . 36
4.6 Communication layers of the system . . . 36
4.7 Database structure . . . 40
4.8 AP connected to PoE injector . . . 41
4.9 OpenWRT interface showing AC enabled . . . 41

5.1 WiFi Data Collection Experiment Diagram . . . 43
5.2 Nomad node elements assemble on a tray . . . 45
5.3 Structure and experiment components . . . 45
5.4 5G experiment data collection area . . . 47
5.5 System network map . . . 49
5.6 Main Experiment Route . . . 51

6.1 Box plot experiment 3 (DMA) . . . 54
6.2 Box plot experiment 4 (DMA) . . . 54
6.3 Box plot experiment 6 (Wifi) . . . 55
6.4 Box plot experiment 7 (5G) . . . 55
6.5 Box plot experiment 8 (Random) . . . 55
6.6 Average bandwidth values of 5G . . . 56
6.7 Average bandwidth values of WiFi . . . 56
6.8 Correct choices of RAT according to throughput history . . . 57
6.9 RAT choices made by the decision making module . . . 57

Abbreviations

SDN	Software Defined Network
URLLC	Ultra Reliable Low Latency Communication
OS	Operating
eMMB	enhanced Mobile Broad Band
MTC	Machine Type Communication
V2V	Vehicle to Vehicle
V2I	Vehicle to Infrastructure
V2X	Vehicle to Everything
VRU	Vulnerable Road User
LTE	Long Term Evolution
RTP	Real-time Transport Protocol
RTCP	RTP Control Protocol
IEEE	Institute of Electrical and Electronics Engineers
GPU	Graphics Processing Unit
RAT	Radio Access Technology
CPU	Central Processing Unit
DMA	Decision Making Algorithm
OAI	Open Air Interface
NSA	Non-Stand Alone
SSH	Secure Shell Protocol
MEC	Multi-access Edge Computing
ETSI	European Telecommunications Standards Institute

Chapter 1

Introduction

1.1 Context

In the past few years, plenty of new technologies capable of significant change have emerged. Self-driving cars are one of these technologies that have seen many advancements; however, these improvements have not been sufficient to consider them a safe alternative to human driving, yet. One contributing factor is that onboard vehicle video sensors can sometimes not detect particular objects in the road due to lack of line-of-sight, as shown in *Fig.1.1*. One way of approaching this situation is based on cooperative communications, where agents share information with one another with the intent of helping each other. These can be done between vehicles, road infrastructure and Vulnerable Road Users(VRUs), who communicate using their mobile devices. As referenced, the environment information can be captured through different types of sensors; however, this work will focus only on the video feed captured by intelligent vehicles. By sharing captured information about the environment between road users, it is possible to improve situational awareness of all the parties involved [33].

Nevertheless, it is not enough to capture a video feed of the environment; in order to detect objects, the frames of the feed must be processed. Since the context of the problem entails the safety of people (VRUs and vehicle drivers), it is critical that delays associated with object detection are minimized so that time to react is maximized. The task for detecting VRUs or obstacles in video feed is achieved with video analytics libraries, often designed to be run on GPUs. Programs that entail computer vision are usually very computationally heavy, which leads to the need for powerful hardware if delays are to be minimized. European Telecommunications Standards Institute (ETSI) investigations point for a delay budget of 300 ms for critical road safety applications, as stated in ETSI TS 101539-1 and ETSI TS 101539-3 [29]. One possible strategy for minimizing said delays would be to process the frames in an external device with better performing hardware. Also, in order to minimize the delay of communications, the location of this module should be close to the place where the video feed is captured. Edge computing is a possibility that fits in the moulds of this problem since servers are placed near the end-users, i.e. in the edge[30]. However,

to stream video feeds and maintain very low delays, a high throughput connection is paramount since the amount of data generated will be high. Current network protocols such as 802.11ac/ad and 5G allow for the quick transport of such data at much higher rates than other wireless technologies already being used for similar purposes, such as 802.11p. [31]

It is possible to define a clear path of information through these technologies. Large amounts of video feed data are generated in visual sensors of the car. This same data can then be transported using 5G networks and the 802.11ac protocol, which can achieve data rates of 10 Gbit/s[35] for millimeter wave and 1Gbit/s on midband frequencies for 5G and 3.4 Gbit/s for 802.11ac. Finally, the data can be delivered to and processed by the edge nodes. Since time is a critical factor in this context, having the video processing module closer to the vehicle could be vital in meeting the delay requirements.

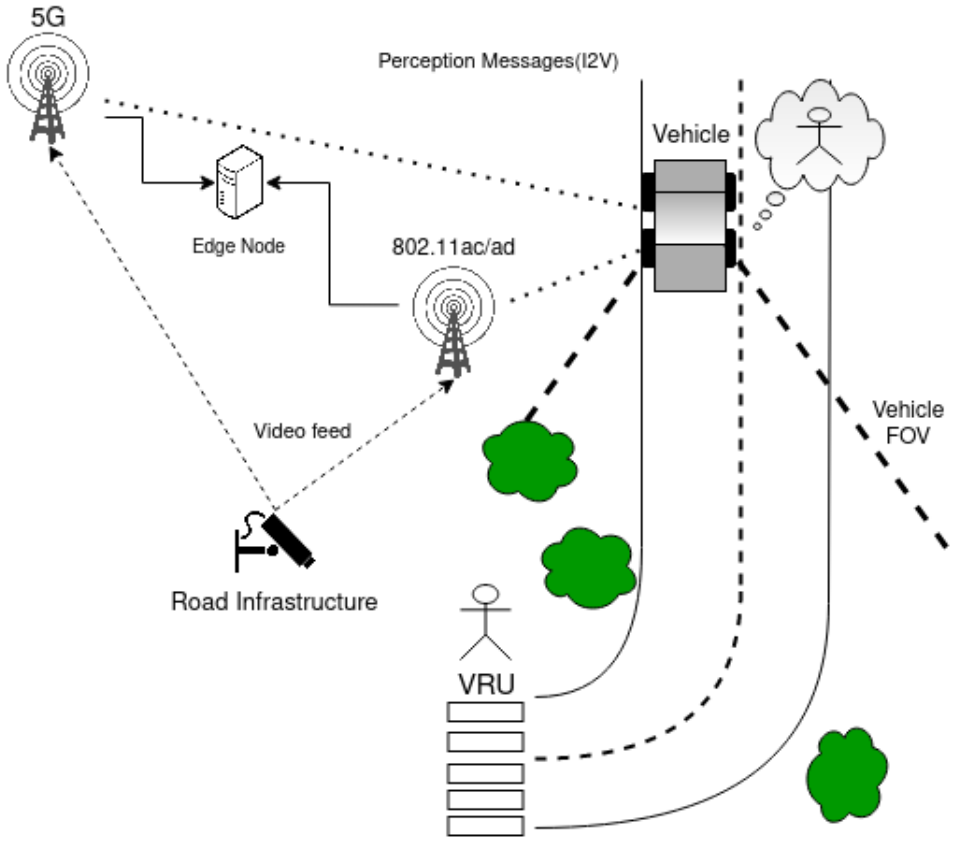


Figure 1.1: Scenario and system model

1.2 Motivation

The mentioned technologies, combined, can provide a possible improvement for a task crucial for the current systems of self-driving cars, which is the object detection of obstacles and other road users. This module captures a video feed that the car's computer vision system can process to detect objects, ranging from other vehicles to crosswalks or pedestrians.

The current method for object detection by vehicles presents two problems; the first one is that the vehicles' visual sensors can sometimes not detect particular objects if these are behind other objects or hidden in some other way, as shown in Fig.1.1. This limitation can lead to situations where the vehicle has little time to react when it detects the object, which can be a safety problem. The second problem comes from the car's hardware used for visual computations to detect objects on the captured video feed. Available solution's hardware has limitations in terms of computational resources. Usage of more powerful GPUs can lead to reductions in image processing delay up to 80% [36], however integrating this type of hardware on-board of vehicles has its own set of challenges, such as energy consumption and financial costs [17]. An alternative consists on providing access to more potent GPUs in an external location. In this work, the vehicles and road infrastructure send the data to be processed to the nodes.

It is important to note that, in any given moment, there will be multiple *Radio Access Technologies (RATs)* available for communication with the edge nodes. However, whenever multiple RATs accessible, choosing the best one is not a trivial process. The hypothesis being explored in this work consists in elaborating a decision making process for RAT choice in the context of edge computing. The critical aspect of this problem is the ability to share information promptly, in a way that the entities who make the decision have enough information to make the right decision in any situation they are presented with.

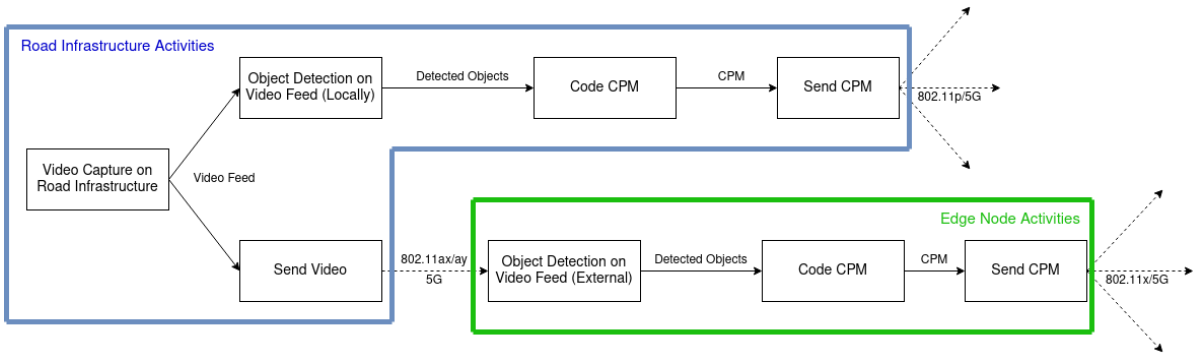


Figure 1.2: Local processing vs. computation offloading

1.3 Objectives

The work has one objective which consists in the design and implementation of a system that enables the use of heterogeneous access networks to offload a video stream for remote analytics; in order to empirically validate the system, a suitable testbed is implemented. In this case, the testbed supports access to edge computing resources in a heterogeneous network environment. The methodology follows a systems approach; this entails system design, identification of critical components, definition of their interactions and communications, implementing said components and testing the to validate evaluate their performance. The hypothesis of this work can be express as the following:

> "Is it possible and advantageous to use historic network performance to inform the choice of RAT in a heterogeneous network environment for access to edge computing resources?"

This work can be subdivided in five tasks which revolve around two main concepts a *Decision-Making Algorithm (DMA)* and a network performance monitoring system.

The first contribution consisted of defining and implementing the Decision Making Algorithm; this entailed studying previous projects and available literature on the subject and writing the algorithm's code.

This work also required definition of the monitoring system; this system simulates the scenario described in the *Context* section and functions as a testbed capable of providing diverse situations for testing the DMA. The system includes the transmission of a video feed from a user node to an edge node, the monitoring of network indicators relevant for the decision making process and the existence of two available RATs for the transmission of said video.

After the system was defined, it was assembled. The approach of this work is experimental; therefore, the described system was implemented in real hardware and integrated in out of lab components such as an operator 5G Next Generation NodeB (gNB). The choice of adopting an experimental approach, instead of using, for example, simulations, is explained in a future chapter.

When both the testbed and the DMA were ready, experiments were conducted where the DMA ran on different scenarios, and its performance was observed. Requirements-wise it was expected that the algorithm could autonomously choose the RAT, which is expected to provide the higher throughput, and also switch its decision if a change in the conditions justifies it.

1.4 Document Structure

The document follows the ensuing structure: the first chapter consists of the *Introduction*, which presents the work context, the motivation behind it and its objectives.

The second chapter introduces the state of the art of concepts related to this project; the chapter is divided into two sections. The *Relevant Background* section where bibliography associated with

the work is present and the *Related Work* section where projects key to the problem are described and succinctly discussed. Each of these sections is then divided into subsections in which each key concept is analysed.

The third chapter presents the problem in more detail and proposes a system design. This chapter also describes the modules of said system, their responsibilities and the interactions between them. In essence, this chapter further explains the problem and shows an overview of the proposed system abstractly, without delving into the implementation of said system.

The fourth chapter depicts every aspect necessary for carrying the ideas described in the previous chapter to the real world. Here is where the choices of hardware and code libraries are discussed, and the implementation details are described. The fifth chapter describes the experimental setup needed to conduct the experiments.

The sixth chapter contains the results of the experiments and the validation of the implemented system.

The last chapter presents remarks about possible future work on this subject, a list of contributions of this work and also the conclusions about the project.

Chapter 2

Current approaches to Vehicular Communication

Although this work will use relatively recent technologies, it will not seek to evolve any of them; instead, it will analyse the viability of using them together in the referred context. There are existing works that, even though, do not specifically approach the problem of self-driving cars, approach either similar enough problems or propose solutions which include relevant strategies to solve the problem in question. In this section, two types of works will be referred to. Firstly, some of the important technologies for this project will be presented in the "Background on Relevant Technologies" section, although still relevant, these technologies provide ideas or solutions that are loosely connected to this work. Bibliography included in the next section, "Related Work", is directly connected to this project's objective, concepts pointed on this section are crucial and will take centre stage on this work.

2.1 Background on Relevant Technologies

2.1.1 Protocol in current use

Currently, the network protocol used for communications with vehicles is 802.11p. This protocol defines the *MAC*, and *PHY* layers with a *Carrier Sense Multiple Access with Collision Avoidance(CSMA/CA)* access to the medium. The protocol supports Vehicle-to-Vehicle (V2V) and Vehicle-to-Infrastructure (V2I) communications; however, its bit rate is somewhat limited (27Mbit/s) [8]. This protocol is more suited for situations where data rates are not very high and reliability is not one of the main concerns.

Research provided [10] suggests an architecture for the transmission of information relative to VRUs to nearby vehicles. Through the architecture it is possible to obtain a birds-eye view of the challenges involved in the definition of a system for VRU protection. The article mentions

problems such as routing of information, time needed to issue a warning, available communication technologies (strengths and weaknesses), positioning of users and the possible use cases where the system could be used. In this work the focus will be on the communication technologies and the routing of information.

2.1.2 5G and 4G Networks

5G networks are the next step on broadband cellular networks, being considered an evolution on the previous generation, 4G. For the past year, this new generation of networks has been deploying in the real world, in certain countries; however, its coverage in most of the world is still very limited, especially when compared to 4G [11]. This is because the adoption of this technology is still ongoing and also due to 5G using higher frequencies, granting it a smaller coverage range. Integrating the two generations of networks is an essential objective in many systems. The 5G infrastructure available for this project is non-standalone (NSA) meaning 4G network infrastructure complements the functioning of 5G. Since 5G coverage is still low, ensuring there are no coverage holes will also be a responsibility of 4G networks[11]. If there is too much load being sent through 5G, 4G can also help ease those loads. Cooperation between the two is also necessary for moments of mobility between RATs. In the case of highly-delay sensitive applications, time of inter-RAT handover must be minimal, to the point where any change in service quality must be unnoticed by the user. This can only be achieved if there is integration and cooperation between the two RATs.

In [11] researchers propose a multi-RAT architecture that supports tight integration between *LTE (Long Term Evolution)* and 5G. This architecture tries to centralize whenever possible, and divides responsibilities into two planes: *Control-Plane (C-plane)* and *User-Plane (U-plane)*. The multi-RAT control is done on the C-plane managing resources for LTE, 5G, WLAN and LAA. U-plane functions include routing traffic to appropriate RATs and duplicating links in cases where an ultra-reliable connection is being provided.

