FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

# Routing for Flying Networks using Software-Defined Networking

**André Duarte Correia de Oliveira**

MASTER IN ELECTRICAL AND COMPUTERS ENGINEERING

Supervisor: Dr. Rui Lopes Campos

Co-Supervisor: Eng. André Filipe Coelho

July 21, 2019

# Abstract

In recent years, Unmanned Aerial Vehicles (UAVs) are being increasingly used in various applications, both military and civilian. Their miniaturisation and low cost paved the way to the usage of swarms of UAVs, which provide better results when performing tasks compared to single UAVs. However, to enable cooperation between the UAVs, always-on and reliable communications must be ensured. Moreover, swarms of UAVs are being targeted as a way to provide Internet access to ground users in scenarios such as disaster reliefs and Temporary Crowded Events (TCEs), taking advantage of the capability of UAVs to carry Wi-Fi Access Points (APs) or Long-Term Evolution (LTE) Base Stations (BSs). Solutions relying on a Control Station (CS) capable of positioning the UAVs according to the users' traffic demands have been shown to improve the Quality of Service (QoS) provided by the network. However, they introduce important challenges regarding network routing.

Recently, a solution was proposed to take advantage of the knowledge provided by a CS regarding how the network topology will change, by dynamically updating the forwarding tables before the links in the flying network are disrupted, rather than recovering from link failure, as is the case in most of the existing routing protocols. Although it does not consider the impact of reconfigurations on the access network due to the mobility of the APs, it is a promising approach worthy of being improved and implemented in a real system.

In this dissertation, a routing solution for flying networks based on Software-Defined Networking (SDN) was developed. This solution addresses the mobility management and network load balancing challenges from a centralised perspective, while simultaneously enabling uninterruptible communications between ground users and the Internet, thus allowing UAVs to reposition and reconfigure themselves without disrupting the terminals' connections to the network.

# Resumo

Nos últimos anos, os Veículos Aéreos Não Tripulados (UAVs) estão a ser usados de forma crescente em inúmeras aplicações, tanto militares como civis. A sua miniaturização e o preço reduzido abriram o caminho para o uso de enxames de UAVs, que permitem melhores resultados na realização de tarefas em relação a UAVs independentes. Contudo, para permitir a cooperação entre UAVs, devem ser asseguradas comunicações contínuas e fiáveis. Além disso, os enxames de UAVs foram identificados como meio para permitir o acesso à Internet a utilizadores terrestres em cenários como prestação de socorros e Eventos Temporários Lotados (TCEs), tirando partido da sua capacidade para transportar Pontos de Acesso (APs) Wi-Fi e Estações Base (BSs) *Long-Term Evolution* (LTE). Soluções que dependem de uma Estação de Controlo (CS) capaz de posicionar os UAVs de acordo com as necessidades de tráfego dos utilizadores demonstraram aumentar a Qualidade de Serviço (QoS) oferecida pela rede. No entanto, estas soluções introduzem desafios importantes no que diz respeito ao encaminhamento do tráfego.

Recentemente, foi proposta uma solução que tira partido do conhecimento da CS sobre o estado futuro da topologia da rede para atualizar dinamicamente as tabelas de encaminhamento, de modo a que as ligações na rede voadora não sejam interrompidas, em vez de se recuperar da sua interrupção, como é o caso na maioria dos protocolos de encaminhamento existentes. Apesar de não considerar o impacto das reconfigurações na rede de acesso, como consequência da mobilidade dos APs, ou o balanceamento da carga na rede, esta abordagem é promissora e merece ser desenvolvida e implementada num sistema real.

Nesta dissertação foi desenvolvida uma solução de encaminhamento para redes voadoras baseada em *Software-Defined Networking* (SDN). Esta solução dá resposta aos desafios de mobilidade e de balanceamento da carga na rede de uma perspetiva centralizada, garantindo simultaneamente comunicações ininterruptas entre utilizadores em terra e a Internet, permitindo assim que os UAVs se possam reposicionar e reconfigurar sem interromper as ligações dos terminais à rede.

# Acknowledgements

I would like to express the respect I have for my supervisors, Prof. Rui Campos and Eng. André Coelho. Thank you for guiding me throughout the journey that was this dissertation.

To my friends, thank you for all the conversations we had during this dissertation. Two heads are better than one.

For my family, a deep feeling of gratitude for always encouraging me. This would not have been possible without you. A special thanks to my brother Leonel, who is always present when I need his help, and to my brother Tiago, who can bring joy to the most mundane of days.

André

*"I ask not for a lighter burden, but broader shoulders."*

Seneca

# Contents

# List of Figures

# List of Tables

# Acronyms and Abbreviations

| | |
|---|---|
| AP | Access Point |
| API | Application Programming Interface |
| ARP | Address Resolution Protocol |
| BS | Base Station |
| BSS | Basic Service Set |
| BSSID | Basic Service Set IDentifier |
| CS | Control Station |
| CSA | Channel Switching Announcement |
| DFS | Dynamic Frequency Selection |
| DHCP | Dynamic Host Configuration Protocol |
| DNS | Domain Name Service |
| DS | Distribution System |
| DSSS | Direct Sequence Spread Spectrum |
| ESS | Extended Service Set |
| FANET | Flying Ad-hoc Network |
| FCC | Federal Communications Commission |
| FF | Fittingness Factor |
| FHSS | Frequency Hopping Spread Spectrum |
| FMAP | Flying Mesh Access Point |
| FMN | Flying Multi-hop Network |
| GPS | Global Positioning System |
| HWMP | Hybrid Wireless Mesh Protocol |
| IBSS | Independent Basic Service Set |
| IEEE | Institute of Electrical and Electronics Engineers |
| IoT | Internet of Things |
| IP | Internet Protocol |
| ISM | Industrial, Scientific and Medical |
| JSON | JavaScript Object Notation |
| JVM | Java Virtual Machine |
| LoS | Line-of-Sight |
| LTE | Long-Term Evolution |

| | |
|---|---|
| LVAP | Light Virtual Access Point |
| MAC | Medium Access Control |
| MANET | Mobile Ad-hoc Network |
| MIMO | Multiple-Input Multiple-Output |
| MN | Mobile Node |
| MVAP | Multi-connection Virtual Access Point |
| NAT | Network Address Translation |
| NIC | Network Interface Card |
| OFDM | Orthogonal Frequency Division Multiplexing |
| OS | Operating System |
| OvS | Open vSwitch |
| PHY | Physical Layer |
| QoE | Quality of Experience |
| QoS | Quality of Service |
| RF | Radio Frequency |
| RSGFF | Recovery Strategy Greedy Forwarding Failure |
| RSSI | Received Signal Strength Indicator |
| SDN | Software-Defined Networking |
| SSH | Secure Shell |
| SSID | Service Set IDentifier |
| STA | Station |
| TCE | Temporary Crowded Event |
| TCP | Tranmission Control Protocol |
| TMFN | Traffic-aware Multi-tier Flying Network |
| TPC | Transmit Power Control |
| UAV | Unmanned Aerial Vehicle |
| UDP | User Datagram Protocol |
| USB | Universal Serial Bus |
| VANET | Vehicular Ad-hoc Network |
| VAP | Virtual Access Point |
| VLAN | Virtual Local Area Network |
| VNF | Virtualised Network Function |
| WLAN | Wireless Local Area Network |

# Chapter 1

# Introduction

## 1.1 Context

Over the past few years, Unmanned Aerial Vehicles (UAVs) have been increasingly used for various applications, both military and civilian, such as search and rescue operations, border surveillance, managing wildfires, relays for ad hoc networks, wind estimation, remote sensing and traffic monitoring [1]. More recently, the advent of small size and low cost UAVs paved the way to the usage of swarms of UAVs, which increase effectiveness and efficiency when performing tasks in comparison to a single UAV. However, these multi-UAV systems require good and reliable communications in order to cooperate when performing tasks [1].

In this sense, swarms of UAVs carrying network hardware, including Wi-Fi Access Points (APs) and Long-Term Evolution (LTE) Base Stations (BSs), are being studied as a way to provide connectivity for mobile terminals where network infrastructure does not exist or is inadequately prepared to respond, including in Temporary Crowded Events (TCEs). In [2], the authors explore a communications solution comprised of Flying Mesh Access Points (FMAPs) that position themselves according to the users' traffic demands, in order to provide always-on broadband Internet access. For this purpose, the positions of the UAVs are defined centrally and autonomously. Due to the dynamic behaviour of such networks, it becomes imperative to address the routing challenges introduced in order to ensure the existence of high capacity and uninterrupted paths between the UAVs and meet both the Quality of Service (QoS) and Quality of Experience (QoE) expected by users.

## 1.2 Problem and Motivation

Nowadays, users around the globe can connect to the Internet using a myriad of devices. One way this can be achieved is by means of a wireless link, in which a user does not need to be physically connected to the network. Instead, a mobile terminal connects to an access node that can be used as a relay for data being exchanged.

In situations where Internet access is required but no capable infrastructure is present (e.g., music festivals and rescue missions), a network of UAVs carrying APs can be deployed, as proposed in [2]. However, the dynamic and reconfigurable nature of this network brings up significant routing challenges in order to ensure both QoS and QoE. Typically, state of the art routing protocols only react to link failures after they occur, which implies degradation in the overall service of the network.

In [3], the authors introduce a centralised routing solution for Flying Multi-hop Networks (FMNs) that takes advantage of the holistic knowledge provided by a Control Station (CS) to predict the future state of the network. This CS, as suggested in [2], contains the full vision of the network and its traffic demands, and is thereby responsible for dynamically adapting the network's topology to better suit the needs of its end-users. However, this routing solution does not take into account the impact of access network reconfigurations on mobile terminals. To allow uninterrupted connectivity between mobile terminals and the Internet, the solution should be improved to address the roaming of ground users between APs, making handovers seamless from the user perspective. With a CS orchestrating the UAVs' positions, there is also the need to ensure that connections are not disrupted while the network repositions itself. Yet another important consideration is the scalability of such a system: an increase in the number of users and, consequentially, APs, can cause network unbalance, forcing a heavier load on some of the nodes and thus leading to potential bottlenecks, packet loss and longer delays. Nevertheless, given the promising results of such a routing approach, validated in simulation environment, it is important to solve its drawbacks and validate it in a real environment.

In this dissertation, the Software-Defined Networking (SDN) paradigm is the basis to provide a better response to users' needs, ensuring a centralised and flexible way of configuring and managing the network.

## 1.3 Objectives

This dissertation presents as its main objective the development of a routing solution for flying networks composed of UAVs, allowing the definition of high-capacity, uninterrupted multi-hop paths between users' terminals and the Internet, following the SDN paradigm. As such, it envisions the following specific objectives:

- Development of an algorithm for a centralised routing controller that sustains high-capacity and uninterrupted connectivity for users during UAV repositioning;

- Implementation of the centralised routing controller for 1) calculating the forwarding tables, based on a metric defined for the purpose, and 2) managing the mobility of users' terminals;

- Implementation of an agent for configuring the routing tables in the UAVs.

## 1.4 Contributions

This dissertation's key contribution is an algorithm that handles routing for flying networks in a predictive fashion. Through virtualisation, this solution allows the reconfiguration of the flying network's topology while scheduling the association of user terminals to their future APs beforehand. This algorithm was developed and implemented as an SDN application, and it was evaluated experimentally using a real testbed.

## 1.5 Document Structure

The remainder of this document consists of the following:

- Chapter 2 [State of the Art] – the fundamental knowledge and the related work that were considered for the solution proposed in this dissertation;
- Chapter 3 [Proposed Solution] – the problem statement and the solution proposed to achieve the objectives of this dissertation;
- Chapter 4 [Implementation] – the implementation of the proposed solution, including the system model, its architectural design, and its technical concepts;
- Chapter 5 [Evaluation] – the testbed used to evaluate the performance of the implemented solution against its state of the art counterparts is presented, and discussion on the obtained results;
- Chapter 6 [Conclusions] – the conclusion of the dissertation, a discussion of the achieved results, and future work.

# Chapter 2

# State of the Art

## 2.1 Introduction

This chapter covers topics considered relevant for understanding the problems addressed in this dissertation and the proposed solution. It is divided into the following sections:

- IEEE 802.11 – a brief introduction on the Institute of Electrical and Electronics Engineers (IEEE) 802.11 working group, its standards, and some concepts;
- Flying Networks – how flying networks came to be, issues surrounding them, and design approaches;
- Software-Defined Networking – its concept and relevance in flying networks;
- Routing – routing solutions for flying networks, their pros and cons, and important aspects that can help meet some networking requirements in flying networks;
- Mobility Management – mobility challenges in Wi-Fi flying networks, its key issues, and some solutions;
- Summary and Main Conclusions – the main ideas taken from the literature and their relevance in the scope of this dissertation.

## 2.2 IEEE 802.11

IEEE 802.11 [4], commonly known as "Wi-Fi"—a term coined by the Wi-Fi Alliance[1]—, is a standardisation effort for Wireless Local Area Networks (WLANs) which saw the first hint of its existence in 1985, when the United States Federal Communications Commission (FCC) released several portions of the wireless spectrum for unlicensed use. The Industrial, Scientific and Medical (ISM) radio bands, no longer requiring licensing fees, gave rise to a new IEEE committee: the IEEE 802.11 working group. This group's purpose was to create an open standard that defines

[1]"Wi-Fi Alliance" - https://www.wi-fi.org/

an over-the-air interface for how wireless stations communicate with each other. In other words, protocols and requirements are specified for the Physical Layer (PHY) and Medium Access Control (MAC) to ensure interoperability between wireless devices. This can refer both to a client communicating with an AP or multiple clients communicating with each other [5].

Following the first publication of the standard, IEEE 802.11 Legacy in 1997, several corrections, features and improvements have been added in the form of amendments (e.g., IEEE 802.11b-1999) and revisions (e.g., IEEE 802.11-2016). These are set in motion by task groups. A few of the most relevant include:

- 802.11 – provides data rates of 1 or 2 Mbps in the 2.4 GHz band for WLANs using either Frequency Hopping Spread Spectrum (FHSS) or Direct Sequence Spread Spectrum (DSSS);
- 802.11a – sometimes referred to as "Wi-Fi 2", it uses an Orthogonal Frequency Division Multiplexing (OFDM) scheme instead of FHSS or DSSS, providing data rates up to 54 Mbps in the 5 GHz band;
- 802.11b – also referred to as "High Rate" or "Wi-Fi 1", extends data rates up to 11 Mbps using only DSSS and the 2.4 GHz band;
- 802.11e – an amendment that adds QoS support to existing standards;
- 802.11g – "Wi-Fi 3", which extends data rate up to 54 Mbps in the 2.4 GHz band;
- 802.11h – Spectrum and Transmit Power Management Extensions, which included Dynamic Frequency Selection (DFS) and Transmit Power Control (TPC) to solve problems due to interference with other devices on the same 5 GHz band;
- 802.11k – "Radio Resource Measurement" enhancements providing higher layer interfaces for radio and network measurements;
- 802.11n – "Wi-Fi 4", which further extends the maximum data rate from 54 Mbps to 600 Mbps, this time in both frequency bands, 2.4 GHz and 5 GHz, by adding Multiple-Input Multiple-Output (MIMO) technology for separate spatial streams, frame aggregation and wider channels (from 20 MHz to 40 MHz);
- 802.11r – known as "Fast Basic Service Set (BSS) Transition" or "Fast Roaming", it provides enhancements to the MAC layer to minimise or eliminate the amount of time data connectivity between the Station (STA) and the Distribution System (DS) is absent during a BSS transition;
- 802.11v – "Wireless Network Management" enhancements as well as extending prior work in radio measurement, providing a complete and coherent interface for managing 802.11 devices in wireless networks;

- 802.11ac – “Wi-Fi 5” adds support for even wider channels (up to 160 MHz), more MIMO spatial streams (up to eight), downlink *multi-user* MIMO[2], and high-density modulation, enabling very high throughput in the 5 GHz band;

- 802.11ad – modifications to the PHY and MAC layers, in order to enable operation in the unlicensed 60 GHz frequency band (typically 57-66 GHz) capable of very high throughput (over 1 Gbps).

Over the years, IEEE 802.11 has been evolved to provide ever-increasing data rate capabilities, bandwidth and efficiency of the modulation. For these reasons, it is nowadays widely regarded as a fundamental technology. More information on IEEE 802.11, including active and superseded standards, task groups and timelines, can be found in [6].

IEEE 802.11 networks support two modes of operation, *Infrastructure* and *Ad-hoc*, both of which act upon the fundamental building block of the IEEE 802.11 architecture: the Basic Service Set (BSS). A BSS is defined as a set of mobile or fixed stations able to communicate directly.

### 2.2.1 Infrastructure mode

In Infrastructure mode, BSSs are connected to a DS by means of an AP. Every time a STA needs to communicate, it must do so through the AP in its BSS. APs can act as gateways and provide network services such as Network Address Translation (NAT) and Dynamic Host Configuration Protocol (DHCP) [7]. A set of BSSs connected to the same DS forms an Extended Service Set (ESS), which may in turn be connected to another IEEE 802.*X* network through what is called a Portal. Figure 2.1 depicts a network operating in Infrastructure mode.

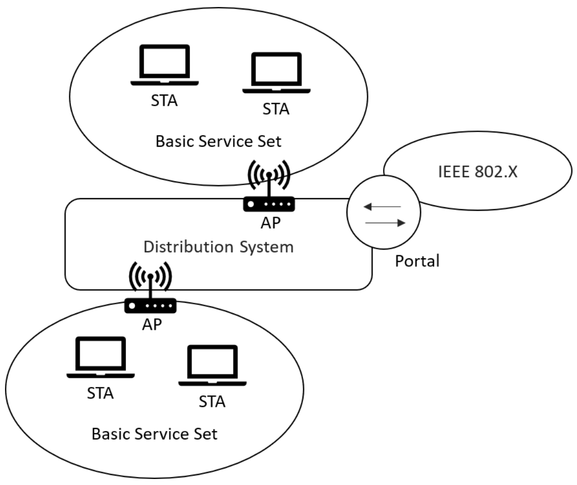


Figure 2.1: IEEE 802.11 network in Infrastructure mode.

[2]Introduced in *wave 2* of IEEE 802.11ac

### 2.2.2 Ad-hoc mode

In Ad-hoc mode, STAs can communicate directly to one another without relying on an AP. A group of STAs operating on the same BSS forms an Independent Basic Service Set (IBSS). Although this is not a typical deployment, it allows an 802.11 network to be quickly deployed and torn down without the need for in-place infrastructure, reducing costs and adding flexibility [7]. Figure 2.2 depicts a network operating in Ad-hoc mode.

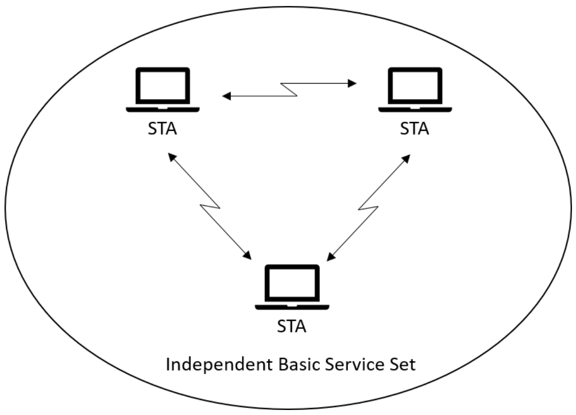


Figure 2.2: IEEE 802.11 network in Ad-hoc mode.

## 2.3 Flying Networks

With the birth of human culture and civilisation was also born our desire to connect. More than a desire, we could call it a necessity, deeply rooted in human nature. More recently, this takes on the shape of an effort to provide universal Internet access to ensure everyone and anyone who desires to do so can connect to people from all over the world. As we strive to progress as a species, our methods evolve, and our technologies become more and more sophisticated. Luxuries turn into needs as a movement towards globalisation brings about the ability to access the Internet from anywhere, anytime, and network connectivity starts to be considered an essential utility. More so, the advent of the Internet of Things (IoT) paradigm, the growing number of connected devices (such as smart phones and watches) and the way that being connected shapes everyday life pose a new challenge to the telecommunications area: providing ubiquitous network access [8]. Nowadays, the traffic volume keeps rising and its nature evolving. An example of this stems from the usage of social media, where users are encouraged and given the tools to upload their own content, very often in the form of images or video, increasing the relevance of upstream traffic.

With UAVs becoming smaller, more efficient, less costly, and with their ability to hover, deploying single or multiple UAVs as communication relays or aerial base stations is proving to be a possible solution to meet the network provisioning demands of ground users in situations where infrastructure does not exist or is inadequately prepared to respond, such as temporary events. For this purpose, IEEE 802.11, commonly called *Wi-Fi*, is worth considering [9].

In [10], Cisco predicts that by 2021 50 % of all Internet Protocol (IP) traffic will be Wi-Fi, 30 % will be wired, and 20 % will be mobile. Of all mobile data traffic, 63 % will be offloaded to the fixed network by means of Wi-Fi devices and femtocells (i.e., APs). Wi-Fi is so far the most widespread access network for providing connectivity to end-users' wireless devices.

### 2.3.1 Issues in UAV Networks

An important issue in multi-UAV systems relies on coordination and control for effective task planning. When deploying swarms of UAVs, it is important to implement an efficient algorithm to control each UAV so that the whole system can produce complex, adaptable and flexible team behaviour [11].

Another relevant aspect is that flying networks are inherently more dynamic and subject to frequent topology changes than mobile and vehicular networks. In Figure 2.3 [12], we can see that flying networks are in fact a particular case of vehicular networks, which in turn are a part of mobile networks. These are typically set up in Ad-hoc mode, being referred to as Flying Ad-hoc Networks (FANETs), Vehicular Ad-hoc Networks (VANETs) and Mobile Ad-hoc Network (MANET), respectively, and their differences are approached in [1]. Since UAVs are non-permanent, the addition and removal of UAVs in the network (e.g., due to battery exhaustion) causes topology changes. The fact that these nodes may be in motion for significant amounts of time, link quality in flying networks changes very rapidly. This leads to intermittent links and outages, causing, once again, topology changes [1, 13].

