Rui Miguel Lopes Gaspar de Matos

Design and development of a communication platform with the capability of monitoring the communication quality in a drone separation system

for support of U-Space Separation Management Service

Dissertação no âmbito do Mestrado em Engenharia Informática/Sistemas Inteligentes orientada pelo/a Professor/a Doutor/a Tiago Cruz e apresentada à Faculdade de Ciências e Tecnologia/Departamento de Engenharia Informática.

Setembro de 2022

Faculty of Sciences and Technology

Department of Informatics Engineering

Design and development of a communication platform with the capability of monitoring the communication quality in a drone separation system

for support of U-Space Separation Management Service

Rui Miguel Lopes Gaspar de Matos

Dissertation in the context of the Master in Informatics Engineering, Specialization in Intelligent Systems advised by Prof.Tiago Cruz and Prof. César Teixeira and presented to the Faculty of Sciences and Technology / Department of Informatics Engineering.

September 2022

1 2 9 0

UNIVERSIDADE Ð COIMBRA

This page is intentionally left blank.

Abstract

Unmaned Aerial Systems (UAS) or drones as they are most commonly known, are an emerging concept that progressively has entered the publics conscience, having the capability to provide never seen services and fixing some existing problems.
To seize this opportunity several different groups are creating projects with the intent to define rules, starting experimentation and constructing basic system architecture; one of these projects is BUBBLES.
BUBBLES intends to create safe spaces around the UASs, similar to bubbles, allowing drones to fly efficiently and safely as well as avoiding problems with the already existing manned aviation. However, since these bubbles are relative to the drones, there is a need for the telemetry to be constantly sent to the BUBBLES system and the system needs to be able to define the separation, taking into account the existing context, such as network problems.
This document describes the planned work done achieve the main goal of detecting and minimizing network problems by utilising artificial intelligence, trying a multitude of models and techniques, comparing each of them and concluding which is the most appropriate.
Furthermore the process of the dataset acquisition, the environment description, as well as problems and solutions throughout the development of the work and the analysis of the results will also be present in this report.
Finally, this work will try to propose several more practical implementations based on the results that were previously discussed.

Keywords

UAS; ANN; Drones; BUBBLES; Drone networks; Bayes Networks; Time Series;

This page is intentionally left blank.

Resumo

Os Sistemas Aéreos Não Tripulados(UAS) ou drones como são mais conhecidos, são um conceito emergente que progressivamente tem entrado na consciência do público, tendo a capacidade de prestar serviços nunca vistos e corrigindo alguns problemas existentes.
Para aproveitar esta oportunidade, vários grupos diferentes estão a criar projetos com o intuito de definir regras, iniciar a experimentação e construir a arquitetura básica do sistema; um desses projetos é o BUBBLES.
O BUBBLES pretende criar espaços seguros ao redor dos UASs, semelhantes a bolhas, permitindo aos drones voarem com eficiência e segurança, além de evitar problemas com a aviação tripulada já existente. No entanto, como essas bolhas são relativas aos drones, há a necessidade de que a telemetria seja constantemente enviada ao sistema BUBBLES e o sistema precisa ser capaz de definir a separação, tendo em consideração o contexto existente, como problemas de rede.
Este documento descreve o trabalho planeado para atingir o objetivo principal de detectar e minimizar problemas de rede utilizando inteligência artificial, experimentando vários modelos e técnicas, comparando cada um deles e concluindo qual é o mais adequado.
Para além disto, vai ser explicitado neste relatório o processo de aquisição de datasets, a descripção do ambiente usado, problemas e soluções observados durante o desenvolvimento deste trabalho e uma análise dos resultados também vai estar presente.
Finalmente, neste trabalho várias propostas num sentido mais prático vão ser propostas baseado nos resultados discutidos previamente.

Palavras-Chave

UAS;ANN;Drones;BUBBLES;Redes de drones; Redes de Bayes; Séries temporais;

This page is intentionally left blank.

Contents

This page is intentionally left blank.

Acronyms

DDS Data Distribution Service. 14

MLP Multi-layer perceptron. 42

QoS Quality of service. 13

RNN Recurrent Neural Network. 42, 50

UAS Unmaned Aerial System. 1

UAV Unmaned Aerial Vehicle. 3

UPV Universitat Politécnica de Valência. 28

This page is intentionally left blank.

List of Figures

This page is intentionally left blank.

List of Tables

This page is intentionally left blank.

Chapter 1

Introduction

Whenever new technologies appear or become readily available to the general public there are as many advantages as there are disadvantages. In our current time, due to the several factors, a great number of new technologies are emerging or changing to fit the needs of the general public. One of these cases are Unmaned Aerial System (UAS), more commonly known as drones, i.e. an aircraft without a human pilot aboard.
Being initially created for the use of the military, drones have become progressively more accessible and popular, being one of the uses with many foreseeable uses in the medium to long term future, as its the case for transportation. However, leveraging UAS's to their full potential is something that depends on a series of critical factors, also requiring proper risk evaluation.
One of the main problems with bringing UAV services to the masses has to do with avoiding conflicts, such as collisions between UAS, other airborne vehicles, people of even buildings. Regardless of being controlled by an autopilot or manned ground station, navigation and surveillance are critical aspects, providing information for location and tracking of UAVs which is necessary for automatic or manual conflict detection and/or avoidance.
Currently these needs are not guaranteed neither by infrastructure, nor services, rules or most necessary tools. Nevertheless several governments are trying to change that so that they can use these UAS's in a safe way. To achieve this, several organizations including Single European Sky ATM Research (SESAR), Air Traffic Control(ATC), European Organisation for Civil Aviation Equipment (EUROCAE), Civil Aviation Organization (ICAO) and European Union Aviation Safety Agency (EASA) approached the problem by adapting manned aviation infrastructure to support unmanned aviation.

1.1 Context

As previously mentioned one of the organizations that is trying to promote RD efforts towards building a future UAV service ecosystem is SESAR[8]. This organization, created in 2004 has as its main goal the creation of a central technological pillar: the Single European Sky(SES). This SES is an attempt created by the European Commission to update European Air Traffic Management(ATM).
To manage SESAR a public-private partnership known as the SESAR Joint Undertaking(SESAR JU) was established. This conglomerate of organizations was set in 2007 and includes more than 19 members, consolidating 100+ organizations each one with its respective experts.
In 2015 the EU entrusted to SESAR JU the definition of a blueprint on how to safely and

efficiently use UAV's. From there the concept of U-Space originated, i.e. an "ecosystem" made to support drone functionalities as well as improve all missions on the airspace and in all kinds of environment. Currently, these U-Spaces are in a exploratory phase and to achieve an understanding of them several different projects were created; being one of these proposals the Building Basic Blocks for a U-Space Separation Management Service (BUBBLES) project.

1.1.1 BUBBLES

BUBBLES is one of the projects approved by SESAR JU in the description of the call of proposals H2020-SESAR-2019-2.
Its main challenge is the identification of the needs for service users as well as the discovery of safety levels in terms of collision using algorithms developed ad-hoc for UAS. Furthermore, there is a necessity to define the architecture of the system as well as the formulation of the Separation Management Service.
The intent of the Separation Management is to create safety areas around the UAS, envisioned as bubbles, for operations in Very Low Level(VLL) (below 150m), that need to be large enough to insure the safety of the drones, while being small enough to maximize the number of drones in use, considering that there might be communication problem between them.
Additionally, these VLL spaces are not exclusive for UAS, allowing conflicts between manned and unmanned aerial vehicles, that have to be considered in the development of the service.
In short, the spaces defined by BUBBLES will need to communicate between each other, guarantee that the separation between each space is appropriate to all conditions and, in the case of failure or changes in conditions, resolve the issues. All this while not affecting the current manned aircraft services.

1.2 Motivation

UAS are a recent concept and are still maturing, both in terms of social awareness to their capabilities as well as a technology. BUBBLES and this work by proxy grant several advantages to this emerging technology as will be presented in the two following points.

1.2.1 UAS's market growth

As it was previously stated, the demand for drone-based services is expected to grow in the next years, with an equal impact in terms of revenue. In a report from 2016, SESAR JU analysed the market of UAS in the EU and foresaw that by 2050 the demand for drone services will approximately be 400.000 drones. Of these services it is also foreseen that 90% of the flight time of operations will be on professional VLL missions.
This increase of drones will create a need for more new jobs, stimulating the economic growth, however to accompany this development there will be a need to study and prepare the foundations.

1.2.2 Social acceptance

There have been a large number of occasions where the illegal or unsupervised use of UAS's have caused danger for manned aviation. This is due to the fact that due to the increase in production of drones there will be more private owners. Just in 2018 the UK Airprox Board reported a steady increment in separation minima infringements due to UAS, ranging from 6 in 2014 to 125 in 2018.
Due to this it is important to study and prepare better systems and environments that detect or automatically avoid these situation, or at least register the infringement for public safety, which will allow for greater acceptance from the general public in the use of this technology, possibly accelerating the point provided in the previous section.

1.3 Contributions

The work done in the scope of this thesis is aimed to the creation of AI agents that will contribute to the BUBBLES' development by allowing the control and monitoring of the Communications performance.
For the decision of the correct separation minima there is a need to accurately and consistently evaluate the state of the communication. To accomplish this several different models with different advantages and disadvantages are going to be evaluated in a 'laboratory like' environment.
Furthermore, since this work is also in charge of the maintenance of the system used by several of the BUBBLES projects groups, such as the fault injector and the UPV team, there is a need to guarantee that several of the components are changed to allow those teams to work correctly. The impact of this work is also more objectively noticeable in the deliverables where

1.4 Objectives

To add to the previous section and to be more concrete the main objectives will be introduced in this section.
The intent of this work is to create models that allow for the detection of failures in the communication network. While the main objective of the work is to prove experimentally that those agents can be constructed in practice, due to the inclusion of this work in the BUBBLES project it would be interesting to be able to implement these models in practice, however due to the limitation of the scope of the project, being more of an introductory study in the field this aspect might not be accomplished.

1.5 Document structure

This document is organized in five sections. This introductory section has the intent of contextualizing the work, both in scope as well as in the relation with other projects, present the main motivation for this work, especially in terms of the capabilities of Unmaned Aerial Vehicle (UAV) and display the structure of the entire document.

Chapter 2 presents the State-of-the-Art and related research done to identify the techniques/models related to the scope of this project; furthermore some technologies that are vital to the validation necessary for the work. More specifically the areas approached will be related to the management of networks and discovery of problems within those.

Chapter 3 presents the project goals with more detail. It also presents the methodology chosen, the tasks and their status as well as a threshold of success considered for this project.

Chapter 4 introduces some of the work that was accomplished in the first semester, with a more preparatory nature, as well as the impact that it will have in the future of the project.

Chapter chapter 5 presents both the dataset gathering process as well as an analysis of that dataset.

Chapter chapter 6 contains the values acquired in this work, offers a brief analysis in each and discusses both conclusions extrapolated from them as well as possible practical uses.

Finally, chapter 7 presents an evaluation of the work, possible future works and some result discussion.

This page is intentionally left blank.

Chapter 2

SoA and Technologies

2.1 State Of Art

The study of the UASs is a more recent subject, due to the original difficulty of finding an environment where they could be studied efficiently and safely, as well as the main military focus on which they first appeared. However, drone networks are basically a series of interconnected nodes that transmit and exchange information, like directions, video and audio, that move along the physical space. If we subtract the movement that the UASs naturally do, we can see that the definition is very similar to a computer network or a telecommunication network, and these subjects have been thoroughly studied, with a variety of different techniques. Due to this fact, this report will present techniques that have been already studied with the intent of utilizing those in an UAS network. From the research done three possible techniques will be beheld: Time Series Analysis, Bayesian Networks and Artificial Neural Networks.

2.1.1 Time Series Analysis and Forecasting

A time series is a series of data points ordered by time, being equally spaced. Time series analysis is the statistical study of these time series.

One of the most common techniques is ARIMA which uses several smaller techniques and combines them into a final model.The AR in ARIMA signifies the auto-regressive model, that uses the previous values of a time series to extrapolate the following. This is given by the formula:

$X_t = c + \sum_{i=1}^{p} \phi_i X_{t-i} + \epsilon_t$

Where X_t is the value we are trying to predict, c is a constant, X_{t-i} is the previous values, ϕ_i is a variable parameter and ϵ_t is white noise. The idea behind this formulation lies in the fact that it is expected that previous elements influence the next element and, depending on the concrete problem, there might exist an upwards or downwards tendency. Furthermore, in a real scenario there is a stochastic part that can not be calculated; thus there is a need to add some randomness to the predicted value.

The MA part of the acronym represents the moving average model, that tries to minimize the current error introduced by the white noise parcel by using the previous errors. The formula is given by:

$X_t = \mu + \epsilon_t + \sum_{i=1}^{q} \theta_i \epsilon_{t-i}$

Where X_t is the value we want to predict, μ is the mean of the series, the θ_i represents the parameters and ϵ_t is the white noise. Similarly to the previous model the MA also assumes that previous values determine the next, however the difference is in how those previous data points are considered. In the moving average the mean of all values is used and the elements that are considered are the errors between the predicted and actual results. This allows for a better global notion of the values, however it minimizes the impact of sudden shifts in expected results.

To finish the construction of the ARIMA model, other than joining both of the previous models, the only thing that needs to be changed is the fact that instead of using the raw data points, it is used the difference between the consecutive data points. Thus the ARIMA formula is a cascade of 2 models:
$Y_t = (1 - L)^d X_t$
$(1 - \sum_{i=1}^{p} \phi_i L^i) Y_t = (1 + \sum_{i=1}^{q} \theta_i L^i) \epsilon_t$

Where d is the degree of differencing, L is the lag operator ($L^i * X_t = X_{t-i}$) and the remaining variables are the same as in the MA and AR previously explained. The ARIMA model is useful since a great number of time series don't have immediately stationarity to be able to use AR and MA, as well as utilising the advantages of the general notion of the series provided by the MA model and the more recent tendencies provided by the AR model.
The FARIMA model is an adaptation of the ARIMA model where the d parameter can take fractional values, which makes the decay follow a power-law distribution.

