Towards Softwarized Drone Networks

by

Mohannad Alharthi

A thesis submitted to the
School of Computing
in conformity with the requirements for
the degree of Doctor of Philosophy

Queen's University
Kingston, Ontario, Canada
December 2021

Abstract

Drones, or Unmanned Aerial Vehicles (UAVs), are considered essential tools in search and rescue, disaster relief, remote sensing, aerial surveillance and security. Drone-assisted communication networks are gaining considerable attention as a cost-effective and flexible network infrastructure offering new capabilities and opportunities. In addition to enabling connectivity for complex multi-drone tasks, drone networks can be deployed to facilitate connectivity in remote areas and extreme environments and supplement and extend the coverage of mobile networks in response to variable demands. However, utilizing such flexibility requires dealing with the inherent dynamics of drone networks, characterized by a high level of mobility and limited resources. Network softwarization using Software Defined Networking (SDN) and Network Functions Virtualization (NFV) enables flexible and adaptive control and reconfigurability in drone networks through centralized programmability and virtualized network functions.

In this thesis, we investigate drone network softwarization by identifying potential gains from the flexibility offered by softwarization not investigated previously in the context of drone networks. To enable the utilization of SDN and NFV in drone networks, we propose and describe an architecture for softwarized drone networks. Furthermore, we address an important challenge relating to SDN control, a key element in SDN architectures, allowing the programmability of the network through an interface between logically centralized controllers and the programmable network nodes. To adapt to the network mobility and connectivity constraints, we propose schemes for deploying and assigning SDN controllers embedded in drones, allowing for continuous operation of control functions with changing network topology and possible unavailability of ground infrastructure. We also utilize the flexibility offered by NFV. New deployment and orchestration schemes are needed to efficiently deploy and manage drone networks defined by Virtual Network Functions (VNFs) implementing task and network functionalities. To this end, we describe the applicable use cases that benefit from this flexibility and propose schemes that efficiently deploy and manage NFV-based drone networks.

Dedication

To my mother, Fatima, my father, Atiyah, and my wife, Bayan.

Acknowledgments

Praise and thanks are due to God for his countless blessings.

I am most grateful and thankful to my supervisors Prof. Abd-Elhamid Taha and Prof. Hossam Hassanein. Thank you both for the invaluable academic guidance throughout my Ph.D. Your knowledge, ideas, and discussions were inspiring for my work. I cannot put into words my appreciation for your patience and understanding, and for encouraging me to carry on and overcome challenges. I am honored to have been supervised by you, and I will always aspire to your examples in ethics and dedication to your work.

I also would like to thank my supervisory committee members Prof. Juergen Dingel and Prof. Mohammad Zulkernine for their valuable feedback.

I'm thankful to King Abdulaziz University for granting me the scholarship to pursue my graduate studies. I extend my thanks to the Cultural Bureau in Ottawa and to the Faculty of Computing and Information Technology for their support and assistance during my studies. I am thankful to Queen's University, the School of Graduate Studies, and the School of Computing for the opportunity to experience and pursue my graduate studies in an outstanding environment here in Canada.

I am fortunate to have been a member of the Telecommunications Research Lab where I met and worked with many amazing colleagues and friends. To Amir Ibrahim, Dr. Mohannad Alswailim, Dr. Ashraf Alkhresheh, Ahmad Nagib, Dr. M. Adel Ibrahim, Dr. Abdalla Abdelrahman, Galal Hassan, Dr. Wenjie Li, Faria Khandaker, Mohamed Anas, Sherif Azmy, and everyone I met during my time at TRL, my thanks and sincere appreciation are due to all of you for the great company, interesting conversations, and many delightful memories. To Basia Palmer, thank you for your kindness and for all the help you provided me.

To my dear parents, who have endured my long absence away from home, you are always with me no matter the distance. I would not have been able to pursue my dreams in life without the many sacrifices you made. I owe all my achievements to you.

To my loving wife, Bayan, words cannot express my love and gratitude for the endless and unconditional support and encouragement you have given me, even in the most challenging times. My journey here would not have been the same without you in my life.

Mohannad Alharthi
Kingston, Ontario

Co-Authorship

Journal articles:

- M. Alharthi, A. -E. M. Taha and H. S. Hassanein, “Evaluating Softwarization Gains in Drone Networks”. 2021. (in preparation)
- M. Alharthi, A. -E. M. Taha and H. S. Hassanein,“Deployment and Orchestration of NFV-Based Drone Networks”. 2021. (in preparation)

Conference publications:

- M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “An Architecture for Software Defined Drone Networks”. In: *IEEE International Conference on Communications (ICC)*. May 2019, pp. 1–5
- M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Dynamic Controller Placement in Software Defined Drone Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2019
- M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Utilizing Network Function Virtualization for Drone-Based Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2020, pp. 1–5

- M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Evaluating Softwarization Gains in Drone Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2021 (Accepted)

Statement of Originality

I hereby certify that this Ph.D. thesis is original and that all ideas and techniques attributed to others have been properly referenced.

Mohannad Alharthi

December 2021

Contents

List of Tables

List of Figures

Acronyms

3GPP 3rd Generation Partnership Project

4G 4th Generation Wireless

5G 5th Generation Wireless

AP Access Point

API Application Programming Interface

BS Base Station

CPP Controller Placement Problem

D2D Device to Device

FANET Flying Ad-hoc Network

HAP High-altitude Platform

ILP Integer Linear Programming

LAP Low-altitude Platform

LTE	Long-Term Evolution
MANET	Mobile Ad-hoc Network
MEC	Multi-access Edge Computing
NBI	Northbound Interface
NFV	Network Function Virtualization
QoS	Quality of Service
SAGIN	Space-Air-Ground Integrated Network
SBI	Southbound Interface
SDN	Software Defined Networking
SDNC	SDN Control
SDR	Software Defined Radio
SFC	Service Function Chain
SLA	Service-Level Agreement
SP	Service Provider
UAS	Unmanned Aerial System
UAV	Unmanned Aerial Vehicle
UE	User Equipment

UTM	UAS Traffic Management
VANET	Vehicular Ad-hoc Network
VNF	Virtual Network Function
WCPP	Wireless Controller Placement Problem
WSN	Wireless Sensor Network

Chapter 1

Introduction

Drones, or Unmanned Aerial Vehicles (UAVs), are considered essential tools in search and rescue, disaster relief, remote sensing, aerial surveillance and security. One of the goals of using drones is reducing the cost of missions and eliminating risks associated with sending human personnel to conduct hazardous or costly tasks, especially in the case of natural disasters and inaccessible areas. Drones have also been used to conduct and automate tasks such as geographical mapping, environmental monitoring, and industrial inspections [Sha+19]. Drones can be equipped with communication capabilities and deployed as a fleet of cooperating drones to conduct tasks more efficiently and to extend the coverage of communication services.

Networked drones have gained a great deal of attention due to the increasing capabilities and new opportunities they offer. Wireless communication is a vital part of drone missions; apart from command and control, communication is used to coordinate between drones in complex multi-drone missions and to relay information. Ongoing research efforts focus on utilizing the mobility of drones to provide wireless communications to ground-based devices by mounting communication hardware on drones. [GJV16]. . The flexibility and 3-dimensional positioning capability of drones enable their utilization in various networking applications such as sensing, data collection, relaying, and assisting the ground-based network by expanding wireless service coverage. Drone networks are especially beneficial when deployed in temporary situations and in difficult areas as they are considered a cost-effective alternative to deploying fixed infrastructure on the ground. [Moz+19]. Drones can also be equipped with computing capabilities for on-board decision-making and processing mission-related data to deliver real-time results. Such computing power coupled with connectivity increase drone capabilities enabling them to execute more tasks and services.

1.1 Motivation

Utilizing drones for providing network services has several challenges spanning mission planning and deployment, as well as various networking aspects, such as topology control, connectivity, monitoring and management. Drone networks are inherently complex because of their time-varying state and resource constraints. Drone networks have a high degree of mobility and operate in different types of environments and scenarios. [GJV16]. Compared to other wireless networks the resources available to drone networks, such as wireless connectivity, energy, and processing capacity, are limited and prone to malfunctions. [GJV16]. Therefore, techniques for deploying and managing such systems efficiently and adaptively are needed. Research is ongoing in addressing various challenges in drone networks. However, given the complexity of drone networks, facilitating easier development and reconfiguration as well as enabling adaptive control can enhance the network performance as well as the capabilities and agility drone networks [GJV16; Ala+20].

Network softwarization technologies, namely Software Defined Networking (SDN), and Network Function Virtualization (NFV) have been used to address challenges generally prevalent in communication networks. SDN separates the control plane from the data plane by separating the control logic from network devices. The control plane is moved to a centralized controller that configures and programs network devices through a control channel and a unified programming interface. As a result, heterogeneous network devices become easily manageable and programmable. Furthermore, centralized control allows the controller to acquire a global view of the network. Together with the programmability of network devices, adaptive network techniques can be utilized, which allows the network to deal with variable network dynamics. Using NFV, packet processing functions are realized by software implementations known as Virtual Network Functions (VNFs) instead of traditional hardware. Such functions can be instantiated easily on commodity hardware using virtualization technologies, enabling fast, flexible, and cost-effective deployment of services that can be upgraded and scaled as needed. The programming flexibility offered by softwarization allows for accelerated and streamlined development of innovative technologies without the upgrade cost of new hardware.

By adopting softwarization in drone networks, we gain the same qualitative benefits mentioned earlier. We can quickly develop new technologies that utilize the centralized control and global view to employ adaptive control in response to drone network time-variant dynamics and resource constraints. We also gain the ability to easily reconfigure and reuse drones for multiple types of tasks and missions. Functionalities that drones can perform can be implemented as VNFs and reconfigured easily as needed. These aspects allow us to enhance drone networks and enable more innovative applications.

Several works embraced softwarization to enhance network functionality and address the challenges of drone networks mentioned earlier. SDN is generally utilized to enhance network throughput and connectivity by leveraging centralized control and programmability. As well, NFV was proposed to provision reconfigurable softwarized services running on drone-mounted computing resources. In these works, SDN and NFV are considered enablers of the proposed novel applications and adaptive techniques. However, we seek to identify and quantify direct benefits gained from softwarization that justify and support the softwarization of drone networks and enable more innovative applications.

In terms of enabling the SDN architecture in drone networks, a key challenge remained unaddressed in existing works. The centralized SDN control requires ensuring wireless connectivity, which is a challenging aspect considering network mobility. Furthermore, by adopting NFV for reconfigurable drone functions, novel schemes are needed to deploy drone networks and dynamically allocate resources to run interconnected VNFs that implement the functionality of the drone network.

1.2 Research Statement

In this thesis, we aim to identify and demonstrate softwarization gains in drone networks and overcome challenges in enabling softwarization and in deploying and managing such networks. Towards that end, we pose the following research questions:

1. What are the possible gains that can be attributed to softwarization and reconfiguration in drone networks?

2. Given the control plane challenges for software defined drone networks, how can we efficiently deploy airborne network controllers?

3. How can we enable efficient planning and deployment of drone networks with functionality defined by VNFs? and how can we enable the efficient continuous operations of the deployed networks considering mobility constraints?

1.3 Contributions

The contributions of this thesis are the following.

- **Evaluation of Softwarization Gains:** To justify softwarization in drone networks, we investigate the potential gains of softwarization. To do so, we propose a model for evaluating the performance gains of softwarized drones. We model a system consisting of a fleet of drones that conducts multiple tasks with different requirements. We provide an evaluation of the effect of reconfigurability under several scenarios compared to alternative systems with limited or no softwarization.

- **Architecture for Softwarized Drone Networks:** : We propose an architecture to enable softwarization of drone networks. We describe the various components and functionalities needed to enable drone network deployment and reconfiguration and address the limitations of drone network softwarization. The architecture encompasses the following proposed contributions.

- **SDN Control Deployment:** To ensure the availability of SDN control to the network given the rapid mobility of drones and the need to deploy the network to remote areas without access to an SDN controller, we propose a scheme for deploying airborne controllers. The goal is to deploy a limited number of drones as controllers given capacity and communication constraints and to allow certain flexibilities that ensure node-controller and controller-controller connectivity. Furthermore, we expand by providing a dynamic adjustment scheme to allow the airborne controllers to track the drone network nodes changing topology. We enable controllers to do so while limiting their movement to reduce the time and energy required to adjust the network topology.

- **NFV-Based Drone Networks:** The use of NFV to deploy multiple network and processing functions on drone-mounted computing facilities enables a wide range of applications. It translates to several advantages: flexible network planning and deployment, efficient use of resources, and dynamic reconfigurability. This capability is beneficial for tasks in remote areas with no access to computing infrastructure to offload data for processing. We show applicable scenarios that require capturing, processing, and delivering data within different locations in the task area. For such applications, we first express the network or mission functions as Service Function Chains (SFCs) composed of a series of Virtual Network Functions (VNFs) to process and transport network traffic. The contribution here is a joint drone network deployment and SFC placement scheme, to construct a minimal drone network covering the task area and having sufficient resources allocated to accommodate the supplied VNFs and their traffic requirements. To accommodate the mobility of the deployed network, we propose a dynamic orchestration scheme that maintains the network connectivity and SFC requirements as drones move and trigger topology changes. Thus, the scheme maintains the network while limiting disruptive network adjustments and overheads resulting from repeated movement and adjustment cycles.

1.4 Thesis Organization

The remainder of this thesis is organized as follows. Chapter 2 provides a review of drone networks, their applications and challenges. We provide a background of softwarization technologies, namely SDN and NFV, and their general applicability in communication networks. We then review the motivations for softwarization in drone networks. We present an overview of the literature related to the integration of SDN and NFV in drone networks and discuss current limitations. In Chapter 3, we show softwarization gains in drone networks. We propose a model for evaluating softwarization gains in drone networks where drones are required to conduct different kinds of tasks. We evaluate the performance gains due to reconfiguration in a variety of scenarios. In Chapter 4, we describe our purposed architecture for softwarized drone networks that enables programmability and reconfigurability. The architecture consists in part of components that we propose to facilitate aerial SDN control and deployment of NFV-based drone networks. Chapter 5 is dedicated to SDN controller deployment. We describe the challenges encountered when drone networks are deployed in areas disconnected from the ground network controller. We motivate and propose schemes that allow for deploying a number of network nodes as aerial controllers in an efficient and dynamically adjustable manner. We follow with an evaluation of the proposed schemes. In Chapter 6, we focus on incorporating NFV into drone networks. We discuss the motivations and possible use cases which would benefit from such integration. An initial deployment scheme for NFV-based network and dynamic orchestration procedures are described and evaluated. Finally, in Chapter 7, we provide a summary of our work, present our conclusions, and discuss possible future research directions.

Chapter 2

Background

In this chapter, we provide a brief background on drones, their applications in wireless communications, and the associated challenges. We then present an overview of SDN and NFV, and the motivations of their use in drone networks. Next, we review the relevant literature on applying softwarization to drone networks. Finally, we discuss the softwarization challenges that motivates the work in this thesis.

2.1 Connected Drones

Drones are UAVs or aircrafts used in military operations and various civil and industrial applications, such as search and rescue, remote sensing, infrastructure inspection, agriculture, and for providing wireless communications [Sha+19] [Tak+18]. A drone can be described as part of an Unmanned Aircraft System (UAS), where a UAS consists of the aircraft, its payload (e.g., sensors and computing subsystems), a ground control station, and any other subsystems such as communications and support subsystems (e.g., launch, recovery, and transport) [Aus10].

There exist several types of drones. The most commonly used are rotary-wing and fixed-wing drones. Rotary-wing drones such as quadcopters (e.g. [Dji]) have the ability to hover and travel at various speeds and tend to consume more energy. Fixed-wing drones (e.g. [Ebe]), on the other hand, cannot hover but they permit higher speeds and endurance [Moz+19]. There are also hybrid types. Both types come in a variety of sizes, capabilities, weight, speed, range, and flying altitude [Moz+19] and are classified based on these characteristics [HA17]. Rotary-wing drones including quadcopters are the most commonly discussed and assumed in the literature due to their flexibility in deployment and ability to hover. Such drones vary in their specifications. They can weigh less than 1 kg or up to around 20 kg depending on size and payload. Their endurance can range from 25 minutes to about an hour in the more advanced models [HA17; Mar]. On the other hand, fixed-wing drones can fly for several hours depending on their size, weight, and whether they are powered by battery or fuel [Moz+19]. Additional aircraft types such as High-altitude Platforms (HAPs) and Low-altitude Platforms (LAPs) including airships [Itu] and balloons [Loo] are also considered for applications of aerial networks. Such platforms are deployed to fly at higher altitudes. LAPs are more flexible in terms of ease of deployment and mobility and can be deployed at altitudes of a few kilometers. On the other hand, HAPs, are deployed at altitudes over 17 kilometers and are considered quasi-stationary [Moz+19].

In addition to using drone networks to support multi-drone tasks, the use of drones to provide wireless communication services is one of the main applications recently being pursued [Sha+19]. This is due to the new opportunities they provide and due to their low cost and positioning flexibility compared to a fixed infrastructure. Drone networks can be deployed as airborne Base Stations (BSs) for providing and expanding cellular coverage to mobile users [Cic+19] , as communication relays [ZYS11], and for sensing and information collection. Using drone networks to assist in coverage is a key advantage as drones can be deployed and positioned as needed for variable and temporary demands such as in crowded outdoor sporting events and crisis situations [Kal+17; Erd+17; Mal+19].

The wireless technologies that can be utilized by drones include the current existing technologies such as Wi-Fi and Long-Term Evolution (LTE). Experiments where a networked fleet of drones utilize Wi-Fi technologies to establish connectivity among other drones within the fleet have shown promising results [Gu+15; JP+12]. Drones can also utilize available cellular networks by connecting to LTE and 5th Generation Wireless (5G) networks as User Equipment (UE) [ZZL16b]. Support for drone connectivity in LTE and 5G was introduced in 3rd Generation Partnership Project (3GPP) Releases 15 through 17 [3GP19]. Moreover, technologies such as Device to Device (D2D) [ZZL16b], mmWave, and Software Defined Radio (SDR) have been proposed for use by drones [LFZ19].

In terms of network structure, drone networks can utilize both infrastructure-less (ad-hoc) and infrastructure-based networking. Given the dynamic nature of drone networks, often a form of Mobile Ad-hoc Networks (MANETs), known as Flying Ad-hoc Networks (FANETs) [BST13] is used. Similar to MANETs, in FANETs, nodes (drones) join and leave the network dynamically and form links with neighboring nodes. Nodes also act as relays to forward traffic across the network. The network logic in MANET is distributed among nodes, as there is no network infrastructure. FANETs are differentiated from MANETs based on the mobility rate of change and continuously changing topology as a result. Infrastructure-based networking is also utilized in drone networks, where drones either communicate with each other through cellular BSs or a ground control station. Hybrid architectures are also possible [GJV16]. The choice of architecture depends on factors such as the communication technology used, the application domain requirements, and the type of control required for networking and task coordination, which can be centralized using a control station or distributed among drones.

Drones can also be equipped with on-board computing resources such as single-board computers for flight control, and for executing algorithms, machine learning models, and specialized programs for conducting autonomous tasks [Wan+20b]. The Raspberry Pi is an example of a single-board computer commonly used with drones [Nog+18]. Examples of more capable computing systems include the Manifold 2 [DJI21] and NVIDIEA Jetson X2 [Nvi].

2.1.1 Applications of Drone Networks

Utilizing drones for communication and networking for various applications is studied extensively in the literature. Here we provide the most notable and representative examples of current works. Figure 2.1 depicts examples of drone-based or drone-assisted communications.

A number of works discussed the deployment and formation of flying network infrastructures. In [CS17], authors propose deploying a drone-based multihop backhaul network to connect ground BSs to the gateway to allow BSs to continue operating during an outage. An aerial network formation scheme based on game theory is proposed, with the goal of achieving the best network delay and data rate per drone. The scheme achieves better network performance compared to other approaches without

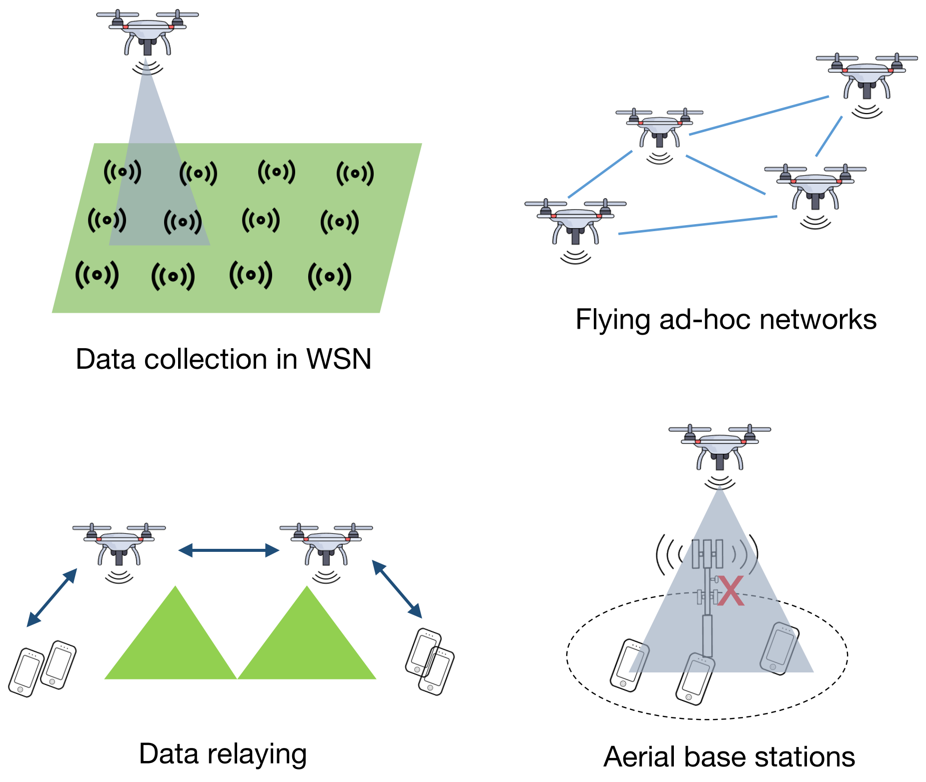


Figure 2.1: Examples of using drones in wireless communications

network formation among drones and instead using a direct connection to the gateway. A related proposal is made in [Par+16] in which authors propose a network formation algorithm for deploying drones forming a multihop ad-hoc network in 3-dimensional space. The algorithm enhances network provision compared to other formations with uniform altitudes. In [Zha+19], authors propose a framework for drone-assisted emergency wireless networks to serve disaster-affected users in several scenarios. In one scenario, a drone is utilized to serve users in an area with a damaged BS. The drone trajectory is optimized to enhance Quality of Service (QoS) of served users while limiting interference to users served by other BSs. When all ground BSs are damaged, the drone establishes multihop connectivity using D2D communication among ground devices using combined optimal drone transceiver design and allocation of D2D links. As well, a stationary multi-drone relaying system is proposed to link the serving drone to the core network. The proposed framework compares the effect of different network parameters on performance in terms of the total network data rate and outage probability.

Using drones to form relay links between distant locations has been one of the most popular applications of drones in wireless communications [ZYS11; HSL09; ZZL16a]. The flexible mobility of drones is leveraged to expand the range of wireless connectivity in terms of reachability and data rate. In [HSL09], authors propose algorithms for optimizing the location and movement of drones to improve the connectivity or reachability of MANETs. The paper demonstrated how optimal trajectories can improve different types of connectivity such as global message connectivity, worst-case connectivity, and k-connectivity. In [ZZL16a], a drone relaying system is proposed for relaying messages between two distant fixed wireless nodes. The system jointly optimizes the drone trajectory and transmit power to maximize the system throughput, achieving better performance than a fixed relay node. Recent efforts focus on utilizing multiple drones to form relay chains to link distant locations while aiming to minimize the number of drones needed. However, others aim to enhance network performance or mission objectives. For instance, authors of [Yan+19] utilize drones to form relay links between distant drones performing tasks and a gateway node. Authors provide a scheme for relay drones to autonomously organize and form relay links. The scheme's objective is to minimize the number of relay drones. The work proposed in [Ols+10], provides a scheme for connecting drones performing monitoring tasks in areas that do not allow for line-of-sight links to the BS, such as mountainous and urban valley areas. The proposed scheme can form multihop relay chains and relay trees while balancing trade-offs between the number of drones, the quality of formed links, and the quality of sensed information (surveillance). For example, changing a drone location to form a relay chain can affect the quality of images captured by the drone, leading to lower surveillance accuracy.

Drones are also utilized as airborne data collectors to efficiently collect data in resource-constrained Wireless Sensor Networks (WSNs). In drone-assisted WSNs, drones fly near sensor nodes (SNs) to collect data wirelessly. Drone trajectories are optimized jointly with the transmission scheduling of drones and SNs. In this manner, data can be collected quickly while SNs consume less energy compared to traditional WSNs due to the reduced transmission power and time needed to collect data [Say+16]. The work in [ZZZ18] is an example of such a technique that demonstrates enhanced energy efficiency compared to stationary data collection.

A number of articles proposed using drones to deploy caching and computation offloading services to ground users. In such applications, drones utilize drone-mounted computing devices known as micro edge devices or cloudlets. The goal is to offload computations from power-constrained ground devices and minimize delay for time-sensitive tasks by deploying drones near the users. The work in [Che+17] develops a proactive deployment of cache-enabled drones. The objective is to improve the users' quality of experience using a predictive scheme based on information collected about served users. In [JSK18], a drone with mounted computing resources, called a cloudlet, provides computation offloading for ground devices to minimize their energy consumption and improve QoS. Path planning and resource allocation are considered while optimizing for drone energy constraints.

One of the most promising uses of airborne platforms is their utilization as aerial BSs or drone-BSs in 5G networks. Drone-BSs are used to increase the capacity and coverage of the cellular network and relieve congestion in response to variable and unexpected demand, such as crowded events. They are also used to expand coverage in areas with low or nonexistent coverage and to restore connectivity in disaster recovery scenarios. A great deal of flexibility is added to the cellular network infrastructure due to the dynamic positioning and ease of deployment of drone-BS . 3GPP Release 15 introduced support of non-terrestrial 5G networks involving satellite and aerial access services [BY+19]. Different from network topology formation works discussed earlier, drone-BSs establish connectivity to the terrestrial network through wireless backhaul links to ground BSs or backhaul nodes. Drone-BSs can also be tethered to ground nodes to connect to the core network. In the literature, the optimal deployment of drone-BS is the subject of many studies. The objective of such studies is to determine the optimal 3D placement of drone-BSs according to certain performance metrics. The literature is mainly classified according to whether a single drone or multiple drones are used and the objectives of coverage. Objectives include maximizing the number of covered users, increasing profit, maintaining or enhancing users data rate and other QoSs metrics, and reducing the energy consumption of drones. The literature can be also classified based on whether the proposed schemes are for instantaneous (static) state or dynamic coverage, where drones move in response to variable demands over time. The survey provided in [Cic+19] lists the state-of-the-art in this area.

While most deployment techniques focus on deploying a network for single tasks, the work in [Mal+19] proposes deploying a multi-service drone network for disaster situations. Different from works described earlier, the authors propose a task scheduling scheme, utilizing drones with a set of sensors or payloads each corresponding to a task needed in disaster recovery. The paper studies the benefits of equipping drones with single or multiple payloads (e.g., radio, camera, and medicine) for performing network coverage, video monitoring, and medicine delivery tasks. Their assessment showed that assigning multiple tasks to drones can lead to reducing the number of required drones if tasks are allocated optimally.

2.1.2 Challenges

Drone networks are characterized by their high degree of mobility and fast-changing topology. This is an advantage as it allows networks to reorganize and conduct tasks along varying trajectories. However, it is also a challenge in terms of managing the network and ensuring connectivity in the presence of intermittent links and frequent topology changes [GJV16]. Furthermore, drone networks are constrained in terms of communication, energy, and computing resources compared to traditional fixed networks [GJV16]. As well, drones can be of varying hardware, software, and network technologies. All of those aspects combined complicate the management and operation of such highly dynamic networks and limit their performance [Sha+19]. These challenges motivate the need for innovative and unified management technologies that can simplify the network operation and allow for highly adaptive and informed network techniques. Softwarization technologies such as SDN and NFV can facilitate such dynamic approaches and simplify the management and control in drone networks. In the next sections we introduce network softwarization and we discuss the advantages and applicability of SDN and NFV to drone networks.

2.2 Network Softwarization

Network softwarization is a network paradigm in which the design, architecture, deployment, and management of network components are based on software [Afo+18] rather than traditional hardware. Software programmability then is leveraged to make networks flexible and reconfigurable by means of software updates. This flexibility allows softwarized networks to be dynamically reconfigurable on-demand in response to varying requirements and conditions. The enabler technologies of network softwarization are SDN and NFV.

2.2.1 Software Defined Networking (SDN)

In wired networks, SDN addresses the limitations caused by the coupling of networking devices (e.g., switches and routers) and the implementation of network functions or protocols. Traditionally, these functions are implemented through hardware, which allows fast packet processing. However, this constrains the rapid development of new protocols and hinders their adoption, requiring long development cycles and costly hardware upgrades. In addition, network devices use proprietary interfaces for configuration and management, which complicates the management of large-scale networks with multivendor devices.

The above limitations motivated the need for flexible and programmable networks. Various efforts in programmable networks led to what is currently known as SDN [Nun+14]. Initially, SDN referred to the OpenFlow project [McK+08] which began in 2011. Presently, SDN refers to any network architecture that adopts SDN principles, which are:

1. the separation of the data (forwarding) plane and the control plane,

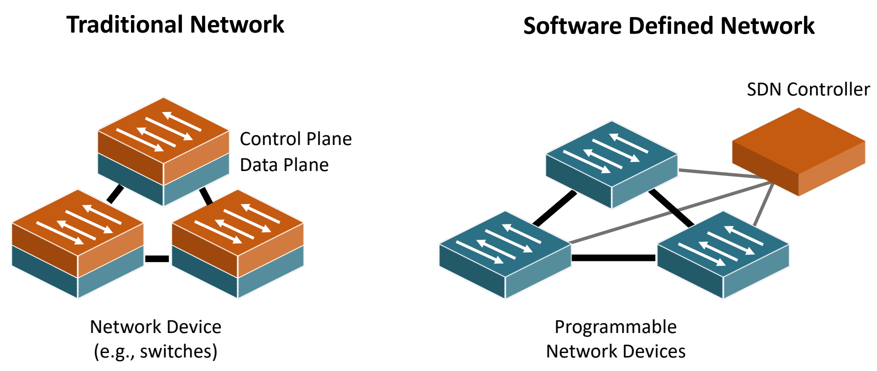


Figure 2.2: Data and control planes in traditional networks and in SDN

2. the logical centralization of the control plane,

3. the ability to program network devices, and

4. managing diverse network devices in the data plane using a unified interface [Kre+15].

In SDN, the handling and forwarding of packets (the data plane) is separated from packet forwarding decisions (the control plane) as demonstrated in Figure 2.2. Network devices are turned into simple but flexible devices that expose a unified programming interface. Instead of operating independently with hardcoded functionality, network devices are instructed by centralized software controllers known as SDN controllers. SDN controllers implement the logic of the network and dynamically direct how network devices process and route packets. Or more precisely, the internal flow tables in network devices are manipulated by instructions received from SDN controllers.

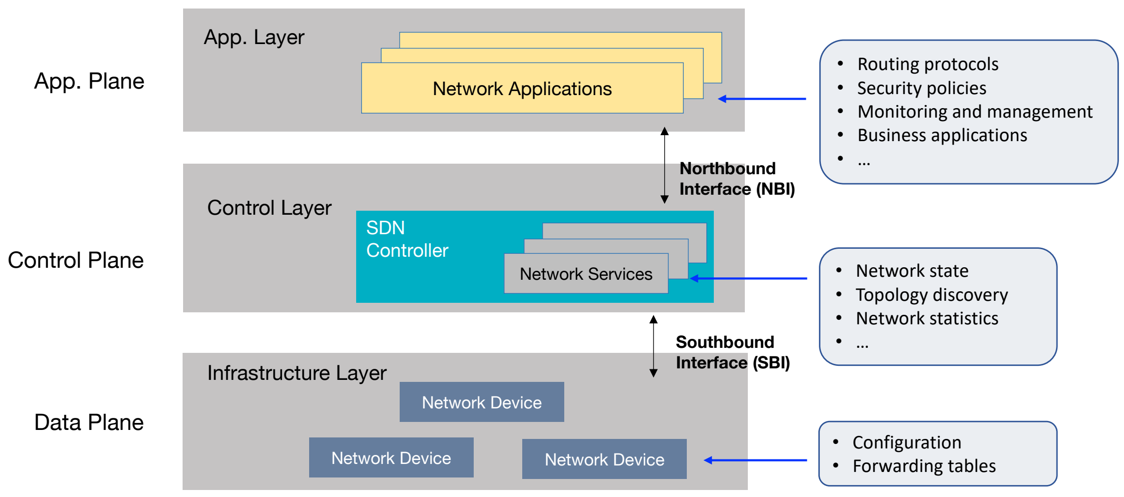


Figure 2.3: SDN architecture [Onf]

SDN Architecture

The typical SDN architecture is shown in Figure 2.3. The architecture consists of the infrastructure layer, the control layer, and the application layer. The infrastructure layer consists of the programmable network devices representing the data plane. Network devices are programmable by the SDN controller in the control layer through a unified interface known as the Southbound Interface (SBI). OpenFlow [McK+08] is the most common standard interface.

The control plane consists of a centralized SDN controller that implements the control logic of packet forwarding and programs network devices by manipulating their configurations and forwarding tables. The SDN controller communicates with network devices using the SBI via a control channel. The controller, given its centrality and knowledge of network devices, maintains a global view of the network state which includes the connected devices, the network topology, flows, and link statistics. The controller also provides a set of network services that implement network functions such as topology discovery. The controller abstracts the network and provides facilities and a high-level programming interface to network applications (or SDN applications). This interface is known as the Northbound Interface (NBI).

Network applications operate at a higher level than the control layer. Such applications include network monitoring and management applications as well as advanced adaptive network protocols and policies. Such applications utilize the global network view and services provided by the controller to inform their decisions and react to network conditions. In turn, the controller compiles requests from applications and manipulates network devices to update the data plane behavior.

As the SDN controller is logically centralized, a single or multiple distributed controllers can exist and work in conjunction to control the network and maintain an up-to-date global state. SDN applications can also be distributed and communicate with the controller to program the network.

Using the SDN architecture, network devices are not tied to hardcoded protocol implementations, as network devices can operate as programmed by the controller and SDN applications. Due to this separation, innovation is accelerated, since SDN applications can implement experimental and innovative protocols while using commodity hardware. Developed protocols and applications can dynamically adapt the network as needed using the facilities provided by the controller and the unified interface regardless of the underlying network hardware. Thus, operational costs are reduced due to the lower cost of equipment and the simpler effort required to operate and manage the network.

OpenFlow

OpenFlow is a standard that defines the specifications for programmable network devices (known as OpenFlow switches) and the protocol that allows for the interaction between the controller and programmable switches. The standard is maintained by the Open Network Foundation (ONF) [Ope15]. The standard specifies the components or abstractions that OpenFlow switches must implement to enable their programmability. The main OpenFlow abstraction is the flow table, which processes and forwards packets incoming to the switch according to rules installed by the controller. Figure 2.4 demonstrates the flow table structure. Each table entry specifies the actions applied for traffic flows with packet header content matching specified values. Actions applied on matched flows involve rewriting headers, dropping packets, and forwarding packets through specific ports, or sending packets to the controller for further processing. Each entry has a set priority if more than one entry has matching header values. Per-flow statistics are also collected. The flow table represents the basic functionality of the switch, which enables flexible data plane programmability. OpenFlow provides a sophisticated programmable packet processing pipeline in which packets can be handled by multiple flow tables for multi-stage processing. Apart from switch specifications, the standard defines the OpenFlow protocol with which control messages and state queries are exchanged between switches and the controller. Such messages handle switch configuration, rule installations, and statistics polling to name a few. The OpenFlow protocol is also extensible allowing for customized control messages and actions.

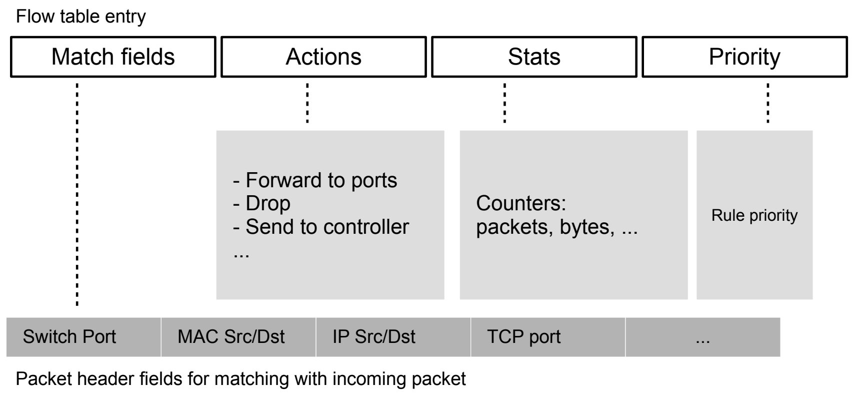


Figure 2.4: Flow table structure in an OpenFlow-based switch

2.2.2 Network Function Virtualization (NFV)

NFV [Han+15] intends to reduce the cost of network infrastructure by replacing hardware network appliances, such as routers, firewalls and load balancers by softwarized implementations known as Virtual Network Functions (VNFs). VNFs run on virtualized general purpose high-capacity servers as virtual machines, as opposed to specialized hardware appliances. The aim of network operators in using NFV is similar to SDN, which is reducing costs in terms of capital expenditures and operating expenditures.

By separating network functions from specialized hardware, greater scalability and flexibility is achieved in addition to reduced costs as illustrated in Figure 2.5. Functions run as virtual machines and can be deployed, instantiated, and migrated on available hardware resources. Network operators also benefit from automated provision of such functions or services. Functions can also be updated and improved as often as required using software updates.