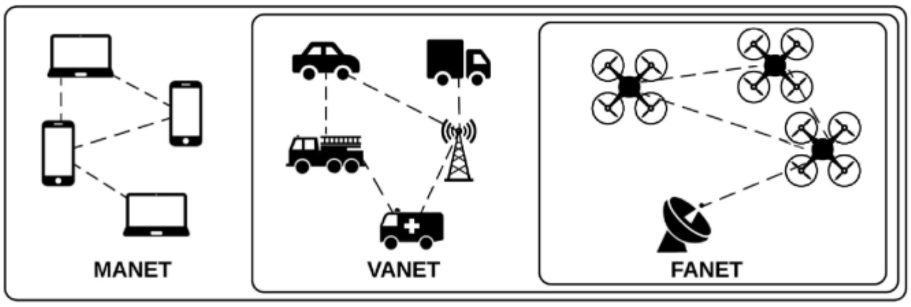


Figure 2.3: Relation between MANETs, VANETs and FANETs.

More issues may arise from the adoption of different access technologies in UAV networks. Even UAVs that use the same access technology may have problems integrating in another network due to differences in the higher layers of the protocol stack (e.g., the routing protocol in the network layer). This makes protocols limited in scalability and increases the cost of protocol development for different scenarios [13, 14].

Some other challenges introduced when deploying a UAV network include limited onboard resources, as UAVs may have limited power and processing capabilities, and the presence of intentional disruptions on the access medium, which may result in some links being severed in a certain radius [14].

### 2.3.2 Design considerations in UAV networks

#### 2.3.2.1 Infrastructure-based vs Ad-hoc

In the literature, UAV networks are usually configured in Ad-hoc mode. This is the result of extrapolating conclusions from MANETs and VANETs to UAV networks, commonly known as FANETs. However, unlike what happens in mobile and vehicular networks, depending on their application, the nodes in a flying network may range from stationary—such as when acting as Base Stations hovering above the ground—to highly dynamic. Thus, in some cases, an infrastructure-based approach may be more adequate. In applications where nodes are highly mobile, they may form a topology and select which nodes forward data in a dynamical manner, i.e., forming an Ad-hoc network. The aforementioned issues with flying networks can affect both modes of operation. [11]

#### 2.3.2.2 Server vs Client

In MANETs, nodes are clients for most of the time and may act as forwarding nodes for other clients as well. This is the case in military applications when providing support communication and coordination needs between soldiers, military vehicles and information headquarters [15]. In vehicular networks, nodes are usually clients needing to communicate to road-side fixed infrastructure (V2I – Vehicle to Infrastructure), sometimes participating in vehicle coordination platforms as well as the routing of other communications [15]. In UAV networks, however, the trend is for nodes to act as servers. More specifically, UAVs usually route packets for clients or act as relays for data. [11]

#### 2.3.2.3 Star vs Mesh

In a Star configuration, nodes must communicate through a central point; hence, a higher latency is expected, since typically the link length is longer than inter-UAV distance. If the central point fails, all the communications are interrupted.

Compared to Star networks, Mesh networks are flexible, reliable and offer better performance characteristics. Provided that two nodes are in radio range, they can exchange data without the need for a third party.

Table 2.1 [11] presents a comparison of relevant aspects between Star and Mesh networks.

### 2.3.3 Topology Control

In [2], the authors present a novel concept: Traffic-aware Multi-tier Flying Network (TMFN). The main purpose of the TMFN—consisting of a mobile and physically reconfigurable network of Flying Mesh Access Points (FMAPs) and Gateway UAVs—is to dynamically reconfigure the flying network's topology according to the users' traffic demands, which are characterised by their positions and offered traffic. For that, a novel algorithm is proposed: NetPlan. The NetPlan

Table 2.1: Comparison of Star and Mesh network properties.

| **Star Network** | **Mesh Network** |
|---|---|
| Point-to-point | Multi-point to multi-point |
| Central control point present | Infrastructure-based may have a control centre; Ad-hoc does not have one |
| Infrastructure-based | Infrastructure-based or Ad-hoc |
| Not self-configuring | Self-configuring |
| Single hop from node to central point | Multi-hop communication |
| Devices cannot move freely | Ad-hoc devices are autonomous and free to move. Infrastructure-based movement is restricted around the control centre |
| Links between nodes and central points are configured | Inter node links are intermittent |
| Nodes communicate through central controller | Nodes relay traffic for other nodes |
| Scalable | Not scalable |

algorithm dynamically determines the FMAPs' coordinates and Wi-Fi cell ranges. The authors propose a two-tier architecture:

- Access Network – the first tier is a group of UAVs carrying APs, called FMAPs, creating Wi-Fi small cells;
- Backhaul Network – the second tier corresponds to the Gateway UAVs, responsible for forwarding traffic to the Internet using dedicated broadband wireless links.

This concept does not consider the routing challenges associated with the provisioning of always-on broadband Internet connectivity, such as the ability to select high-capacity and uninterruptible paths towards the Internet when the network topology is being reconfigured, nor does it take advantage of knowing in advance the future network topologies in order to prevent link failures. It is, then, important to address these challenges in order to provide users with the best possible QoS [3].

## 2.4 Software-Defined Networking

Software-Defined Networking (SDN) is a centralised approach to network management. The key idea is to separate the control and the data forwarding planes of the network, moving the responsibility of the former to a software-defined controller, while also following a flow-based paradigm that enables highly scalable mobile and Wi-Fi networks [16]. This makes it possible to maintain a global view of the network, thus allowing intelligent decision-making regarding UAV trajectory control, data forwarding paths, packet transmission parameters (data rate or transmission power), and others, which may require computational power not available onboard the UAVs [14, 17].

These decisions are made from the massive information that is collected throughout the UAV network, regarding itself or ground nodes, such as flight control, locations, and link quality [13]. SDN also allows handling frequent topology changes, unreliable wireless links, and flexible switching and routing strategies [13]. The SDN paradigm facilitates the management of various dissimilar protocols, an issue introduced in 2.3.1, as well as tuning network policy and performance absent change to the data plane infrastructure [11, 14].

Through the use of virtualisation mechanisms on the controller, different functions and applications can be deployed on the network without the need to add or modify UAV hardware [14].

Figure 2.4 [18] depicts a conceptual SDN architecture. Mapping it onto a centralised routing solution for a flying network: the CS contains the *network controller*, the sole element of the Control layer; the APs (on board the UAVs) controlled through its Southbound Application Programming Interface (API) are the *programmable switches*, located at the Infrastructure layer; and applications which take decisions regarding the network pass them on through the network controller's Northbound API, causing it to act upon the APs.

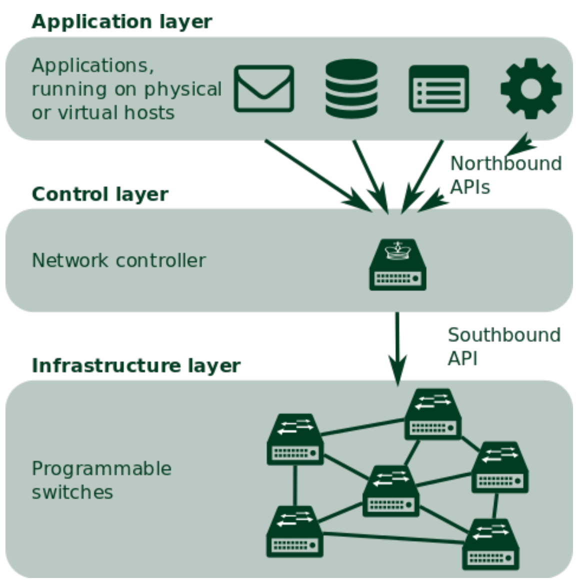


Figure 2.4: SDN architecture concept and the interactions between its layers.

Detailed information and definitions regarding SDN can be found in RFC 7426 [19].

Table 2.2 presents a few relevant UAV network features and how SDN proposes to address them.

In [13], the authors design a scalable SDN framework for UAV networks. They show that a monitoring platform in the SDN controller, together with a load balancing algorithm, can take full advantage of the network statistics to effectively balance the traffic.

In [14], the authors corroborate the fact that SDN can facilitate the utilisation of multiple wireless link access technologies (specifically, LTE, IEEE 802.11ad and IEEE 802.11ac). Furthermore, it is shown that SDN can effectively use the information available for intelligent decision making, improving the resiliency of the network by means of a multi-path routing protocol, thus

Table 2.2: UAV communication features and SDN capabilities.

| UAV network features | SDN capability |
|---|---|
| Coordination and control | Reconfiguration through an orchestration mechanism |
| Flexible switching and routing strategies | Flexible definition of rules based on header or payload for routing data |
| Frequent topology changes | |
| Unreliable links | Can be addressed through path/channel selection |
| Intentional disruptions | |
| Limited onboard resources | Supports switching off devices when not in use, data aggregation in the network and offloading operations to the controller |

reducing end-to-end outage. However, end-to-end delays turned out slightly worse in comparison to traditional routing protocols.

### 2.4.1 Network Functions Virtualisation

A concept that integrates well with the SDN paradigm consists in the virtualisation of network functions, i.e., creating Virtualised Network Functions (VNFs). It enables a general hardware appliance to provide more complex network services by using software, meaning new network functions can be deployed on the fly, such as creating an AP, a DHCP server, or even a Domain Name Service (DNS), on an existing UAV (adhering to the purpose of this dissertation) [20].

The authors in [20] show that the automated deployment of network services as lightweight VNFs is feasible using the limited resources available in small UAVs.

## 2.5 Routing

Flying networks are much more dynamic and prone to topology changes than mobile or vehicular networks, as mentioned before. The fact that nodes may move rapidly from one location to another makes radio links unreliable, i.e., they are intermittent. For these reasons, existing routing protocols in mobile and vehicular networks cannot be directly applied to those composed of UAVs [21]. Table 2.3 [22] contains some factors to consider for the routing protocol.

Figure 2.5 [21] shows an extensive list of existing routing protocols for UAVs divided into categories.

### 2.5.1 Single-hop routing

In Single-hop routing, UAVs fly from source to destination carrying data, much like a courier. Static routing is used and routing tables do not need to be updated. It is a lightweight approach to routing protocols, mainly used in fixed topology scenarios.

Table 2.3: Informations for routing.

| Layer | Information |
|---|---|
| Application Layer | QoS, Reliability, Expandability |
| Network Layer | Hop Count, Fault Tolerance, Robustness |
| Data Link Layer | Time Delay, Throughput |
| Physical Layer | Interference, Capacity, Channels |

The main problems include poor fault-tolerance and the fact that they are not suitable for dynamic environments [21].

### 2.5.2 Multi-hop routing

In Multi-hop routing, packets go from source to destination hop by hop. As such, nodes must choose a node to be the next hop to forward packets to. The way the next hop node is selected is the core of routing discovery. Multi-hop routing is further divided into Topology-based and Position-based protocols.

#### 2.5.2.1 Topology-based

Proactive routing protocols, a subcategory of Topology-based protocols, rely on the periodic exchange of control messages to keep an updated view of the network. Whenever a node has to forward a packet, it can do so without waiting, since its forwarding table is most likely not empty.

**Proactive** routing protocols introduce a significant overhead on the network, occupying bandwidth to keep the routing tables up to date, and they react slowly to topology changes, which causes delays [11].

**Reactive** routing protocols, also called passive or on-demand routing, don't exchange periodic messages in order to fill their routing tables. Instead of this, they calculate the best path when there is a packet to send, meaning there's possibility of delay during the route finding process, which may not be negligible [11].

Table 2.4 [21] depicts a comparison between proactive and reactive routing.

In addition to proactive and reactive protocols, a **Hybrid** routing protocol may be used. The purpose of such a protocol is to reduce the amount of control message overhead in the network introduced by proactive protocols while also tackling the undesirably longer end-to-end delays introduced by reactive protocols. Hybrid routing divides the network into zones where intra-zone routing uses the proactive approach, and inter-zone routing uses a reactive approach [11].

#### 2.5.2.2 Position-based

Due to the high mobility and frequent topology changes in UAV networks, proactive routing is not always effective, since it relies on stored routing tables. Conversely, in reactive routing the

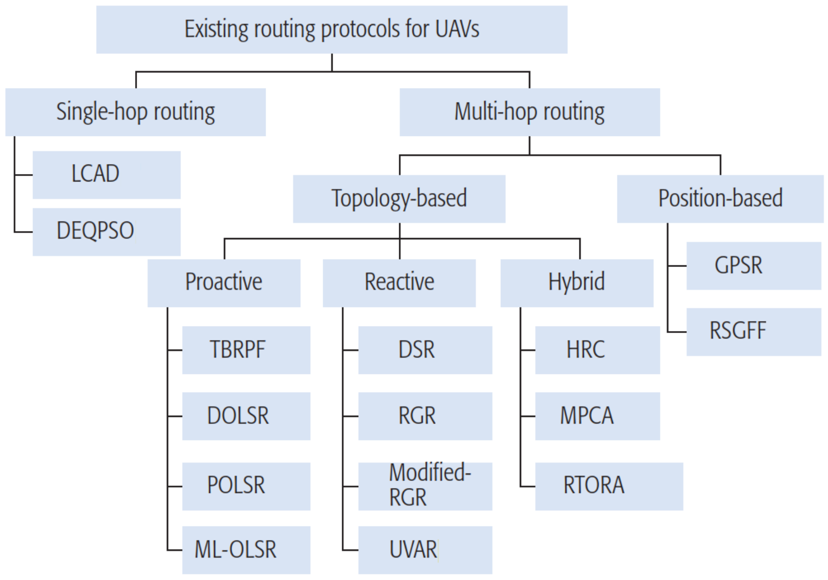


Figure 2.5: Classification of existing routing protocols.

best path is calculated every time the source node needs to send a message, which implies longer transmission delays. Position-based protocols are an alternative approach that aims at addressing these constraints.

Position-based routing protocols take advantage of the geographic location information to calculate the routing tables. An example on how this can be done is to forward the packet to the neighbour node moving faster towards the destination (which is one of the strategies of the Recovery Strategy Greedy Forwarding Failure (RSGFF) protocol) [21].

This sort of protocol is, in practice, less desirable to implement because it introduces a need for additional hardware. Specifically, routing performance depends heavily on GPS accuracy, and if GPS is not available or otherwise inaccurate, it is not possible to determine the best path.

## Summary

Table 2.5 [21] contains a comparison of the routing protocols laid out in Figure 2.5. It is important to note that all existing routing protocols for flying networks (even ones which adopt a hybrid paradigm) do not take into account the future state of the network, simply because it should not be possible to do so without the assumption that at least one of the two following conditions are met:

1. the UAVs' future positions can be determined in a deterministic manner;

2. there is a CS orchestrating the geographical positions of the UAVs and it has decision-making capabilities over routing in the network.

Table 2.4: Comparison of proactive and reactive routing.

| Parameters | Proactive routing | Reactive routing |
|---|---|---|
| Availability of routing information | Always available regardless of need | Only available when needed |
| Periodic route updates | Required | Not required |
| Coping with mobility | Inform other nodes to update routing tables | Use localised routes or rebuild routes when needed |
| Traffic load of control packets | Rapidly grows with increasing node mobility | Less than in proactive routing |
| Packet transmission delay | Packets can be transmitted immediately | Longer than in proactive routing |
| The protocol overhead | Higher than in reactive routing | Lower, since only a part of topology information is needed |

### 2.5.3 Open issues

The following are considerations taken from the literature regarding the design of a routing protocol for flying networks. They arise from the critical analysis over existing solutions and current research efforts being made. Open research issues that can be tackled by this dissertation are included.

#### 2.5.3.1 Cross-layer architecture

In literature, the general consensus is that UAV networks require a different approach when it comes to efficient routing [11], since protocols designed to support mobile and vehicular networks aren't properly prepared to deal with the intermittent nature of wireless links connecting highly dynamical nodes—including UAVs.

To meet the QoS requirements of ground users connected to the flying network, a cross-layer design becomes imperative [38] in order for the routing protocol to be aware of physical layer parameters, allowing it to estimate the quality of the radio links, thus making a decision on the best and most reliable path [21]. Such an architecture will pave way to fully utilise system resources and take advantage of the networking capability of a flying network [22].

#### 2.5.3.2 Load Balancing

Load balancing techniques, to avoid congestion and maintain the QoS levels of a network, are an important open issue. In order to do so, an intelligent management of the resources on board the UAVs, in order to fully utilise them and provide the best experience for as many users as possible, is worthy to be considered [13].

Table 2.5: Comparison of existing routing protocols.

| **Routing protocol** | **Type** | **Periodic broadcast** | **Multi-path** | **Load balancing** | **Loop-free** | **Route update** | **Dynamic adaptive** | **Energy-efficient** | **Routing metric** |
|---|---|---|---|---|---|---|---|---|---|
| LCAD [23] | Single-hop | No | Yes | No | Yes | No update | No | No | No metric |
| DEQPSO [24] | Single-hop | No | No | No | Yes | No update | No | No | No metric |
| TBRPF [25] | Single-hop | Yes | No | No | Yes | Periodically | No | No | Minimum-cost path |
| DOLSR [26] | Proactive | Yes | Yes | No | Yes | Periodically | No | No | Optimised link |
| POLSR [27] | Proactive | Yes | Yes | No | Yes | Periodically | No | No | Optimised link |
| ML-OLSR [28] | Proactive | Yes | Yes | Yes | Yes | Periodically | No | No | Optimised link |
| DSR [29] | Reactive | No | Yes | No | Yes | As needed | Yes | No | Shortest path |
| RGR [30] | Reactive | No | No | No | Yes | As needed | Yes | No | Shortest path |
| Modified-RGR [31] | Reactive | No | No | No | Yes | As needed | Yes | No | Shortest path |
| UVAR [32] | Reactive | No | No | No | Yes | As needed | Yes | No | Shortest path |
| HRC [33] | Hybrid | Yes | No | No | Yes | Hybrid | Yes | No | No metric |
| MPCA [34] | Hybrid | Yes | No | No | Yes | Hybrid | Yes | No | Freshest path |
| RTORA [35] | Hybrid | Yes | Yes | No | Yes | Hybrid | Yes | No | Optimised link |
| GPSR [36] | Position-based | No | No | No | Yes | As needed | Yes | No | Shortest path |
| RSGFF [37] | Position-based | No | No | No | Yes | As needed | Yes | No | Shortest path |

#### 2.5.3.3 Centralisation

In addition to the flexibility and control capabilities provided by a centralised controller in the network, the authors in [3] show that it is possible to shape a routing protocol considering the holistic knowledge that a Control Station (CS) has over the network. Contrary to traditional routing, the proposed solution, *RedeFINE*, does not use control packets for neighbour discovery and link sensing, meaning it wastes a significantly smaller amount of bandwidth for control traffic, which is composed only of the routing tables sent from the CS to the UAVs. Instead, it is up to the CS to calculate and disseminate the forwarding tables taking into consideration the future positions of the UAVs—which it is responsible for calculating—, meaning the network predicts link failures *before* they happen. Such an approach is worth considering, as guaranteeing end-to-end uninterrupted paths is still an open issue in flying networks, especially during periods in which it suffers reconfigurations, for instance when updating the topology.

RedeFINE [3], however, does not consider important aspects such as:

- the impact of access network reconfigurations on mobile terminals (e.g., Wi-Fi Roaming);
- a load-balancing scheme, allowing scalability and an even distribution of traffic [39];
- gateway diversity, as a congested network suffers from an inevitable bottleneck when considering a single gateway to the wired network.

Nevertheless, the concept of finding the best paths using the deterministic knowledge of how the topology is and will be has been proven to work [3] and should be studied; finding the optimal paths in UAV networks still proves to be an open issue [40].

## 2.6 Mobility Management

One of the main concerns of having a highly dynamic (and flying) network rests within an inherently higher handover frequency. The concept of *Handover*—also referred to as *Handoff* in American English, or even *Wi-Fi Roaming* in the context of IEEE 802.11 networks—is an attempt to provide seamless communications (i.e., with no user perceivable interruptions) to a Mobile Node (MN) even when it changes its current point of attachment to the Internet [41]. Its main purpose is to mask changes that happen in the lower levels of the Internet protocol suite, ensuring that the QoS at the Application Layer suffers the least degradation in the presence of possible interruptions, while keeping on the lookout for better access alternatives.

Certain types of handover should be distinguished [41]:

- Vertical Handover – handover between points of attachment supporting different Data Link Layer network technologies (e.g., Wi-Fi to Cellular; see Figure 2.6);
- Horizontal Handover – handover between points of attachment supporting the same Data Link Layer network technology (e.g., moving from one AP to another across the same IEEE 802.11 network; see Figure 2.6);
- Hard Handover – a horizontal handover using a *break before make* approach, i.e., it must break off the connection to the old AP before switching to a new one (at any one time, the MN keeps a single connection);
- Soft Handover – a horizontal handover using a *make before break* approach, meaning the connection to the old AP is retained until the MN successfully establishes a connection to the new AP.

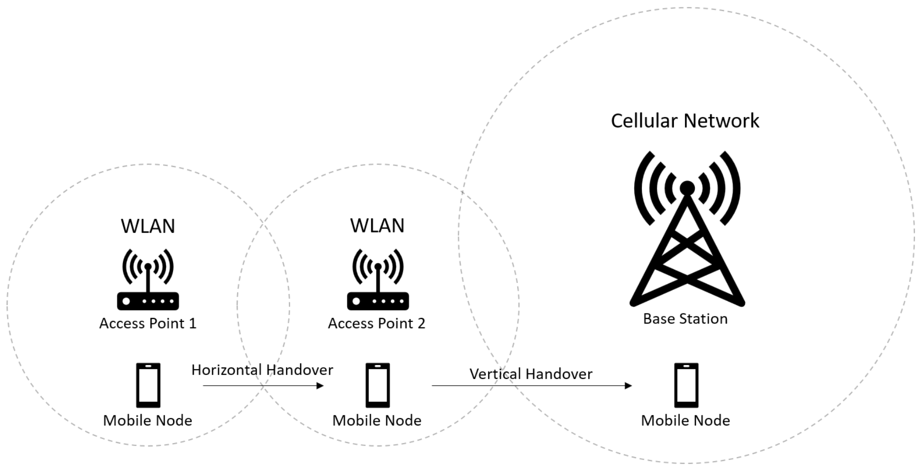


Figure 2.6: Horizontal and vertical handovers.