2.1.2 Bayesian Network

A Bayesian network is a graphical representation that utilises the Bayes rule and Markov chains to predict, given some pre-established events, the probability of a certain event to occur. The structure itself is a directed acyclic graph (DAG) that connects the known variables (effects) to the possible causes and predicts based on the values of the effects the chance that the cause is also happening.

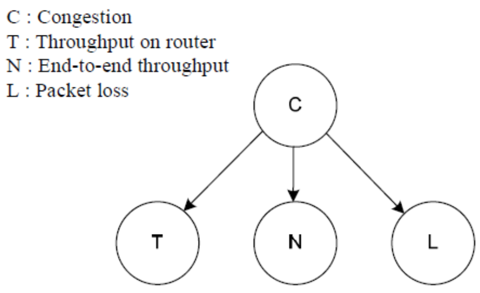


Figure 2.1: Example of a directed graph in context

The Bayes rule states that for two given events, A and B, the probability of A happening, knowing B happened, is dependent of the probability of B knowing A, the probability of A and of B. In other words, this theorem allows the connection between cause and effects, i.e. in the previous example if B is an observable event and A is a cause, knowing that B is happening allows the calculus of the probability that A is happening.
One of the advantages for the use of this technique is the capability to use previously

known information and combine with statistical data, however this is also its greatest disadvantage, since without this data it is impractical to construct such a model.

2.1.3 Artificial Neural Networks

An artificial neural network is a collection of connected nodes, that simulate the functioning of the brain, where the nodes "feed" each other a signal, which is a real number, that is processed by the node. Eventually this signal will reach the output layer where the network will decide on an action. This technique is very powerful, since the number of nodes can be increased to allow for more computational power, however they are less intuitive for a human to understand their decisions and the training time might be quite high. The presented study considers 2 different neural networks: the multilayer perceptron and the recurrent neural network.
To explain in a more complete way we can look at ANNs as directed graphs, where each node has a underlying function and parameters, known as weights, that multiply the inputs to give each the correct importance. As the values received in the input level are fowarded in the network these functions alter them, in a way that the last layer, the output layer, can accurately decide which state the system is in or what action to take. However, these functions hardly are correct in a pre-trained neural network, thus there is a need to change them to fit the expected outputs. This is achieved by the use of learning techniques such as backpropagation.
To better understand backpropagation let's first properly formalize the last paragraph. Let us assume that x is the input vector, y is output, g is the function represented by the entirety of the ANN, L is the number of layers,W^i is the weights matrix for layer i and f^i is the activation function for layer i. This means that we can write $y = g(x) = f^L(W^L * f^{L-1}(W^{L-1} * f^{L-2}(....f^1(W^1x))))$. Now, assuming that r is the correct output we can say that the error or cost (C) is given by C(r,y), which in turn can be represented by C(r,g(x)). It is evident that the best value for C is 0, where r and g(x) has the same value. To arrive at this value we can consider the gradient of C, ∇C that will point in the direction of 0, which in turn can be calculated using the chain rule, i.e. $\nabla C = \frac{\partial C}{\partial a^l} \circ \frac{\partial a^l}{\partial z^l} * \frac{\partial z^l}{\partial a^{l-1}} \circ \frac{\partial a^{l-1}}{\partial z^{l-1}} * .. \frac{\partial z^1}{\partial x}$ where z^i is the input for a layer and a^i is the output of a layer i. This gradient can be rewritten as $\nabla_x C = (W^1)^T * (f^1)'.... \circ (W^{L-1})^T * (f^{L-1})' \circ (W^L)^T * (f^L)' \circ \nabla_{a^L} C$ or for a specific layer $\delta = (f^L)' \circ (W^{L+1})^T .... \circ (W^{L-1})^T * (f^{L-1})' \circ (W^L)^T * (f^L)' \circ \nabla_{a^L} C$. By changing the order of actions we can notice that it can be calculated recursively by utilising the formulation $\delta^{L-1} = (f^{L-1})' \circ (W^l)^T * \delta^l$, so the gradient for each layer can be calculated with a single pass if don recursively from end to beginning. That is backpropagation. The wrights are then updated per node depending on the learning rate, where a big value causes great fluctuations in results and small values diminishes the speed of convergence to the optimal result.

A Multilayer Perceptron is a shallow feedforward neural network, considered one of the simplest. It is composed by at least 3 layers: the input layer where each node receives a value, one or more hidden layers and an output layer.

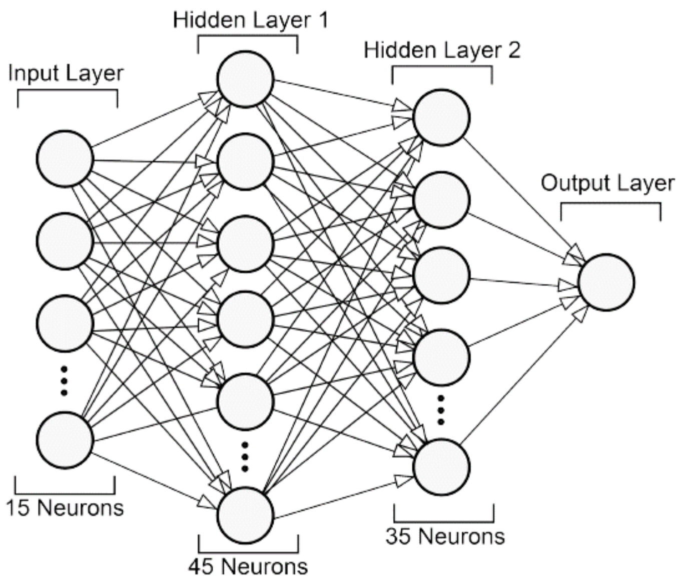


Figure 2.2: Example of the MLP network used in the study by Oliveira et all[11]

A non-linear activation function is used in each node for both the hidden and the output layers. MLP utilizes backpropagation, a supervised learning approach, for training. In the analysed literature another version of the backpropagation was used, called the resilient backpropagation, that dynamically alters the learning rate parameter to avoid either high fluctuation or slow improvement in the training process.

There is also another option for the construction of the MLP in the choice of the chosen activation function, that will impact both the convergence rate as well as the performance of the network. In the studied approach the activation function was the sigmoid, one of the most used functions, however it will be interesting to explore other options. A RNN is a network that is based on the MLP network that utilises memory by feeding the outputs from either the hidden layer (Elman network) or the output layer (Jordan) backwards to the hidden layer serving as part of its input. In this way cycles are created which serves as the previously mentioned memory.

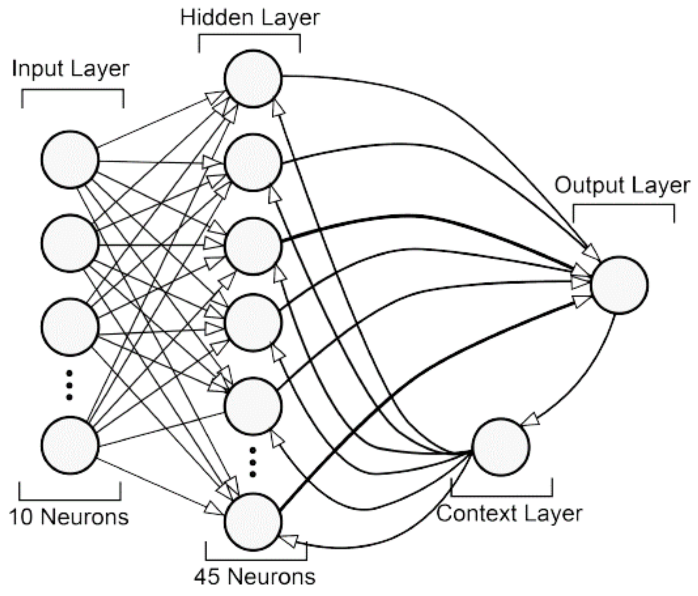


Figure 2.3: Example of the JNN network used in the study by Oliveira et all[11]

2.1.4 A comparative study between the proposed approaches

Three major models were proposed: the time series analysis, the Bayesian network and the ANN. In this section the studies will be more thoroughly discussed.

In the study by Feng et all[9], 4 different datasets were analysed and compared using the different models. The first three datasets are taken from a real scenario, *t040318* and *t030801* being from the WiFi testbed in the Network Research Laboratory of Tianjin University and final.anon was from the Mobile Computing Group at Stanford University. The final dataset was acquired from a synthetic trace using Chaotic Map. The results are presented in the following table:

	ARIMA	FARIMA	ANN
t040318	0.31	0.27	0.28
t030801	0.83	0.80	0.81
final.anon	0.80	0.72	0.70
synthetic	0.57	0.55	0.55

Table 2.1: Adapted results for the comparative study of Feng et all in NRSME

It is important to denote that the used metric is the NRMSE, which is a comparison made between the tested predictor and a trivial predictor, being 0 the perfect prediction and 1 the same capabilities between the two. As we can see, all the results are smaller than 1. Furthermore, all the techniques have some similar values, except ARIMA; nonetheless, since the environment of the UAS network is more unstable and can have faster changes the ARIMA modelling will also be tested for this case.
This sort of modelling is going to mostly be used as the baseline for comparison of the following approaches, since this model doesn't learn information, it just utilizes the previous values for the calculation of the current one, however the weights and values are given parameters that might be optimized.

For the specific case of the network analysis and prediction with Bayesian networks, Harahap et al[10] utilises a model that, from the throughput on router, end-to-end throughput and the packet-loss, extrapolates the chance for the congestion to occur. After training the model, they tested it on a simulated scenario represented in figure 2.4. and then applied the model which gives results presented in figure 2.5.

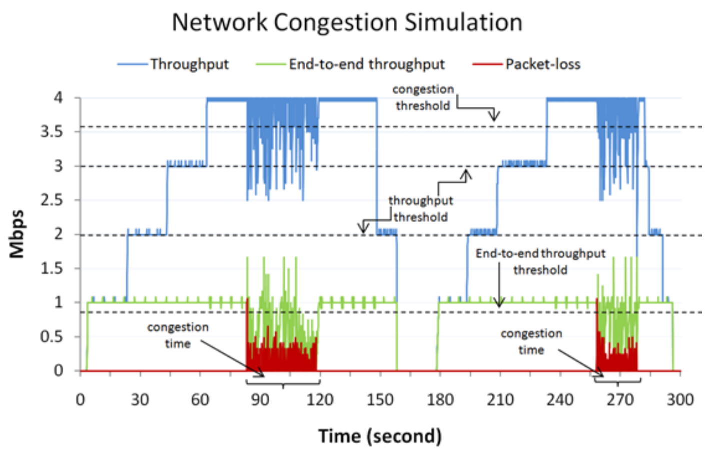


Figure 2.4: Network test for the Bayesian modelling

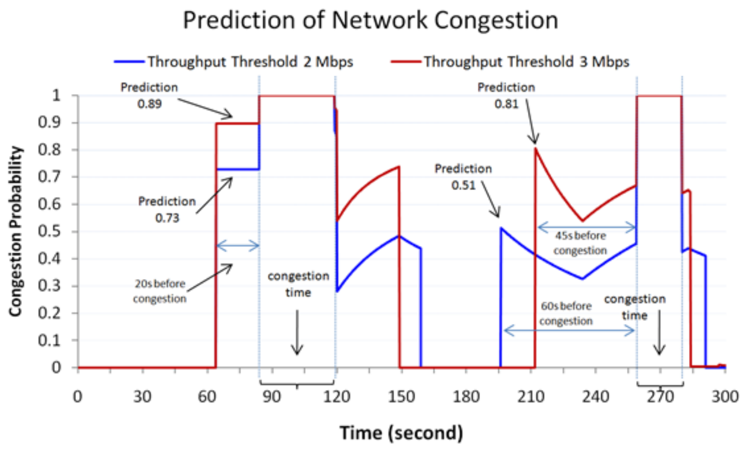


Figure 2.5: Real time predictions of congestion

There is an argument that the result isn't statistically significant, due to the low number of runs, however the values and the ease of interpretability makes it appealing enough to warrant further study. To further elaborate, if a human can more easily understand the problem they can also have a better understanding of the specific problem the network is having, which is even more interesting in the case of a dynamic network such as an UAS network. Furthermore, the time window where the prediction occurs will allow the controller more reaction time to take precautions.

In the work of Oliveira et all[11], 3 different types of neural networks with differing degrees of complexity were implemented to study its application on computer networks by evaluating: time needed to train, efficiency in result and scalability of the data set. Those three networks being: MLP, with and without resilient backpropagation; a JNN network and a stacked autoencoder. The results of the study will be shown in the following tables.