In [11], the concept of network slicing is explored. This concept consists of network virtualization techniques to provide the multiple types of traffic supported by 5G. The first type is enhanced mobile broadband (eMBB), whose focus is providing very high bit rates, in the order of 1Gbps, while maintaining end to end latency around 10ms, relevant for any down-link hungry applications. The second, Machine Type Communication(MTC), allows for the simultaneous connection of billions of devices, with great potential for areas like IoT. The last type of traffic is ultra-reliable low-latency communications (URLLC); this type allows for fairly high bit rates (50Mbps) while maintaining a very reliable and low latency (<1ms) connection and has found uses in cases of telecontrol of machines, telemedicine, autonomous driving and factory automation[3]. The characteristics of this type of traffic are ideal for this work due to its ultra reliability and very low latency and for that reason its the type of traffic that will be used in this work. Since only URLLC traffic will be used in this project, there is no need for network slicing to implement other traffic types.[11, 14]

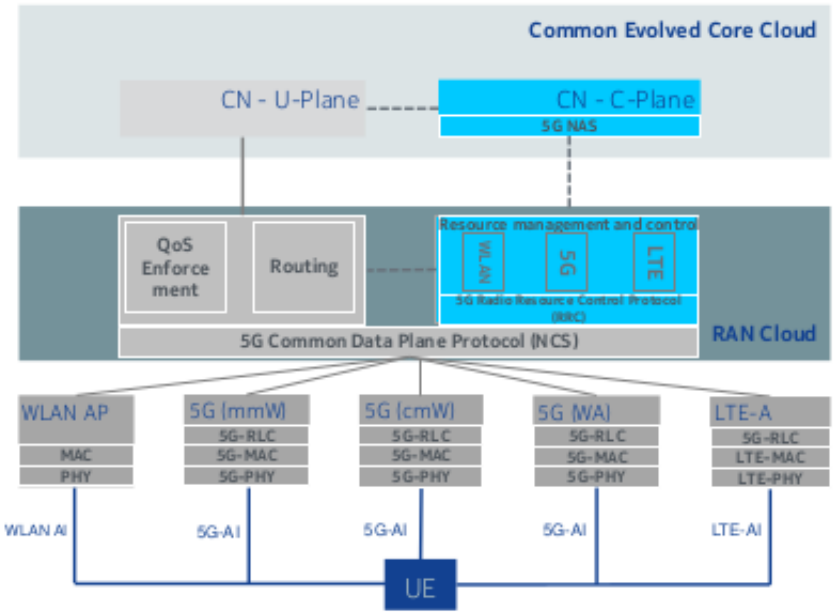


Figure 2.1: Example of dual plane architecture for 5G/4G integration, source: [11]

2.1.3 Vehicular communication

V2V, V2I and Vehicle-to-Everything (V2X) messages are a crucial concept on communication with vehicles. These can be used to share information with or from vehicles. Uses for this kind of communication can go from sharing information about the vehicle and detected objects, to platooning. There are already defined standards for the generation of messages used in this types of communications. For example, the European Telecommunications Standards Institute (ETSI) has proposed a standardization of the Cooperative Perception Message (CPM). This messages carry information about the vehicle, which generates the message, its sensors and the detected objects, and can be used to share sensory data with other entities or inform other road users of the existence of an obstacle on the road. [33]

V2X messages are a type of messages used for communication between vehicles and any other entity, and as such, are important in this work. In [33] researchers analyse the exchange of this type of messages between vehicles. They conclude that the channel is not efficiently used with these types of messages since many messages are sent to communicate the existence of very few objects. The authors then propose a look-ahead algorithm which tries to predict if any of the detected objects not included in the present message will be included in the following one, reducing the number of messages exchanged.

Work presented in [27] explores the usage of 802.11ac network protocols for the exchange of V2X messages, as a possible alternative to the current use of the 802.11p standard. The researchers set up an experiment similar to the one in [21, 22], but instead of the vehicle follows a course connected to one access point. Again results point out that mobility features like speed and direction of movement have a considerable weight on the performance of the throughput observed.

In [12], a prototype for communication of cooperative perception messages through V2X channels is presented. Finally, in [20], a multi-radio architecture is proposed with the aim to connect self-driving cars. 5G and other millimetre wave frequency systems are also studied as possible new RATs for V2X communications. The paper focuses on the usage of the 802.11p protocol and the integration between it and the 5G networks.

2.2 Related Work

2.2.1 Edge computing

Edge computing as a framework aims at offloading computing tasks from user devices with weak processing power to edge nodes with more robust hardware. Although similar, at first sight, to cloud computing, the main difference that distinguishes the two resides on the fact that edge nodes are closer to the users.

In [13], researchers propose an algorithm that aims to optimally allocate work and communications in a distributed environment of edge nodes. The proposed algorithm is only applicable if the communications scheduling policy follows three lemmas, which are

1. Non-preemptive communication scheduling policies are optimal
2. It is optimal to schedule all the forward communications before all the backward communications
3. It is optimal for the communication scheduling policies to be non-idle between forward communications and between backward communications, respectively.

. The algorithm defines the work distribution, message order and node selection for optimal performance, in most cases. In cases where neither the communication delays nor the computation rate are uniform, the authors propose a greedy algorithm as an alternative. Even though a distributed algorithm at run time execution is not being considered for this work, [13] provides a good framework for communication scheduling and node selection in an edge computing scenario.

There are other works which ascertain the gains of using such a framework. In [7] researchers divide a visual mechanism for indoor localization, into a local module, computed on the mobile device, and an edge module, computed on the edge node. However, achieving edge computing can not be done by only decoupling the modules and moving one of them to the edge nodes, since that would break the correct functioning of SLAM, the computer vision program used. In this article the authors modify the algorithm in order to allow it to run in two different machines.

As seen in [36], a recurring challenge on applying the edge computing framework seems to be the division of the local algorithm in two parts to be sequentially processed in different devices. While in [7] the solution for this problem consisted in adding a local map and a third connection to update it; in [36], since the task consisted in the usage of a deep neural network, the solution found was modifying an early layer to compress its output.

In [36], there is also a performance analysis and comparison of NVIDIA hardware on the task of object identification in images using Deep Neural Networks. The two GPUs are NVIDIA Jetson TX2, on the local device, and an NVIDIA GeForce RTX 2080Ti, on the edge server. For a bit rate of 4Mbps, a pure offloading approach would lead to delays of 17% the full local computation delay; however for bit rates of 1Mbps the difference is minimal, 97% of the original delay. These values show that when the hardware differences are noticeable, and the bit rate is high enough, edge computing can be a viable strategy in reducing computing delays.

In [23], researchers study a possible method to improve energy efficiency and latency of the system through the usage of Multi-access Edge Computing. The context of the work is also inserted on the protection of VRUs through information sharing, however, in this work the information comes from users' smartphones. The decision making process for the decision between local processing and offload to the edge is based on a prediction of reduction in latency and energy consumed.

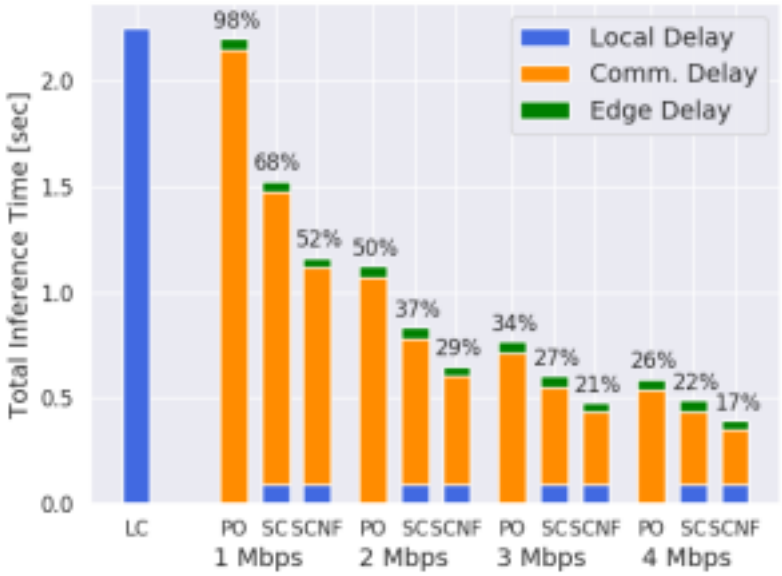


Figure 2.2: Edge computing performance in [36]

2.2.2 Network Performance Metrics

Understanding network performance and how specific network indicators will influence it is crucial when designing a decision making algorithm, responsible for choosing RATs (Radio Access Technology). Fortunately, there is already existing work on this matter. In [19] researchers explore the implementation of a "Quality of Experience"(QoE) metric to quantify network performance. Most of the time networks follow a "Quality of Service"(QoS) approach, which treats all network packets with the same priority, however, in times of congestion, delay-sensitive applications will experience delays much more sharply than other applications. This creates a need for managing

packets according to different kinds of priority, and this is where QoE proves to be useful. Following a QoE approach, packet management aims at maximizing user experience, an indicator which depends on the type of service the user is using. The researchers of the mentioned work use a Software Defined Network(SDN) approach, where a centralized entity collects data about the network to virtualize network resources and make decisions based on their usage. Through this approach, the researchers are able to unify multiple RATs and manage them to work cooperatively to ensure the best QoE possible. Another essential concept developed in [19] is the concept of User Agents, which are network modules which collect information about one user and calculate its perceived QoE. These modules are a vital component of information sharing needed for mobility between different RATs. The collected data used for the virtualization of resources consisted of Signal-to-Noise Ratio (SNR), bandwidth, signal strength, throughput, spectrum usage, jitter, delay connectivity and packet loss rate.

Work presented in [22] on the other hand, explores the bit-rate achieved between a vehicle and access points placed in a crossing, using the 802.11n, 802.11ac and 802.11ad network protocols. Researchers started by performing data collection experiments with a car passing through a crossing multiple times. In the referenced work, the researchers realized that achieved bit rate depended not only on network metrics but also on mobility features and location of the vehicle. Algorithms that considered mobility features had lower data rate forecasting errors, showing that user movement is an essential factor in predicting network performance. In the following graph, both MRDPS and MRDP represent approaches that only use mobility features to predict network throughput. According to this article, prediction of throughput must take in consideration a history of mobility features and network indicators in order to make accurate predictions. This will be the main method in this work for predicting the network throughput, using the history of network and mobility factors in a given location.

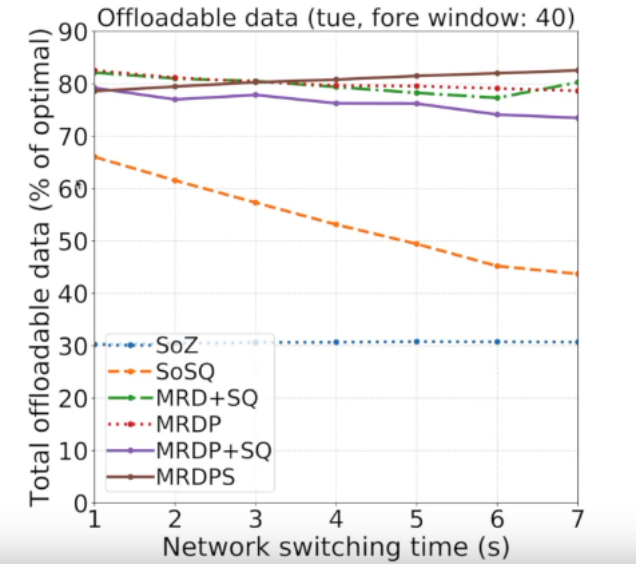


Figure 2.3: Influence of mobility metrics vs. network indicators, source: [22]

2.2.3 Throughput prediction

Throughput prediction as a research area is still an ongoing subject, and although there is some literature on this field of study, it is not much. It is also primarily focused on the TCP protocol, which is not used in this work. The available work is also more focused on the problem of prediction of throughput, which attempts at predicting the throughput before the transmission starts, instead of estimation, which uses information from the ongoing transmission.

From the studied articles, it is possible to define three types of throughput prediction techniques: Formula based (FB), history based(HB) and machine-learning based(MB); this last method can use network data or throughput history data in order to make predictions. Formula based approaches attempt to predict throughput through mathematical formulas which relate network indicators such as round trip time (RTT) or path loss to the network throughput. On the other hand, history based methods use data from previously recorded throughput values. Finally, machine-learning methods use machine-learning techniques to create throughput prediction models. These models can be trained and used with throughput history data or network indicators data, similar to history based and formula based methods.

In [24] researchers propose a novel stochastic throughput prediction method based on the study of already existing linear predictors such as ARMA, FARIMA and GARCH, for mobile networks. The proposed method used variables such as SINR, distance to the transmitter, number of competing nodes in the network to predict throughput through formulas.

Research presented in [25] compares formula based and history based approaches for throughput prediction in TCP traffic. The formula based predictor used in this paper is represented by the following equation:

$$\hat{R} = \begin{cases} \min\left(\frac{M}{\hat{T}\sqrt{\frac{2b\hat{p}}{3}} + \hat{T}_o \min(1, \sqrt{\frac{3b\hat{p}}{8}})\hat{p}(1+32\hat{p}^2)}, \frac{W}{\hat{T}}\right) & \text{if } \hat{p} > 0 \\ \min\left(\frac{W}{\hat{T}}, \hat{A}\right) & \text{if } \hat{p} = 0 \end{cases}$$

Figure 2.4: Formula based predictor used in [25]

For history based methods, the researchers use three types of linear predictors: Moving Average(MA), Exponentially Weighted Moving Average(EWMA) and non-seasonal Holt-Winters(HW). For the Moving Average and the Holt-Winters methods, a second version of these methods was used, which added the detection of level shifts and outliers (MA-LSO and HW-LSO).

Results show that history based predictors achieve significantly better accuracy than the formula based predictor chosen. Within the history based approaches, the inclusion of level shift and outlier detection substantially improves the performance of the Moving Average and Holt-Winters methods. Without level shift and outlier detection, the EWMA and HW methods show the best performance. When LSO is included in MA and HW, these two predictors achieve the best performance of all methods considered, with HW-LSO being slightly better.

In [28] the effects of using different sets of data in throughput prediction in mobile networks are studied. The predictor model used is machine-learning based and is built using the Random Forest algorithm with a K-fold method for validation with K=10.

Researchers designate four sets of data to serve as input to the prediction mechanism: User Equipment Categories and Cell Frequency Band set; Physical Layer (Radio) set which includes Reference Signal Received Power, Reference Signal Received Power, Reference Signal Received Power and RSSI; Context Information set which designates if the user is indoors or outdoors, their speed and their distance to the cell; and RAN measurements set whose metrics include average cell throughput, the average number of users in a cell, block error ratio(BLER) of the cell and radio resource control setup success rate.

Results point to information from the Radio and RAN sets to be the most influential in reducing prediction errors. Nonetheless, the inclusion of the Context set could also lead to improvements. Predictors that used RAN and Radio data achieved median absolute error ratios of 0.1, being considered by the authors as accurate predictors.

Predictor	# variables	# entries	R^2	$\bar{E}_i$
x = UE cat. + Cell band	2	5757	0.39	0.28
x + Radio	4	4677	0.70	0.19
x + RAN	6	2842	0.71	0.17
x + Context	5	3827	0.65	0.20
x + Radio + RAN	8	2626	0.85	0.11
x + Radio + Context	7	3193	0.81	0.13
x + RAN + Context	9	1871	0.74	0.15
x + Radio + RAN + Context	11	1813	0.84	0.10

Figure 2.5: Predictor performance relative to data used in [28]

Finally, the study presented in [37] compares multiple history and machine-learning based throughput prediction methods using TCP traffic. All methods in this work use previous values of recorded throughput in order to make a prediction. History based methods include: arithmetic mean, harmonic mean, geometric mean and the exponentially weighted moving average; while machine learning predictors include multiple linear regression, neural network regression and support vector regression.

In order to evaluate the performance of the algorithms, the chosen metric was the root mean squared error. According to the results, the article's authors distinguish two approaches to the throughput prediction problem: the multiple linear regression and the arithmetic average. Although less complex than the other machine learning methods, the first predictor showed the best results of all approaches. The arithmetic mean, even though it is extremely simple to implement and does not require training, "performs very well". The following graph shows the performance of the methods according to the amount of data used; note that the highlighted methods perform the best and with a slight difference in RMSE.

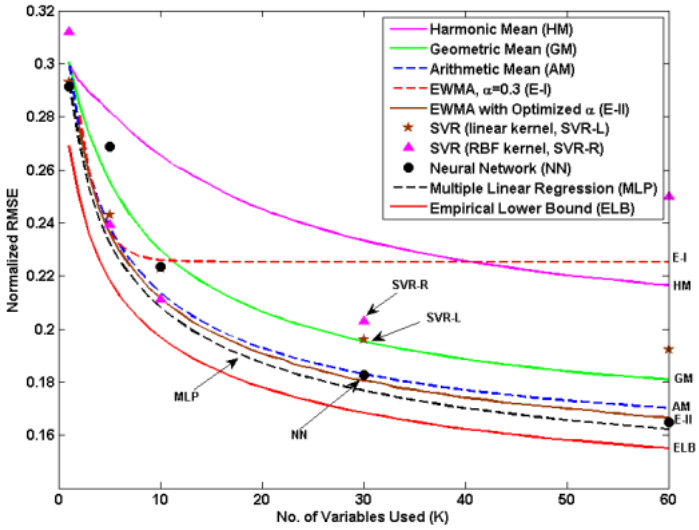


Figure 2.6: Prediction method performance in [37]

2.2.4 Video Streaming

The envisioned system requires the streaming of video data from a sender to a receiver. To understand how this could be achieved, a study of the currently available media streaming protocols and tools was conducted. The three concepts studied were the Real-time Transport Protocol, the Web Real-Time Communication and the Real-Time Messaging Protocol.

The Real-time Transport Protocol (RTP) is a media streaming end-to-end protocol used for real-time data such as video and audio. RTP does not implement a transport layer protocol instead, it uses the already existing User Datagram Protocol (UDP). One of RTP main advantages is the number of features it comes with. RTP provides sequencing, intra-media and inter-media synchronization and payload and frame identification. It is also media-independent; in other words, it can use any type of video or audio format without impacting its functions.[32]

The Real-time Transport Control Protocol (RTCP) accompanies the RTP and transfers control information about the RTP sessions. It uses three types of information structures: Sender report, Receiver report, and Source descriptors, all periodically sent in RTCP packets. In cases where control data does not fit in the structure of RTCP packets, the protocol allows for the usage of RTCP APP packets which can be defined and used according to the application needs.[1, 2, 32]

The Web Real-Time Communication (WebRTC) is a software tool that equips web browsers with the possibility of transferring video and audio with low latency. The system has a peer-to-peer architecture where users communicate directly with each other. Communications between browsers are set up using HTTP requests and WebSockets, while streaming of video and audio is achieved through the RTP and RTCP protocol. [18]

At last, the Real-Time Messaging Protocol (RTMP) is an application-level protocol developed by Adobe aimed at video and audio streaming. The protocol uses persistent connections and uses the TCP transport protocol to achieve low latency in the transmission of data. The protocol uses a

client and server architecture where media fragmentation size is negotiated between both parties. [9]

Chapter 3

Heterogeneous Wireless Access to Edge Computing Resources

3.1 Defining the problem

The objective of this work is the development of a system capable of simulating the relevant scenarios of heterogeneous networks for testing the decision making module based on geo-referenced historic throughput data. In the given context, the autonomous vehicles have access to multiple RATs in order to offload computing tasks. However, computation offloading entails communication between the mobile node and the external device, which itself adds another delay, the communication delay. Nonetheless, in an edge computing architecture the computational resources are placed closer to the end-user (than if they were placed in the cloud) resulting in smaller communication delays.