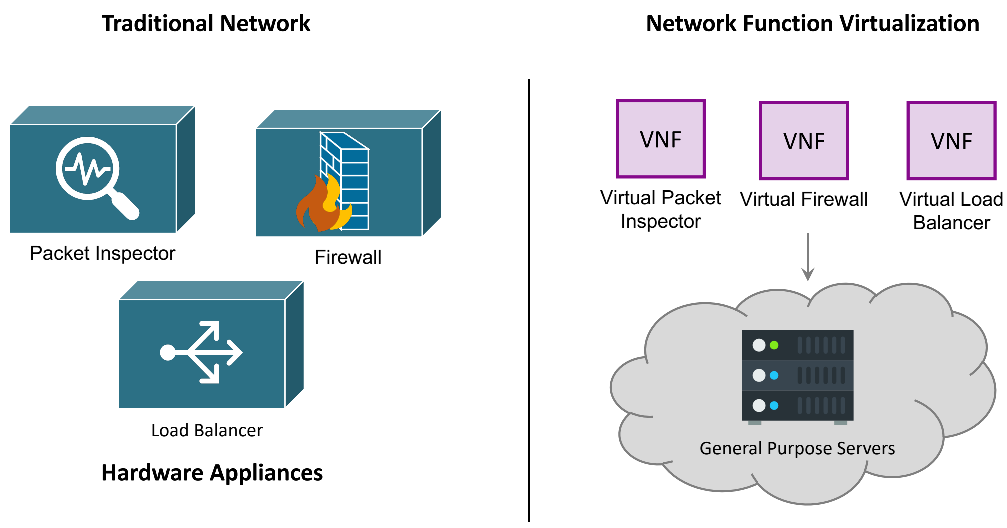


Figure 2.5: VNFs as an alternative to hardware appliances

In wired networks, NFV operates within an architecture standardized by ETSI [Ets]. The simplified architecture is shown in Figure 2.6 and are described at a higher level as follows.

- **NFV Infrastructure (NFVI):** consists of hardware resources and software components that together represent the physical infrastructure in which VNFs are deployed and managed. Hardware resources include commodity computing and networking devices. The virtualization layer abstracts the underlying hardware resources and provides virtualized computing and networking resources used by VNFs.

- **NFV Management and Orchestration (NFV MANO):** responsible for the overall management and provision of network services and constituent VNFs. It consists of the following components:

  - **NFV Orchestrator:** responsible for orchestrating the lifecycle of network

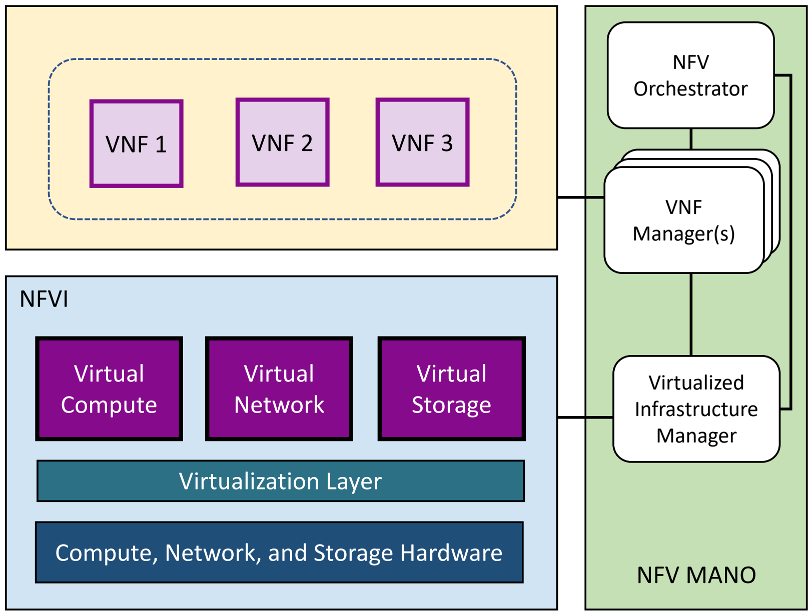


Figure 2.6: NFV architecture [Ets]

services by coordinating with the following two components.

– **VNF Manager:** responsible for the lifecycle management of individual VNFs (e.g., instantiating and terminating VNFs).

– **Virtualized Infrastructure Manager:** responsible for managing the NFVI and allocating resources for VNFs.

Based on this architecture, different network services can be provisioned by network providers according to Service-Level Agreements (SLAs) with greater deployment flexibility and less cost compared to hardware-based equipment. A service can be composed of a series of ordered VNFs that process network traffic as mandated by SLAs. Such a service is also known as a Service Function Chain (SFC) where traffic is processed through a predetermined sequence of VNFs. In the simplest form, an SFC can be represented as a line graph such as:

$$\text{Firewall} \rightarrow \text{Deep Packet Inspection} \rightarrow \text{Cache}$$

A service has a set of requirements in terms of traffic throughput and delay that must be satisfied by the service provider.

It is important to note that SDN and NFV are similar but complementary technologies. SDN is concerned with the configuration and reprogramming of the network forwarding plane, whereas NFV is concerned with replacing network appliances with softwarized alternatives via virtual machines. Both technologies can coexist in a single network where VNFs implement various networking functions traditionally performed by specialized network devices and application servers, while SDN controls the underlying forwarding and routing in the network. The NFV orchestrator can be considered as an SDN application that programs the network to route traffic through VNFs according to service requirements.

2.2.3 Softwarization in Wireless and Mobile Networks

While softwarization is used primarily in wired networks, softwarization also has been applied in wireless and mobile networks. The use of SDN in wireless networks enables uniform management and configuration and contributes to reducing cost and adding flexibility. Since the traditional design of SDN was intended for wired networks, several efforts were made to extend SDN abilities in several types of wireless networks [Cos+12]. SDN can be utilized to facilitate centralized QoS management in heterogeneous networks by selecting radio access technologies for users in order to eliminate congestion [Ras+17]. SDN can also be utilized for channel selection in Wi-Fi access points to minimize interference [Sey+16]. More advanced wireless SDN applications include an SDN-based wireless backhaul for small cells to facilitate global network monitoring and online optimization against interference [Hur+15].

In the context of MANETs, SDN is utilized to enable centralized and adaptive routing and failure recovery. The work described in [YQR17] addresses practical implementation aspects in medium access control to enable SDN in wireless multihop networks. Results with a small test topology show that centralized routing can maintain throughput in cases with fast-changing topology and unexpected link interruptions. SDN also is proposed for Vehicular Ad-hoc Networks (VANETs) to implement routing protocols informed by the SDN controller to select reliable links and ensure QoS requirements for the networks of moving road vehicles.

Softwarization using SDN and NFV has also been considered in cellular networks to handle the explosive growth of data traffic and types of services provided and to enable cost-effective and flexible management and scalability. Both SDN and NFV are proposed for the management and virtualization of the Evolved Packet Core (EPC), the core network of the LTE system [Bas+13; Jai+16]. Softwarization is presently considered the basis of 5G networks spanning both the core network and the Radio Access Network (RAN). Using softwarization, service providers can deploy multiple virtualized networks known as network slices on top of common infrastructure. Network slices are programmable and automated networks that are tailored for specific services and requirements [Afo+18].

2.2.4 SDN and NFV for Drone Networks

Softwarization and its advantages have great appeal to wired network providers. However, given the dynamic nature of drone networks as discussed in Section 2.1 and the challenges discussed in Section 2.1.2, softwarization advantages are more beneficial for drone networks.

Using SDN, drones become the programmable nodes of the network. Management, and command and control of drone networks can benefit from a unified programming interface [Sha+19]. Innovative networking schemes and protocols can be rapidly developed and utilized rapidly. Dynamic and adaptive networking can be easily facilitated using the SDN centralized control and visibility to adjust the network behavior in response to rapidly changing events and requirements [Ala+20; Sha+19]. This enables the network to manage uncertainties in terms of link failures and unavailability of resources as well as time varying demands and conditions.

For instance, drones deployed as a multihop ad-hoc network may experience intermittent links due to mobility, interference, and malicious jamming [Seç+18a]. By utilizing SDN and the controller visibility into drone locations and environment conditions, the routing paths can be updated accordingly. SDN can facilitate proactive centralized routing using prior knowledge of drone trajectories to predict link failures [Iqb+16; CBM17]. The same approach is considered for VANETs, which are characterized by highly mobile nodes [CBM17; GK18; Gha+19]. A number of works demonstrate that routing performance is improved using centralized SDN [CBM17]. Even with the availability of cellular networks, drones equipped with cellular connectivity can experience connectivity issues in some environments when cellular coverage is not optimized for drone altitudes [Tak+18]. Drones equipped with multiple radio access technologies can be dynamically programmed to use alternative connections. In Section 2.2.4, we describe research efforts that apply SDN in drone networks.

Furthermore, the use of NFV enables reconfigurability by utilizing drone-mounted computing resources to run VNFs that implement a multiplicity of network and task-related functions. Using NFV, a drone network can be considered as an airborne physical network infrastructure where network services provisioned as SFCs can be embedded within the drone network in an automated and dynamically reconfigurable manner. Such capabilities can be utilized in drone networks supporting ground users or assisting ground network infrastructure. Additionally, VNFs can be utilized to define tasks performed by drones, where VNFs implement various algorithms used to perform certain types of tasks such as area mapping and agriculture monitoring. This capability also allows the behavior of a drone to be dictated by VNFs. Functions or the type of tasks performed by a drone can easily be configured at deployment time and dynamically adjusted during operation. Such modular implementation of functions allows for easy composition of missions or services using manageable and upgradeable components. In dynamic scenarios with time-varying requirements, VNFs can be instantiated dynamically, enabling more scalability and reconfigurability. This flexibility can also facilitate easier recovery in the case of drone failures or malfunctions, as network or task functions lost due to drone failures can be reinstantiated quickly in another drone. In Section 2.2.4, we review works that adopt NFV in drone network scenarios.

SDN in Drone Networks

The adoption of SDN in drone networks focused on optimizing network forwarding and connectivity by utilizing the SDN controller visibility. In [Seç+18a], authors provide a software defined drone-based network architecture for reliable connectivity in drone networks. In this architecture, drones are assumed to have multiple network interfaces, such as Wi-Fi and LTE. Drones act as SDN forwarding nodes, while the SDN controller implements a multi-path disjoint routing protocol to avoid malicious link jamming by utilizing the multiple interfaces in each drone and the visibility given by the SDN controller. The authors extend their work in [Seç+18b]. The routing scheme focused on maintaining connectivity (reachability) and reducing the outage rate due to jamming. The proposed routing protocol was only compared to a centralized shortest path routing protocol and mobility was not considered. In [Seç+20], authors propose a drone mesh network architecture with fog computing capabilities to provide computation offloading to ground devices. In this architecture, computation requests made by ground devices are offloaded to either drones (called fog nodes) or a server on the ground. A joint optimization framework is proposed to make optimal allocations of network flows and computation the assignments with the goal of optimizing the network latency or computational response of the system. The system demonstrates improved latency and balanced task allocation across the network. The system is compared to greedy and static allocation schemes. A small scale test bed implementation demonstrates similar results to simulation. In [ZWZ18], a framework for software defined drones with mounted Wi-Fi Access Points (APs) is proposed. The SDN controller monitors the network traffic load and battery status of drones, with the goal avoiding congestion by balancing the traffic load among the network of APs serving ground mobile users. The paper demonstrated the ease of implementing load balancing using the SDN controller. In [Bar+17], the authors discuss connectivity and routing challenges in drone networks, including HAPs and balloons such as the platforms used in Project Loon [Loo]. The authors make the case for the centralized SDN control for predicting and mitigating link interruptions due to network dynamics in time and space. Details of their SDN implementation are presented in [BC18], however, only qualitative findings are discussed with respect to SDN . A similar theme but on a smaller scale is discussed in [Iqb+16]. The SDN controller predicts link disconnections and recomputes routing paths to improve end to end-to-end availability compared only to the Open Shortest Path First (OSPF) routing protocol. SDN has also been utilized in drone networks involving military operations [Zac+17], where multiple drones are employed to monitor areas ahead of advancing troops and stream videos to the ground. An SDN controller on the ground manages the network and deals with drones' mobility and ground units. The ability of the controller to discover the network topology allows it to optimize the delivery and quality of video streams from drones to the ground. In [Sha+17], drones are deployed as drone-BS with SDN forwarding capabilities to facilitate fast handovers in cellular networks. Both centralized and distributed SDN controllers are used to inform handover decisions to reduce end-to-end delay, handover delay, and signaling overhead. The paper demonstrated enhanced performance to other SDN-based handover schemes.

NFV in Drone Networks

The use of NFV in drone networks has been explored only recently. NFV has been considered to offload drone computation tasks to a virtualized ground infrastructure and also to provide processing functions on drone-mounted computing resources.

In use cases with available access to an infrastructure, VNFs can be hosted in the cloud or at the ground control station of the drone mission. In [BBT19], authors propose using the NFV architecture to deploy VNFs that communicate with drones to provide certain services. VNFs are deployed in Multi-access Edge Computing (MEC) [MB17] nodes placed at various locations at the edge of the network. An orchestration scheme places VNFs in optimal edge nodes given learned drone trajectories to meet the QoS required by drones to communicate with VNFs. The use of NFV in reconnaissance and military-related operations is presented in [Whi+17]. The authors describe a system that uses container-based (light-weight virtual machines) VNFs deployed at the network edge on the ground or in control stations to support flying drones. The VNFs perform mission monitoring and anomaly detection. The authors reported the ease of use and flexibility provided by the SDN/NFV system to support missions with demanding mobility requirements. Using the proposed system, VNFs are migrated across different ground stations according to drone trajectories. As well, VNFs that perform specialized functions can be deployed on demand for special drone types and situations.

Other works considered hosting VNFs on-board drone computing resources. Authors in [RS17] describe an SDN/NFV-based architecture for drone networks for rural zone monitoring. The drone network provides video monitoring as a service where cameras on the ground and on drones capture footage of monitored rural areas and stream captured footage to users. The network consists of backbone drones as the nodes of the network along with SDN/NFV controllers and orchestrators. The VNF chains required for the service include video transcoding, storage, and streaming. The authors only provide a model for analyzing the computational load on network nodes based on factors such as the number of video processing functions and traffic flows.

In [Nog+18], a configurable NFV-system for multi-drone services is proposed. The system enables dynamic reconfiguration of drones for different missions. The

VNFs are deployed on drones with Raspberry Pi boards as computing resources. Drones form a wireless ad hoc network, and the NFV aspects are managed by an orchestration system in the ground station. A prototype implementation is described and validated using several field experiments that measure the time and delay of VNF instantiations and their effect on running functions. A Voice over IP (VoIP) service was also demonstrated.

Recent proposals focused on integrating SDN/NFV in Space-Air-Ground Integrated Networks (SAGINs). SAGINs are networks that integrate satellites and aerial networks with terrestrial networks to increase coverage [Liu+18]. Both SDN and NFV have been proposed to manage such heterogeneous networks. Recent articles considered provisioning network services as SFCs to serve users and ground vehicular networks. SAGINs receive requests to deploy services as SFCs composed of a series of VNFs. VNFs of accepted services are instantiated on ground BSs and on virtualized aerial nodes (drones and LAPs or HAPs) if needed to satisfy service requirements [Wan+20a; Li+21]. In [Wan+20a], authors propose an SFC placement scheme to increase the number of accepted services and thus increase revenue while satisfying QoS requirements mandated by SFCs. The scheme is evaluated using a fixed network topology and compared to other SFC placement schemes to demonstrate optimality in terms of total revenue. In [Li+21], authors consider SAGINs for supporting ground vehicular networks. The authors introduce an online SFC placement scheme that continuously adjusts and reschedules the placement of VNFs to accommodate SFC requirements and to accept new requests to increase revenue.

2.3 Limitations of SDN and NFV in Drone Networks

In this section, we look into some limitations in the existing literature on SDN and NFV in drone networks.

Benefits of Softwarization: The use of softwarization in drone networks is motivated by programming and configuration flexibility and the ease of control expected from softwarization. The centralized control and the visibility into the network state can be considered as an appealing tool for developing adaptive network solutions to deal with dynamic networks. Benefits such as application programming interfaces (APIs), visibility into network state, modularity, and reuse and upgradability of existing hardware are considered qualitative benefits. However, additional measurable benefits can be valuable, providing more insight and motivation for softwarization.

The works that utilized softwarization in drone networks as reviewed in this section mostly argued for softwarization for its qualitative benefits. As for their qualitative results, the focus of such works was on demonstrating the feasibility of softwarizing drone networks and achieving certain goals for the targeted networking domains. For example, the utilization of SDN to enhance routing made use of centralized control and prior knowledge of drone trajectories to proactively program the network and avoid or limit network performance degradation or interruptions (outages). In [Seç+18a; Seç+18b], resilient routing was only compared to centralized routing algorithms (SDN-based). In [Iqb+16], comparisons were made against a single traditional routing protocol. Similarly, in the context of NFV, proposals such as [RS17] did not provide comparisons over non-softwarized drones. Works such as [Wan+20a; Li+21] utilized the reconfigurability of both aerial nodes (drones) and ground nodes to adjust provisioned services (VNFs or SFCs) to accommodate new traffic demands. Their orchestration schemes were compared against other SFC orchestration schemes. While softwarization is an enabler for the proposed techniques in those studies, such studies do not provide direct measurable benefits for softwarizing drone networks. Different from traditional networks, drone networks are characterized by the additional functionalities or tasks they perform. As discussed in Section 2.2.4, softwarization brings flexibility to this aspect as well in terms of the ability to reconfigure drone tasks. The benefits of softwarization and reconfigurability have not been studied. Therefore, further studies are required to measure the gains of softwarizing drone networks.

SDN Control Connectivity: The controller is the key component in the SDN architecture as the controller is responsible for implementing the network logic and programming network devices. In such an architecture, all network devices establish control channels with the SDN controller, or in the case of multiple distributed controllers, each controller is assigned a subset of network devices to control. Controllers communicate with network devices to program the data plane and to query the state of network devices. Due to the role of the SDN controller, the network functionality depends on the existence of a control channel between SDN controllers and network devices.

SDN control connectivity is a key challenge in adopting SDN in drone networks, where drones represent the programmable network nodes. The challenge lies with ensuring control plane connectivity through wireless links given the variability of network topology. Existing works reviewed in this chapter assume the availability of an accessible ground infrastructure with SDN controllers. In SAGINs, SDN controllers can be located in satellites or HAP nodes which provide coverage using low-altitude drones. However, in some practical use cases, drone networks are deployed in remote areas with limited or no access to ground SDN controllers. Satellites and HAPs may not be available. In such cases, the availability of SDN control links is critical and challenging as the network is dependent on the controller during the operation of the network. To address this challenge, techniques are needed to ensure that persistent control channels and pairing can be established between SDN controllers and drones. A possible approach is deploying drones that function as SDN controllers along with the network. Considering the network mobility, the size of the area covered by the network, and the range control links, dynamic schemes for controller deployment and assignment are needed.

NFV-based Drone Network Deployment and Orchestration: Drone networks with on-board computing capabilities can be utilized in various drone use cases. The computing power within the network offers the flexibility to implement task-related computations involving data acquisition through sensing or imaging and real time processing and analysis. This is useful for tasks in remote areas with no access to ground computing infrastructure and to provide network services to end users served by drones (e.g., computation offloading and caching). As discussed earlier, NFV can be utilized to implement such functions as modular components configured on drones; however, deployment and orchestration techniques are needed.

While several works reviewed in this chapter demonstrated the use of NFV including SFCs in drone networks, these proposals lack orchestration schemes that also deal with the variability of drone networks. The works in the context of SAGINs proposed orchestration schemes, however, the topology is dependent on ground nodes, and the aerial segment is considered fixed or not controllable by orchestration schemes.

For applications such as remote sensing and monitoring, drone networks can be comprised mainly of aerial nodes. In such settings, the network size differs depending on the task at hand and the topology of the deployed network as it varies over time. Given the flexibility of NFV, a drone mission or deployment can be specified in terms of SFC comprising VNFs that implement the different network and mission functions. Then, it is important to determine the physical network topology that needs to be deployed in addition to the allocation of SFCs within the deployed network. To the best of our knowledge to date, no works exist that jointly determine the physical network topology and SFC orchestration.

Furthermore, the variability of network topology poses an additional challenge. Due to drone movements and topology changes, often SFC reconfiguration is required. With this in mind, dynamic network orchestration schemes are needed to maintain the connectivity of the physical network as well as the deployed SFCs. However, frequent reconfigurations may disrupt network traffic. Therefore, orchestration schemes are needed to deal with such dynamics.

2.4 Summary

In this chapter, we presented an overview of connected drones and their applications in communication networks. We followed by discussing the challenges of drone networks. Next, we provided an introduction to network softwarization enabled by SDN and NFV and described the flexibility it introduced to different networking paradigms. Then, we detailed the case for the softwarization of drone networks and reviewed the literature that incorporated softwarization in drone networks. Lastly, we highlighted the current void in the literature and provided a more detailed motivation for this thesis. To this end, we discussed the need for assessing the direct softwarization gains for drone networks. Furthermore, we highlighted the limitations of enabling SDN and NFV in drone networks, which included an unaddressed challenge in SDN control connectivity, and the need for deployment and orchestration schemes for NFV-enabled drone networks.

Chapter 3

Evaluating Softwarization Gains in Drone Networks

Softwarizing drone networks promises ease of control, enhanced performance, and reconfigurability. While performance enhancements were shown in the literature reviewed in Chapter 2, it was not apparent whether gains resulted specifically from softwarization. The benefits of reconfigurability were not clearly demonstrated compared to non-softwarized drones.

The goal of this chapter is to further motivate the use of softwarization in drone networks and verify possible gains through a proposed evaluation model. We model a system of reconfigurable softwarized drones and provide an extensive evaluation with comparison to non-softwarized drones in several scenarios.

3.1 Introduction

In addition to enhancing the management and utilization of networking resources, softwarization lends itself to leveraging on-board computing power and offering intelligence and autonomy in conducting tasks. For example, reconfigurable drones can use sophisticated machine learning models to conduct sensing tasks or industrial operations to process sensed data on-board and broadcast relevant results in real time. Examples of such uses include detecting damage in infrastructure [Jor+18; Ker+19], forest fire detection [Jia+19] and post-disaster survivor detection [DOD21]. Computations may include dynamically finding optimal trajectories to achieve required results efficiently. This is a powerful alternative to collecting and transmitting all data and process it later in a ground station.

Since task implementations are packaged reusable modules, as virtual machines or VNFs, the drone task becomes defined by the configured module. Tasks implementations can also be interchangeable, either at deployment time or while inflight, leading to an enhanced flexibility and value of a physical drone. The ability to switch tasks while inflight can enable an immediate response to urgent events. For instance, when a sudden event occurs that requires deploying a drone to assess the situation, the operator may reconfigure a drone flying nearby, allowing for a faster response. The newly configured drone can respond to the event and provide immediate results given the on-board computational capabilities.

As discussed in Section 2.3, softwarization is proposed for its qualitative benefits, which include ease of programmability, flexibility, and the logically centralized control as an instrument for adaptive network functions. Existing works in drone networks such as those reviewed Section 2.2.4 demonstrated the feasibility of softwarization, including SDN and NFV, towards their respective problems; however, they do not show the direct value of softwarization or reconfigurability. To the best of our knowledge, no available quantitative results to date demonstrate the direct measurable value of softwarization, particularly the dynamic reconfiguration in drone networks and scenarios involving drones performing several tasks.

In this chapter, we further motivate the softwarization of drone networks and show potential quantifiable gains. Based on the aforementioned motivations for drones with respect to conducting a variety of tasks, we focus on reconfigurability enabled by NFV. To this end, we propose a model for evaluating the gains of reconfigurable softwarized drones. We describe a motivating scenario for drone reconfigurability where a system of softwarized drones managed by a Service Provider (SP) conducts drone tasks as a service using reconfigurable drones that perform multiple tasks per flight implemented as VNFs. We describe the model elements for such a system and describe the orchestration strategies used to receive and deploy tasks efficiently and to respond to urgent events. We present an investigation of the possible benefits of softwarized drones used to perform tasks in the proposed scenario. We evaluate the softwarization gains of the system in terms of its ability to efficiently utilize an available fleet of softwarized drones to complete a campaign of tasks. We also evaluate the system's ability to deploy high priority tasks to respond to urgent events. We compare the system to alternatives with limited and no softwarization capabilities. This evaluation shows novel results quantifying the possible gains due to the softwarization and reconfigurability of drone networks in future applications.

The remainder of this chapter is organized as follows. In Section 3.2, we provide an overview of the model proposed to evaluate softwarized drones. In Section 3.3, we describe the system model and orchestration strategies used for evaluations. In Section 3.4, we provide an extensive evaluation of the proposed model. A summary of this chapter is given in the last section.

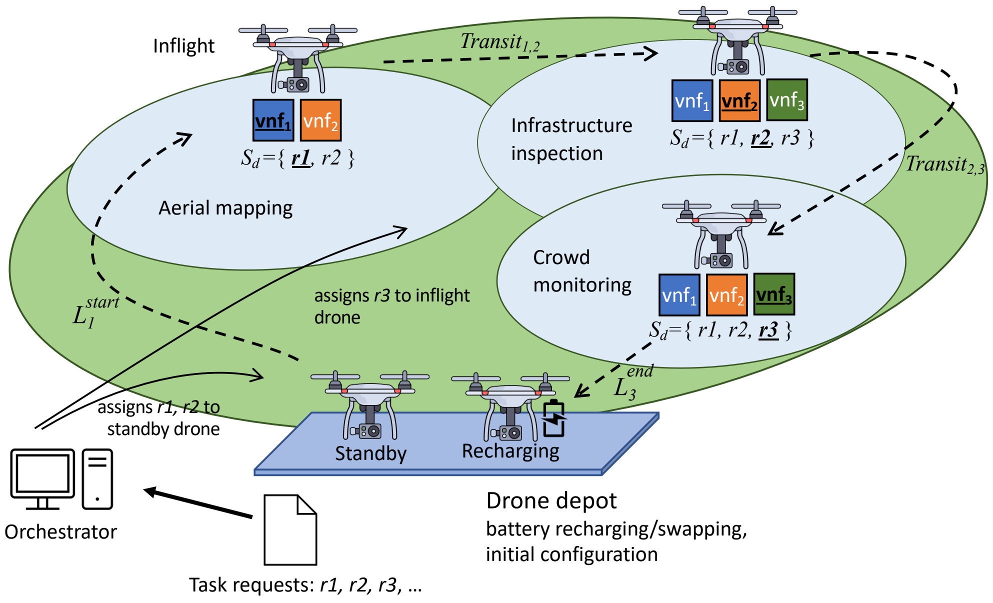


Figure 3.1: Overview of the model elements and the overall operation including drone states, task assignments, and VNF activations. Underlines mark the active task and VNF. Dashed arrows indicate the trajectory of a flight, marked with requests transition distances. Our model is described in further detail in Section 3.3.

3.2 Model Overview

To further motivate drone reconfigurability, we illustrate with the following scenario. An industrial entity, for instance, could utilize a fleet of programmable and reconfigurable drones, thus allowing the fleet to conduct different operations and tasks more efficiently while automating the process. The automation is enabled by softwarization, since it becomes easier to install task software given an enabling virtualized system such as NFV. Automation can include receiving and deploying tasks by automatically reconfiguring drone task software and deploying automated flights. The SP benefits from such flexibility by completing more tasks with a shorter turnaround time, and completing several tasks quickly. Such an application brings an opportunity for SPs to offer or utilize their fleets of softwarized drones to conduct tasks as services for customers. Such a service can provide convenience to different customers such as industries requiring infrastructure inspection tasks, municipalities conducting aerial surveys and mapping missions, law enforcement, weather services, and academic institutions. Most importantly, from the SPs point of view, the reconfigurability of such a system can translate to improved efficiency and increased profitability.

3.2.1 Overview of Model Elements

Consider an industrial entity or an SPs that owns a fleet of drones is offering their fleet to perform various tasks and is stationed in an area with a demand for these services. Such a system consists of a drone depot or station, a set of reconfigurable drones, and an orchestrator that controls the entire system and is responsible for receiving and deploying tasks. The system is depicted in Figure 3.1.

The general operation of the system can be described as follows. Requests for conducting drone tasks are submitted to the orchestrator. Requests state the task requirements in terms of the trajectory, duration, and energy required. As well, requests may supply the software implementation of the task as a VNF or requests may be selected from a catalog of VNFs offered by the SP, which include VNFs that implement algorithms for conducting different tasks such as those shown in Figure 3.1. Requests are handled by the orchestrator, which processes a queue of task requests and schedules tasks to drones using an orchestration strategy. Drones, fitted with a common set of sensors and a computing system, are softwarized and reconfigurable by installing VNF images corresponding to assigned tasks. Due to this flexibility, drones can be assigned a series of tasks to perform in a single flight. While the drone cruises along the combined trajectory of assigned tasks, VNF images corresponding to the active task are instantiated and terminated as the drone transitions from one task area to another. This process is illustrated in Figure 3.1. In this work, we do not consider the time to instantiate and terminate VNFs. In this work, without loss of generality, we assume that VNFs are instantiated and terminated while the drone is transitioning to the next task area.

The depot is where drones are stationed and initially configured. The depot is equipped with facilities to connect to drone computing systems to configure them as directed by the orchestrator. The depot also handles the charging or battery swapping when drones land after performing assigned tasks.

The reconfigurability of drones is enabled by an on-board computing system with virtualization capability to host VNFs. The system should be designed so VNFs are given controlled access to drone sensors and an ability to specify a flight path for the duration of the task. The drone computing system is programmed by the orchestrator with the times to activate VNFs and then deactivating VNFs when tasks exceeds their allotted time or energy. This reconfigurability permits the orchestrator to assign additional tasks to drones while inflight, assuming a high data rate connectivity between the drone and the orchestrator to upload new VNFs and task schedules.

The orchestrator oversees the operation of the system. It employs orchestration strategies to prioritize task requests into a queue and makes assignment decisions in order to reconfigure drones with assigned tasks. Also, the orchestrator monitors and maintains the overall state of the system. More specifically, it maintains the deployment status, current locations, trajectories and energy levels of each drone. Once the orchestrator assigns a number of tasks to a drone, the orchestrator generates the flight plan for the drone to travel from the area of one task to the next. Task VNFs may choose to reprogram its own flight path for the duration when in control of the drone.

The orchestrator is assumed to communicate the above aspects with drones using a SBI similar to the SDN architecture. The SBI allows the orchestrator to control and reconfigure drones as described earlier. Essential control interactions between the orchestrator and drones include uploading VNFs, assigning a list of tasks, and configuring the overall trajectory for drones based on assigned tasks. These interactions take place at the depot initially and optionally for drones inflight when required. During flights, the orchestrator monitors drones and may instruct drones to alter their trajectories when an unanticipated events occurs, such as a sudden battery drain or if task VNFs malfunction.

Our assumption is that reconfigurability relates to what drones can do with the computing resources and common set of sensors fitted on drones. The tasks that can be performed by this system may involve sensing tasks or tasks using specialized algorithms to conduct the task, collect, process and transmit data. The flexibility lies with the ability to change how the drone operates in a flexible and automated manner.

3.3 System Model

We denote by D the set of available drones. At any time instant, a drone $d \in D$ can be at one of three states: standby, inflight, and recharging. The inflight state involves traveling the task area and performing the task. After landing, a drone stays at the recharging state for a duration T_{charge} to charge or swap batteries, after which the drone goes into standby state. C_d denotes the fully charged energy capacity of d.

A task request r arriving to the orchestrator is represented by a tuple $\langle r, Dur_r, K_r, L_r^{start}, L_r^{end}, T_r^{arrival} \rangle$. Here, Dur_r is the time duration of the task, E_r is the energy required to fly while conducting the task, and K_r is the type of the task, which maps to the task VNF image. Furthermore, L_r^{start} and L_r^{end} represent the distances from the depot to the task area and from the end point of the task back to the depot, respectively, while $T_r^{arrival}$ is the arrival time of the request. R denotes the set of all requests. We denote by $Transit_{r,\bar{r}}$ a matrix holding the distances between the end and starting locations of any pair of tasks r and $\bar{r}$. $Tr_v(.)$ denotes a function that calculates the time required for a drone to travel a given distance at a predetermined speed v. Figure 3.1 demonstrates the distances associated with task requests.

The orchestrator maintains a priority queue Q of all arrived task requests. The orchestrator engages its assignment procedure at the arrival of task requests or at the availability of standby drones in order to process Q and assign tasks to drones. At any time instant, a softwarized drone d, can be assigned a series of tasks denoted by S_d, which represent the task list of a particular drone flight with different tasks $r \in R$ as its constituents. We also denote by S_{di} and S_{de} the ith and last tasks assigned to S_d, respectively, whereas the active task is denoted by S_{da}. At the end of each orchestration procedure, drones are assumed to be configured with VNFs of assigned tasks. Then, assigned drones which are in the standby state are deployed.

3.3.1 Orchestration Strategy

The orchestration strategy is responsible for prioritizing requests in Q and matching tasks to drones using certain criteria. We opt to prioritize requests with the shortest Dur_r as this reduces the average starting time of tasks. Then an orchestration procedure assigns tasks to drones by selecting drones that can start assigned tasks in the shortest time given drones' current assignments. A task is assigned to a drone d by allocating from the drone available energy to the assigned task and appending the task to S_d. To make the allocation, the energy required to perform the task on the selected drone must include the energy required to fly the drone to the task area or to transition from the preceding task of the drone, as well as the energy required to return to the station, in addition to the energy required to fly during the task duration.

Suppose we are attempting to assign r to d which has current task assignments S_d. First, we need to calculate the transition time from the task preceding r on d. If no tasks are already assigned to d, then the transition time only involves travel time from the depot to the task location. This is expressed as follows:

$$t_d^{transit} = \begin{cases} Tr_v(L_r^{start}) & \text{if } |S_d| = 0 \\ Tr_v(Transit_{S_{de},r}) & \text{if } |S_d| > 0 \end{cases} \tag{3.1}$$

Then, the energy consumed by the task itself and transition as well as the return trip back to the depot would be:

$$e_d = Energy_v\left(t_{transit} + Tr_v(L_r^{end}) + Dur_r\right) \tag{3.2}$$

where $Energy_v$ computes the drone energy for traveling the given duration in a given speed v as defined later in (3.8). Furthermore, after calculating e_d, we calculate the time to start the task $t_d^{tostart}$, which is the total duration of preceding tasks and their transitions. More specifically, if d is in standby, then $t_d^{tostart}$ involves all durations of tasks in S_d and their transitions, combined with the time to reach the area of the first task:

$$t_d^{tostart} = Tr_v(L_{S_{d1}}^{start}) + \left[\sum_{i \in S_d} Tr_v(Transit_{i-1,i}) + Dur_i\right] + Tr_v(Transit_{S_{de},r}) \quad (3.3)$$

If the drone to be assigned is inflight, then $t_d^{tostart}$ involves the remaining duration of the currently executing task S_{da} and all durations and transitions of pending tasks:

$$t_d^{tostart} = \text{Remaining Duration Of Current Task } S_{da} + \left[\sum_{i \in S_d} x_i \times \Big(Tr_v(Transit_{i-1,i}) + Dur_i\Big)\right] + t_d^{transit} \quad (3.4)$$

where $x_i \in \{0, 1\}$ indicates if $i \in R$ is pending execution when equals to 1.

The above strategy is encapsulated in the procedure listed in Algorithm 1. For each r in Q, the procedure examines all available standby and inflight drones (from all $d \in D$) and calculates $t_d^{transit}$, e_d. If d has sufficient unassigned energy greater than e_d, then it calculates $t_d^{tostart}$ and d is added to the candidate assignment set A_{cand}, where each candidate is expressed as a tuple $\langle d, e_d, t_d^{tostart} \rangle$. Once a number of candidates are collected, the candidate with the lowest $t_d^{tostart}$ is selected and assigned. If no

Algorithm 1 Orchestration procedure

```
Input: $Q, D$
Output: Assignments of tasks to drones
1: $R_{rejected} \leftarrow \phi$
2: while $Q$ is not empty do
3:     $r$ = dequeue($Q$)
4:     $A_{cand} \leftarrow \phi$
5:     for all $d \in D$ on standby or inflight do
6:         Calculate $e_d^{transit}$ as per (3.1)
7:         Calculate $e_d$ as per (3.2)
8:         Calculate $t_d^{tostart}$ as per (3.3) or (3.4)
9:         if GetUnallocatedEnergy($d$) $> e_d$ then
10:            $A_{cand} = A_{cand} \cup \{\langle d, e_d, t_d^{tostart} \rangle\}$
11:        end if
12:    end for
13:    if $A_{cand} \neq \phi$ then
14:        $\langle d, e_d, t_d^{tostart} \rangle \leftarrow$ Select from $A_{cand}$ the tuple with minimum $t_d^{tostart}$
15:        Append $r$ to $S_d$ and update allocated energy
16:    else
17:        $R_{rejected} = R_{rejected} \cup \{r\}$
18:    end if
19: end while
20: Add all $r \in R_{rejected}$ back to $Q$
```

candidate drone is found, then r is rejected. The procedure continues to the next r in Q and repeats the steps above. All rejected requests are put back in Q for subsequent calls to the procedure.

3.3.2 Orchestration with Priority

In certain situations, the orchestrator may receive urgent, high priority task requests that require deploying the task before a deadline. Such tasks can arise in emergency situations where a drone is needed to respond to a certain event within a certain timeframe. The flexibility offered by the system allows for reconfiguring drones for emergency tasks. However, when the system is utilized (all available drones are assigned multiple tasks), then it may not be possible to satisfy some urgent requests before their deadlines. To overcome this, the orchestrator adopts an alternative orchestration strategy that prioritizes urgent tasks and uses preemption if needed to allow such high priority tasks to start before their deadlines. Using preemption, assigned drones pause the ongoing regular tasks and start performing the newly assigned high priority tasks. Preempted tasks are resumed in subsequent flights.

In this setting with high priority tasks, a task request r is further identified by $Prio_r$ and W_r, where $Prio_r \in \{0, 1\}$ indicates the priority class of the request (0 for high priority and 1 for regular requests). W_r indicates the deadline to start the task if it is a high priority one.

The orchestration strategy that considers high priority tasks is described as follows. The orchestrator prioritizes requests in Q by their priority (high priority requests first) then by the durations from the current time instant t to the requested deadline, expressed as $W_r - t$, then by task durations (shortest durations first). To assign task r, the orchestrator examines all $d \in D$ in standby and inflight, and gathers candidate assignments. For low priority requests, candidate assignments are gathered and selected as described in Algorithm 1. For high priority requests, multiple types of weighted assignments are gathered:

- **A1**: non-preemptive assignments that satisfy the deadline
- **A2**: preemptive assignments
- **A3**: non-preemptive assignments that do not satisfy the deadline.