	MLP-BP	MLP-RP	RNN	SAE
A-1d	268	79	67	8,874
A-1h	2,373	1,482	1,473	585,761
A-5m	86,296	56,078	33,981	6,724,641
B-1d	558	326	108	17,280
B-5m	117,652	78,078	45,968	8,691,876

Table 2.2: Time in milliseconds that will take for the network to converge

	MLP-BP	MLP-RP	RNN	SAE
A-1d	0.19985	0.20227	0.19724	0.366
A-1h	0.05524	0.04145	0.04197	0.09399
A-5m	0.01939	0.01657	0.01649	0.02226
B-1d	0.12668	0.14606	0.11604	0.21552
B-5m	0.01306	0.01008	0.00994	0.01949

Table 2.3: Normalized mean squared error for the prediction performance

As we can see, in table 2.2 the time taken by the stacked autoencoder is much greater than for the other two, due to the fact of it being a deep network. Furthermore, in terms of performance it is worse than the other networks and the other networks have some considerably good results, as shown in table 2.3. It is also important to denote that the MLP network with resilient backpropagation is strictly better then with classic backpropagation. Thus, the stacked autoencoder and the MLP approach with backpropagation would not be considered for the future work, however, in the further studies it was discovered that most libraries do not use resilient backpropagation and due to the more practical nature of this work, it was decided that it would be more valuable generalization rather than purely results. Due to that the backpropagation alghorithm will be used. In terms of the rest of the results, the RNN is superior to the MLP in time needed to train and, in most cases, in the normalized root mean squared error. However they are similar enough where there might be an advantage in the context of UAS's.

The time series model has the advantage of its simplicity, however it is not expected that it becomes the best technique, specially due to what was mentioned in its section of not learning and just being a mathematical formulation. The greatest advantage of the Bayesian network is the explainability of the results, still its biggest problem is the dataset, that needs more varied information than the other two and the need for the assumptions being correct. Finally, the ANN are the ones that are expected to have the greatest performance, nonetheless they have low explainability and they are the model with the most complex and costly architecture.

Due to the fact each technique has considerable advantages, in further studies they will be evaluated for their performance.

2.2 Architectural components of the system

As will be explained in further section, this thesis has been based on the architecture planned and constructed in the scope of the BUBBLES project.
To have a better notion of the system, this section exposes several of the components that

make it, allowing for a more critical analysis of the project.

2.2.1 Mavlink

Mavlink is a communication protocol specifically created for Micro Air Vehicles(MAV) [6]. It supports TCP, UDP as well as serial telemetry low bandwidth channels operating under the 1Ghz.
Its architecture has several different systems, forming a network, where each has an id. For each message sent all different systems will access it, due to the fact that it will be broadcast; then each system will verify if they are the intended recipient, proceeding to broadcast the message again to all channels that haven't seen the message.
In terms of Quality of service (QoS), it depends on how it is setup. If the message was sent following a predefined Command Protocol, i.e. a set of rules for how the system will treat the message. If the system is using a Protocol that does not expect an acknowledgment, MAVLink offers at-most-once delivery. However, if the system expects a matching acknowledgment, it offers the at-least-once type of delivery because if no ACK is received, it will resend the message.

2.2.2 Kafka

Apache Kafka is an open-source message-broker middleware with additional streaming capabilities[1]. It uses a binary protocol over TCP/IP that defines all APIs as response-request message pairs.

The principles behind it are the same as other message-oriented middleware: Producers push messages to a queue while Subscribers access those messages. These messages are pushed to a topic, which are similar to a folder in a filesystem, while the messages will be the files. In Kafka all topics are multi-producer and multi-subscriber, being able to accommodate 0,1 or many consumers.

It is also important to denote that each of these topics are partitioned , as in, a topic is spread on several Kafka brokers, which allows for greater scalability. Furthermore Kafka also guarantees that the messages are sent to the same partition if their key is the same and that consumers will acquire the messages ordered by the time where they were written.

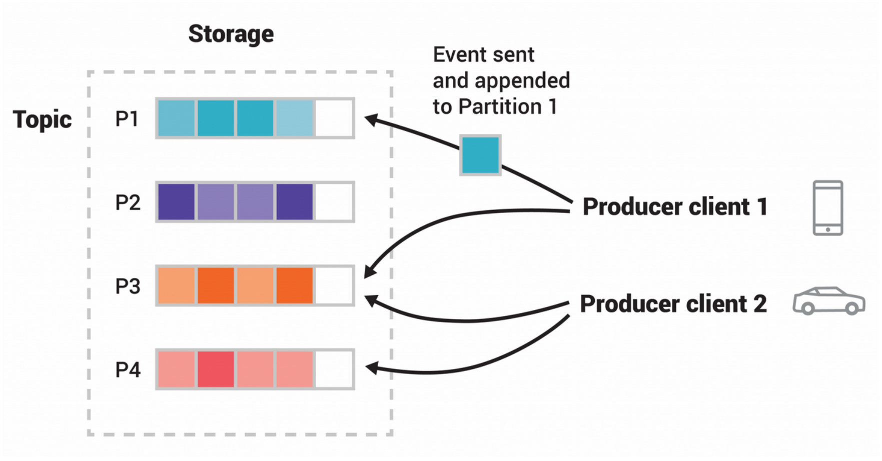


Figure 2.6: Example of two producers sending messages to topic from the Kafka documentation[1]

Kafka also has three different levels of QoS: at-most-once, at-least-once and exactly-once. At-most-once means that the message sent might be lost, but will never be redelivered, which allows less congestion in the network by sacrificing reliability; at-least-once guarantees that all messages will be delivered, however there is a chance that they will be redelivered, its benefits are symmetrical to previous level of QoS. Finally, exactly-once guarantees that the message is delivered only once, with the additional cost on the network.

2.2.3 DDS

Just like the previous protocol, the Data Distribution Service (DDS)[2] is a publish-subscribe model, utilised for real-time systems and machine-to-machine communication standard.It is also brokerless, utilising instead multicast to provide reliable and efficient communication over both TCP and UDP.
One of its main advantages is the way that the communication is setup almost automatically, reducing programming time; this includes not needing to know where the other machine is, determining who should receive the message or what happens if the message is not received.
In terms of QoS, DDS loads the intended QoS message into the packet, in case of a publisher, while the subscriber expects that the package it receives have a corresponding level of QoS.
From the QoS files, two attributes were configured: RELIABILITY_QOS and the DURABILITY_QOS.
The RELIABILITY_QOS was configured on both the DataReaders and DataWriters to offers at least once delivery (RELIABLE). For the communications to be reliable, the HISTORY_QOS needed to be configured on the DataWriter.
For durability, the selected type was KEEP_ALL_HISTORY_QOS. This way, all messages sent by the DataWriter are saved, and in case of error, due to the RELIABILITY_QOS being configured, it will resend saved messages that the subscriber could not access.

2.2.4 Px4

PX4 is an open-sourced autopilot software system[7]. It is hosted under the governance of the Dronecode Project. Since it was created in 2009 it has had time to improve and refine its systems.
PX4 can utilise both Software-in-the-loop(SITL) or Hardware-in-the-loop(HITL). In the testbed environment it was only tested using SITL, i.e. every thing is simulated utilising software, from the sensor data to the environment. The PX4 autopilot receives the data from the simulator utilising Mavlink; these are then processed and the commands will be sent back to the virtual drone.

2.2.5 Gazebo

Gazebo is a collection of open source libraries that is designed to simplify development of high-performance applications[3].
This system is widely used in various different robotics applications due to its principles of generality, stability, constant updates, ease of use, modularity, extensibility, flexibility, maintainability and portability.

The facet of Gazebo that was used was Gazebo simulator, a robotics and physics simulator where the drones were already defined in one of the public libraries, which the BUBBLES team used to perform most of the tests.

2.3 Architecture for the communications platform

The BUBBLES communications architecture proof-of-concept was also integrated within the BUBBLES UC-UPV validation scenario (instantiated in the VMs created on the CANVAS testbed). From an integrated service perspective, the BUBBLES communications architecture relies on publisher-subscriber message queue mechanisms for supporting the communication flows between U-Space service components. Also, UAV communications are provided by means of a communications agent embedded in its embedded software stack, which will provide a continuous telemetry feed to the tracker service.
The specific implementation for the BUBBLES communications architecture (depicted in Figure 2.7), encompasses the key components of the communications architecture, which is also used for service integration, with a particular focus on the clear separation between edge and core communications. Edge brokers provide the first communications endpoint, ideally within a 1-hop radius from the UAV, while core brokers provide a reliable backbone, designed to provide scalability and reliability properties by means of partitioning mechanisms and distributed geographic points-of-presence.

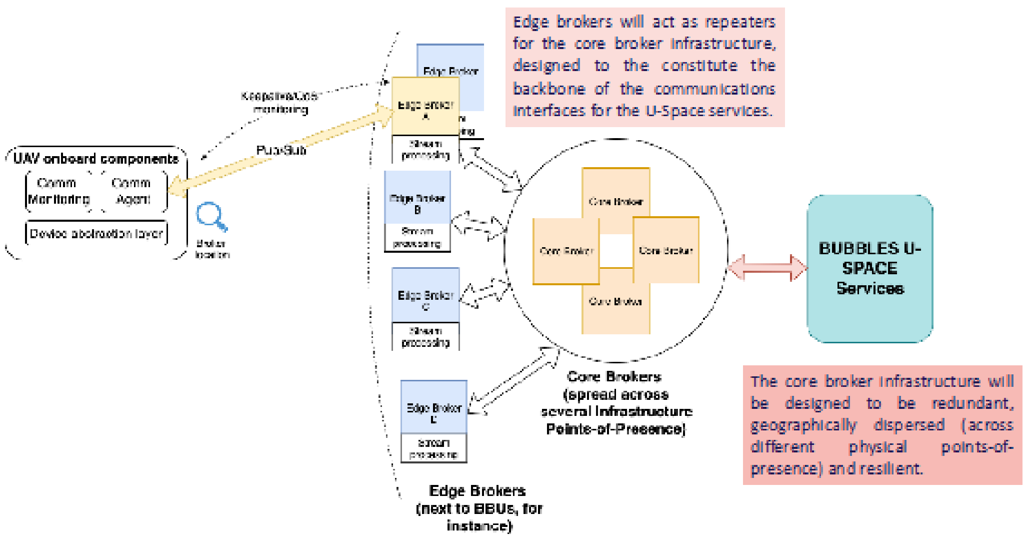

Edge brokers will act as repeaters for the core broker infrastructure, designed to the constitute the backbone of the communications interfaces for the U-Space services.
Keepalive/Comm monitoring
Pub/Sub
UAV onboard components
Comm Monitoring
Comm Agent
Device abstraction layer
Broker location
Edge Broker
Edge Broker A
Stream processing
Edge Broker B
Stream processing
Edge Broker C
Stream processing
Edge Broker L
Stream processing
Core Broker
Core Broker
Core Broker
Core Broker
BUBBLES U-SPACE Services
Core Brokers (spread across several Infrastructure Points-of-Presence)
Edge Brokers (next to BBUs, for instance)
The core broker infrastructure will be designed to be redundant, geographically dispersed (across different physical points-of-presence) and resilient.

Figure 2.7: BUBBLES Communications Platform Concept

The Communications Performance Monitoring mechanisms (CPM) are implemented by resorting to instrumentation of the agent, edge broker and core broker components. In this way, it becomes possible to measure the impact of communications. A network emulator (which is part of the validation environment, providing communications shaping/impairment capabilities) is managed by the fault injection tool, providing the means to generate network fault/failure situations for controlled experimental validation, as shown in Figure 2.8.

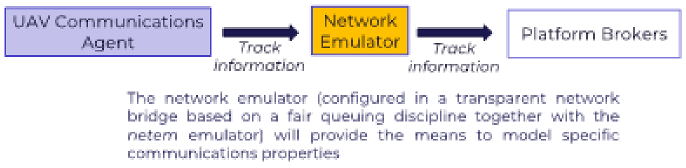

UAV Communications Agent
Track information
Network Emulator
Track information
Platform Brokers
The network emulator (configured in a transparent network bridge based on a fair queuing discipline together with the netem emulator) will provide the means to model specific communications properties

Figure 2.8: Network Emulator usage for Communications Platform tests

A publisher/subscriber model is used to implement communications between the drone and edge brokers, having communication between the edge and core to forward the messages. To achieve this, the DDS protocol was used for drone-edge communication and Kafka for edge-broker communication. In a real scenario there would be a need for some external equipment to be attached to the drone, so that it could have access to the flight information, needing some sort of serial communication, but in the specific case of the UC-UPV platform, information is extracted from PX4 via Mavlink.

2.4 Traffic shaper

Other than the communication agent, a network controller was added to the communication by utilising a bridge. This controller sets several network parameters in the communication such as the latency, packet loss, jitter, etc.
This is achieved by the use of a Token Bucket based queue, more specifically Hierarchical Token Bucket(HTB) queue, where each packet is put in a queue and sarcastically gets attributed a token. These tokens are refreshed by a set rate and represent a certain amount of bandwith, that will be used to pass the messages.
Since the technique used is the HTB a class hierarchy can be setup, wheretree-like topology where subclasses (who have their own token buckets) can borrow tokens from the upper classes, if they are available.

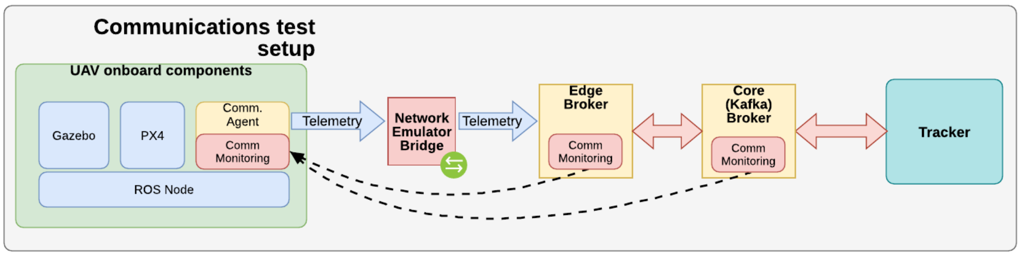


Figure 2.9: Environment with traffic shaper

Another advantage for the network emulator is the fact that it remains invisible from any of the machines, not directly affecting them. It is also easily exportable allowing a seamless deployment in several scenarios.