In IEEE 802.11, handovers[3] typically follow a three-phased process: scanning (for new APs), authentication, and re-association.

[3]From this point onward, the term *handover* assumes an horizontal handover unless explicitly stated otherwise.

### 2.6.1 Channel Scanning

In IEEE 802.11 networks, STAs are in charge of the handover process. This means that different vendors may implement different algorithms to decide when to trigger the handover. Since STAs do not have a global view of the network, this implies a discovery process, i.e., they have to scan through the available channels, sequentially, searching for a candidate AP. While scanning, a STA sends out a *Probe Request* management frame, then waits for a *Probe Response*. After a waiting period, two scenarios can happen [42]:

- there is no response, so the STA assumes there are no APs operating on that channel and moves on to the next one, sending a new Probe Request;
- there is a response and, after waiting to see if more APs respond, the authentication process may start.

Figure 2.7 [43] depicts a typical IEEE 802.11 handover process. The larger the number of channels a STA has to scan, the lengthier the scanning delay, increasing the time it takes to switch from one AP to another (authentication and re-association delays are relatively short compared to the scanning delay). In fact, depending on the waiting periods in each channel, a client may fail to find a candidate AP [43]. This is aggravated in the case of a passive scan, in which a STA listens on channels without sending a Probe Request.

Having to scan through different channels in the hope of finding a better AP is a symptom of a bigger problem: STAs have limited knowledge of the network. A centralised approach may help in solving this problem, since it enables a central entity with access to privileged information on the network and can decide which AP is suitable for every client.

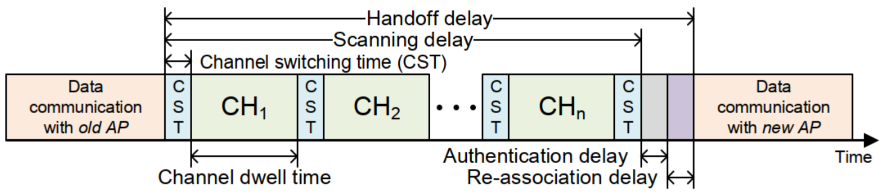


Figure 2.7: Timing diagram of the IEEE 802.11 handover process.

### 2.6.2 Virtual Access Points

Some different approaches to the issue of implementing a seamless handover scheme in IEEE 802.11 networks can be found in literature, such as focusing on solving the channel scanning mechanism's flaws [43] or adding new standards for current Wi-Fi networks [44]. The first requires a different interface for APs which they use to broadcast beacons on the same channels as their neighbour APs, while the latter requires compliance from wireless devices. Two other main approaches are often used, both of which are built upon the idea of virtualisation:

- The first consists in assigning a personal Virtual Access Point (VAP) to a user the first time it associates to an AP (e.g., a Light Virtual Access Point (LVAP) [45]). This VAP is then moved across physical APs during the process of handover.

- The second masks all physical APs in the network as a single AP. An example of this is *BIGAP* [46].

In [47], where a personal VAP based solution is introduced, an extensive list of frameworks addressing the aforementioned issue can be found. Table 2.6 contains the key differentiating aspects between them.

Table 2.6: Characteristics of frameworks providing fast handovers and/or virtualisation in IEEE 802.11 WLANs.

| | **Standard** | **Cloud MAC** [48] | **Anyfi** [49] | **KHAT** [50] | **Em-POWER** [51] | **Nakauchi** [52] | **M-SDN** [53] | **MP-SDWN** [54] | **Odin** [55] | **Wi-5** [47] |
|---|---|---|---|---|---|---|---|---|---|---|
| Multichannel | yes | yes | not reported | yes | not reported | yes | yes | no | no | yes |
| Multi RAT | - | no | no | no | yes | no | no | no | no | no |
| Requires changes in the STA | no | no | no | yes | no | no | no | no | no | no |
| Who triggers the handover | STA | not reported | controller | STA | controller | controller | controller | controller | controller | controller |
| Mobility Management | reactive | - | reactive | proactive | reactive (proactive also possible) | tested manually | reactive | reactive | reactive | reactive and proactive |
| Parameters used for mobility management | RSSI | - | not reported | RSSI | RSSI, loss rate (others are possible) | AP load (others are possible) | RSSI | RSSI (others are possible) | RSSI, rate, noise, number of packets | RSSI, AP load (others are possible) |
| Virtualisation | no | virtual WLAN cards | Service-specific VAP | no | LVAPs | Service-specific BS | no | MVAPs | LVAPs | LVAPs |
| Handover delay | 1-2 s | not reported | not reported | not reported | not reported | 65 ms | 1.5 s | not reported | not reported | 50-75 ms |

## 2.7 Summary and Main Conclusions

From the content exposed throughout the previous sections of this chapter, it is possible to withdraw key conclusions to guide this dissertation:

- Wi-Fi networks are evolving to meet the demand that users create regarding a ubiquitous network access that provides high-speed and low-latency connectivity.

- Flying networks composed of UAVs carrying APs are gaining traction in supporting temporary events that require an on-the-fly communications solution. It is a hot topic in current research.

- SDN provides a privileged insight into a network and allows flexibility and an intelligent control and management of a network. Currently, it is also explored in the context of wireless networks, giving rise to problems not considered before, including interference, mobility, and channel management.
- A cross-layer routing approach is commonly suggested as a key open issue in flying networks so as to guarantee the QoS of more stringent applications, such as real time, voice, and video. This translates into a network that is aware of its physical properties, namely quality of links between nodes.
- To solve the problems mentioned throughout this chapter, an interesting approach is to manage the network from a centralised perspective, wherein a controller can orchestrate its topology and predict upon its future state to prevent its performance degradation and loss of QoS.

# Chapter 3

# Proposed Solution

The previous chapter was focused on exposing the fundamental knowledge and the state of the art solutions that were considered to solve the problem addressed by this dissertation. In this chapter, a solution is formulated and laid out, including the system model, its architectural design, and its concepts.

## 3.1 Problem Statement

When deploying a flying network composed of UAVs that carry APs able to provide Internet connectivity to a set of users on the ground, it is important to place the UAVs in a way that the users' traffic demands are met. To achieve that, a centralised orchestration of the network topology, including the UAVs positions, is worthy to be considered. In addition, due to the highly dynamic behaviour of flying networks, which induce frequent disruptions on radio links, the selection of the paths taken by the packets in the network is a major challenge. Therefore, a routing approach that takes advantage of a holistic knowledge of the network, including the future geographical positions of the UAVs, which were defined to fulfil the traffic demands of the users on the ground, to calculate in advance the best paths amongst UAVs, and update the forwarding tables before links are disrupted, represents a step forward to the state of the art. Moreover, the access links from ground users to the APs on board the UAVs must also be taken into account, as it is a point of failure for network connectivity. Regarding this, it is important to address three additional aspects:

1. Seamless handovers, so that connectivity is not interrupted whenever a client leaves the range of its current AP onto the domain of its neighbour;

2. Repositioning UAVs without disrupting active connections, considering their future positions and client associations;

3. Load balancing on APs, so that the maximum number of clients can be fairly served.

## 3.2 System Model

The proposed solution is composed of four elements: a CS, UAVs, IEEE 802.11 APs, and IEEE 802.11 STAs.

Figure 3.1 depicts the elements and the way they interact. It is important to note that the network of UAVs may grow as needed, meaning that certain UAVs may even act purely as relay nodes in the path between the APs and the gateway(s).

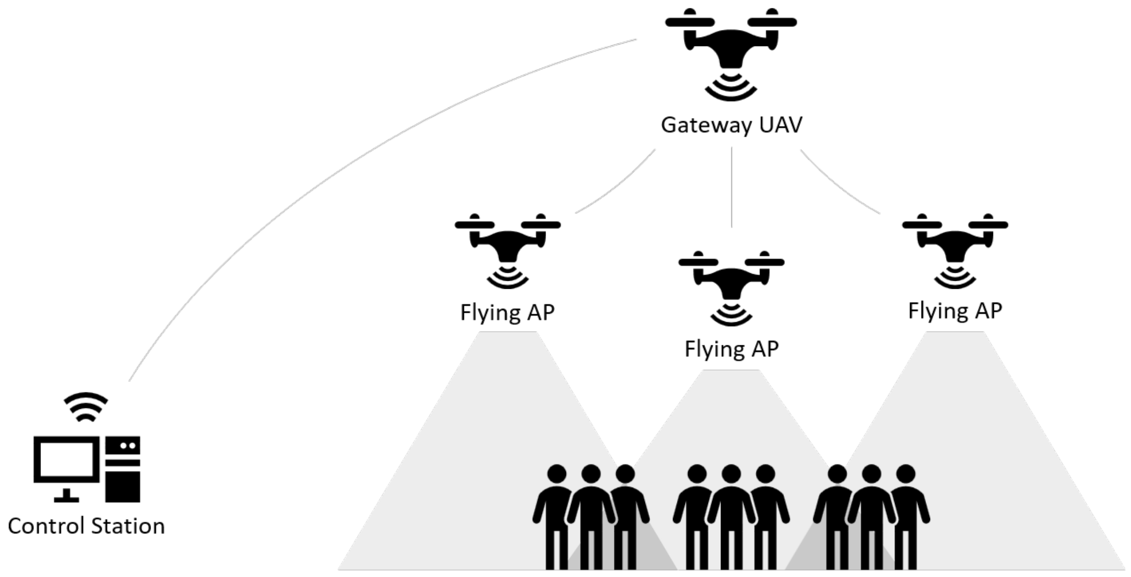


Figure 3.1: System elements and their interactions.

### 3.2.1 Control Station

The CS is the fundamental node in this proposed solution, as it is responsible for intelligently controlling both the movement of the UAVs and the network configurations, including 1) the forwarding tables and the instants they shall be updated in the UAVs, 2) load-balancing and 3) the issues that arise from the network mobility, such as handovers.

The CS is running an SDN controller compatible with OpenFlow, which is a set of open-source SDN specifications maintained by the Open Networking Foundation, a member of the Linux Foundation [56]. This controller can be deployed locally or as a service in the cloud, and deployed anywhere on the Internet.

### 3.2.2 UAVs

The UAVs meet the specifications required to deploy a solution as the one presented in [2]. In particular, the UAVs are able to hover above the ground while carrying APs, forming Wi-Fi cells that grant terminals the ability to connect to the flying network. There must be at least one UAV whose purpose is to act as a gateway, connecting the flying network to the Internet.

To allow for better network orchestration by making use of a wider selection of radio frequencies, UAVs should contain an additional wireless interface for monitoring purposes. This way, they can detect and collect RSSI values from STAs associated with APs on neighbouring UAVs, even if they operate on different channels.

UAVs should also be compatible with OpenFlow, making them subject to decisions from a centralised SDN controller.

### 3.2.3 IEEE 802.11 Access Points

The IEEE 802.11 APs should, much like in [14], be able to run an OpenFlow compatible programmable switch: Open vSwitch (OvS) [57]. Other than that, it is noteworthy to mention that the implementation of an SDN solution enables APs with the possibility to be standard "agnostic".

### 3.2.4 IEEE 802.11 Stations

IEEE 802.11 STAs are the devices that connect to the Internet through any of the existing Flying APs. Their position may be static, dynamic, or both (e.g., a mobile phone in the hands of a person at a music festival).

## 3.3 Architecture of the Proposed Solution

Figure 3.2 depicts the architectural design of the proposed solution: a centralised controller following application commands on its northbound API and acting on network equipment through its southbound API.

The proposed solution includes Odin [58], an open-source framework that allows programmers to implement network services as applications. It is composed of the Odin Master—implemented as an application on top of an OpenFlow controller—, multiple agents and a set of Odin applications. This framework was later improved in the Wi-5 project [47, 59], which added multi-channel handovers, different scanning and handover mechanisms, and other functionalities.

The applications, the Odin Master and the OpenFlow Controller coexist in a single environment (i.e., the Java Virtual Machine (JVM)). Whenever an application reaches a point in which it needs to actuate on the network (e.g., if it needs to add or remove an LVAP), it uses the interface provided by the Odin Master to propagate its requests to the networking equipment. This can happen by using a) the OpenFlow protocol to communicate with the OpenFlow Switches, or b) a custom protocol implemented by Odin, to communicate with its agents [58].

## 3.4 Virtual Access Points

To allow for seamless handovers, users should not be aware of any changes that happen on the network connection. As such, and given the fact that APs operate on a stateful basis, the best way for a STA to move to another AP and not lose its current connection status is to have the AP move

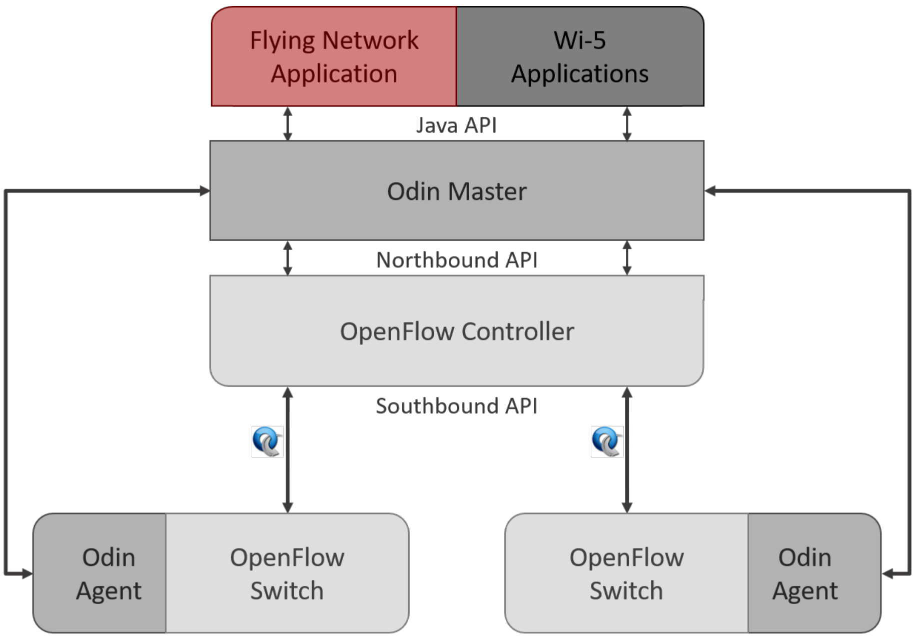


Figure 3.2: Architecture of the proposed solution. The red colour marks the application developed in this dissertation.

with it. Since that is not possible without physically moving the UAV that is carrying that AP, thus interfering with other users, virtualisation is needed. For each STA that connects to a Flying AP (i.e., a physical AP), a Light Virtual Access Point (LVAP) is created [55], giving clients the illusion of owning their own AP. Usually, this is initiated by a STA (i.e., in managed mode) sending out a *Probe Request* frame. Frames received on physical APs are handled according to the algorithm in Figure 3.3 [58]. A single physical AP may contain multiple LVAPs, each one represented by means of a tuple containing four elements:

- the STA's MAC address;
- the STA's IP address;
- a Basic Service Set IDentifier (BSSID) used by the AP to communicate with a single STA;
- a Service Set IDentifier (SSID) used by the AP to communicate with a single STA.

This way, the STA is agnostic to the physical AP it is connected to, regardless of its geographical position, provided that it is in the coverage area of at least one of the APs. This paves the way to both seamless mobility management and load balancing without interrupting connectivity.

## 3.5 Routing metric and Inter-UAV Routing

The routing metric used in this dissertation to define the cost of links in the network is the Euclidean distance. As such, the path cost is the sum of the Euclidean distances of its composing

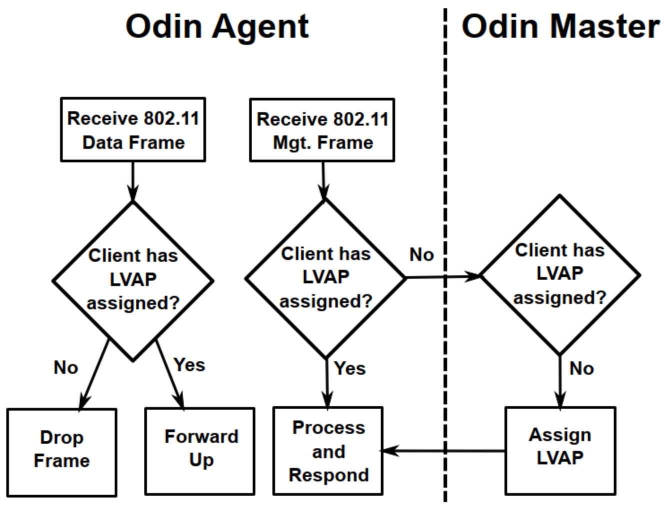

Odin Agent
Odin Master
Receive 802.11 Data Frame
Receive 802.11 Mgt. Frame
Client has LVAP assigned?
No
Yes
Drop Frame
Forward Up
Client has LVAP assigned?
No
Yes
Process and Respond
Client has LVAP assigned?
No
Assign LVAP

Figure 3.3: Processing path for IEEE 802.11 frames received on the Flying APs.

links. As the distance decreases, the capacity of the link increases, according to the Shannon-Hartley theorem. Using the Euclidean distance as path cost allows to ensure high-capacity links between the UAVs. To find the shortest path between the UAVs, the Dijkstra's algorithm is employed. This routing metric is fed by the positions of the UAVs provided by the CS. This follows the rationale of RedeFINE, which is presented in [3].

The mapping between STAs and their candidate APs relies on the monitoring capabilities of the network over STAs, which will be presented in the next section.

## 3.6 Mobility Management

In order to address the dynamic topology of the flying network and make it seamless to the user terminals, two distinct sequential time periods are considered:

- **Stationary** – the time during which the network is not being reconfigured, i.e., users may move from one AP to another, but APs are hovering in the sky, approximately in the same geographical position.

- **Dynamic** – the time during which UAVs, and consequently the APs they carry, are moving to new geographical positions, which were defined by the CS to fulfil the traffic demands of the users. During this period, changes in the quality of the wireless links will occur in short amounts of time.

### 3.6.1 Monitoring

For monitoring purposes, the following matrix was proposed in [47]:

$$\begin{pmatrix} wRSSI_{1,1} & wRSSI_{1,2} & wRSSI_{1,3} & \dots & wRSSI_{1,k} \\ wRSSI_{2,1} & wRSSI_{2,2} & wRSSI_{2,3} & \dots & wRSSI_{2,k} \\ wRSSI_{3,1} & wRSSI_{3,2} & wRSSI_{3,3} & \dots & wRSSI_{3,k} \\ \dots & \dots & \dots & \dots & \dots \\ wRSSI_{m,1} & wRSSI_{m,2} & wRSSI_{m,3} & \dots & wRSSI_{m,k} \end{pmatrix} \tag{3.1}$$

The matrix stores information regarding the signal quality between every STA $i$ and AP $j$. It is updated periodically using the values reported by APs upon receiving a request to scan for STAs from the controller, using Eq. (3.2).

$$wRSSI_{i,j}^{N+1} = \alpha \cdot RSSI_{i,j}^{N} + (1-\alpha) \cdot wRSSI_{i,j}^{N} \tag{3.2}$$

The *weighted* RSSI (*w*RSSI) of STA $i$ at AP $j$ calculated on iteration N is $w\text{RSSI}_{i,j}^{N+1}$, considering the average value measured in the last iteration, $\text{RSSI}_{i,j}^{N}$, the previous weighted RSSI, $w\text{RSSI}_{i,j}^{N}$, and a smoothing factor $0 < \alpha < 1$. The value of $\alpha$ defines how important the most recent observation is. In [47], which does not consider AP mobility, $\alpha$ was empirically determined to be better as 0.8 for various client moving speeds.

### 3.6.2 Stationary period

The stationary period is based on the *Proactive Handover* approach presented in [47], which is shown in Figure 3.4 [47]. The controller simply runs a *scan - gather results - make assignment decision* loop: it requests all its agents to scan the IEEE 802.11 channels (using their monitoring interfaces), gathers the resulting data, and updates the matrix of weighted RSSIs; then, according to the collected data, it decides which physical AP is best for each STA; finally, for each STA that was assigned a new physical AP, it removes its LVAP from the physical AP it currently resides in and moves it onto the new physical AP (if the two APs are in different channels, a Channel Switching Announcement (CSA) burst is triggered prior to the removal of the LVAP, prompting the STA to switch to its future AP's channel). This loop is repeated indefinitely.

The *Proactive Handover* approach provides various working modes (algorithms) for the decisioning part of the loop:

- *RSSI* – assigns STAs to the AP with the best RSSI;
- *FF* – assigns STAs to the AP with the highest Fittingness Factor (FF) and with an RSSI above a certain threshold (useful when there is a target throughput);

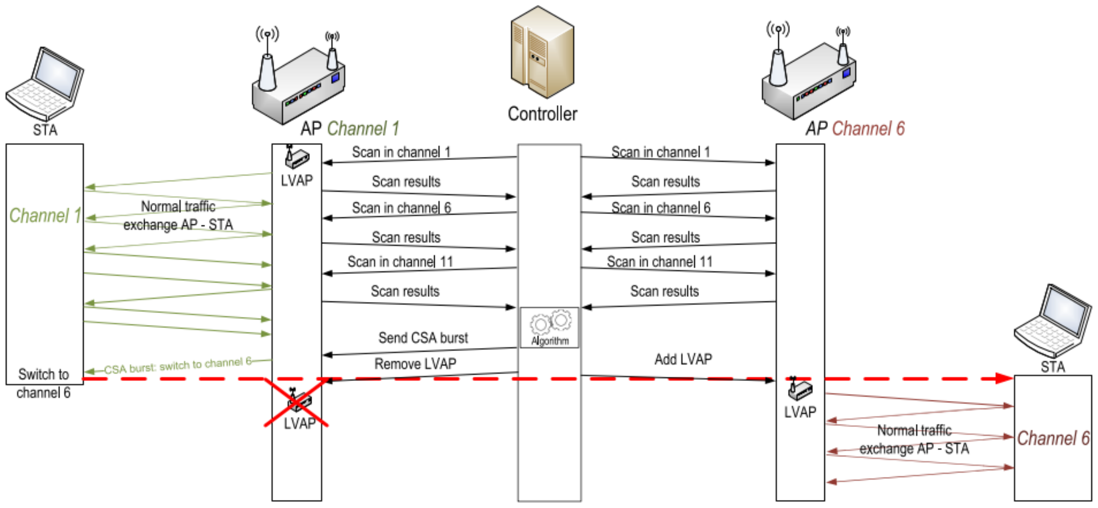


Figure 3.4: Scheme of the proactive handover. The CSA burst is required when physical APs are operating on different radio channels.