A1 assignments are regular assignments where the assigned task is appended to the drone task list and still can be started before the deadline. This is the preferred assignment as preemption is avoided to limit forestalling regular tasks. The calculation of $t_d^{transit}$, e_d, and $t_d^{tostart}$ is as described earlier using (3.1), (3.2), (3.3) and (3.4). However, in order to meet the deadline, the assignment must satisfy $t_d^{tostart} < W_r - t$ where t is the current time instant.

A2 assignments use preemption if meeting the task deadline is not possible with an A1 assignment, i.e., if the deadline will expire before completing the tasks previously assigned to the drone. A2 assignments require different calculation of $t_d^{transit}$, e_d and $t_d^{tostart}$ since existing tasks in S_d of the considered drone would be preempted and ignored in calculations. Suppose we are considering a preemptive assignment of r to d which has a current task assignments S_d. The calculation of $t_d^{transit}$ is:

$$t_d^{transit} = \begin{cases} Tr_v(L_r^{start}) & \text{if } d \text{ is in standby} \\ Tr_v(ImmTransit_{S_{da},r}) & \text{if } d \text{ is inflight} \end{cases} \tag{3.5}$$

where $ImmTransit_{S_{da},r}$ yields the immediate transition distance from the currently executing task S_{da} to r. Then, e_d is calculated using (3.2). Finally, $t_d^{tostart}$ is equal to $t_d^{transit}$.

A3 assignments are A1 assignments that do not satisfy the task deadline condition expressed above. These assignments are used as a last resort when no drone can be assigned using preemption (i.e., when drones are already performing other high priority tasks).

Assignments are associated with weights ($w = 1$, 2, and 3) corresponding to their types (A1, A2, and A3, respectively). The weights are given to candidate assignment to indicate the assignment type and to allow for selecting the most desired assignments which have the lowest weights.

The algorithmic steps involving the above strategy are listed in Algorithm 2. For each r in Q, all available drones are considered for assignments. For each d in D on standby or inflight, $t_d^{transit}$, e_d and $t_d^{tostart}$ are calculated normally using equations (3.1), (3.2), (3.3), and (3.4). If r is a low priority task and d has sufficient unassigned energy then d is added to A_{cand}. If r is a high priority task and $t_d^{tostart}$ meets the deadline W_r, then d is added to A_{cand} as an A1 candidate. If the deadline is not met, then d is added as an A3 candidate. Then, d is considered for an A2 (preemptive) assignment if it is not already performing a high priority task. After making the calculations using (3.1) and (3.2), d is added as an A2 candidate for r if d has sufficient remaining energy. Such assignment candidates in A_{cand} are represented as tuples $\langle d, e_d, t_d^{tostart}, w \rangle$ where w is the assignment weight, which also indicates the assignment type. Then, the procedure selects the drone with the lowest w first and then the lowest $t_d^{tostart}$, and assigns the task to it according to w.

It should be noted that we only attempt to satisfy the deadlines of high priority requests, while low priority requests are served in a best-effort manner according to their order in Q without preemption. Another aspect to note is that non-preemptive candidate assignments are considered if the associated drone has sufficient unallocated energy for the task. This guarantees that all tasks assigned to the drone are performed in the same flight. However, for preemptive assignments, a drone with sufficient remaining energy for the high priority task is selected, which guarantees completing the assigned high priority task but not the preempted tasks.

Algorithm 2 Orchestration procedure for high priority tasks

```
Input: $Q, D$ with current assignments
Output: Assignments of tasks to drones
1: $R_{rejected} \leftarrow \phi$
2: while not IsEmpty($Q$) do
3:     r = dequeue($Q$), $A_{cand} \leftarrow \phi$
4:     for all $d \in D$ on standby or inflight do
5:         $w = 1$
6:         Calculate $t_d^{transit}$, $e_d$ and $t_d^{tostart}$ as Algorithm 1.
7:         if GetUnallocatedEnergy($d$) $> e_d$ then
8:             if IsHighPrio($r$) then
9:                 if $t_d^{tostart} < W_r - t$ then $w = 1$ else $w = 3$
10:            end if
11:            $A_{cand} = A_{cand} \cup \{\langle d, e_d, t_d^{tostart}, w \rangle\}$        ▷ Add as A1 or A3 candidate
12:        end if
13:        if IsHighPrio($r$) and HasNoHighPrio($S_d$) then
14:            Calculate $t_d^{transit}$ as per (3.5)
15:            Calculate $e_d$ for preemption using (3.2)
16:            $t_d^{tostart} = t_d^{transit}$
17:            if  $e_d <$ GetRemEnergy($d$) then
18:                $A_{cand} = A_{cand} \cup \{\langle d, e_d, t_d^{tostart}, w = 2 \rangle\}$        ▷ Add as A2 candidate
19:            end if
20:        end if
21:    end for
22:    if $A_{cand} \neq \phi$ then
23:        $\langle d, e_d, t_{tostart}, w \rangle$ = Select from $A_{cand}$ tuple with minimum $w$ and $t_d^{tostart}$
24:        Assign $r$ to $S_d$ based on $w$ and update allocated energy
25:    else
26:        $R_{rejected} = R_{rejected} \cup \{r\}$
27:    end if
28: end while
29: Add all $r \in R_{rejected}$ back to $Q$
```

3.3.3 Flight Duration and Energy Consumption

Once assignment decisions are made and drones configured, flight plans are created for standby drones and updated for inflight drones. Once a drone d is assigned a set of tasks S_d, the flying duration can be calculated as:

$$FlightDur = Tr_v(L^{start}_{S_{d1}} + L^{end}_{S_{de}}) + \sum_{i \in S_d} Tr_v(Transit_{i-1,i}) + Dur_i \tag{3.6}$$

Note that $Transit_{0,1} = 0$. If the drone is already inflight, the remaining duration from the current time instant is:

$$\begin{aligned} RemDur = {} & \text{Remaining Duration of } S_{da} + \\ & \left[\sum_{i \in S_d} x_i \times \Big(Tr_v(Transit_{i-1,i}) + Dur_i \Big) \right] + Tr_v(L^{end}_{S_{de}}) \end{aligned} \tag{3.7}$$

where $x_i \in \{0, 1\}$ indicates if $i \in S_d$ is pending execution when equal to 1.

The energy consumption in joules for a drone in forward flight for any duration T in seconds and speed v in meters per second is:

$$Energy_v(T) = P_{ff}(v) \times T \tag{3.8}$$

where $P_{ff}(v)$ is a function that computes the power required for flight in watts given v. We adopted an energy model for rotary-wing drones that models the power consumption of drones in hovering ($v = 0$) and in forward flight. The model involves a host of parameters that include the drone mass, air density, rotors and blade configuration. The model is described in detail in [SAP18]. Note that this is a simplified power consumption model intended for illustration, thus we omit the power consumption for ascending and descending. We also ignore the power consumption for computing and communications as they are negligible compared to the energy required for flight [SAP18] [Seç+20]. The maximum flight time at speed v and energy capacity C_d is

$\frac{C_d}{P_{ff}(v)} \times 0.9$, which excludes 10% of energy capacity for safety.

3.3.4 A Model for Limited Softwarization (Fixed NFV)

For the sake of evaluation, we describe a simpler softwarization model to use as a baseline. In this model, a softwarized drone is only configurable during the standby state and only assigned a single task to perform at a time. We call this model fixed NFV.

The orchestration procedure, listed in Algorithm 3, prioritizes requests with the shortest duration and is activated only with the availability of drones in the standby state, denoted by $D_{standby} \subset D$, since it is only possible to assign tasks to standby drones. The flight duration of a fixed NFV drone assigned task request r can be calculated simply as:

$$FlightDur = Tr_v(L_r^{start} + L_r^{end}) + Dur_r \tag{3.9}$$

Algorithm 3 Orchestration procedure for fixed NFV

```
Input: $Q, D_{standby}$
Output: Assignments of tasks to drones
1: while not IsEmpty($Q$) and $D_{standby} \neq \phi$ do
2:     r = dequeue(Q)
3:     $d$ = select any drone from $D_{standby}$
4:     Assign $r$ to $d$
5:     $D_{standby} = D_{standby} - \{d\}$
6: end while
```

3.3.5 A Model for Non-Softwarized Drones

We describe a model for a system with non-softwarized drones. This model serves to compare against the proposed system model and the fixed NFV baseline. In this system, drones are only fitted or configured to perform a single type of task. Hence, it is not considered softwarized or reconfigurable. However, we assume the system has a simpler orchestrator that monitors drones and assigns tasks. This is in order to compare it fairly with the proposed model.

The orchestration strategy for non-softwarized drones is similar to fixed NFV. However, it only assigns tasks to drones matching the types of the requested tasks. For any non-softwarized drone d, $\bar{K}_d$ indicates the type of the task d performs. A task r can be assigned to d only if $K_r = \bar{K}_d$. The orchestration procedure involves the steps listed in Algorithm 4. Finally, the flight duration calculation is the same as fixed NFV using (3.9).

Algorithm 4 Orchestration procedure for non-softwarized drones

```
Input: $Q, D_{standby}$
Output: Assignments of tasks to drones
1: $R_{rejected} \leftarrow \phi$
2: while not IsEmpty($Q$) and $D_{standby} \neq \phi$ do
3:     r = dequeue(Q)
4:     $d \leftarrow$ select $d$ from $D_{standby}$ where $K_r = \bar{K}_d$
5:     if any $d$ is selected then
6:         Assign $r$ to $d$
7:         $D_{standby} = D_{standby} - \{d\}$
8:     else
9:         $R_{rejected} = R_{rejected} \cup \{r\}$
10:    end if
11: end while
12: Add all $r \in R_{rejected}$ back to $Q$
```

3.4 Performance Evaluation

We evaluate the benefits of softwarized drones (referred to as dynamic NFV) compared to drones with limited and no softwarization as described in sections 3.3.4 and 3.3.5.

We built a detailed simulation environment using Python. The environment models the three states of drones: standby, charging and flying, as well as the energy consumption, and the generation and assignment of task requests according to the described system model and orchestration procedures. The environment employs discrete-event simulation to trigger task generation, orchestration and drone deployment states. Task starting and completion times on assigned drones are recorded according to task durations and transition times.

We investigate the benefits of softwarizing drones performing tasks as services in two scenarios. In the first scenario, the batch of task requests is known beforehand. For this scenario, we report the total time to complete all tasks, which is the landing time of the last drone after performing all tasks. In the second scenario, task requests are not known a priori and thus arrive at random times. For this scenario, the total completion time is affected by inter-arrival times. Instead, we report the average task completion time, which is the delay from the arrival of a task to the end of its execution, averaged over the total number of tasks. The average task completion time can be calculated as:

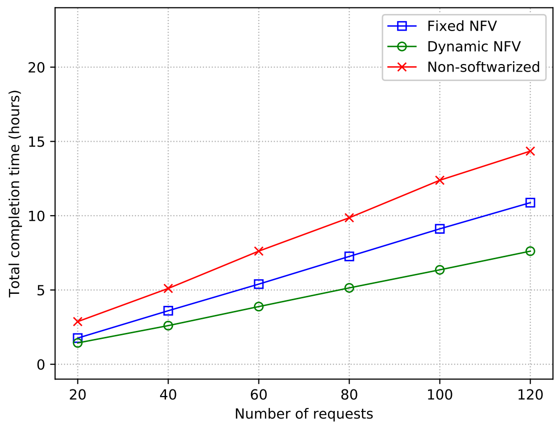
$$\frac{1}{|R|} \times \sum_{r \in R} T_r^{completion} - T_r^{arrival} \tag{3.10}$$

where $T_r^{completion}$ is the recorded time of deactivating the task VNF in the drone after completing the task. We also calculate the total energy consumption for executing all tasks, which is the sum of energy consumed by all flights.

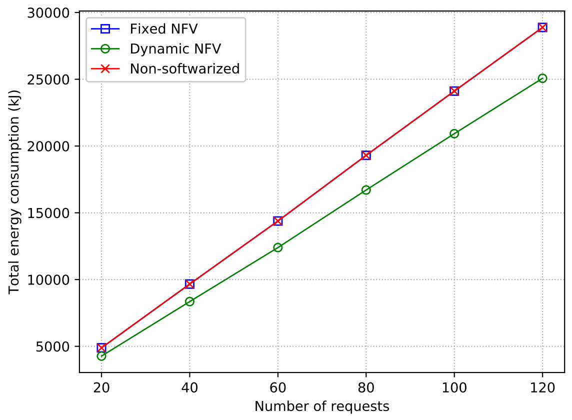
Furthermore, in an additional scenario, we evaluate the ability of the system to satisfy high priority task requests. For this scenario, we report the success rate $R_{success}$, which is the fraction of high priority tasks successfully started before deadline expiry.

3.4.1 Simulation Setup

The simulation is setup as follows. Task durations are uniformly distributed in the range $[5, 20]$ minutes, while energy requirements are equal to the energy required for forward flight for the respective task durations. Request task types are also uniformly distributed over five task types. Assuming tasks take place in a 2×2 km^2 area and the depot located at $(0, 0)$, all distance in L and $Transit$ are drawn from two normal distributions. The first is parametrized with mean $\mu = 1.6$ and standard deviation $\sigma = 0.5$, while the other is parametrized with $\mu = 1.2$ and $\sigma = 0.5$. For the scenario with random requests, inter-arrival times are exponentially distributed with a mean of 5 and 10 minutes, denoted as $R_{arrival}$. In simulations with non-softwarized drones, drones are divided equally to task types. For example, in simulation runs involving five non-softwarized drones, there is one drone available for each task type. Drone flying speeds are fixed at 10 meters per second, while battery capacity is 702.58 kJ resulting in about 44 minutes of maximum flight time. The parameters of the energy model used are as stated in [SAP18], except drone mass, which is set to 3.5 kg. The battery swap or charge time T_{charge} is set to 10 minutes. Simulations are terminated when all tasks are completed. All reported results are averages of ten experiments.



(a) Total task completion time

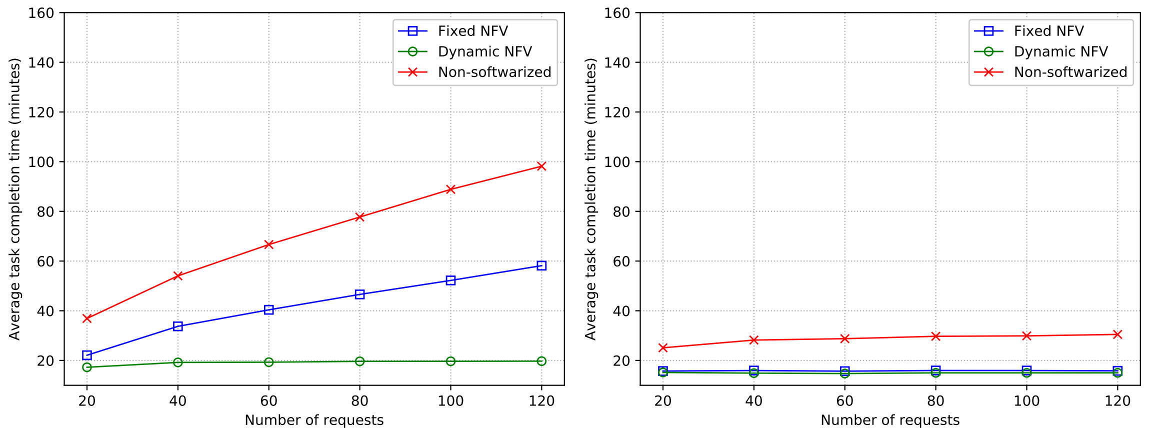


(b) Total energy consumption

Figure 3.2: Performance of the predefined tasks scenario with 5 drones

3.4.2 Results

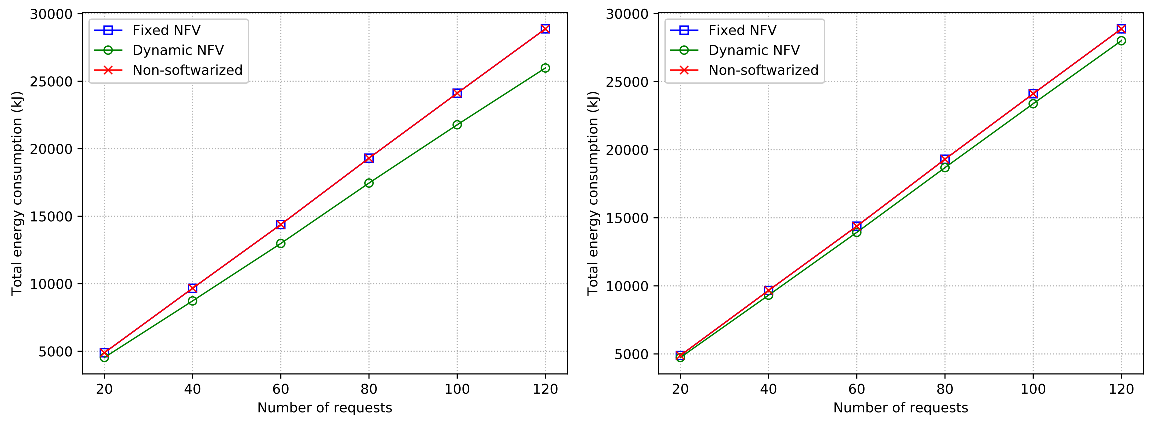
First we investigate the performance of a fixed number of drones while varying the number of task requests. Figures 3.2a and 3.2b show the total completion time and energy consumption for the predefined requests scenario. The results are for five softwarized drones compared to five fixed NFV and five non-softwarized drones. Tasks are completed faster using dynamic NFV compared to fixed NFV and non-softwarized drones. This is due the ability of reconfiguring and assigning multiple tasks to any available drone and due to the time saved traveling between tasks instead of returning to the depot after every task. Without softwarization, tasks take the longest time to complete due to having to wait for the availability of drones that match requested task types. However, with fixed NFV, the system waits for the availability of any drone to be reconfigured for the requested task type. As expected, the total completion time increases linearly with the number of requests, as does the difference in performance. The reduction in energy consumption, as shown in Figure 3.2b, is a direct result of the reduced travel times between tasks due to reconfigurability. Fixed NFV and non-softwarized drones are equal in energy consumption due to the identical operation in terms of performing a single task per flight.



(a) $R_{arrival} = 5$ (b) $R_{arrival} = 10$

Figure 3.3: The average task completion time for the random tasks scenario with 5 drones and $R_{arrival} = 5$ and 10 minutes

Next we repeat the experiment above but with task requests spread out with 5 and 10 minutes average inter-arrival time $R_{arrival}$. In Figures 3.3a and 3.3b, we report the average task completion time. In both cases, with dynamic NFV, the orchestrator capitalizes on the dynamic reconfiguration ability to assign tasks to drones inflight when possible resulting in the shortest per task completion time, which includes waiting time since the request is received by the orchestrator. With $R_{arrival}$ = 5, the dynamic system deploys and completes tasks faster than the fixed NFV and non-softwarized systems. Using fixed NFV, tasks wait for drones to return and recharge before getting assigned, whereas with the non-softwarized system, tasks wait further for drones of the required task type, leading to longer completion times. The

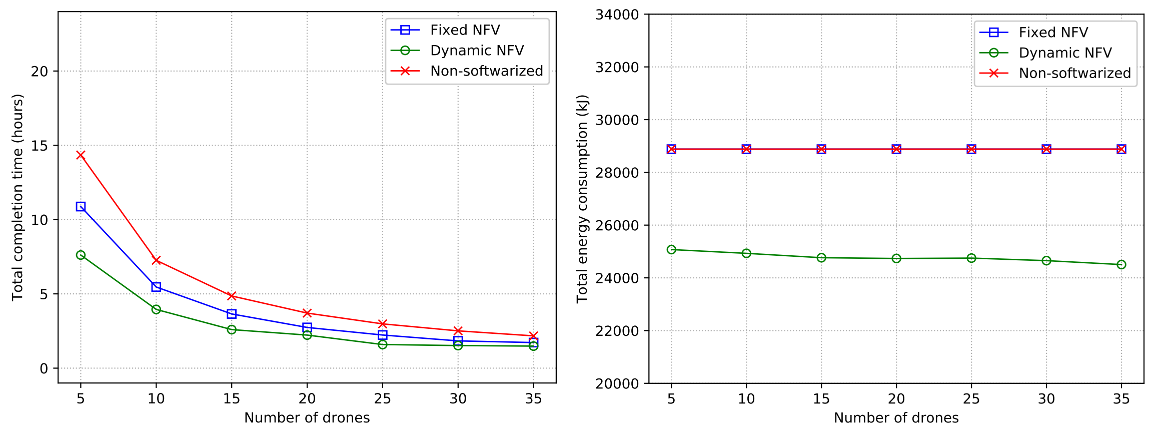


(a) $R_{arrival} = 5$ (b) $R_{arrival} = 10$

Figure 3.4: The total energy consumption for the random tasks scenario with 5 drones and $R_{arrival} = 5$ and 10 minutes

advantage of reconfigurability is also evident with $R_{arrival} = 10$, but with equal performance for both dynamic and fixed NFV due to the lower utilization of the fleet. Both softwarized systems show an advantage over the non-softwarized system, albeit with a narrower difference. The corresponding energy consumption is shown in Figures 3.4a and 3.4b. The energy consumption difference decreases as $R_{arrival}$ increases. This is due to the fact that task requests are more spread out in time and drones become less utilized. This increases the chances of having available drones on standby to serve requests regardless of reconfigurability. As a result, many drone flights in dynamic NFV will constitute a single task per trip leading to similar energy consumption across all variants.

Next, we examine the relationship between the softwarization performance and the size of the drone fleet. We fix the number of requests to 120 requests for both predefined and random scenarios and vary the number of drones in increments of 5 starting from 5 up to 35. For each comparison, non-softwarized drones are divided



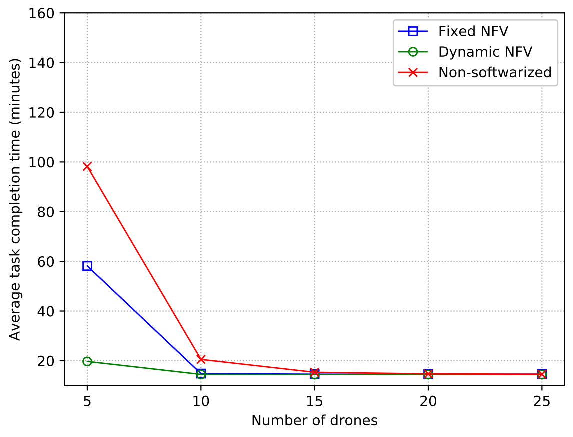
(a) Total completion time

(b) Total energy consumption

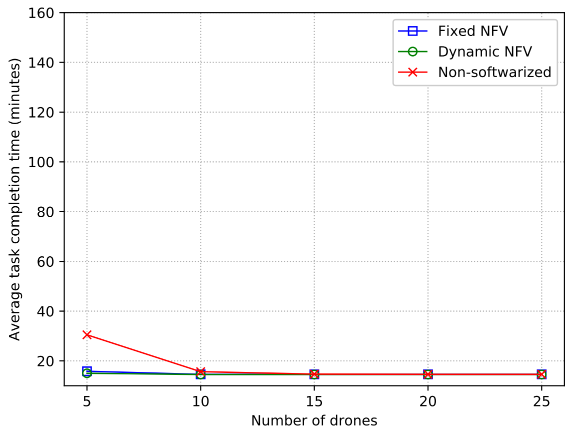
Figure 3.5: Total completion time and energy consumption for 120 predefined tasks with respect to the number of drones

equally to the five types of tasks.

Figure 3.5a shows the total completion time for 120 predefined task requests against the number of drones. Tasks are completed faster using more drones. Naturally, an abundance of drones reduces the performance gains of softwarization. However, the notable result here is that softwarization enhances the performance of a limited number of drones to an extent equal to or better than a larger non-softwarized fleet. For example, 5 and 10 softwarized drones perform similar to 10 and 20 non-softwarized drones, respectively. The corresponding energy consumption is shown in Figure 3.5b. Note that energy consumption does not increase with the number of drones since it depends on the number of tasks and their respective energy requirements in addition to the distances from and back to the depot. As discussed earlier, both fixed NFV and non-softwarized drones have the same energy consumption. Dynamic NFV reduces the energy consumption by about 14% in this instance due to drones traveling from one task to another more frequently than always from and back to the depot. The distances traveled between tasks varies slightly depending on the number of available drones due to the orchestrator assigning tasks to drones closer to task locations. This results in slightly different distances travelled and energy consumption with different numbers of drones.



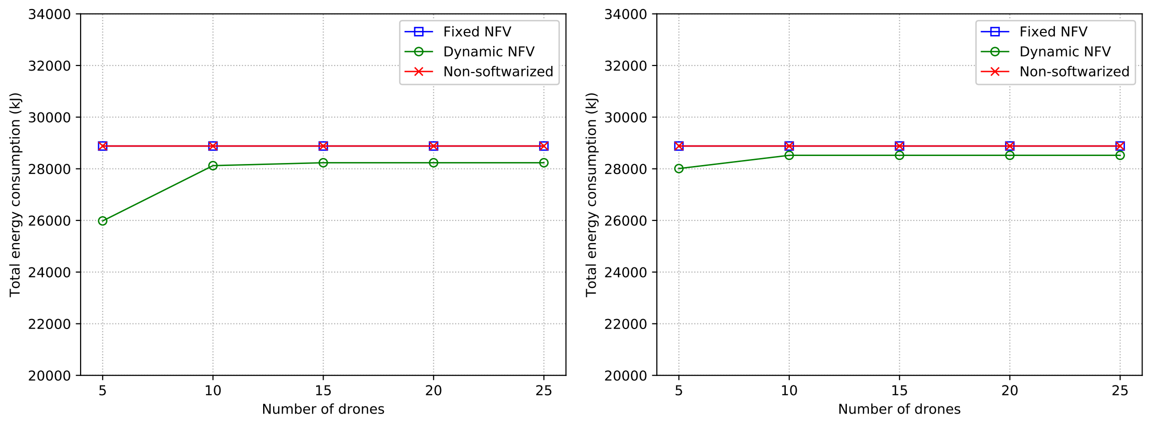
(a) $R_{arrival} = 5$



(b) $R_{arrival} = 10$

Figure 3.6: Average task completion time for 120 random task requests

Figures 3.6a and 3.6b show the average task completion time for 120 random task requests with $R_{arrival} = 5$ and 10, respectively. In both figures, we also observe the efficiency of five softwarized drones compared to the fixed NFV and non-softwarized drones. This is due to the dynamic reconfigurability as the orchestrator can reconfigure any available drone that can start the task in the shortest amount of time, whether the drone is in standby or inflight, leading to a short time to start and complete individual tasks. The same discussion of the previous setting (predefined tasks) applies here in terms of the effect of the abundance of drones. Figure 3.7a shows the total energy consumption. With a limited number of drones and short inter-arrival

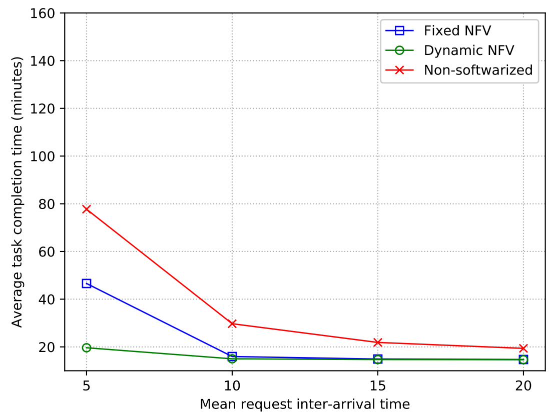


(a) $R_{arrival} = 5$ (b) $R_{arrival} = 10$

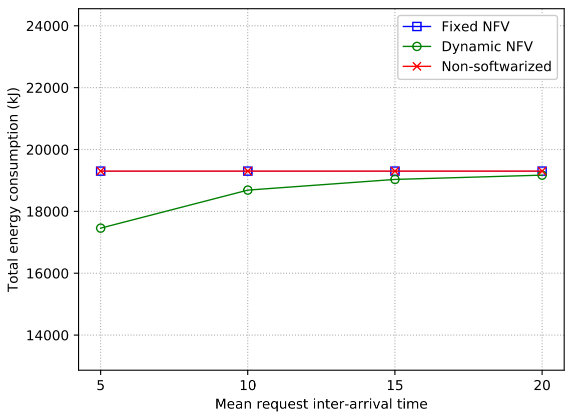
Figure 3.7: The total energy consumption for 120 random task requests

times, the system is clearly more utilized. However, dynamic NFV is able to conserve energy by having drones transition from one task to another and avoid making more frequent and longer trips to the depot. With more drones, dynamic NFV conservers less energy as the system tends to utilize all available drones to serve requests faster by deploying a single task per drone, which leads to similar energy consumption as the alternative systems. A similar effect is shown in Figure 3.7b with a larger inter-arrival time (10 minutes). Energy conservation is limited in this case as dynamic NFV tends to deploy a single task per drone more often.

The effect of the inter-arrival time on the performance with regards to random requests is explored next. A setup similar to the previous one is used with five drones and 80 random task requests. Figure 3.8a shows the value of softwarization in terms of average task completion plotted against $R_{arrival}$. The figure shows how the difference in performance decreases as the inter-arrival time becomes larger and the system less utilized. For energy consumption, as shown in Figure 3.8b, tasks become more spread



(a) Average task completion time



(b) Total energy consumption

Figure 3.8: The avg. task completion and total energy consumption for the random tasks scenario with 5 drones plotted against $R_{arrival}$

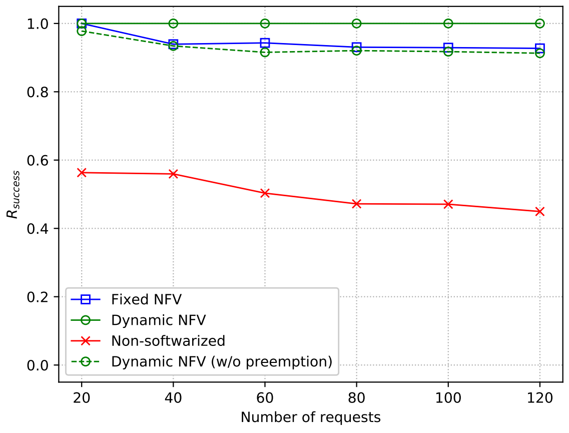
out to the degree that most flights will carry out a single task at a time, making it gradually equal in energy consumption to fixed NFV and non-softwarized drones as $R_{arrival}$ increases.

3.4.3 High Priority Scenario Results

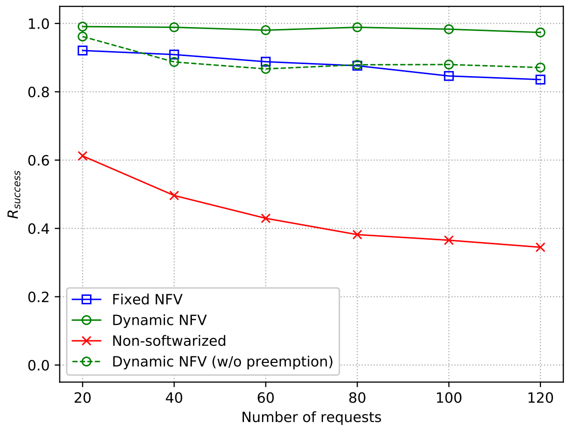
In this section, we demonstrate the ability of softwarized drones with a preemptive orchestration strategy to accommodate high priority requests. We compare the results with the original strategy without preemption, as well as with fixed NFV and non-softwarized drones. Furthermore, we examine the effect of preemption on low priority tasks.

We demonstrate this scenario first with a varied number of random requests with $R_{arrival} = 5$ minutes, to test the dynamic reconfigurability in a relatively utilized system. Requests have a probability 0.2 or 0.4 for being high priority tasks. This probability is denoted by P_{high}. Request deadlines are uniformly distributed between

5 and 15 minutes after request arrival times $T_r^{arrival}$. In the simulation, the procedure $ImmTransit$ draws distances from a normal distribution with same parameters as the $Transit$ matrix. All other parameters are the same as previous evaluations. The metric reported is the rate of successfully started high priority requests, denoted by $R_{success}$, which is the ratio of successfully started high priority tasks before their deadlines. For the effect of preemption, we report the average task completion time and the fraction of preempted tasks.



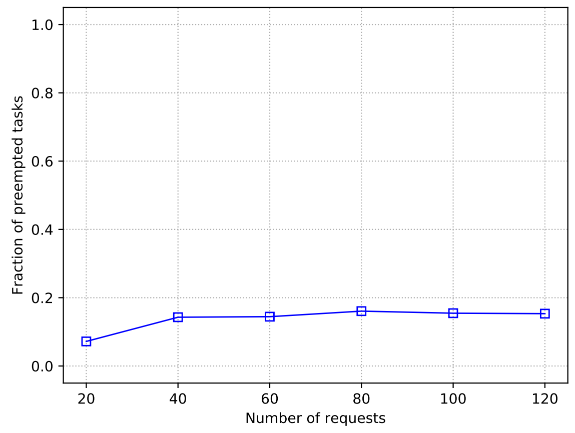
(a) $P_{high} = 0.2$



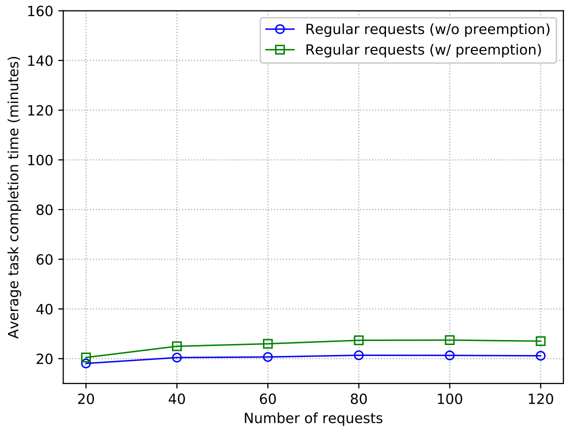
(b) $P_{high} = 0.4$

Figure 3.9: $R_{success}$ for high priority requests with 5 drones

Figures 3.9a and 3.9b show $R_{success}$ for high priority requests with five drones while varying the number of requests from 20 to 120. We can observe how non-softwarized drones are unable to satisfy more than 60%–40% of high priority requests even with the prioritization of requests. On the other hand, all softwarized systems (fixed, and dynamic with and without preemption) are able to satisfy the majority of high priority requests but not all due to the demand for tasks and the limited number of drones. However, with preemption of lower priority tasks, the dynamic system can satisfy almost all higher priority tasks, even with a limited number of drones and an increasing number of requests.



(a) Fraction of preempted low priority tasks

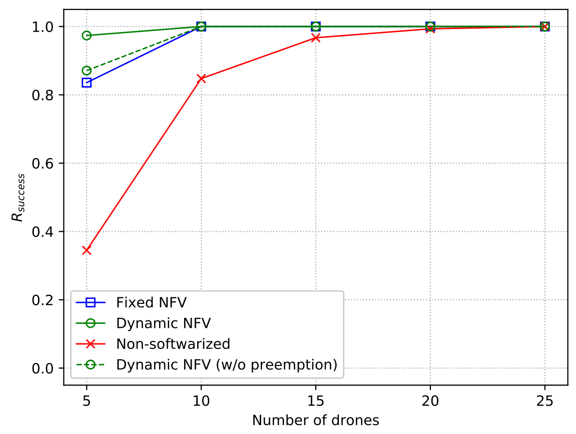


(b) Average task completion time of low priority tasks

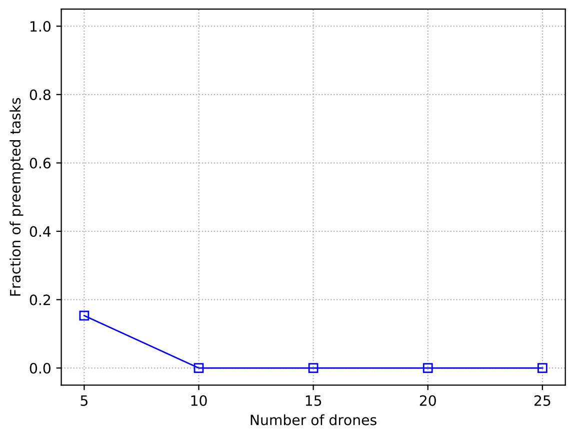
Figure 3.10: Effects of preemption on low priority tasks ($P_{high} = 0.4$)

The effects of preemption on low priority requests are shown in Figures 3.10a and 3.10b. We only show results for $P_{high} = 0.4$, as it is the more demanding instance. As can be observed in Figure 3.10a the fraction of preempted tasks is less than 20% of low priority tasks. This is in line with how the orchestrator avoids preemptions when not needed. The average task completion time for low priority tasks is somewhat affected due to the delay introduced by preemption (Figure 3.10b).

Next, we repeat the evaluation above but we set the number of requests to 120 and vary the number of drones. Figure 3.11a shows the performance in terms of $R_{success}$. We observe the superior performance of preemptive dynamic NFV in contrast to the other variants. Note that we can meet the demand for high priority requests with a small number of drones due to the dynamic reconfigurability and the preemptive strategy. On the other hand, with an abundance of drones, all variants converge to



(a) $R_{success}$



(b) Fraction of preempted tasks

Figure 3.11: The performance of high priority tasks with 120 requests and varied number of drones and the corresponding fraction of preempted tasks (P_{high}=0.4)

the same performance. Figure 3.11b shows the respective fraction of preempted tasks to the number of low priority tasks, which confirms that preemptions are needed with a limited number of drones and are avoided when not needed, when a larger number of drones is available.

3.5 Summary

In this chapter, we evaluated the gains of softwarization in softwarized drones. We first discussed the motivations for drone softwarization to enable reconfigurable drones to provide a variety of tasks. We also discussed the need for quantifying direct softwarization gains that further motivates softwarization. To this end, we presented a model for softwarized drones where drones are reconfigurable using NFV and can perform a multiplicity of dynamically configured tasks. The model incorporates two orchestration strategies. The first strategy targets completion of tasks in the shortest possible time when the campaign is known in advance and also to deploy individual tasks rapidly when requests arrive at random times. The second strategy is tailored for scenarios with urgent requests and employs preemption along with the dynamic reconfigurability to deploy high priority tasks immediately. We compared the performance of the proposed model with variants exhibiting limited and no softwarization. Our results show that a service provider can efficiently perform such a service and complete a predefined campaign of tasks in a shorter period of time compared to the alternatives. A short task completion time was also achieved for random tasks. As well, most high priority tasks can be deployed before the expiry of their deployment deadline. More importantly, in the presented scenarios, softwarization allows a smaller number of drones to perform similar or better than a larger number of drones without softwarization.