2.5 Testbed scenario

Figure 2.9 depicts the network topology for the UC-UPV validation environment, hosted within a VMware ESXi type-1 hypervisor infrastructure. Represented components include virtual machines (green) and virtual network switches with respective port groups (yellow for the test network, blue to the management network).

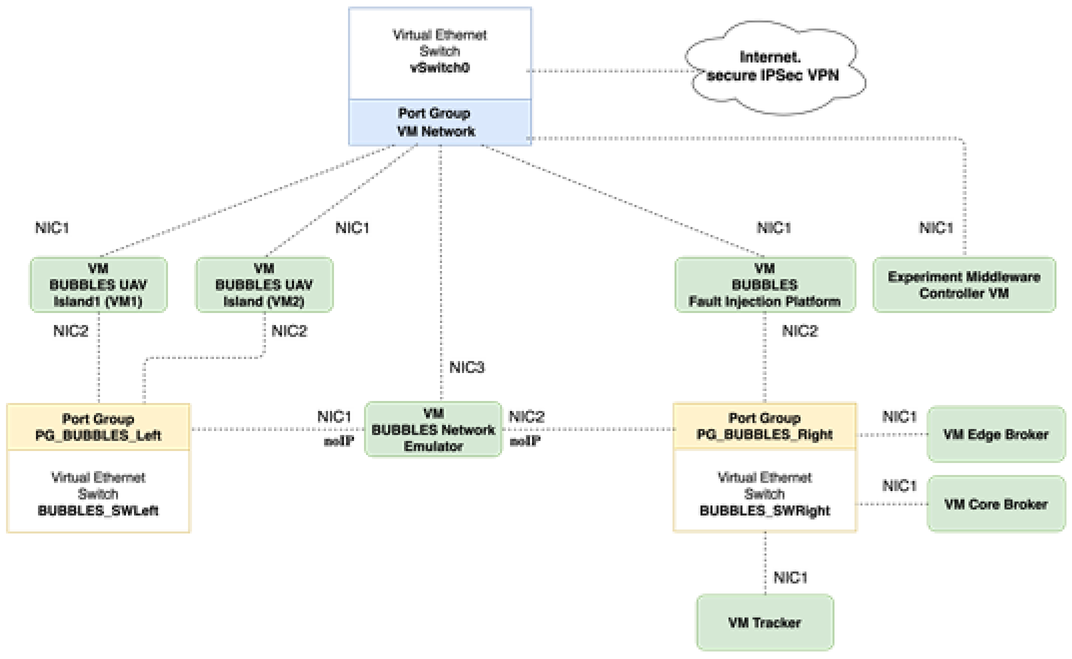

Virtual Ethernet Switch vSwitch0
Port Group VM Network
Internet. secure IPSec VPN
NIC1
NIC1
NIC1
NIC1
VM BUBBLES UAV Island1 (VM1)
VM BUBBLES UAV Island (VM2)
VM BUBBLES Fault Injection Platform
Experiment Middleware Controller VM
NIC2
NIC2
NIC3
NIC2
Port Group PG_BUBBLES_Left
Virtual Ethernet Switch BUBBLES_SWLeft
NIC1 noIP
VM BUBBLES Network Emulator
NIC2 noIP
Port Group PG_BUBBLES_Right
Virtual Ethernet Switch BUBBLES_SWRight
NIC1
VM Edge Broker
NIC1
VM Core Broker
NIC1
VM Tracker

Figure 2.10: VM and network topology deployment on the virtualized environment

VM and network topology deployment on the virtualized environment There are two networks in the testbed setup: the internal Departmental network (DEI stands for UC's "Department of Informatics Engineering") and the private BUBBLES network. Each VM may have more than one virtual Network Interface Card (NIC) connected to different networks. The connectivity is arranged as such:
-Connections on the Private BUBBLES Network (yellow – SWLeft and SWRight): BUBBLES_VM1, BUBBLES_VM2, BUBBLES_Fault_Injector (via NIC2 in the three cases, private IP addresses to be defined) and BUBBLES_Emunet (via NIC1 and NIC2, which do not require an IP).

- Connections on the Public DEI Network (blue - VM Network): BUBBLES_VM1, BUBBLES_VM2, BUBBLES_Fault_Injector (via NIC1 in all cases), BUBBLES_Emunet (NIC3). In this network, IP addresses are already configured.

- Connections to the DEI network already have a configured IP address – the VMs are accessible via SSH. These interfaces must be used for out-of-band configuration of the test setups and VM maintenance/configuration – this is due to the fact to these connections will never be affected by the network emulator VM

The private BUBBLES network is spread across two virtual ethernet switches (BUBBLES_SWLeft and BUBBLES_SWRight). The idea is that emulated drones will be connected on the Left side and the fault injector VM will be connected on the Right side. This terminology is also used on the configurations for the sidekick agent, described in subsection 3.4 – communications between this agent and the fault injector are provided by means of the Experimental Middleware Controller VM. Connectivity between the two sides (Left and Right) will be established by means of the BUBBLES_Emunet VM, which is in charge of introducing communication disturbances and/or faults (using a network emulator). For instance, in the depicted scenario, all communications between BUBBLES_VM1 and BUBBLES_VM2 (via NIC2) and the BUBBLES_Fault_Injector (via NIC2) VMs will only be possible through the path that is controlled by the BUBBLES_Emunet VM emulator. Special care must be taken regarding the following aspects:

1. All the components involved in the PX4/Gazebo/Fault Injection scenario must communicate with each other only within the private BUBBLES network

2. NIC2 on BUBBLES_VM1, BUBBLES_VM2 and BUBBLES_Fault_Injector must have manually configured IPs (to be defined during the configuration of the components referred in item 1). These NICs belong to the private network – therefore, an IP address range must be chosen in order to avoid conflicts with the rest of the DEI networks (the range 192.168.2.0/24 is probably adequate).

3. NIC1 on BUBBLES_VM1, BUBBLES_VM2 and BUBBLES_Fault_Injector must only be used for remote access/maintenance or for test configuration purposes, controlled via the Fault Injector VM (which has a single NIC)

NIC1 and NIC2 on the BUBBLES_Emunet VM do not require a configured IP, since this VM operates as a transparent network bridge (thus, at Layer 2).

2.6 Network emulator implementation

To limit outgoing traffic on a NIC, a Token Bucket based queue discipline is used, namely the Hierarchical Token Bucket (HTB) [4]. In the HTB queuing discipline egress traffic bytes are serviced by tokens, refreshed at an established output rate. Tokens are saved up in a bucket of limited size, allowing for a bandwidth credit, since smaller bursts of traffic can be handled at a higher rate.
On a HTB arrangement, the class hierarchy expands upon the basic token bucket concept, allowing for the creation of a tree-like topology where subclasses (who have their own token buckets) can borrow tokens from the upper classes, if they are available.

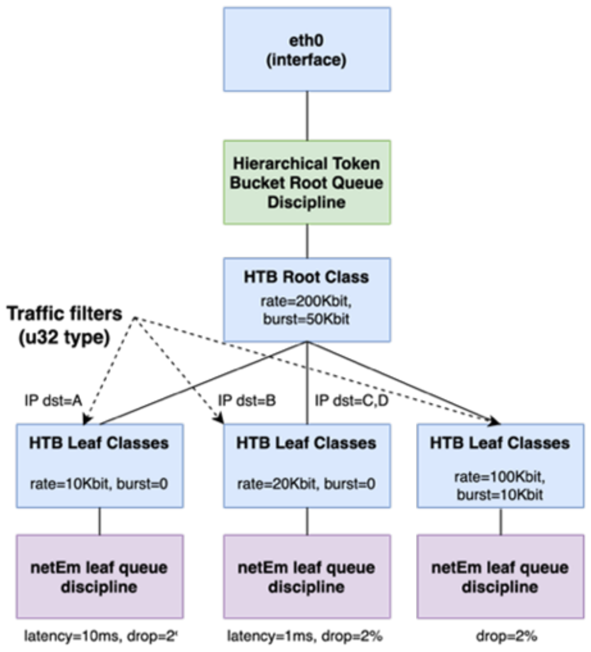

eth0 (interface)
Hierarchical Token Bucket Root Queue Discipline
HTB Root Class rate=200Kbit, burst=50Kbit
Traffic filters (u32 type)
IP dst=A
IP dst=B
IP dst=C,D
HTB Leaf Classes rate=10Kbit, burst=0
HTB Leaf Classes rate=20Kbit, burst=0
HTB Leaf Classes rate=100Kbit, burst=10Kbit
netEm leaf queue discipline
latency=10ms, drop=2'
latency=1ms, drop=2%
drop=2%

Figure 2.11: Network Emulator Traffic Shaper Hierarchy

To limit outgoing traffic on a network interface, a Token Bucket based queue discipline is used, namely the Hierarchical Token Bucket (HTB). In the HTB stochastically fair queuing discipline, egress traffic bytes are serviced by tokens. Such tokens are refreshed at an established output rate, controlled by a clock source. Tokens are saved up in a bucket of limited size, allowing for a bandwidth credit, since smaller bursts of traffic can be handled at a higher rate. With HTB providing rate control, the netEm[5] capability embedded within the Linux traffic control feature set provides emulation mechanisms to implement packet loss, latency and jitter disturbance/impairment.
The deployment of the HTB + netEm tandem forms the basis for the Network Emulator tool, which is be hosted on a Linux Virtual Machine (VM), fine-tuned for a real-time profile, with locked CPU and memory reservations and configured to behave as a transparent layer 2 bridge (thus being completely invisible for any communications parties). The design of the network emulator allows for it to be packaged as an Open Virtualization Format (OVF) appliance to be seamlessly deployed in scenarios involving network communications between 2 or more nodes.

This page is intentionally left blank.

Chapter 3

Methodologies and planning

This chapter objective is the definition of the methodology chosen as well as delineating the tasks and planned progress for the entirety of the project lifetime.

3.1 Methodology

For a project to be successful it is important to have a set of rules and structure, allowing easier segmentation of objectives. These structured processes are the methodologies, tested over a different number of projects and with definite strengths and weaknesses.

In this project the adopted methodology will be the waterfall model, where the more rigid structure allow for easier estimation of deadlines and the initial evaluations help in avoiding wastefulness. Furthermore the biggest weakness of this model, its rigidity, will not be detrimental, since the requirements won't change.

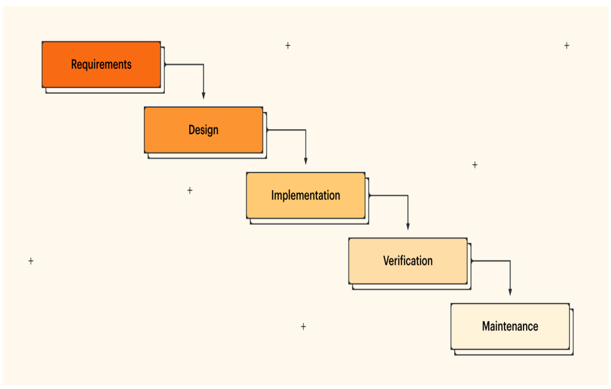


Figure 3.1: Example of the waterfall model, extracted from Lucidchart.com

The only difference from the classical approach of this methodology and the one that will be used is that the test phase and development phase will be merged into one, due to the more stochastic relation of the developed models.

3.2 Tasks

In the following table several different tasks are presented that allow for the natural progression of the work. These tasks aren't expected to change, however there is a chance that more will be added.

Tasks	Progress
Understanding the architecture	Done
Research on communications monitoring techniques	Done
Gather datasets	In Progress
Analyse datasets	In Progress
Adapt the system to run faulty runs	Done
Gather requirements	To Be Done
Construct the models(TSA and ANN) for QoS active	To Be done
Construct the models(TSA and ANN) for Qos active and inactive	To Be done
Construct models aware of the situation (Bayesian and ANN)	To Be done
Construct a model to be placed on the edge broker	To Be Done
Run tests	To Be done
Write thesis	To Be done

Table 3.1: Tasks At the intermediate defense

After the study of the area and the more concrete realisation of the project's goals, the requirements were created and described as tasks. Furthermore, the planned order of difficulty in the creation of the agents, as well as the attempt to organize the work with the rest of the BUBBlES team allowed for a more structured development.

Throughout the timeline of the project the report was changed and adapted to accurately reflect the status of the work. Moreover, each task was tested several times, even after their intended completion, to guarantee the results were irreproachable.

Tasks	Progress
Understanding the architecture	Done
Research on communications monitoring techniques	Done
Gather datasets	Done
Analyse datasets	Done
Adapt the system to run faulty runs	Done
Gather requirements	To Be Done
Construct the models(TSA and ANN) for QoS active	Done
Construct the models(TSA and ANN) for Qos active and inactive	Done
Construct models aware of the situation (Bayesian and ANN)	Not done
Construct a model to be placed on the edge broker	Done
Run tests	Done
Write thesis	Done

Table 3.2: Tasks at the end of the project

The more impactfull change was the impossibility of the use of realistic datasets, due to reasons that will be explained in further sections.

3.2.1 Task details

In this subsection each task will be explained with more detail.

- **Understanding the architecture** - Since the architecture of the BUBBLES communication system was created and needed to be used, in the beginning, the main task was the understanding of the protocols as well as the logic behind it.

- **Research on communications monitoring techniques**- The main objective of this work is the detection of network problems; due to this there was a need to research about this topic.

- **Gather datasets**- This task will be better explained in the following section 4.

- **Analyse datasets**- Some studies apply a more brute force approach to the use of the datasets, i.e. trying to utilise either the known methods or the most complex ones, however it was decided that the analysis of the dataset, utilising some pattern recognition techniques would greatly enrich this work.

