Investigation into Control of Swarm Drones

Alexander Stewart MacKerron

School of Civil &. Mechanical Engineering Project Thesis
Curtin University

29/10/2024

30 Resolute Way
Ocean Reef,
WA 6027

29/10/2024

The Head
School of Civil & Mechanical Engineering,
Curtin University,
Kent Street,
Bentley,
WA 6102

Dear Sir,

I submit this thesis entitled “Investigation into Control of Swarm Drones”, based on MXEN4000 Mechatronic Engineering Research Project 1 and MXEN4004 Mechatronic Engineering Research Project 2, undertaken by me as part-requirement for the degree of B.Eng. in Mechatronic Engineering.

Yours faithfully,

Alexander Stewart MacKerron
20174943

Acknowledgments
I would like to acknowledge my supervisor Yifei Ren for helping me in this project, despite his busy schedule. I would like to additionally extend this gratitude to Siavash Khaksar for recommending Yifei to me as a project supervisor.

I would also like to acknowledge my family, for being incredibly supportive, understanding and helpful during this last semester.

To my colleagues and to my friends, I also thank them for allowing me to enjoy my final semester in great company. I couldn’t ask for a better set of people in my life.

Abstract
Drone swarms are a relatively novel technology with lots of avenues of research. In particular, communication within swarms is of interest, as there are many significant challenges that prevent it from having a singular simple solution. One primary way in which drones are established to communicate with each other is through a topology; a network structure that defines the main routes by which information is permitted to travel. In this investigation, three topologies are evaluated against each other. These include a Bus Topology, a Tree Topology, and a third topology proposed within this report. The proposed structure defines all potential links, then connects to the top-ranking ones based on maximising the signal quality of the selected links. This is done in the hopes of creating a stronger topology that is more resistant to adverse environments and scenarios. Topologies were evaluated based on their average XYZ positions and delays across multiple simulations, with varying drone spacing and speeds. The results proved that this is not as effective as it could be and will require additional work in order to refine outcomes. The Bus Topology performed poorly as expected, and the Tree Topology was surprisingly effective, though it may not perform similarly in more adverse conditions.

Nomenclature

Abbreviation	Meaning
AODV	Ad Hoc On-Demand Distance Vector Routing
AWGN	Added White Gaussian Noise
CH	Cluster Head
CM	Cluster Member
FANET	Flying Ad-Hoc Network
FSPL	Free Space Path Loss
GPD	gym-pybullet-drones
IDE	Integrated Development Environment
LOS	Line of sight
MANET	Mobile Ad-Hoc Network
SNR	Signal to Noise Ratio
UAV	Unmanned Aerial Vehicle

TABLE OF CONTENTS

TABLE OF FIGURES viii

TABLE OF TABLES ix

PAGES REFERENCING PROGRESS REPORT ix

1.0 INTRODUCTION 1

2.0 BACKGROUND 3

2.1 Challenges in Swarm Drone Technology 3

2.1.1 Limited Battery Power 3

2.1.2 Wireless Security 3

2.1.3 Optimised Pathways 3

2.1.4 Collisions 4

2.1.5 Reliable Intra-Swarm Communication 4

2.1.6 Swarm Homogeneity or Heterogeneity 5

2.1.7 Interesting Developments in the Swarm Drone Space 5

2.2 Established Ideas in FANETs 6

2.2.1 Routing Protocols 6

2.2.2 Topologies 7

2.2.3 Proposed Topologies 9

2.2.4 Flooding vs Routing 10

2.3 Specifications 11

2.3.1 Wi-Fi (IEEE 802.11) 11

2.3.2 Zigbee (IEEE 802.15.4) 11

2.3.3 LoRa 11

2.3.4 Selected Standard 12

2.3.5 Values For Use in The Simulator 13

2.4 Existing Simulators for swarm drones 13

2.4.1 Desired Qualities 13

2.4.2 Existing Simulators 14
2.4.3 Selected Simulator 14
2.5 Understanding gym-pybullet-drones 15
2.6 Legal Constraints Regarding Swarm Drones in Australia 18
3.0 EXPERIMENTAL PROCEDURE 19
3.1 Thesis Progress 19
3.1.1 Main Focus Established 19
3.1.2 Graph Representation of a Swarm 23
3.1.3 Topologies Used 23
3.1.4 Incorporating Delays into GPD 26
3.1.5 Simulation Parameters 28
3.1.6 Assumptions Made 30
3.1.7 Results Gathering 30
3.2 Materials and Equipment 31
3.3 Method 31
4.0 RESULTS AND DISCUSSIONS 33
4.1 Results Gathered 33
4.1.1 XYZ Positions 33
4.1.2 Position Error 37
4.1.3 Delays 39
4.2 Discussion Of Results 42
4.2.1 XYZ Positions 42
4.2.2 Position Error 43
4.2.3 Delays 44
4.2.4 Optimal Topology Inconsistencies 45
4.2.5 Summary of Observations 45
4.3 Comparison With Previous Works 46