Chapter 4

Architecture for Softwarized Drone Networks

Motivated by the performance evaluation of softwarized drones in the previous chapter, we follow by describing an architecture for softwarized drone networks that enables softwarization and addresses challenges discussed in Chapter 2. This chapter describes the general architectural requirements and components of the core management system of softwarized drones to enable reconfigurable drone networks.

4.1 Introduction

We propose an architecture for softwarized drone networks based on an extended SDN architecture. The architecture provides a reconfigurable drone network for different services and applications enabled by programmability, reconfiguration, and centralized control through SDN and NFV. Drones represent the programmable nodes of the network, and they are equipped with various sensors, network radios, and computing resources, allowing drones to perform network and task-related functions implemented as on-board VNFs. The goal of this architecture is describing the requirements to enable the softwarization and add the component that deal with SDN control connectivity and NFV orchestration as described in Section 2.3.

Several architectures were proposed in the works presented in Section 2.2.4. Our architecture was one of the earliest proposals of softwarized drone networks [ATH19a]. Our proposed architecture is differentiated by the following aspects:

- enables both SDN and NFV capabilities,
- incorporates deployment modules concerned with initial planning and deployment of the network and configuration of SDN according to requested services
- addresses the challenge of SDN control connectivity during inaccessibility to ground infrastructure
- employs deployment and dynamic orchestration schemes of drone networks defined by SFCs.

The remainder of this chapter is organized as follows. An overview of the architecture is given in the next section, followed by a description of the architecture components in Section 4.3. Example use cases are discussed in Section 4.4.

4.2 Architecture Overview

The overall network architecture is depicted in Figure 4.1. The infrastructure layer is comprised of programmable drones that are flexible in their network structure and functions. The network is managed by a core management system that consists of two primary components. The first component is the drone deployment modules, which are a collection of modules and facilities that carry out the planning and deployment of the drone network. The second component is the SDN control platform, referred to as SDNC. SDNC is the SDN controller software that manages the network infrastructure comprising programmable drones. SDNC exposes an NBI to enable higher level components realized as SDN applications to implement the control of the network and mission. SDNC also controls the network infrastructure through a control channel and protocol representing the SBI (e.g., OpenFlow). The interface is extended to support a wide range of messages for monitoring, command and control of drones and the mission, and the dynamic reconfiguration of drone virtualized functions. The shaded components in Figure 4.1 represent components we propose in the chapters following as part of this thesis.

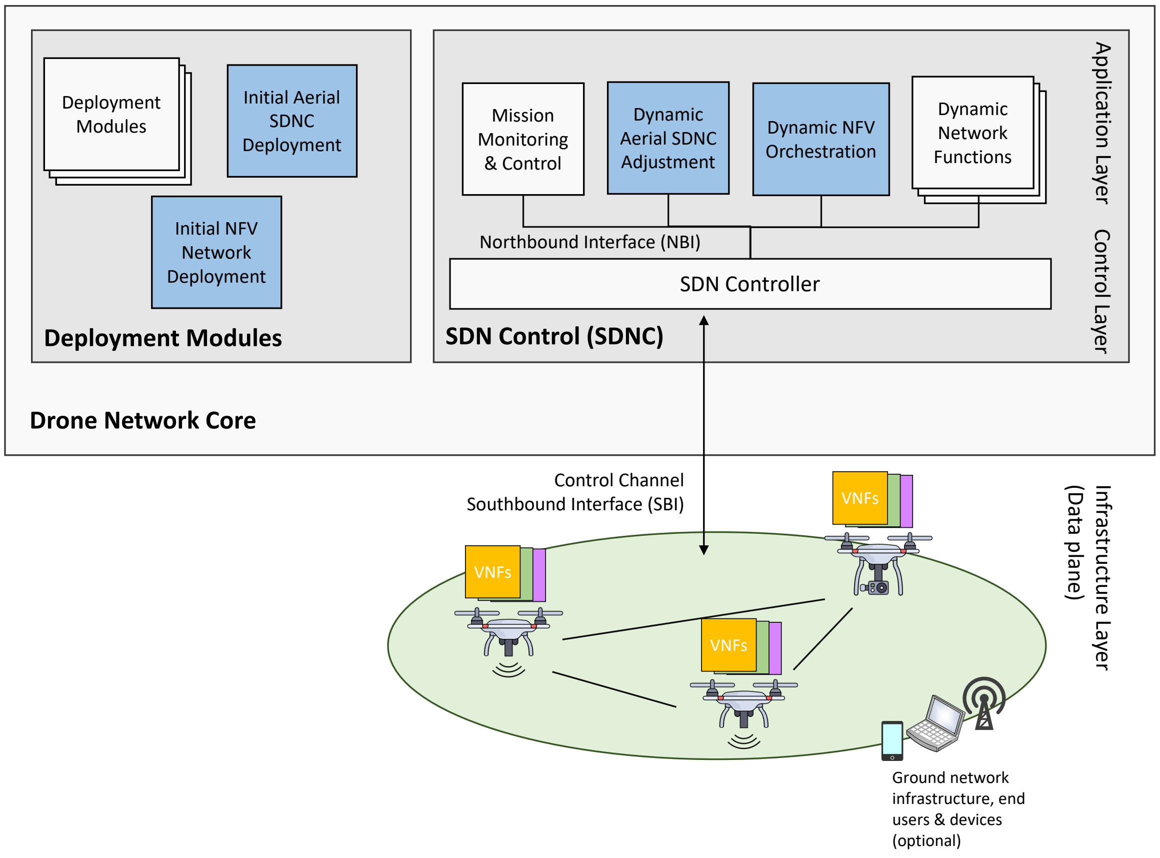

Deployment Modules
Initial Aerial SDNC Deployment
Initial NFV Network Deployment
Deployment Modules
Mission Monitoring & Control
Dynamic Aerial SDNC Adjustment
Dynamic NFV Orchestration
Dynamic Network Functions
Application Layer
Northbound Interface (NBI)
SDN Controller
Control Layer
SDN Control (SDNC)
Drone Network Core
Control Channel
Southbound Interface (SBI)
VNFs
VNFs
VNFs
Infrastructure Layer (Data plane)
Ground network infrastructure, end users & devices (optional)

Figure 4.1: The proposed architecture for softwarized drone networks

Missions are specified in terms of the SDN applications running on top of the SDN controller and used to manage the mission and the network, as well as the

VNFs images to be deployed onto the computing resources of each drone. For example, for a sensing mission, SDN applications may include monitoring and control applications that collect measurements from drones, monitor the mission, and present real-time information in a user interface made for mission operators. Mission control applications may receive commands from operators to direct the mission operation. Other SDN applications include dynamic network functions tasked with implementing the networking aspects and forwarding traffic through available links. The VNFs running on drones can include VNFs that perform network functions, control drone flight paths, or utilize the on-board sensors to collect data. VNFs can also implement local on-board intelligence using, for example, machine learning models to analyze acquired data and deliver relevant results. Such VNFs can optionally interact with corresponding SDN applications or ground users depending on the use case requirements. Additional mission specifications can include the required configuration data used by the supplied applications and VNFs, such as geographic location limits and predetermined flight trajectories.

4.3 Architecture Components

The components of the architecture are as follows.

Drones: Drones represent the physical infrastructure nodes in the SDN architecture. A typical drone is equipped with one or more wireless interfaces and a set of sensors commonly used. Drones are also equipped with computing resources used by the programmable software components described below.

Drones can have varying roles depending on their capabilities and assigned roles given the mission requirements. Two main roles which can be assigned to drones are, *task drones* and *network drones*. Task drones perform the actual tasks of the mission, such as sensing, monitoring, or providing wireless access to ground end users, while network drones provides connectivity for the mission. Both task and networking related functions as well as the overall configuration of drones are managed through the assumed extended SBI.

In terms of network functionality, drones can establish connectivity to terrestrial networks or function as the nodes of a wireless multi-hop network, depending on the deployment scenario. Network drones can be separate from task drones, or a capable drone can carry out both roles provided if the drone is equipped with the needed components and resources. Network drones perform additional roles such as data relaying (forwarding) and SDN control. SDN control drones are aerial SDN controllers that are deployed with the drone network to allow the network to operate in a geographically distant area from the ground infrastructure as will be discussed later.

Since drones represent the softwarized infrastructure, drones must consist of hardware and software components that enable programmability and reconfiguration. Similar to architectures such as [Nog+18; RS17], the expected internal drone components are depicted in Figure 4.2, and described as follows.

- *Hardware Components:* The hardware components of the drone include the computing and networking devices, sensors, and any specialized equipment such as thermal imaging cameras.

- *Operating System and Virtualization Manager:* The hardware components are abstracted by an operating system (OS) with a hypervisor or virtualization manager for virtualizing the underlying hardware. The virtualization manager is responsible for instantiating and managing VNFs run on drones and facilitating how VNFs access the underlying physical resources. The virtualization manager is configured and instructed through the control interface. A light-weight form of virtualization, known as containerization can be used to run VNFs [Fel+15] to preserve computing resources. Rather than supplying VNFs as full virtual machines (VMs) each with its own OS kernel, VNFs are supplied as containers, consisting of VNF programs that run in isolation and share the host OS when instantiated.

- *SDN Control Channel*: This component is one end of the control channel between the SDNC and drones. It operates at a lower level associated with the operating system and virtualization manager. Control messages from the SDNC configure the different components of drones. These include instructions for configuring the hardware components and controlling network routing and forwarding. Instructions also include those for configuring flight control and VNF instantiations. All such commands are delegated to relevant components within drones. Through this interface, drones report their current state such as position and battery level back to the SDNC.

Apart from the OS and virtualization manager, the remaining components are virtualized components deployed as VNFs:

- *Network Forwarding*: responsible for forwarding network traffic through the available wireless interfaces. It represents a virtualized forwarding or routing table for the drone allowing it to act as an SDN device.

- *Flight Control*: responsible for flying the drone. Its functionality is controlled by

SDNC applications and may implement autonomous or preprogrammed flight control following preconfigured trajectories.

- *Additional VNFs*: are related to the mission and can be configured and instantiated on the drone. VNFs include task-specific VNFs intended to run on the drone if needed such as sensors and camera drivers and VNFs for traffic or data processing.

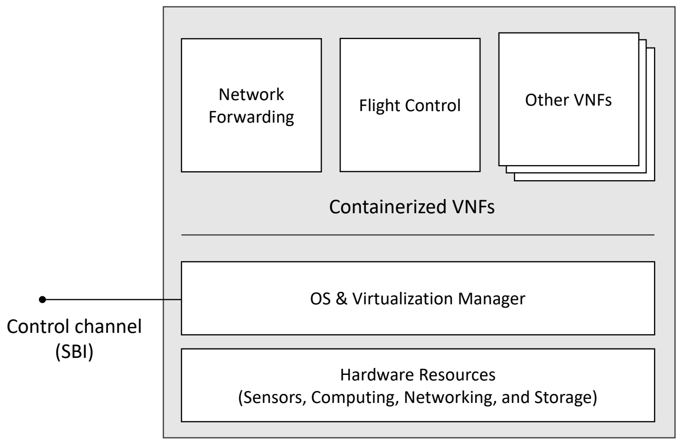

Network Forwarding
Flight Control
Other VNFs
Containerized VNFs
OS & Virtualization Manager
Control channel (SBI)
Hardware Resources (Sensors, Computing, Networking, and Storage)

Figure 4.2: Drone architecture

Drone Network Core: The network core, as depicted in Figure 4.1, consists of two components, namely, the network deployment modules and the SDNC components, which are described as follows.

Deployment Modules: This component is a collection of planning and deployment modules responsible for configuring the SDNC and drones for the mission according to requirements specified by the service requester. The component is assumed to be associated with physical stations for drone payload installation, setup and deployment. This configuration is done according to the modules implementing initial planning and deployment schemes of the network. Such schemes select the drones required for the task and install required configuration and VNF images to be used. The SDNC is configured with the supplied SDN applications. The modules include deployment schemes for specific use cases as well as the initial deployment schemes proposed in this thesis.

SDN Control (SDNC): This component consists of the SDN controller and several accompanying applications. While traditionally the SDN controller is responsible for controlling the network data plane, here it is extended to support the overall control of the drone network including mission command and control, drone monitoring and reconfiguration, in addition to network functions. The SDN controller can be an extensible off-the-shelf SDN controller platform such as OpenDayLight [Ope] and ONOS [Ono]. Typically, the SDN controller keeps track of the network topology and maintains an up-to-date global state of the network. To support drone networks, the SDNC must maintain additional state information related to drones, such as current positions and battery levels. As well, the SDNC exposes a programming interface (the NBI) to higher-level SDN applications. The higher-level applications include mission monitoring and control applications which are mission or task-related functionalities as described in Section 4.2. SDN applications also include dynamic network functions such as routing protocols, network or security policies, and NFV orchestration modules. Such applications rely on the network state maintained by the SDN controller to support their decisions. In turn, such applications push commands and configurations through the NBI to the SDN controller, which the controller push to drones via the control channel (SBI). Drones and active VNFs receive such commands and may communicate back with corresponding applications in SDNC through the control channel.

The proposed dynamic schemes, namely the Dynamic Aerial SDN Control Adjustment and the Dynamic NFV Orchestration, part of our proposals discussed in Chapter 5 and Chapter 6, are implemented as SDN applications and are used when needed. The additional network functions are additional SDN applications supplied for specialized network functions (e.g., routing) and specialized types of missions.

Ideally, the SDN controller and applications reside in servers or drone control stations. We assume in such cases that the control channel can be established using a suitable wireless technology. For missions in remote areas with no access to an infrastructure, drones can be configured as controllers and are deployed as proposed in Chapter 5. Controller drones are configured as instances of SDNC with a select set of SDN applications as needed. Multiple controller drones can exist in addition to the ground SDNC, each controlling a different set of drones, and as a whole representing a logically centralized SDNC.

End Devices and Ground Infrastructure: End devices are optional components that represent equipment used by end-users to access the network service provided by the drone network.

4.4 Use Cases

A network system based on this architecture is suitable for operators that require limited deployments or others that require multi-purpose large-scale deployments. Below we gives examples of use cases of this architecture.

4.4.1 Small-Scale Scenarios

This type of deployment suits small and recurrent single applications. A matching example is monitoring and scanning of areas impacted by a natural disaster, such as hurricanes and earthquakes. A drone network system based on our architecture can be utilized by agencies that handle such situations. The agency maintains a fleet of drones with SDN-enabled components and core management system, as well task-specific equipment such as thermographic and regular cameras. The softwarized components that define such a network include SDN applications that conduct the overall command and control of the mission with associated user interfaces for human operators. Drone VNFs are also provided for capturing and encoding photos and video from drone cameras and forwarding them to operators or end devices for monitoring purposes. Such VNFs may incorporate machine learning models for identifying and mapping affected areas, which are made available to operators without the need for further processing. All softwarized components are updated regularly to improve the mission operating and networking performance without requiring frequent updates to physical drone components.

4.4.2 Large-Scale Scenarios

This type of deployment is suitable for a large SP in an urban area that owns a fleet of drones that can be deployed as a flying network infrastructure integrated with the terrestrial network. Assuming the existence of facilities that automate configuration and drone deployment, the drone network can be utilized for several purposes. The SP can be a telecom provider that employs some drones in the fleet as aerial BSs to assist the coverage of mobile users. The SP uses this flexibility to meet time-varying service demands, improve or compensate coverage in crowded events or during outages, in which drones provide connectivity to mobile users and also establish backhaul links if needed. The computing resources can also be utilized to deploy computing services at the edge of the network and close to users. Examples of such services are discussed in Chapter 2 and include information caching and offloading of resource-intensive computation tasks from user devices. The reconfigurability of the system allows the SP to effectively manage the provisioning of services according to demand.

Due to this reconfigurability, the SP can offer its drones as a service to other customers or entities interested in performing drone tasks as proposed in Chapter 3. Customers of drone services can include city municipalities, law enforcement, researchers and others who may optionally use customized software components to enable their use cases of drones. The SP capitalizes on the flexibility and automation of its system to reconfigure drones and provide the requested services.

4.5 Summary

In this chapter, we proposed a softwarized drone network architecture and described its components. The architecture enables the flexible deployment, management and reconfiguration using SDN and NFV technologies. As well, the architectures addresses current limitations relating to SDN control and orchestration of NFV-enabled networks. As part of this architecture, Chapter 5 is dedicated to SDN control deployment and adjustment, while Chapter 6 is dedicated to NFV-based deployment and orchestration.

Chapter 5

SDN Controller Deployment

Ensuring the SDN controller connectivity in software defined drone networks flying in areas isolated from ground network infrastructure is paramount. In this chapter, we discuss the need for deploying aerial SDN controllers, and we propose schemes for the deployment and dynamic adjustment of drones with SDN control capabilities.

5.1 Introduction

The controller is a key component in the SDN architecture as the controller implements the data plane (forwarding) logic. The controller, through its core functionality and with decisions from higher level SDN applications, interacts with network nodes (network devices) to monitor the network and build a global network view. As well, the controller programs the data plane behavior by sending instructions to the network nodes. To do so, the controller maintains control links or connections with all network nodes to enable continuous messaging between the controller and network nodes. Although the controller is logically centralized, multiple distributed controllers can exist to increase the capacity and resilience of controllers, which collectively coordinate to monitor and program the network. Due to the role of the controller in

SDN, the processing capacity of the controller and the connectivity to network nodes are key to the network performance. The controller allows the network to utilize the flexibility of SDN, and also allows the controller to quickly react to changes in network conditions and program the network accordingly. As a result, considerable research efforts focused on the SDN controller placement, known as the controller placement problem (CPP), in wired networks. The CPP is concerned with determining the number and locations of controllers needed for a given network topology. The CPP also determines the assignment between controllers and network devices (i.e., switches in wired networks) [Wan+17]. We overview works related to CPP in Section 5.2.

In drone networks, maintaining the SDN controller connectivity with network nodes becomes a challenge due to the network mobility and changing topology. As the network has some degree of mobility, network nodes need to maintain connectivity with the controllers located in the ground network infrastructure or control stations. The lack of control channel connectivity will limit where nodes can be deployed in many practical scenarios. This aspect is especially challenging when the drone network is required to operate independently from the ground network infrastructure in a remote area or where no other network infrastructure is available. We propose deploying dedicated drones as SDN controllers to allow more flexibility in network deployment locations and mobility while the network is separated from ground infrastructure. Having dedicated controllers allows for more freedom in positioning controllers independently from the topology of the drone network. Also, it allows for more flexibility in optimizing control links and inter-controller links as discussed below.

With that in mind, the dynamic and wireless nature of drone networks mandates careful consideration to controller deployment. Due to the range of control links, the processing capacity of controllers and the size of the area where the network is deployed, multiple drones may be required to operate as controllers. In addition, inter-controller connectivity is also required for controllers to collectively maintain the state of the network and control the network accordingly. As well, the mobility of the network and the continuous change in topology requires that controllers adapt to such changes to maintain connectivity to network nodes.

To this end, our aim is to address the aspects discussed above by implementing controller deployment schemes that deploy a minimum number of drones that operate as SDN controllers, and dynamically readjust controller locations to maintain control links as the network topology evolves during the mission. The goal of the dynamic adjustment scheme is to readjust the controller deployment in the shortest possible time.

The contributions presented in this chapter are as follows. First, we propose a controller deployment scheme that initially deploys a minimum number of drones as controllers as part of the drone network. Controllers are deployed to ensure the connectivity between controllers and network nodes through direct single-hop links within a threshold of a link quality metric and within the processing capacity of controllers. The scheme also ensures inter-controller connectivity and prioritizes direct inter-controller links but with the flexibility to allow for indirect links. Second, we propose a dynamic deployment scheme that updates the deployment by maintaining the original constraints while minimizing the controllers' transition distance. This is done to make controllers readjust quickly and become available to moving nodes while limiting the energy required to travel. We also present alternative initial and dynamic schemes used for cases when only a limited number of controllers can be deployed. Evaluations show the possible results in a number of deployment scenarios.

The reminder of this chapter is organized as follows. In the following section, we discuss related works and the motivation to address controller deployment in drone networks. In Section 5.3, we discuss the considered scenario, network architecture, and assumptions. Then, we describe the details of our system model and the proposed initial deployment and dynamic adjustment schemes. The evaluation and results are presented in Section 5.4. Lastly, we provide some additional remarks and recommendations followed by a summary of this chapter.

5.2 Controller Placement in the Literature

SDN controller placement problem or the CPP is one of the key topics in the SDN literature in wired datacenter and Wide Area Networks (WANs). The work by Heller et al. [HSM12] introduced the CPP and considered the average latency between switches and controllers, then others followed suit. Existing works differ in the placement metrics used, such as switch-controller latency [HSM12], controller capacity [Yao+14], and deployment cost [Bar+13]. Other works focused on the availability and fault tolerance of SDN controllers [Als+18; Tan+18]. CPP is also examined considering inter-controller connectivity and the resilience of the distributed control plane. In [Tan+18], Tanha et al. present a capacity-aware placement scheme that places and assigns multiple controllers to each switch while guaranteeing switch-controller and inter-controller delay. An important aspect in CPP is the dynamic adjustment to controller placements and assignments to switches considering current conditions of the network. In [Bar+13], authors build a dynamic controller provisioning scheme that runs at specified periods, activating and deactivating controllers depending on current demand. The placement in this scheme considers various controller costs including flow-setup time and statistics collection, as well as inter-controller connectivity.

In [Abd+17], Abdel-Rahman et al. introduced wireless CPP (WCPP) where each of the links between switches and controllers are wireless. A stochastic approach was used to model the uncertainties of the wireless channel, with the goal of minimizing the number of controllers. In [DHZ18], Dvir et al. introduced another model for WCPP using a new placement metric called *transparency*, which is the marginal average latency in the data plane caused by interference from controllers. The model's objective is to minimize the average outage of control links and the average latency while constraining the link throughput and transparency below certain thresholds.

While incorporating SDN into this domain has been discussed in works such as in [Seç+18a], the requirements and deployment objectives of the control plane in drone networks has not been studied. As discussed in Section 5.1, certain deployments may require deploying drones as controllers to increase the deployment flexibility of the network. This introduces additional challenges since these deployments require considering the number of controllers and the mobility of the drone network. Controller placement needs to be dynamically adjusted so that all controlled nodes (drones) have persistent access to controllers. None of the above works can be applied directly to our proposed architecture. Existing works considered fixed network topologies where controller locations are also fixed. Works addressing WCPP did not consider inter-controller connectivity. In our architecture, controllers are airborne and can be positioned freely in space. We also consider a dynamic network topology due to drone mobility. This imposes an additional challenge which requires continuous adjustment to controllers to maintain control link connectivity.

5.3 System Description

In this section, we describe the deployment scenario, network architecture, and assumptions. Then, we describe the system model and the initial deployment and dynamic adjustment schemes that we propose in this chapter.

5.3.1 Deployment Scenario and Network Architecture

We consider a scenario where the drone network is deployed to conduct certain tasks such as monitoring a remote area and providing connectivity to ground users in a disconnected area. In such a setting, the drone network forms a topology over the target area where data traffic is exchanged between drones in the network. As well, drones have a limited degree of mobility, such that they hover and maintain a topology for a certain amount of time, then their locations are adjusted at some points in time to accommodate new requirements. As the network is based on SDN, controllers are required to monitor and program the data plane according to network conditions. Due to the lack of ground network infrastructure in this scenario, the deployed network needs the deployment of drones as SDN controllers along with the network.

The physical architecture in this scenario is depicted in Figure 5.1. The architecture constitutes instances of the SDNC deployed as control drones. These are deployed to control the network while away from the ground core system. Hereinafter, we refer to control drones as controllers, and refer to all network nodes as nodes and drones interchangeably.

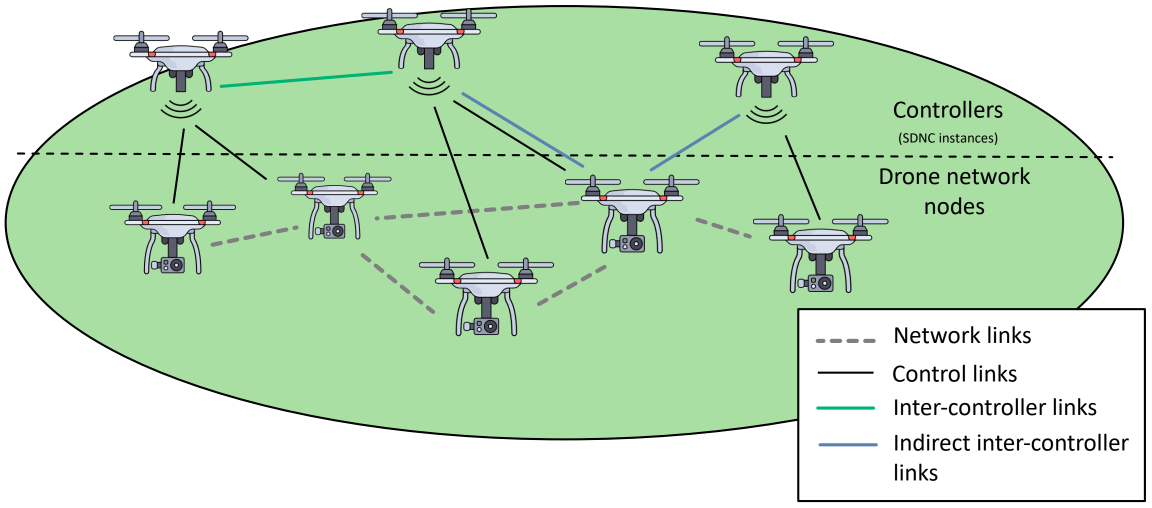


Figure 5.1: A schematic of the physical architecture of the network and controller deployment, including control links and types of inter-controller links

As shown in Figure 5.1, network nodes are interconnected as required for the mission. Controllers establish control links to nodes to monitor and reconfigure them as dictated by the SDN applications overseeing the mission. Controllers are also connected through inter-controller links to maintain a synchronized global view of the network state. Controllers are inter-connected directly or by using indirect links through the drone network.

The network deployment is done within the framework of the logical network architecture discussed in Chapter 4. We assume a drone network is already deployed and adjusted at different times according to mission requirements. Controllers are deployed separately by the initial deployment scheme we propose in this chapter and part of the network deployment modules. Dynamic adjustments to controllers in response to network topology changes are carried out by the proposed dynamic scheme, and considered as part of SDNC, as the scheme relies on the current state of the network given by SDNC.

5.3.2 Assumptions

We assume that the network can afford to dedicate a number of network nodes (drones) as controllers, with the freedom to move them independently from the network topology required by the network. Generally, many existing works determine the controller placement by minimizing the node-controller latency or by ensuring that it stays below a certain threshold [Lu+19]. This is in order to minimize the effect of the control latency on reconfiguring the network and setting up network flows. In this work, we ensure or maximize direct single-hop connectivity between controllers and network nodes, and assume that control links use different channels from the network data links. In this manner, we indirectly reduce the node-controller delay and its potential negative effects on network performance.

5.3.3 System Model

The drone network comprises a connected and undirected graph $G = (D, E)$, where the vertices set $D = \{d_1, d_2, \ldots, d_{|D|}\}$ is the set of network drones which are also considered SDN devices, E is the set of edges representing the wireless links formed between each pair $d_i, d_j \in D$, and $|D|$ is the number of drones in the network. Each network drone d_i generates a number of requests to its controller. The rate of requests in a unit of time is R_{d_i}

The network system will deploy a number of controllers limited by the maximum number of controllers to deploy N, which is defined by the network operators according to available budget. Controllers can be deployed anywhere in the 2D plane of the coverage area at a suitable altitude. L is a discretized set of all allowable controller locations, where $l_i \in L$ represents a possible 2D coordinates for a controller location.

The placement will result in N_{ctl} controllers represented as $C = \{c_1, c_2, ...c_{N_{ctl}}\}$. Each $c \in C$ maps to a placement location $l_i \in L$. All controller drones have a capacity for processing drone requests expressed as C^{cap}.

The wireless communication model between all drones including controllers is based on a Friis-based free space propagation model, where path loss, in dB, over transmission distance d is defined as [LH06]:

$$PL(d) = 10log_{10}\left(\frac{4\pi df_c}{c}\right)^2 \tag{5.1}$$

where f_c is the carrier frequency and c is the speed of light.

5.3.4 Initial Controller Deployment

Initially, the ground station deploys the drone network and determines its topology and physical locations for the drones. Once a need for a flying control plane is determined, a placement will be computed for controller drones. The number of controller drones and their locations will be determined, as well as the assignment of each node to a controller. The assignment of drones to controllers is based on establishing a single-hop control link within a path loss threshold. The joint placement and assignment scheme deploys a minimum number of controllers, limited by the number of available drones that can function as controllers. We consider an alternative scheme that deploys a fixed and predetermined number of controllers. This scheme is useful when the network operator has a limited number of drones that are capable to function as SDN controllers. In this scheme, controllers establish direct links to all nodes within their control link range. Controllers then may establish indirect control links with the remaining nodes. The objective of such scheme is to place the controllers so that most nodes are within the range of control links. We refer to these two schemes as MinCtl and FixedCtl, respectively.

For connectivity between controllers, each controller must maintain a connection to every other controller to enable synchronization of the state of the network. Ideally, direct connections between controllers is best to reduce latency. However, this can make the deployment difficult to achieve or not feasible given the limited number of controllers all attempting to maintain connectivity to all network nodes. A possible approach is that controllers utilize multi-hop connections through the drone network, forming indirect links overlaid within the drone network links when it is difficult to form direct connections. However, this approach may overuse limited network and computational resources. We adopt a balanced approach by placing controllers in a way that maximizes single-hop links and leverage multi-hop links when needed to ensure inter-controller connectivity as shown in Figure 5.1.

Deployment Problem Formulation

Here we describe the formulation for initial deployment, which is a joint controller placement and assignment problem. Every placement of a controller at l_i and its assignment to any d_i is constrained by achieving a threshold for a link quality metric for the wireless control channel established between the controller-node pair. Let Q_{d_i,l_j} denote the quality of wireless channel that can be achieved between d_i and a controller placed at l_j. The quality metric can be the transmission delay or channel path loss. The threshold for this required measurement is given by Q_{max}.

We model inter-controller links by considering direct and indirect links between any pair of controllers located at l_i, l_j. Let Q_{l_i,l_j} denote the path loss between a pair of controllers at l_i, l_j. A direct link can be established if $Q_{l_i,l_j} \leq Q_{max}$, otherwise, an indirect link is considered. An indirect link is a link over the drone network and allows a pair of controllers to communicate using multi-hop links using network drones as intermediate hops. We express the availability of any type of inter-controller links between a pair of controllers l_i, l_j in terms of hop count $L^{cc}_{l_il_j}$ defined as:

$$L^{cc}_{l_il_j} = \begin{cases} 1 & \text{if } Q_{l_il_j} \leq Q_{max} \\ hops_{l_il_j} & \text{otherwise} \end{cases} \tag{5.2}$$

where $hops_{l_il_j}$ is the number of hops in indirect links between controllers at l_i, l_j when connected to the drone networks. The maximum number of hops for inter-controller links is limited by $hops^{cc}_{max}$.

To calculate $hops_{l_il_j}$, we construct a master graph G_{all} that constitutes the network nodes (all nodes in G) and controller nodes equal to the total number of all possible locations in L. Then, links are added between controllers and all network nodes within the control channel range ($Q_{l_i,l_j} \leq Q_{max}$). Then, $hops_{l_il_j}$ is the shortest path between the a pair of controllers at l_i, l_j in G_{all}.

The decision variables of the problem are defined as follows. We define $y_{l_i} = \{0, 1\}$ as a binary variable to indicate if a controller is placed at l_i. We also define the binary variable $x_{d_il_j} = \{0, 1\}$ to express assignment of drone d_i to a controller located at l_j. We define the binary variable $a_{d_i} = \{0, 1\}$ to indicate whether d_i is assigned to any controller.

The MinCtl Scheme

The objective function of this scheme aims to minimize the number of deployed controllers:

$$N_{ctl} = \sum_{l_j \in L} y_{l_j} \tag{5.3}$$

Additionally, we attempt minimize the number of hops for inter-controller connections, which increases chances of one-hop connections ($L^{cc}_{l_i l_j} = 1$):

$$H_{total} = \sum_{l_i, l_j \in L, i \neq j} y_{l_i} y_{l_j} L^{cc}_{l_i l_j} \tag{5.4}$$

Then, the model, denoted as program MinCtl, is formulated as:

$$\begin{aligned}
&\text{min} && w_1 N_{ctl} + w_2 H_{total} && && (5.5)\\
&\text{subject to} && x_{d_i l_j} \leq y_{l_j} && \forall d_i \in D, l_j \in L && (5.6)\\
& && x_{d_i l_j} \cdot Q_{d_i l_j} \leq Q_{max} && \forall d_i \in D, l_j \in L && (5.7)\\
& && \sum_{l_j \in L} x_{d_i l_j} = 1, && \forall d_i \in D && (5.8)\\
& && \sum_{d_i \in D} x_{d_i l_j} \cdot R_{d_i} \leq C^{cap} && \forall l_j \in L && (5.9)\\
& && \sum_{l_j \in L} y_{l_j} \leq N && && (5.10)\\
& && y_{l_j} y_{l_j} L^{cc}_{l_i l_j} \leq hops^{cc}_{max} && \forall l_i, l_j \in L, i \neq j && (5.11)
\end{aligned}$$

where w_1 and w_2 are weight factors used to adjust the terms of the objective function.

The first constraint (5.6) expresses the assignment and placement relationship where d_i can be assigned to a controller at l_j only if a controller is placed at l_i. Using the constraint in (5.7), the drone-controller assignment is limited by the required path loss Q_{max} for the link between them. The constraint in (5.8) ensures that each drone is assigned to one controller only. Then, using the constraint in (5.9), each controller drone is assigned a number of drones with demand R_{d_i} below its processing capacity. The constraint in (5.10) limits the number of placed controllers to the number of available controllers N, and the constraint in (5.11) ensures that the number of hops for inter-controller links that exist between any pair of controllers is below the maximum allowed number $hops^{cc}_{max}$.

Alternative Scheme (FixedCtl)

The FixedCtl scheme aims to utilize the limited number of available controllers to be assigned and connected directly to most drones, given all constraints expressed before. The goal is to maximize the number of assigned drones expressed as:

$$A_{total} = \sum_{d_i \in D} a_{d_i} \tag{5.12}$$

Note that the scheme does not consider the indirect connectivity to remaining nodes. However, if we maximize the number of nodes assigned with direct control links, it is easy to connect to the remaining nodes through the shortest paths. In addition, we also attempt to minimize H_{total} to reduce the number of hops for inter-controller links.

The scheme can be formulated as follows (program FixedCtl):

$$\begin{aligned}
\text{min} \quad & -w_3 A_{total} + w_2 H_{total} && && (5.13)\\
\text{subject to} \quad & a_{d_i} = \text{or}\{x_{d_i l_{j=1}}, \ldots, x_{d_i l_{j=|L|}}\} && \forall d_i \in D && (5.14)\\
& \sum_{l_j \in L} x_{d_i l_j} \leq 1, && \forall d_i \in D && (5.15)\\
& \sum_{l_j \in L} y_{l_j} = N && && (5.16)\\
& (5.6), (5.7), (5.9) \text{ and } (5.10)
\end{aligned}$$

The combined objective function maximizes A_{total} (by minimizing the negative value) and minimizes H_{total}. The constraint in (5.14) ensures that $a_{d_i} = 1$ if d_i is assigned a controller. This expression is derived from all values of $x_{d_i l_j} \forall l_j \in L$ using the logical OR operator [CBD11]. The constraint in (5.15) ensures that nodes are assigned to none or one controller at most while the constraint in (5.16) ensures that only N controllers are deployed.

5.3.5 Dynamic Controller Adjustment

During the operation of the network, the formation of the drone network may change, and controller links may break as a result of changes. In turn, the drone-controller assignment needs to be updated. However, only updating the assignment using the same controller locations may not be feasible, as not all moved drones can establish links to controllers. In such cases, controller locations need to be recomputed to accommodate changes to the network topology and to allow re-establishing links with their assigned drones. Since controllers are already deployed, they will travel to their new locations in the shortest possible time to become available to their assigned drones. Controller travel distance needs to be minimized, as traveling long distances at high speeds can drain energy resources quickly. As well, controllers must remain connected to each other to ensure a timely synchronization of the state of the network.

The dynamic scheme involves multiple steps. First, it attempts to reassign moving drones with broken control links to other controllers without changing controller locations. This is accomplished using a simple reassignment procedure that abides by the initial constraints (capacity and link quality). This step is referred to as the reassignment step. When a simple reassignment is not feasible due to unsatisfied constraints, the dynamic scheme performs the next step, which is computing an updated controller formation and assignment using the already deployed set of controllers. The scheme attempts to find new locations that require a short time for controllers to transition to, by minimizing the controllers' total travel distance. The minimization of the transition distance allows controllers to adjust quickly to keep up with the network and travel within speeds that conserve energy. This step is referred to as placement with minimum transition distance. Finally, when a new formation of existing controllers is no feasible, the scheme falls back to the initial deployment scheme, which may require deploying additional controllers and more time for the controllers to travel. We refer to this step as unrestricted placement.

In what follows we describe in detail the dynamic scheme, named Dyn-MinCtl, which is used alongside the MinCtl initial scheme. An alternative dynamic scheme based on FixedCtl is described next.

Dyn-MinCtl Scheme

The steps of the dynamic scheme are outlined in Algorithm 5.

Reassignment: The reassignment step is implemented by the procedure listed in Algorithm 6. The procedure attempts to assign drones to existing controllers without moving controllers. The procedure receives a set of updates U, each $u \in U$ is expressed as a tuple $\langle d_i, loc_{d_i} \rangle$, where $d_i \in D$ is a drone that traveling to some location loc_{d_i}. The procedure iterates the set of updates, and for each update, the procedure first checks whether the currently assigned controller is still able to communicate with d_i when d_i arrives at loc_{d_i}. If that is the case, then the procedure continues to the next $u \in U$ (lines 3-7). If d_i, when at loc_{d_i}, cannot communicate with its assigned controller, then the procedure, utilizing its knowledge of the other controller locations, lists all controllers and computes their Q_{d_i,l_i}. For any controller located at $l_i \in L$, if Q_{d_i,l_i} is less than or equal to Q_{max}, then it is added to a list of controllers that it is possible to be assigned to d_i. Then, the list is traversed to pick the first controller that can satisfy the capacity constraint (lines 8-21).