- **Adapt the system to run faulty runs** - Another team that is working on the project developed a fault injector, that realistically recreates in the simulation several common problems in drone aviation. Due to this there was a need to assure that the fault injector and the communication agent worked together, especially because for the datasets to contain more interesting data this merging will need to be complete.

- **Gather requirements**- After obtaining the datasets there was the need to evaluate what exactly needs to be accomplished to achieve the following tasks.

- **Construct the models(TSA and ANN) for QoS active** - In a first attempt and as the minimum for the success of the project, the agents that are more directed to the original, less complete datasets for the main state of the communication system, i.e, with the QoS active.

- **Construct the models(TSA and ANN) for Qos active and inactive**- After the success of the previous task an interesting improvement is the capability of the communication system to dynamically change the level of QoS, however to accomplish this the agent has to be able to understand both modes of the system and provide accurate estimations.

- **Construct models aware of the situation (Bayesian and ANN)**- When the gathering of faulty runs datasets is completed it will be interesting the use of these more complete datasets for the prediction of the network state. This is especially true for the Bayes network, since it will allow for a human pilot to have a better notion of the problems.

- **Construct a model to be placed on the edge broker**- As stated in **??** the edge broker has only the job of forwarding messages, which is an inefficient use of resources, especially considering the fact that it is the broker that is closest to the drones. To improve this an agent will be implemented that, with less information, can provide an estimation of problems that will be forwarded to the core broker, as well as a regulator that will allow the activation of measures to counterbalance any problems that were detected.

- **Run tests**- Throughout the development phase several different tests will be done to assure the correct functioning of each agent, also providing inputs for further refinement, in line with the spirit of the waterfall development approach.

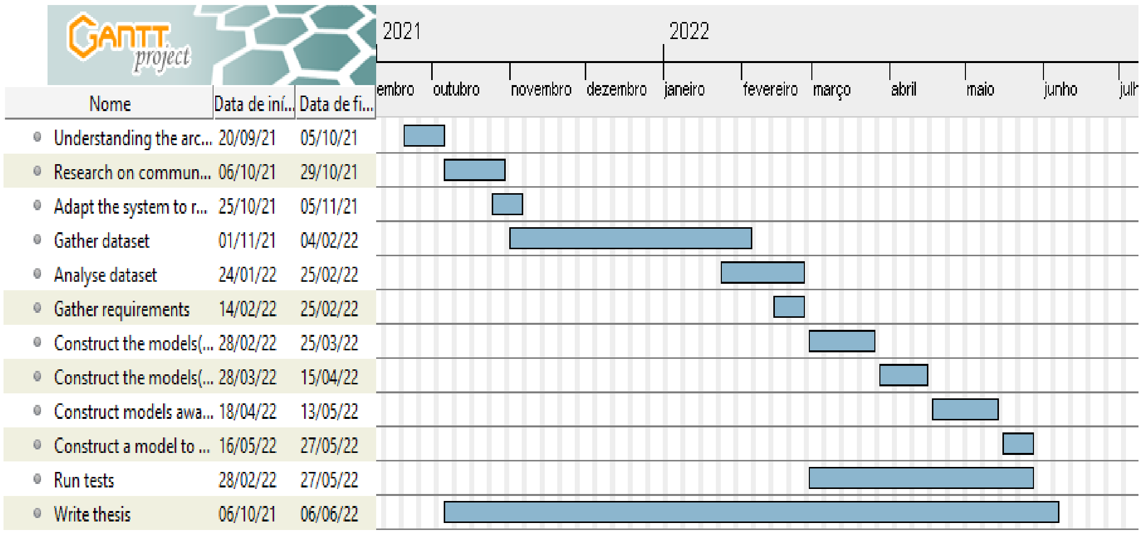
- **Write thesis**- To allow for an easier and more detailed comprehension of the work done, the thesis will be constantly updated while the other tasks are being done.Moreover, this effort is also closely linked with the need to provide inputs to official BUBBLES project deliverables, which will be undertaken in line with the outputs of the undergoing work, also incorporated into the thesis document.

- **Gather dataset - Second iteration** - After studying the datasets in task **Analyse datasets** it was discovered that due to various reasons they were insufficient. Thus there was a need to create new datasets taking into account those conclusions.

- **Resolve problems with agents** - In the review of the models and their training process some inconsistencies and non-expected values gave rise to a need to understand the problems and correct them, guaranteeing the correct behaviour of the agents.

3.3 Planned Gantt

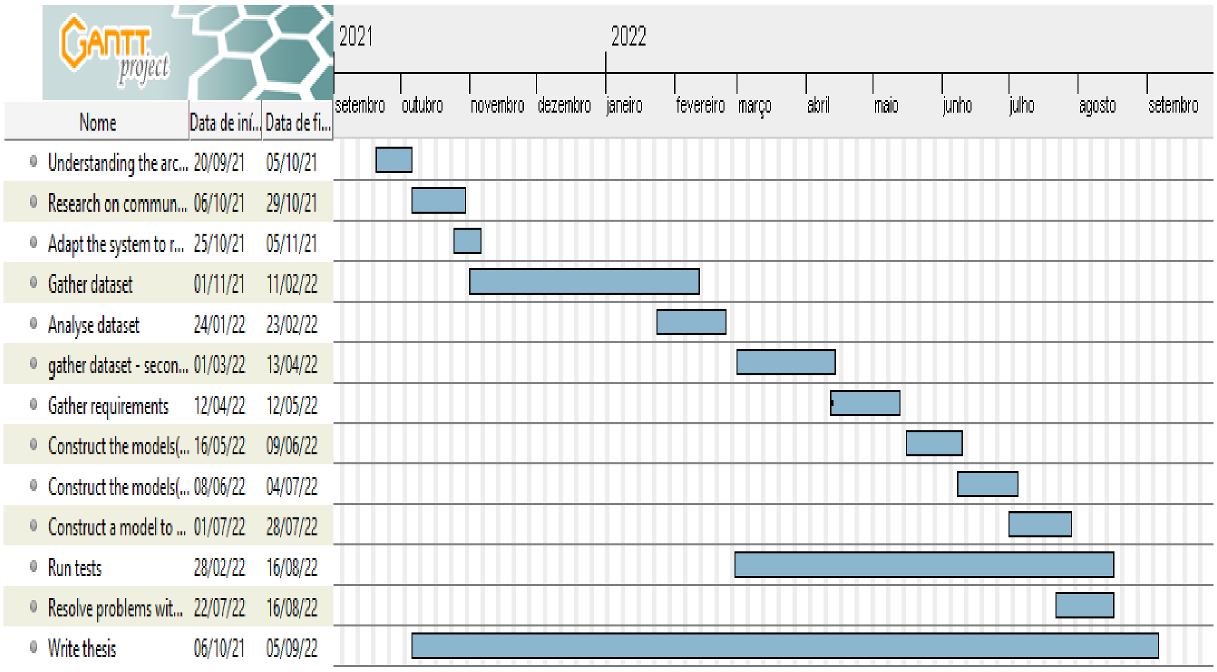
In this section a Gantt diagram of work done and to be done is presented and discussed for the first semester. This Gantt is based on the work that has been done, as well as some estimations.
Furthermore, there is a Gantt representing the actual development of the work, which will be used as a comparison tool to allow for the detection of problems with the work itself.



Nome	Data de iní...	Data de fi...
Understanding the arc...	20/09/21	05/10/21
Research on commun...	06/10/21	29/10/21
Adapt the system to r...	25/10/21	05/11/21
Gather dataset	01/11/21	04/02/22
Analyse dataset	24/01/22	25/02/22
Gather requirements	14/02/22	25/02/22
Construct the models(...	28/02/22	25/03/22
Construct the models(...	28/03/22	15/04/22
Construct models awa...	18/04/22	13/05/22
Construct a model to ...	16/05/22	27/05/22
Run tests	28/02/22	27/05/22
Write thesis	06/10/21	06/06/22

Figure 3.2: Gantt for the project

The most important thing in this Gantt is the size of the dataset gathering task. This happened due to some problems in setting the environment, as well as the natural time that it takes to generate a dataset, which originally was much larger. Other than that, the waterfall model is very explicit when the practical side is examined.



Nome	Data de iní...	Data de fi...
Understanding the arc...	20/09/21	05/10/21
Research on commun...	06/10/21	29/10/21
Adapt the system to r...	25/10/21	05/11/21
Gather dataset	01/11/21	11/02/22
Analyse dataset	24/01/22	23/02/22
gather dataset - secon...	01/03/22	13/04/22
Gather requirements	12/04/22	12/05/22
Construct the models(...	16/05/22	09/06/22
Construct the models(...	08/06/22	04/07/22
Construct a model to ...	01/07/22	28/07/22
Run tests	28/02/22	16/08/22
Resolve problems wit...	22/07/22	16/08/22
Write thesis	06/10/21	05/09/22

Figure 3.3: Gantt for the project evaluated at the end

We can notice several differences from the expected gantt, mostly centered around the second iteration of the gathering of datasets and the removal of crating agents using realistic

datasets.
This was caused due to different reasons: firstly, a large amount of problems only discovered after the analysis of the dataset created a need to redo them. Furthermore, as will be discussed ahead, the units used for the fault injector, the UPV team and the system utilised were not compatible and since the tests with real drones were not done with just the BUBBLES team, they could not be rescheduled.
We can additionally also notice a problem with the model chosen, where the impact from the need of adding a second iteration for the datasets delayed the entirity of the remaining work.

3.4 Threshold of Success

For a project to be successful there is a need to define clear and concise objectives that, if completed guarantee its completion or failure. A Threshold of Success is a technique which defines the minimum requirements or tasks that if not completed mean that the project failed. The ones that will be defined for this project are:
-having all the requirements completed in the allotted time for the work.
- guaranteeing that at least one of the models can detect problems, as in the accuracy and precison have to be over 0.6.

This page is intentionally left blank.

Chapter 4

Preliminary phase

In the first semester the main focus was on preparatory work that allowed the later implementation of the more practical variety that is present in chapter 5 and chapter 6. On the first half the objective was to learn the communication system developed by Pedro Ribeiro. After this initial moment and with the help of Pedro there was some decisions made to improve the system. Following this moment the work was done individually which includes the dataset acquisition and adaptation to an environment of faulty runs.

This section will describe in more detail the work previously mentioned.

4.1 Alteration of packet data

As was stated in chapter 1 the BUBBLES project is undertaken by a consortium as a group, thus this system will be used for the entire team, one of which is the Universitat Politécnica de Valência (UPV), which has the job of constructing a tracker to detect conflicts both in the strategic (pre-flight) tactical (in flight) phases.
In the integration with the tracker developed by the team from the UPV it was discovered that the information sent was insufficient, so there was a need to adapt the information sent by the system from the model identified by table 4.1 to the model identified by table 4.2.
The main difficulty in this part was discovering if PX4 and Mavlink could provide the required data.

Attributes	Description	Attribute type	Data Specificity
altitude	altitude	float	m
latitude	latitude coordinate	float	°
longitude	longitude coordinate	float	°
yaw	yaw angle	float	°
roll	roll angle	float	°
pitch	pitch angle	float	°
battery	Battery remaining	float	%
time	time for telemetry poll	uint64	ms Unix format

Table 4.1: Previous content of the messages

Attributes	Description	Attribute type	Data Specificity
altitude	altitude	float	m
latitude	latitude coordinate	float	°
longitude	longitude coordinate	float	°
yaw	yaw angle	float	°
roll	roll angle	float	°
pitch	pitch angle	float	°
x_vel	Velocity in X	float	m/s
y_vel	Velocity in Y	float	m/s
z_vel	Velocity in Z	float	m/s
horizontal_uncertainty	Position uncertainty	float	m
vertical_uncertainty	Altitude uncertainty	float	m
vel_uncertainty	Velocity uncertainty	float	m/s
battery	Battery remaining	float	%
time	time for telemetry poll	uint64	ms Unix format

Table 4.2: Current content of the messages

4.2 Dataset acquisition

To acquire datasets there were two approaches that were followed: the first, in which there was a research effort to investigate possible datasets and a second where the datasets were created using the testbed explained in section 2.3.

After looking through several different datasets the conclusion achieved was that the faults given were less about communication and more about either mechanical problems of the drone or security attacks to the network. Thus the majority of the data used will be from the second approach, since it allows more control over the created data.

The methodology for acquiring data is as follows: with the scenario activated a certain number of drones will be initialized both in gazebo and in the system; furthermore the core broker will be started, using the timeout capabilities of Linux to guarantee an even amount of time. Core will also log the received time of each packet as well as the rest of the message.

The previous experiment will be run for 1,5 and 10 drones with a rate of 4 Messages per second and the QoS might be active or not. Furthermore the network emulator will simulate abnormal conditions such as Packet Loss and Latency.

It was planned that in a posterior attempt these datasets will be acquired utilising the faulty run setup, in which some more complex faults inherent to UAS's will be beheld in a more accurate simulation.

4.3 Adapting the architecture for faulty run setup

The creation of datasets is vital to the development of the rest of the project, to utilise a faulty run setup, i.e. a setup where network faults are injected as the simulation is run, there was a need to remake the architecture due to the fact that both the drone and the fault injector were using the same UDP port, because the stream would be blocked by one of the services and unable to receive from the two sources simultaneously.

To deal with these problems a TCP server was nested in the VM responsible for the drone simulator that then received messages by TCP from the python fault injector and then sent them to the communication platform using the DDS protocol.

More specifically, the fault injector connects to the virtual UAS, sending it the instructions after some distortions, created by typical telecommunication problems such as GPS malfunctions, or areas with a large amount of disturbances. However, as previously mentioned this capability controls the only direct communication with the drone, not allowing it to connect to the communication system. This was solved by making it so each thread from the fault injector, responsible for each drone, creates a TCP connection to what was previously described as the Drone.
The main challenge of this part was the discovery of the problem, as well as understanding if the solution could affect the rest of the system or not.
However, due to a difference in units, that is not established yet for the scope of UAS's, it was impossible to utilise the fault injector without several changes for the dataset acquisition, moreover the fault injector architecture was not studied in detail so it could not be changed.