5.0 CONCLUSION 48
6.0 FUTURE WORK 49
6.1 Reinforcement Learning 49
6.2 Power Modelling 49
6.3 Dynamic Linking 49
6.4 Algorithm Effectiveness 49
6.5 Simulator 49
6.6 Additional Areas of Interest 50
7.0 REFERENCES 51

TABLE OF FIGURES

Figure 1: A visual representation of message routing. Drone 1 can communicate with Drone 3 through its common neighbour Drone 2. 7
Figure 2: Examples of some network topologies in a swarm context. Based on evaluated topologies from [5] 8
Figure 3: The visualisation of the simulation generated by running fly.py 15
Figure 4: Logger plot outputted from an example flight in the gym-pybullet-drones simulator. This shows a singular drone flying upwards 2 metres and maintaining its position. 16
Figure 5: Debugger outputs for the waypoint counters array and target positions array. The waypoint counters point to an XYZ position in the target positions array for each drone in the simulation. 18
Figure 6: A visual representation of the link budget of a typical transmitter and receiver set up 20
Figure 7: Drone swarm with labelled IDs 24
Figure 8: Bus Topology 25
Figure 9: Tree Topology 25
Figure 10: Optimal Topology 26
Figure 11: Connections that each drone made in the Optimal Topology in Figure 10 26
Figure 12: Example of how delay increases across the swarm 27

Figure 13: The Run Configurations window in PyCharm IDE 28
Figure 14: Step-by-step guide for reproducing results..... 32
Figure 15: Average XYZ Positions of a Drone Swarm in a Bus Topology..... 34
Figure 16: Average XYZ Positions of a Drone Swarm in a Tree Topology..... 35
Figure 17: Average XYZ Positions of a Drone Swarm in an Optimal Topology..... 36
Figure 18: Position error as a percentage on a Bus Topology 37
Figure 19: Position error as a percentage on a Tree Topology 38
Figure 20: Position error as a percentage on an Optimal Topology 39
Figure 21: Messaging delays incurred for each drone in a Bus Topology..... 40
Figure 22: Messaging delays incurred for each drone in a Tree Topology 41
Figure 23: Messaging delays incurred for each drone in the proposed Optimal Topology 42

TABLE OF TABLES

Table 1: Variables in the algorithm used, and their values along with their sources. 22
Table 2: List of all arguments used and all of their values in the simulations..... 29

PAGES REFERENCING PROGRESS REPORT

The progress report from the previous semester has been used to aid in filling sections such as the background of this report. Some of these have been adapted or changed outright, but others have been directly copied. The following chapters have elements that are taken from the progress report in some manner:

Chapter 1.0 page 1 first paragraph

Chapter 2.1.1 on page 3 through to Chapter 2.2.1 on page 6

Chapter 2.2.2 on page 7, except first two paragraphs and figure

Chapter 2.2.3 on page 9, except final paragraph

Chapter 2.2.4 on page 10, first paragraph

Chapter 2.3.1 on page 11 through to Chapter 2.3.4 on page 12, first paragraph

1.0 INTRODUCTION

Swarm drones are a rapidly evolving technology, becoming more and more popular in both civil and commercial applications. Drone swarms are increasingly being used in many fields, such as light displays, search and rescue, agriculture, and surveillance. Recent research investigated the use of swarms for extending the reach of 5G networking [1]. There are also use cases in the military field, such as coordination and location of soldiers on the ground, target finding, and ground strikes. Another research paper uses an autonomous UAV swarm to track and follow a malicious UAV [2], such as the one that deliberately flew through London Gatwick Airport in 2018. Such swarms make use of a high mobility platform to rapidly deploy sensors, activate payloads or link devices to a larger network, in a highly scalable manner that allows them to be used in almost any scenario. There are undoubtedly many more implementations of swarm technology beyond this, which will be discovered as our understanding and the technology advances.

This thesis project demonstrates a model for creating a swarm drone topology based on maximising the signal strength between linked nodes. By calculating the link budget and signal-to-noise ratio of each potential link for each drone, and then selecting the best two, the topology aims to deliver a robustly constructed, highly redundant structure. This is hypothesised to enable a consistently interconnected, decentralised swarm across the duration of its flight.