Placement with Minimum Transition Distance: If there are remaining nodes that cannot be assigned to any current controllers, then we recompute an updated placement and assignment that requires a minimal time to readjust. To

Algorithm 5 Steps of the dynamic scheme

```
1: Reassign($U$)                                  ▷ Reassignment step
2: **if** there are unassigned nodes **then**
3:     $sol$ = Dyn-MinCtl()     ▷ Placement with minimum transition distance
4:     **if** $sol$ is infeasible **then**
5:         MinCtl()                               ▷ Unrestricted placement
6:     **end if**
7: **end if**
```

Algorithm 6 Reassign

```
1: Output: new drone assignments for $U$, unassigned drones
2: unassignedDrones ← empty
3: for $\langle d_i, loc_{d_i} \rangle \in U$ do
4:     compute path loss between $d_i$ at $loc_{d_i}$ and its current controller
5:     if path loss $\leq Q_{max}$ then
6:         continue
7:     end if
8:     nearbyCtls ← empty
9:     for $c \in C$ do
10:        compute path loss between $c$ and $d_i$ at $loc_{d_i}$
11:        if path loss $\leq Q_{max}$ then
12:            add $c$ to nearbyCtls
13:        end if
14:    end for
15:    while nearbyCtls is not empty do
16:        $c$ ← pop(nearbyCtls)
17:        if $c$.load $+R_{d_i} \leq C^{cap}$ then
18:            assign $d_i$ to $c$
19:            break
20:        end if
21:    end while
22:    if $d_i$ is not assigned then add it to unassignedDrones
23: end for
```

achieve this, we adapt the initial placement model. First, we limit N by the number of deployed controllers N_{ctl} as we are only repositioning existing controllers. Then, we attempt to minimize the total relocation distance expressed as:

$$Dist_{total} = \sum_{l_j \in L} y_{l_j} \cdot dist(l_j, Prev(l_j)) \tag{5.17}$$

where $dist(.,.)$ is the distance between two controller locations. $Prev(l_j)$ maps l_j to a controller location in the previous placement. A mapping to previous locations is required to calculate transition distances between existing and new controller positions after each placement. To obtain such mapping, we map each candidate location $l_j \in L$ to the closest location of an existing controller from the previous placement. This effectively clusters L into k partitions ($k = N_{ctl}$). Each of the resulting partition $S = \{S_1, S_2, \ldots, S_k\}$, where each $S_i \subset L$, is associated with an existing controller location. Then we limit each partition to one new controller placement, and consider each placement a new location for each existing controller in its partition. We achieve this using the constraint:

$$\sum_{l_j \in S_i} y_{l_j} = 1, \forall S_i \in S. \tag{5.18}$$

This allows us to map new placements to existing ones and compute the distance difference, and then minimize how far a controller can move using (5.17).

When using the dynamic scheme in conjunction with the MinCtl scheme, the objective is to reduce the relocation distance, and if possible, minimize hops for inter-controller links. Minimizing the number of controllers is not needed in this step since we are only attempting to reposition existing ones. This step is implemented using the following program (program Dyn-MinCtl):

$$\begin{aligned} \text{min} \quad & w_4 Dist_{total} + w_2 H_{total} && (5.19) \\ \text{subject to} \quad & \sum_{l_j \in L} y_{l_j} = N && (5.20) \\ & \text{(5.6)-(5.9), (5.11), (5.18)} \end{aligned}$$

Unrestricted Placement: If no feasible placement can be obtained, then we resort to finding a new placement using the initial placement program. This step does not involve restricting the number of controllers to N_{ctl} and limiting travel distances using the constraint in (5.18). If the resulting N_{ctl} increases from the previous number, then we consider additional controllers as new ones that need to be deployed. If N_{ctl} decreases, extra controllers are assumed to travel back to the ground station.

After obtaining an unrestricted placement, we determine which controller travels to which new controller placement location. This includes determining which controller travels back to the station, or which controller travels from the station to a new placement locations. Therefore, we compute a mapping between existing and new placement's locations that allows each drone to travel to the closest new placement location. The mapping allows us to compute the total controller transition distance resulting from the placement update.

The mapping can be determined by solving an assignment problem that assigns locations of existing controllers to new ones such that the total travel distance is minimal. This problem corresponds to the linear sum assignment problem [RT12], which can be demonstrated by the worker-job assignment problem with the goal of minimizing the total cost of assignments. We consider workers as the new placements and jobs as the existing placements. The assignment cost is the distance between pairs of existing and new locations.

Suppose the current number of controllers is $\bar{N_{ctl}}$. After a network update, the dynamic scheme produced N_{ctl} controllers using unrestricted placement. To determine the mapping, we construct two ordered vectors $A = [a_1, a_2, \dots a_N]$, containing existing placements positions, and $B = [b_1, b_2, \dots b_N]$ containing new placement positions, where $N = \max\{\bar{N_{ctl}}, N_{ctl}\}$. If either vector A or B is short of N elements, the vector is complemented with additional elements representing the location of the station

$loc_{st} = (0, 0)$. Mapping locations to loc_{st} allows for determining controllers incoming or leaving from or to the station. We construct a distance matrix $M_{N \times N}$, where M_{ij} is the distance between a_i and b_j. The goal is to find a one-to-one mapping $f : A \rightarrow B$ that minimizes the total distance:

$$Dist_{total} = \sum_{a_i \in A} M_{i,f(i)} \tag{5.21}$$

The optimal mapping will determine the distances to transition to new placement positions. This assignment can be solved optimally using the Hungarian method in polynomial time [Cro16].

Alternative Scheme (Dyn-FixedCtl)

The alternative dynamic scheme is used in conjunction with the FixedCtl initial placement scheme. It involves a reassignment step and, when required, a placement with minimum transition distance.

Reassignment: The reassignment step is the same as the step described in Algorithm 5. The only difference is that the update set U involves only drones with assigned controllers. If one of such drones cannot be assigned a controller, then the next step is engaged to avoid losing assignments. The goal of this step in this scheme remains to avoid expensive computation when the same assignment can be maintained with simple reassignment.

Placement with Minimum Transition Distance: When the reassignment step fails, the objective of this step is having the maximum number of nodes assigned with direct control links and then minimize the transition distance for controllers.

The step is formulated as follows (program Dyn-FixedCtl):

$$\begin{aligned} &\text{min} && -w_3 A_{total} + w_4 Dist_{total} && (5.22)\\ &\text{subject to} && \sum_{l_j \in L} y_{l_j} = N && (5.23)\\ & && \text{all constraints of program FixedCtl (5.13)} && \\ & && \text{and constraint (5.18)} && \end{aligned}$$

The Dyn-FixedCtl scheme does not require falling back to unrestricted placement using the initial scheme, since it always finds a placement (given constraints are satisfied) as not all drones are required to be assigned.

5.4 Performance Evaluation

In this section, we describe our evaluation process and results. The evaluation environment along with the reassignment algorithm were implemented using Python. The optimization models were implemented using Gurobi (v9.1.2). Simulations were run on a machine with an Intel Core i7-9750H CPU at 2.60GHz and with 16GB of RAM.

Since the objective functions include multiple objectives, we employ the hierarchical or lexicographic approach [MA04]. Using this approach, the solver finds a solution for each objective separately in a predetermined priority or order without compromising the solution of the preceding objective. For instance, in solving the MinCtl model given in (5.5), the solver first finds the optimal number of controllers N_{ctl} (highest priority). Then, the solver seeks a solution that minimizes H_{total} without compromising the solution obtained for N_{ctl}. This approach is suitable for models with objectives with distinct priorities where lower priority objectives are not required to affect the higher priority objective. In this work, priority of each objective is according to its order in the combined objective function. All weights of individual objectives are set to 1. The non-linear terms in the objective function (5.4) and in the constraint (5.11) can be linearized as described in Appendix A.1.

For tractability, we discretize controller placement locations L. The area is divided into a grid of cells of size $w \times h$, where each cell is associated with a single controller location $l_i \in L$. Each $l_i \in L$ is positioned in the center of its cell. The size of the cell can be selected to control the resolution of placement locations. Higher resolutions (smaller cell size) require a larger number of decision variables. We found that a cell size of 100×100 m^2 is suitable for the initial (static) case, whereas a size of 50×50 m^2 is best suited for the dynamic case in order to allow more freedom for controllers to move and optimize transition distances.

We assume controllers are placed at a higher altitude than network nodes to avoid obstructing network nodes. All simulation parameters are shown in Table 5.1. Parameter values for controller capacity C^{cap} and request rates R_{d_i} are informed by studies such as [Too+12] and [Tan+18] and adjusted to account for the limited scale of

Table 5.1: Evaluation Parameters

Parameter	Value	Parameter	Value
$\|D\|$	20, 40, 60, 80 nodes	Frequency f_c	2 GHz
N	10	Area	1×1 km^2
Q_{max}	85, 90, 95 dB	L cell size (initial)	100×100 m^2
R_{d_i}	100 req/s	L cell size (dynamic)	50×50 m^2
$hops_{max}^{cc}$	3, 5 (dyn. scenario)	C^{cap}	5000 req/s
w_1, w_2, w_3, w_4	1		

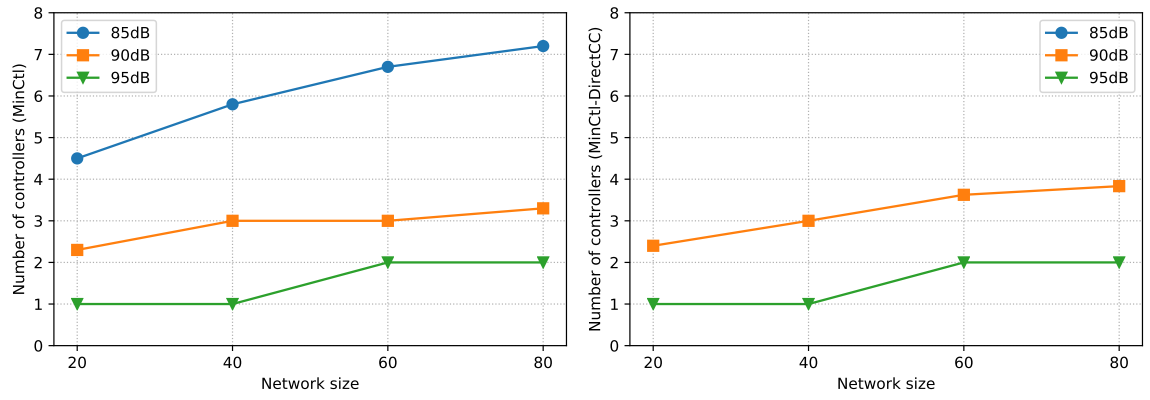
drone networks and drone computing capacities. The communication channel range is controlled by different values for Q_{max}.

5.4.1 Initial Deployment

Evaluation Setup

To evaluate initial deployment schemes, we ran experiments for different network sizes $|D|$ as shown in Table 5.1. For each network size, three scenarios associated with three values of Q_{max} were evaluated, which represent limited, medium, and large communication ranges for both control links and inter-controller links, affecting the difficulty of finding optimal placements. For each network size, we evaluated the MinCtl and FixedCtl schemes. We also compared the MinCtl scheme with a baseline scheme that only allows direct inter-controller links (referred to as MinCtl-DirectCC) to show the flexibility of indirect inter-controller links using MinCtl. As well, the constraint for limiting the hop count for inter-controller links was omitted to allow for finding and compare optimal solutions without this restriction.

The drone network topology used in evaluations are randomly generated graphs, where in each graph instance, nodes are positioned uniformly in the simulation area. Links between nodes are formed between every node pair that can achieve a path loss equal or below 90 dB, while the maximum node degree is limited to 4 for all nodes. Only connected graphs were selected. All results shown are averages of 10 trials (a different graph for each trial).



(a) MinCtl scheme

(b) MinCtl with direct inter-controller links

Figure 5.2: The number of controllers for MinCtl scheme

Results

Figures 5.2a and 5.2b show the number of deployed controllers using MinCtl and MinCtl-DirectCC, respectively with different network sizes and Q_{max} requirements. Both schemes produce a similar number of controllers depending on the network size and Q_{max} requirements. However, using indirect inter-controller links with MinCtl gives an advantage over MinCtl-DirectCC and can result in a slightly lower number of controllers for $Q_{max} = 90$ dB. The flexibility of MinCtl is shown for $Q_{max} = 85$ dB where MinCtl can find a controller deployment whereas MinCtl-DirectCC cannot find a feasible deployment of controllers. Hence, there is no plot for $Q_{max} = 85$ dB in Figure 5.2b. Figure 5.3 shows the ratio of the number of controllers to the network size using the MinCtl scheme.

Figure 5.4 shows the average hop count for inter-controller links. Generally, a low number of hops is achieved. For $Q_{max} = 95$ dB, no plot is shown for network sizes 20 and 40 as only a single controller is deployed and no inter-controller links exist.

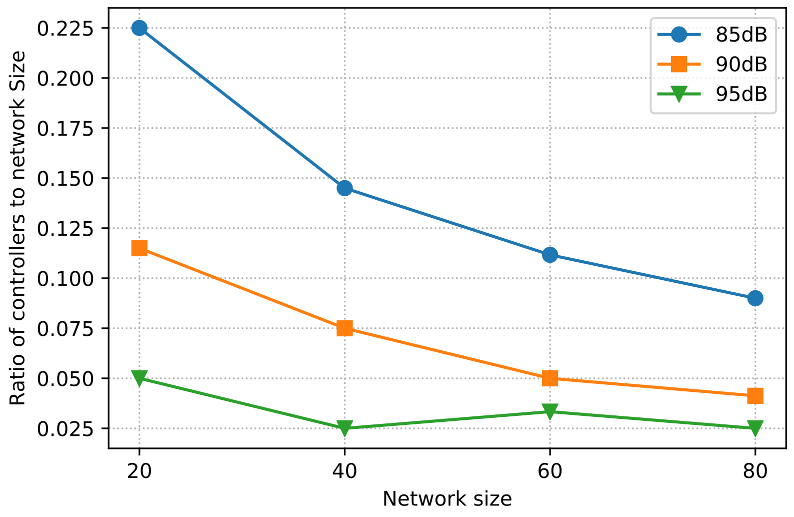


Figure 5.3: Ratio of the number of controllers to the network size - MinCtl

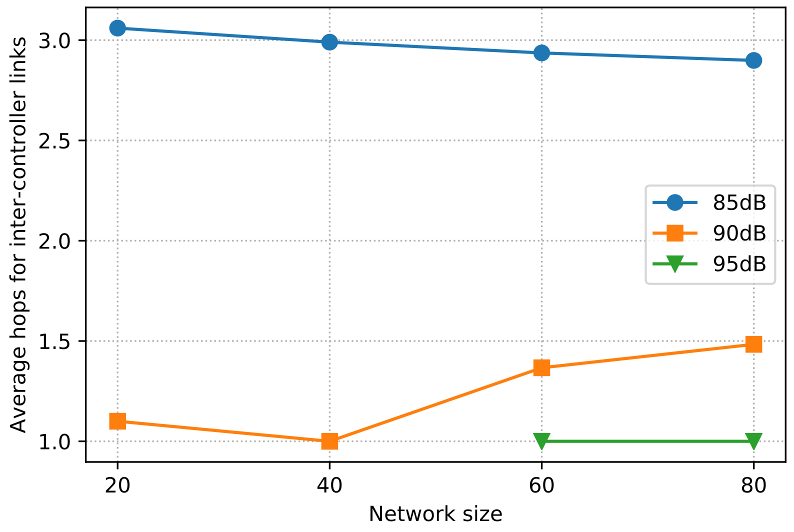
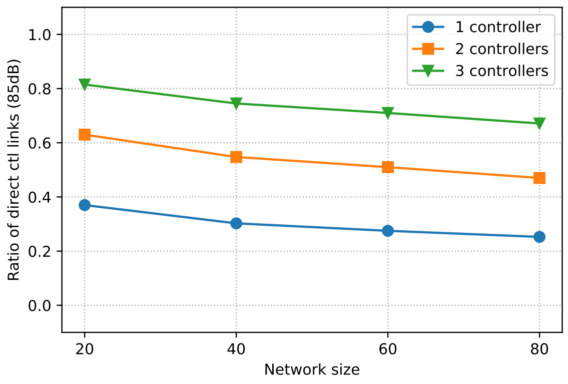


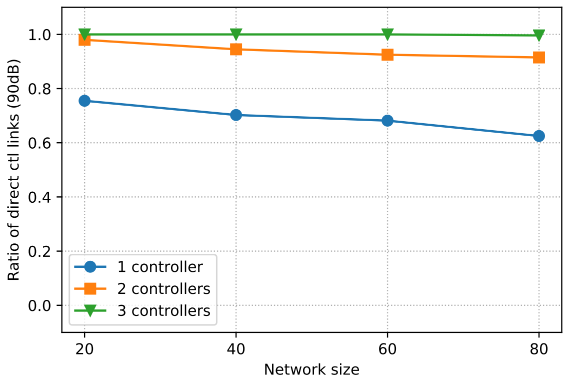
Figure 5.4: Average number of hops for inter-controller links - MinCtl

For network sizes 40 and 60, two controllers are deployed with $Q_{max} = 95$ dB (which require inter-controller links). For $Q_{max} = 90$ dB, the average hop count is between 1 and 1.5 hops. However, for the more restricted range ($Q_{max} = 85$ dB), an average of 3 hops is required, which offers flexibility of having connected controllers even when direct inter-controller links are not possible.

The FixedCtl scheme is evaluated with different network sizes and numbers of controllers $N = 1$, 2, and 3. We also set the values for Q_{max} to 85 dB and 90 dB only as they are the more restrictive cases.



(a) $Q_{max} = 85$ dB



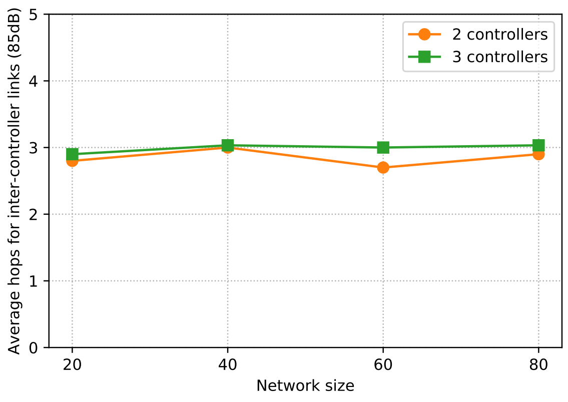
(b) $Q_{max} = 90$ dB

Figure 5.5: Ratio of network nodes assigned direct control links - FixedCtl

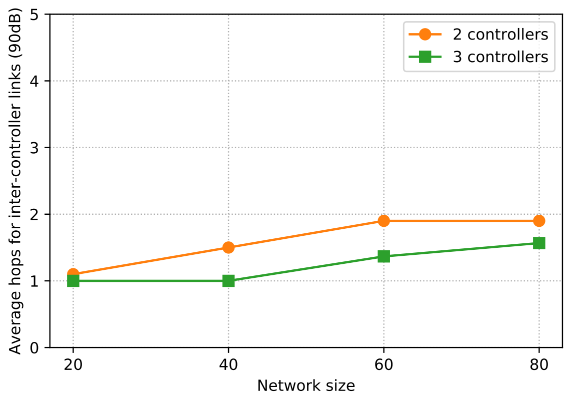
Figures 5.5a and 5.5b show the ratio of assigned nodes when deploying 1, 2, and 3 controllers using the FixedCtl scheme. It can be observed that with a limited range (85 dB) the scheme can reach direct link coverage ratio between 0.2 and 0.8 depending on the number of controllers. With a higher range (90 dB) almost all nodes are assigned direct control links except when using a single controller.

Figures 5.6a and 5.6b show the average hop count for inter-controller links. Similar to the MinCtl scheme, a single-hop inter-controller connectivity can be achieved with the range of 90 dB, and a limited number of hops is required with the more limited range of 85 dB.

We demonstrate the running time of the MinCtl and FixedCtl schemes in Figure 5.7. The figure shows that the running time (measured as wall clock time) increases



(a) $Q_{max} = 85$ dB



(b) $Q_{max} = 90$ dB

Figure 5.6: Average number of hops for inter-controller links - FixedCtl

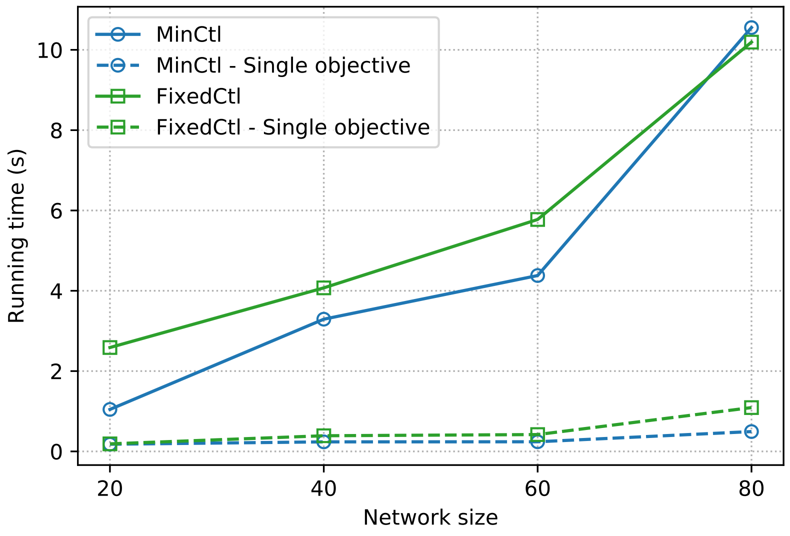


Figure 5.7: Running time for initial schemes

with the network size. It was found that most of the time is consumed by the second objective of these schemes which is minimizing the hop distance of inter-controller links. For comparison, we show in the same figure the running time for both schemes when only involving a single objective (the first objective), which is minimizing the number of controllers for MinCtl and maximizing assigned nodes for FixedCtl. When employing a single objective, the running time is significantly reduced and also increases slightly with the network size.

5.4.2 Dynamic Adjustment

Evaluation Setup

For evaluating the dynamic schemes, we used network topologies generated in the same manner as the evaluation of initial deployment as starting topologies. The cell size for controller placement was set to 50×50 m^2 as it allows for optimizing the relocation distance. The remaining parameters are listed in Table 5.1. Q_{max} values were set to 85 dB and 90 dB as they are the most challenging due to their limited range.

Each simulation run starts with an initial placement scheme applied to the starting topology. After the initial controller placement, 10 successive topology update events are applied to simulate changes in network node locations. In each event, a random number of drones is selected and moved to random points within a 300×300 m^2 area around each drone that moved. This is to simulate a somewhat local movement as it is not intended for drones to move far across the network area. The dynamic placement scheme is triggered with each network update. Update events are assumed to take place after completion of all movements and placement changes of the preceding update.

The Dyn-MinCtl scheme was evaluated with three network sizes $|D| = 20$, 40, and 60 nodes, each with $Q_{max} = 85$ dB and 90 dB. The Dyn-FixedCtl scheme was evaluated with $|D| = 40$ and varied number of controllers ($N = 1$, 2, and 3) and Q_{max}

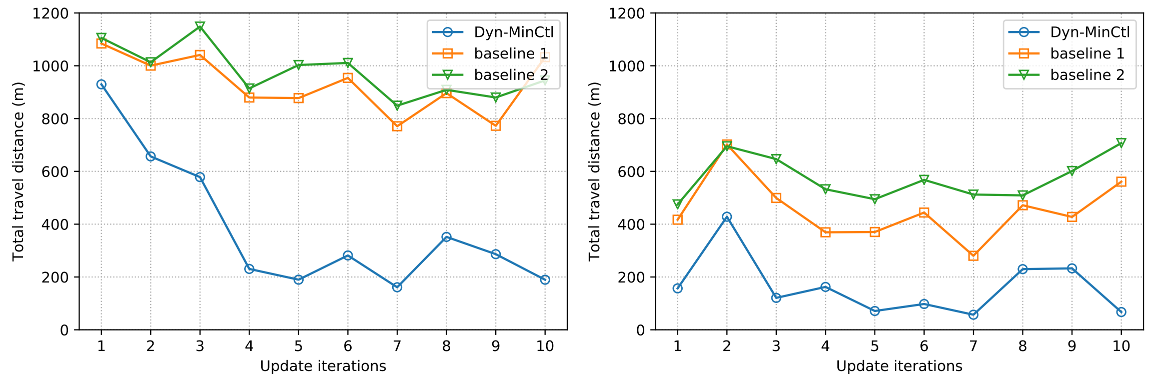
values of 85 dB and 90 dB. The limit on hop distance for inter-controller $hops^{cc}_{max}$ was set to 8 for $Q_{max} = 85$ dB and 3 for $Q_{max} = 90$ dB.

To evaluate the effect of distance minimization, Dyn-MinCtl and Dyn-FixedCtl were compared with two variants of the same scheme. The first, referred to as **baseline 1**, involves the reassignment step and unrestricted placement. The other, referred to as **baseline 2**, only involves unrestricted placement.

Reported results are averages of 10 different runs for each combination of scenario parameters.

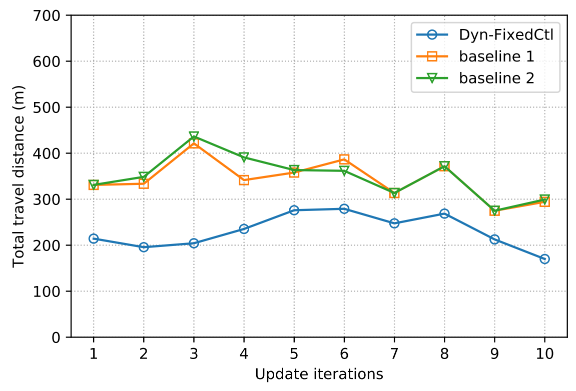
Results

First, we demonstrate select instances of the dynamic scenarios. We show how such scenarios progress while applying the dynamic schemes in response to network topology updates. We show the total distance for controllers to transit to new placements following each network topology update.

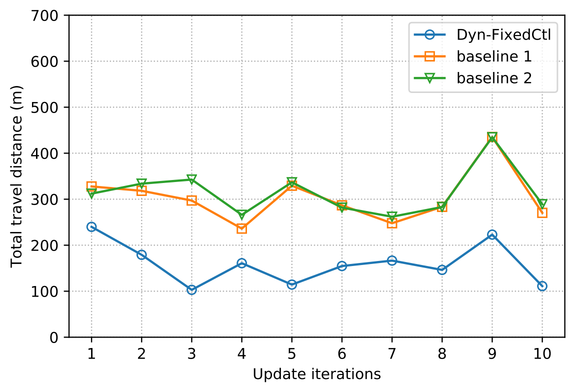


(a) $Q_{max} = 85$ dB (b) $Q_{max} = 90$ dB

Figure 5.8: Total distance to adjust controllers after each network update. Dyn-MinCtl and Network size $|D| = 40$.



(a) $Q_{max} = 85$ dB



(b) $Q_{max} = 90$ dB

Figure 5.9: Total distance to adjust controllers after each network update. Dyn-FixedCtl, Network size $|D| = 40$ and number of controllers $N = 2$.

Figures 5.8a and 5.8b show the travel distance after each topology change using the Dyn-MinCtl scheme for two connectivity ranges. The scheme achieves the minimum distance compared the baseline schemes due to the reassignment step and the minimization of distance when computing an updated placement. Baseline 1 performs better than baseline 2 due to making use of the reassignment step which avoids the need to move controllers when the link range is permissive. However, both baselines tend to perform in a similar way or very close to each other with the limited range.

The same discussion applies with the Dyn-FixedCtl schemes as shown in Figures 5.9a and 5.9b. In this scenario, movement is not required as much as the previous scenario as the number of controllers and their connectivity are limited. The baselines perform more closely as the reassignment step is not useful most of the time due the limited connectivity of the few controllers. The Dyn-FixedCtl scheme is able to reduce the distance much more than the baselines due to the optimization of the transition distance.

The above scenarios show that when minimizing the transition distance, the control network can readjust quickly and possibly save energy while maintaining connectivity to network nodes after network topology updates.

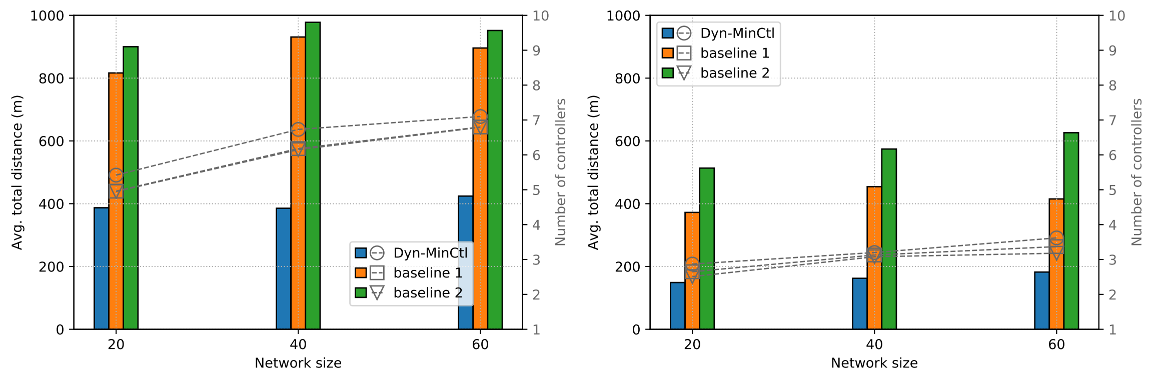
Next, we compare the average total distance travelled by controllers on aggregate for all evaluated scenarios. Figure 5.10a shows the total distances of Dyn-MinCtl and the baseline schemes for $Q_{max} = 85$ dB. The associated average number of controllers produced by the schemes is shown on the right y-axis. As shown in the figures, Dyn-MinCtl scheme achieves low transition distances, more than half the distances required by the baseline schemes. Figure 5.10b shows the case with $Q_{max} = 90$ dB. The permissive communication range leads to deploying a limited number of controllers and a smaller transition distance than the aforementioned scenario. As expected, the optimal scheme significantly reduces the transition distances compared to the baseline schemes. Baseline 1 performs better than baseline 2 due to making use of the reassignment step, which is more applicable with a wider communication range.

Note that the Dyn-MinCtl scheme appears to require slightly more controllers on average. The reason for this is that Dyn-MinCtl tends to maintain a consistent number of controllers due to the minimal transition placement step, which restrict the number of controllers from changing. On the other hand, the baseline schemes involve the unrestricted placement step (reapplying the initial placement), which, in some update events, will deploy a smaller number of controllers than what was required in preceding events. The baselines continuously produce the minimal number of controllers required for the current network topology and deploy a smaller number of controllers on average.

As for the Dyn-FixedCtl scheme, as shown in figures 5.11a and 5.11b, we show the total distances along with the average assignment ratios produced (right y-axis). This scenario was evaluated with 40 network nodes while varying the fixed number of controllers, which results in different assignment ratios. Similar to the former scheme, the optimal scheme reduces the transition distances, while the baselines perform in a similar way to each other, as the reassignment step is not useful unless all nodes are assigned ($N = 3$ controllers and $Q_{max} = 90$ dB), in which case, the proposed scheme takes advantage of both reassignment and distance minimization, leading to shorter distances than $N = 2$.

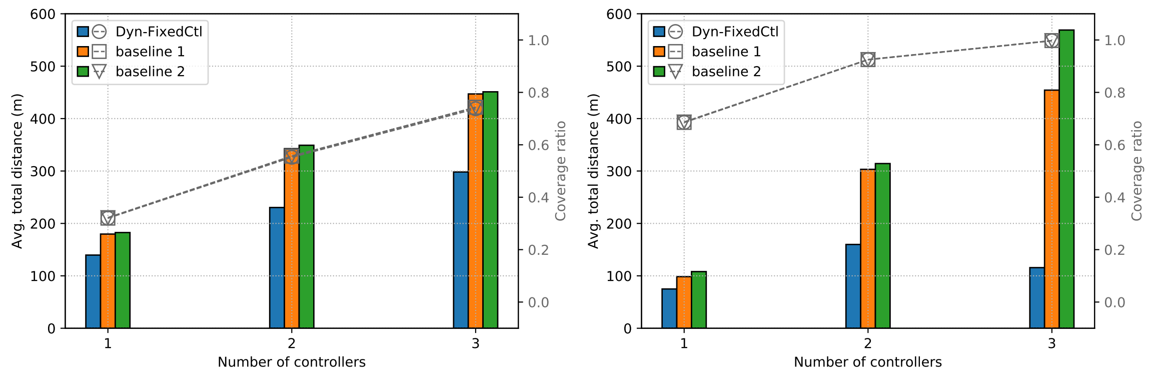
It is important to note that performance of the optimal schemes (both Dyn-MinCtl and Dyn-FixedCtl) is mainly due to the distance minimization step. The reassignment step is useful for avoiding unnecessary computations when simple reassignment is sufficient. On average, for the MinCtl scenarios, distance minimization is involved in updating the placement 67% of the time with the limited range and 49% of the time with the wide range. For the FixedCtl scenarios, distance minimization is involved 94% and 82% of the time for limited and wide ranges, respectively.

Lastly, we show the running time for the dynamic scheme when employing the distance minimization step. The running time for Dyn-MinCtl scenarios is shown in Figure 5.12a. It increases with the network size, but maintains a very short running time under 2 seconds. The running time for the Dyn-FixedCtl scheme is shown in Figure 5.12b for the scenario discussed earlier in the evaluation (Fixed network size $N = 40$ and varied number of controllers), where the running time was found to be higher. This is due to the nature of the objective function which involves maximizing the assignment and minimizing the transition distance. There also can



(a) $Q_{max} = 85$ dB (b) $Q_{max} = 90$ dB

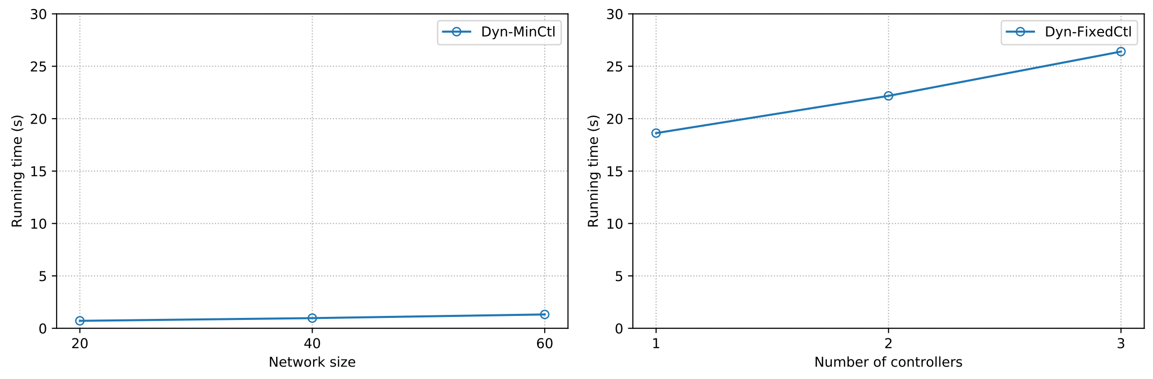
Figure 5.10: Total transition distance for controllers (left y-axis) and the number of controllers (right y-axis) - Dyn-MinCtl



(a) $Q_{max} = 85$ dB (b) $Q_{max} = 90$ dB

Figure 5.11: Total transition distance for controllers (left y-axis) and the coverage (ratio of drones assigned direct control links) (right y-axis) - Dyn-FixedCtl

be different solutions that yield the same objective value, for which the solver spends more time looking for the optimal solution. This may require further investigation and experimentation with different formulations or an algorithm that reduces complexity.



(a) Dyn-MinCtl (b) Dyn-FixedCtl

Figure 5.12: Running time for the distance minimization step of Dyn-MinCtl and Dyn-FixedCtl

5.5 Discussion

In this section, we discuss our results and offer insights into future work.

Employing two objectives for the initial scheme is ideal to achieve optimal deployment for both objectives. However, the second objective (minimizing inter-controller hop distance) led to additional running time. In practice, it is best to set an empirical limit on the hop distance using constraints given the characteristics of the network and control channels. As a result, the running time is reduced as well.

The FixedCtl and Dyn-FixedCtl schemes are useful when only a fixed and limited number of controllers can be deployed. However, the benefit of Dyn-FixedCtl comes at the cost of some additional running time. A possible improvement is maximizing the assignment to nodes that require communication with controllers the most. The next step after obtaining a partial assignment involves allocating control channels to the remaining nodes through the network and dynamically adjust such allocations. Making such allocations requires the visibility of the SDN controller into the network traffic. This can be a topic of future study.

The proposed schemes can be modified to serve in more general use cases. For instance, a modified scheme can be used as a form of topology control to relocate more important and central nodes and connect to the remaining nodes of the network.

The placement problem is a variant of the facility location problem, where controllers are the facilities, and network nodes are the clients with fixed demands. This problem and its dynamic variants has been shown to be NP-hard [FH09]. Fast heuristic algorithms are desired for solving larger problems due to the exponential complexity of solving large integer programs with an increasing number of variables and constraints. However, in this work, the goal is to show the objectives and requirements for deploying dedicated nodes for control plane functions and outline the general procedures to deploy and maintain controllers optimally, as well as to demonstrate the results and applicability for the specified scenarios. Solutions were mostly obtained in a reasonable time using a commercial solver. Future work can look into developing fast and scalable heuristic algorithms. This poses a challenge due to the difficulty in capturing the multiple requirements of dynamic deployment.

5.6 Summary

In this chapter, we presented controller deployment schemes for drone-based software defined networks that require deploying dedicated drones for SDN control. The proposed schemes ensure different requirements for such deployments, including the assignment of direct control links as well as inter-controller links. A flexible approach is utilized to ensure inter-controller connectivity by prioritizing direct single-hop links and allowing short indirect links through the drone network when required. An initial deployment scheme is proposed to deploy and assign the minimum of controllers to network nodes. A dynamic adjustment scheme is proposed to relocate the deployed controllers and update controller assignments while minimizing the distance required for controllers to travel after each network topology update. A variant of each of the above schemes is proposed for cases where only a limited number of drones is allowed to function as controllers. The evaluation of the initial deployment schemes showed the possible optimal deployments for aerial SDN controllers for drone networks. Evaluation of the dynamic schemes showed that the transition distance of controllers can be significantly minimized following network topology updates, leading to minimizing the controllers' travel delay to maintain control links, and potentially preserving the energy required for controllers to travel during adjustment.