This work aims to experimentally test the feasibility of using heterogeneous access to access edge computation resources for video processing.

In order to efficiently stream the captured video feed to an edge node, the communication delay must be minimized. Usually, there are multiple types of radio access technologies available in a given area; as such, choosing the RAT which has the best performance is a key and not trivial task. For this purpose, mobile nodes must be able to know a prediction relative to RAT performance to make an informed choice. Although many factors define network performance, this work will consider only on predicting throughput since this is already a complex task and tackling other indicators would not be possible in the given time frame. It is worth noting that the throughput prediction is still an open and ongoing research topic and that this work's contribution to the subject is the design of a possible monitoring system.

According to the literature presented in *Chapter 2*, the referred throughput prediction techniques require data about the networks to make their predictions. The data to be collected by the monitoring system must follow the methods' data needs. However, depending on the implementation of the system, the data collected will also be limited by the collection capabilities of

the network monitoring module. This module must capture live data of the RAT in use for video streaming and send it to a data storage module. On the other hand, the data storage module must constantly be listening for new monitoring data being sent to it, storing said data, and providing it to users at their request.

Another challenge to test the hypothesis consists in the definition and creation of the communication agents of the modules of the system. According to what is known of the problem, there needs to be three types of communication: video feed streaming transports the video feed from the capture device to the edge node; network monitoring sends the network monitoring data to the storage entity; and monitoring data requests, retrieve data about a given network and location, and transfers said data from the storage entity to the entity responsible for the decision making process. Another essential requirement is that the video streaming can be done by any of the available RATs.

These are the challenges associated with the problem that are tackled by this work. However, there are other challenges related to the given problem that will not be explored in this work. The system will use already available solutions developed by other parties and researchers to solve the following tasks.

Although the work uses object detection as motivating application, no actual work will be done on this matter and this functionality will not be included in the system. The edge node would receive and process the video feed, extracting information, such as speed and position, about present obstacles, road users and elements. After processing the video frame, the edge node would broadcast the extracted information.

The throughput prediction techniques are also out of the scope of this work. Therefore, this work will not attempt to develop new throughput prediction methods, further develop already existing ones or compare their performances.

Finally, an often crucial factor in the performance of video streaming applications resides in the encoding of the video stream. Nonetheless, the impact of this task will not be studied on this work, neither will the performances differences that come from using different encodings.

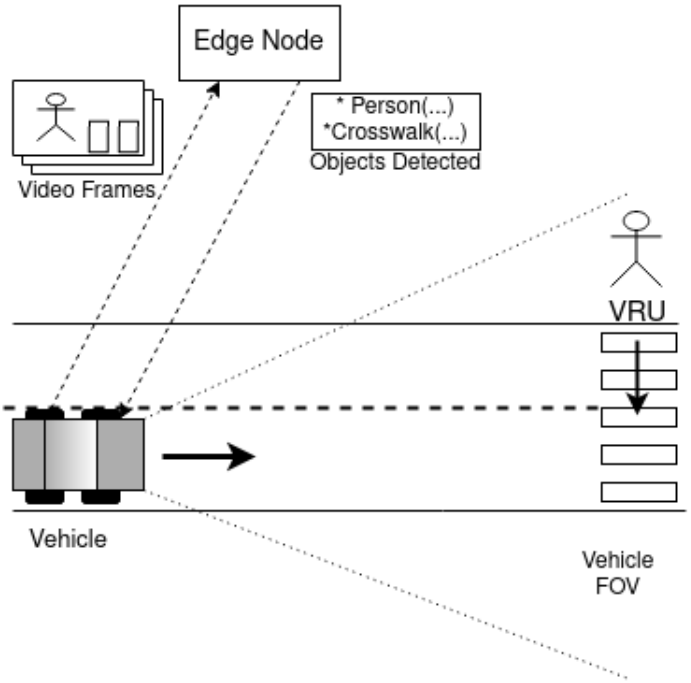


Figure 3.1: Data exchange in computation offloading

3.2 Use Case and System Design

The presentation of the problem in the previous section portrays the complexity of the needed system to provide a solution to the problem. The system is divided in three modules: *Decision Making* module, the *Network Monitoring* module and the *multi-RAT Forwarder module*.

The system architecture follows a typical network design where communications can be associated with one of two planes: the control-plane (C-plane) and the data-plane (D-plane). The C-plane is associated with messages that contain information about the network status and performance, thus this plane coincides with the Network Monitoring module. On the other hand, the D-plane encompasses the messages sent and generated by the mobile nodes of the network, therefore, in the given system, this plane contains the multi-RAT Forwarder module and the exchange of video streaming messages.

The use case of the system consists of a mobile node requesting in advance historic performance data associated with the location to which it will travel. After retrieving said information, the node makes a prediction on the performance of the available RATs in the area. With the prediction information, the decision making module can then decide which RAT to use to stream video when the mobile node is in the associated location. Whenever the mobile node reaches the location, it already knows the RAT it will use to stream the video. The multi-RAT forwarder then sends the traffic using the chosen RAT.

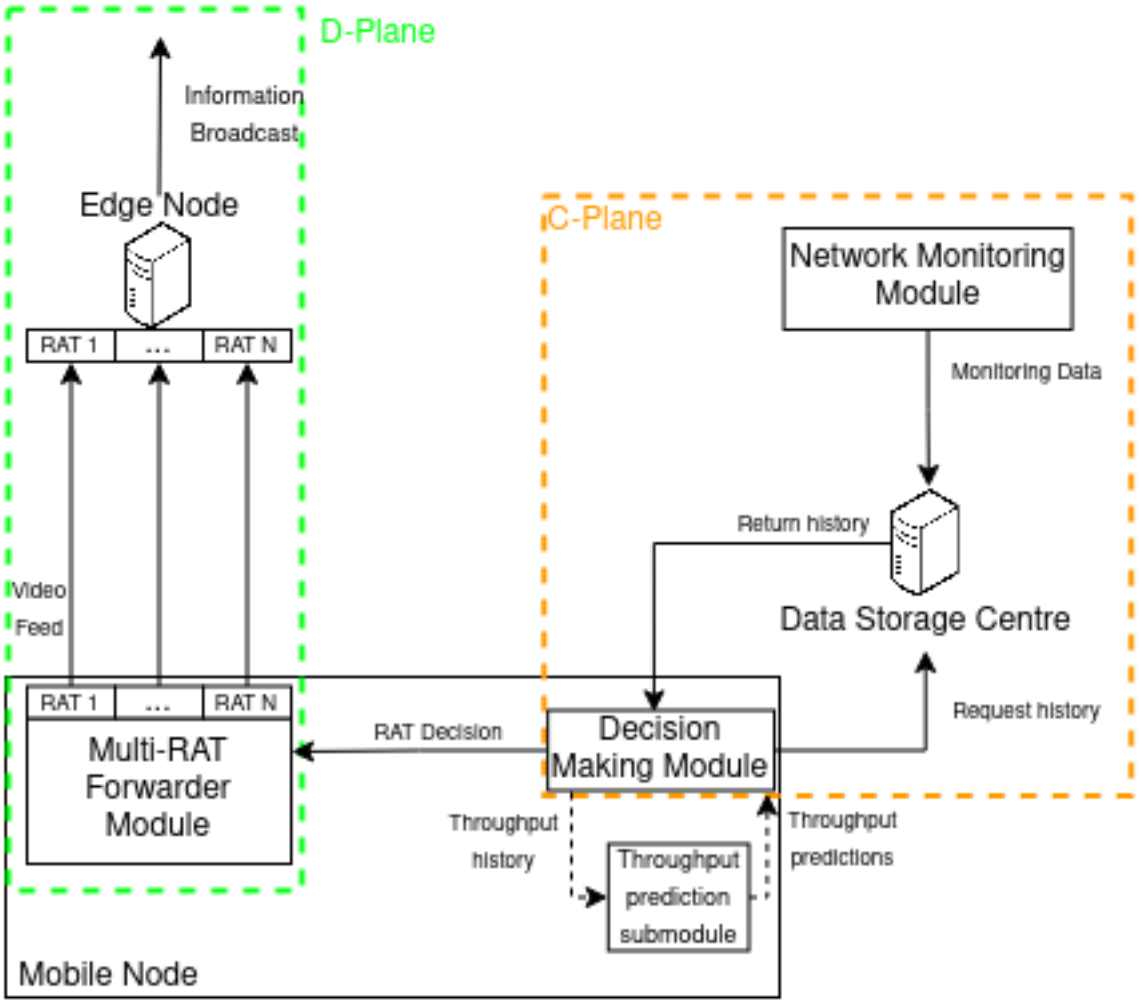


Figure 3.2: System network planes

3.2.1 Decision Making module

The *decision making module* is responsible for analyzing the available network monitoring data and predicting the best RAT for video streaming in a given area. This module needs to communicate with the two remaining ones in order to accomplish its objective. To have enough information to make a decision, the decision making module must request the network monitoring module for data about a given access network in a specific area. This data can consist of live data captured in the moment or historic data generated in the past and contained in a storage centre. The decision making module should also filter the data according to their relevance to the current situation. The decision is context aware, as such, the network monitoring data must be accompanied by the

location to which is associated. The module also communicates with the multi-RAT Forwarder module in order to inform it of the best RAT available at the moment and influence the type of communication to use to stream the video.

This module can be of two natures: it can be a centralized node that collects information about the mobile node's networks and position and then informs them of the best available RAT or a decentralized node present in each mobile node.

3.2.2 Network Monitoring module

The network monitoring module is responsible for the collection and storage of the network data. Since the context of the problem contemplates heterogeneous networks, the network environment will change, and the mobile nodes must be able to adapt to said alterations. For these adjustments to be possible, the mobile nodes must be provided with data about the network to make an informed decision.

The data to be collected must be relevant to the decision making process. Although this is highly dependent on the decision making algorithms employed, it is safe to say that the data should include network performance indicators and also information about the mobile node that is relevant to predict future network performance, such as its position, speed and bandwidth needs.

The place on the system or in which sub-modules the data collection is conducted is a crucial design decision since this will influence the data collected. In other words, the data will be generated according to one perspective instead of other. It is possible to conduct network monitoring through two ways: active and passive measures. In passive measures no extra traffic is generated, the network is only observed while with active measures the module generates extra traffic in order to extract more information about the network.

The monitoring module can either be a centralized node or multiple decentralized ones. In a decentralized approach, data collection nodes would be spread across different areas or even included in the mobile nodes, whereas, in a centralized approach, one single node could collect network data relative to one or more RATs in a large area and even also function as a data storage node. In decentralized nodes the data collection method can either be passive or active. Choosing a centralized approach requires the use of active measures to monitor the networks, since mobile nodes would have to specifically communicate with it. In this work the monitoring will be done exclusively through passive methods, therefore, this places the centralized approach out of the scope of the project.

There is also the question of how and where to store the data. This design decision is dependent on the type of monitoring node chosen. Suppose we opt for a centralized monitoring node. In that case, the data can be stored on the same node resulting in it having the double responsibility of collecting, storing and providing the network data. This could also be possible in a decentralized architecture, in which nodes ask each other about the data they collected. Finally, there could also be a designated node whose only responsibility is receiving, storing and providing the data. Independently of the type of node chosen, the data storage node should store the data in an organized way and have it available at low latency to the mobile nodes. This is important in order to filter the

relevant data for the decision making process and also to reduce data retrieval delays. The access to this information should be simple, therefore, the API to access it has two access points: one to retrieve information based on a series of criteria and another to add new monitoring data. This data consists on values such as observed throughput, signal strength indicator, number of sent bits, latency indicators, RAT chosen, location, speed, orientation and movement of the mobile node.

Although we aim to reduce delays on the requests for network data, these delays are not critical to the system. This is because, in the envisioned scenario, autonomous vehicles can request the historic performance data long before arriving at the place where they will have to stream the video. The usual delay for this type of request for information is usually much smaller than the time vehicles can request the data before arriving at the location associated with said data. This opens up the possibilities of the location of the data storage node. Since the delay is not crucial this node can be places within the system network, in the edge or even in the cloud.

3.2.3 Multi-RAT Forwarder Module

The multi-RAT forwarder module is the part of the system in charge of transmitting the video feed. This module represents a part of a user of the system, i.e. an autonomous vehicle or equipment of road infrastructure. The critical requirement of the user is that it should be able to transmit a video feed that must be processed for object detection. In the envisioned system, the video feed is transmitted to an edge node responsible for processing it and conducting the object detection tasks.

As stated previously, this module is part of the D-plane of the system since its primary function consists of forwarding the video stream. This type of data transmission must be possible through more than one RAT since the system is built around the use of multiple access possibilities. Thus, the multi-RAT Forwarder module must support communications through multiple RATs. The concurrent transmission of data through more than one RAT, although an interesting solution, is too complex to approach under the existing time constraints and as such is out of the scope of this work. As such, the system will support the video streaming through multiple interfaces, but only one at a time. Although the figure 3.3 shows only two types of RATs the system, should support more than two and other RATs apart from 5G and WiFi 802.11ac.

Before starting to forward video segments across one access network, the module must wait for a response from the decision making module, or if no response is given, opt for a default choice.

Finally since this module can run in a vehicular node, its mobility must be considered, because it is a factor than can influence the performance of the network. The multi-RAT Forwarder module needs to be able to move or be repositioned from time to time to simulate the movement of a vehicle or the displacement of a road equipment. In a situation where the module is moved, it is the responsibility of the decision making module to detect the change, make a prediction based on the new conditions and communicate it the to the multi-RAT Forwarder module. A change of position could lead to a difference in network performance due to new obstacles, or increased distance to networks elements, as such a rectificative action is taken in order to maximize performance.

3.2.4 Edge Node

In the system context, the edge node is independent and does not fit in any of the mentioned modules. Its sole objective consists in processing the video feed and sharing the information of the detected objects with the sender of the video feed and possibly with other mobile nodes which could be interested in the data. The placement of the processing device on the edge aims to reduce data transport delays by decreasing the physical distance between the mobile node and the processing device. Since the mobile nodes will use multiple RATs to transmit their video feeds, we assume the edge node will be accessible by all types of RAT used by the mobile nodes and reconstruct the received video even if segments of it are received using different RATs. Although this node is independent, since it still receives video streaming data, it is included on the D-plane of communications of the system.

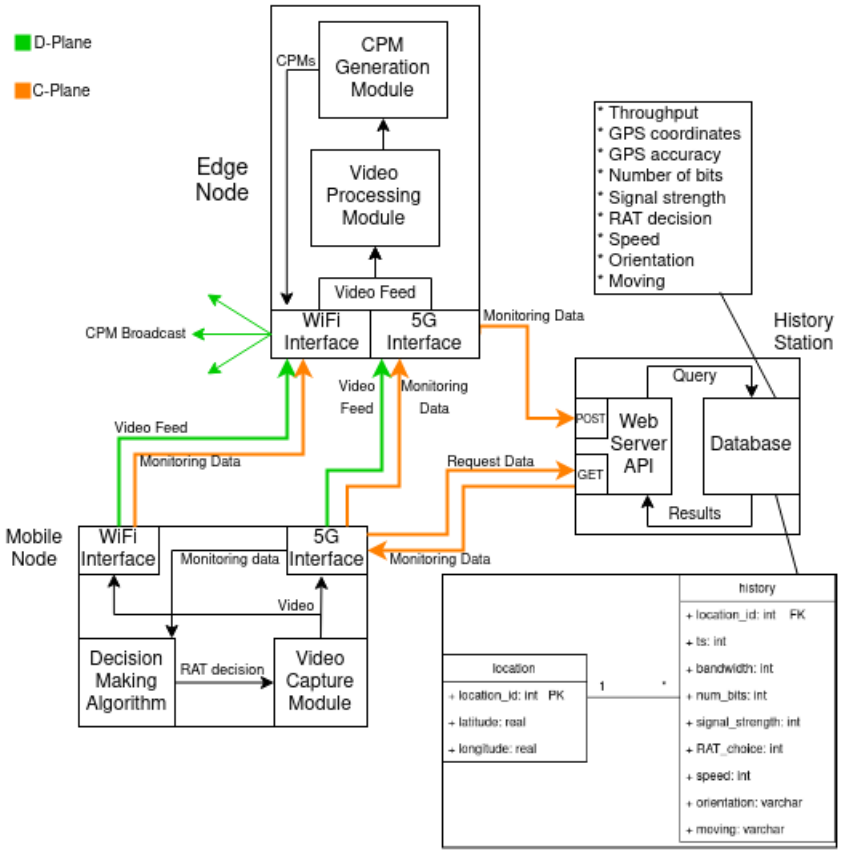


Figure 3.3: Hardware and software components of the system

3.2.5 Minimal functionality

Given the limited time of this project, this subsection defines a set of minimal functionalities that need to be implemented and validates. Regarding the Decision Making module, it needs to be able to retrieve historic network performance data and make a decision, selecting one of the available RATs, in an autonomous way. The multi-RAT forwarder module needs to be able to transmit the

video frames from the mobile node to the edge node. Finally, the network monitoring module should be able to gather data about a network, store it in a storage centre and have it available to mobile nodes.