- *BALANCER* – performs load balancing and manages mobility at the same time (it assures RSSI values above a certain threshold and the number of STAs associated to each AP stays close to the average);
- *JAIN-BALANCER* – same as *BALANCER*, except it uses the Jain's Fairness index algorithm.

This solution aims at providing load-balancing, hence it requires one of the last two working modes (further detail in Section 4.1). It should be noted that, if the periodicity of the loop ran by the controller is high enough, this solution performs fast load balancing combined with seamless handovers, assuming either the *BALANCER* mode or the *JAIN-BALANCER* mode is in use. An important factor to take into account is the *hysteresis time*, which determines how frequently clients are allowed to be handed over from one AP to another. During stationary periods, this value should remain at 4 seconds, as empirical tests determined it to be a good option [47].

### 3.6.3 Dynamic period

In order to preserve the state of the clients' connections to the flying network during periods of high mobility (i.e., when a topology control algorithm requests the UAVs to move to new positions), the controller needs to make sure that each user's LVAP stays within its reach. It does so considering not only the origin and destination positions of all UAVs, but also the position of the STAs. However, the controller does not possess information regarding the positions of STAs, which means that they must be calculated whenever there is a shift in the flying network's topology due to a controller's decision.

When a network topology reconfiguration is set in motion, it is assumed that terminals on the ground stay in place. Thus, the *wRSSI* matrix (3.1) is kept in its current state throughout the dynamic period.

Looking at Figure 3.5, it is clear that the coverage areas of neighbouring APs may overlap. As such, using the centralised CS, this information can be leveraged to calculate a rough estimate for the positions of the terminals, based on (3.1) and the geographical positions of the UAVs. This follows the premise that UAVs are operating in an open area, with a strong Line-of-Sight (LoS) component.

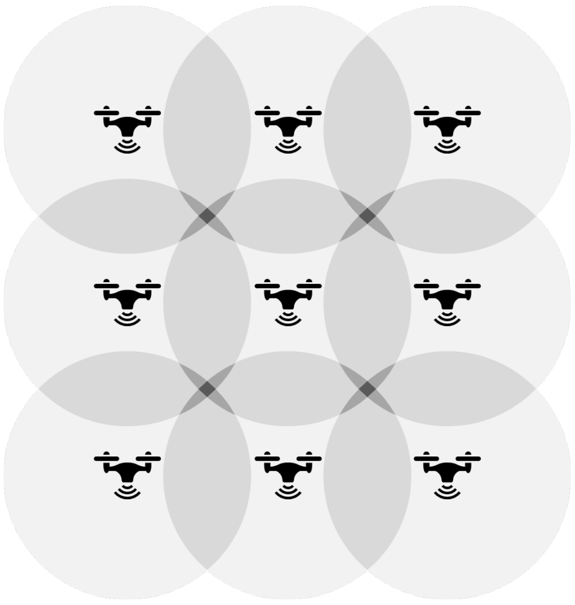


Figure 3.5: Simplified coverage representation of the access level in a flying network.

In (3.1), every STA corresponds to a row. If any STA is not in radio range of any AP, it leads to an undefined/invalid value (e.g., −99.9) on that STA's row in the matrix.

As such, the estimated STA's position can be obtained by calculating the weighted mean of the positions of valid APs, considering their stored *wRSSI* values. Equation (3.3) assumes a set of $k$ APs that have registered valid RSSI values in recent measurements.

$$x = \frac{\sum_{j=1}^{k} wRSSI_j \cdot x_j}{\sum_{j=1}^{k} wRSSI_j} \qquad y = \frac{\sum_{j=1}^{k} wRSSI_j \cdot y_j}{\sum_{j=1}^{k} wRSSI_j} \tag{3.3}$$

Afterwards, inter-UAV routing logic can be extended to consider terminals on the ground and the access network can take advantage of the benefits of using the predictive approach method introduced in RedeFINE [3]: each user's LVAP is swapped from its current physical AP to the best physical AP according to the future topology when the UAVs that carry the APs are equidistant to the user—since the Friis propagation loss model is assumed. Once the dynamic period is approaching its end, the *wRSSI* matrix is flushed and new values start being collected, to resume the normal functioning (i.e., stationary period).

### 3.6.4 Analysis of a reference case

In order to demonstrate the concept, the analysis of a reference case is presented herein. Let us consider a terminal whose RSSI can be measured in APs 5, 7, and 8, with a topology similar to the one depicted in Figure 3.5. The weighted RSSI values are: $wRSSI_5 = 20$, $wRSSI_7 = 50$, and $wRSSI_8 = 30$. As such, the row of the $wRSSI$ matrix that corresponds to this terminal is:

$$\begin{pmatrix} \infty & \infty & \infty & \infty & 20 & \infty & 50 & 30 & \infty \end{pmatrix}$$

The APs are located at $(x_5, y_5) = (20, 20)$, $(x_7, y_7) = (10, 30)$ and $(x_8, y_8) = (20, 30)$. Using Equation (3.3), it is possible to calculate both $x$ and $y$ coordinates for the terminal:

$$x = \frac{50 \cdot 10 + 30 \cdot 20 + 20 \cdot 20}{50 + 30 + 20} = 15$$

$$y = \frac{50 \cdot 30 + 30 \cdot 30 + 20 \cdot 20}{50 + 30 + 20} = 28$$

Thus, the terminal's position is determined from the $wRSSI$ obtained by measuring the RSSI at the three APs, as well as knowing their coordinates, as shown in Figure 3.6. The paths and the time instants in which the forwarding tables will be updated are calculated in advance, considering the UAVs' final positions.

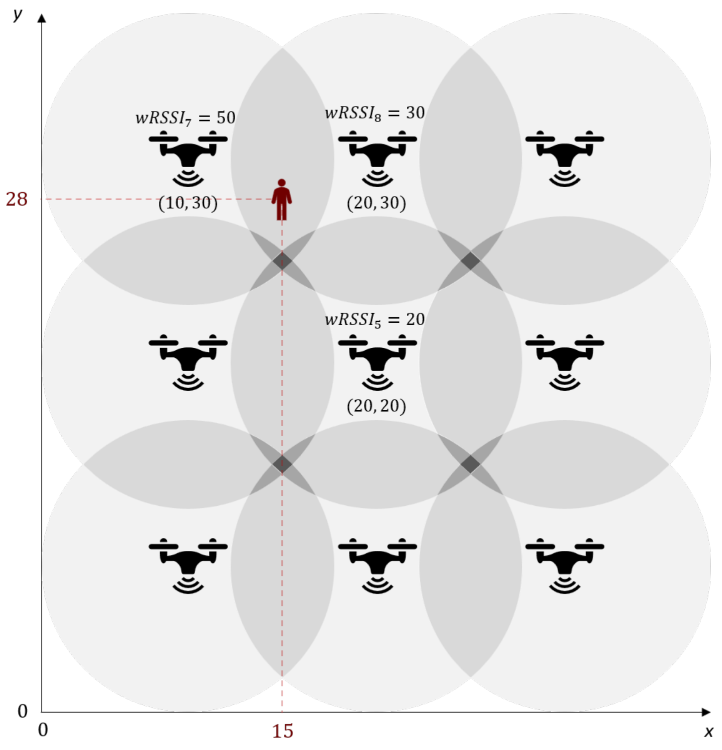


Figure 3.6: Terminal position calculated based on the measurements obtained in APs 5, 7, and 8.

## 3.7 Flying Network Application

During the network reconfiguration (dynamic period), a set of tasks need to be performed in order to determine and schedule in advance the handovers that will take place, thus adhering to the predictive nature that separates this solution from conventional routing. Algorithm 1 encompasses these tasks as a routine, demonstrating the purpose to the Flying Network Application, considered in the architecture of the proposed solution.

**Algorithm 1** Handling Virtual Access Points during network mobility periods

```
 1: function HANDLEHANDOVERS($UAVs'\ informations$)
 2:     $longestFlight \leftarrow 0$
 3:     for each $agent$ in $agents$ do
 4:         $convertGpsToNed(origin, reference)$          ▷ GPS coordinates as in the WGS 84 [60]
 5:         $convertGpsToNed(destination, reference)$
 6:         $t \leftarrow getFlightDuration(time, destination)$
 7:         if $t > longestFlight$ then
 8:             $longestFlight \leftarrow t$
 9:         end if
10:     end for
11:     for each $client$ in $clients$ do
12:         $currentAgent \leftarrow getAgent(client)$
13:         $pos \leftarrow getCoordinatesFromHistoricalRssi(client)$    ▷ Described in Subsection 3.6.3
14:         $futureAgent \leftarrow getClosestAgent(pos)$                ▷ Considering line 5
15:         if $futureAgent \neq currentAgent$ then                      ▷ A handover is required
16:             $delay \leftarrow calculateTimeToHandover(pos, currentAgent, futureAgent)$
17:             $scheduleHandover(client, delay)$                        ▷ Also considers the start time
18:         end if
19:     end for
20:     return $longestFlight$
21: end function
```

This application gets invoked whenever the network needs to adjust the positions of the UAVs. It is responsible for scheduling the handovers caused by the mobility of the UAVs, and, in order to do so, it requires the following information from each UAV:

- IP address;
- Origin Global Positioning System (GPS) coordinates;
- Destination GPS coordinates;
- Reference GPS coordinates (for coordinate conversions);
- Velocity.

Aside from the occasional overhead imposed over the network, the information included in this message is similar to the one sent to the UAVs using the MAVLink[1] protocol to set a target position[2].

In order to determine the correct time to schedule a handover, the following steps are considered:

1. the AP to which a STA is currently associated to is closer than the AP it will be associated to in the future;
    - if not, this particular handover is simply ignored and the STA remains connected to the closest AP.
2. the AP to which a STA will be associated to in the future is closing in on its distance;
3. there will be a turning point, i.e., a time instant for which both APs (UAVs) are equidistant to a STA, for every handover.
4. the information regarding the UAVs' trajectories is available since it is the same that is used to move the UAVs.

Formulating the aforementioned statements, considering the current AP is placed at point $x$ with coordinates $(x_1, x_2, x_3)$, the future AP at point $y$ with coordinates $(y_1, y_2, y_3)$, and the STA at point $p$ with coordinates $(p_1, p_2, p_3)$, we get:

$$d(x,p) = d(y,p) \iff d^2(x,p) = d^2(y,p) \iff \sum_{i=1}^{3}(x_i - p_i)^2 = \sum_{i=1}^{3}(y_i - p_i)^2 \tag{3.4}$$

If we consider the kinematic equation for constant velocity motion, $x = x_0 + v_x \cdot t$, (i.e., disregard acceleration), we get the following:

$$\sum_{i=1}^{3}(x_{i,0} + v_{x,i} \cdot t - p_i)^2 = \sum_{i=1}^{3}(y_{i,0} + v_{y,i} \cdot t - p_i)^2 \tag{3.5}$$

Since $(a+b-c)^2 = a^2 + b^2 + c^2 + 2 \cdot (ab - ac - bc)$, then:

$$\begin{gathered}\sum_{i=1}^{3}\left(x_{i,0}^2 + v_{x,i}^2 + p_i^2 + 2 \cdot (x_{i,0} \cdot v_{x,i} \cdot t - x_{i,0} \cdot p_i - v_{x,i} \cdot p_i \cdot t)\right) \\ = \\ \sum_{i=1}^{3}\left(y_{i,0}^2 + v_{y,i}^2 + p_i^2 + 2 \cdot (y_{i,0} \cdot v_{y,i} \cdot t - y_{i,0} \cdot p_i - v_{y,i} \cdot p_i \cdot t)\right)\end{gathered} \tag{3.6}$$

[1] https://mavlink.io/en/
[2] *SET_POSITION_TARGET_GLOBAL_INT*

Considering the quadratic equation in its standard form, solving for time $t$, we get:

$$at^2 + bt + c = 0 \iff t = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a} \tag{3.7}$$

Finally, we have:

$$\begin{cases} a = \sum_{i=1}^{3} \left(v_{x,i}^2 - v_{y,i}^2\right) \\ b = 2\sum_{i=1}^{3} \left(v_{x,i}\left(x_{i,0} - p_i\right) - v_{y,i}\left(y_{i,0} - p_i\right)\right) \\ c = \sum_{i=1}^{3} \left(x_{i,0}^2 - y_{i,0}^2 - 2p_i\left(x_{i,0} - y_{i,0}\right)\right) \end{cases} \tag{3.8}$$

Thus, being able to identify the moment in which a future AP (UAV) starts getting closer to a given STA, and starts to be preferred over the AP (UAV) that is currently serving that STA, a handover should be triggered.

# Chapter 4

# Implementation

This chapter addresses the implementation of the proposed solution. Specifically, this chapter describes how the different elements of the system were implemented, and the way they interact with each other. To test the solution, there were two possible approaches: simulation or experimentation. Amongst these two, simulation allows for the possibility of running larger and more complex scenarios, and it is far easier to repeat them under the same conditions. However, simulation does not represent an accurate depiction of reality, especially when it comes to the constraints imposed by Flying Networks. Namely, it can be difficult to accurately represent the physical properties of the links that connect the UAVs and the surrounding environment, as they are very dynamic and may change drastically over time. Moreover, being able to configure a simulation setup and run different simulations successfully does not translate into a working solution, since an additional effort is required to deploy the solution in real hardware. This dissertation provides a working solution, facing head-on the intricacies and hardships of a real environment. To do so, Section 5.1 presents a testbed which aimed at ascertaining the performance of the proposed solution regarding the initial requirements and objectives exposed in Chapter 1.

## 4.1 Controller

The controller was implemented using the Floodlight OpenFlow controller [61], which runs on the Java platform. On top of it, the Odin Master runs as an application. It uses an independent control channel and the OpenFlow protocol to communicate with the agents running on the APs, in order to update their forwarding tables. This decouples the data plane from the control plane, with the latter depending on a centralised entity. The Odin master enables additional applications to run on top of it as a thread, which by design run asynchronously (e.g., an application for retrieving statistics about the physical medium and another to handle handovers and/or load balancing). The Flying Network Manager application developed in this dissertation, which enables predictive handovers, runs in a synchronised fashion alongside the Smart AP Selection application that was previously developed in the Wi-5 project [47], first addressed in 2.6.2, and expressed in further detail in 3.6.2. Odin did not have a functionality that allowed synchronising concurrent applications prior to this dissertation. Hence, it was implemented.

The Flying Network Manager listens to incoming JavaScript Object Notation (JSON) messages that represent an impending mobility of the network, i.e., the physical movement of UAVs. Therefore, it stays idle, and the Smart AP Selection is triggered. In this implementation, *JAIN-BALANCER* was the selected mode, managing both mobility and load balancing, using the Jain's Fairness index algorithm. It was chosen since it is scale independent (i.e., the dimension of the values does not affect the measure), continuous, applies to any number of users, it is bounded between 0 and 1, and it has an intuitive relationship with user perception [62]. Whenever a message arrives, after checking if it is a proper JSON message (i.e., to avoid interrupting other applications unnecessarily), the Flying Network Manager interrupts the Smart AP Selection application. Then, it computes the optimal handover times for all the necessary STAs and schedules the handovers to happen and the Smart AP Selection application to resume, which occurs when the flying network has finished its mobility, taking into account the last moving UAV.

## 4.2 UAVs

Three types of UAVs were considered for the implementation of this solution:

1. Gateway – as its name states, it acts as a gateway for the network.
2. Relay – it forwards the traffic between a flying AP and the gateway, when a direct communications link between both is not possible.
3. AP – it provides an access network for ground users to connect to the flying network.

### 4.2.1 Gateway

A single Gateway UAV was used in this implementation, though multiple gateways could be deployed, in theory, following the same principle of the load balancing mechanism: since the OpenFlow protocol works on a flow basis, which translates into the paths taken by the data packets that are travelling in the network, these flows may be orchestrated in such a way that the UAVs directly connected to the gateways are assigned their next-hop taking into account the global distribution of flows per gateway. This idea can benefit from the Jain's fairness index algorithm, which is already implemented on the controller for the purpose of the Smart AP Selection application.

The Gateway UAV was implemented using a low cost, small sized computer (e.g., a Raspberry Pi), with two wireless Network Interface Cards (NICs) connected as Universal Serial Bus (USB) dongles: one was used to communicate with the flying network, and the other aimed at connecting the flying network to the Internet. The NIC connecting the gateway to the flying network must be IEEE 802.11 compliant, while the other, depending on the available hardware, can vary—an alternative approach would be to use a 4G LTE USB NIC—, so long as it was capable of providing IP connectivity between the controller and the agents running on the APs. The NIC which served the flying network was configured to use the IEEE 802.11s WLAN Mesh Standard [63].

If hardware availability or energy constraints require to do so, it is possible to use a single wireless NIC. This would, however, effectively halve the bandwidth in the gateway, which is a critical bottleneck-prone hop in the network. It would also mandate that the connection from the Gateway UAV to the Internet complies with the IEEE 802.11 standard, which may be impractical, considering the usage of the spectrum in the flying network and that the adapter would be bound to a single channel.

### 4.2.2 Relay

The Relay nodes are an optional element in the flying network. These were not considered during this implementation, but may be added in future implementations. They aim at providing connectivity to AP UAVs that do not have a direct link to a gateway. The relay nodes would be configured to use the IEEE 802.11s WLAN Mesh Standard, and may use one or two wireless NICs for their purpose, following the same considerations mentioned in Subsection 4.2.1. Should two wireless NICs be used—as was the case for other elements in this implementation—, in order to connect devices using different channels, the two NICs may form a network bridge and share the same IP address.

### 4.2.3 AP

The AP UAVs carried, as their name states, the APs that grant users on the ground access to the flying network. They have three fundamental requirements: a wireless NIC configured in infrastructure mode and acting as AP (cf. Subsection 2.2.1), the ability to run an OpenFlow Switch, and a wireless NIC compatible with the ‘ath9k’ driver, which requires a modification to be able to handle the concept of an LVAP [58]. In this implementation, the Open vSwitch [57] was used. The AP UAVs were also running the Click Modular Router [64], including the custom elements that define the Odin Agent [58] that communicates with the Odin Master running on the controller.

Regarding the number of wireless NICs to be used, there were four possibilities, in no particular order:

1. A single wireless NIC – the cheapest solution, but also the one providing the worst performance. It handles the AP and the IEEE 802.11s connection to the rest of the network on a single adapter. This will lead to the whole network operating on a single channel and all the communications nodes sharing the same spectrum and available bandwidth.

2. Two wireless NICs – this option provides better performance from a channel orchestration viewpoint, as the wireless links may operate on different channels simultaneously, thus not competing for the same resources. However, this is not yet optimal as it limits an important aspect of this solution: since the monitoring interface will be virtually implemented on top of a physical NIC being used to exchange traffic, the monitored spectrum will be constrained to either one of the two channels that are in use by the interfaces.

3. One wireless NIC to handle connections and one for monitoring purposes – this solution is optimal when it comes to monitoring, which benefits the handover performance during the stationary period (Subsection 3.6.2), as it allows for a more complete matrix of weighted RSSIs. Still, it is not the best option regarding channel orchestration, since it halves the available bandwidth.

4. Three wireless NICs – this option provides the best performance. One wireless NIC is configured in infrastructure mode (acting as AP), providing connectivity for ground users, while another one connects the UAVs amongst themselves. A third NIC scans all available channels to fill the matrix of weighted RSSIs, since it is independent and thus not limited to the channel currently in use by the other NICs. This third wireless NIC may be a low-cost USB dongle.

## 4.3 Monitoring

As mentioned in Subsection 4.2.3, the monitoring capabilities are restricted by the available wireless NICs. In this implementation, we underwent the second option: two wireless NICs were used to handle connections. This translated into having poorer monitoring capabilities. However, due to the nature of the available hardware of the testbed (which is described in the next section), and following what may be a typical deployment scenario, no more than two wireless NICs are typically available per UAV. The monitoring interfaces were created through virtualisation and they shared the hardware used for the creation of the APs.

Another solution for a more complex flying network would be to have some dedicated probe nodes, eventually with higher receiver sensitivity, for measuring the RSSI of the frames received from the STAs. These nodes should not be a part of any data paths, otherwise they would fall under the same conundrum of the AP UAVs. For testing purposes, however, it is not straightforward to implement this solution and it would require some further modifications to the controller.

# Chapter 5

# Evaluation

This chapter presents the performance evaluation of the Flying Network Manager against two counterpart solutions: a standard configuration and a configuration running the standalone Wi-5 Smart AP Selection application. The testbed setup is described and the performed tests are explained. The results are then analysed and the solutions are compared in terms of their performance and capabilities.

## 5.1 Testbed

The hardware components that composed the testbed used to evaluate the proposed solution are presented in Table 5.1. It also includes information regarding the processing and wireless capabilities of the components, such as their host name, architecture, Operating System (OS), and wireless NICs.