Chapter 6

Deploying NFV-Based Drone Networks

In this chapter, we address the need for deployment and SFC placement schemes for NFV-based drone networks. After demonstrating applicable scenarios, we propose schemes for static deployment and dynamic orchestration of drone networks composed of SFCs with location requirements. The proposed schemes determine the optimal network topology and optimal placement of VNFs and traffic paths required by SFCs. As well, dynamic orchestration reconfigures the deployed drone network with minimal overhead.

6.1 Introduction

In situations where a communication network is required but where a ground network infrastructure is not available, the use of drones equipped with communication capabilities to deploy a temporary network is a plausible solution [Moz+19; May+19; SK18]. This would be especially useful for scenarios involving remote sensing, rural broadband access, and search and rescue operations [Seç+20] particularly since deploying a fixed infrastructure for temporary situations is costly. While deploying drone-based backhaul networks for emergency situations has been discussed in the literature [Par+16; SK18], there is further merit in designing a flexible network architecture for multi-task drone network deployments.

In some drone network scenarios, performing complex computational tasks, such as analysis of collected data, is often required to assist the mission. Such computing tasks can be offloaded to the edge network and cloud infrastructure. However, given that missions are deployed in remote areas that lack access to ground network resources, utilizing computing capabilities mounted on drones is promising. NFV allows for provisioning network services as SFCs, where each SFC is composed of a series of VNFs that implement network packet processing functions. VNFs are easily deployed on virtualized hardware. As well, VNFs can be used to implement various aspects in addition to networking, such as flight control and operating sensors mounted on drones. NFV enables mission operators to reconfigure drones with multiple services in that way operators use the same drone for multiple missions. This would allow for different novel use cases, such as deploying emergency voice communications and video surveillance services, provided VNFs implement such functionalities.

In order to facilitate and utilize the capabilities stated above, we propose planning drone network deployments by expressing missions as a set of SFCs that implement the mission functions. Each SFC is composed of modular VNFs for data acquisition, processing, and transmission. Data is collected at certain locations in the mission area and then transmitted to target locations while being processed by a number of VNFs in a certain order. Based on such requirements, optimal planning is needed to determine the size of the network, node locations, topology, and the placement of SFCs within the network topology, which includes placing VNFs and routing traffic between them. The goal is to create a network capable of carrying traffic across the mission area. Furthermore, network and mission mobility requires continuously adapting the network topology and SFC placements. This introduces an additional challenge, as continuously relocating VNFs and rerouting traffic between them may introduce additional overhead, which must be minimized while ensuring the network meets SFC requirements.

The contributions of this chapter are as follows:

- We propose a joint drone network deployment and an SFC placement scheme for planning and deploying NFV-based drone networks. The goal is to construct a minimal drone network able to host a given set of SFCs that define the network functions. The scheme minimizes the number of deployed drones and routes SFC traffic through a minimum hop distance. The scheme is intended for deploying a stationary or initial NFV-based drone network.

- We design a greedy algorithm as a faster alternative to the aforementioned scheme. The algorithm constructs a network as required to host the given SFCs and yields near optimal results.

- We propose a dynamic orchestration scheme for maintaining the network while drones are moving according to changing SFC location requirements. The scheme adjusts the topology and SFC placement of the deployed network to accommodate the movement of drones while performing tasks as mandated by SFCs. The scheme minimizes the possible overhead of VNF relocations and rerouting of SFC traffic due to changes in network topology.

The remainder of this chapter is organized as follows. In the following section, we review the relevant literature and the limitations that motivate this work. In

Section 6.3, we describe example scenarios that could be served by the proposed system. Then, we define the problem and discuss architectural considerations and assumptions. In Section 6.4, we present the system model, and in Section 6.5, we describe the static deployment scheme, and provide an alternative greedy algorithm for static deployment. In Section 6.6, we describe the dynamic orchestration scheme. The evaluation setup and results of both static deployment and dynamic orchestration scenarios are presented in Section 6.7. Finally, in Section 6.8, we summarize this chapter.

6.2 SFC orchestration in the Literature

SFC placement or orchestration is an actively explored area in NFV literature in wired data centers and cloud networks. In such settings, the SP receives service requests to provision network services represented as SFCs according to QoS requirements. This is known as orchestration, and it is accomplished by embedding, or placing SFCs within the physical network topology and possibly adjusting previously provisioned SFCs. SFCs are placed within a physical network by mapping VNFs to physical nodes (servers) and mapping or routing links between consecutive VNFs through physical network links. Placements can be made with respect to different objectives [HB16]. For example, the SP seeks to make optimal allocation of resources to provision SFCs in order to satisfy the demand while reducing costs related to the number of servers and their energy consumption [Bar+16], satisfying QoS in terms of throughput or delay [Taj+19], and increasing utilization and profit by accepting most requested services [Wan+20a].

In Section 2.2.2, we reviewed the relevant literature in which authors proposed use cases and applications that utilize NFV in drone networks. In [RS17] a numerical analysis of the network computational load was provided for VNF assignments, while in [Nog+18], the feasibility and performance of deploying VNFs on drones equipped with small computing boards were evaluated through a practical implementation. In other proposals, drones are connected to and supported by VNFs hosted in ground infrastructure (e.g., a cloud platform) [BBT19; Whi+17], where such VNFs provide different services that include monitoring and control services. The focus of these studies was on the optimal placement of VNFs in ground infrastructure to meet the connectivity requirements of moving drones.

While the aforementioned works considered NFV in applications of drone networks, such works assume a fixed network topology and/or VNF assignments. As well, they considered different use cases and objectives. To the best of our knowledge, no other works have considered the problem of jointly deploying the physical drone network and allocating resources for VNFs with chaining requirements. This work is unique in that the proposed system determines the physical network size and topology needed to embed the required SFCs within the physical network. Our system is composed of a static deployment scheme that targets initial or stationary network deployment and a dynamic variant that targets orchestrating the network with mobility requirements. The latter maintains the network in a manner that limits possible overhead resulting from network reconfiguration.

6.3 Problem Description

In this section, we describe an example use-case that motivates this work, then outline the problem definition and the considered network architecture and assumptions.

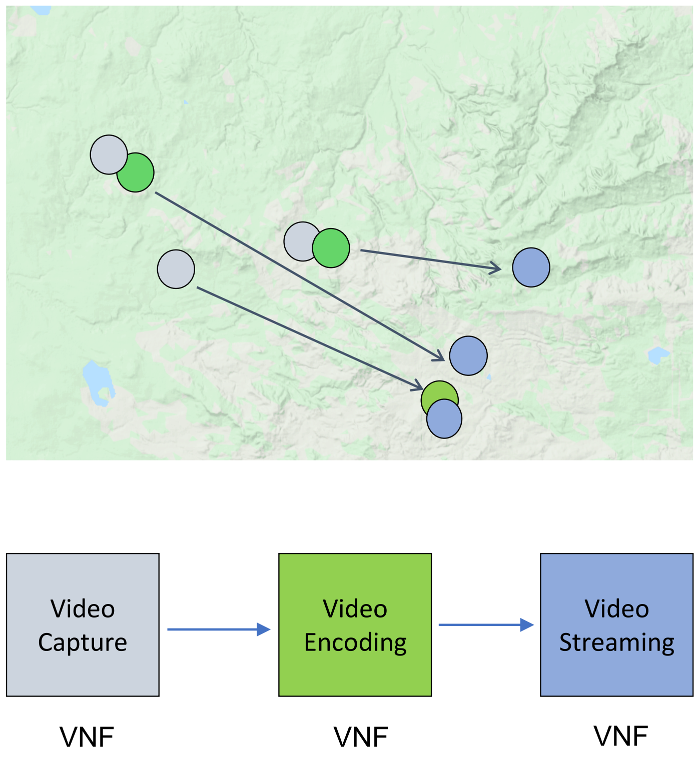


Figure 6.1: A use case of a video monitoring mission defined using an SFC

6.3.1 Use Cases

A remote monitoring mission can express its network and computation tasks as SFCs. These are composed of a series of VNFs that perform network and processing tasks while traffic passes through VNFs to reach its destination. An example use case is a mission involving monitoring and video streaming. Figure 6.1 depicts the associated SFC of this mission. The video monitoring SFC consists of VNFs for video capture, video transcoding, and video streaming [RS17]. These functions can be deployed where video monitoring traffic will be initiated from a video capturing drone deployed to an area of interest. The video monitoring drone hosts the initial video capture function. The final streamer function can be hosted on the video monitoring drone, or a different drone depending on the SFC location requirements or based on user locations. Intermediate functions can be placed in any of those drones or intermediate drones if traffic flows through them, depending on available computing resources. To illustrate, Figure 6.2, demonstrates a drone network with a placement of a number of SFCs. Consider SFC 1 as a video monitoring chain, where VNF 1 implements video capture, VNF 2 implements video transcoding, and VNF 3 implements streaming.

Other use cases can be served by this architecture. One may involve a telecom operator deploying drones with MEC capabilities as temporary micro BSs to enhance coverage at crowded sporting events or festivals [Bou+19]. The operator can deploy services with low-latency requirements as SFCs close to the covered users. Services can include serving or caching popular or event-related social media content [Che+17] and computation offloading [Zhe+19; Wan+20c]. In this instance, source locations in Figure 6.2 can be considered the gateway to the operator's core network. Another example is a drone-based cloud-like service [Nog+18] for emergency networks. Such a system may involve different VNFs functions as well as mobility requirements to serve mobile users.

Due to the ease of configuring drones with different VNFs, computing resources offered by drones can be shared between multiple SFCs, possibly as replicas of the same chain to video monitor multiple areas. For instance, other SFCs with different functionalities may be paired with the video monitoring chain, to provide other types of sensing or network functions besides video monitoring, leading to flexible and modular composition of the mission.

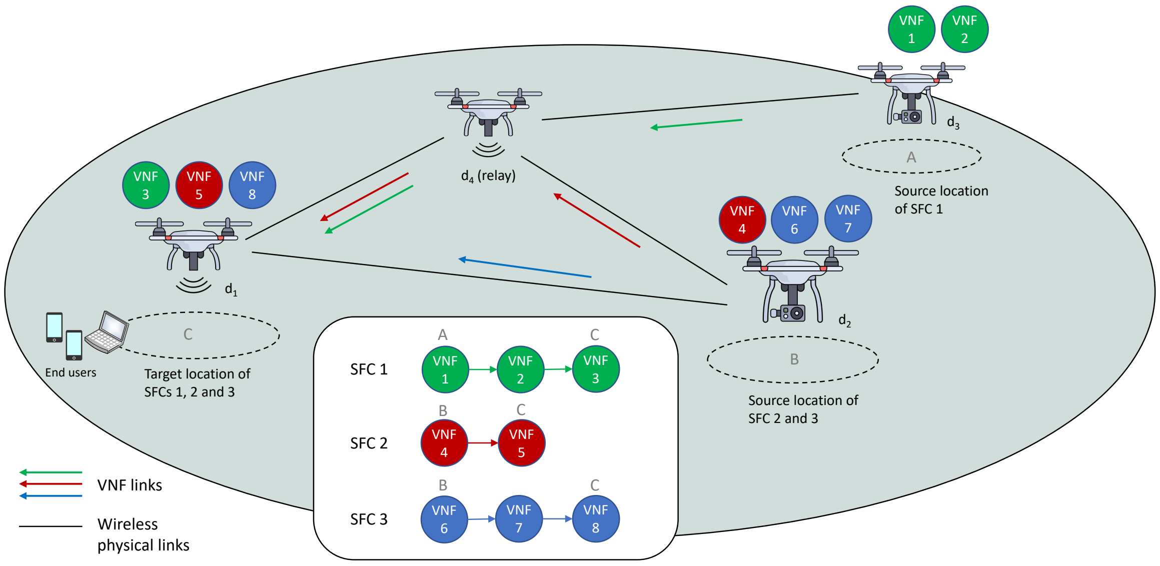


Figure 6.2: An example of a drone network deployment and SFC placement given a set of SFCs. Each SFC consists of chained VNFs and location requirements for first and last VNFs. Drones are deployed to locations required by SFCs (d_1, d_2, and d_3 are primary drones). For network connectivity, links are formed between drones, and relay drones are deployed if needed (d_4 is a relay). SFCs are placed within the physical network.

6.3.2 Problem Definition

Static Deployment

We are given the set of SFCs R, each with their requirements: VNF processing capacities, required link throughput, and source and target locations. All available drones and their capacities are also given. The aim is to jointly accomplish the following:

- Construct a drone network formation with the minimum number of drones and with the required connectivity and capacity to host the given SFCs. Specifically, the topology is generated so that drones, referred to as primary drones, are placed at the source and target locations of SFCs, while ensuring connectivity is established between them. Relay drones are deployed if needed.

- Place or embed SFCs (VNFs and VNF links) within the generated physical network. This includes the placement of the VNFs on drones computing resources and routing SFC traffic to support the chaining of VNFs where VNF links are within the capacity of physical network links. We also minimize the hop distance for SFC traffic flows to reduce delay.

We propose an optimal scheme and an alternative greedy algorithm for static deployment. Figure 6.2 illustrates a set of SFCs given and the resultant drone network formation and SFC placement.

Dynamic Orchestration

We target the case where SFCs have mobility requirements where SFCs change their source locations (to, e.g., capture video from different areas) while target locations remain stationary serving as sinks for the traffic. In this manner, drones hosting the first VNFs of SFCs transition over a series of locations during the mission as SFCs update their source locations.

Dynamic orchestration is needed to adjust the network topology and SFC placement in response to location updates. Primary drones are moved to the updated source locations. If this mobility results in link disconnections or changes in link capacities, then SFC traffic flows are disrupted. Dynamic orchestration then adjusts the network topology and SFC placements to maintain the operation of the network during drone movements. It does so while maintaining the network size and minimizing hop distances for SFCs. However, the network is expected to experience many adjustments over time, which can lead to disruptive overheads such as VNF relocations and rerouting of SFC flows. Such overhead should be minimized to limit disruptions to network functions. We propose a dynamic orchestration scheme tasked with adjusting the network topology and SFC placement during drone movements with minimal overhead (i.e., with minimal changes to SFC placements).

6.3.3 Architecture and Assumptions

In the context of the proposed softwarized drone network architecture, the static deployment scheme and algorithm are implemented as a module of the network deployment modules. The dynamic scheme is part of SDNC. We assume that either a ground-based or aerial controller is connected to the network and is able to reconfigure the network using the dynamic scheme proposed in this chapter.

Without loss of generality, we assume that there are sufficient computing resources available using on-board computing boards. Such hardware can be used along with container-based virtualization [Fel+15] to host VNFs, as it is a light-weight alternative to full virtual machines. VNF images can be preloaded and configured on the on-board computers prior to the initial mission deployment and instantiated at deployment time.

While energy consumption is an important aspect in drone networks, we do not consider the energy consumed by drones in the dynamic scenario. We assume that another independent module handles the charging and scheduling of drone deployments. This can be considered in a future study.

6.4 System Model

A drone network hosting a set of SFCs is to be deployed in a mission area. The set of all allowable drone locations is denoted by $L = \{l_1, l_2, \dots\}$, where locations are in a 2-D horizontal plane and assuming a predetermined altitude. The set of all SFCs to be deployed on a drone network is denoted by $R = \{r_1, r_2, \dots\}$. Each $r \in R$ is associated with a directed path graph $S_r = (F_r, \mathit{VL}_r)$, which represents the chain's ordered VNFs $F_r = \{f_1^r, f_2^r, \dots f_{|F_r|}^r\}$ and the VNF links $\mathit{VL}_r = \{(f_m^r, f_{m+1}^r) \mid m = 1, \dots, |F_r| - 1\}$ that connect them. We denote by $s_r, \tau_r \in L$ the source and target locations required by an SFC r, where drones hosting r must be deployed. The required link throughput for any SFC r is denoted by δ_r, while $cpu_{f_m^r}$ and $ram_{f_m^r}$ represent the CPU cores and RAM capacities required by VNF f_m^r.

The aim is to construct the physical topology for the network from the set of available drones $D = \{d_1, d_2, \dots d_{|D|}\}$. We denote by $E = \{(d_i, d_j) \mid d_i, d_j \in D\}$ all possible links between drones. The final topology of the network to be constructed is represented by a graph $G = (\bar{D}, \bar{E})$, where $\bar{D} \subseteq D$ is the selected subset of drones and $\bar{E} \subseteq E$ is the selected subset of links. Each $d_i \in \bar{D}$ is deployed to a location $l_i \in L$. Selected drones $\bar{D}$ constitute two subsets of drones. The first is referred to as primary drones $\bar{D}_{prim}$, which are drones deployed to locations required by SFCs. The other is relay drones $\bar{D}_{rel}$, which are drones deployed to support network connectivity. Each $d_i \in D$ has a known capacity of computing resources in terms of the available CPU cores and RAM, expressed as cpu_{d_i} and ram_{d_i}, respectively.

We adopt free space propagation to model wireless links between drones due to the lack of obstacles in drone altitudes in remote areas. The path loss $PL(dist)$ in dB, over transmission distance $dist$ in meters between a pair of locations in L is as defined in 5.1, described in Chapter 5. We assume orthogonal channel allocation, where the capacity of a wireless channel between a location pair in bits per second is given by [CS17]: $C = B\ log_2(1 + \frac{P_t - PL(dist)}{P_n})$, where B is the channel bandwidth, P_t is the transmission power, and P_n is the noise.

Based on the channel model above, we define the following constants for every pair of locations in L. We define $pl_{l_k,l_l} \in \{0,1\}$ as equal to 1 if the path loss between a pair of drones placed at $l_k, l_l \in L$ is below the path loss threshold pl_{max}. We also define β_{l_k,l_l} as the achievable link capacity between the pair of nodes placed at l_k, l_l. We use the constants defined above to determine path loss and link capacity between any pair of drones in D when placed at any pair of locations in L.

6.5 Static Deployment

The static deployment scheme proceeds as follows. The scheme is given the set of SFCs R along with the VNF processing capacities, required link throughput, and source and target locations for each SFC. All available drones and their capacities are also given. An initialization step determines the placement of primary drones. Then, using integer linear programming (ILP), the scheme jointly determines the placement of relay drones, the formation of physical links between drones, and SFC placements.

6.5.1 Initialization

Since the locations required by SFCs R are given (source and target locations), we initialize the deployment by placing primary drones at these locations. This formalized as follows. Let $L_{req} \subset L$ denote the set of all locations required by SFCs, obtained as $L_{req} = \{s_r, \tau_r \mid \forall r \in R\}$. Then, we select subset from the available drones D and place each of these drones at the source and target locations as primary drones $\bar{D}_{prim}$. This placement results in a one-to-one mapping $M : \bar{D}_{prim} \rightarrow L_{req}$. Therefore, $M(d_i)$ yields the location $l_k \in L_{req}$ assigned to d_i. For convenience, $M(l_k)$ also yields the reverse assignment, that is, the drone $d_i \in \bar{D}$ placed at l_k.

6.5.2 ILP Formulation

The following decision variables determine the location of drones, including relays, and the physical links formed between them as well as the placement of VNFs on each drone and the assignment of VNF links to physical links. The decision variables are:

- $u_{d_i,l_k} \in \{0, 1\}$ indicates if d_i is placed at location l_k.
- $u_{d_i} \in \{0, 1\}$ indicates whether drone d_i is deployed regardless of selected location.
- $k_{d_i,d_j} \in \{0, 1\}$ indicates if the physical link (d_i, d_j) is active (selected) between the pair of drones d_i and d_j.
- $x_{f^r_m,d_i} \in \{0, 1\}$ indicates if VNF f^r_m is placed in d_i.
- $y^{f^r_m,f^r_{m+1}}_{d_i,d_j} \in \{0, 1\}$ indicates if VNF link (f^r_m, f^r_{m+1}) of SFC r is mapped or routed through the physical link (d_i, d_j).

The constraints for the ILP model are formulated as follows.

1) Drone Deployment and Placement: Primary drones are selected and positioned as dictated by the initialization step. This is done by setting the decision variables for drone deployment status and placement location :

$$u_{d_i} = 1, \quad \forall d_i \in \bar{D}_{prim} \tag{6.1}$$

$$u_{d_i,M(d_i)} = 1, \quad \forall d_i \in \bar{D}_{prim} \tag{6.2}$$

For all drones, the following constraints ensure that a drone is assigned to one location only, and each location is assigned to one drone only:

$$\sum_{l_k \in L} u_{d_i,l_k} = u_{d_i}, \quad \forall d_i \in D \tag{6.3}$$

$$\sum_{d_i \in D} u_{d_i,l_k} \leq 1, \quad \forall l_k \in L \tag{6.4}$$

2) Formation of Physical Links: Let $z(d_i, l_k, d_j, l_l)$ denote a binary decision variable that equals 1 if a pair of drones d_i, d_j, when placed at locations l_k, l_l, can achieve the required path loss pl_{max}. Values of said variable instances are determined using the following constraint:

$$\begin{aligned} z(d_i, l_k, d_j, l_l) = 1 \text{ iff } u_{d_i,l_k} + u_{d_j,l_l} + pl_{l_k,l_l} = 3 \\ \forall (d_i, d_j) \in E, \forall (l_k, l_l) \in L \end{aligned} \tag{6.5}$$

The following constraint activates links between a pair (d_i, d_j) based on the values of $z(d_i, l_k, d_j, l_l)$, which, if sum to 1, indicate the drone pair are placed at a location pair within range from each other to allow for establishing a physical links:

$$k_{d_i,d_j} \leq \sum_{(l_k,l_l) \in L} z(d_i, l_k, d_j, l_l), \quad \forall (d_i, d_j) \in E \tag{6.6}$$

Furthermore, we impose a limit on the number of physical links per drone (node degree), indicated by γ_{max}, to load balance traffic.

$$\sum_{(d_i,d_j)\in E} k_{d_i,d_j} \leq \gamma_{max}, \quad \forall d_i \in D \tag{6.7}$$

3) VNF Placement: The constraints for placing VNFs on drones are as follows. Using the placement of primary drones in the initialization step, we place the first VNFs of each SFC in drones placed at SFCs' source locations. Similarly, last VNFs are placed in drones at the SFCs' target locations. This expressed by the following constraints:

$$x_{f_1^r, M(s_r)} = 1, \quad \forall r \in R \tag{6.8}$$

$$x_{f_{|F_r|}^r, M(\tau_r)} = 1, \quad \forall r \in R \tag{6.9}$$

The following two constraints ensure that each VNF must be placed on a deployed drone, that a VNF must be placed on one drone only:

$$x_{f_m^r,d_i} \leq u_{d_i}, \quad \forall r \in R, \forall f_m^r \in F_r, \forall d_i \in D \tag{6.10}$$

$$\sum_{d_i \in D} x_{f_m^r,d_i} = 1, \quad \forall f_m^r \in F_r, \forall r \in R \tag{6.11}$$

The following constraints ensure that each drone hosts VNFs within its CPU and RAM capacities:

$$\sum_{r\in R} \sum_{f_m^r \in F_r} x_{f_m^r,d_i} \times cpu_{f_m^r} \leq cpu_{d_i}, \quad \forall d_i \in D \tag{6.12}$$

$$\sum_{r \in R} \sum_{f_m^r \in F_r} x_{f_m^r, d_i} \times ram_{f_m^r} \leq ram_{d_i}, \quad \forall d_i \in D \tag{6.13}$$

4) VNF Link Placement: In the following constraint, we ensure that links between VNFs are placed on active physical links:

$$y_{d_i,d_j}^{f_m^r, f_{m+1}^r} \leq k_{d_i,d_j} \qquad \forall r \in R, \forall (f_m^r, f_{m+1}^r) \in VL_r, \forall (d_i, d_j) \in E \tag{6.14}$$

As well, for each physical link (d_i, d_j), we ensure that the VNF links mapped to it do not exceed the capacity of the physical link:

$$\sum_{r \in R} \sum_{f_m^r \in F_r} y_{d_i,d_j}^{f_m^r, f_{m+1}^r} \times \delta_r \leq \sum_{(l_k,l_l) \in L} z(d_i, l_k, d_j, l_l) \times \beta_{l_k,l_l} \quad \forall (d_i, d_j) \in E \tag{6.15}$$

Finally, we ensure the connectivity between consecutive pairs of VNFs by placing VNF links along the physical links between VNFs. This is achieved using flow conservation constraints as used in [Bar+16]. For each d_i, we ensure that there is an equal in-flow and out-flow of VNF links between each consecutive VNF pair. The produced VNF links are mapped to physical links between d_i and its adjacent nodes in G, denoted by $n(d_i)$. The constraint is expressed as follows:

$$\begin{gathered} \sum_{d_j \in n(d_i)} (y_{d_i,d_j}^{f_m, f_{m+1}} - y_{d_j,d_i}^{f_m^r, f_{m+1}^r}) = x_{f_m^r, d_i} - x_{f_{m+1}^r, d_i} \\ \forall r \in R, \forall (f_m^r, f_{m+1}^r) \in VL_r, \forall d_i \in D \end{gathered} \tag{6.16}$$

5) Objective Functions: Our goal is to minimize the number of deployed drones to reduce the mission cost, and have the shortest possible paths for placed SFCs. The number of deployed drones, $N_{deployed}$, is calculated simply as:

$$N_{deployed} = \sum_{d_i \in D} u_{d_i} \tag{6.17}$$

The routes for SFCs can be minimized by minimizing the total number of VNF links, denoted by N_{vlinks} and calculated as:

$$N_{vlinks} = \sum_{r \in R} \sum_{\substack{(f_m, f_{m+1}) \\ \in VL_r}} \sum_{\substack{(d_i, d_j) \\ \in E}} y_{d_i d_j}^{f_m f_{m+1}} \tag{6.18}$$

Then, the complete ILP model is expressed as follows:

$$\min \quad w_1 N_{deployed} + w_2 N_{vlinks} \tag{6.19}$$

subject to constraints:

(6.1) - (6.16)

where w_1 and w_2 are weights used to reflect the importance of the respective objective function.

6.5.3 Greedy Algorithm for Static Deployment

For static deployments, the model described above can be replaced with a simple and fast greedy algorithm that can produce near-optimal results. In what follows we explain the basic idea of the algorithm, then we describe it in more detail.

Algorithm Overview

The algorithm starts by placing primary drones at all source and target locations required by SFCs. To place relay drones if needed, additional drones for relaying are placed at candidate locations. Then, the algorithm builds a nearly complete graph G_{full} of all placed drones D with all possible links that satisfy the path loss threshold.

A key challenge in the algorithm is determining locations for relay drones. Delaunay triangulation is used to produce candidate location points for relay drones similar to [HSL09; LH05]. The algorithm performs several iterations with different subsets of candidate locations until a complete solution is found.

In each iteration, the algorithm seeks to place SFCs in the shortest paths available in G_{full} while avoiding selecting paths with relay drones to avoid deploying relays when not needed to minimize the network size. The algorithm assigns low weights to paths involving primary drones only, and high weights to paths involving the additional relay drones. The algorithm then places SFCs (VNFs and VNF links) on the shortest weighted paths wherever possible. Selected links paths (links) are added to the target graph G_{target} along with any relay drones newly selected. If an SFC is placed on a path with relay drones, the weights assigned to this path are lowered so the path can be used to place remaining SFCs.

Algorithm Details

The algorithm is listed as Algorithm 7 and described in more detail as follows. The algorithm takes as input all SFCs R, their source and target locations, their throughput requirements, available drones D and their computing capacities. The output of the algorithm is the final graph G_{target} as the physical network, and the solution vectors

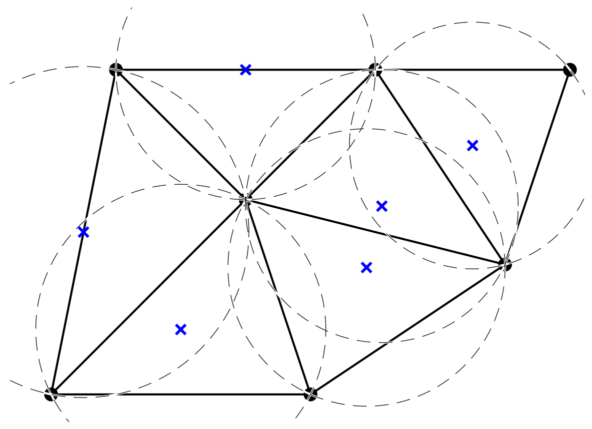


Figure 6.3: Delaunay triangulation with circumcenters shown in blue.

X and Y, which describe the assignment of VNFs and VNF links, respectively.

The algorithm starts by obtaining the set of all source and target locations required by SFCs denoted by $L_{req} \subset L$ (line 1). The algorithm then generates a set of candidate locations for relay drones, denoted by L_{rel} (line 2). Delaunay triangulation is used to generate triangles from all locations in L_{req} as demonstrated in Figure 6.3. For each triangle, the circumcenter point is calculated and added as a candidate location to L_{rel}. Only locations that can satisfy the threshold of pl_{max} with locations forming the triangle are added. In line 3, the algorithm divides L_{rel} into subsets each with length $\leq |D| - |L_{req}|$.

The algorithm performs a number of iterations (line 4). In each iteration, a different subset of untested relay locations, denoted by $\bar{L}_{rel} \subset L_{rel}$, is selected (line 4). In line 5, drones are assigned to both required and relay candidate locations, L_{req} and $\bar{L}_{rel}$. In line 6, the algorithm builds a connected graph G_{full} with drones $\bar{D}$ as nodes with all possible links within the pl_{max} threshold, given their assigned locations. The algorithm obtains the capacities of the constructed links in the graph. Then, in line 7, G_{target} is initialized with drones from $\bar{D}$ that are assigned to L_{req} and without

Algorithm 7 Greedy algorithm for static deployment

```
Input: $D$ with their capacities and $R$ with all requirements
Output: $G_{target}, X, Y$
1: $L_{req}$ = Get all source and target locations of SFCs.
2: $L_{rel}$ = Circumcenters(DelaunayTrig($L_{req}$))
3: Divide $L_{rel}$ into subsets each of length $\leq |D| - |L_{rel}|$
4: for subset $\bar{L}_{rel}$ of $L_{rel}$ do
5:     $\bar{D}$ = Assign drones to locations $L_{req}$ and $\bar{L}_{rel}$
6:     Build connected graph $G_{full}$ with $\bar{D}$ and get link capacities.
7:     Initialize $G_{target}$ with drones assigned to $L_{req}$
8:     for $r \in R$ do
9:         Update weights $W$ for $G_{full}$ using (6.20) and (6.21)
10:        $P$ = Get shortest simple paths in $G_{full}$ for $r$
11:        for $p \in P$ do
12:            if path $p$ is feasible for $r$ then
13:                Place $r$ in $p$ and update $X$ and $Y$
14:                Update $G_{target}$ with new drones/links from $p$
15:                break
16:            end if
17:        end for
18:        if $r$ is not placed then break
19:    end for
20:    if all $r \in R$ placed then
21:        return $G_{target}, X, Y$
22:    end if
23: end for
```

any links yet.

For each SFC $r \in R$, weights for edges in G_{full}, denoted by W are calculated (line 9), where W_{d_i,d_j} is the weight for the edge or link (d_i, d_j) in G_{full}. The weight calculation is done so that initially only the links between drones placed at the required locations (drones that exist in G_{target}) have low weights and thus are preferred for placing SFCs. Other links involving relay drones (not in G_{target} at the start of the algorithm) have greater weights to avoid or delay selecting them if they are not needed. If such links are selected in later steps, they are added to G_{target} and their weights are lowered so that they can be utilized for subsequent placement iterations. The weights for a link (edge) (d_i, d_j) involving drones d_i and d_j in the complete graph G_{full} are:

$$W_{ij} = w(d_i) + w(d_j) \tag{6.20}$$

where

$$w(d_i) = \begin{cases} 1 & \text{if } d_i \in G_{target} \\ 10 & \text{otherwise} \end{cases} \tag{6.21}$$

In line 10, the algorithm obtains from G_{full} paths P, the set of weighted shortest simple paths between the two drones placed at the source and target locations of r. Each path $p \in P$ is examined to test if it is feasible for placing r (line 11). In line 12, the feasibility is determined by testing the residual capacity in links and drones along p. This is based on capacities of drones and links in G_{full} and current placements X and Y, as well as whether selecting p will result in having nodes with links greater than γ_{max}. If p is a feasible path, r is placed in p (line 13). This step places VNFs of r in drones in path p sequentially. First, it places the initial and last VNFs in the first and last drones in the path. Then, intermediate VNFs are placed greedily in the first drone in the path or in subsequent ones as capacity is available. VNF links are placed in all links in p. Then, updated assignments for VNFs and links are reflected in X and Y. In line 14, G_{target} is updated with drones and links from p that were not previously in G_{target}.

If a single SFC r is not placed then the iteration fails (line 18). If all SFCs are placed, the algorithm is successful and G_{target}, X, and Y are returned (lines 20 and 21). At the step in line 23, if not all $r \in R$ are placed, then the current iteration is not successful. If there are remaining untested relay locations, then the algorithm proceeds to the next iteration to try a different set of relay locations. If no available locations can be tried, then the algorithm fails and exits.

Algorithm Complexity: Candidate relay locations are generated at the start of the algorithm. This step is performed once and it mainly depends on Delaunay triangulation which has complexity $O(NlogN)$ [HSL09], where N is the number of locations required by SFCs. The main outer loop performs a few iterations depending on the number of divisions of relay candidates. It can be limited by a maximum number I. In each iteration, the algorithm loops over all $r \in R$ to place SFCs. Assuming that each $r \in R$ has a unique source-target location pair, the algorithm obtains the K shortest simple paths in G_{full} for each $r \in R$ using an algorithm with complexity $O(K|D|^3)$ [Yen71; HSSC08], with $|D|$ as the number of nodes in G_{full}. The overall complexity of the algorithm excluding the initial triangulation is $O(IK|D|^3|R|)$.

6.6 Dynamic Orchestration

Next, we consider scenarios with mobility. Once a network is deployed initially using the static scheme, SFCs change their source locations repeatedly during the mission while target locations remain static where drones placed there serve as sinks for traffic, as discussed in Subsection 6.3.2. Once the next source locations for SFCs are determined and received by the orchestrator, dynamic orchestration is triggered to update the network if needed. We refer to such events as update events.

Dynamic orchestration proceeds as follows. The scheme takes as input the same input parameters as given to the static scheme with updated SFC source locations in addition to the network deployment and SFC placement decided in the previous update event or initial deployment. Similar to the static scheme, an initialization step first updates the placement of primary drones. Then, using ILP, the scheme updates the placement of relay drones, the formation of physical links between drones, and SFC placements.

6.6.1 Initialization

Let $\hat{\hat{D}} \subseteq D$ be the set of drones deployed in the last update event, where $\hat{\hat{D}}$ includes primary drones $\hat{\hat{D}}_{prim}$ and relay drones $\hat{\hat{D}}_{rel}$.

Primary drones are moved to the updated locations. Let $L_{req} \subset L$ contain the updated locations required by SFCs. Then, the deployed primary drones $\hat{\hat{D}}_{prim}$ are mapped to the updated locations L_{req}. The mapping M is now defined as $M : \hat{\hat{D}}_{prim} \to L_{req}$.

After determining the placement of primary drones, the orchestrator can test whether the network can keep the same links and SFC placement without compromising SFC requirements. This is done by updating the previous solution to reflect the new primary drone locations and checking if the constraints for link connectivity and capacity are violated. If there are no violations, the network retains the same links and SFC placements and avoids adjustment. If there are violations, the orchestrator proceeds to the next step.

6.6.2 ILP Formulation

Using the decision variables described in Section 6.5, we formulate the following constraints.

First, we maintain the deployment status of all deployed drones including relays:

$$u_{d_i} = 1, \quad \forall\ d_i \in \hat{\hat{D}} \tag{6.22}$$

Then, we position primary drones in the updated locations as indicated by the updated mapping M using the following constraints:

$$u_{d_i, M(d_i)} = 1, \quad \forall\ d_i \in \hat{\hat{D}}_{prim} \tag{6.23}$$

As well, first and last VNFs of each SFC remain in the same drone placement. This is expressed with the following constraints:

$$x_{f_1^r, M(s_r)} = 1, \quad \forall r \in R \tag{6.24}$$

$$x_{f_{|F_r|}^r, M(\tau_r)} = 1, \quad \forall r \in R \tag{6.25}$$

As locations of primary drones change, it is likely that locations of relay drones will need to change to ensure connectivity or capacity along SFC paths. It is useful to select new locations for relay drones that minimize the travel distance, and in turn, minimize the time required to adjust the network to the new topology. This is done by considering the previous locations of relay drones using the following constraint:

$$\sum_{l_k \in L} u_{d_i, l_k} \cdot dist(l_k, PrevLoc(d_i)) \leq dist_{max}, \qquad \forall d_i \in \hat{\hat{D}}_{rel} \tag{6.26}$$

where $dist(.,.)$ is the distance between a pair of locations in L, and $PrevLoc(d_i)$ obtains the previous location where d_i was deployed, and $\hat{\hat{D}}_{rel} \subset \hat{\hat{D}}$ is the subset of drones deployed as relays in the previous update. The maximum allowed distance is denoted by $dist_{max}$ which can be set to an arbitrary suitable value or to the largest movement step to be travelled by primary drones. This constraint is not required but it is useful when it becomes important to reduce or limit reconfiguration travel delay.

Objective Functions

The original two objective functions are required in the dynamic scheme. The first is keeping the network size minimal, which in a dynamic scenario, allows us to avoid deploying additional relay drones unless it is not possible to maintain network connectivity. The second is routing SFCs through the minimum hop distance, which in the dynamic setting serves to maintain short hop distances for SFCs.

In addition to the aforementioned objectives, the dynamic scheme aims to minimize the possible overheads resulting from adjusting the network topology. These include VNF placement changes, which are considered disruptive to the network traffic. Overheads also include changing the routing paths for SFCs due to changes in the network topology. In a highly dynamic setting, such overheads should be minimal, as network adjustments can take place frequently due to constant topology changes during drone movements.

Overhead can be minimized by minimizing the difference between previous and new decisions involving placements of VNFs and VNF links as well as network link states, namely, all decision variables x, y and k. As a result, changes to VNFs and VNF link placement as well as changes to the states of physical links will be minimized. This can be done by minimizing the distance between two vectors that involve previous and new states. The distance between such state vectors can be computed using the Hamming distance [FM14] which calculates the distance between values of two vectors in terms of the number of disagreeing pair-wise elements. Since all states are represented by 0 and 1 values, the Hamming distance represents the number of changed vector elements or changed states.