3.3 Methodology

This work will follow an experimental methodology. The modules being developed will be tested in a real-world scenario under controlled conditions. The alternative to this methodology would be to test the modules using simulations and theoretical models; however, this is not intended with this work.

By following an empirical approach, the testing environment will encompass variables and limitations that are not present in the referred theoretical models, allowing for more realistic scenarios to observe the algorithm's behaviour and the system. The number of challenges to be found on the development of an empirical testbed and real-life system is also greater; however, this leads to the creation of a real-life system that could be used to make conclusions on the viability of the developed system in a real-world scenario. Using theoretical models or simulations includes advantages that do not reflect the real world, such as the availability and no cost of acquiring information or the lack of interference from obstacles between the client and node. In this work, we want to test the developed system in the most realistic situations possible, thus leading to the adoption of an experimental approach.

3.4 Tools

In order to implement the presented system and analyse its performance, certain tools are needed. Developing the needed tools from scratch for the specific purposes of this work, would not be possible in the available time frame and would be suboptimal, since there are already existing tools that would fit the needs of the work.

3.4.1 JRTPLIB

The transmission of data in the system will be dependent on the communication protocol used. Although this will be tackled in more detail in the next chapter, the protocol chosen for video streaming was RTP/RTCP. After studying multiple available libraries that implement said protocol, such as *live555* and *ccRTP*, the library chosen was *JRTPLIB* by Jori Liesenborgs [16]. *JRTPLIB* consists of a C++ object oriented library which implements the construction and sending of RTP and RTCP packets. The library is free, open-source and compatible with Windows and Ubuntu operating systems.

3.4.2 Consumer and Producer

Consumer and *Producer* are two programs that complement each other with the purpose of network stressing and monitoring. Both programs were developed in C and are based on the *iperf* tool. The Producer program generates network traffic which is sent to the Consumer. Both programs record and monitor network indicators such as bitrate, number of bytes/packets received and interarrival time of packets. These tools were provided by their authors Rui Meireles and António Rodrigues and were developed for their work [26].

3.4.3 TEMS Pocket

TEMS Pocket is a mobile application tool, developed by Infovista, used for monitoring and testing networks. The critical factor of this app is the fact that it supports 5G NR technologies which is paramount to gather data on 5G networks.[6]

Chapter 4

System Implementation

This chapter describes the implementation details of the project. The realization of each of the modules described in the previous chapter, the challenges met, and the solutions found are presented in this chapter. Each section describes one of the system's modules, and these are then subdivided by their hardware and software components.

4.1 Nomad Node

In the previous chapters, the term "mobile node" was used to represent the network users. These users had mobility capabilities similar to a vehicle and could predict their future locations and path. However, in the implementation of the system, these capabilities were not recreated. As such, in this chapter, we use a different term, nomad node. This node can move, however, not with the speeds of a vehicle or with the capability of predicting its following location. This node can be seen as a parallel to portable equipment of road infrastructure.

4.1.1 Central Manager

The central manager of the nomad node is the leading software component of this module. It is responsible for managing the other components, the network interfaces and the communications with the edge server and the history server.

The program runs on a *Raspberry Pi 4*(RP4) computer with *Ubuntu 18.04 LTS OS*. The choice for this hardware comes from its portability, which is something needed to carry the experiments, its availability as a component, its ease of programming and integration with network modules. Ubuntu 18.04 comes already equipped with valuable tools for network monitoring such as *iw tools*, programming tools, it is compatible with most code libraries available, and it is a familiar OS to the researcher.



(a) Closed Raspberry PI 4



(b) Open Raspberry PI 4

The communication through WiFi with the edge server is done using the 802.11ac protocol. Fortunately, the Raspberry Pi 4 Wifi card supports this protocol, so no external devices are needed in this case. The same cannot be said for 5G networks, since the RP4 does not support this type of communications. In order to make use of 5G, the RP4 is connected to a 5G smartphone, which then works as a 5G interface.

The central manager uses the iw tool to obtain monitoring information about the WiFi connection. This can be achieved with the command:

```
iw dev <interface name> link
```

More specifically the following values can be obtained:

- RX bytes
- TX bytes
- RX packets
- TX packets
- RX bitrate
- TX bitrate
- Signal strength
- Frequency

The command's output is piped to a python script that parses it into a more processable form.

In order to monitor the 5G network, the central manager communicates with the 5G smartphone. This communication is done through sockets and a USB connection. The connector consists of a USB type A (Male) to USB-C (Male) cable. This module is responsible for creating a socket on port 8083, listening to said port and parsing the information received to be sent to the edge node. The 5G monitoring app handles communications on the side of the smartphone.

```
f0nf0n@f0nf0n-lenovo:~$ iw dev wlp2s0 link
Connected to d0:77:14:be:a5:8f (on wlp2s0)
        SSID: AfonsoT1
        freq: 2422
        RX: 39521515 bytes (35653 packets)
        TX: 2707920 bytes (18847 packets)
        signal: -35 dBm
        rx bitrate: 72.2 MBit/s MCS 7 short GI
        tx bitrate: 72.2 MBit/s MCS 7 short GI

        bss flags:      short-preamble short-slot-time
        dtim period:    2
        beacon int:     100
```

Figure 4.2: Example of command output

Before the socket communication can start, a port forwarding operation is needed. This operation is done with the *Android Debug Bridge (ADB)* tool. Although the tool is destined for debugging operations on Android smartphones connected to computers, it allows forwarding contents from a device(smartphone) port to a host (computer) port. The command to achieve this is:

```
adb reverse tcp:<device_port> tcp:<host_port>
```

In this work's experiments the ports used were 8083 and 8083.

The monitored information of 5G networks was:

- Rx Bandwidth
- Tx Bandwidth
- Signal Strength

The realization of the C and D planes was done using the *Real-time Transport Protocol(RTP)* and its sister protocol *RTP Control Protocol (RTCP)*. There were already existing code libraries with the implementation of the protocols; the ones found during development were *live555*, *jrtplib*, *GStreamer* and *ccRTP*. In the end, the chosen library was jrtplib due to its better documentation, available code examples and ease of programming.

The code flow for the communication operations can be divided into the following steps:

1. **Setup RTPSession** - Initializes an object of the RTPSession class (jrtplib) responsible for setting up the RTP and RTCP communications. This involves defining the IP address and port of the destination, and packet characteristics, such as payload type and maximum packet size;

```
// Setup RTPSession
 jrtplib::RTPSession session;

    jrtplib::RTPSessionParams sessionparams;
    sessionparams.SetOwnTimestampUnit(1.0 / 8000.0);

    jrtplib::RTPUDPv4TransmissionParams transparams;
    transparams.SetPortbase(8000);

    int status = session.Create(sessionparams, &transparams);

    uint8_t localip[] = {127, 0, 0, 1};
    jrtplib::RTPIPv4Address addr(localip, 9000);

    status = session.AddDestination(addr);
    if (status < 0)
    {
        std::cerr << jrtplib::RTPGetErrorString(status)
        << std::endl;
        exit(-1);
    }
    status = session.SetMaximumPacketSize(32000);
    if (status < 0)
    {
        std::cerr << jrtplib::RTPGetErrorString(status)
        << std::endl;
        exit(-1);
    }
    session.SetDefaultPayloadType(96);
    session.SetDefaultMark(false);
    session.SetDefaultTimestampIncrement(160);
```

2. **Open video stream** - A series of operations needed to open the file that contains the video and read it frame by frame. These operations include finding the correct stream, the suitable decoder and initializing the codec parameters and context. All these operations are done in the *readNAL2Buffer* function which consists on an adaptation of code found in [4] and makes use of the *libav* library [5]. In order, to extract each frame from the video feed two Libav objects are needed *AVCodecContext* and *AVFormatContext*. The *readNAL2Buffer* function returns both of these values.

```
    //Find video stream, allocate codec context and find decoder
    AVCodecContext *pCodecCtx;
    int stream = av_find_best_stream(pFormatCtx,
    AVMEDIA_TYPE_VIDEO,
    -1,
    -1,
    NULL,
    0);

    if (stream == AVERROR_STREAM_NOT_FOUND)
        cerr << "Didn't find a video stream" << endl;
    if (stream == AVERROR_DECODER_NOT_FOUND)
        cerr << "Didnt find decoder" << endl;

    const AVCodec *pCodec = avcodec_find_decoder(
    pFormatCtx->streams[stream]->codecpar->codec_id);
    pCodecCtx = avcodec_alloc_context3(pCodec);
```

3. **Setup NetMonitor** - The *NetMonitor* class uses the *ThreadPollerWifi* and *ThreadPoller5G* classes to monitor the WiFi and 5G connections, respectively. This class is responsible for generating and handling all RTCP communications.

   Each of the ThreadPoller classes is responsible for extracting the data, parsing it and returning it in a suitable state for its inclusion in the next RTCP message; the ThreadPoller5G class must also set up the socket (based on C Sockets (sys/socket.h))connection with the 5G phone.

   The *NetMonitor::dataLoop()* function is started on a different thread in order not to block the execution of the code which sends the video feed. During this loop, the function extracts network monitoring data from the *ThreadPoller* classes, and with that data creates the payload of the next RTCP message. Since this type of data is not contemplated in any type of RTCP packet, an RTCP APP packet is used, whose finality extends the use of RTCP packets for application-specific purposes [2].

   It is important to note that the execution of this code is concurrent with the one responsible for sending the RTP messages (video feed frames) and that the class responsible for sending both types of messages (RTPSession) is not thread-safe. Therefore, a semaphore was implemented in order to protect the code from eventual race conditions.

```
while (true)
        {
```

```
3               if (socket)
4                   fiveGData = m5G.parseDataToArray(m5G.getData());
5               if (currentRAT == 0)
6               {
7                   geoInfo = getGeoInfo(fiveGData);
8                   wifiData = mWifi.getData();
9                   msg = generateControlWiFiMsg(wifiData,
10                  geoInfo,
11                  &msg_size);
12              }
13              else if(currentRAT == 1){
14                  msg = generateControl5GMsg(fiveGData,
15                  &msg_size);
16              }
17              else{
18                  cerr << "ERROR: Unknown RAT specified\n";
19              }
20              while (thread_lock)
21                  std::this_thread::sleep_for(timespan);
22              thread_lock = true;
23              int r = sess->SendRTCPAPPPacket((uint8_t)0,
24              (uint8_t *)"APP1",
25              (void *)msg,
26              msg_size);
27              thread_lock = false;
```

4. **Read and send frames** - After the *AVFormatContext* and *AVCodecContext* objects are set up with the needed information, the program can start reading the the video feed frame by frame. Each frame after being read is placed in a *AVPacket* structure. In this project, we are only interested in the video, so audio frames are ignored. The data of each *AVPacket* is sent on an RTP packet; however, since sending an RTP packet requires the *RTPSession* class, a semaphore is also present to protect from race conditions. A prerecorded video is sent in the experiments instead of a live one; however, this creates a problem where the entire video can be sent in less than a second, while its length is of multiple seconds. In order to simulate the real-time streaming of the video, a delay of 1/video_frames_per_second seconds is introduced between each RTP message sent.

```
1   //RTP message loop
2       while (av_read_frame(codec_format, packet) >= 0)
```

```
3       {
4           // check if packet is from video stream
5           if ((*packet).stream_index == 0)
6           {
7               while (thread_lock)
8                   RTPTime::Wait(RTPTime(0, 100));
9               thread_lock = true;
10              status = session.SendPacket(packet->data, packet->size);
11              cout << "Sent Frame\n";
12              thread_lock = false;
13              cout << i++ << endl;
14
15              // simulate real time streaming, wait 1/(fps) seconds
16              RTPTime::Wait(RTPTime(1, frame_delay_micro));
17
18              av_packet_unref(packet);
19          }
20          if (status < 0)
21          {
22              std::cerr << jrtplib::RTPGetErrorString(status)
23              << std::endl;
24              exit(-1);
25          }
26      }
```

5. **Cleanup** - After all messages are sent, the code frees used data structures, closes files, streams and codecs, and joins with the RTCP thread. It also prints to stdout the amount of time in seconds that was needed to execute the code.

4.2 5G Access Modem

4.2.1 Hardware and OS

The smartphone connected to the Raspberry Pi 4 is a Xiaomi MI 10 5G running the Android 10 OS (API level 29). The critical factor of this piece of hardware is its ability to connect and use 5G cellular networks, serving as an interface for 5G networks to the nomad node.

4.2.2 App description - UI Component

The app was developed for Android devices, specifically for the API level 29 or above (Android 10 and onward). The app does not make use of the Android 11 API functions explicitly made for



Figure 4.3: Xiaomi MI 10 5G

5G. The reason for this is due to the available hardware for the experiments, which consisted of 5G phones with Android 10. Upgrading these phones to the latest OS would cause them to lose their 5G functionalities.

There is only one screen on the app composed of 3 elements: table, transmit button and refresh button. The table displays the values of three network indicators: Rx throughput, Tx throughput and Signal Strength of the current network. The refresh button refreshes the table with the latest values. The transmit button starts sending observed values and geo-coordinates to the 8083 local port; pressing the transmit button again stops the data transmission.

The table values are defined through the use of the *Network Capabilities* Android class functions *getLinkDownstreamBandwidthKbps*, *getLinkUpstreamBandwidthKbps*, *getSignalStrength*. Pressing the refresh button calls these functions again and updates the activity with the returned values.

The app was developed using *Kotlin* programming language and libraries and *Android Studio*. The elements present on the screen consist of *TextViews* and *FloatingActionButtons*. The app contains only one activity (activity_main) whose structure is managed by a *ConstraintLayout*.

4.2.3 App description - Communication with host

The host in this context is the Raspberry PI 4 connected to the device. For the experimental setup to work, the host must access the 5G network monitoring data generated in the device. The device sends this data to the host through the use of sockets and a USB connection.

Before the experiment starts port forwarding must be set up in order for the data to flow from the device to the host. This was achieved with the ADB (Android Debug Bridge) tool, which supports the following command to redirect the data from the device port, to the host port (*adb reverse tcp:<device_port> tcp:<host_port>*). The attempt at creating the socket connection is

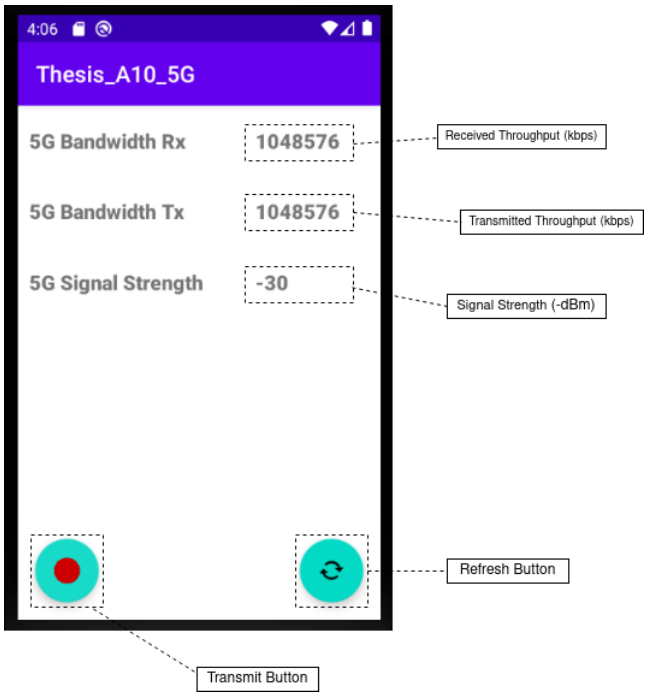


Figure 4.4: Labeled monitoring app screen

made on app start, even if after that no data is transmitted, as such, the host should be already listening for connections when the app is starting. If the socket creation fails due to there being no listener at the given port, resulting in a ECONNREFUSED error, the UI elements of the app can still be accessed.

The device then starts periodically sending the network monitoring data, the current location of the phone and the location accuracy in a string object, where values are comma separated. This data is sent to the local 8083 port and the ADB redirects it to the host's own 8083 port which is being listened by the central manager. The app will send updated values to the host each second, however, the transmission of these new values will not update the app UI.