Table 5.1: Information on the hardware components forming the testbed.

| Host Name | Role | Brand/Model | Architecture | Operating System | Kernel | Wireless Adapters | Wi-Fi Capabilities |
|---|---|---|---|---|---|---|---|
| controller | Controller | Asus K55VJ-SX015H | x86_64 | Fedora 29 | Linux 5.1.11 | Qualcomm Atheros AR9485 | IEEE 802.11b/g/n |
| gateway | Gateway UAV | Raspberry Pi Model B | armv61 | OpenWRT 18.06.2 | Linux 4.9.152 | Linksys WUSB600N v2 | 802.11a/b/g/n, 2.4/5 GHz |
| | | | | | | Panda PAU09 N600 | 802.11b/g/n/ac, 2.4/5 GHz |
| agent1 | AP UAV | ALIX 3d3 | i386_geode | OpenWRT 18.06.2 | Linux 4.14.95 | 2 x MikroTik RouterBOARD R52n-M | 802.11a/b/g/n, 2.4/5 GHz, 2x MMCX |
| agent2 | AP UAV | ALIX 3d3 | i386_geode | OpenWRT 18.06.2 | Linux 4.14.95 | 2 x MikroTik RouterBOARD R52n-M | 802.11a/b/g/n, 2.4/5 GHz, 2x MMCX |
| smartphone | STA | Samsung Galaxy S10e | aarch64 | Android 9.0 (Pie) | Linux 4.14.85 | Broadcom BCM4375 | 802.11 a/b/g/n/ac/ax, 2.4/5 GHz |

### 5.1.1 Controller

The responsibility of running the OpenFlow controller, including the Odin Master and the applications that run on top of it—the Flying Network Manager and the Smart AP Selection—relies on a laptop, shown in Figure 5.1. It is an ASUS K55VJ-SX015H running the Linux-based distribution Fedora 29 OS [65]. The controller is connected as a STA to the gateway, which is running an AP on its secondary wireless NIC, otherwise used for connecting to the Internet. This follows the concept that the controller can be anywhere, provided that it has IP connectivity to the agents running on board of the UAVs.

(a) Top view. (b) Frontal view.

Figure 5.1: The laptop used to run the controller.

### 5.1.2 Gateway

The gateway is a Raspberry Pi Model B running the OpenWRT 18.06.2 (r7676-cddd7b4c77) OS with two USB wireless NICs: a Linksys WUSB600N v2, which is configured in infrastructure mode and acts as an AP, providing the controller access to the flying network, and a Panda PAU09 N600, which is configured to use the IEEE 802.11s WLAN Mesh Standard. The latter allows to form a mesh network between the gateway and the UAVs, ensuring the communications between the controller and the agents.

The Linksys WUSB600N v2 has two internal antennas. It has been designed to operate with an antenna having a maximum gain of 0.5 dBi (TX0) and 1.4 dBi (TX1) at 2.4 GHz and 4 dBi at 5 GHz [66]. The Panda PAU09 N600 has two external 5 dBi antennas.

The gateway is running DHCP and firewall services. It is responsible for leasing IP addresses to nodes connecting to the network. In this testbed, both the controller and the STA (a smartphone) have a static lease, thus being assigned the same IP address whenever their lease expires (every 12h). These IP addresses are 172.16.1.100 and 172.16.2.10, respectively, and can be seen in Table 5.2.

The Gateway node is shown in Figure 5.2.

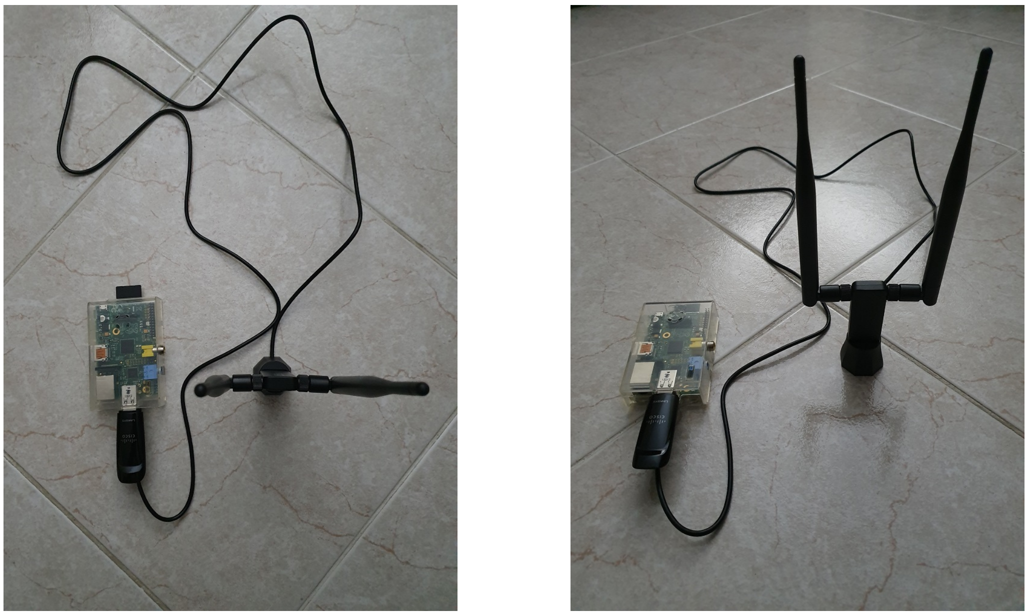

(a) Top view. (b) Frontal view.

Figure 5.2: The Raspberry Pi acting as the Gateway.

### 5.1.3 Agents

The agents are both running on an individual PC Engine ALIX 3d3 with OpenWRT 18.06.2 (r7676-cddd7b4c77) OS, and each of them has two MikroTik RouterBOARD R52n-M miniPCI wireless NICs, both with two omnidirectional antennas connected to them. The NICs responsible for providing network access to ground users have two 5 dBi antennas each, same as the NICs responsible for connecting the agents to the rest of the flying network, with the difference that the former are using 20 dB Radio Frequency (RF) attenuators [67], with the purpose of decreasing their radiated power, thus allowing smaller cells that do not overlap with the ones formed by the remaining UAVs. Attending to the Friis propagation loss model, for a transmission power of 15 dBm and an antenna gain of 3 dBi on the receiver, the received power decays with distance as shown in Figure 5.3, at 2.472 GHz (channel 13). The horizontal axis represents the LoS distance from Agent 1, and Agent 2 is placed 13 m away from it.

The agent nodes are shown in Figure 5.4.

### 5.1.4 Network configurations

In this testbed, all the communications nodes were configured in the same subnet (172.16.0.0/16). Typically, network interfaces are separated according to their function in the system. Specifically,

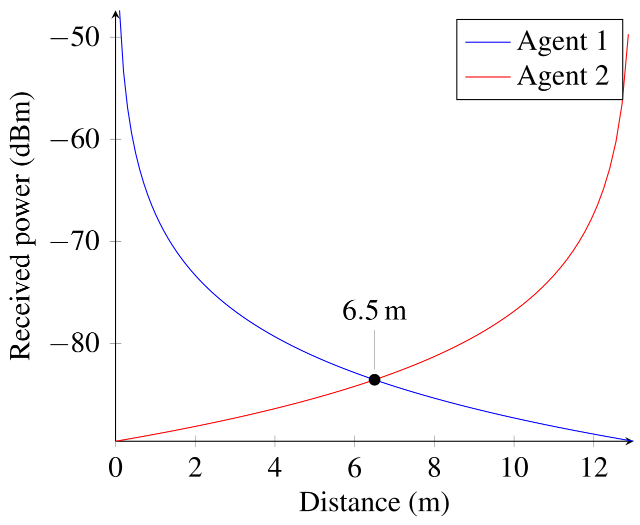
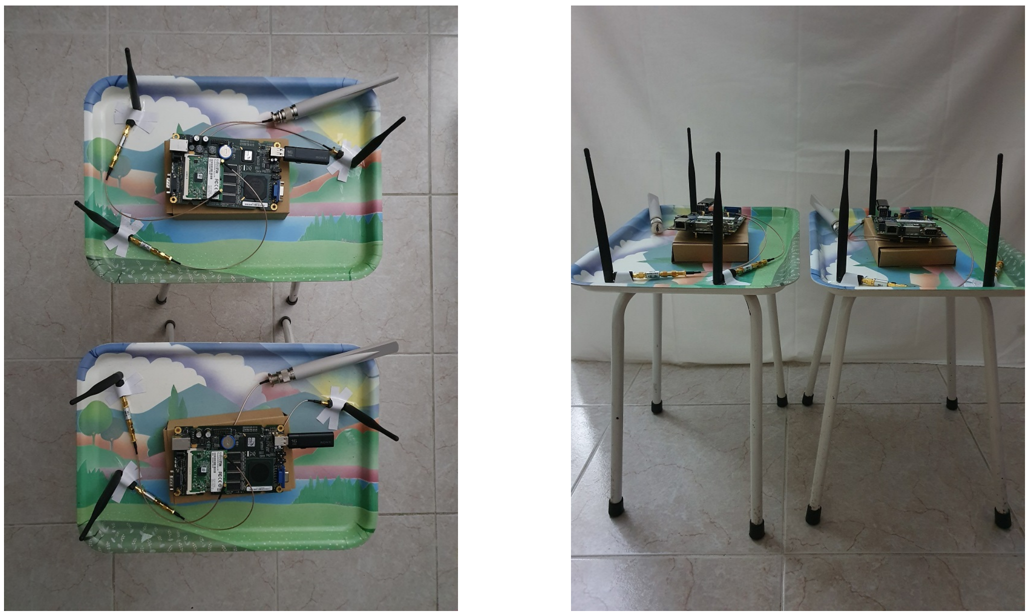


Figure 5.3: Theoretical power received from the agents according to the Friis propagation loss model. Agent 1 is located at 0 m and Agent 2 is at 13 m.



(a) Top view.

(b) Frontal view.

Figure 5.4: The ALIX 3d3 PC Engines acting as agents.

Table 5.2: The network configurations set up for the testbed entities.

| Entity | Wireless NIC | Mode | Channel | Bandwidth | Physical Address | IPv4 Address | Netmask | Default Gateway |
|---|---|---|---|---|---|---|---|---|
| Controller | Qualcomm Atheros AR9485 | 802.11n | 1 (2.412 GHz) | 40 MHz | DC:85:DE:73:3D:2D | 172.16.1.100 | 255.255.0.0 | - |
| Gateway | Panda PAU09 N600 | 802.11n | 1 (2.412 GHz) | 40 MHz | 9C:EF:D5:FD:CF:F4 | 172.16.0.254 | 255.255.0.0 | - |
| | Linksys WUSB600N v2 | 802.11s | 36 (5.180 GHz) | 20 MHz | 00:25:9C:B5:53:4F | | | |
| Agent 1 | MikroTik RouterBOARD R52n-M | 802.11s | 36 (5.180 GHz) | 20 MHz | D4:CA:6D:11:DC:49 | 172.16.0.1 | 255.255.0.0 | 172.16.0.254 |
| | MikroTik RouterBOARD R52n-M | 802.11n | 13 (2.472 GHz) | 20 MHz | D4:CA:6D:11:DC:3F | | | |
| Agent 2 | MikroTik RouterBOARD R52n-M | 802.11s | 36 (5.180 GHz) | 20 MHz | D4:CA:6D:11:DC:48 | 172.16.0.2 | 255.255.0.0 | 172.16.0.254 |
| | MikroTik RouterBOARD R52n-M | 802.11n | 13 (2.472 GHz) | 20 MHz | D4:CA:6D:11:DC:3E | | | |
| Smartphone | Broadcom BCM4375 | 802.11n | 13 (2.472 GHz) | 20 MHz | 6C:C7:EC:B2:4B:AB | 172.16.2.10 | 255.255.0.0 | 172.16.0.254 |

the network can be divided into smaller independent networks: one for the control plane, one for the data plane, another for management, and so on. This would, however, require Virtual Local Area Network (VLAN) trunking over Wi-Fi links, which cannot be done out-of-the-box and adds complexity and overhead (e.g., tunnelling) to the network [68, 69]. To allow for some organisation, static IP addresses are assigned in the 172.16.0.0/24 range, control plane addresses are dynamically assigned in the 172.16.1.0/24 range, and data plane addresses are dynamically assigned in the 172.16.2.0/24 range. It is also possible to statically assign multiple addresses to a single interface (i.e., the agents could have IP addresses across the three ranges), but this would cause Address Resolution Protocol (ARP) overhead with no performance improvement. Table 5.2 contains the network configurations used in this testbed, which were applied via Secure Shell (SSH) when first connecting to each device, using the Ethernet interfaces isolated from the rest of the traffic and for management purposes only. Figure 5.5 depicts the network setup.

### 5.1.5 Scenarios

Three evaluation scenarios were considered:

- Baseline – this scenario is based on a standard configuration where all APs broadcast the same SSID, relying on the clients to decide whenever to perform a handover. It has no balancing mechanism.
- Wi-5 – this scenario implements the proactive solution developed by the Wi-5 consortium [47]. Specifically, it uses the 'Smart AP Selection' with the selected mode *JAIN-BALANCER*. Hence, it is capable of proactively manage the clients' associations, handing

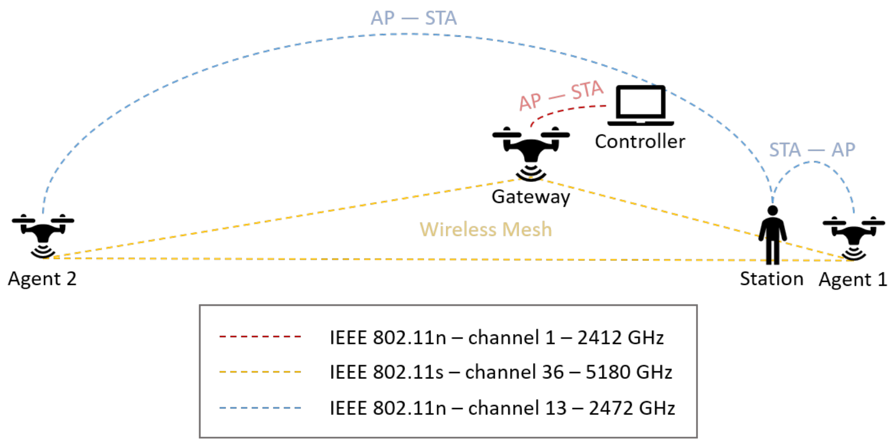

AP — STA
AP — STA
Controller
STA — AP
Gateway
Wireless Mesh
Agent 2
Station
Agent 1
IEEE 802.11n – channel 1 – 2412 GHz
IEEE 802.11s – channel 36 – 5180 GHz
IEEE 802.11n – channel 13 – 2472 GHz

Figure 5.5: Representation of the initial placement of the elements in the testbed and the way they are connected amongst themselves.

them over whenever necessary, while also balancing the load on the APs using the Jain's Fairness index algorithm. It is set to run with a threshold of $-56$ dBm, a hysteresis time of 4 s, and a smoothing factor $\alpha$ of 0.8, as presented in Section 3.6. It has no mechanism for dealing with the dynamic nature of flying networks other than its proactive scans.

- Flying Network Manager – this scenario implements both the Wi-5 solution and the Flying Network Manager application in a synchronised manner. Using the mechanisms described in Chapter 3, it schedules handovers in a predictive fashion whenever the UAVs move instructed by some topology control algorithm/mechanism.

For every scenario, a downstream flow from the gateway to the smartphone was generated using *iperf3* [70], while *horst* [71] was capturing data packets on both the gateway and the APs. Each scenario was executed 10 times, 5 for a Tranmission Control Protocol (TCP) flow, and another 5 for a User Datagram Protocol (UDP) flow. While the flow was being generated, the agents were physically swapped, i.e., Agent 1 switched its positions with Agent 2. Since the Agents were manually moved in tests, the agents' velocity may suffer regarding its accuracy, compared to when they are deployed in real UAVs. Nevertheless, we estimate the two nodes were moved at approximately $1\,\mathrm{m\,s^{-1}}$ over a distance of 13 m.

Figure 5.6 marks the points of interest for all three scenarios on a photograph of the tests venue. The cross in the middle of the photograph pertains to the third scenario, where the geographical location of the handover was calculated beforehand. In the first two scenarios, the agents started their movement at the same time, and cross each other after 6.5 m. In scenario 3, Agent 1 started its movement 1 s before Agent 2, hence they were equidistant to the smartphone after moving 7 m and 6 m, respectively. In this last scenario, the pacing at which the transportation of the agents carried out was timed more strictly, since it is imperative that the agents have crossed over each other at the time instant that was previously scheduled.

It is important to note that all nodes have their clocks synchronised to the controller's clock, to accurately log all results.

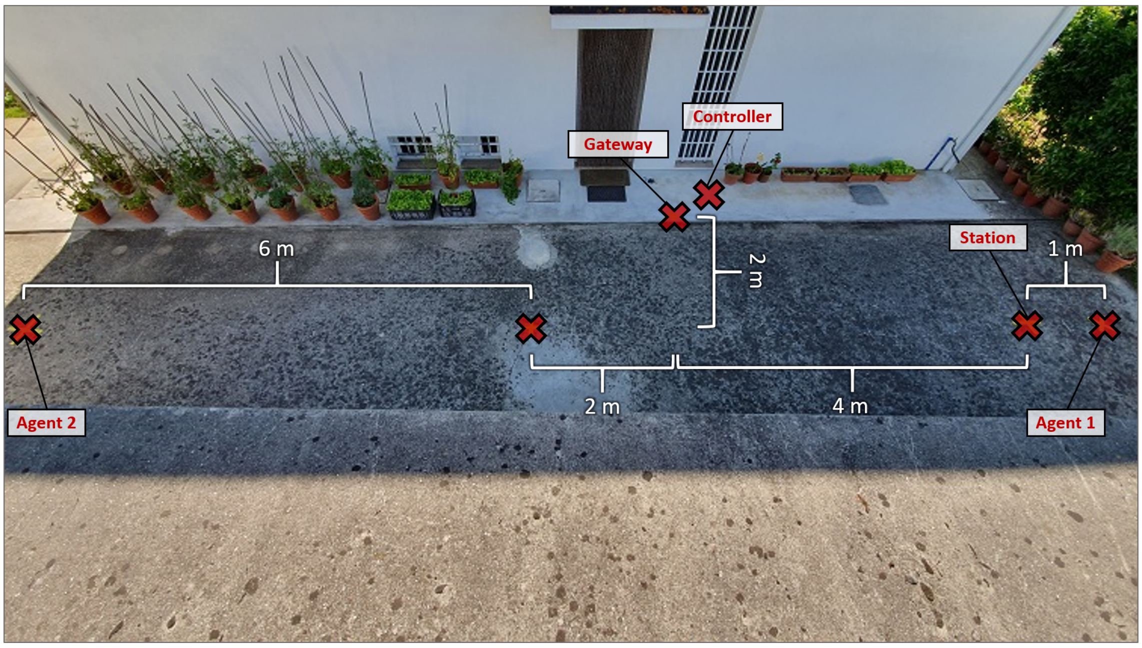


Figure 5.6: Photograph of the tests venue, including the markings of the points of interest, as well as the distances between them. The cross in the middle marks the handover point for the third scenario.

## 5.2 Results

Each scenario's subsection contains four types of charts, which represent the data that was gathered from the *iperf3* tests. All of them focus on stacking the data collected across the five trials of each flow type (TCP and UDP)—as if they were run during the same time period—, and plotting the measured values as points. Connecting the average of those values, a line is constructed, representing the overall performance of the scenario during the test. These charts are:

- **Throughput** – it contains two vertical axes: one on the left for the actual throughput measures, and another on the right to represent how many of the five trials were transmitting data at any given time during the 40 s period, which correlates with throughput but is a discrete measure, allowing an easier reading of the tests' connectivity status;
- **Data transfer** – it contains the amount of data that was transferred over time during the *iperf3* tests, and its slope represents the throughput (when no data is transferred, the slope is zero);
- **Packet loss** – the percentage of lost datagrams over time, with 100 % meaning there is no connectivity;

- **Jitter** – the packet delay variation over time, which remains constant during periods of no connectivity (i.e., it preserves the last value that was measured).

Out of these four chart types, 'Throughput' and 'Data transfer' are constructed for both TCP and UDP, while 'Packet loss' and 'Jitter' are present for UDP only.

Before running the three scenarios, the 2.4 GHz spectrum was measured in order to compare the received signal strength from both APs with its theoretical plots shown in Figure 5.3. The plot built from the spectrum measures is presented in Figure 5.7. It shows what was observed while executing some preliminary tests on the venue: there was a slight asymmetry between the received signal from both APs. At 6.5 m from each AP—the midpoint of the line segment that unites the two APs—, the signal received from the first AP is stronger than the one from the second AP, which means that the former's cell radius is larger than the latter's, causing the optimal handover point to be approximately 1.5 m closer to AP 2.

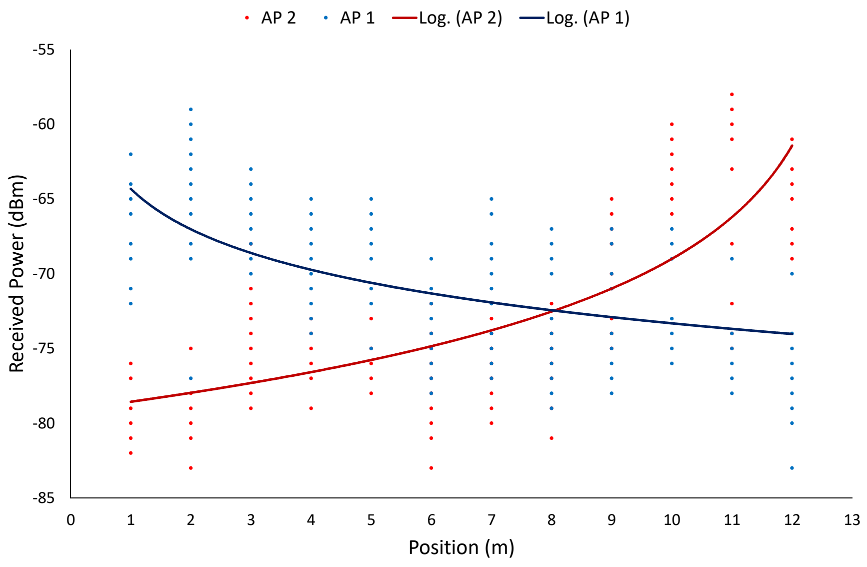


Figure 5.7: Received power from both APs measured in the ASUS laptop. Twenty samples were collected at 1 m intervals and a logarithmic trendline was added for each AP.