We formalize the above as follows. To simplify the notation, let X denote the ordered vector of VNF placement decision variables (i.e., all $x_{f^r_m,d_i}$ variables) and let $\hat{X}$ denote values of X in the previous state in the same order, and given as constants to the problem. The Hamming distance between VNF placement states X and $\bar{X}$ is:

$$\Delta(X,\hat{X}) = \sum_{\substack{1\le i\le |X| \\ \hat{X}_i=0}} X_i + \sum_{\substack{1\le i\le |X| \\ \hat{X}_i=1}} (1 - X_i) \tag{6.27}$$

where i is the index of the ith vector element. The same is done for VNF link placements, where Y is the vector of all $y^{f^r_m,f^r_{m+1}}_{d_i,d_j}$ decision variables and $\hat{Y}$ is the vector of their previous values:

$$\Delta(Y,\hat{Y}) = \sum_{\substack{1\le i\le |Y| \\ \hat{Y}_i=0}} Y_i + \sum_{\substack{1\le i\le |Y| \\ \hat{Y}_i=1}} (1 - Y_i) \tag{6.28}$$

and for variables involving physical link state decisions, where K and $\bar{K}$ encompass all k_{d_i,d_j} variables and their previous values, respectively:

$$\Delta(K,\hat{K}) = \sum_{\substack{1\le i\le |K| \\ \hat{K}_i=0}} K_i + \sum_{\substack{1\le i\le |K| \\ \hat{K}_i=1}} (1 - K_i) \tag{6.29}$$

As such, minimizing $\Delta(X,\hat{X})$ limits changes to placements of intermediate VNFs. Changes to SFC paths are reduced by minimizing changes to states of physical links $\Delta(K,\hat{K})$ and also by minimizing $\Delta(Y,\hat{Y})$ which restricts the solver from selecting routes different from previous ones because they are of the same length.

The objective function and constraints of the dynamic scheme is expressed as:

$$\min \quad w_1 N_{deployed} + w_2 N_{vlinks} + w_3 \Delta(K, \hat{K}) + w_4 \Delta(X, \hat{X}) + w_5 \Delta(Y, \hat{Y}) \tag{6.30}$$

subject to constraints:

(6.3) - (6.16)

(6.22) - (6.26)

where w_i are weights used to reflect the importance of the respective objective function.

After applying an update, the number of relocated VNFs is calculated as $\frac{1}{2}\Delta(X, \hat{X})$ since a Hamming distance of 2 involves a single VNF relocation from one drone to another. The set of rerouted SFCs in the last step $R_{rerouted}$ is defined as:

$$R_{rerouted} = \{\ r\ :\ |y_{d_i,d_j}^{f_m^r, f_{m+1}^r} - \hat{y}_{d_i,d_j}^{f_m^r, f_m^r}| = 1\ \forall r \in R\ \forall (f_m^r, f_{m+1}^r) \in VL_r\ \ \forall (d_i, d_j) \in E\ \} \tag{6.31}$$

where $\hat{y}_{d_i,d_j}^{f_m^r, f_{m+1}^r}$ is the value of the variable in the previous update. Thus, the number of rerouted SFCs is $|R_{rerouted}|$. The number of links which their states changed is $\Delta(K, \hat{K})$.

Dynamic orchestration can be solved using existing solvers in a reasonable time for the network instances considered in this work. Note that in this work we consider a limited degree of mobility in which drones will hover after each movement for a certain amount of time. In future work, we will consider developing a lower-complexity algorithm for dynamic orchestration.

6.7 Performance Evaluation

In this section, we present the performance evaluation of the proposed static and dynamic schemes and results.

The simulation environment and deployment algorithm were developed in a Python-based environment. The proposed optimization models for the static and dynamic orchestration were implemented using CPLEX (version 12.10) using the DOcplex Python interface. The NetworkX library [HSSC08] was used for graph algorithms. Simulations were run on a machine with a 2.60GHz Intel Core i7-9750H processor with 16GB of RAM.

We assume 10 drones are available for deployment, with fixed CPU and RAM capacities as shown in Table 6.1. The channel model parameters are shown in Table 6.1. The deployment area size is 1000 m $\times$ 1000 m. For tractability, the placement locations in L are discretized into a grid with cells or tiles each of size $w \times h$. The size of cells can be varied according to the density required for the task at hand and the size of the area. However, more density increases the complexity of solving the problem. We set it to 100 m $\times$ 100 m, which is a reasonable area size for drones to conduct tasks. For each pair $l_k, l_l \in L$, we compute β_{l_k,l_l} and pl_{l_k,l_l} according the propagation and capacity models described in Section 6.4.

For evaluations of the static and dynamic schemes, SFCs are generated as follows. The number of SFCs $|R|$ is varied between 4 and 16. SFC throughput requirements as well as the configurations of associated VNFs such as CPU requirements are drawn from a uniform distribution with the ranges shown in Table 6.1. For each comparison of $|R|$, we assume each pair of SFCs share a single source location. We select $|R| \,/\, 2$ locations, sampled from L as source locations for SFCs, where each pair of SFCs share the same source location thus the same primary drone. For target locations, all SFCs are assigned a single location at the corner of the simulation area, which means only a single target or sink node exists for the network. This leads to requiring $1 + |R| \, / \, 2$ primary drones for each evaluation of $|R|$, where the deployment scheme minimizes the number of relay drones to deploy in addition to the remaining objectives. Hence, the number of primary drones corresponding to $|R|$ is shown in the secondary (top) x-axis.

For the optimal static scheme, the solver time limit is set to 5 minutes, while for the dynamic scenario evaluation, it is set to 2 minutes. All reported results are averages of 10 independent simulation runs.

We use the weighted sum approach for solving both multi-objective ILP problems (static and dynamic), as it allows for obtaining solutions relatively quickly. Since objective functions can have different magnitudes and in order to set their weights easily, we normalize each objective function in the range $[0, 1]$. With normalization, weights are set such that $\sum_{i=1}^{5} w_i = 1$. Selected weights are shown in Table 6.1. The details of normalization are discussed in Appendix A.2.

6.7.1 Static Deployment

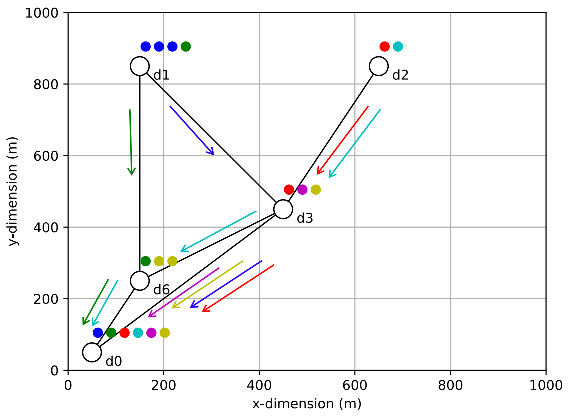
We evaluate the proposed static deployment scheme and greedy algorithm in terms of the ability to deploy a static NFV-based drone network. We assess the size of the produced network in terms of the number of deployed drones and the placement of SFC traffic in terms of hop distances for SFCs with varied number of SFCs $|R|$. We also report additional network and VNF placement properties. For the size of the network, we show the total number of deployed drones including relay drones. Note

Table 6.1: Evaluation Parameters

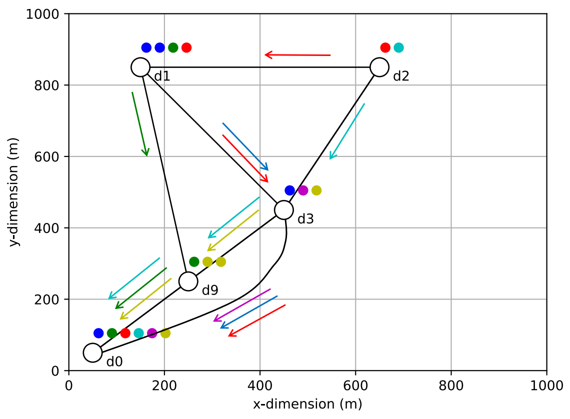
Parameter	**Value**
Transmission power P_t	15 dBm
Carrier frequency f_c	2 GHz
Channel bandwidth B	20 MHz
pl_{max}	95 dB
Noise power P_n	-80 dBm
Area	1000 m $\times$ 1000 m
L cell size $w \times h$	100 m $\times$ 100 m
Number of drones $\|D\|$	10 drones
Drone CPU cores CPU_{d_i}	8 cores
Drone RAM RAM_{d_i}	16 GB
Max. number of links per drone γ_{max}	4
Number of SFCs $\|R\|$	$[4, 14]$
CPU_{f_m}, RAM_{f_m}	$[1, 3]$ cores, $[1, 3]$ GB
VNFs per SFC $\|F_r\|$	$[2, 4]$
SFC throughput δ_r	5, 10, 20, 25 Mbps
w_1, w_2 (proposed static)	0.7, 0.3
w_1, w_2, w_3, w_4, w_5 (proposed dynamic)	0.4, 0.2, 0.1, 0.2, 0.1
w_1, w_2 (dynamic baseline)	0.7, 0.3

that the number of deployed drones includes the number of primary drones, which is known a priori given SFC specifications as discussed earlier in this section, while the number of relays is the result of network size minimization. However, we show the total number of deployed drones so the size of the overall network is noted. The number of relay drones is also reported.

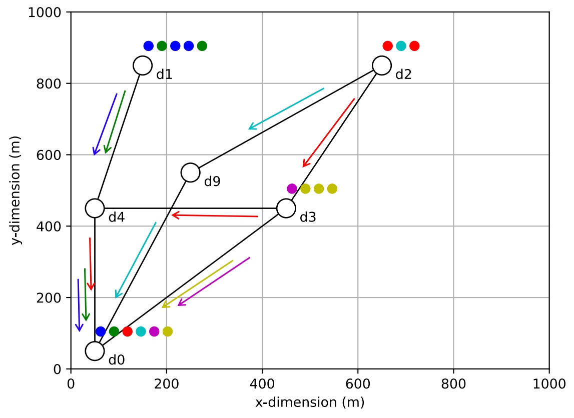
For each comparison of $|R|$, we compare the proposed optimal scheme with the greedy algorithm. We also compare with a variant of the optimal scheme that only minimizes the number of deployed drones and does not optimize VNF links, to compare the effect of the objective function. We refer to this latter scheme as the baseline.



(a) Minimal drones and VNF links (proposed)



(b) Minimal drones (baseline)



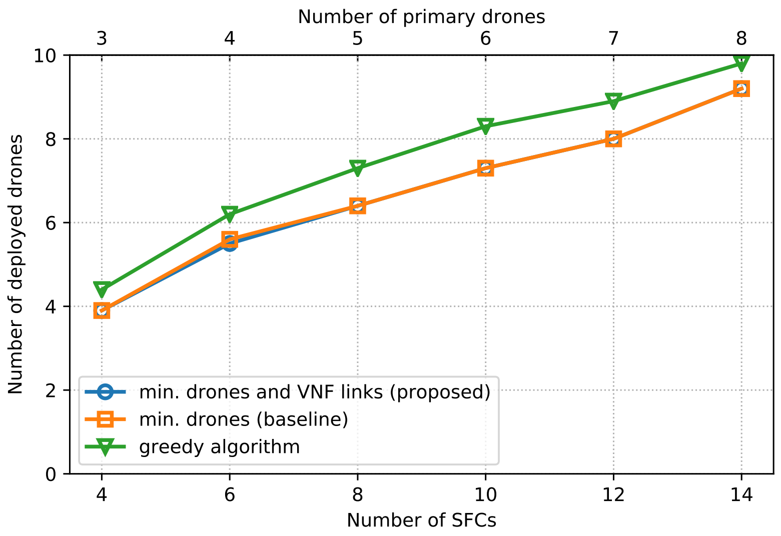
(c) Greedy algorithm

Figure 6.4: Topology and placement results example for 6 SFCs. Open circles represent drones and the small solid dots represent VNFs placed in drones. VNFs of the same color belong to the same SFC. Drone d0 is the sink node at the target location of SFCs, while other drones are sources or relays. Black lines are wireless links. Arrows represent the VNF links passing through physical links as the paths allocated for SFCs in corresponding color.

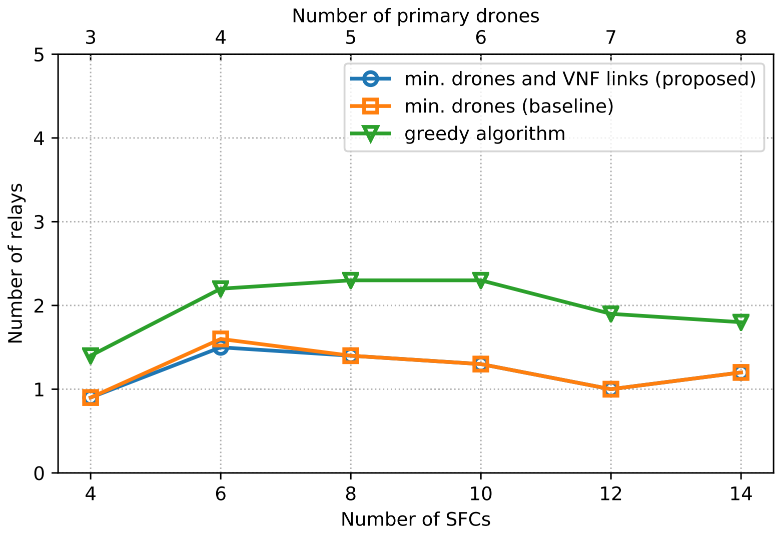
Results

We demonstrate the static deployment with a single scenario instance with 6 SFCs, shown in Figure 6.4. The figures show three network topologies and SFC placements produced by the evaluated schemes. Figure 6.4a shows the network produced by the proposed scheme, Figure 6.4b shows the network produced by the baseline (minimum drones) while Figure 6.4c shows the network produced by the greedy algorithm. In this scenario, d_0 is the sink node deployed at the target location where last VNFs of all SFCs are placed. Drones d_1, d_2 and d_3 are primary drones placed at SFC source locations while the remaining drones are relays. It can be observed that some VNFs are placed along the route of SFCs while other functions are placed on a single node. Drone computing resources are shared between functions of all SFCs, and drones are positioned to support the delivery of SFCs from source to target locations. Both the proposed and baseline schemes produced a single relay drone, while the greedy algorithm required two relays, due to the limited ability of the algorithm to find optimal relay placements. The proposed optimal scheme and the algorithm allocate shortest paths for SFCs, depending on available link capacity. While not apparent in this instance, the baseline scheme tends produce sub-optimal routes in larger instances

First we examine the number of deployed drones. As discussed earlier, the number of drones is affected by the number of locations required by SFCs (the number of primary drones). Thus, the deployment schemes minimize the number of relay drones and position them in optimal locations in order to support the network connectivity. Figure 6.5a shows the number of deployed drones by the three schemes. The number of relay drones is shown in Figure 6.5b. The number of relay drones is unpredictable since it depends on where the primary drones are located and how they are spaced from each other. The proposed scheme and the baseline produce the same number of relay drones since they both minimize the number of drones. The algorithm, however, shows a tendency to require more relays. This is because the algorithm efforts at reducing the number of relay drones are limited since it does not explore



(a) Total number of deployed drones including primary and relay drones



(b) Number of relay drones

Figure 6.5: Number of deployed drones

all possible relay placements to select ones that minimize the number of relays as it is limited by the triangulation-based placement of relays.

In Figure 6.6, we report the average hop distance per SFC. While the number

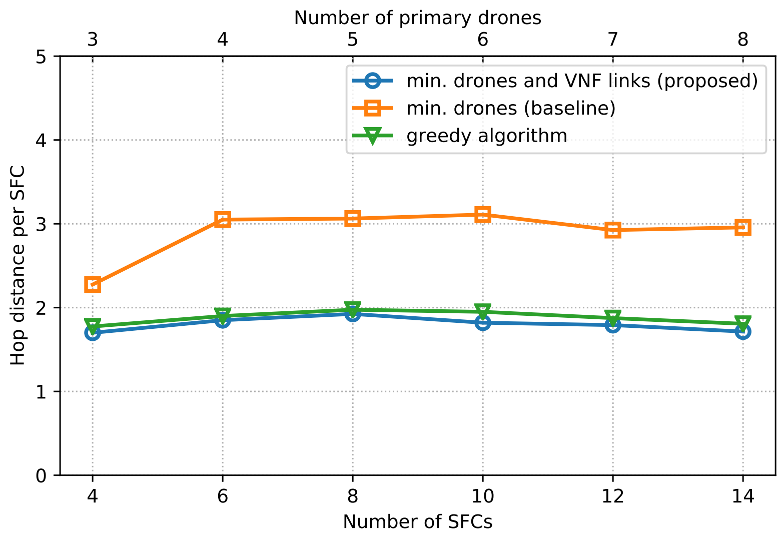


Figure 6.6: Average hop distance per SFC

of SFCs does not affect hop distance, it can be observed that both the optimal proposed scheme and the greedy algorithm perform similarly. This is due to the optimal scheme optimizing the placement of VNF links, whereas the greedy algorithm selects the first short path available leading to a performance that is similar to the optimal. The baseline scheme results in additional hops for SFCs due to not optimizing traffic routes. While the advantage is not pronounced due to the limited network size and deployment area, this optimization is needed to ensure the efficiency of selected paths. In different settings, for example in larger networks or in resource constrained networks, this aspect is important to be considered.

It is interesting to note the number of processing drones per SFC, as reported in Figure 6.7. The number of processing nodes per SFC is the number of nodes that host VNFs for a particular SFC. The minimum expected number is two nodes which include the source node of the SFC and the node hosting the terminal VNFs. Intermediate VNFs (when the SFCs consists of more than two VNFs) are placed either

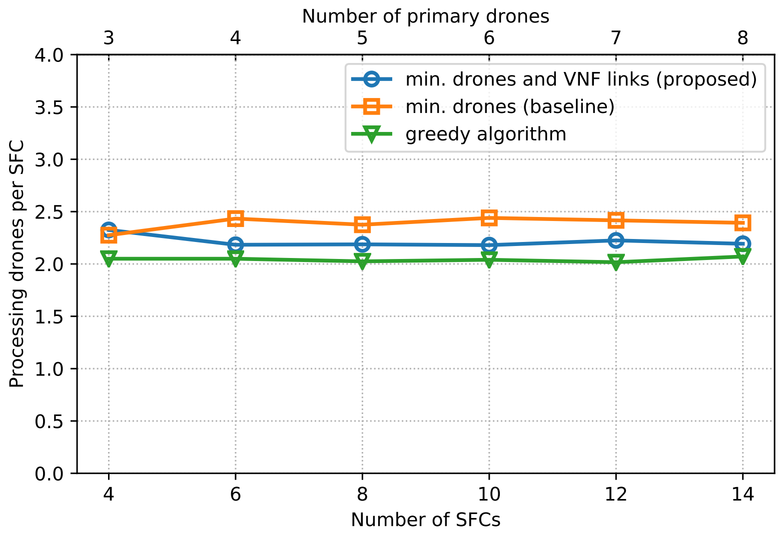


Figure 6.7: Average processing nodes per SFC

in the first, last or any intermediate drones along the path. It is generally better to reduce this fragmentation to the lowest, which leads to having more VNFs of the same chain in the same physical node. Then VNFs can communicate locally and not depend on external links. The optimal and the baseline do not directly optimize for this fragmentation. However, this is affected indirectly by the optimization of traffic routes. The baseline spreads out traffic VNFs along the routes. The difference is minimal. However, the optimal scheme and greedy algorithm tend to require the minimum number of two processing nodes. The algorithm tends to be lower since the algorithm places intermediate VNFs in the first nodes of SFC paths, and does not move to the next node until the current node is at capacity.

Next, we examine the running time of the optimization schemes compared also to the greedy algorithm. The comparison is shown in Figure 6.8. For the optimization based schemes, it is observed that when a small number of SFCs and drones is required, there are often difficult instances to solve which take and extended amount

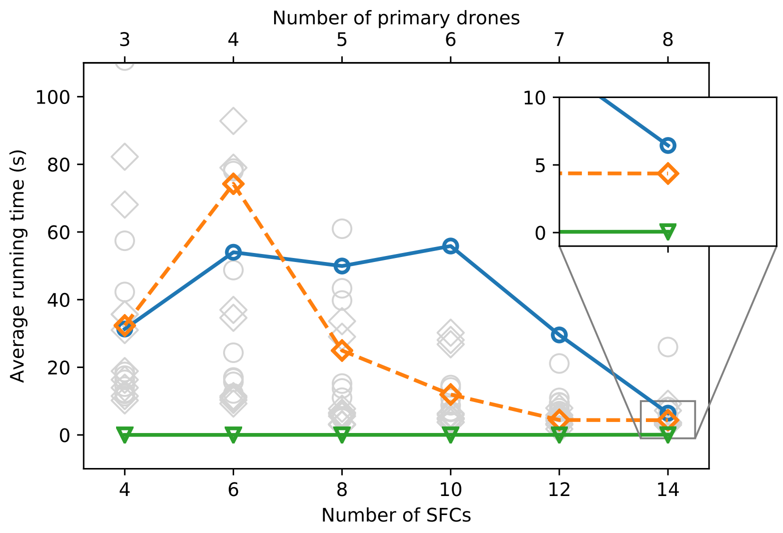


Figure 6.8: Average running time

of time. These are usually instances where the required locations are far apart and the solver spends a long time to find optimal placements for relays and SFCs. This difficulty is observed in the optimal scheme and baseline where solve times tend to be inconsistent. On the other hand, as the number of drones increases, there is less chance of having drones far apart. As a result, the difficulty tends to decrease and most instances are solved in shorter times that are closer to the mean. However, the baseline tends to solve those instances faster due to optimizing a single objective function. The greedy algorithm demonstrates an extremely short solve time measured at less than half a second. It increases linearly by unnoticeable fractions of a second with the number of SFCs. While the running time is not important in the static deployment since it is computed once; however, the algorithm is useful for fast prototyping or planning. It also provides an alternative to using commercial optimization solvers.

6.7.2 Dynamic Orchestration

Setup

In this section, we evaluate the dynamic scheme in terms of its ability to adjust the deployed network and reduce the overhead resulting from continuous mobility and orchestration cycles.

We evaluate a dynamic network with a varied number of SFCs. The setup is the same as the static scenario in terms of SFC capacities and location requirements as well as the parameters shown in Table 6.1. For each comparison of $|R|$, the simulation starts by making an initial placement using the optimal static scheme. Then, ten successive movement steps are generated. In each step, $|R| / 2$ random locations are generated and set as source locations for each pair SFCs in the same manner used as the static deployment evaluation. Locations are generated using a variation of the random waypoint mobility model [AT14] where new locations are within the vicinity of 400 m from previous locations. This is to simulate limited incremental movements. At each step, the dynamic scheme is invoked to adjust the network.

We compare the dynamic proposed scheme with a baseline variant that does not minimize overhead, where the objectives are only minimizing the number of drones and VNF links. The weights used for both schemes are shown in Table 6.1. In this evaluation, we report a number of metrics that demonstrate the changes undertaken by the network during the 10 movement steps. These include the number of drones and wireless links established between drones. We report the number of VNF relocations, which is the total number of intermediate VNFs that underwent changes in placement. As well, we report the number of rerouted SFCs as a result of adjustment.

Results

Both schemes produce the same number of deployed drones. We do not report it since it rarely increases in our simulations above the number of initially deployed drones as shown in Figure 6.5a in the static scenario. This is attributed to the fact that both schemes minimize the number of drones with the highest weight among objectives. This leads to preventing the deployment of additional relays during movements steps unless it is necessary. When required, both schemes only deploy an additional drone in about 10% of the simulation runs.

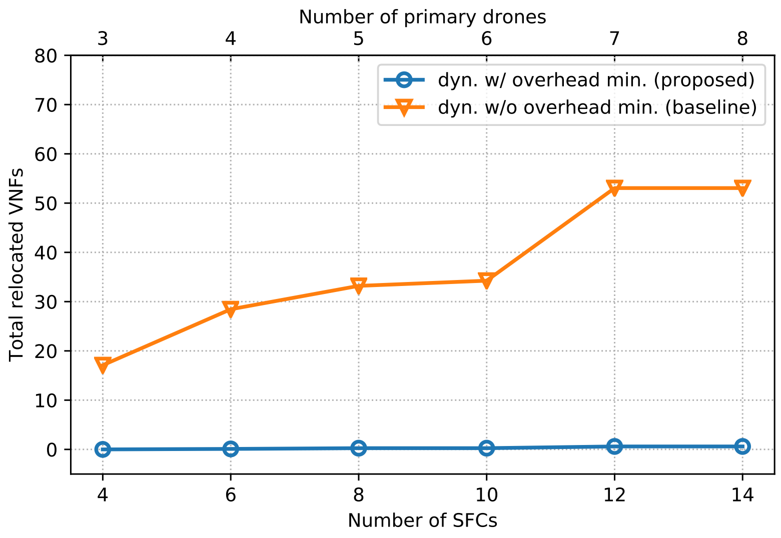


Figure 6.9: Total number of migrated VNFs

Figure 6.9 shows the total number of VNFs that underwent placement change during the simulation. VNF relocations occur due an interplay of topology changes, route allocations, and the solver exploring diverse solutions leading to the same objective value. It can be observed that a high number of relocations occurs if no considerations are made to limit them. Note that these placement changes only involve intermediate

VNFs.

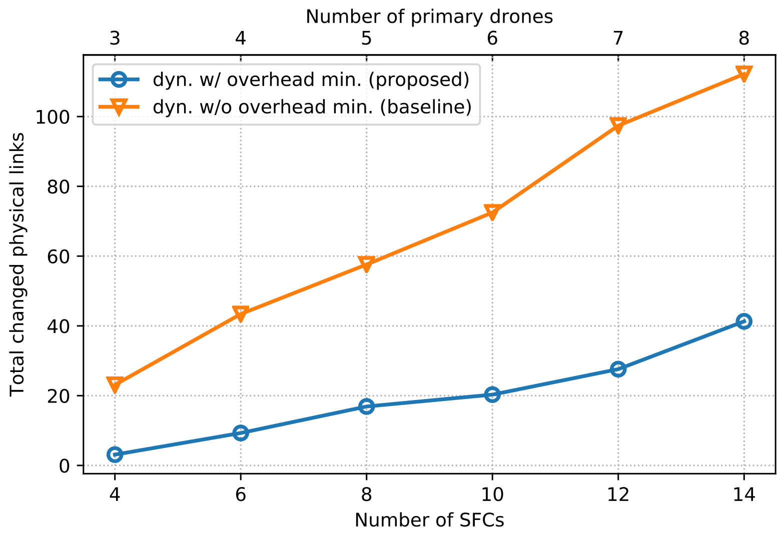


Figure 6.10: Total number of changes to states of physical links

Figure 6.10 shows the total number of changed links. This represents the total number of links connected or disconnected due the reconfiguration made by the dynamic schemes. It can be observed that with overhead minimization, links undergo significantly less changes than without minimization. As such, the network maintains the topology as drones move. We reiterate that the reason we reduce this change is to minimize the need to reroute SFC traffic along affected links.

In Figure 6.11, we show the total number of rerouted SFCs. With overhead minimization, the number of SFCs affected by topology changes is reduced compared to no overhead minimization. This is the combined result of minimizing changes to network links and VNF link paths.

As a result, it can be observed that with only a limited number of movement steps, a considerable amount of overhead can be generated which requires constant maintenance of SFCs and their traffic routes. Practical scenarios are expected to involve

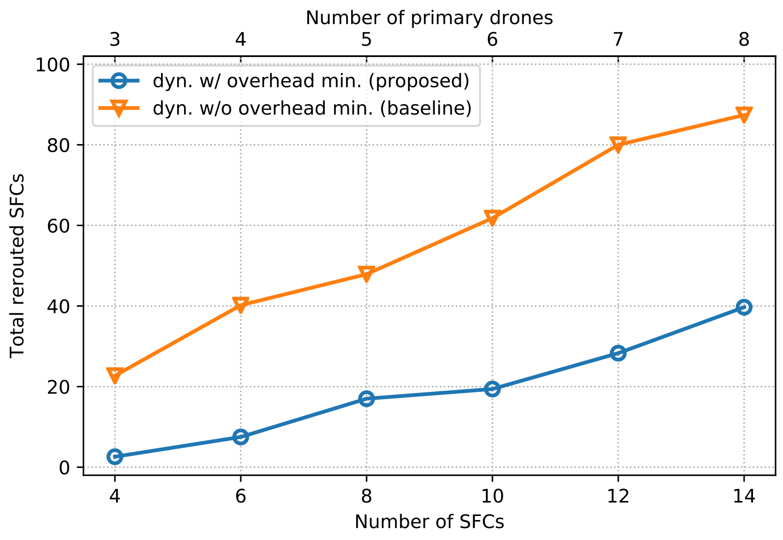


Figure 6.11: Total number rerouted SFCs

a larger number of movement steps and different patterns of location requirements. Therefore, overhead reduction is crucial to limit the load on controllers overseeing the network and to ensure smooth network operation.

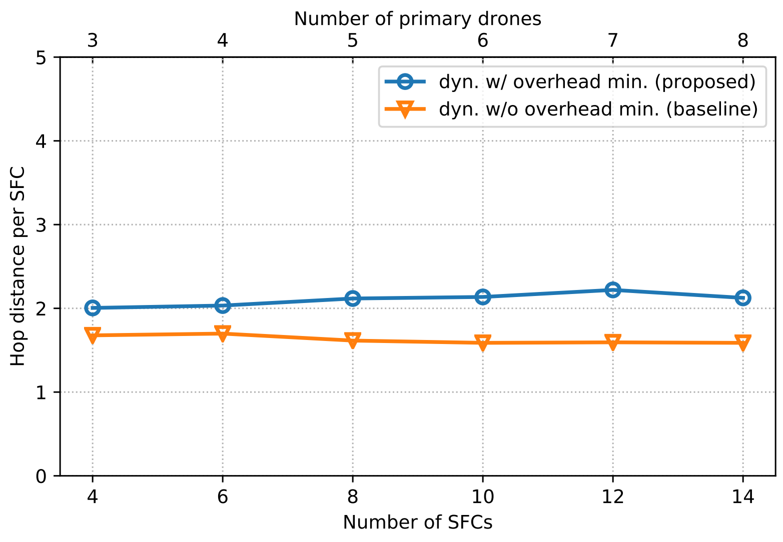


Figure 6.12: Average hop distance per SFC

As noted earlier, the second objective of minimizing VNF links conflicts with overhead minimization, as the former seeks to select the best paths for SFCs while the latter seeks to limit changes in consecutive steps. We compare the overall average hop distance, as shown in Figure 6.12. We observe that the average hop distance is slightly affected due to overhead reduction. While the difference is small, with overhead reduction, hop distances are not optimized at the same degree as the alternative scheme.

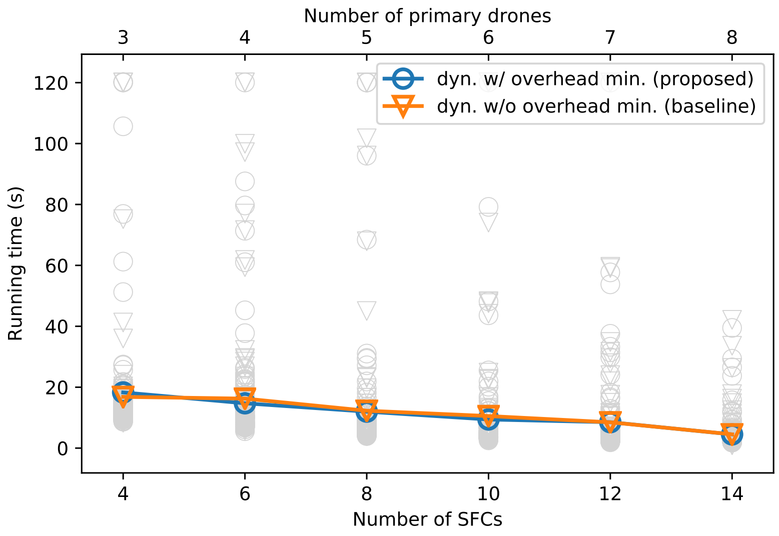


Figure 6.13: Average solve time (s)

Finally, we examine the running times of the compared dynamic schemes. Note that the same pattern of difficulty decreasing with the increase of SFCs and deployed drones is present here for the reasons discussed in the static scenario. However, in the dynamic case, average running times are generally lower than the static scheme, as the dynamic scheme uses decisions made by the initial placement and previous steps.

6.8 Summary

In this chapter, we reiterated the motivations and advantages of utilizing NFV in drone networks. The use of NFV in drone networks is motivated by the ability and ease of configuring drones for various tasks, especially those that require in-network computing. This is especially useful for missions in remote areas with no access to reliable infrastructure. We illustrated how chained VNFs can be used in drone networks to deliver and relay traffic in applications such as remote video monitoring. We then detailed the problem definition, motivating the need to determine an efficient deployment of the physical network to host and route SFC traffic as required. We also detailed special considerations needed to orchestrate the network given SFC mobility requirements. Therefore, specialized deployment and orchestration schemes are needed to facilitate such applications.

To target scenarios with stationary networks, we proposed a deployment scheme implemented as an ILP. The scheme deploys drone networks with VNFs hosted in drone computing resources. A minimal network topology is determined based on supplied SFCs and associated VNFs with location requirements. We followed by providing a fast greedy algorithm that produces reasonable deployments and provides an alternative to depending on optimization solvers.

For scenarios with mobility requirements, we addressed the case where SFCs require the hosting drones to move over the mission area. A dynamic orchestration scheme is developed for this purpose. The scheme's objective is to maintain the deployed network during movement while minimal adjustments are made to the network topology and the hosted SFCs. This ensures that a limited number of disruptive network changes take place. This is an important consideration since updates made by orchestration are expected to take place many times during repeated movements.

The proposed optimal static scheme and greedy algorithm were compared with a variant of the optimal scheme in terms of attributes of the produced networks, such as the number of relays, SFC hop distance, and other attributes. Then we evaluated the proposed dynamic scheme. The proposed scheme with overhead minimization was compared to a baseline scheme with no overhead minimization. We reported the total number of relocated VNFs and rerouted SFCs, and the extent of changes to the network topology after a number of movements and update events. Results showed a significant reduction of such overheads which confirmed the need to make such a consideration in orchestrating NFV-based drone networks.

Chapter 7

Conclusions and Future Directions

The use of unmanned aerial vehicles or drones is increasingly growing [Fed20] and is expected to be a part of many industrial and civilian applications including future mobile networks [Sha+19]. Communications and computing are integral parts of drone networks [Wan+20b], involved in command and control and enabling connectivity and intelligence for multiple networked drones conducting a variety of tasks. Drone-assisted communication networks are enabling many promising applications, as drones' flexibility and mobility are utilized to deploy network services on demand and extend the coverage of terrestrial networks.

Considerable challenges are faced in drone networks due to their distinct characteristics compared to traditional networks. Mobility, the likelihood of intermittent connectivity, and lack of access to ground infrastructure in remote areas are among the aspects of drone networks that present unique challenges.

Softwarization is a key element in today's wireless and mobile networks. Using softwarization technologies, namely SDN and NFV, the traditional integration between the control and data planes is separated. This separation consolidated control functions into logically centralized components that control diverse commodity hardware representing programmable network devices. This leads to cost savings and ease of control and management of various and easily upgradable network devices. Using NFV, an additional layer of softwarization is added, where network processing functions are deployed as virtualized elements instead of hardware elements. Nowadays, softwarization spans cloud and mobile networks and is one of the foundations of the 5G architecture [Afo+18].

SDN was introduced to drone networks to circumvent the challenges mentioned above by utilizing centralized control and programmability [SO+20]. NFV was also applied to enable more abstraction and reconfigurability as well as virtualized onboard processing. While this integration showed performance improvements from networking and service perspectives, more investigation is needed to measure the benefits of softwarization concerning reconfiguration and the possible new applications delivered based on realized gains. A few additional challenges in applying softwarization remain unaddressed.

In this thesis, we focused on demonstrating the benefits gained from reconfigurability offered by softwarization. We also addressed some open challenges of enabling software defined drone networks. Specifically, we targeted enabling aerial SDN control, the deployment, and dynamic orchestration of NFV-based missions of networked drones.

7.1 Conclusions

Chapter 3 was dedicated to demonstrating the direct advantages of reconfiguration. We proposed a model for evaluating the gains of a network of softwarized drones that conduct a multiplicity of tasks and utilizes inflight dynamic reconfigurability to enhance the networks efficiency in response to urgent events. We motivated the model by proposing a representative use case, where an industrial entity uses a fleet of drones to conduct different routine tasks. A similar use case is an SP that offers its drone fleet as a service in a setting where customers from municipalities, law enforcement, and research are interested in utilizing the fleet to conduct different tasks. We described the elements of the model, its operation, and how it is enabled by softwarization. For the purpose of evaluation, we introduced two additional variants of the model, one with limited softwarization and another with non-softwarized drones, each fixed to serve a single type of task. In our evaluations, the models were compared based on the ability to complete assigned tasks in the shortest amount of time. The reconfigurability of the softwarized system allowed the fleet to accomplish tasks sooner than the non-softwarized fleet. As well, the softwarized models demonstrated energy savings. Another evaluated scenario focused on the ability of the fleet to respond quickly to critical events, such as monitoring a crisis or sensing data of a natural phenomenon within a limited time frame. Such tasks are designated as high-priority and have a deadline by which the task must be completed. Thanks to the dynamic reconfigurability, softwarized drones are highly successful at fulfilling high-priority tasks owing to its flexibility over non-softwarized or non-reconfigurable drones. In this chapter, we demonstrated that softwarization increases the efficiency of a small fleet of drones to a level similar to, or better than, a non-softwarized fleet of the same or twice the size, due to dynamic reconfigurability in accordance to the requested tasks. Such gains can translate to more utilization and revenue for the SP, as it is able to serve more tasks, while costs could be minimized by maintaining a smaller fleet.

In Chapter 4, we proposed an architecture for softwarized drone networks and described its components. The architecture enables flexible deployment, management, and reconfiguration using SDN and NFV technologies. The various requirements and components to enable a reconfigurable drone network suitable for various applications were described. Within this architecture, we proposed components that enable addressing limitations with respect to SDN control and orchestration of NFV-enabled networks. We also discussed possible use cases of the proposed architecture.