Chapter 5

Experimentation

5.1 Data set acquisition

The first step which created some concrete value of the thesis project was the dataset acquisition. As was previously mentioned the datasets were obtained by utilising the testbed scenario mentioned in section 2.3 by utilising several delay and loss parameters. For the first we used values of 10, 20 and 30ms, while for the latter we introduced losses of 10, 20 and 30%.

In the following subsections each level of dataset creation will be presented, justified and any problems during these steps will be discussed, as well as the corrections.

5.1.1 First phase-Dataset acquisition with QoS active

The first group of datasets that were gathered were the ones where the full system is completely operational, i.e. with the QoS active and using data in the core.

However as the datasets were obtained several problems were detected:

- since this was the first time the setup was constructed and with the intention of setting this exact setup for future studies that need it, all the software was updated to work in Ubuntu 20. However some of the libraries used were outdated.

- since the system has a number of different VM's, if the clock isn't coherent between them timestamps will be unreliable.

- due to the use of a multicast-based technique, the DDS protocol is too efficient in discovering alternative routes to reach the address of the edge, which made it so the packets avoided the supposed added loss from the traffic shaper.

- due to the fact that it was a shared environment with other projects the network was affected by the use of other virtual machines.

- another problem that will be further discussed in the section 5.2 was the fact that a lot of the possible parameters for the network faults were not useful, either because there is no noticeable impact or the values were too unrealistic.

To diagnose and solve these problems some time was needed, both for the reruns of the system as well as for their discovery. These problems were individually solved, using several methods:

- for the update problem the approach was an investigation of the current versions of the software and a trial and error approach to fit the scenario.

- for the problem related with the clock of the system an option was activated to guarantee that the VM clocks and the hypervisor host clock are in sync.

- to solve the problems with DDS any network interface that wasn't part of the system was disabled, making it so no alternate routes could be used forcing the packets into the correct path.

- to solve the problem with other projects, the non-necessary machines for the running of the simulation were nonoperational during the dataset acquisition.

- to solve the parameterization problem two approaches were used, one which was a trial and error approach, balancing the distinctiveness of the data with the possibility of those errors occurring; the second was to use both loss and delay to create a more realistic failure that produced actual results.

5.1.2 Second phase-Dataset acquisition with QoS inactive

In this phase the QoS of the DDS was disabled to allow for a scenario where it was considered that it was most efficient to disable the QoS.
This section of the work did not have many problems and the ones that existed overlapped with the ones presented in the previous subsection, mainly the parameterization problem.

5.1.3 Third phase-Dataset acquisition in Edge

In this phase more tools were used other than the system presented in section 2.3, such as the linux iperf tool to monitor the datastream. The problem with this data is that, as we will see in the next subsections, the features that were used for both the MLP and the RNN are differences between consecutive values, and that caused that the core and edge to have similar values, making it so the models obatined for these datastes to be very similar in terms of results to the ones obtained for the first phase datasets.

5.1.4 Fourth phase-Dataset acquisition in a realistic scenario

To balance the not expected results of the previously mentioned edge datasets and in line with the BUBBLES project milestones, real drone data was acquired by the UPV team using the system developed by the Coimbra team, which allowed for the use of some realistic datasets. However, these datasets were not indicative of rules, guidelines or studies that involved a safe control of drones, which impacted the work as we will see in the section section 5.3.
Furthermore, due to the fact that distinct units were used for the UPV/injector team it was discovered that these datasets could not be used in the training of the model, nonetheless since there was work in adapting the system to allow a the creation of these datasets there was a need to discuss it.

5.2 Data set analysis

After obtaining the datasets it was important to analyse them to identify interesting characteristics, especially depending on the parameters utilised. This section presents both the datasets in graphical form as well as the analysis.

5.2.1 Normal operation

The first phase is to set a baseline for the datasets establishing their normal properties and creating assumptions for how the errors will impact them. The following graphs will represent the time normalized in the simulation interval, where 0 is the instant the first packet is received and 1 is the instant where the last packet was received, compared to the difference between the time it was received with the time it was sent in microseconds.

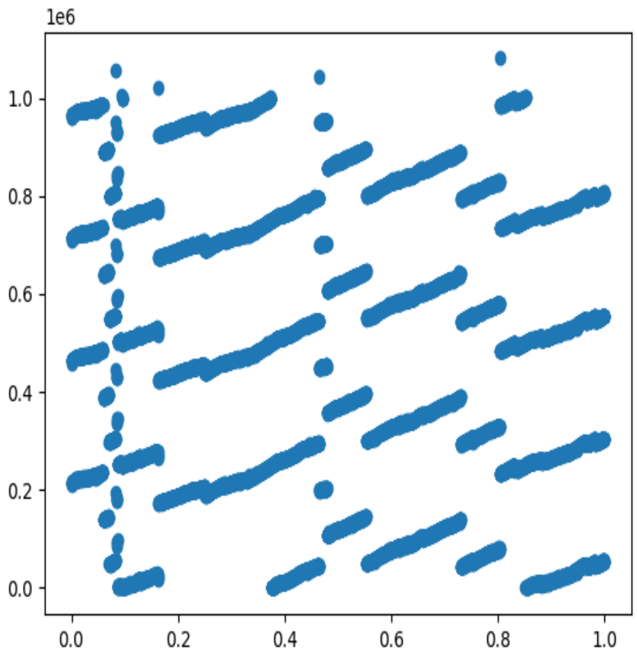


Figure 5.1: Normalized time received/time difference for 1 drone - normal distribution

The initial expected result was a linear distribution where most of the packets would take the same amount of time with some variance, however as we can see in Figure 5.1 while the interval of differences is delineated, some interesting aspects can be observed, such as the initial packets having a larger variance of difference between the time they were sent and the time they were received as well as some patterns in some ranges of time. This was probably caused due to some queueing effects, in an initial phase were expected to be exacerbated by the network problems.

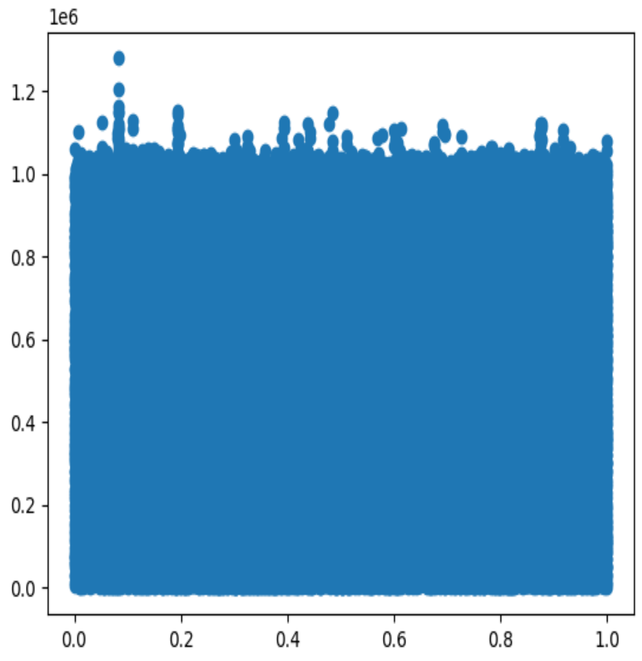


Figure 5.2: Normal distribution for 5 drones

To verify the scalability of the models, there was a need to increase the number of drones in the simulation, represented in Figure 5.2. Contrary to an initial supposition instead of making the intervals significantly larger, the main difference with increasing the number of drones was the increase in the spread of values making it so the area between 0.0 and 1.0^6 microseconds to be fully occupied with points. This phenomena repeats itself for the close interval of 5 to 10 drones, however there were no studies that involved more than 10 drones. It is interesting to denote, that for the moment of around 0.1 we can see the greatest difference.

5.2.2 Faults

Based on the distributions discussed in subsection 5.2.1 it was expected that adding loss or delay would create a more random variance for the use of 1 drone and that for the use of more than 5 drones the interval of differences would increase, nevertheless the results obtained differed. The following graphs will show a run with errors where the failures started being injected from 0.33 to 0.66.

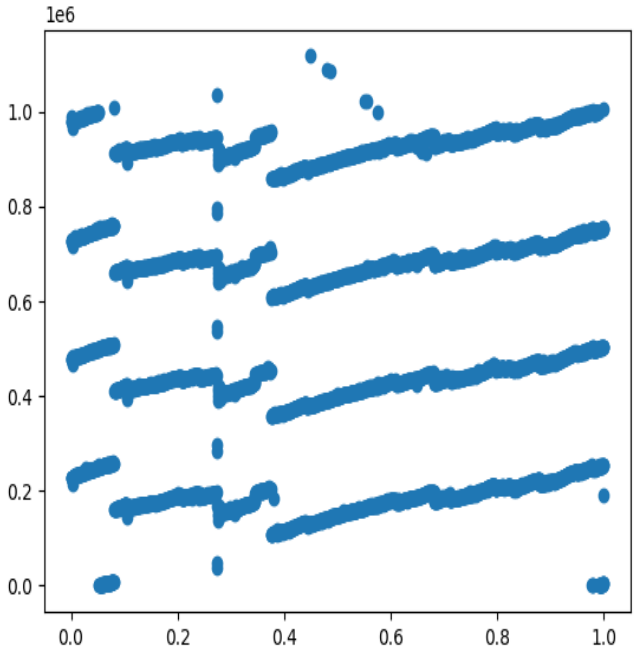


Figure 5.3: Normalized time received/time difference 1 Drone - Error distribution

As we can see in Figure 5.3 instead of a more random behaviour with the distributions of the delay, what was instead presented was a more stable queuing phenomena. Furthermore, the maximum delay wasn't increased substantially. However, in both the 0.33 and 0.66 mark of the time, we can notice a shift in the values, indicating that the faults have a noticeable impact on the datasets.

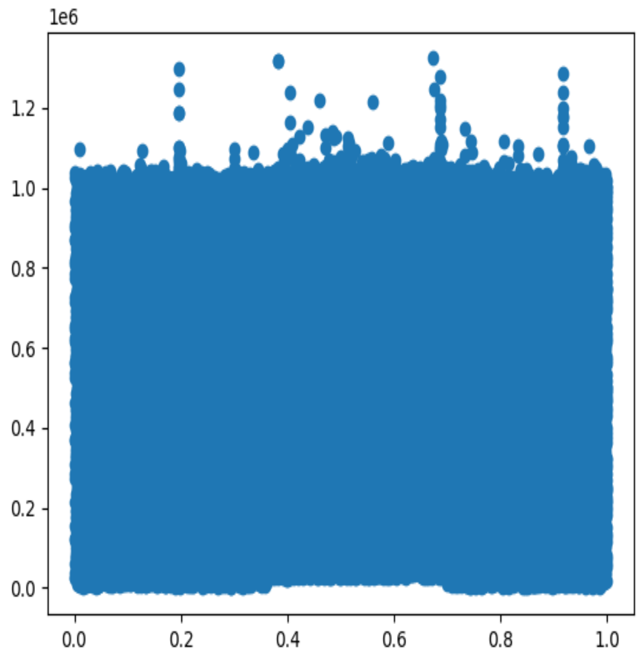


Figure 5.4: Error distribution for 5 drones

Figure 5.4 seems very similar to the normal datasets, however there are some noticeable differences such as the increased number of points above the 1^6 microseconds and the slight increased in the minimum value of the delay in the fault interval.

Since loss is also introduced in these datasets it is also important to observe the distribution of consecutive message numbers, through the difference between them. In Figure 5.5 and Figure 5.6 the difference between consecutive messages is presented for 1 drone, since consecutive messages between different drones wouldn't be consistent due to queuing phenomena. The first value being 0 signifies the first message that is compared to itself, used for convenience in the analysis.

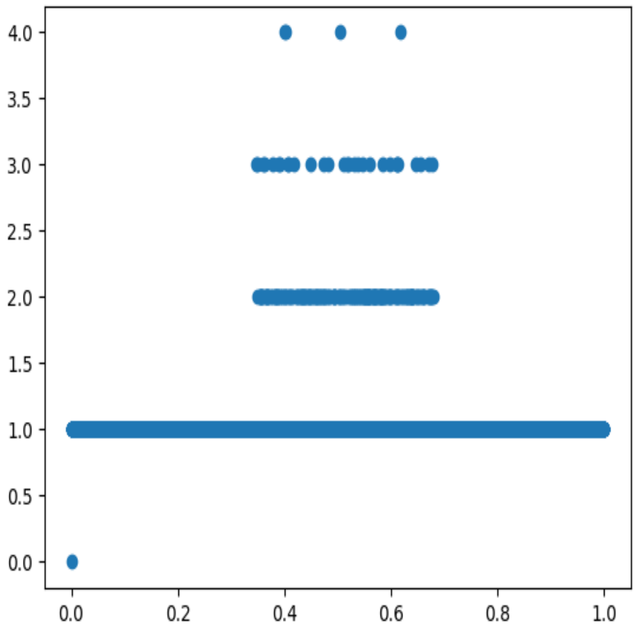


Figure 5.5: Differential of messages per normalized time for 1 drone

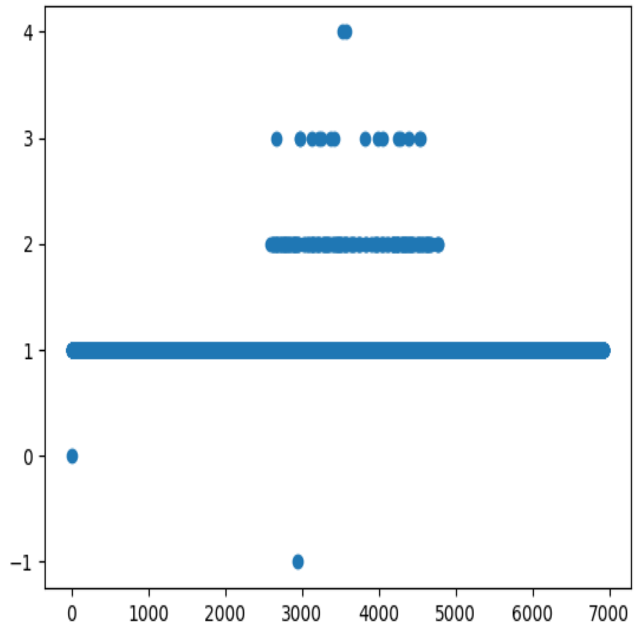


Figure 5.6: Differential of messages per normalized time for 1 drone in 5 drone simulation

It is important to notice that in the interval where the faults are injected there starts to be a delay in the consecutive packet number. Furthermore, the bigger the delay the less usual it occurs, noticeable for the value of 4 that has in both cases less than 4 occurrences.