### 5.2.1 Baseline

In the Baseline scenario, the biggest concern lies on a well documented problem typically referred to as the "sticky client problem" [47]. A user faces this problem when its device remains connected to an AP despite having moved away from it and being in the coverage range of APs for which the received power is higher. The main consequences of this problem are degradation of the throughput over time (since the client's distance to the AP is increasing) and even the loss of connectivity to the network. This goes against the nature of flying networks, which require their APs to be able to reconfigure themselves according to the needs of clients on the ground.

In this experiment, both flows ran for a period of 40 s, with the mobility of the APs lasting approximately 13 s and scheduled to be executed 5 s into launching *iperf3*.

#### 5.2.1.1 TCP

The TCP flow was expected to either decrease in throughput or break entirely throughout this experiment. This is confirmed in Figure 5.8, which plots both the scattered points that represent the individual throughputs and the average throughput of the 5 trials that the scenario underwent in 1 s intervals. In fact, we see that the client permanently lost connectivity in a somewhat unpredictable manner. In 3 out of the 5 trials, the connectivity was lost somewhere during the crossing of the APs, when there was a shift in the received signal strength. The client failed to receive the *iperf3* test summary in every trial (which occurs at the very end), which allows to conclude that in this scenario the connection always broke down and the client was unable to recover.

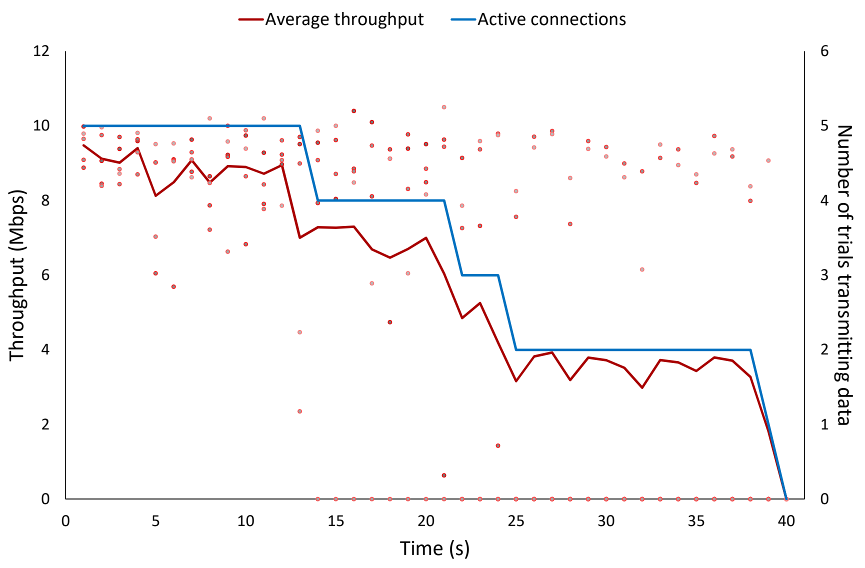


Figure 5.8: Throughput results from the TCP flow for the 5 trials of the Baseline scenario.

Figure 5.9 supports these conclusions showing that eventually all trials faced a stopping point in the data transfer.

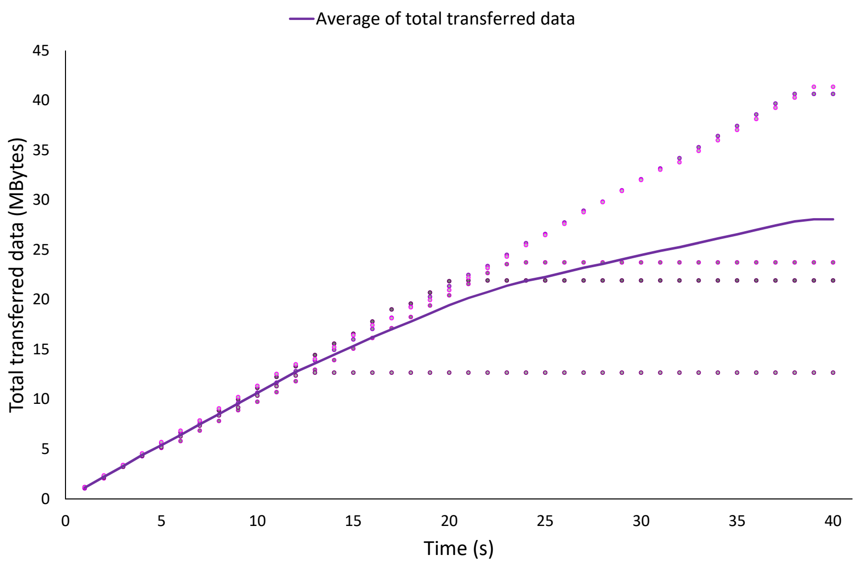


Figure 5.9: Data transfer results from the TCP flow for the 5 trials of the Baseline scenario.

#### 5.2.1.2 UDP

For the UDP flow, *iperf3* was set to saturate the link, using up all of its available bandwidth. Since UDP is a connectionless protocol, it is expected that the user may be able to recover, especially if it is able to overcome the sticky client problem and connect to the second AP. In Figures 5.10 and 5.11, we see this happening. In 3 of the 5 trials, the client is able to receive some more data after the APs have swapped places.

In Figures 5.10, 5.11 and 5.12, we can see that one of the trials had an active UDP flow at the 40 s mark. In addition, in the same trial, the STA successfully received the *iperf3* test summary. This allows to conclude that UDP flows are simple enough to recover even in the Baseline scenario.

While analysing the results, it is observable that the measures for the packet jitter (packet delay variation) do not change for a flow that was disrupted. This is shown in the time interval between 20 s and 25 s, during which all flows were disrupted; it represents standard behaviour, since this metric ignores lost packets. Nevertheless, it should be remarked to avoid a misinterpretation of Figure 5.13.

In this scenario, *iperf3* also reported several packets arriving out of order. Specifically, trial 1 had 3 packets out of order, trial 2 had 10, trial 3 had none, trial 4 had 17 packets out of order, and trial 5 had 14. This is known behaviour in UDP flows, since the protocol offers no assurances that packets will arrive in order, unlike its counterpart TCP.

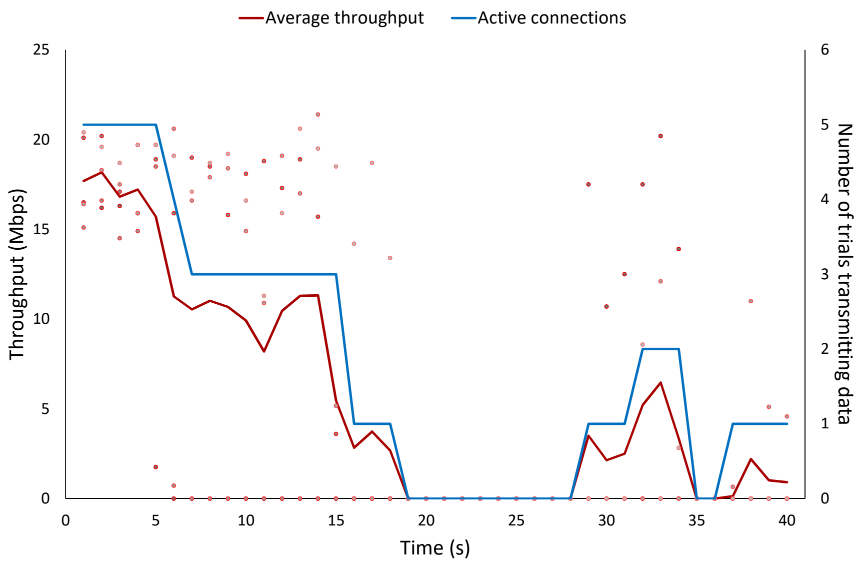


Figure 5.10: Throughput results from the UDP flow for the 5 trials of the Baseline scenario.

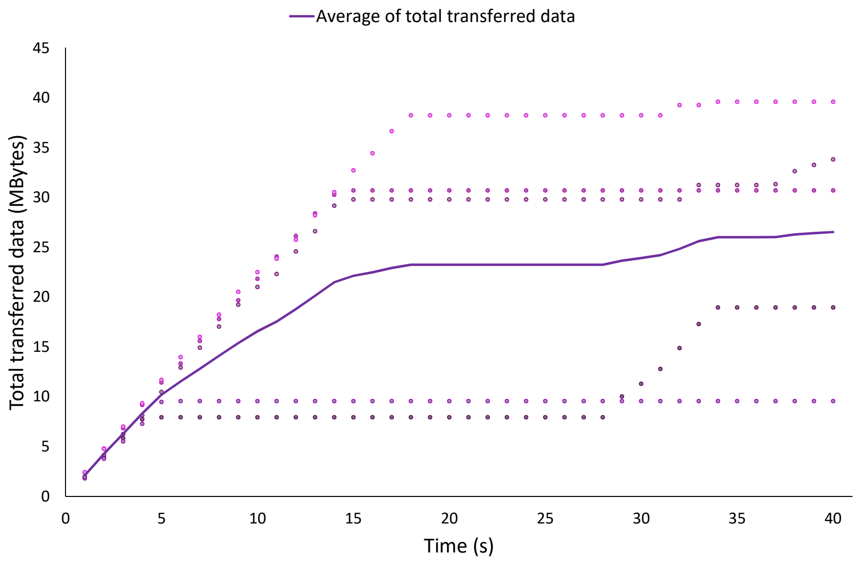


Figure 5.11: Data transfer results from the UDP flow for the 5 trials of the Baseline scenario.

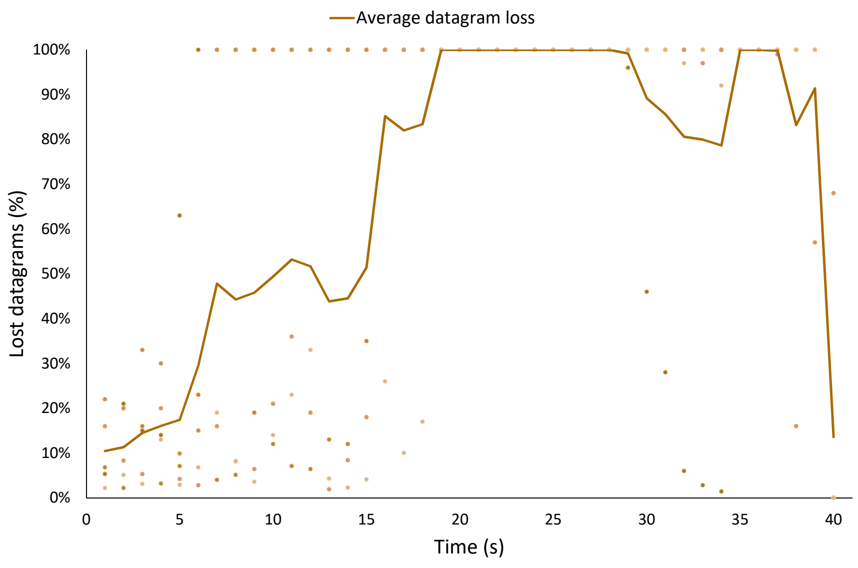


Figure 5.12: Packet loss results from the UDP flow for the 5 trials of the Baseline scenario.

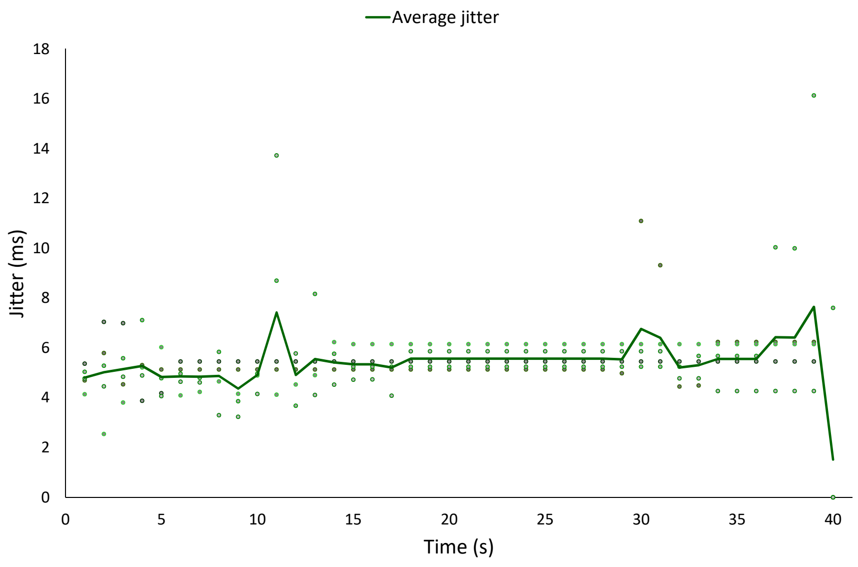


Figure 5.13: Jitter results from the UDP flow for the 5 trials of the Baseline scenario.

### 5.2.2 Wi-5

Initially, the Wi-5 scenario was defined to follow the same conditions imposed on the Baseline scenario. After running some initial tests, it was observed that the ALIX 3d3 PC Engines faced some issues while running the necessary requirements on the OpenWRT distribution, namely during the creation of the network bridge for the OpenFlow Switch (Open vSwitch), which caused unpredictable, non-deterministic, kernel panic. This increased the current and the following scenarios' difficulties, since setting up the agents required more effort and tenacity than initially considered. On top of this, it was observed that, while running UDP flows from the STA to the gateway, only low throughput flows lasted longer than a few seconds, with packet losses rising exponentially, which is an indicator that a system is receiving packets faster than it can process (UDP has no congestion control mechanism). Hence, this and the next scenarios' UDP flows are running at a fixed 3 Mbps bitrate. The remaining conditions are similar to those of the previous scenario.

#### 5.2.2.1 TCP

While running the TCP trials, and considering that changing LVAPs from a physical AP to another is not supposed to be perceivable to an IEEE 802.11 STA [58], the results depicted in Figures 5.14 and 5.15 are somewhat surprising at first glance. While in the first scenario some of the trials lasted considerably longer, in this scenario the TCP flows generated with *iperf3* are all disrupted. However, they all occur almost at the same time, taking into account that a) there are some timing discrepancies between trials, b) the APs start moving at around 5 s, and c) the optimal handover instant occurs close to 6.5 s after the APs start moving. After checking the controller's log files, we concluded that the *iperf3* TCP connections are disrupted when the controller moves the STA from one AP to another. This behaviour is very clear in the next scenario, where handovers are scheduled *a priori* and there are virtually no time discrepancies for similar events across trials.

It is very important to note that, adding to the fact that the connections appear to disrupt after a handover is performed, all TCP trials in this scenario concluded with the STA presenting the *iperf3* test summary, contrary to the first scenario.

#### 5.2.2.2 UDP

Other than the aforementioned issues, the standard behaviour of UDP also brings attention to a feature that should go hand-in-hand with the implementation of an SDN solution: it is crucial that the control flows are prioritised over the data flows in order to ensure the correct functioning of the system. If the controller is unable to obtain new informations regarding the state of the network, or orchestrate the elements of the network, the system effectively crashes. This happens because the Smart AP Selection application is not adapted to recover from agent failures—and it considers an agent not responding as failure—, making it crash, despite the Odin Master itself being able to recover from failures [58]. Coupled with the agents' kernel stability issues previously addressed, this increased the complexity of testing this implementation.

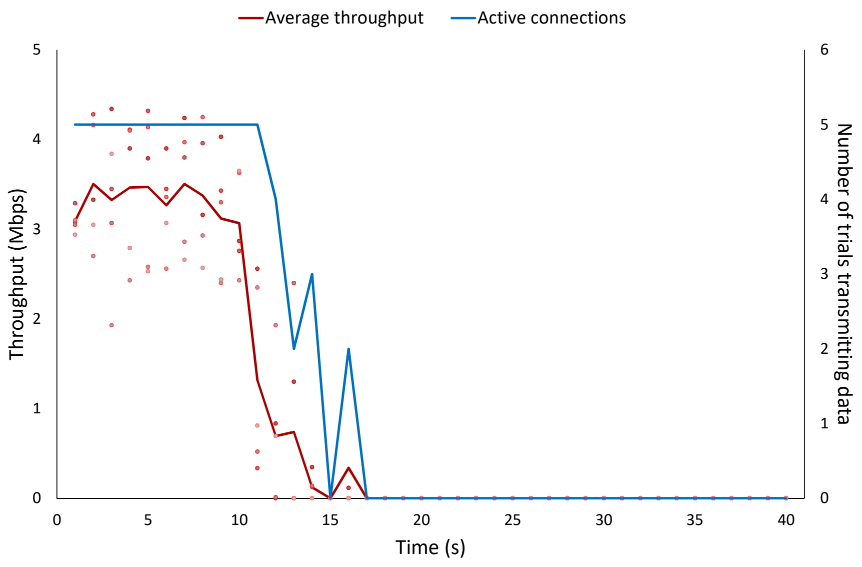


Figure 5.14: Throughput results from the TCP flow for the 5 trials of the Wi-5 scenario.

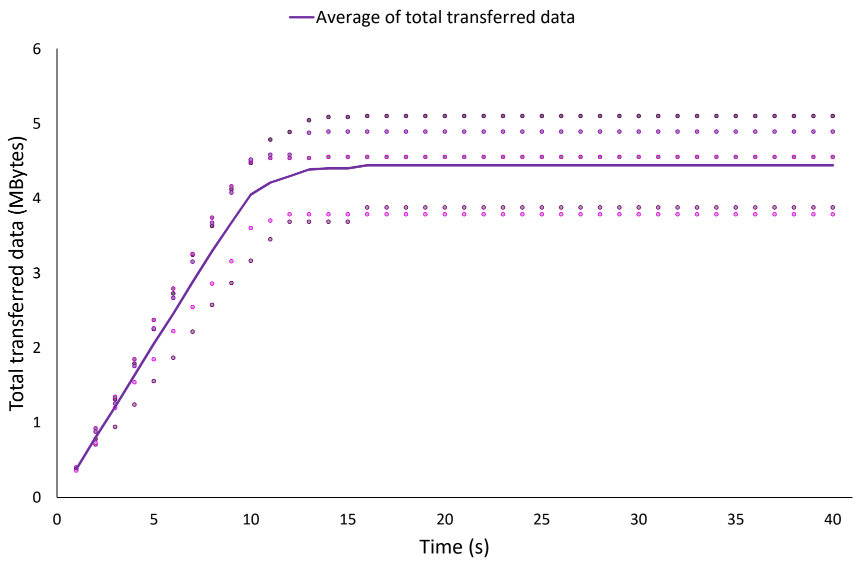


Figure 5.15: Data transfer results from the TCP flow for the 5 trials of the Wi-5 scenario.

One solution to avoid this is to implement QoS mechanisms in the network, such as *tc* for Linux, thus making sure that the controller and the agents can still communicate amongst themselves even if the network links are under heavy stress from a lack of congestion control mechanisms, as is the case with UDP.

As can be seen in Figures 5.16, 5.17 and 5.18, there is a noticeable disruption in the connectivity during the handover process. Furthermore, only 2 out of the 5 UDP trials were able to present the *iperf3* test summary, which is the same number of trials that had connectivity at the 40 s mark. This shows that, even despite the scanning capabilities implemented in Wi-5, it is not fully prepared to handle the dynamic behaviour of flying networks. Figure 5.19 shows that the packet delay variation did not change for most of the trials during long periods of time, further proving that the connectivity is lost according to the behaviour previously explained. It is also possible to observe the maximum values in jitter whenever there are changes in the connectivity status.

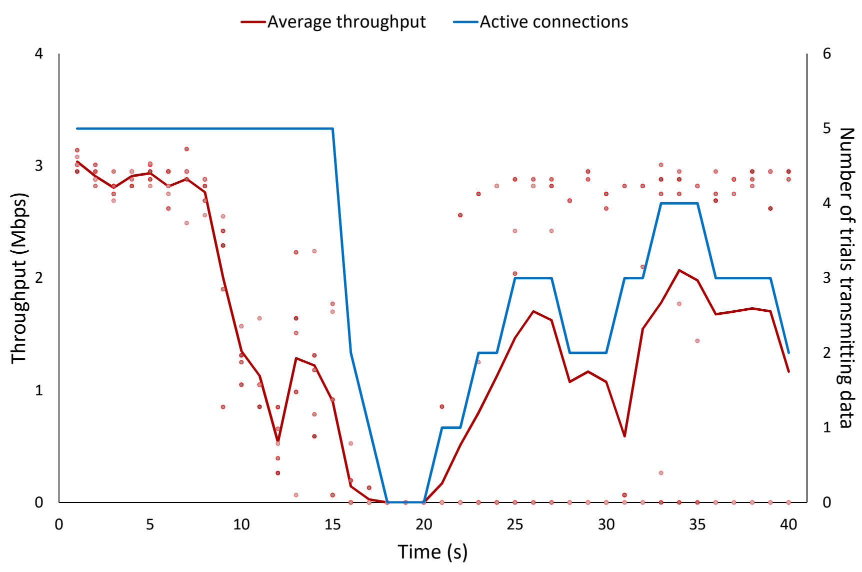


Figure 5.16: Throughput results from the UDP flow for the 5 trials of the Wi-5 scenario.

### 5.2.3 Flying Network Manager

The final test in this dissertation aims at validating the predictive concept in routing while also extrapolating its concept to the access network. It runs for a 60 s period total, with traffic flowing for 50 s, following the schedule in Table 5.3. The JSON message format[1] designed for this testbed is as follows:

[1]This was a separate mode created for the purpose of this testbed. The solution described in Chapter 3 is also implemented in the application.