4.3 Edge Station

4.3.1 RTP and RTCP messages

The main responsibility of the edge station is receiving and processing all the information generated on the nomad nodes. This information comes in two types: data messages contain video frames to be processed and follow the RTP format; control messages have network monitoring information and are contained in RTCP APP packets. For this purpose a program in C++ was developed using the *jrtplib* and *libav* libraries, called *RTPReceiver*. The program handles the reception of the RTP and RTCP messages, reconstructing the sent video feed in a .mp4 files and sending the monitoring data to the history station. This is a challenge since the received packets can come out of order if the difference in trip time is significant between RATs. However, since

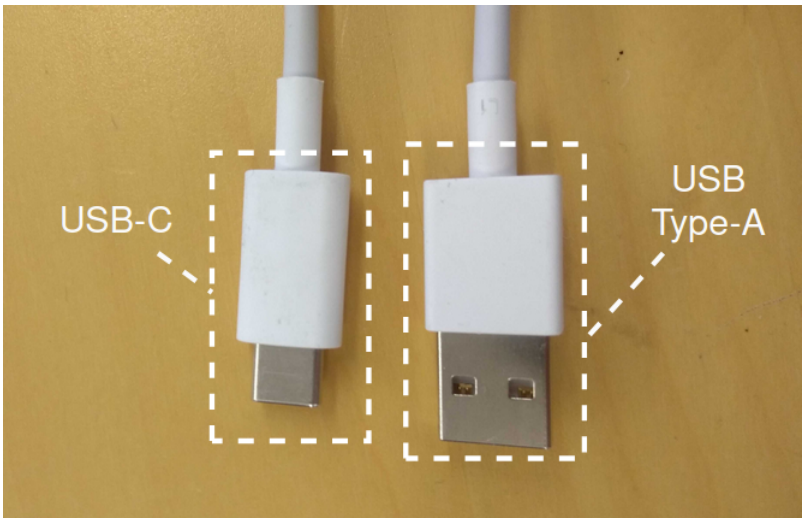


Figure 4.5: Cables

the frames are split up into packets in the IP layer, this problem becomes much easier. The split being done in the IP layer leads to the packets which contain the data relative to a frame to always come from the same RAT, making it much less likely for packets that have data of the same frame to arrive out of order. After the frame is assembled, its "presentation time stamp" (pts) allows for the correct ordering of frames and the successful reconstruction of the video. In cases where one packet is lost, the entire frame associated with said packet is discarded.

The following code segment shows the process of the RTP message reception and writing the received frames to a file.

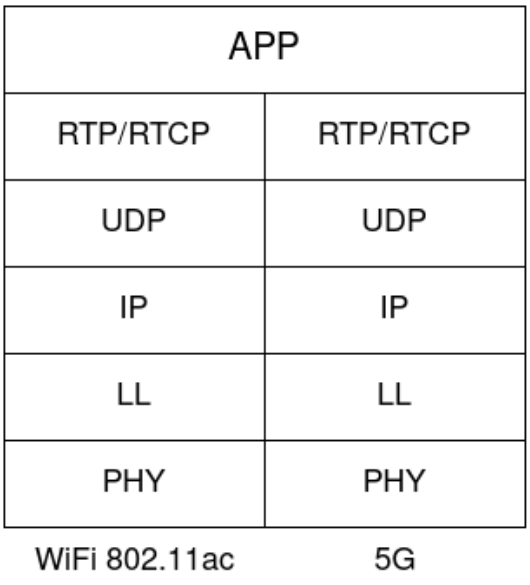


Figure 4.6: Communication layers of the system

```
    //Allocate structures and read packet
    uint8_t *buffer = (uint8_t *)av_malloc(pack->GetPayloadLength());
    memcpy(buffer, pack->GetPayloadData(), pack->GetPayloadLength());

    err = av_packet_from_data(packet, buffer, pack->GetPayloadLength());

    (...) // Error checks for invalid packets or problems during decoding

    if (packet->duration == 0) // check for invalid duration
        packet->duration = 512;

    // Modifying frame fields for duration and
    // position/order in stream/during decoding
    packet->stream_index = outVideoStream->index;
    packet->pts = i;
    packet->dts = i;
    packet->duration = av_rescale_q(packet->duration,
    inVideoStream->time_base,
    outVideoStream->time_base);
    i += packet->duration;
    packet->pos = -1;

    //Write frame to file
    if (av_interleaved_write_frame(destination, packet) < 0){
        cerr << "Error writing frame" << endl;
    }
```

On the reception of an RTCP message, the program has a more straightforward task to execute, sending an HTTP POST request to the history station with the information received in the message. To send these requests, the program makes use of the *curl* library.

The execution of the handling of these messages occurs in distinct threads, being therefore, concurrent. However, as referred to in the previous section, the *RTPSession* class needed also to read and receive messages is not thread-safe, leading to the necessity of implementing a semaphore mechanism. The structure of the program shows resemblance to the code of the central manager.

Regarding the initialization of the RTPSession object the process is very similar, the only difference being the absence of the definition of a destination IP address and port. This process must also open a video file to initialize an *AVFormatContext* and *AVCodecContext* objects, in order to write the received video frames to a file. This might seem an odd way of functioning for a video stream receiver, however during development this obstacle couldn't be surpassed without using too

much of the available time. Therefore, this caveat was introduced in the program where to stream the video, another video with equal characteristics such as encoding format, must be present in the receiver to initialize the two AV objects.

4.3.2 Hardware and OS

The computer used for the edge node consists on a tower PC equipped with an Intel Core i5-9600K processor, 32GB of RAM and a NVIDIA GeForce GTX 2070 graphics processing unit. The essential hardware component is the GPU, which will be responsible of running the object detection libraries in the received video frames. These libraries are developed by NVIDIA and having hardware compatible with them is also vital. It is worth mentioning that the GPU is much more robust than most other GPUs available, specially those found in self-driving cars.

The edge station runs the Ubuntu 18.04 OS. Since the nomad node also runs Ubuntu 18.04 and most software for this project was also developed using Ubuntu it makes sense to maintain this consistency in order to reduce possible incompatibilities.

4.3.3 Communications and FEUP network

The hardware specifications show that the edge server does not have capabilities to communicate using 5G networks or the 802.11ac protocol. Since the nomad node will use these two technologies to communicate with it, this becomes a problem. In order to circumvent this, the following solution was implemented. To support the 802.11ac protocol, the edge station is connected to an access point (AP), which supports the given protocol. The connection is made via Ethernet cable, and with the correct setup, a network with both the AP and the edge station can be created. For the 5G network, the edge station was connected directly to the network of FEUP via, again, an Ethernet connection within the PC and one of the network jacks in the B building corridor. The intricacies of the 5G connection will be further discussed in a future section.

4.3.4 Decision making module

The decision making module is included in the central manager as a part of the software. The module uses the Raspberry PI 4 communication capabilities to request and retrieve the historic performance from the History Station associated with its current location. When the data arrives, it is delivered to the throughput prediction submodule which uses the Arithmetic Mean throughput prediction technique 2.2.3 to predict the future throughput of the network in the area of the nomad node. The module then chooses the RAT which demonstrates a higher throughput. This process of throughput prediction and choice was defined to be periodic; more specifically, it is done each three seconds. The use of three second time intervals between data requests and throughput predictions seemed appropriate to the size of the location cells (radius of 8 meters) and the speed of the nomad node during experiments (average walking speed 5km/h).

4.4 History Station

4.4.1 Main function

The history station is an entity of the system whose primary function consists of storing and providing the network monitoring data collected by the nomad nodes. The history station receives this data from the edge station and makes it available to all nomad nodes through endpoints. This station is composed of two sub-modules: a database and a web server.

4.4.2 Endpoints

The web server provides two endpoints: */history* and */add_history*.

The first endpoint takes three parameters latitude, longitude and startTime; the server will return all entries whose timestamp is after startTime and whose geo coordinates match the provided ones. This endpoint is intended to be used by the nomad nodes whenever they want to retrieve network history data to decide which RAT to choose.

The */add_history* endpoint allows for the addition of new network monitoring entries into the database. This endpoint is intended to be accessed by the edge station whenever it wants to add new monitoring data to the database.

The web server was written using Javascript and the ExpressJs framework.

4.4.3 Database

The database stores all sent data by the nomad nodes and edge node about network monitoring. This translates into entries about registered throughput, the number of bytes sent, signal strength, the RAT chosen and mobility indicators (speed, orientation and movement). Although the current system does not support using mobility factors to predict throughput, this inclusion could prove helpful in future work. The geo-coordinates of the location where this data is generated are also sent, accompanied by the coordinates' accuracy. The precision of the geo-coordinates is four decimal places, translating into areas of around 8 to 10 meters of radius.

The database uses a geolocation index system to return entries by pair of coordinates more efficiently. Each recorded pair of coordinates is associated with an ID; network monitoring values are associated with the ID instead of the coordinates to ease the search process.

The database was implemented using the PostgreSQL DBMS system.

4.4.4 Hosting

As previously stated, the delay of communication between the edge station and the remaining entities is not critical; therefore, it is acceptable to host the history station on the cloud, even though it increases the delay of communications. This module of the system was hosted on a personal computer connected to the 802.11ac network of the experiment. In order to give the PC capabilities to use this network protocol, an AP was setup in Client mode and connected to the

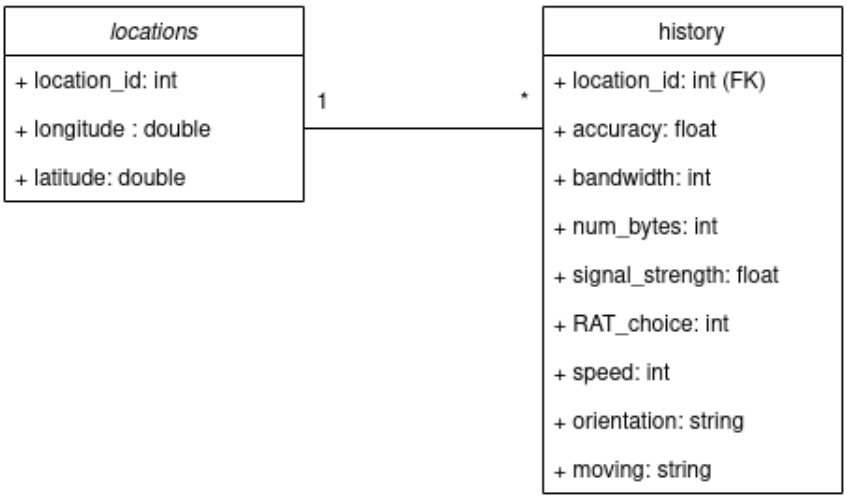


Figure 4.7: Database structure

PC. The computer ran Ubuntu 20.04 OS and both the webserver and database were deployed as Docker containers.

4.4.5 Populating the database

In order to collect data to populate the database, two data collection experiments were conducted. Both experiments were done on the lawn of FEUP, between building B and the departments. The first experiment intended to collect data about the performance of 5G. For that purpose, a setup provided and developed by Hugo Guia was used.

The second experiment had a similar objective, but instead of collecting 5G data, it aimed to collect WiFi 802.11ac information. The monitoring system for this experiment was based on iperf, provided and developed by Rui Meireles and António Rodrigues [26].

4.5 Setting up the RATs

4.5.1 WiFi Access Point

In order to maintain a communication link between edge node and nomad node using the 802.11ac protocol, both entities of the system must support the protocol. Fortunately, the Raspberry PI 4 model comes already equipped with 2.4GHz and 5GHz 802.11 ac wireless support. On the other hand, the PC that serves as edge node does not support this type of communications. To circumvent this problem, a wireless access point (AP) is connected to the edge node. The model of the AP chosen is the UniFi AP AC Lite, which as the name suggests, supports 802.11ac communications.

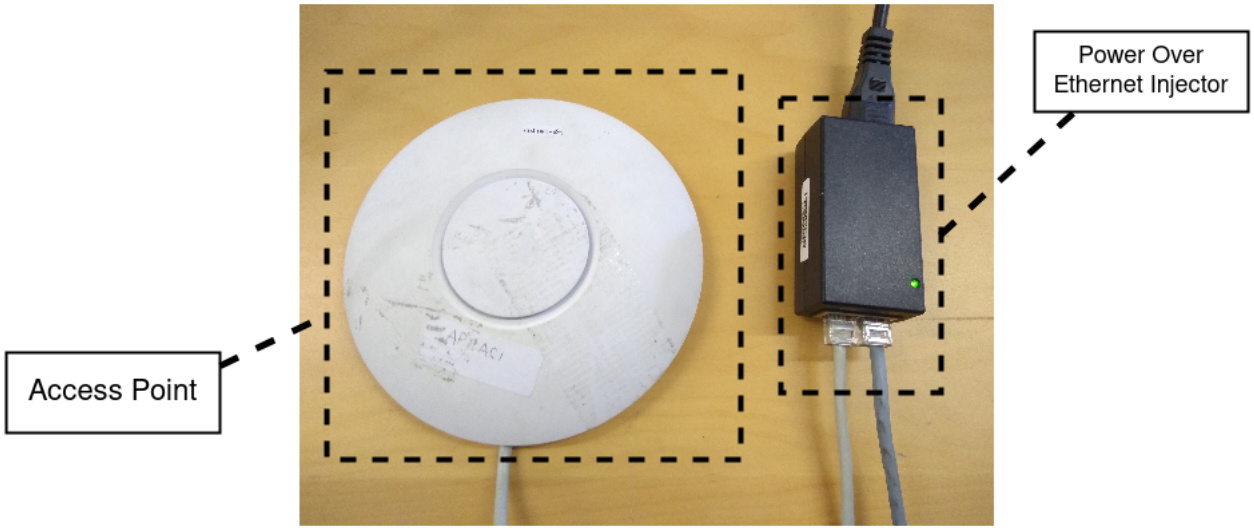


Figure 4.8: AP connected to PoE injector

For the setup of the AP, first, it must be connected to the edge node through a power over Ethernet (PoE) connection. After the physical connection is set up, a network is create by associating an IP address to the edge node (example of IP address definition):

```
sudo ifconfig enp3s0 192.168.1.21/24
```

In the next step the AP IP address is access through the browser which opens the OpenWRT interface for changing the AP definitions. Although these can be changed, the default values are appropriate for the experiments, the only thing needed is to activate the 802.11ac wireless connection on the Wireless tab.

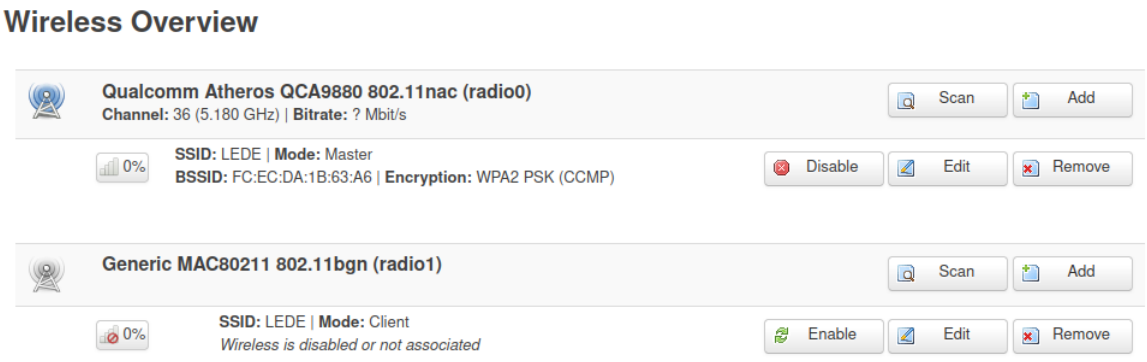


Figure 4.9: OpenWRT interface showing AC enabled

On the Raspberry PI 4 side, the setup is similar to the connection to any wireless network. The tasks needed consist on editing the wpa_supplicant.conf file with the SSID, password, frequency of the network and adding a *disable_vht=0* option. Wpa_supplicant.conf contents example and command:

```
country=pt
ctrl_interface=DIR=/var/run/wpa_supplicant GROUP=netdev
update_config=1

network={
    frequency=5180
    ssid="LEDE"
    psk="rasp_net"
    disable_vht=0
}
```

After editing the file just run the following command and restart the device:

```
sudo wpa_supplicant -c /etc/wpa_supplicant/wpa_supplicant.conf -i wlan0
```

By this stage, a network with the edge computer, the AP and the raspberry PI 4 should exist, where the communications with the RPI 4 use the 802.11ac protocol.

4.5.2 5G gNodeB

The use of a 5G network is possible thanks to the existence of a 5G next generation NodeB (gNodeB) in the premises of FEUP. The equipment is located on top of building A, pointing south. One caveat to using the 5G equipment is that all traffic is redirected first to the Vodafone Core centre in Boavista before being sent back to the FEUP network. The delay added due to this routing of the packets consists in a few milliseconds and is negligible. The equipment's model is the AIR 6488 by ERICSSON.

Chapter 5

Experimental Setup

5.1 WiFi Data Collection Experiment

This experiment aims at collecting 802.11ac WiFi throughput data. This data will be then used to populate the database of the history station during the main experiment. The experiment was conducted in the same place as the other two experiments, the lawn of FEUP between building B and the departments. The AP was placed outside, near the B building entrance to the amphitheatres, commonly known as "cheeses".