This distribution was the expected before the datasets were created. The case of the one negative value in Figure 5.6 that is also the one that show the need for the use of both loss and delay, since it could only happen if there is a delay and the retransmited message is lost.

After this session of dataset acquirement there was the need to analyse the datasets that had no QoS system active.

5.2.3 No QoS errors

In the dataset for faults with no QoS, several things were expected to happen. Firstly, it was expected that the delay datasets wouldn't be too impacted, since the packets would never reach the core, but the loss graph would have an increase in the number of ocurrences and the difference between consecutive messages.

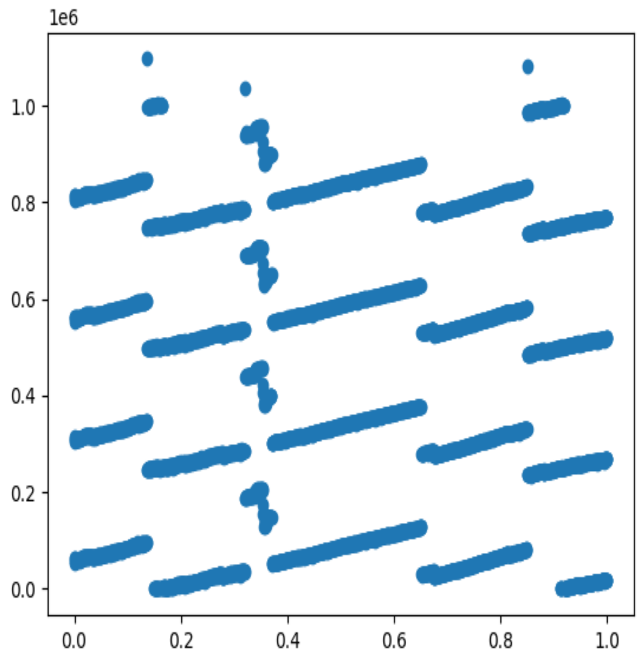


Figure 5.7: Normalized time received/time difference for 1 drone - no QoS distribution

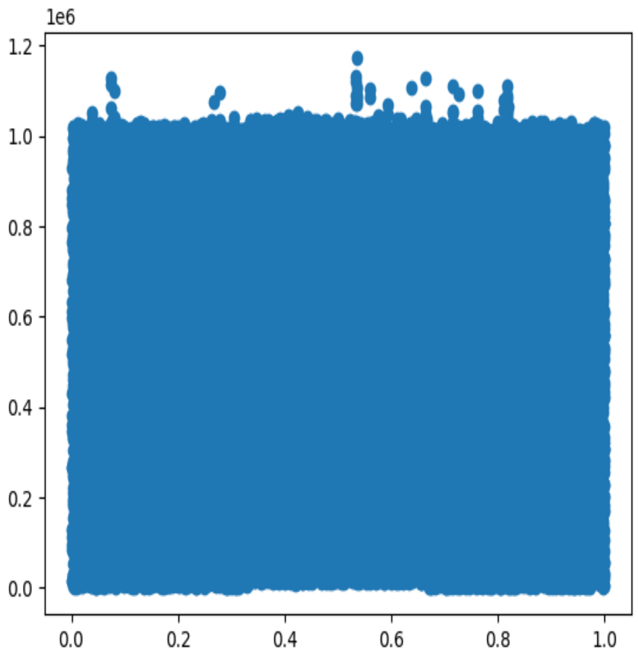


Figure 5.8: Normalized time received/time difference for 5 drones - no QoS distribution

We can see in Figure 5.7 that it is similar to the other scenarios, however there are less irregularities in each pattern. This happens due to the fact that since messages are being lost the TCP queue is less full, which causes a more continues dispersion of delays. This phenomena can be extrapolated to Figure 5.8 where we can notice that the values greater than 1.0^6 are less frequent and less distant.

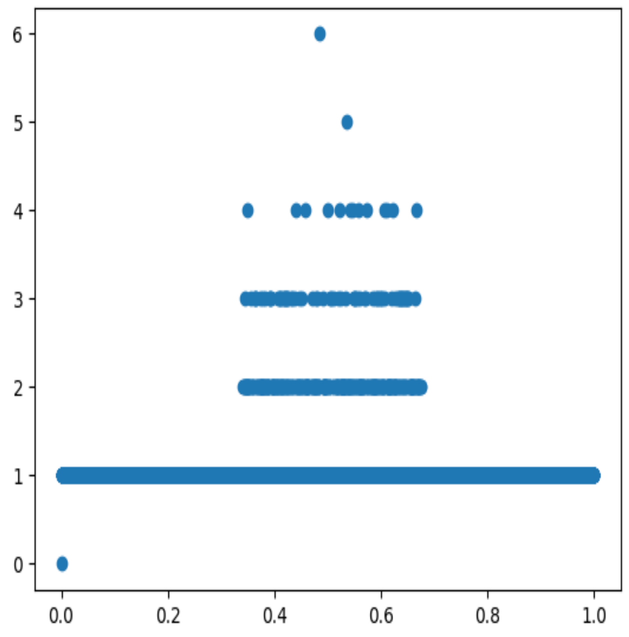


Figure 5.9: Differential of messages per normalized time for 1 drone with no QoS

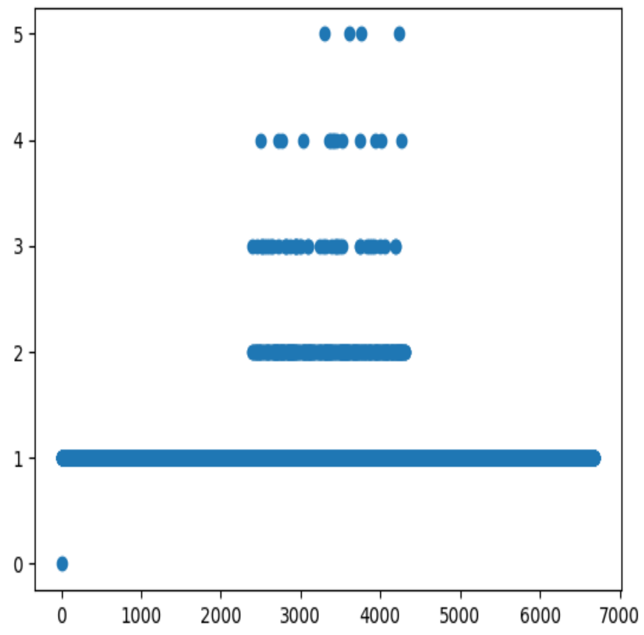


Figure 5.10: Differential of messages per normalized time for 1 drone in 5 drone simulation with no QoS

For the loss, the proposed impact of the disabling of the QoS was verified, being an increase in both the number of occurrences and their value.

5.3 Development of the agents

In this section, we will discuss how the agents were created in more concrete terms, such as the datasets utilised, the training method, the crossvalidation method.

5.3.1 CrossValidation Method

For the purpose of verifying if the results are significant, in this work an adapation of the repeated random sub-sample validation method was used. This method consists in extracting some samples for testing from the training set, depending on a probability, in this case 33%, meaning that around a third of the dataset is used for validation. Each of the values is guaranteed to be used either for training set or dataset.

5.3.2 ARIMA

To utilise the time series analysis the dataset had to be transformed into a time series. To achieve this the unit utilised was number of packets per second, which allows for the implicit utilisation of the features used in the following agents.
In terms of training, it utilises the first half of the values to obtain the coefficients and the testing set would be the remaining data. Furthermore, in terms of testing this model, contrary to the following sections, the for of measure is the NRMSE to be in line with the values presented in section 2.1, since this model can't expressly detect errors.

5.3.3 MLP

For the Multi-layer perceptron (MLP), it was necessary to define the features to be used and, since the MLP does not use previous results directly to train there was a need to create those features in a way that allowed for a time window to be considered. To achieve this a window of values was used. The window corresponded to the double of the number of packet differences considered, because half pertained to the delay from the previous packet (the odd index) and the rest to the difference of the checksum.
In terms of the model itself, it has 100 neurons in the hidden layer utilizing the tanh activation function, using the adam solver and the learning rate of 0.01. In a first attempt the learning rate of 0.1 was used, however the model never converged.
In terms of training 30 different models were obtained, furthermore each model was trained by achieving either 1000 iterations or 150 iterations with no changes. The trainig of each model used a specific seed so it can be replicable.

5.3.4 RNN

The construction of the Recurrent Neural Network (RNN) was very similar to the construction of the MLP, however the RNN only needed to ascertain the number of datapoints that were considered as inputs, due to the fact that the features .
The model consists of 2 layers: the Simple RNN layer and the Dense layer. The first layer is the one where the features are processed while the second one compounds the values to obtain the output which is the probability of the existence of an error; to simplify the

calculus of the statistical measurement this probability is rounded and that value is the one used to evaluate the agent. The parameters of the model are different to the MLP agent being 150 hidden nodes, due to the higher complexity, the number of epochs is 50 due to time constraints, the activation function for the Simple RNN layer is the tanh and for the Dense layer the sigmoid function is used.

5.3.5 Bayesian network

As was mentioned previously, the Bayesian network was one of the agents with the greatest chance of not being able to succeed, due to the datasets of the paper where the idea of the implementation originated. However the problem that was found was that since the UAV sector hasn't been developed sufficiently for the creation of the proper distributions of the states. Nevertheless, there is value in the creation of the model even if the implementation isn't feasible in this case, due to the possible modularity and ease of integration in future works.

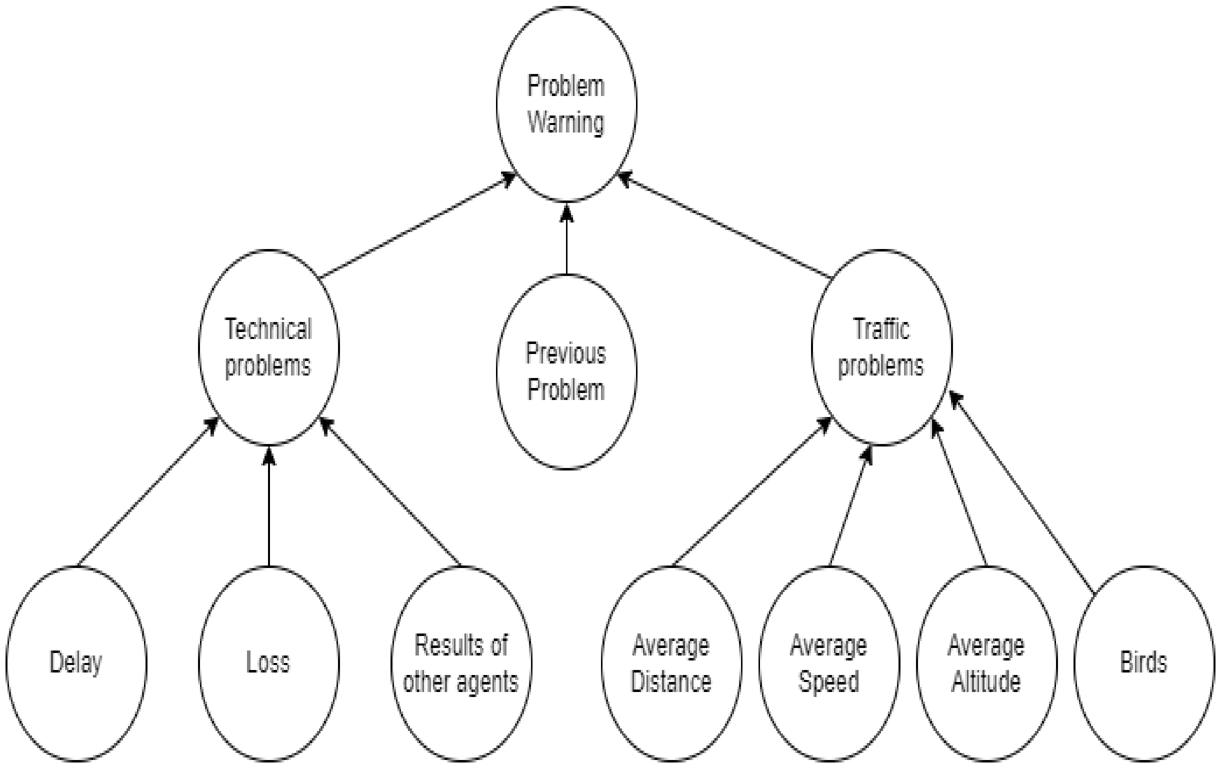


Figure 5.11: Proposed Bayesian Network

Figure 5.11 is the proposed bayesian network. It includes three main states: technical problems, previous problems and traffic problems; the first identifies the same type of fault as the rest of the agents, being things such as delay and loss of the network and the possibility of stacking the other existing agents utilising the possibility of error. Previous problem is used due to the fact that the bayesian network is a state machine and it is believed that errors that occur will extend for an amount of time; this state could be changed to allow for a bigger window of previous states. Finally, the traffic problems state is dependent on the physical condition of the space where the drones fly and as such the substates pertain to such aspects as the average distance between drones, the average speed of the UAV group, the average altitude where the drones are flying and, as a normal danger to aviation, the density of birds in the area. This last state was the one that created the impossibility of the implementation, since the idea behind it was to use norms or regulations, much like for normal traffic, however, as stated previously, the UAV area hasn't been explored to that point, so it would be an agent based entirely on estimations. Nevertheless, due to the nature of Bayeseian Networks when these regulations are established the implementation of this model could bring the previously proposed benefits.