Chapter 5 focused on addressing SDN control connectivity challenges within the architecture described in Chapter 4. We identified the limitation when the network is deployed in areas with limited or no access to a ground infrastructure to connect to SDN controllers. In such a setting, several drones are dedicated as SDN controllers. Certain requirements need to be met for deploying controllers, which are limiting the number of controllers and ensuring the direct connectivity for control links and inter-controller links. We described two schemes with different priorities with respect to the number of drones allowed to be deployed as controllers. The first scheme is more permissible in terms of the number of controllers and thus can guarantee controller connectivity to all drones. The second scheme allows only a few controllers and guarantees direct connectivity to the maximum possible number of drones. The proposed schemes include variants for initial deployment and dynamic adjustment as the network topology changes. Dynamic adjustment schemes adjust controller locations to maintain control links while limiting the distance required for controllers to travel. This optimization thus allows controllers to travel efficiently by limiting the time and energy required for controllers to travel during adjustments. Evaluations showed the performance of the proposed schemes. For the initial deployment of controllers, we showed the number of controllers needed for different topologies and showed the flexibility of establishing multi-hop inter-controller links when needed. In the dynamic setting, we demonstrated the reduced distances required to adjust controllers during network topology changes. The result of our work is that a drone network can be deployed with airborne controllers that dynamically adapt to the network topology independently from the actual network formation related to the task, while maintaining established control links. This ensures that the control plane is functional to operate a dynamic network.

In addition to enabling modularity of functions and reuse of hardware (drones), NFV also enables reconfigurable on-board processing in drone networks. To facilitate such flexibility, in Chapter 6 we discussed the need for planning and orchestration techniques for NFV-based drone network missions. We demonstrated a remote video monitoring use case that expresses its mission using several SFCs, where SFCs capture, process, and transmit data between different locations through the drone network. To deploy such a network, a minimal and efficient formation of the network needs to be determined along with optimal placement and routing of SFCs within the drone network. An optimal scheme was proposed for static and initial deployments. To facilitate the mission mobility requirements, a dynamic orchestration scheme was proposed to adjust the network topology and SFC placements as needed while limiting disruptive overheads such as continuous rerouting and relocation of VNFs. Evaluations showed the feasibility and properties of networks deployed using the static scheme in terms of network size and selected traffic paths. A comparison was made to a proposed fast heuristic algorithm that achieves deployments close to the optimal scheme. For dynamic scenarios, evaluations showed how dynamic orchestration can efficiently maintain the deployed networks while keeping overheads to a minimum. This is an important consideration in such highly dynamic settings due to network mobility.

7.2 Future Directions

Several future directions could be pursued based on the work presented herein. We list these below.

- To investigate additional gains due to softwarization, future work may consider expanding on the evaluation presented in Chapter 3 and investigate additional use cases. For instance, within the same SP or industrial contexts, evaluations may involve more specialized tasks that seek different independent objectives while deployed simultaneously. For example, objectives may include maintaining coverage of a set of services or maintaining up-to-date sensing information. We anticipate that such tasks can be deployed more effectively using softwarized reconfigurable drones, where the different task goals are met with an enhanced efficiency due to softwarization.

- The architecture described in Chapter 4 can be expanded further by investigating detailed requirements and specifications for both the northbound and southbound interfaces based on real world experiments. Such work can involve experimental design and implementation of a softwarized network of drones with associated software stack. The next step is standardization of developed interfaces to enable an open and standardized softwarized drone networks.

- For the SDN controller deployment proposed in Chapter 5, we relied on optimization solvers to solve the optimization models of the proposed schemes. Thus, low complexity and fast heuristic algorithms are needed. A challenge for potential algorithms is meeting all the discussed link and movement requirements at an acceptable level compared to the optimal schemes.

- The SDN controller deployment schemes can be further improved by taking into account realistic network conditions. The problem can be investigated using detailed mobility and traffic models of drone networks, including the different interactions between the drone network nodes and controllers. Then, controller deployment, movement, and assignment can be dynamically adjusted according to different requirements, such as the varying demands of drones, QoS requirements, and mobility based on the current state of the network.

- The work presented in Chapter 6 can be extended further by developing a heuristic or greedy algorithm for dynamic orchestration to increase scalability and reduce the complexity of the dynamic scheme. Another important direction is incorporating drones' power consumption in dynamic network adjustments. The dynamic scheme can be improved by efficiently relocating VNFs and rerouting traffic away from drones with depleted batteries.

- In Chapter 6, the focus was on deploying drone networks where SFCs require instantaneous VNF operation and connectivity within each chain. An extension that is worth investigating is scenarios where SFCs operate similar to delay-tolerant networking. While in-network computation is still required, computation results are not required in real-time. This can be useful in situations where drones have limited computing capacity or cannot establish connectivity at all times. In such scenarios, individual VNFs in the chain operate independently and with space and time constraints. For example, data acquisition VNFs can be activated to collect data at certain times or in remote disconnected areas. VNFs for data processing are activated at different times when results are requested and connectivity can be established. Such scenarios require orchestrating the network with such constraints in mind and schedule the operation of VNFs and their connectivity accordingly.

Bibliography

[3GP19] 3GPP. *3GPP UAS UAV*. Nov. 2019. URL: `https://www.3gpp.org/uas-uav`.

[Abd+17] M. J. Abdel-Rahman, E. A. Mazied, A. MacKenzie, S. Midkiff, M. R. Rizk, and M. El-Nainay. "On Stochastic Controller Placement in Software-Defined Wireless Networks". In: *IEEE Wireless Commun. and Netw. Conf. (WCNC)*. Mar. 2017, pp. 1–6.

[Afo+18] I. Afolabi, T. Taleb, K. Samdanis, A. Ksentini, and H. Flinck. "Network slicing and softwarization: A survey on principles, enabling technologies, and solutions". In: *IEEE Communications Surveys and Tutorials* 20.3 (2018), pp. 2429–2453.

[Ala+20] I. Alam, K. Sharif, F. Li, Z. Latif, M. M. Karim, S. Biswas, B. Nour, and Y. Wang. "A Survey of Network Virtualization Techniques for Internet of Things Using SDN and NFV". In: *ACM Comput. Surv.* 53.2 (Apr. 2020).

[Als+18] A. Alshamrani, S. Guha, S. Pisharody, A. Chowdhary, and D. Huang. "Fault Tolerant Controller Placement in Distributed SDN Environments". In: *IEEE Int. Conf. Commun. (ICC)*. May 2018, pp. 1–7.

[AT14] M. Alharthi and A.-E. M. Taha. “Modeling mobility for networked mobile cyber-physical systems”. In: *Proceedings of the 4th ACM SIGBED International Workshop on Design, Modeling, and Evaluation of Cyber-Physical Systems*. 2014, pp. 1–6.

[ATH19a] M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “An Architecture for Software Defined Drone Networks”. In: *IEEE International Conference on Communications (ICC)*. May 2019, pp. 1–5.

[ATH19b] M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Dynamic Controller Placement in Software Defined Drone Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2019.

[ATH20] M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Utilizing Network Function Virtualization for Drone-Based Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2020, pp. 1–5.

[ATH21] M. Alharthi, A.-E. M. Taha, and H. S. Hassanein. “Evaluating Softwarization Gains in Drone Networks”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2021.

[Aus10] R. Austin. *Unmanned Aircraft Systems*. Wiley, 2010.

[Bar+13] M. F. Bari, A. R. Roy, S. R. Chowdhury, Q. Zhang, M. F. Zhani, R. Ahmed, and R. Boutaba. “Dynamic Controller Provisioning in Software Defined Networks”. In: *Proc. 9th Int. Conf. Netw. Service Manag. (CNSM)*. Oct. 2013, pp. 18–25.

[Bar+16] F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte. “Orchestrating Virtualized Network Functions”. In: *IEEE Trans. Netw. Service Manag.* 13.4 (2016), pp. 725–739.

[Bar+17] B. Barritt, T. Kichkaylo, K. Mandke, A. Zalcman, and V. Lin. “Operating a UAV mesh internet backhaul network using temporospatial SDN”. In: *IEEE Aerospace Conference*. 2017, pp. 1–7.

[Bas+13] A. Basta, W. Kellerer, M. Hoffmann, K. Hoffmann, and E. Schmidt. “A Virtual SDN-Enabled LTE EPC Architecture: A Case Study for S-/P-Gateways Functions”. In: *2013 IEEE SDN for Future Networks and Services (SDN4FNS)*. Nov. 2013, pp. 1–7.

[BBT19] O. Bekkouche, M. Bagaa, and T. Taleb. “Toward a UTM-Based Service Orchestration for UAVs in MEC-NFV Environment”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2019, pp. 1–6.

[BC18] B. Barritt and V. Cerf. “Loon SDN: Applicability to NASA’s next-generation space communications architecture”. In: *IEEE Aerospace Conference*. 2018, pp. 1–9.

[Bou+19] C. Boucetta, A. Dridi, H. Moungla, H. Afifi, and A. E. Kamal. “Optimizing Drone Deployment for Cellular Communication Coverage During Crowded Events”. In: *IEEE Military Communications Conference (MILCOM)*. IEEE, Nov. 2019, pp. 622–627.

[BST13] I. Bekmezci, O. K. Sahingoz, and Ş. Temel. “Flying Ad-Hoc Networks (FANETs): A survey”. In: *Ad Hoc Networks* 11.3 (2013), pp. 1254 – 1270.

[BY+19] I. Bor-Yaliniz, M. Salem, G. Senerath, and H. Yanikomeroglu. “Is 5G ready for drones: A look into contemporary and prospective wireless networks from a standardization perspective”. In: *IEEE Wireless Communications* 26.1 (2019), pp. 18–27.

[CBD11] D.-S. Chen, R. G. Batson, and Y. Dang. *Applied integer programming: modeling and solution*. John Wiley & Sons, 2011.

[CBM17] S. Correia, A. Boukerche, and R. I. Meneguette. “An Architecture for Hierarchical Software-Defined Vehicular Networks”. In: *IEEE Communications Magazine* 55.7 (2017), pp. 80–86.

[Che+17] M. Chen, M. Mozaffari, W. Saad, C. Yin, M. Debbah, and C. S. Hong. “Caching in the Sky: Proactive Deployment of Cache-Enabled Unmanned Aerial Vehicles for Optimized Quality-of-Experience”. In: *IEEE Journal on Selected Areas in Communications* 35.5 (May 2017), pp. 1046–1061.

[Cic+19] C. T. Cicek, H. Gultekin, B. Tavli, and H. Yanikomeroglu. “UAV Base Station Location Optimization for Next Generation Wireless Networks: Overview and Future Research Directions”. In: *1st International Conference on Unmanned Vehicle Systems-Oman (UVS)*. Feb. 2019, pp. 1–6.

[Cos+12] S. Costanzo, L. Galluccio, G. Morabito, and S. Palazzo. “Software Defined Wireless Networks: Unbridling SDNs”. In: *2012 European Workshop on Software Defined Networking*. Oct. 2012, pp. 1–6.

[Cro16] D. F. Crouse. “On implementing 2D rectangular assignment algorithms”. In: *IEEE Transactions on Aerospace and Electronic Systems* 52.4 (2016), pp. 1679–1696.

[CS17] U. Challita and W. Saad. “Network Formation in the Sky: Unmanned Aerial Vehicles for Multi-Hop Wireless Backhauling”. In: *IEEE Global Communications Conference (GLOBECOM)*. Dec. 2017.

[DHZ18] A. Dvir, Y. Haddad, and A. Zilberman. “Wireless controller placement problem”. In: *15th IEEE Annu. Consum. Commun. Netw. Conf. (CCNC)*. Jan. 2018, pp. 1–4.

[Dji] *DJI Inspire 2*. URL: `https://www.dji.com/ca/inspire-2`.

[DJI21] DJI. *Manifold 2*. 2021. URL: `https://www.dji.com/manifold-2`.

[DOD21] J. Dong, K. Ota, and M. Dong. “UAV-based Real-Time Survivor Detection System in Post-disaster Search and Rescue Operations”. In: *IEEE Journal on Miniaturization for Air and Space Systems* (2021).

[Ebe] *eBee X Fixed-Wing Mapping Drone - senseFly*. URL: `https://www.senseFly.com/drone/ebee-x-fixed-wing-drone/`.

[Erd+17] M. Erdelj, E. Natalizio, K. R. Chowdhury, and I. F. Akyildiz. “Help from the Sky: Leveraging UAVs for Disaster Management”. In: *IEEE Pervasive Computing* 16.1 (Jan. 2017), pp. 24–32.

[Ets] *ETSI GS NFV-MAN 001: Network Functions Virtualisation (NFV); Management and Orchestration*. Dec. 2014. URL: `https://www.etsi.org/deliver/etsi_gs/NFV-MAN/001_099/001/01.01.01_60/gs_nfv-man001v010101p.pdf`.

[Fed20] Federal Aviation Administration. *FAA Aerospace Forecast Fiscal Years 2021–2041*. 2020. URL: `https://www.faa.gov/data_research/aviation/aerospace_forecasts/media/FY2020-40_FAA_Aerospace_Forecast.pdf`.

[Fel+15] W. Felter, A. Ferreira, R. Rajamony, and J. Rubio. "An updated performance comparison of virtual machines and Linux containers". In: *IEEE Int. Symp. Performance Analysis Systems and Software (ISPASS)*. Mar. 2015, pp. 171–172.

[FH09] R. Z. Farahani and M. Hekmatfar. *Facility location: concepts, models, algorithms and case studies*. Springer Science & Business Media, 2009.

[FM14] M. Fischetti and M. Monaci. "Proximity search for 0-1 mixed-integer convex programming". In: *Journal of Heuristics* 20.6 (2014), pp. 709–731.

[Gha+19] K. Z. Ghafoor, L. Kong, D. B. Rawat, E. Hosseini, and A. S. Sadiq. "Quality of Service Aware Routing Protocol in Software-Defined Internet of Vehicles". In: *IEEE Internet of Things Journal* 6.2 (2019), pp. 2817–2828.

[GJV16] L. Gupta, R. Jain, and G. Vaszkun. "Survey of Important Issues in UAV Communication Networks". In: *IEEE Communications Surveys Tutorials* 18.2 (2016), pp. 1123–1152.

[GK18] H. Ghafoor and I. Koo. "CR-SDVN: A Cognitive Routing Protocol for Software-Defined Vehicular Networks". In: *IEEE Sensors Journal* 18.4 (2018), pp. 1761–1772.

[Gu+15] Y. Gu, M. Zhou, S. Fu, and Y. Wan. “Airborne WiFi Networks Through Directional Antennae: An Experimental Study”. In: *IEEE Wireless Communications and Networking Conference (WCNC)*. Mar. 2015, pp. 1314–1319.

[HA17] M. Hassanalian and A. Abdelkefi. “Classifications, applications, and design challenges of drones: A review”. In: *Progress in Aerospace Sciences* 91.5 (May 2017), pp. 99–131.

[Han+15] B. Han, V. Gopalakrishnan, L. Ji, and S. Lee. “Network function virtualization: Challenges and opportunities for innovations”. In: *IEEE Communications Magazine* 53.2 (2015), pp. 90–97.

[HB16] J. G. Herrera and J. F. Botero. “Resource allocation in NFV: A comprehensive survey”. In: *IEEE Transactions on Network and Service Management* 13.3 (2016), pp. 518–532.

[HSL09] Z. Han, A. L. Swindlehurst, and K. J. Liu. “Optimization of MANET connectivity via smart deployment/movement of unmanned air vehicles”. In: *IEEE Transactions on Vehicular Technology* 58.7 (2009), pp. 3533–3546.

[HSM12] B. Heller, R. Sherwood, and N. McKeown. “The Controller Placement Problem”. In: *Proc. 1st Workshop Hot Topics in Software Defined Networks (HotSDN)*. Helsinki, Finland: ACM, 2012, pp. 7–12.

[HSSC08] A. Hagberg, P. Swart, and D. S Chult. *Exploring network structure, dynamics, and function using NetworkX*. Tech. rep. Los Alamos National Lab.(LANL), Los Alamos, NM (United States), 2008.

[Hur+15] A. Hurtado-Borràs, J. Palà-Solé, D. Camps-Mur, and S. Sallent-Ribes. "SDN wireless backhauling for Small Cells". In: *IEEE International Conference on Communications (ICC)*. June 2015, pp. 3897–3902.

[Iqb+16] H. Iqbal, J. Ma, K. Stranc, K. Palmer, and P. Benbenek. "A software-defined networking architecture for aerial network optimization". In: *IEEE NetSoft Conference and Workshops (NetSoft)*. 2016, pp. 151–155.

[Itu] *HAPS – High-altitude platform systems*. Dec. 2019. URL: `https://www.itu.int/en/mediacentre/backgrounders/Pages/High-altitude-platform-systems.aspx`.

[Jai+16] A. Jain, N. Sadagopan, S. K. Lohani, and M. Vutukuru. "A comparison of SDN and NFV for re-designing the LTE packet core". In: *IEEE Conference on Network Function Virtualization and Software Defined Networks (NFV-SDN)*. IEEE. 2016, pp. 74–80.

[Jia+19] Z. Jiao, Y. Zhang, J. Xin, L. Mu, Y. Yi, H. Liu, and D. Liu. "A Deep Learning Based Forest Fire Detection Approach Using UAV and YOLOv3". In: *1st International Conference on Industrial Artificial Intelligence (IAI)*. 2019, pp. 1–5.

[Jor+18] S. Jordan, J. Moore, S. Hovet, J. Box, J. Perry, K. Kirsche, D. Lewis, and Z. T. H. Tse. "State-of-the-art technologies for UAV inspections". In: *IET Radar, Sonar and Navigation* 12.2 (2018), pp. 151–164.

[JP+12] A. Jimenez-Pacheco, D. Bouhired, Y. Gasser, J. C. Zufferey, D. Floreano, and B. Rimoldi. "Implementation of a wireless mesh network of ultra light MAVs with dynamic routing". In: *2012 IEEE Globecom Workshops*. Dec. 2012, pp. 1591–1596.

[JSK18] S. Jeong, O. Simeone, and J. Kang. "Mobile Edge Computing via a UAV-Mounted Cloudlet: Optimization of Bit Allocation and Path Planning". In: *IEEE Transactions on Vehicular Technology* 67.3 (Mar. 2018), pp. 2049–2063.

[Kal+17] E. Kalantari, M. Z. Shakir, H. Yanikomeroglu, and A. Yongacoglu. "Backhaul-aware robust 3D drone placement in 5G+ wireless networks". In: *IEEE International Conference on Communications Workshops (ICC Workshops)*. May 2017, pp. 109–114.

[Ker+19] N. Kerle, F. Nex, M. Gerke, D. Duarte, and A. Vetrivel. "UAV-Based Structural Damage Mapping: A Review". In: *ISPRS International Journal of Geo-Information* 9.1 (Dec. 2019), p. 14.

[Kre+15] D. Kreutz, F. M. V. Ramos, P. E. Veríssimo, C. E. Rothenberg, S. Azodolmolky, and S. Uhlig. "Software-Defined Networking: A Comprehensive Survey". In: *Proceedings of the IEEE* 103.1 (Jan. 2015), pp. 14–76.

[LFZ19] B. Li, Z. Fei, and Y. Zhang. "UAV Communications for 5G and Beyond: Recent Advances and Future Trends". In: *IEEE Internet of Things Journal* 6.2 (Apr. 2019), pp. 2241–2263.

[LH05] N. Li and J. C. Hou. "Improving connectivity of wireless ad hoc networks". In: *The Second Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*. 2005, pp. 314–324.

[LH06] M. Lacage and T. R. Henderson. “Yet Another Network Simulator”. In: *Proc. Workshop NS-2: The IP Network Simulator*. WNS2 ’06. Pisa, Italy: ACM, 2006.

[Li+21] J. Li, W. Shi, H. Wu, S. Zhang, and X. Shen. “Cost-Aware Dynamic SFC Mapping and Scheduling in SDN/NFV-Enabled Space-Air-Ground Integrated Networks for Internet of Vehicles”. In: *IEEE Internet of Things Journal* 4662.c (2021), pp. 1–15.

[Liu+18] J. Liu, Y. Shi, Z. M. Fadlullah, and N. Kato. “Space-air-ground integrated network: A survey”. In: *IEEE Communications Surveys and Tutorials* 20.4 (2018), pp. 2714–2741.

[Loo] *Loon*. URL: `https://x.company/projects/loon/`.

[Lu+19] J. Lu, Z. Zhang, T. Hu, P. Yi, and J. Lan. “A Survey of Controller Placement Problem in Software-Defined Networking”. In: *IEEE Access* 7 (2019), pp. 24290–24307.

[MA04] R. T. Marler and J. S. Arora. “Survey of multi-objective optimization methods for engineering”. In: *Structural and multidisciplinary optimization* 26.6 (2004), pp. 369–395.

[MA05] R. T. Marler and J. S. Arora. “Function-transformation methods for multi-objective optimization”. In: *Engineering Optimization* 37.6 (2005), pp. 551–570.

[Mal+19] F. Malandrino, C. Rottondi, C.-F. Chiasserini, A. Bianco, and I. Stavrakakis. “Multiservice UAVs for Emergency Tasks in Post-Disaster Scenarios”. In: *Proceedings of the ACM MobiHoc Workshop on Innovative Aerial*

Communication Solutions for FIrst REsponders Network in Emergency Scenarios. iFIRE '19. Catania, Italy: ACM, 2019, pp. 18–23.

[Mar] L. Martin. *Indago 3 - UAV*. URL: `https://www.lockheedmartin.com/en-us/products/indago-vtol-uav.html`.

[May+19] V. Mayor, R. Estepa, A. Estepa, and G. Madinabeitia. "Deploying a Reliable UAV-Aided Communication Service in Disaster Areas". In: *Wireless Communications and Mobile Computing* (2019).

[MB17] P. Mach and Z. Becvar. "Mobile Edge Computing: A Survey on Architecture and Computation Offloading". In: *IEEE Communications Surveys Tutorials* 19.3 (2017), pp. 1628–1656.

[McK+08] N. McKeown, T. Anderson, H. Balakrishnan, G. Parulkar, L. Peterson, J. Rexford, S. Shenker, and J. Turner. "OpenFlow: Enabling Innovation in Campus Networks". In: *SIGCOMM Comput. Commun. Rev.* 38.2 (Mar. 2008), pp. 69–74.

[Moz+19] M. Mozaffari, W. Saad, M. Bennis, Y. Nam, and M. Debbah. "A Tutorial on UAVs for Wireless Networks: Applications, Challenges, and Open Problems". In: *IEEE Communications Surveys Tutorials* 21.3 (2019), pp. 2334–2360.

[Nog+18] B. Nogales, V. Sanchez-Aguero, I. Vidal, and F. Valera. "Adaptable and automated small uav deployments via virtualization". In: *Sensors* 18.12 (2018), p. 4116.

[Nun+14] B. A. A. Nunes, M. Mendonca, X. Nguyen, K. Obraczka, and T. Turletti. "A Survey of Software-Defined Networking: Past, Present, and Future of

Programmable Networks". In: *IEEE Communications Surveys Tutorials* 16.3 (2014), pp. 1617–1634.

[Nvi] Nvidia. *Jetson TX2*. URL: `https://www.nvidia.com/en-us/autonomous-machines/embedded-systems/jetson-tx2/`.

[Ols+10] P. Olsson, J. Kvarnström, P. Doherty, O. Burdakov, and K. Holmberg. "Generating UAV communication networks for monitoring and surveillance". In: *11th International Conference on Control Automation Robotics Vision*. Dec. 2010, pp. 1070–1077.

[Onf] *Software-Defined Networking: The New Norm for Networks*. Tech. rep. Open Networking Foundation (ONF), 2012. URL: `https://opennetworking.org/sdn-resources/whitepapers/software-defined-networking-the-new-norm-for-networks/`.

[Ono] *Open Network Operating System (ONOS)*. URL: `https://opennetworking.org/onos/`.

[Ope] *OpenDayLight*. URL: `https://www.opendaylight.org`.

[Ope15] Open Networking Foundation. *OpenFlow Switch Specification. Version 1.3.5 (Protocol version 0x04)*. 2015. URL: `https://opennetworking.org/wp-content/uploads/2014/10/openflow-switch-v1.3.5.pdf`.

[Par+16] S. Park, H. Kim, K. Kim, and H. Kim. "Drone formation algorithm on 3D space for a drone-based network infrastructure". In: *IEEE 27th Annual International Symposium on Personal, Indoor, and Mobile Radio Communications (PIMRC)*. Sept. 2016, pp. 1–6.

[Ras+17] A. Raschellà, F. Bouhafs, G. C. Deepak, and M. Mackay. “QoS aware radio access technology selection framework in heterogeneous networks using SDN”. In: *Journal of Communications and Networks* 19.6 (Dec. 2017), pp. 577–586.

[RS17] C. Rametta and G. Schembra. “Designing a Softwarized Network Deployed on a Fleet of Drones for Rural Zone Monitoring”. In: *Future Internet* 9.1 (2017).

[RT12] L. Ramshaw and R. E. Tarjan. “On minimum-cost assignments in unbalanced bipartite graphs”. In: *HP Labs, Palo Alto, CA, USA, Tech. Rep. HPL-2012-40R1* (2012).

[SAP18] H. Sallouha, M. M. Azari, and S. Pollin. “Energy-Constrained UAV Trajectory Design for Ground Node Localization”. In: *IEEE Global Communications Conference (GLOBECOM)*. 2018, pp. 1–7.

[Say+16] S. Say, H. Inata, J. Liu, and S. Shimamoto. “Priority-Based Data Gathering Framework in UAV-Assisted Wireless Sensor Networks”. In: *IEEE Sensors Journal* 16.14 (July 2016), pp. 5785–5794.

[Seç+18a] G. Seçinti, P. B. Darian, B. Canberk, and K. R. Chowdhury. “SDNs in the Sky: Robust End-to-End Connectivity for Aerial Vehicular Networks”. In: *IEEE Commun. Mag.* 56.1 (Jan. 2018), pp. 16–21.

[Seç+18b] G. Seçinti, P. B. Darian, B. Canberk, and K. R. Chowdhury. “Resilient end-to-end connectivity for software defined unmanned aerial vehicular networks”. In: *IEEE International Symposium on Personal, Indoor and Mobile Radio Communications, PIMRC* 2017-Octob (2018), pp. 1–5.

[Seç+20] G. Seçinti, A. Trotta, S. Mohanti, M. Di Felice, and K. R. Chowdhury. “FOCUS: Fog Computing in UAS Software-Defined Mesh Networks”. In: *IEEE Transactions on Intelligent Transportation Systems* 21.6 (2020), pp. 2664–2674.

[Sey+16] M. Seyedebrahimi, F. Bouhafs, A. Raschellà, M. Mackay, and Q. Shi. “SDN-based channel assignment algorithm for interference management in dense Wi-Fi networks”. In: *European Conference on Networks and Communications (EuCNC)*. June 2016, pp. 128–132.

[Sha+17] V. Sharma, F. Song, I. You, and H. Chao. “Efficient Management and Fast Handovers in Software Defined Wireless Networks Using UAVs”. In: *IEEE Network* 31.6 (2017), pp. 78–85.

[Sha+19] H. Shakhatreh, A. H. Sawalmeh, A. Al-Fuqaha, Z. Dou, E. Almaita, I. Khalil, N. S. Othman, A. Khreishah, and M. Guizani. “Unmanned Aerial Vehicles (UAVs): A Survey on Civil Applications and Key Research Challenges”. In: *IEEE Access* 7 (2019), pp. 48572–48634.

[SK18] M. Y. Selim and A. E. Kamal. “Post-Disaster 4G/5G Network Rehabilitation Using Drones: Solving Battery and Backhaul Issues”. In: *IEEE Globecom Workshops (GC Wkshps)*. Dec. 2018, pp. 1–6.

[SO+20] O. Sami Oubbati, M. Atiquzzaman, T. Ahamed Ahanger, and A. Ibrahim. “Softwarization of UAV Networks: A Survey of Applications and Future Trends”. In: *IEEE Access* 8 (2020), pp. 98073–98125.

[Taj+19] M. M. Tajiki, S. Salsano, L. Chiaraviglio, M. Shojafar, and B. Akbari. "Joint Energy Efficient and QoS-Aware Path Allocation and VNF Placement for Service Function Chaining". In: *IEEE Trans. Netw. Service Manag.* 16.1 (Mar. 2019), pp. 374–388.

[Tak+18] A. Takacs, X. Lin, S. Hayes, and E. Tejedor. *Drones and networks: Ensuring safe and secure operations.* Tech. rep. 2018. URL: `https://www.ericsson.com/en/reports-and-papers/white-papers/drones-and-networks-ensuring-safe-and-secure-operations`.

[Tan+18] M. Tanha, D. Sajjadi, R. Ruby, and J. Pan. "Capacity-Aware and Delay-Guaranteed Resilient Controller Placement for Software-Defined WANs". In: *IEEE Trans. Netw. Service Manag.* 15.3 (Sept. 2018), pp. 991–1005.

[Too+12] A. Tootoonchian, S. Gorbunov, Y. Ganjali, M. Casado, and R. Sherwood. "On Controller Performance in Software-Defined Networks". In: *2nd USENIX Workshop on Hot Topics in Management of Internet, Cloud, and Enterprise Networks and Services (Hot-ICE 12).* 2012.

[Wan+17] G. Wang, Y. Zhao, J. Huang, and W. Wang. "The Controller Placement Problem in Software Defined Networking: A Survey". In: *IEEE Network* 31.5 (2017), pp. 21–27.

[Wan+20a] G. Wang, S. Zhou, S. Zhang, Z. Niu, and X. Shen. "SFC-Based Service Provisioning for Reconfigurable Space-Air-Ground Integrated Networks". In: *IEEE Journal on Selected Areas in Communications* 38.7 (July 2020), pp. 1478–1489.

[Wan+20b] H. Wang, H. Zhao, J. Zhang, D. Ma, J. Li, and J. Wei. "Survey on Unmanned Aerial Vehicle Networks: A Cyber Physical System Perspective". In: *IEEE Communications Surveys Tutorials* 22.2 (2020), pp. 1027–1070.

[Wan+20c] Y. Wang, Z. Y. Ru, K. Wang, and P. Q. Huang. "Joint Deployment and Task Scheduling Optimization for Large-Scale Mobile Users in Multi-UAV-Enabled Mobile Edge Computing". In: *IEEE Transactions on Cybernetics* 50.9 (2020), pp. 3984–3997.

[Whi+17] K. J. S. White, E. Denney, M. D. Knudson, A. K. Mamerides, and D. P. Pezaros. "A programmable SDN+NFV-based architecture for UAV telemetry monitoring". In: *14th IEEE Ann. Consumer Comm. Net. Conf (CCNC).* Jan. 2017, pp. 522–527.

[Yan+19] T. Yang, C. H. Foh, F. Heliot, C. Y. Leow, and P. Chatzimisios. "Self-Organization Drone-Based Unmanned Aerial Vehicles (UAV) Networks". In: *IEEE International Conference on Communications (ICC).* May 2019, pp. 1–6.

[Yao+14] G. Yao, J. Bi, Y. Li, and L. Guo. "On the Capacitated Controller Placement Problem in Software Defined Networks". In: *IEEE Commun. Lett.* 18.8 (Aug. 2014), pp. 1339–1342.

[Yen71] J. Y. Yen. "Finding the k shortest loopless paths in a network". In: *management Science* 17.11 (1971), pp. 712–716.

[YQR17] H. C. Yu, G. Quer, and R. R. Rao. "Wireless SDN mobile ad hoc network: From theory to practice". In: *IEEE International Conference on Communications (ICC)*. May 2017, pp. 1–7.

[Zac+17] I. Zacarias, J. Schwarzrock, L. P. Gaspary, A. Kohl, R. Q. A. Fernandes, J. M. Stocchero, and E. P. de Freitas. "Employing SDN to control video streaming applications in military mobile networks". In: *IEEE 16th International Symposium on Network Computing and Applications (NCA)*. Oct. 2017, pp. 1–4.

[Zha+19] N. Zhao, W. Lu, M. Sheng, Y. Chen, J. Tang, F. R. Yu, and K. Wong. "UAV-Assisted Emergency Networks in Disasters". In: *IEEE Wireless Communications* 26.1 (Feb. 2019), pp. 45–51.

[Zhe+19] X. Zheng, M. Li, M. Tahir, Y. Chen, and M. Alam. "Stochastic Computation Offloading and Scheduling Based on Mobile Edge Computing". In: *IEEE Access* 7.2 (2019), pp. 72247–72256.

[ZWZ18] X. Zhang, H. Wang, and H. Zhao. "An SDN framework for UAV backbone network towards knowledge centric networking". In: *IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. 2018, pp. 456–461.

[ZYS11] P. Zhan, K. Yu, and A. L. Swindlehurst. "Wireless Relay Communications with Unmanned Aerial Vehicles: Performance and Optimization". In: *IEEE Transactions on Aerospace and Electronic Systems* 47.3 (July 2011), pp. 2068–2085.

[ZZL16a] Y. Zeng, R. Zhang, and T. J. Lim. “Throughput Maximization for UAV-Enabled Mobile Relaying Systems”. In: *IEEE Transactions on Communications* 64.12 (Dec. 2016), pp. 4983–4996.

[ZZL16b] Y. Zeng, R. Zhang, and T. J. Lim. “Wireless communications with unmanned aerial vehicles: opportunities and challenges”. In: *IEEE Communications Magazine* 54.5 (May 2016), pp. 36–42.

[ZZZ18] C. Zhan, Y. Zeng, and R. Zhang. “Energy-Efficient Data Collection in UAV Enabled Wireless Sensor Network”. In: *IEEE Wireless Communications Letters* 7.3 (2018), pp. 328–331. arXiv: `1708.00221`.

Appendix A

Notes On Optimization

A.1 Linearization of Non-Linear Terms

The non-linear terms in the objective function (5.4) and in the constraint (5.11) for both initial and dynamic MinCtl and FixedCtl schemes can be linearized using [Tan+18; CBD11] by introducing a binary variable z_{ij} to replace the product of y_{l_i} and y_{l_j}. Then, we replace the constraint (5.11) with the following constraints:

$$z_{ij} \leq y_{l_i} \qquad \forall l_i \in L \tag{A.1}$$

$$z_{ij} \leq y_{l_j} \qquad \forall l_j \in L \tag{A.2}$$

$$z_{ij} \geq y_{l_i} + y_{l_j} - 1 \qquad \forall l_i, l_j \in L, i \neq j \tag{A.3}$$

$$z_{ij} L^{cc}_{l_i l_j} \leq hops^{cc}_{max} \qquad \forall l_i, l_j \in L, i \neq j \tag{A.4}$$

Then, objective function (5.4) is expressed as:

$$H_{total} = \sum_{l_i, l_j \in L, i \neq j} z_{ij} L^{cc}_{l_i l_j} \tag{A.5}$$

A.2 Multi-Objective Optimization

In this section, we discuss the approach used in Chapter 6 to solve static and dynamic schemes with multiple objective functions.

Multiple, sometimes conflicting, optimal solutions can exist for a multi-objective optimization problem. There exist different techniques for solving multi-objective optimization [MA04]. The hierarchical or lexicographic approach as used in Chapter 5, ranks objectives by priorities set by the decision maker and solves each objective separately without compromising solutions of higher priority objectives. Another approach is to generate the set of Pareto optimal solutions [MA04] which are the set of solutions where each objective value cannot be improved without compromising other objective values. This allows the system or decision maker to select the preferred solution. However, the above techniques are computationally expensive due to the number of objective functions. Due to the dynamic and iterative nature of the dynamic adjustment, solutions must be obtained in the shortest amount of time. We use the weighted sum approach to combine all objective functions into a single function, where weights dictate the importance and contribution of each individual objective function to the total value according to the preference of the system.

Since objective function values can have different magnitudes, it is recommended to normalize or rescale them so it is easier to set the weights. We normalize the objective functions by their minimum and maximum feasible values. Each objective function f_i is transformed into $f_i^{norm} = \frac{f_i - f_i^{min}}{f_i^{max} - f_i^{min}}$. Ideally, such normalization entails finding the minimum and maximum feasible objective function values by solving for each objective separately or obtaining such values from the Pareto set [MA05]. However, to reduce the time required to obtain a solution, we normalize by estimated values for each objective function based on problem intuition and experimentation. With normalization, all objective functions are assumed to be in the range $[0, 1]$ and $\sum_{i=1}^{5} w_i = 1$.

All objective functions are transformed as described above. Rough estimations of the feasible minimum and maximum objective values are made using formulas shown and discussed below. The weights given to each objective function are shown in Table 6.1. We experimented with different variations of these formulas and weights and as well without normalization which did not alter the overall finding of the evaluation presented in Chapter 6. It is expected that in different settings, system implementers would experiment with different weights and bounds. Estimated minimum and maximum objective values are calculated as follows.

- $N_{deployed}^{min}$: At minimum, only primary drones are deployed which is determined by the number of required locations by SFCs $|L_{req}|$.
- $N_{deployed}^{max}$: The number of deployed drones is limited by $|D|$.
- N_{vlinks}^{min}: The lower bound of the number of VNF links that use physical links is equivalent to at least one VNF link per SFC, i.e., at least one hop is required for SFCs that start from one location and ends at another. This is equivalent to $|R|$.
- N_{vlinks}^{max}: The upper bound of the number of VNF links that uses physical links is bound the total count SFCs VNF links:

$$N_{vlinks}^{max} = \sum_{r \in R} \mid VL_r \mid \tag{A.6}$$

- $\Delta(X,\hat{X})_{min}, \Delta(Y,\hat{Y})_{min}, \Delta(K,\bar{K})_{min}$: The minimum Hamming distance can be zero, meaning no changes happened to the network state in consecutive orchestration steps.

- $\Delta(X,\hat{X})_{max}$: The maximum number of VNF relocations is bound by the total number of intermediate VNFs in the network (All VNFs minus first and last VNFs of each SFC). Each VNF relocation involves a Hamming distance of two.

$$\Delta(X,\hat{X})_{max} = 2\sum_{r\in R}|F_r| - 2 \tag{A.7}$$

- $\Delta(Y,\hat{Y})_{max}$: The maximum number of changes to VNF link placements is hard to estimate but we set it to the total number of SFCs VNF links:

$$\Delta(Y,\hat{Y})_{max} = \sum_{r\in R}|\ VL_r\ | \tag{A.8}$$

- $\Delta(K,\hat{K})_{max}$: The maximum number of changes to network link states is also hard to valuate, but we set it to the number of active physical links formed after the last orchestration step:

$$\Delta(K,\hat{K})_{max} = \sum_{(d_i,d_j)\in E}\hat{k}_{d_i,d_j} \tag{A.9}$$