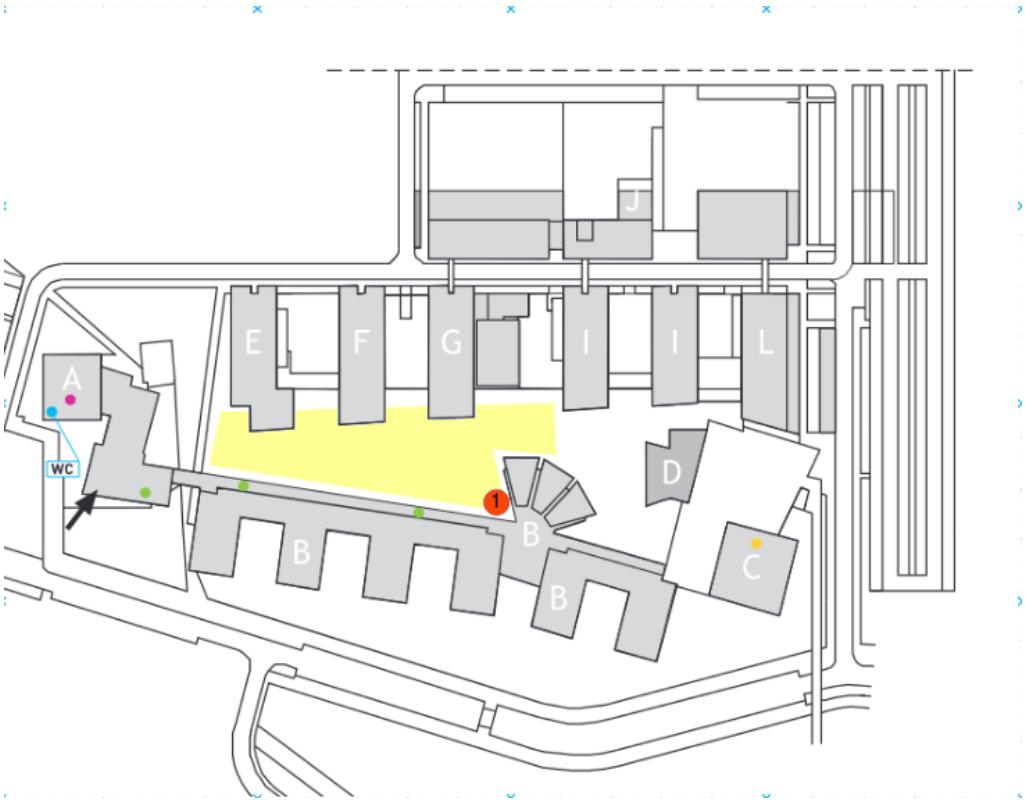


Figure 5.1: WiFi Data Collection Experiment Diagram

In the previous figure, the red circle numerated with a "1" indicates the place of the AP, while the yellow area shows the data collection space. The choice for the AP placement comes from the fact that it needed to be nearby a power source to function; inside the B building, there are multiple power sockets. The space for the experiment encompasses different situations to test the throughput variation. The situations include a variation of distance, the inclusion of small obstacles, such as trees or the departments' supports and large obstacles, which are the amphitheatres.

The experiment's setup required three elements: the access point, the nomad node and a PC. The access point needed to be connected to the PC and to a power source through PoE and have the 802.11ac radio-enabled; the steps to this were described in 4.5.1. The only difference is its placement; the AP is placed at roughly two meters of altitude to improve signal strength and reception in the area; to achieve this, an improvised structure of small wooden tables and a cardboard box was constructed. The nomad node, as described, consists of a Raspberry PI 4 connected to a Xiaomi MI 10 5G smartphone and a power bank. In this module, two programs are run during the experiment: *SocketReceiver* and *Producer*.

SocketReceiver creates a socket connection to the smartphone and listens to the messages received, writing them to a file, accompanied by a Unix timestamp. The app sends a pair of geo-coordinates, and their accuracy each second.

Producer is a program of the authorship of António Rodrigues and Rui Meireles [26], based on iperf, it is used to generate traffic that will be sent and "consumed" by another program, the *Consumer*, which is also of the authorship of the referred researchers. This traffic is then used to extract information about the network's performance, indicators such as throughput, the number of bytes received and mean inter-arrival time between packets. The Producer was slightly modified in order to save the monitoring data to a file, instead of just printing to *stdout*. All entries are accompanied by a timestamp; this way, the data generated by the *Producer* can be matched to the geo-coordinates of the nomad node at the given moment.

During the experiment the elements of the nomad node are assembled on a wooden tray, which is carried through the marked yellow zone. The elements are connected through USB cables.

Lastly, the PC is responsible for hosting the *Consumer* and managing the Raspberry PI 4 and the AP. Since the PC is directly connected to the AP through an Ethernet connection, it does not need to support the 802.11ac protocol to communicate with it or the network members. The connection to the nomad node is achieved through *SSH*, and the connection to the AP since the Raspberry PI 4 is connected to the AP network.

The *Consumer* program is similar to *SocketListener*; it listens to communications in a given port; however, instead of just reading the messages, it also calculates specific performance indicators, similar to the *Producer*. The *Consumer* was also slightly modified in order for the results to be saved to a file.

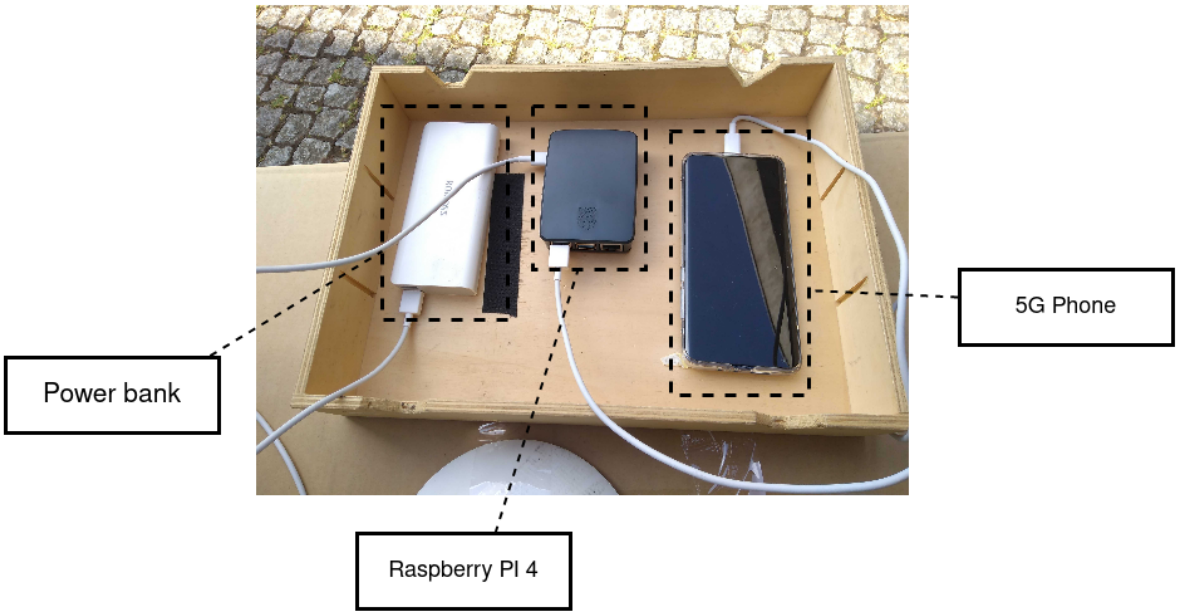


Figure 5.2: Nomad node elements assemble on a tray

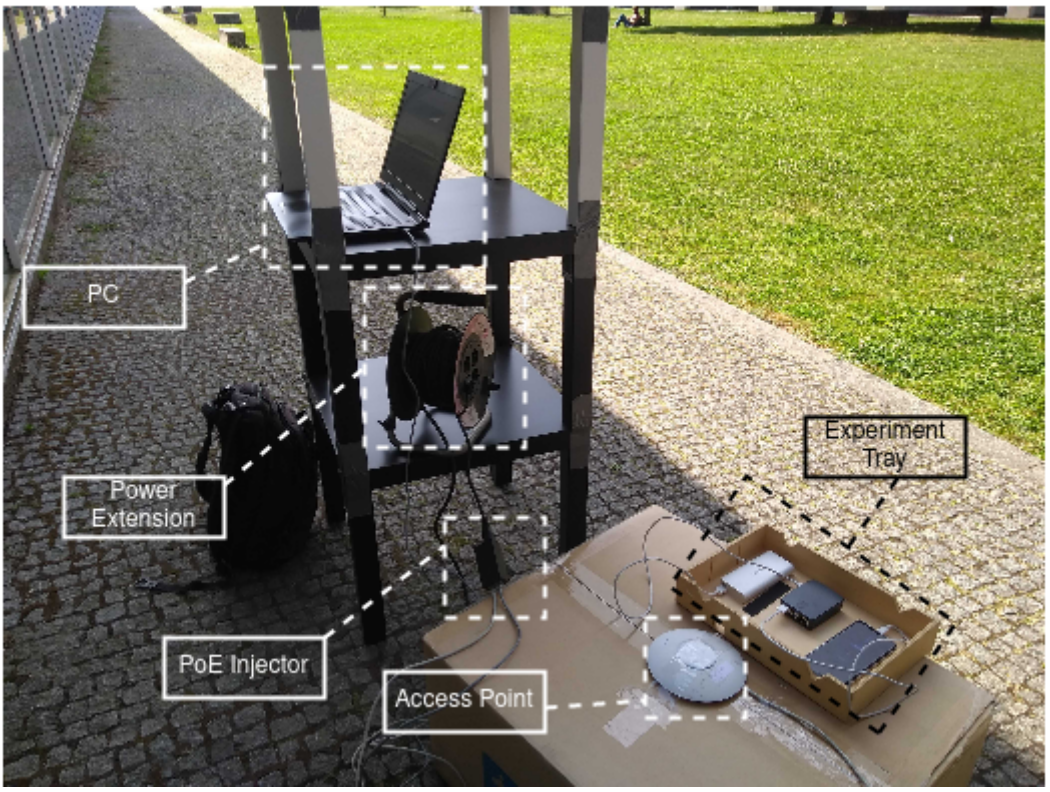


Figure 5.3: Structure and experiment components

The checklist for this experiment is the following:

- Raspberry PI 4
- Power bank
- Xiaomi MI 10 5G
- PC with Ethernet port
- 3 Ethernet cables
- 2 USB Type A - USB C cables
- Unifi WiFi AP AC Lite
- PoE injector
- Power extension
- 2 meter structure
- Wooden tray (optional)

5.2 5G Data Collection Experiment

The setup for the 5G data collection experiment is quite simpler than the previous one, it only needs a rooted 5G smartphone with the TEMS Pocket and TEMS Status applications. Before starting to collect data one must check if all TEMS services are active on the TEMS Status app. If it is not the case, activate them using the *"Start all onDevice services"* option. After this step, on the TEMS Pocket app, force the app to only use New Radio (5G) NSA connections. This step forbids the app of using other RATs while the logging is active. Now, data collection can start by downloading, a preferably large file, in our experiments we used an .iso image file for the *CentOS* and turned on logging on the TEMS Pocket app.

The location for the experiment was the same as the WiFi experiment, the lawn of FEUP between building B and the departments. The area where data was collected was slightly different from the previous experiment. The area further behind the amphitheatres and the inner staircase of the Chemical Engineering department were both included on the data collection area, as shown in the following figure. The green circle labelled with a two represent the location of the 5G gNodeB.

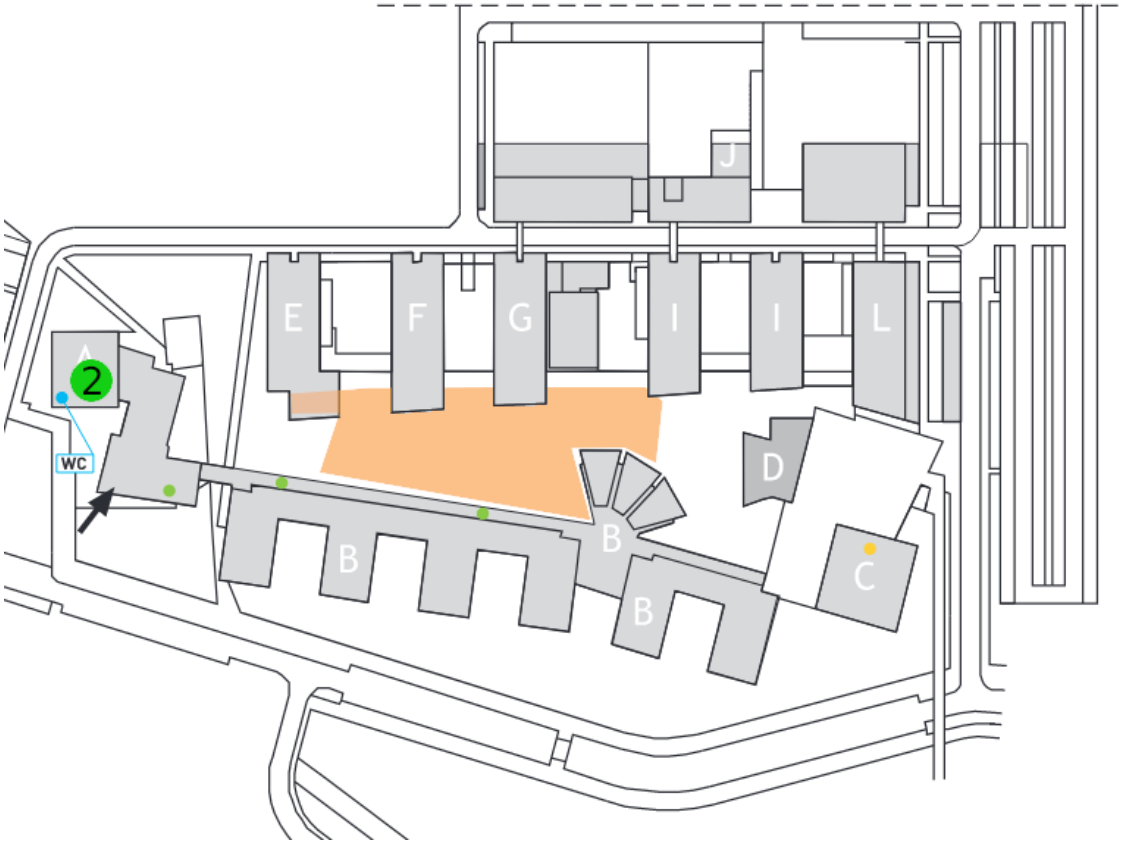


Figure 5.4: 5G experiment data collection area

5.3 Main Experiment

The main experiment of this work tests the functioning of each of the developed modules. It is reasonably similar to the Wifi data collection experiment in the matter of setup and location. The main experiment is also conducted on the FEUP interior lawn, with the 802.11ac AP at the amphitheatre entrance and the 5G gNodeB on the top of building A. However, there are two other modules present in this experiment: the history station and the edge node. The history station is hosted on a personal computer placed in the same location as the WiFi AP. This module is connected to the system network through WiFi; thus, requests to this module must be sent using this interface. The edge node is a tower computer and is placed inside the B building of FEUP. In order to be accessible through the 5G RAT, the computer must be connected to the FEUP network. The solution for this challenge was connecting the computer to one of the corridor's internet sockets and requesting a public IP address for that same socket. The edge node is also connected to the WiFi AP through an Ethernet connection to receive data through this RAT.

In this experiment, contrary to the previous ones, there are multiple networks at play. In fact, there is a total of 5 networks present. The edge node is connected to two of these; it is connected to

the FEUP network in order to be accessible by a public IP address, which is needed for data transmission through 5G and is also connected to the 802.11ac access point, denominated *802.11ac AP*. Both of these connections are wired and done through Ethernet cables. For the history station to be accessible by WiFi, it needs to be connected to the access point. The personal computer used did not support the 802.11ac standard, so another access point is connected to the history station by Ethernet connection to circumvent this limitation. The access point is set up in *Client Mode* and connects to the network provided by 802.11ac AP in order to allow communications between the members of the network and the history station. The mobile node can connect to the 802.11ac AP's network since its hardware natively supports the 802.11ac protocol; however, to connect to the edge node through 5G, two other networks must be set up. A network between the Raspberry Pi 4 and the 5G phone is set up, where the connection between the two is a wired USB connection. The last network connects the 5G phone to the Internet through the 5G gNodeB.