Chapter 6

Results and discussion

In this chapter the several different results will be presented, as well as the progress of the objectives presented in section 3.2 and the Gantt diagram of the work developed as well as a subsequent analysis of each.
In terms of objective results, this chapter is divided by the different types of agents and then those agents results are divided in the results of the agents themselves and a discussion on the values obtained.

6.1 Time Series Analysis

Trained	Tested	Mean	STD
5 drones	5 drones	0.06	0.03
5 drones	10 drones	0.52	0.01
10 drones	10 drones	0.053	0.030
10 drones	5 drones	0.89	0.09

Table 6.1: Arima values in NRMSE for different training and test

We can notice that for the results pertaining to different number of drones for training and testing that the error increases and the deviation is also larger. Nevertheless, compared with the results presented in chapter 2 the error has been smaller for most cases.

6.1.1 Discussion

As we can see in the results the agent is very resilient for determining the next value for the same number of drones. However, the error increases when the number of training and testing differ, even though those cases are also comparable to the studies presented.
The behaviour with the differing number of drones was expected, however the fact that the error was comparable to the studies was not. This possibly indicates a very similar behaviour for the time series distribution, which indicates that utilising the feature presented for the ARIMA allows for a good prediction of the values, being it's greatest fault that it does not understand explicitly when there is a change, needing either a human operator or another agent to trigger the error warning.
In the initial plan it was not expected to develop an agent for the edge or for QoS disabled, due to the fact that it was expected that the values for this agent to be worse, nonetheless

with the results obtained we can conclude that there would exist value in exercising that experimentation.

6.2 Determining window

In this section some agents were prepared, due to the need of the decision of the parameterization, since contrary to the ARIMA there doesn't exist any mathematical method to determine the correct parameters without experimentation.
The main reason was to determine which window should be used for subsequent studies.

	Mean	Std
Accuracy	0.93	0.01
Precision	0.87	0.03

Table 6.2: MLP for dataset with window 2

	Mean	Std
Accuracy	0.96	0.01
Precision	0.93	0.02

Table 6.3: MLP for datasets with window 3

	Mean	Std
Accuracy	0.98	0.01
Precision	0.96	0.01

Table 6.4: MLP for datasets with window 5

As we can see in the above tables the bigger the window the better the results obtained, however, due to the attempt at maintaining realism the window of 5 elements was not chosen, since it would involve the arrival of 6 different packets and it doesn't increase substantially from the values of Table 6.3, thus the value for the window will be 3. Furthermore, since the values themselves seem to be quite high the learning rate of 0.01 will be used, as well as the tanh and adam solver; in future works it might be interesting to test different parameters, even so since it was not the objective of this work and due to the already high values obtained that study wasn't accomplished.

6.3 MLP

This section will present and discuss the results pertaining to the MLP agent, following the timeline of the different developments presented in the Gantt diagram.

6.3.1 With QoS

	Mean	Std
Accuracy	0.87	0.01
Precision	0.77	0.02

Table 6.5: MLP for datasets with QoS using 5 drones

	Mean	Std
Accuracy	0.88	0.01
Precision	0.78	0.02

Table 6.6: MLP for datasets with QoS using 10 drones

The results presented will represent the baseline for the rest of this section. We can see that for both cases the results were greater than 80% accuracy and over 75% precision, indicating that the state detection is very consistent. Moreover, with less than 3% values for standard deviation these model are very consistent in the classification.

6.3.2 Without QoS

	Mean	Std
Accuracy	0.87	0.01
Precision	0.84	0.10

Table 6.7: MLP for datasets with and without QoS using 5 drones

	Mean	Std
Accuracy	0.88	0.01
Precision	0.80	0.10

Table 6.8: MLP for datasets with and without QoS using 10 drones

Comparing the results with the ones from *with QoS*, we can notice that they are strictly better in terms of mean for precision while maintaining similar accuracy. However, for the standard deviation, we can denote an increase of 9%.

6.3.3 In edge

	Mean	Std
Accuracy	0.86	0.01
Precision	0.79	0.02

Table 6.9: MLP for datasets from edge using 5 drones

	Mean	Std
Accuracy	0.87	0.01
Precision	0.79	0.03

Table 6.10: MLP for datasets from edge using 10 drones

The values obtained for the edge are very similar with the baseline, being the main difference reflected in the values of the precision that are higher, although the standard deviation also increased.

6.3.4 Discussion

The values obtained for the MLP were higher than expected, being better than the RNN results. Even though it was not expected, just as was presented in the SoA, these values are not incongruous with the beginning suppositions.
A possible reason for these events is the simplicity of the data and the impact of each feature that possibly is overutilised in the RNN. Another explanation is the fact that the RNN agent overestimates the importance of the order of values, something that is not implicit in the MLP classifier.
However, the fact that the results have higher values than 0.85 for the mean of the accurarcy and greater than 0.75 for precision demonstrates that the agent is robust and that it should be tested in a controlled, realistic scenario to better observe its behaviour.
Moreover, comparing the three different scenarios *with QoS*, *without QoS* and *in edge* we can observe that two phenomena. The first is that the *without QoS* version has better results compared to the baseline and the second is that the *edge* and *with QoS* are very similar. The relation observed in the second phenomena was expected from the moment when the differentials were used, since in terms of distribution they would be very similar, only possibly being affected by the queuing phenomena between the edge and the core, however the first phenomena is unexpected. A possible reason for its occurrence is that, since the impact on the differential between the time of the packets arrival is indistinguishable from the normal operation to the operation with errors, the alteration of normal values from the second feature is very determinative of the state.
With these observations some conclusions can be made; for example, there was the proposition that disabling the QoS dynamically in faulty situations would allow for a more efficient system and these results permit us to conclude that it would be an interesting idea, since the agents can very accurately detect for both scenarios when the faults exist. Furthermore, the nested agent in the edge seems to be an idea that should be considered, since it would be faster in warning the drones and is equally accurate.

6.4 RNN

6.4.1 With QoS

	Mean	Std
Accuracy	0.81	0.01
Precision	0.72	0.03

Table 6.11: RNN for datasets with and without QoS for 5 drones

	Mean	Std
Accuracy	0.78	0.02
Precision	0.70	0.03

Table 6.12: RNN for datasets with and without QoS for 10 drones

Just like the previous section, the results of RNN *with QoS* will be used as a baseline for the rest of the RNN agents.
Contrary to the previous section however the values of both accuracy and precision are lower than 85 and 75% respectively. Furthermore, for most results the standard deviation seems to be larger.

6.4.2 Without QoS

	Mean	Std
Accuracy	0.78	0.02
Precision	0.69	0.04

Table 6.13: RNN for datasets with and without QoS for 5 drones

	Mean	Std
Accuracy	0.76	0.01
Precision	0.67	0.03

Table 6.14: RNN for datasets with and without QoS for 10 drones

As we can observe in the above tables, the values of RNN strictly decreased from the cases where QoS was active. Furthermore, the standard deviation increased in most cases, revealing a larger variety of results.

6.4.3 In Edge

	Mean	Std
Accuracy	0.82	0.01
Precision	0.72	0.03

Table 6.15: RNN for datasets from edge for 5 drones

	Mean	Std
Accuracy	0.79	0.02
Precision	0.71	0.02

Table 6.16: RNN for datasets from edge for 10 drones

Just like with MLP, these results are very similar to the baseline. Contrary to the MLP case, however there isn't a determinant difference that distinguishes the values.

6.4.4 Discussion

Opposing the results of the last agent, the RNN under performed relative to the expectation even though it has acceptable results.
The results obtained for the RNN were near 80% accuracy and 70% precision, being probably caused due to the overestimation of the order of the data. It might also be caused by miss-parameterization or negatively affected by the cross validation method chosen.
Either way more tests should be done with this model, to determine if there exists a solution that can increase the accuracy and precision, placing the values closer to the ones obtained by the MLP classifier.
Utilising this agent in its current state for the classification of more complicated cases such as QoS switching is not advised, however the use of the edge agent is recommended in the case where the RNN is to be considered due to the same reason presented in the MLP discussion.

6.5 Result comparison

In this section some comparisons of agents, advantages and disadvantages will be discussed.
The first agent the *Time Series Analysis* is more difficult to compare relative with the two other agents, however a possible advantage is how it is a more "analog" agent, as in the classification isn't made intrinsically by the agent, allowing for a greater human extrapolation and will always provide value, since, even with error, it will always give an approximation of the next value, not being a binary classification. Nonetheless, it's greatest disadvantage is the fact that there needs to be some specialization of the operator to understand the values that are being predicted, creating a barrier to entry in the system.
The two remaining agents have a more direct comparison, however, with these datasets, the MLP classifier is more consistent than the RNN. There is a high probability that in a more complex setting, such as the realistic scenario, the RNN agent could provide some advantage that is not noticeable in this study.
In terms of a possible implementation, i would argue that using both the *Time Series Analyses* as well as the *MLP* in both the edge and with QoS change would be the configuration that would grant the greatest amount of knowledge.
For future works there are several possibilities: the direct comparison between the edge agent and the QoS switching would be interesting, analysing if one manages to detect faulty state when the other couldn't; furthermore there is a need to explore the RNN agent in more complex situations.

Chapter 7

Conclusion

To reiterate the beginning of this thesis, UAS's are a new concept that have the potential to impact the economy in a global levels, allowing new services that weren't possible before. It was also discussed some dangers that appeared connected to this new concept, especially compared with the already existing manned aviation, so we presented some possible countermeasures that the SESAR JU and especially BUBBLES try to implement.

After that, some technologies and techniques were exposed, the technologies, mostly related with the environment, allowed the comprehension of the system and the techniques that were analysed, in an attempt to better understand how to implement the models that are the main objective of this thesis.

The general work plan is exposed to both structure and delineate clearly how the objectives will be achieved. This also allows a better understanding of the next section of the preliminary phase that allow us to conclude that the project wasn't behind schedule at the milestone of the first semester, as we can see especially one the first Gantt diagram.

The main concern presented in the beginning of the second half of the semester was the practical nature of the work, which could lead into problems, which indeed happened especially in the case of the dataset acquisition, which delayed the rest of the work considerably and diminishing the amount of work that could be alteredadded explicit in the second Gantt diagram where we notice this cascading effect.

In terms of the results themselves and the techniques researched, however the project was a success, since there are several agents that have over 0.8 accuracy and over 0.75 precision, indicative of reliable agents and the not tested Bayes network has potential for when the drone space regulation is better established.

Another part of the work that is interesting is the possibility for the future, with the implementation of the previously mentioned Bayesian network, as well as a more rigorous parameterization attempt and the creation of more generic models, constructed in such a way that even for a larger scale or dynamically changing the number of drones does not create problems with the identification of network faults.

It is also important to reflect on the problems of this project, mainly being the underestimation of creating a credible datatset for an unexplored area. This work would have been able to be more complex if those tasks were more accurately estimated and if a better synchronization between the BUBBLES team occurred this work could have been substantially improved by the use of realistic scenarios.

In conclusion, the objectives were accomplished even though the time frame was not fully respected, however the possibilities of future work and the agents themselves demonstrate that this area has substantial potential that can be achieved if there is effort directed into the area.

References

[1] Apache kafka documentation. `https://kafka.apache.org/documentation/`. 2021-12-20.

[2] Data distribution service. `https://www.dds-foundation.org/what-is-dds-3`. 2022-01-05.

[3] Gazebo simulator. `https://gazebosim.org`. 2022-01-06.

[4] Linux documentation project, traffic control using tcng and htb howto. `https://tldp.org/HOWTO/Traffic-Control-tcng-HTB-HOWTO/intro.html`. 2022-01-06.

[5] Linux foundation, linux netem network emulator. `https://wiki.linuxfoundation.org/networking/netem`. 2022-01-06.

[6] Mavlink. `https://mavlink.io/en/`. 2022-01-05.

[7] Px4 autopilot. `https://px4.io/`. 2022-01-06.

[8] Sesar joint undertaking - u-space. `https://www.sesarju.eu/U-space`. 2022-01-05.

[9] Huifang Feng and Yantai Shu. Study on network traffic prediction techniques. *Proceedings. 2005 International Conference on Wireless Communications, Networking and Mobile Computing, 2005.*, 2:1041–1044, 2005.

[10] Erwin Harahap, Wataru Sakamoto, and Hiroaki Nishi. Failure prediction method for network management system by using bayesian network and shared database. In *8th Asia-Pacific Symposium on Information and Telecommunication Technologies*, pages 1–6, 2010.

[11] Tiago Prado Oliveira, Jamil Salem Barbar, and Alexsandro Santos Soares. Computer network traffic prediction: a comparison between traditional and deep learning neural networks. *Int. J. Big Data Intell.*, 3:28–37, 2016.