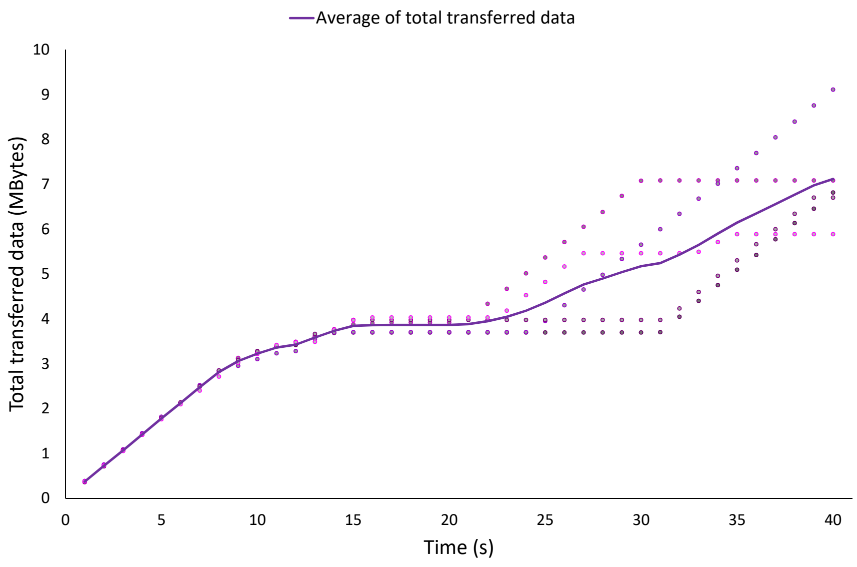


Figure 5.17: Data transfer results from the UDP flow for the 5 trials of the Wi-5 scenario.

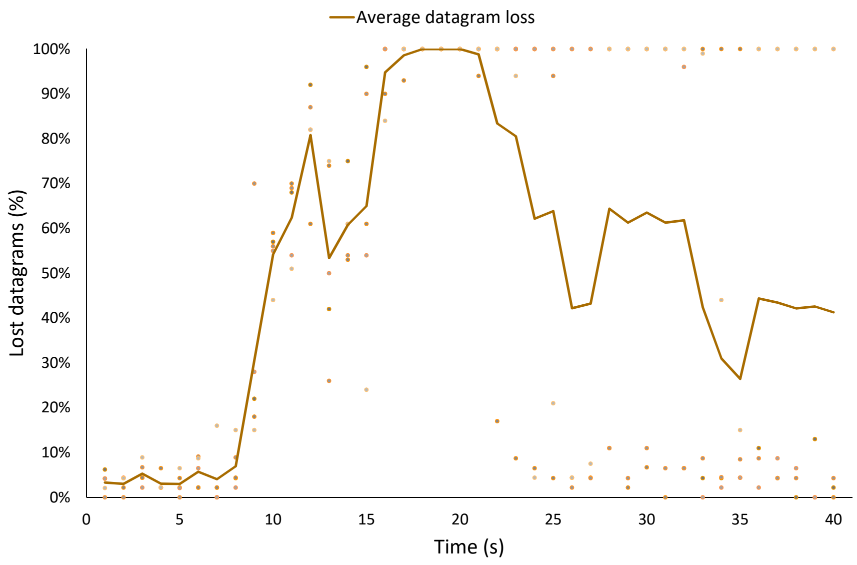


Figure 5.18: Packet loss results from the UDP flow for the 5 trials of the Wi-5 scenario.

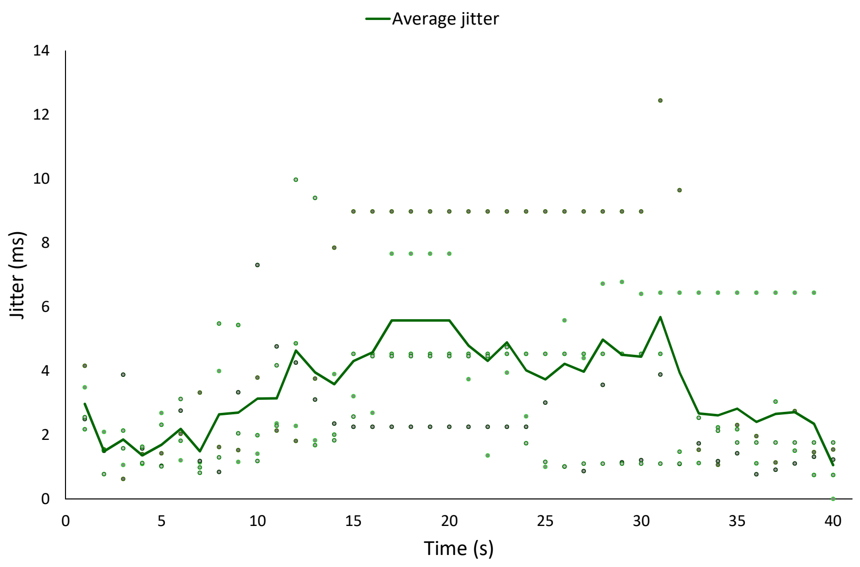


Figure 5.19: Jitter results from the UDP flow for the 5 trials of the Wi-5 scenario.

```
{
    "172.16.0.1":{
        "startTime":"2019-06-21 02:17:00.000",
        "handoffTime":"2019-06-21 02:17:07.000"
    },
    "172.16.0.2":{
        "startTime":"2019-06-21 02:17:01.000",
        "handoffTime":"2019-06-21 02:17:07.000"
    }
}
```

#### 5.2.3.1 TCP

Figures 5.20 and 5.21 show that the TCP flow is disrupted shortly after the two APs start moving, according to the schedule. However, when inspecting the *iperf3* logs on the STA, it was noted that the flow had a delay of about 4 s before starting. This effectively means that the plots in the figures are delayed (i.e., shifted to the left). With this in mind, it becomes clear that the flow disruption always occurs around the time instant that the handover is executed. It is also worth noting that the *iperf3* utility managed to show the TCP test summaries in this scenario.

Table 5.3: Event schedule for the Flying Network Manager test scenario.

| Time (s) | Event |
|---|---|
| -10 | *Horst* starts executing and the *iperf3* server is initiated at the gateway |
| -5 | A JSON topology control message specifically designed for this testbed is sent to the controller using the *netcat* networking utility from the same laptop that is running the controller |
| 0 | The *iperf3* client on the STA connects to the server on the gateway and starts receiving traffic for a duration of 50 s |
| 10 | AP 1 starts moving |
| 11 | AP 1 crosses the STA and AP 2 starts moving |
| 17 | The APs cross one another, are now equidistant to the STA, and the handover is executed |
| 23 | AP 1 arrives to the starting location of AP 2 |
| 24 | AP 2 arrives to the starting location of AP 1 |
| 50 | The test is finished |

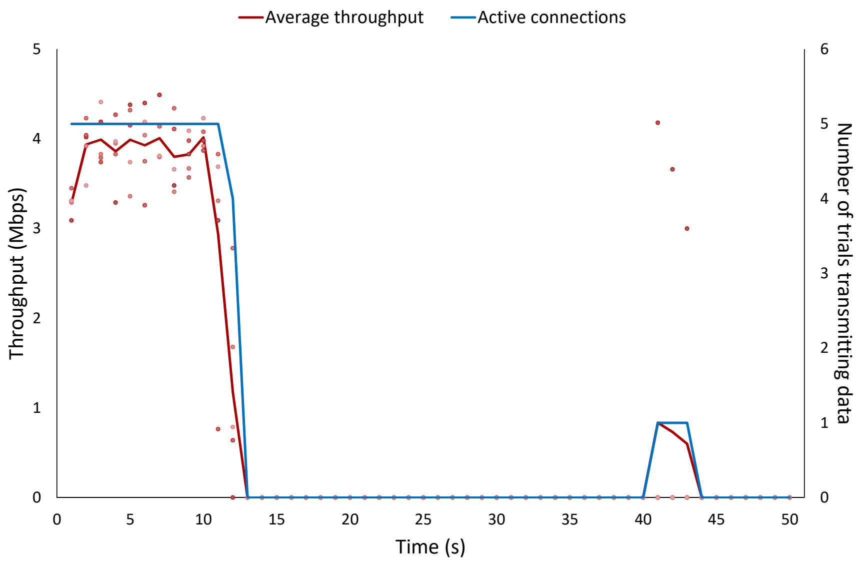


Figure 5.20: Throughput results from the TCP flow for the 5 trials of the Flying Network Manager scenario.

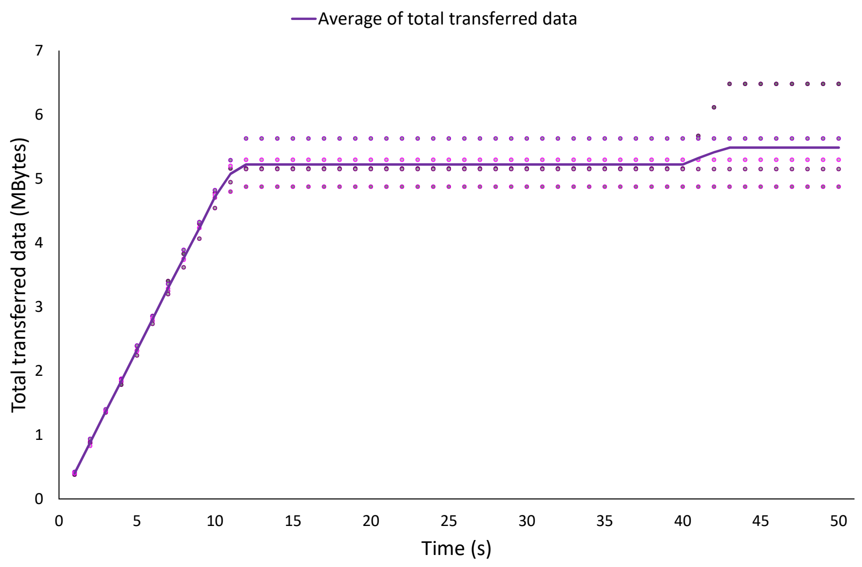


Figure 5.21: Data transfer results from the TCP flow for the 5 trials of the Flying Network Manager scenario.

#### 5.2.3.2 UDP

The Flying Network Manager application managed to recover from the interrupted flows every time during the UDP trials. It is the only of the three scenarios to do so, and more so, it recovers with a consistent throughput. Nevertheless, it is important to note that the bitrate at which this test was running is significantly lower than that of the first scenario. The second scenario could not adequately adapt to the dynamic changes in the physical topology. One aspect that requires further investigation is the fact that this recovery takes place at approximately the same time that the APs stop moving, which is when the Flying Network Manager is scheduled to resume the Smart AP Selection application. Figures 5.22, 5.23 and 5.24 provide visual support for these conclusions.

Figure 5.25 shows bigger maxima than its counterparts, specifically during the recovery period, which also indicates that there is connectivity. Otherwise, the packet delay variation would stay constant.

## 5.3 Summary

All three scenarios faced a dynamic change in the network topology consisting in swapping two APs over a 13 m distance. Three scenarios were tested, including both TCP and UDP flows. Measures regarding throughput, connectivity, transferred data, packet loss and delay variation were collected, analysed and compared. The Flying Network Manager proved to be the most

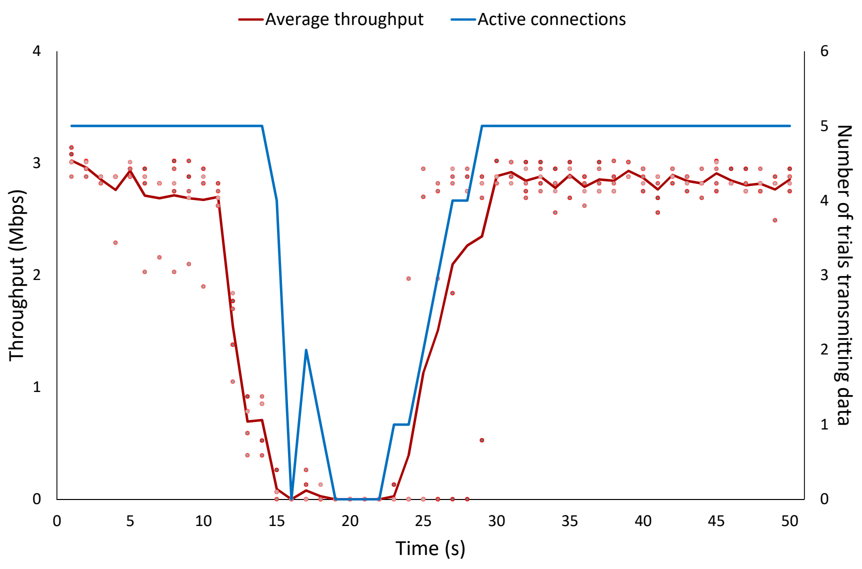


Figure 5.22: Throughput results from the UDP flow for the 5 trials of the Flying Network Manager scenario.

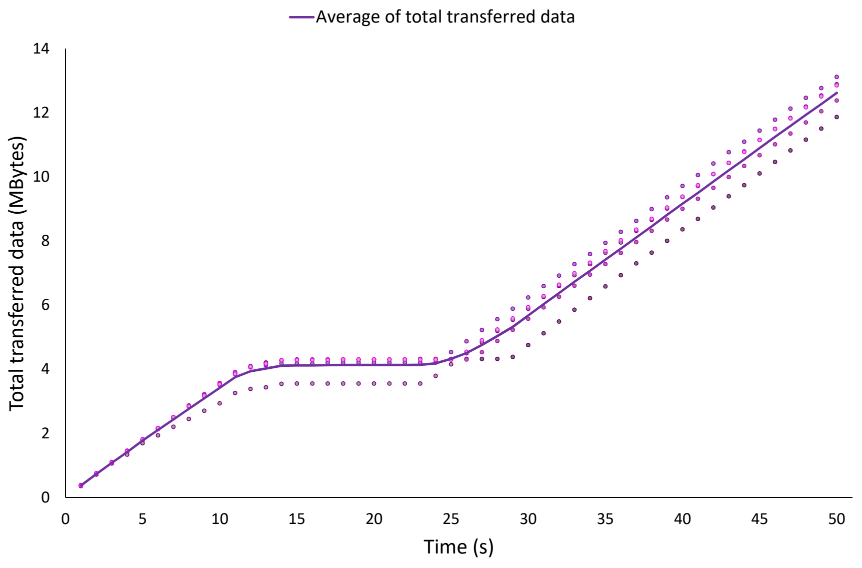


Figure 5.23: Data transfer results from the UDP flow for the 5 trials of the Flying Network Manager scenario.

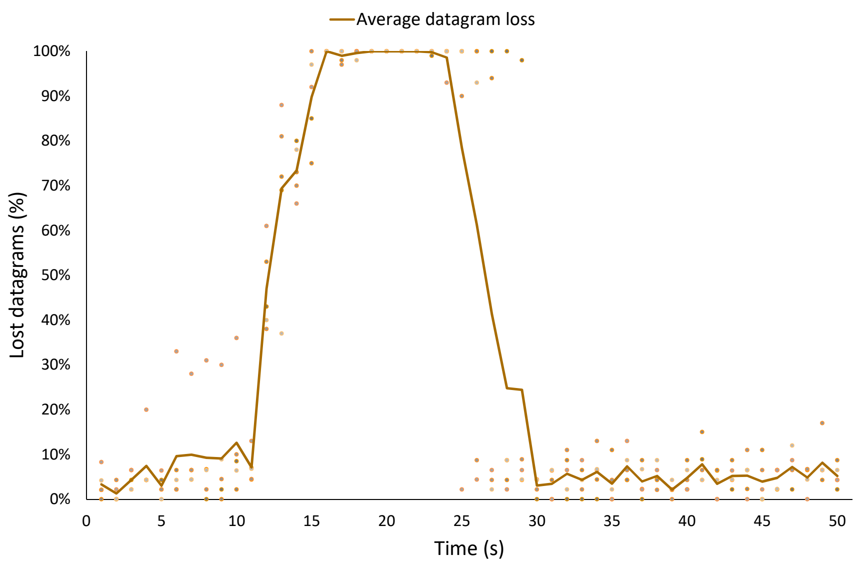


Figure 5.24: Packet loss results from the UDP flow for the 5 trials of the Flying Network Manager scenario.

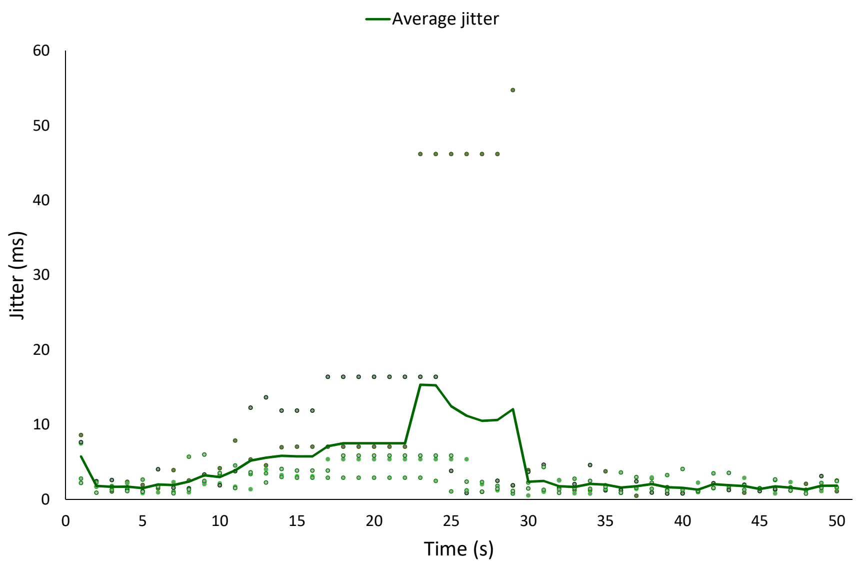


Figure 5.25: Jitter results from the UDP flow for the 5 trials of the Flying Network Manager scenario.

stable, providing the best connectivity. All three scenarios' setbacks were shown, and where and why they should improve.

For the two SDN-based scenarios, the measured throughput was lower than expected. This was a consequence of using virtual interfaces for monitoring purposes on top of the NICs that were being used as APs. However, this was the only available option since using "dummy" interfaces for monitoring resulted in the controller not showing any measured RSSIs for frames going through the agents. This requires a fix in the Wi-5 scanning mechanism, allowing it to register the RSSI of frames that it captures from STAs through the interface that is servicing them. As an alternative, three NICs must be used.

In all three scenarios, there are noticeable periods during which the connection is disrupted or even dropped, mainly when both APs are equally far to the STA (the optimal handover point regarding signal strength). This is a consequence of using the 20 dB attenuators for the antennas of the APs, resulting in a very low power setting, which means that the testbed Wi-Fi cells were not overlapping as much as they could, resulting in periods during which every connection available to the STA was very unstable. Data from the logs show that the handovers were successfully executed, although the frames sent by the APs were not being processed by the STA for a significant amount of time, resulting in what appears as an abrupt connectivity loss.

As mentioned in Sections 4.2 and 5.1, the UAV elements are connected using the IEEE 802.11s amendment, which defines how wireless devices can connect amongst themselves and form a WLAN mesh network. This amendment comes with a default mandatory routing protocol, the Hybrid Wireless Mesh Protocol (HWMP), which may hold a significant impact on the network. However, it was not this dissertation or this evaluation setup's purpose to assess how this protocol fits within the proposed solution.

Table 5.4 quantifies the average UDP results gathered from the collected data across the three scenarios with a 95 % confidence interval. The 'Uptime' column was calculated considering the average percentage of time that each scenario's STA was transmitting data. It should be noted that these were calculated by shortening the 50 s trials of the third scenario by 10 s, making these values valid for comparison with its two counterparts. In addition to this, the first scenario's row has been greyed out since its bandwidth settings differ from the other two scenarios'. As such, they should not be used for comparison and are present for reference only. Figure 5.26 stacks the connectivity status of each scenario's 5 UDP trials, thus being 5 the highest number of trials with connectivity that a scenario can have at any given time. The Flying Network Manager is the only scenario solution that managed to recover connectivity on all 5 trials.

Table 5.4: Summary of the average UDP results from the three scenarios.

| Scenario | Throughput (Kbps) | Packet loss (%) | Jitter (ms) | Uptime (%) |
|---|---|---|---|---|
| Baseline | $5562 \pm 1113$ | $69.58 \pm 5.78$ | $5.44 \pm 0.22$ | $36.00 \pm 11.81$ |
| Wi-5 | $1458 \pm 180$ | $50.89 \pm 6.05$ | $3.52 \pm 0.36$ | $65.50 \pm 6.41$ |
| Flying Network Manager | $1878 \pm 178$ | $38.34 \pm 5.98$ | $5.25 \pm 1.22$ | $76.00 \pm 2.97$ |

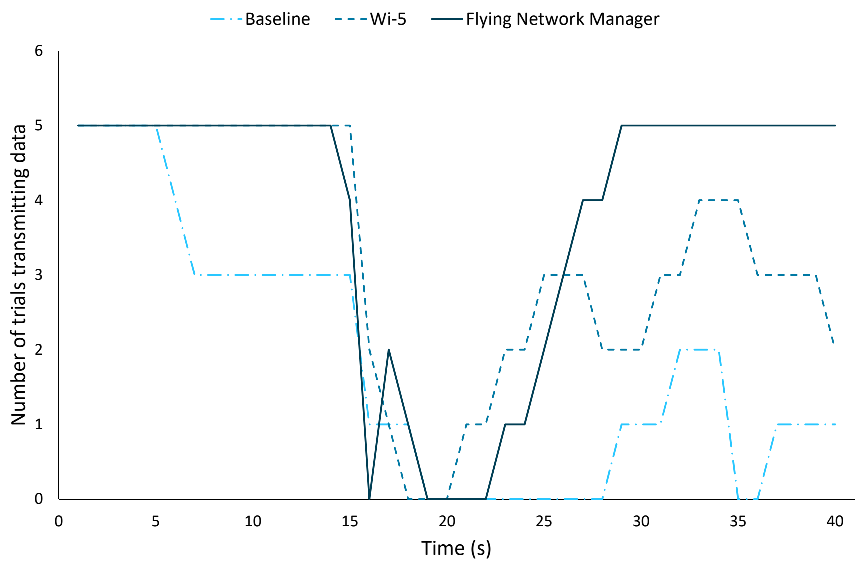


Figure 5.26: UDP connectivity results comparison. The 5 UDP trials of each scenario are stacked.

# Chapter 6

# Conclusions

Swarms of UAVs are seeing an increasing number of applications, namely when it comes to Flying Networks, whose UAVs connect ground users to the Internet. Solutions consisting of a centralised view of the network are promising, in which they allow the UAVs to position themselves according to the users' needs. However, they do not consider the impact of network reconfigurations on the user terminals. In addition to this, the state of the art only offers active, reactive, or hybrid protocols to enable dynamic routing in these networks, all of which only recover from failures in links after they occur.