Still on the topic of network setup for the experiment, before the experiment can start, it is paramount to set up the correct gateways, static routes and IP addresses. The needed routes and gateways are the following, and the IPs are described in 5.5:

- Define edge node default gateway as 10.227.246.254
- Define routing of packets with 192.168.2.0/24 as a destination to 192.168.1.144 in 802.11ac AP
- Define static routes from Raspberry Pi4 and edge node to history station: 192.168.2.0/24 through gateway 192.168.1.1
- Define default gateway of Raspberry Pi 4 as 5G Phone IP address: 192.168.42.129
- Define static route of packets with 192.168.1.0/24 as a destination to 192.168.2.1 in history station

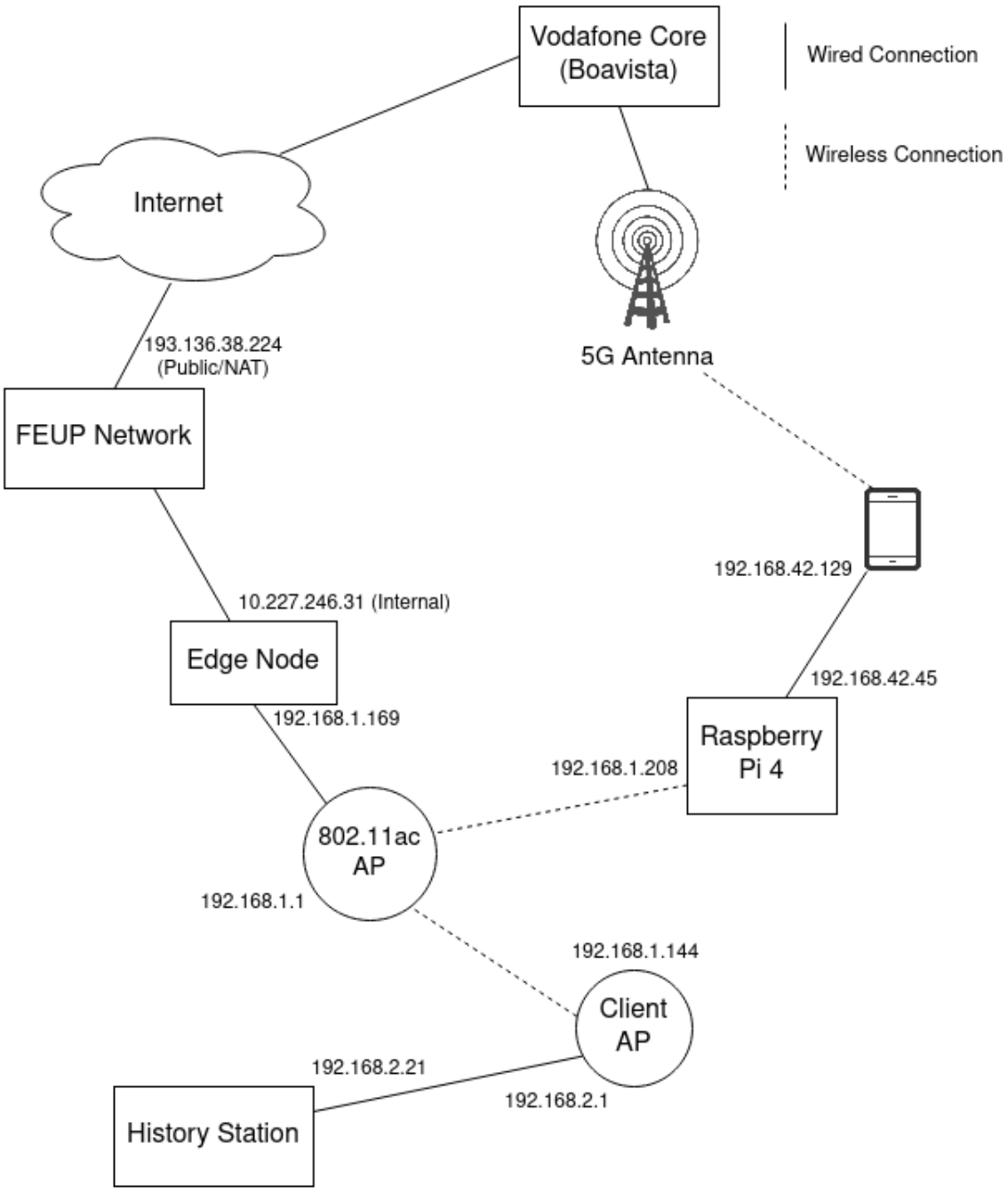


Figure 5.5: System network map

After the network setup is complete, the following must be done before starting the experiments. On the edge node, start the *RTPReceiver* program. This program was developed specifically for this experiment and runs during the entire experiment. It listens for RTP and RTCP packets on port 6970; the RTP packets contain fragments of video frames and are used to reconstruct the video on the edge node, while RTCP packets contain network monitoring information; these packets are redirected to the history station.

The nomad node uses the same setup as in the WiFi data collection experiment where the Raspberry Pi 4 is connected to a 5G phone and a power bank. Before starting any program, the port forwarding operation using the Android Debug Bridge must be executed. In this experiment, the nomad node, instead of using the *Producer* program, uses the*RTPSendVideo* program also explicitly developed for this experiment. This program opens a .mp4 video file and sends it simulating a real-time capture; in essence, it streams the already recorded video according to its frame rate. Right after starting the RTPSendVideo program, start the 5G monitoring app described in 4.2 and initiate the transmit mode. The app will send to the nomad node its geo-coordinates and monitoring data about the 5G network. The RTPSendVideo program will request network data associated with the received coordinates to the history station and send the monitoring data in the RTCP packets.

Finally, the setup of the history station consists only of starting the docker containers, which can be achieved with the *docker-compose up* command since their startup is already defined in the respective *Dockerfile* and *docker-compose.yml* files.

A total of 8 experiments were conducted using the main experiment setup, although with slight variations between them. In the first four experiments, the setup used a random path for the nomad node, used both available RATs and used the developed decision-making algorithm to decide between the two. Unfortunately, a problem during experiment two invalidated its data. The first four experiments served to validate if the decision-making module worked correctly and if the video streaming part of the system was successful. In experiment five, the setup was equal to the first four; however, in this experiment and the following ones, the nomad node followed the path described in 5.6. In experiment six, only the WiFi RAT was used, in experiment seven, only 5G was used, and in experiment eight, both RATs were used, but the decision-making process was random. The last four experiments' purpose consists in understanding if the decision-making module has a positive impact on the performance of the system.

Regarding the path of the experiments, some observations must be made about it. The "X" in yellow represents the WiFi access point, while the "Y" in red represents the 5G gNodeB. The route takes about 4 minutes to complete, roughly the same duration as the transmitted video; usually, experiences would end between the 8 and 9 marks. The path between 4 and 5 is a staircase that blocks the line of sight for both the access point and the gNodeB. Near point 3, there are some trees that block the line of sight to the gNodeB. Points 6 and 7 do not have a line of sight to the access point. By taking this route, the nomad node takes positions where it has a line of sight for both the access point and the gNodeB, where it only has a line of sight for one of them and where it does not have a line of sight to any of them.

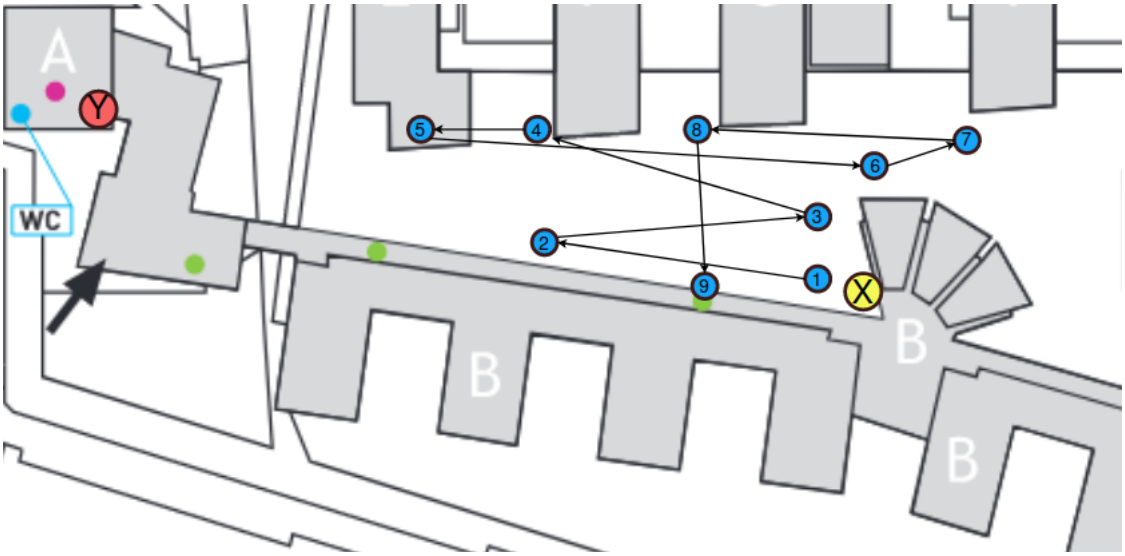


Figure 5.6: Main Experiment Route

Chapter 6

Results and discussion

6.1 Functional Validation

The end results show a functioning system, where all modules achieve their intended purpose.

During both the data collection experiments and the final experiment the system monitoring module was able to collect and properly store relevant data regarding the used networks. The information gathered during the data collection was then successfully used for decision making operations during the final experiment. The data collected consisted on geo-coordinates, their accuracy, number of bytes sent, throughput and signal strength. Although not all of the indicators were gathered at every moment of the experiments, the crucial ones , geo-coordinates and throughput were always monitored and registered. Moreover, incomplete data reflects the reality of any working system.

The decision making module during the final experiments was able to autonomously make decisions on which RAT to use for video streaming. It also changed a previously taken decision whenever the nomad node's position was in a place where the historic throughput data favoured such change. The decision making module, in the space of four minutes, could register around ten RAT changes, depending on the trajectory of the nomad node.

Finally, the RAT management center was able, in every single experiment, to send and receive a video of around four minutes, whose frames were sent through different RATs. In the edge node, the reconstruction of the video would always be successful, even when frames were lost. After the stream ended, the video could be opened and viewed with only slight differences to the original, which in other words show that the RAT management center works as a successful video streaming tool.

Regarding the previous statements, we can conclude that the developed system is able to perform all of the planned functionalities.

6.2 Quality of Received Video

At the end of each experiment run, the video feed received by the edge node would be saved in a file. Although, in runs 1,2 and 5 the video was lost, in the remaining experiment runs the reconstruction of the video feed was successful, and the video could be played afterwards with very few perceivable differences to the original. In this section, we aim to answer the question on whether the received video feed retains satisfactory levels of quality. To evaluate said quality, in a more precise fashion, all video files were compared to the original transmitted video through the peak signal-to-noise ratio (PSNR) metric. The results are presented in the following table:

	Average	Minimum	Maximum
Run 3 (DMA)	37.250	18.676	INF.
Run 4 (DMA)	28.895	12.846	INF.
Run 6 (WiFi)	31.029	12.444	INF.
Run 7 (5G)	28.679	13.966	INF.
Run 8 (Random)	30.616	13.152	INF.
AVG.	30.616	14.217	INF.

Table 6.1: PSNR of Experiment runs in dB

With an average of 30.616 dB we can conclude that the quality loss of the video streaming is good, if we consider 25 dB as the threshold of acceptable [15, 34]. At the same time, the fact that the minimum values of PSNR achieve values below 20 dB show that at certain times during the transmission of the video, the received video looses quality beyond an acceptable point. On the other hand, the PSNR achieving values of infinite, i.e. noise is zero; shows that the video streaming can occur where the received frame is the same as the sent one.

Next we present box plots which show the distribution of the average PSNR values of frames for each experiment run. To generate the graphs values which corresponded to infinite were replace with 990.

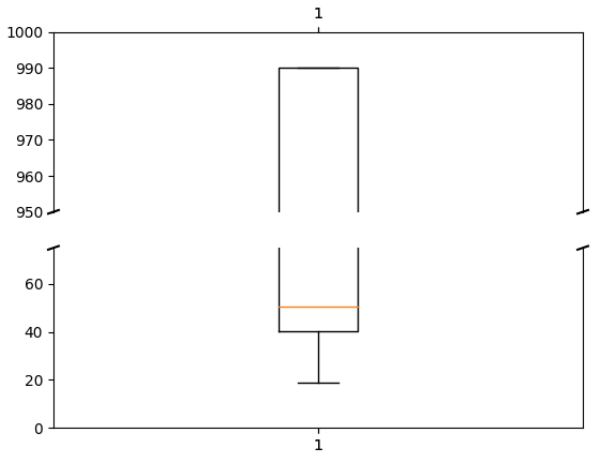


Figure 6.1: Box plot experiment 3 (DMA)

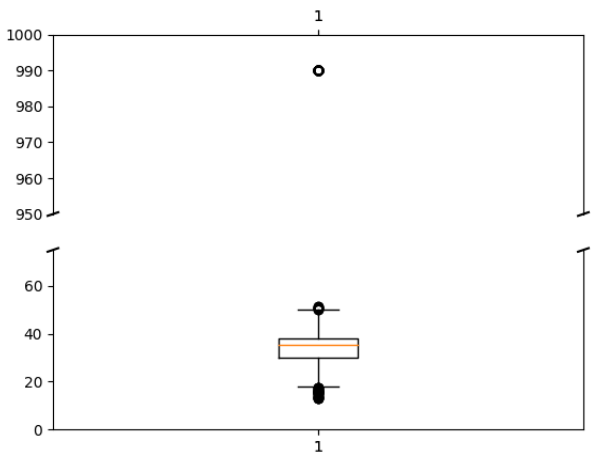


Figure 6.2: Box plot experiment 4 (DMA)

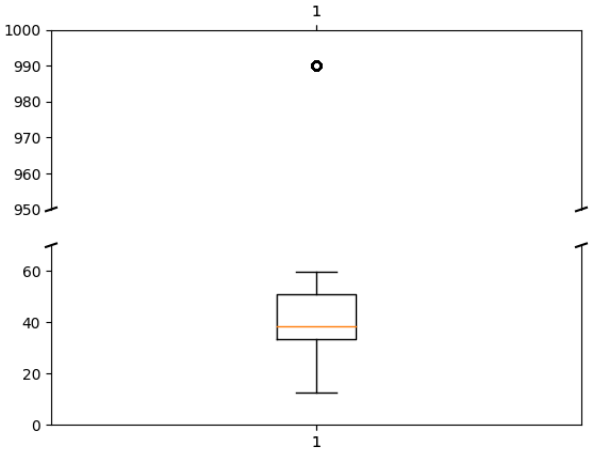


Figure 6.3: Box plot experiment 6 (Wifi)

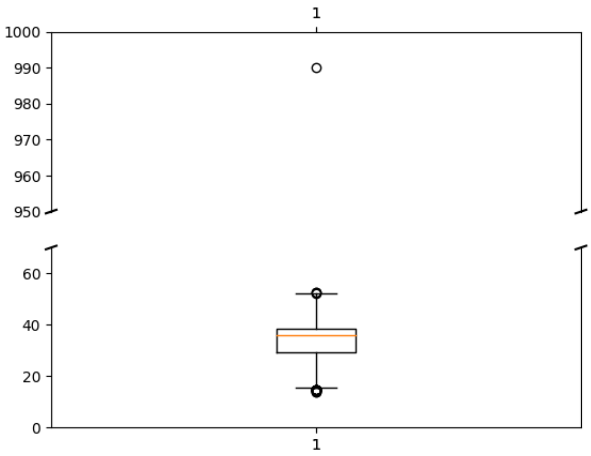


Figure 6.4: Box plot experiment 7 (5G)

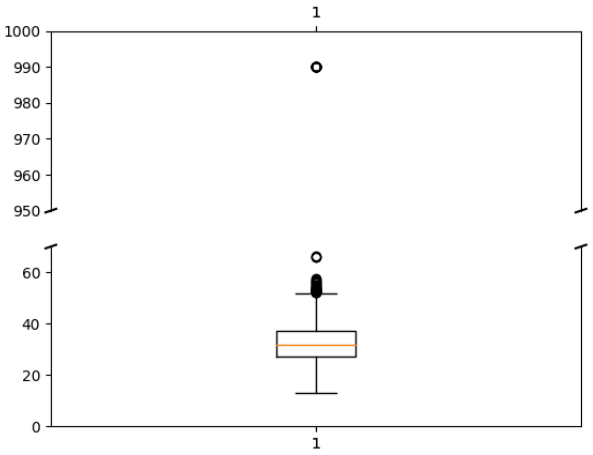


Figure 6.5: Box plot experiment 8 (Random)

It seems clear that the best performing run was number 3 since it achieved the higher values of average PSNR. Interestingly, the performance of runs 3 and 4 is quite different even though both used the same RAT decision, which could show the impact of other factors, such as the route taken in the quality of the transmitted video. The remaining experiments all had similar results, with a possible highlight of run 6, which had a slightly better performance.

6.3 Decision Making Validation

The decision making algorithm, as stated in previous chapters, makes a decision based on the throughput history associated with a given location. To validate the correct functioning of this module, we need to make sure that the decision taken during experiments matches the data available in that given moment. In order to achieve that, we first need to understand which RAT decision

is correct for each area of the experiment, based on the available history. The correct RAT will be the one which demonstrates a higher average throughput in a given area. The following annotated heatmaps will show the average throughput values on each area of the experiments depending on the RAT used. The Y axis of the map shows the longitude of the zone while the X axis shows the latitude; the values consist on the throughput history represented in Mbit/s.

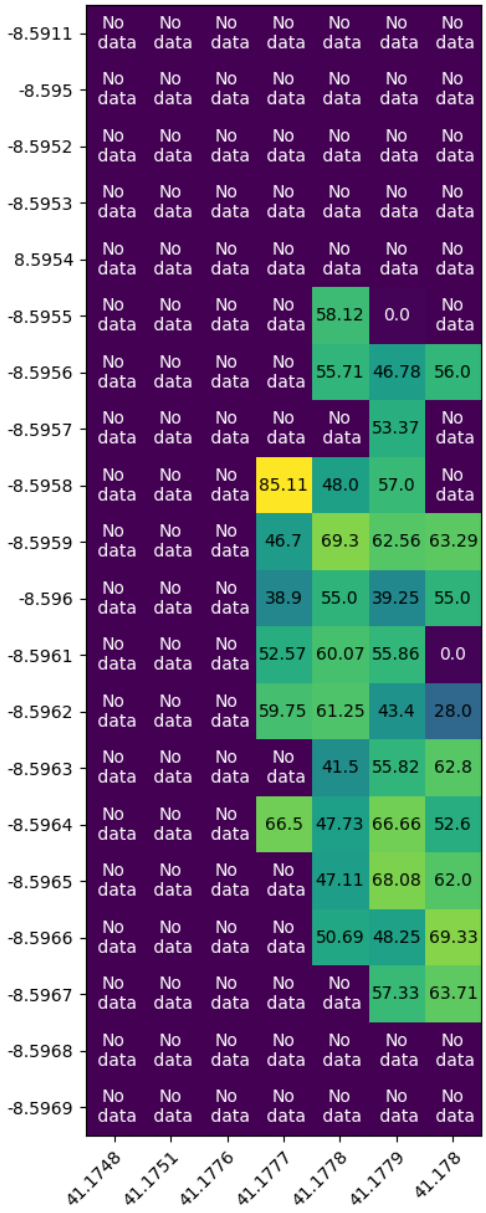


Figure 6.6: Average bandwidth values of 5G

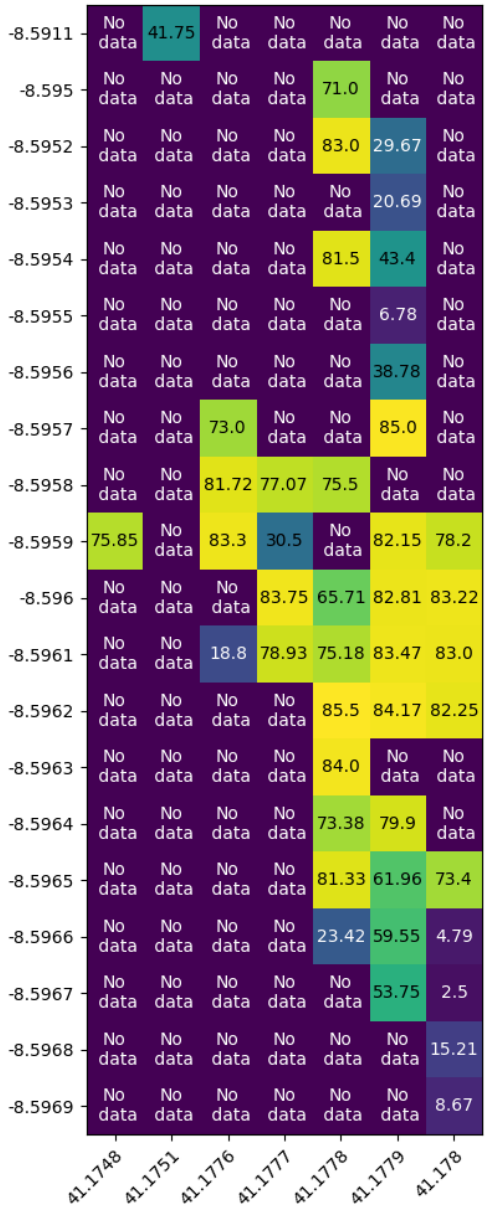


Figure 6.7: Average bandwidth values of WiFi

If we then compute the difference between the two previous heatmaps, we can determine which is the best RAT choice per area and compare it with the choices made by the decision making module. In 6.9 the areas with "Both" represent places where the decision making algorithm chose one RAT in some situations and the other RAT in the remainder of moments. The RAT which was chosen in most situations is indicated inside the parentheses.

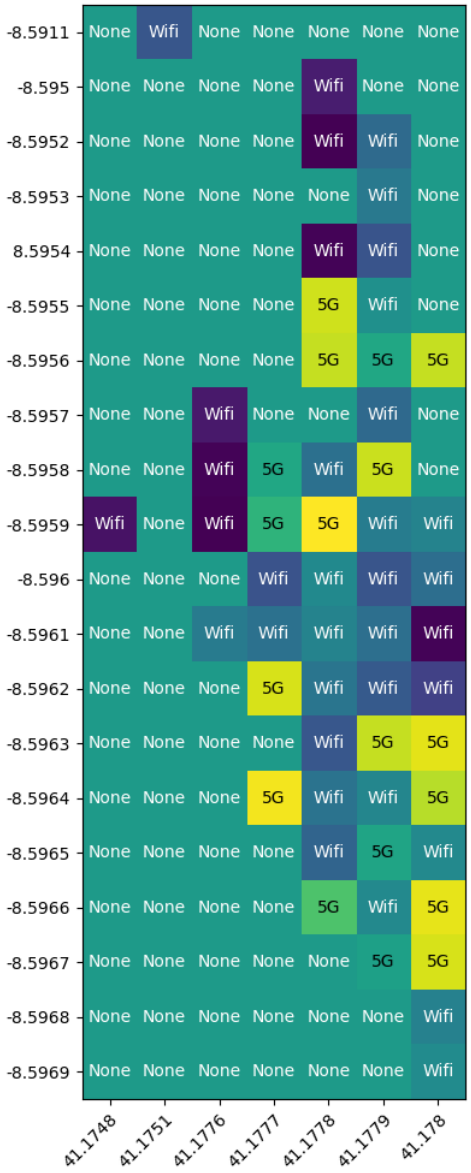


Figure 6.8: Correct choices of RAT according to throughput history



Figure 6.9: RAT choices made by the decision making module

As we can see from graphs 6.8 and 6.9 the decision making module behaves reasonably within the expected, choosing most of the times the RAT, which has a higher average historic throughput. There are, however, some cases where the module sometimes chooses one RAT and some-

times the other, even though the history data does not change. These decisions only happen in "frontier spaces", which separate zones associates with different RAT choices. We believe that these decisions that do not reflect the throughput history come from inaccuracies relative to the geo-coordinates. These inaccuracies lead to the module possibly retrieving data associated with adjacent nearby areas, resulting in wrong predictions, which then can be translated into wrong choices for the location it is currently in. It is still worth noting that even in these cases where the module chooses more than one RAT in a given location, it still chooses the correct RAT most of the time.

Chapter 7

Final Remarks

7.1 Conclusion

With this thesis, the objective was to develop a system that enabled computation offloading through multiple access technologies while monitoring communications to enable a decision making module. We believe the objectives of this work were met since the resulting system works and its modules meet the validation criteria. The experiments conducted show that the multi-RAT forwarder and the decision making module, although they do not function perfectly, function to a satisfactory degree. In other words, the multi-RAT forwarder streams video, which not only suffers an acceptable level of quality loss it also is reconstructable on the edge node. The decision making module autonomously retrieves information about a given RAT and makes a decision, which in most cases consists of the correct choice, given the available history data.

However, the system still has many points that can be improved or functionalities that could be added. However, the functionalities aimed with this work were implemented, and ideas for other improvements are present in the following section.

7.2 Future Work

The follow up of this work can go in many different directions; in this section, we will enunciate some of them.

As stated in chapter 2, there are many ways of predicting throughput in networks. In this work, the methods used were history-based, where values of previously registered throughputs are used in tandem with simple formulas to calculate the future throughput. There is, however, another technique with better-registered performance, which consists of an AI-based approach. One of the main hurdles of machine learning and AI is the need for considerate amounts of data, and this was a critical factor not to follow this approach in this work. Nevertheless, this work developed and used data collection tools that could help in surpassing this obstacle in the future. Developing and

adapting machine learning models to the context of throughput prediction for RAT choice could show improvements relative to the results of this work.

One pivotal design choice of this work was using already encoded (compressed) videos for transmission. This not only removes a considerable processing delay from the system, but it also does not reflect what happens in a real-life scenario, where frames are captured and processed in RAW format. Follow-up work could include the encoding and decoding of a RAW video feed and observe how the introduction of these operations influences the system's performance.

One idea that came up during development was the reduction of the components of the nomad node to become only the 5G smartphone. Following this design, all communications, network monitoring and RAT decision come from the smartphone. Thus both the Raspberry PI 4 and the battery pack can be excluded from the nomad node. This is only possible because the smartphone model supports both 5G and 802.11ac communications.

In this work, the user node is static; although it can be placed in multiple places, it does not move, and it does not consider movement for the throughput prediction. Since this work is inserted in the context of self-driving cars and VRU protection, it is vital to consider the movement of this node. In the future, it would be interesting to see an expansion of this system where information about the user's motion is taken into consideration in the decision-making process. It is worth mentioning that some decisions in this work were already aimed at this development, for example, the *history* table of the database.

As referred to in chapter four, there is a caveat to the developed system. Although this workaround is not critical, it does not reflect the functioning of a real-life system for offloading video processing. Therefore, the system could be further developed so that the information needed to open and write the received frames to a file is transmitted from the user node. This way, there is no need to have a video file similar to the received stream on the edge node to extract the needed information.

In this work, the WiFi standard used was 802.11ac, yet, there many other standards that could still be explored, with their advantages and drawbacks. It would be an exciting project to experiment with other standards, understand their differences in performance, and the best situations for each one of them. Furthermore, the decision making process could be expanded to make a decision not only between WiFi and 5G but also choose which of the WiFi standards available could lead to a better throughput in a given situation.

7.3 Contributions

In this work various contributions were made to the fields of throughput prediction and network monitoring, these contributions are enumerated in this section.

The first contribution consists on the system design proposed. The design can be seen as a possibility on the realm of heterogeneous networks monitoring on the context of video streaming. Implementation choices of this design can vary and diverge from the ones taken in this work. The design defines the needed entities of the system and their responsibilities, dividing them over

modules which focus on a particular task of the system. The design is decentralized, allowing for an easily scalable solution to an increasing number of users and focuses on the reduction of detection delay of offloading of computer vision processing. The design organizes the communications relative to video streaming and the ones associated with the network monitoring in two communication planes.

On the implementation side of this work, a video streaming application was developed. The application used RTP to transmit the video feed and the RTCP to send monitoring data. The differentiator factor of this module consists on the possibility to use multiple RAT to stream the video. The streaming transition between RATs is succinct and near seamless. The application can be used on both static and mobile nodes.

A decision making module was also developed and integrated onto nomad nodes. The module was autonomous and able to make predictions on the which RAT would have the better throughput. Although there were only two available RATs during the experiments, these module is capable of making predictions in environments with more than two RATs and with other RATs than 5G and the 802.11ac protocol.

References

[1] 6.1 RTCP Packet Format. Connected: An Internet Encyclopedia. Retrieved June 17, 2021, from https://www.freesoft.org/CIE/RFC/1889/14.htm.

[2] 6.6 APP: Application-defined RTCP packet. Connected: An Internet Encyclopedia. Retrieved June 17, 2021, from https://www.freesoft.org/CIE/RFC/1889/33.htm.

[3] Achieving industrial automation protocols with 5G URLLC. (2020) ericsson.com. Retrieved February, 2021 from https://www.ericsson.com/en/cases/2020/accelerate-factory-automation-with-5g-urllc.

[4] Ffmpeg tutorial. (2015, February). dranger.com. Retrieved May 25, 2021, from http://dranger.com/ffmpeg/tutorial01.html.

[5] Libav. (2018, February 12). Libav. Retrieved May 25, 2021, from https://libav.org/.

[6] TEMS Pocket - Mobile Application Network Testing | Infovista. Retrieved July, 2021 from urlhttps://www.infovista.com/tems/pocket.

[7] K. Dantu A. Ali, Z. Hashemifar. Edge-slam: Edge-assisted visual simultaneous localization and mapping. *The 18th Annual International Conference on Mobile Systems, Applications, and Services*, June 2020.

[8] Abdeldime M.S. Abdelgader and Wu Lenan. The physical layer of the ieee 802.11p wave communication standard: The specifications and challenges. *Proceedings of the World Congress on Engineering and Computer Science 2014 Vol II*, oct 2014.

[9] H. Parmar an M. Thornburgh. Adobe's real time messaging protocol, December 2012.

[10] D. Carels, Wim Vandenberghe, Ingrid Moerman, and Piet Demeester. Architecture for vulnerable road user collision prevention system (vru-cps), based on local communication. oct 2011.

[11] S. Chandrashekar, Andreas Maeder, Cinzia Sartori, Thomas Höhne, Benny Vejlgaard, and Devaki Chandramouli. 5g multi-rat multi-connectivity architecture. oct 2016.

[12] A. Aguiar G. Pereira, P. M. d'Orey. Poster: Cooperative perception platform for intelligent transportation systems.

[13] X. Gong. Delay-optimal distributed edge computing in wireless edge networks. February 2020.

[14] O. Sunay L. Peterson. 5g mobile network: A systems approach. https://5g.systemsapproach.org, Retrieved Februrary 2021, from https://5g.systemsapproach.org.

[15] Xiangjun Li and Jianfei Cai. Robust Transmission of JPEG2000 Encoded Images Over Packet Loss Channels. In *2007 IEEE International Conference on Multimedia and Expo*, pages 947–950, July 2007. ISSN: 1945-788X.

[16] Jori Liesenborgs. j0r1/JRTPLIB. (2016) github.com. Retrieved July, 2021 from https://github.com/j0r1/JRTPLIB, July 2021. original-date: 2016-12-03T15:16:23Z.

[17] Shih-Chieh Lin, Yunqi Zhang, Chang-Hong Hsu, Matt Skach Md E. Haque, Lingjia Tang, and Jason Mars. The architectural implications of autonomous driving: Constraints and acceleration.

[18] Salvatore Loreto and Simon Pietro Romano. *Real-Time Communication with WebRTC: Peer-to-Peer in the Browser*. "O'Reilly Media, Inc.", April 2014. Google-Books-ID: RP-NfAwAAQBAJ.

[19] Zhaoming Lu, Tao Lei, Xiangming Wen, Luhan Wang, and Xin Chen. Sdn based user-centric framework for heterogeneous wireless networks. June 2016.

[20] Ioannis Mavromatis, Andrea Tassi, Giovanni Rigazzi, Robert J. Piechocki, and Andrew Nix. Multi-radio 5g architecture for connected and autonomous vehicles: Application and design insights. feb 2018.

[21] R. Meireles. Exploring wi fi network diversity for vehicle to infrastructure communication. (2020, December 20). Youtube. Retrieved Februrary 2021 from https://youtu.be/IREMIGV4XLc.

[22] R. Meireles, António Rodrigues, Andrei Stanciu, Ana Aguiar, and Peter Steenkiste. Exploring wi-fi network diversity for vehicle-to-infrastructure communication. 2020.

[23] Quang-Huy Nguyen, Michel Morold, Klaus David, and Falko Dressler. Car-to-pedestrian communication with mec-support for adaptive safety of vulnerable road users. *Elsevier*, oct 2019.

[24] Joerg Widmer Nicola Bui, Foivos Michelinakis. A model for throughput prediction for mobile users. *European Wireless 2014*, 2014.

[25] Mostafa Ammar Qi He, Constatine Dovrolis. On the predictability of large transfer tcp throughput. 2005.

[26] Antonio Damião Rodrigues. adamiaonr/wifi-vehicles. Retrieved June 10, 2021 from https://github.com/adamiaonr/wifi-vehicles, May 2020.

[27] A. Sakuraba, Goshi Sato, Noriki Uchida, and Yoshitaka Shibata. Performance evaluation of improved v2x wireless communication based on gigabit wlan.

[28] Alassane Samba, Yann Busnel, Alberto Blanc, Philippe Dooze, and Gwendal Simon. Instantaneous throughput prediction in cellular networks: Which information is needed? *2017 IFIP/IEEE International Symposium on Integrated Network Management*, 2017.

[29] Johan Scholliers, Marcel van Sambeek, and Kees Moerman. Integration of vulnerable road users in cooperative ITS systems. *European Transport Research Review*, 9(2):1–9, June 2017. Number: 2 Publisher: SpringerOpen.

[30] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. Edge Computing: Vision and Challenges. *IEEE Internet of Things Journal*, 3(5):637–646, October 2016. Conference Name: IEEE Internet of Things Journal.

[31] S. Singh, Harpreet S. Dhillon, and Jeffrey G. Andrews. Offloading in heterogeneous networks: Modeling, analysis, and design insights. *IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS*, may 2013.

[32] Ralf Steinmetz and Klara Nahrstedt. *Multimedia Systems*. X.media.publishing. Springer-Verlag, Berlin Heidelberg, 2004.

[33] Gokulnath Thandavarayan, Miguel Sepulcre, and Javier Gozalvez. Generation of cooperative perception messages for connected and automated vehicles.

[34] N. Thomos, N.V. Boulgouris, and M.G. Strintzis. Optimized transmission of JPEG2000 streams over wireless channels. *IEEE Transactions on Image Processing*, 15(1):54–67, January 2006. Conference Name: IEEE Transactions on Image Processing.

[35] Yi Wang, Jian Li, Lei Huang, Yao Jing, Andreas Georgakopoulos, and Panagiotis Demestichas. 5G Mobile: Spectrum Broadening to Higher-Frequency Bands to Support High Data Rates. *IEEE Vehicular Technology Magazine*, 9(3):39–46, September 2014. Conference Name: IEEE Vehicular Technology Magazine.

[36] M. Levorato Y. Matsubara. Neural compression and filtering for edge-assisted real-time object detection in challenged networks. *25th International Conference on Pattern Recognition*, July 2020.

[37] Jack Y. B. Lee Yan Liu. An empirical study of throughput prediction in mobile data networks. 2015.