In this dissertation, a routing solution for flying networks based on SDN was proposed and implemented. It focused on the mobility management and network load balancing issues from a centralised perspective, using a predictive approach which separates it from its counterpart solutions in the state of the art. Specifically, it considers the future positions of UAVs *a priori*, scheduling the necessary operations to occur before they result in link failures.

This dissertation developed and tested an application that runs on top of an SDN controller with a holistic view of the network. It gathers information from a set of agents running on board of UAVs, which virtualise the association state of APs in order to handle the mobility of ground users seamlessly.

The developed solution was tested against a standard configuration and a proactive SDN approach, and it showed promising results in regard to end-to-end connectivity in Flying Networks. Considering the UDP results, it guaranteed that clients stayed connected with greater success than its counterparts, which did not always manage to even recover the connection that was established prior to a handover. It is an important step forward from conventional routing solutions which focus on recovering from link failures after they occur.

In the end, all of the objectives of this dissertation were achieved. The main difficulties lied in the logistics associated with performing the live tests, mainly due to the importance of having events synchronised across the different trials. The PC Engines used for running the Agents further increased the difficulty in the implementation of this solution, as they would frequently crash while running standard OvS setup operations and also couldn't handle very high throughput UDP flows.

## 6.1 Future work

This dissertation could benefit from further testing and improvements, such as:

- Testing the application in a larger venue, with real UAVs and a topology control algorithm running simultaneously.
- Testing the application with various cell areas, effectively changing how much they overlap one another.
- Validating and measuring the performance of relay nodes.
- Testing the SDN-based scenarios using a QoS mechanism.
- Testing with more complex scenarios, including both multiple gateways and relay nodes.
- Evaluating the impact of the IEEE 802.11s WLAN Mesh Network on the developed solution. Namely, the HWMP.
- Fine tuning the Smart AP Application's mobility parameters for flying network scenarios.
- Having the Smart AP Selection application recover from crashed nodes.

# References

[1] İlker Bekmezci, Ozgur Koray Sahingoz, and Şamil Temel. Flying Ad-Hoc Networks (FANETs): A survey. *Ad Hoc Networks*, 11(3):1254–1270, May 2013. URL: https://linkinghub.elsevier.com/retrieve/pii/S1570870512002193 [last accessed 2018-11-23], doi:10.1016/j.adhoc.2012.12.004.

[2] Eduardo Nuno Almeida, Rui Campos, and Manuel Ricardo. Traffic-aware multi-tier flying network: Network planning for throughput improvement. In *2018 IEEE Wireless Communications and Networking Conference (WCNC)*, pages 1–6, Barcelona, April 2018. IEEE. URL: https://ieeexplore.ieee.org/document/8377408/ [last accessed 2018-11-23], doi:10.1109/WCNC.2018.8377408.

[3] André Coelho, Eduardo Nuno Almeida, Pedro Silva, Jose Ruela, Rui Campos, and Manuel Ricardo. RedeFINE: Centralized Routing for High-capacity Multi-hop Flying Networks. In *2018 IEEE 14th International Conference on Wireless and Mobile Computing, Networking and Communications (WiMob)*, page 8, Cyprus, October 2018. IEEE.

[4] IEEE Standard for Information technology–Telecommunications and information exchange between systems Local and metropolitan area networks–Specific requirements - Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications. *IEEE Std 802.11-2016 (Revision of IEEE Std 802.11-2012)*, pages 1–3534, December 2016. doi:10.1109/IEEESTD.2016.7786995.

[5] Intel. Helping Define IEEE 802.11n and other Wireless LAN Standards, 2012. URL: https://www.intel.com/content/www/us/en/standards/802-11-wireless-lan-standards-study.html [last accessed 2018-12-29].

[6] IEEE 802.11. IEEE 802.11, The Working Group Setting the Standards for Wireless LANs. URL: http://www.ieee802.org/11/ [last accessed 2018-12-29].

[7] Justin Berg. The IEEE 802.11 Standardization: Its History, Specifications, Implementations, and Future, 2011.

[8] A. Guillen-Perez, R. Sanchez-Iborra, M. Cano, J. C. Sanchez-Aarnoutse, and J. Garcia-Haro. WiFi networks on drones. In *2016 ITU Kaleidoscope: ICTs for a Sustainable World (ITU WT)*, pages 1–8, November 2016. doi:10.1109/ITU-WT.2016.7805730.

[9] S. Hayat, E. Yanmaz, and R. Muzaffar. Survey on Unmanned Aerial Vehicle Networks for Civil Applications: A Communications Viewpoint. *IEEE Communications Surveys Tutorials*, 18(4):2624–2661, Fourthquarter 2016. doi:10.1109/COMST.2016.2560343.

[10] VNI Cisco. Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2016–2021 White Paper. Technical report, Cisco, 2017. URL: https:

//www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/mobile-white-paper-c11-520862.html [last accessed 2018-12-29].

[11] L. Gupta, R. Jain, and G. Vaszkun. Survey of Important Issues in UAV Communication Networks. *IEEE Communications Surveys Tutorials*, 18(2):1123–1152, Secondquarter 2016. doi:10.1109/COMST.2015.2495297.

[12] A. V. Leonov and G. A. Litvinov. Applying AODV and OLSR routing protocols to air-to-air scenario in flying ad hoc networks formed by mini-UAVs. In *2018 Systems of Signals Generating and Processing in the Field of on Board Communications*, pages 1–10, March 2018. doi:10.1109/SOSG.2018.8350612.

[13] X. Zhang, H. Wang, and H. Zhao. An SDN framework for UAV backbone network towards knowledge centric networking. In *IEEE INFOCOM 2018 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*, pages 456–461, April 2018. doi:10.1109/INFCOMW.2018.8406959.

[14] Gokhan Secinti, Parisa Borhani Darian, Berk Canberk, and Kaushik R. Chowdhury. SDNs in the Sky: Robust End-to-End Connectivity for Aerial Vehicular Networks. *IEEE Communications Magazine*, 56(1):16–21, January 2018. URL: http://ieeexplore.ieee.org/document/8255732/ [last accessed 2018-11-23], doi:10.1109/MCOM.2017.1700456.

[15] D. Ahmed and O. Khalifa. An overview of MANETs: Applications, characteristics, challenges and recent issues. *IJEAT*, 3:128, 2017.

[16] Oussama Stiti, Othmen Braham, and Guy Pujolle. Virtual openflow-based SDN Wi-Fi access point. In *2015 Global Information Infrastructure and Networking Symposium (GIIS)*, pages 1–3, Guadalajara, Mexico, October 2015. IEEE. URL: http://ieeexplore.ieee.org/document/7347190/ [last accessed 2018-11-30], doi:10.1109/GIIS.2015.7347190.

[17] Zhongliang Zhao, Pedro Cumino, Arnaldo Souza, Denis Rosário, Torsten Braun, Eduardo Cerqueira, and Mario Gerla. Software-defined unmanned aerial vehicles networking for video dissemination services. *Ad Hoc Networks*, 83:68–77, February 2019. URL: https://linkinghub.elsevier.com/retrieve/pii/S1570870518306231 [last accessed 2018-12-05], doi:10.1016/j.adhoc.2018.08.023.

[18] Quentin Monnet. An introduction to SDN, August 2016. URL: https://qmonnet.github.io/whirl-offload/2016/07/08/introduction-to-sdn/ [last accessed 2018-12-31].

[19] Evangelos Haleplidis, Kostas Pentikousis, Spyros Denazis, Jamal Hadi Salim, David Meyer, and Odysseas Koufopavlou. Software-Defined Networking (SDN): Layers and Architecture Terminology. (7426):35, January 2015. URL: https://rfc-editor.org/rfc/rfc7426.txt, doi:10.17487/RFC7426.

[20] Borja Nogales, Victor Sanchez-Aguero, Ivan Vidal, Francisco Valera, and Jaime Garcia-Reinoso. A NFV system to support configurable and automated multi-UAV service deployments. In *Proceedings of the 4th ACM Workshop on Micro Aerial Vehicle Networks, Systems, and Applications - DroNet'18*, pages 39–44, Munich, Germany, 2018. ACM

Press. URL: http://dl.acm.org/citation.cfm?doid=3213526.3213534 [last accessed 2018-11-30], doi:10.1145/3213526.3213534.

[21] Jinfang Jiang and Guangjie Han. Routing Protocols for Unmanned Aerial Vehicles. *IEEE Communications Magazine*, 56(1):58–63, January 2018. URL: http://ieeexplore.ieee.org/document/8255738/ [last accessed 2018-11-23], doi:10.1109/MCOM.2017.1700326.

[22] X. Fan, W. Cai, and J. Lin. A survey of routing protocols for highly dynamic mobile ad hoc networks. In *2017 IEEE 17th International Conference on Communication Technology (ICCT)*, pages 1412–1417, October 2017. doi:10.1109/ICCT.2017.8359865.

[23] Chen-Mou Cheng, Pai-Hsiang Hsiao, H. T. Kung, and Dario Vlah. Maximizing throughput of UAV-relaying networks with the load-carry-and-deliver paradigm. In *Wireless Communications and Networking Conference, 2007. WCNC 2007. IEEE*, pages 4417–4424. IEEE, 2007.

[24] Yangguang Fu, Mingyue Ding, Chengping Zhou, and Hanping Hu. Route planning for unmanned aerial vehicle (UAV) on the sea using hybrid differential evolution and quantum-behaved particle swarm optimization. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 43(6):1451–1465, 2013.

[25] B. Bellur, M. Lewis, and F. Templin. An ad-hoc network for teams of autonomous vehicles. In *Proceedings of the First Annual Symposium on Autonomous Intelligence Networks and Systems*, pages 1–6, 2002.

[26] Abdel Ilah Alshabtat, Liang Dong, J. Li, and F. Yang. Low latency routing algorithm for unmanned aerial vehicles ad-hoc networks. *International Journal of Electrical and Computer Engineering*, 6(1):48–54, 2010.

[27] Stefano Rosati, Karol Kruzelecki, Louis Traynard, and Bixio Rimoldi. Speed-aware routing for UAV ad-hoc networks. In *Globecom Workshops (GC Wkshps), 2013 IEEE*, pages 1367–1373. IEEE, 2013.

[28] Yi Zheng, Yuwen Wang, Zhenzhen Li, Li Dong, Yu Jiang, and Hong Zhang. A mobility and load aware OLSR routing protocol for UAV mobile ad-hoc networks. 2014.

[29] Vineet R. Khare, Frank Z. Wang, Sining Wu, Yuhui Deng, and Chris Thompson. Ad-hoc network of unmanned aerial vehicle swarms for search & destroy tasks. In *Intelligent Systems, 2008. IS'08. 4th International IEEE Conference*, volume 1, pages 6–65. IEEE, 2008.

[30] J. Hope Forsmann, Robert E. Hiromoto, and John Svoboda. A time-slotted on-demand routing protocol for mobile ad hoc unmanned vehicle systems. In *Unmanned Systems Technology IX*, volume 6561, page 65611P. International Society for Optics and Photonics, 2007.

[31] Jean-Daniel Medjo Me Biomo, Thomas Kunz, and Marc St-Hilaire. Routing in unmanned aerial ad hoc networks: Introducing a route reliability criterion. In *Wireless and Mobile Networking Conference (WMNC), 2014 7th IFIP*, pages 1–7. IEEE, 2014.

[32] Omar Sami Oubbati, Abderrahmane Lakas, Fen Zhou, Mesut Güneş, Nasreddine Lagraa, and Mohamed Bachir Yagoubi. Intelligent UAV-assisted routing protocol for urban VANETs. *Computer Communications*, 107:93–111, 2017.

[33] Kesheng Liu, Jun Zhang, and Tao Zhang. The clustering algorithm of UAV networking in near-space. In *Antennas, Propagation and EM Theory, 2008. ISAPE 2008. 8th International Symposium On*, pages 1550–1553. IEEE, 2008.

[34] Chunhua Zang and Shouhong Zang. Mobility prediction clustering algorithm for UAV networking. In *GLOBECOM Workshops (GC Wkshps), 2011 IEEE*, pages 1158–1161. IEEE, 2011.

[35] Z. Zhai, J. Du, and Y. Ren. The Application and Improvement of Temporally Ordered Routing Algorithm in Swarm Network with Unmanned Aerial Vehicle Nodes,|| In ICWMC 2013. In *The Ninth International Conference on Wireless and Mobile Communications*, pages 7–12, 2013.

[36] M. T. Hyland, Barry E. Mullins, Rusty O. Baldwin, and Michael A. Temple. Simulation-based performance evaluation of mobile ad hoc routing protocols in a swarm of unmanned aerial vehicles. In *Advanced Information Networking and Applications Workshops, 2007, AINAW'07. 21st International Conference On*, volume 2, pages 249–256. IEEE, 2007.

[37] Jean-Daniel Medjo Me Biomo, Thomas Kunz, and Marc St-Hilaire. Routing in unmanned aerial ad hoc networks: A recovery strategy for greedy geographic forwarding failure. In *Wireless Communications and Networking Conference (WCNC), 2014 IEEE*, pages 2236–2241. IEEE, 2014.

[38] J. Wang, C. Jiang, Z. Han, Y. Ren, R. G. Maunder, and L. Hanzo. Taking Drones to the Next Level: Cooperative Distributed Unmanned-Aerial-Vehicular Networks for Small and Mini Drones. *IEEE Vehicular Technology Magazine*, 12(3):73–82, September 2017. doi:10.1109/MVT.2016.2645481.

[39] Abhinandan Ramaprasath, Anand Srinivasan, Chung-Horng Lung, and Marc St-Hilaire. Intelligent Wireless Ad Hoc Routing Protocol and Controller for UAV Networks. In Yifeng Zhou and Thomas Kunz, editors, *Ad Hoc Networks*, Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, pages 92–104. Springer International Publishing, 2017.

[40] A. Rovira-Sugranes and A. Razi. Predictive routing for dynamic UAV networks. In *2017 IEEE International Conference on Wireless for Space and Extreme Environments (WiSEE)*, pages 43–47, October 2017. doi:10.1109/WiSEE.2017.8124890.

[41] Stefano Ferretti, Vittorio Ghini, and Fabio Panzieri. A survey on handover management in mobility architectures. *Computer Networks*, 94:390–413, January 2016. URL: http://www.sciencedirect.com/science/article/pii/S1389128615004491 [last accessed 2018-12-28], doi:10.1016/j.comnet.2015.11.016.

[42] N. Montavont, A. Arcia-Moret, and G. Castignani. On the selection of scanning parameters in IEEE 802.11 networks. In *2013 IEEE 24th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*, pages 2137–2141, September 2013. doi:10.1109/PIMRC.2013.6666497.

[43] J. Jeong, Y. D. Park, and Y. Suh. An Efficient Channel Scanning Scheme With Dual-Interfaces for Seamless Handoff in IEEE 802.11 WLANs. *IEEE Communications Letters*, 22(1):169–172, January 2018. doi:10.1109/LCOMM.2017.2763941.

[44] M. I. Sanchez and A. Boukerche. On IEEE 802.11K/R/V amendments: Do they have a real impact? *IEEE Wireless Communications*, 23(1):48–55, February 2016. doi:10.1109/MWC.2016.7422405.

[45] S. Martinez, N. Cardona, and J. F. Botero. Seamless Handoff Management in IEEE 802.11 Networks Using SDN. In *2018 IEEE Colombian Conference on Communications and Computing (COLCOM)*, pages 1–6, May 2018. doi:10.1109/ColComCon.2018.8466702.

[46] Anatolij Zubow, Sven Zehl, and Adam Wolisz. BIGAP—Seamless handover in high performance enterprise IEEE 802.11 networks. In *Network Operations and Management Symposium (NOMS), 2016 IEEE/IFIP*, pages 445–453. IEEE, 2016.

[47] J. Saldana, R. Munilla, S. Eryigit, O. Topal, J. Ruiz-Mas, J. Fernández-Navajas, and L. Sequeira. Unsticking the Wi-Fi Client: Smarter Decisions Using a Software Defined Wireless Solution. *IEEE Access*, 6:30917–30931, 2018. doi:10.1109/ACCESS.2018.2844088.

[48] Jonathan Vestin, Peter Dely, Andreas Kassler, Nico Bayer, Hans Einsiedler, and Christoph Peylo. CloudMAC: Towards Software Defined WLANs. *SIGMOBILE Mob. Comput. Commun. Rev.*, 16(4):42–45, February 2013. URL: http://doi.acm.org/10.1145/2436196.2436217 [last accessed 2019-01-01], doi:10.1145/2436196.2436217.

[49] Anyfi Networks. Software-Defined Wireless Networking: Concepts, Principles and Motivations. Technical report, April 2015. URL: http://www.anyfinetworks.com/get_resource/anyfi-sdwn-concepts-whitepaper [last accessed 2019-01-01].

[50] Nicolas Montavont, Alberto Blanc, Renzo Efrain Navas, Tanguy Kerdoncuff, and German Castignani. Handover Triggering in IEEE 802.11 Networks. In *IEEE 16th International Symposium on A World of Wireless, Mobile and Multimedia Networks (WoWMoM)*, page ., Boston, United States, June 2015. URL: https://hal.archives-ouvertes.fr/hal-01759108 [last accessed 2019-01-01], doi:10.1109/WoWMoM.2015.7158126.

[51] R. Riggio, M. K. Marina, J. Schulz-Zander, S. Kuklinski, and T. Rasheed. Programming Abstractions for Software-Defined Wireless Networks. *IEEE Transactions on Network and Service Management*, 12(2):146–162, June 2015. doi:10.1109/TNSM.2015.2417772.

[52] K. Nakauchi and Y. Shoji. WiFi Network Virtualization to Control the Connectivity of a Target Service. *IEEE Transactions on Network and Service Management*, 12(2):308–319, June 2015. doi:10.1109/TNSM.2015.2403956.

[53] C. Chen, Y. Lin, L. Yen, M. Chan, and C. Tseng. Mobility management for low-latency handover in SDN-based enterprise networks. In *2016 IEEE Wireless Communications and Networking Conference*, pages 1–6, April 2016. doi:10.1109/WCNC.2016.7565105.

[54] C. Xu, W. Jin, G. Zhao, H. Tianfield, S. Yu, and Y. Qu. A Novel Multipath-Transmission Supported Software Defined Wireless Network Architecture. *IEEE Access*, 5:2111–2125, 2017. doi:10.1109/ACCESS.2017.2653244.

[55] Julius Schulz-Zander, Lalith Suresh, Nadi Sarrar, Anja Feldmann, Thomas Hühn, and Ruben Merz. Programmatic Orchestration of WiFi Networks. In *Proceedings of the 2014 USENIX Conference on USENIX Annual Technical Conference*, USENIX ATC'14, pages 347–358, Berkeley, CA, USA, 2014. USENIX Association. URL: http://dl.acm.org/citation.cfm?id=2643634.2643670 [last accessed 2019-01-01].

[56] SDN Technical Specifications, 2019. URL: https://www.opennetworking.org/software-defined-standards/specifications/ [last accessed 2019-01-30].

[57] Open vSwitch, 2016. URL: https://www.openvswitch.org/ [last accessed 2019-01-30].

[58] Lalith Suresh Puthalath. Programming the enterprise WLAN: An SDN approach. *Instituto Superior Técnico*, 2012.

[59] Wi-5 - What to do With the Wi-Fi Wild West. URL: http://www.wi5.eu/ [last accessed 2019-07-03].

[60] Mark M. Macomber. World geodetic system 1984. Technical report, DEFENSE MAPPING AGENCY WASHINGTON DC, 1984.

[61] Floodlight OpenFlow Controller. URL: http://www.projectfloodlight.org/floodlight/ [last accessed 2019-07-03].

[62] Raj Jain. Throughput Fairness Index: An Explanation, February 1999. URL: https://www.cse.wustl.edu/~jain/atmf/atm99-0045.htm [last accessed 2019-06-23].

[63] G. R. Hiertz, D. Denteneer, S. Max, R. Taori, J. Cardona, L. Berlemann, and B. Walke. IEEE 802.11s: The WLAN Mesh Standard. *IEEE Wireless Communications*, 17(1):104–111, February 2010. doi:10.1109/MWC.2010.5416357.

[64] Eddie Kohler, Robert Morris, Benjie Chen, John Jannotti, and M. Frans Kaashoek. The Click Modular Router. *ACM Trans. Comput. Syst.*, 18(3):263–297, August 2000. URL: http://doi.acm.org/10.1145/354871.354874 [last accessed 2019-02-03], doi:10.1145/354871.354874.

[65] Get Fedora. URL: http://getfedora.org [last accessed 2019-06-23].

[66] Linksys Official Support - Dual-Band Wireless-N USB Network Adapter. URL: https://www.linksys.com/us/support-product?pid=01t80000003K7bhAAC [last accessed 2019-06-23].

[67] Mini-Circuits. URL: https://www.minicircuits.com/WebStore/dashboard.html?model=VAT-20%2B [last accessed 2019-06-23].

[68] Youssef Saadi, Bouchaib Nassereddine, Soufiane Jounaidi, and Abdelkrim Haqiq. VLANs Investigation in IEEE 802.11 s Based Wireless Mesh Networks.

[69] Eduard Garcia-Villegas, David Sesto-Castilla, Sven Zehl, Anatolij Zubow, August Betzler, and Daniel Camps-Mur. SENSEFUL: An SDN-based joint access and backhaul coordination for Dense Wi-Fi Small Cells. In *Wireless Communications and Mobile Computing Conference (IWCMC), 2017 13th International*, pages 494–499. IEEE, 2017.

[70] iPerf - The TCP, UDP and SCTP network bandwidth measurement tool. URL: https://iperf.fr/ [last accessed 2019-06-23].

[71] Bruno Randolf. “horst”: Lightweight IEEE802.11 wireless LAN analyzer with a text interface - br101/horst, June 2019. URL: https://github.com/br101/horst [last accessed 2019-06-23].