The project began by outlining the scope, objectives and research goals to establish a solid foundation of work. Every step taken across the semester was documented in a journal to track progress and ideas for the project, as they developed over time. Initial research involved extensive exploration into the field of swarm drones. Some challenges were identified, limiting the current effectiveness of swarm technology. These included the limited battery and computing power of drones, poor wireless security, minimal collision detection, and a lack of reliable intra-swarm communications. Some noteworthy developments in the space were also observed and recorded in chapter 2.1.7. By this point, it was decided the investigation would focus on the communication aspect of swarm drones. Research shifted to place a heavier focus on this element, and established theories and techniques were uncovered. These included concepts such as routing protocols and topologies in swarm communication. As topologies seemed to be a largely imperfected topic

within the field, this aspect guided the course of action for the project. To this end, preliminary equations, values and parameters were researched to guide testing. After reviewing a wide range of available simulators, gym-pybullet-drones was selected as the platform for testing proposed scenarios in this thesis. This particular testing environment was conducive to rapid development and repeatable research. The selected simulator was further investigated for its qualities and uses, and subsequently tailored to suit the project's requirements. Out of interest, some relevant legal constraints, such as CASA regulations, were also reviewed for any real-world applications of swarm technology in Australia.

Understanding the scope of knowledge available is the first step via a literature search. Once this was complete, proper development and testing of the swarm topologies could begin. The full process for this is discussed in detail in chapter 3.0. The functionality of gym-pybullet-drones was evaluated, to find out what it was capable of supporting for the purposes of investigations into swarm drone communications. The effect of a topology on a swarm's performance became an intriguing topic to investigate, and so work was particularly focused in that area. Once the aforementioned topology method which focuses on signal strength was proposed for implementation, differing topologies were then reviewed. The aim was to compare their performance against the proposed configuration. Further modifications were made to the simulator's example code to tailor it to the specific use case and make it work for the given scenarios.

Results were then recorded to provide data driven insights into each topology's individual performance. The obtained data included swarm XYZ positions across the simulations that were run, positional errors and any messaging delays incurred by multi-hop routing protocols.

2.0 BACKGROUND

Research into the topic of swarm drones began by initially reviewing the current state of the field, and then working inwards to find out opportunities for further work to be done. First, challenges were identified, along with any interesting developments in the field that were observed along the way. Then the focus was narrowed by looking at established ideas, where it was decided that communications was an area in which this investigation should predominantly concentrate. Finally, some simulators were investigated for their respective merits in modelling swarm drones.

2.1 Challenges in Swarm Drone Technology

At present, there are still many challenges to the development of this technology.

2.1.1 Limited Battery Power

The available battery power in each drone is one such hurdle [3]. The active time of a drone is determined by its battery size, but it is simultaneously constrained by the amount of weight that it can lift. The lighter the drone, the more mobile it is, and the more energy efficient it will be. This means that there is an optimisation to be made in the selection of battery size, but this usually still leaves the drones with a flight duration measured in tens of minutes. This ultimately prevents drone swarms from operating for very long, as operators must also allow time for their return to the base station to recharge safely.

2.1.2 Wireless Security

Security for any wireless device is an established concern, and one which is also prevalent in drone swarms [4]. Messages are relayed in large quantities, at a high frequency, and as there isn't much capacity for high computational complexity on a drone, security is a reasonably large flaw in the emergent technology. Drones are capable of being hijacked and can be used as weapons or as an otherwise destructive tool. As such there is an ethical responsibility to identify a secure way to coordinate the drones in a swarm. The wireless connections in a swarm network are susceptible to attacks and are a viable point for consideration in the design.

2.1.3 Optimised Pathways

Finding an optimised path from one GPS point to another is an established challenge across many aspects of automation. In a dense, obstacle-rich environment such as a city, finding a path from one location to another can be difficult for a drone, where decisions may need to be executed quickly and with limited sensor data.

Furthermore, there is a much larger margin for error, as flying close to the ground and certain obstacles can induce ground effect. This is where the air pushed by the drone can reflect from a surface, disrupting the drone's flight. Alternatively, wind can push a drone into an obstacle and cause a collision. For autonomous swarms to exist, pathing optimisation must exist in such a way that it guides the drones on a completely safe path, ideally without taking too much time to calculate.

2.1.4 Collisions

Similarly, mid-air collisions are a great concern to any aerial vehicle, and especially for drones that operate autonomously. To avoid collisions, the drones in the swarm must have an acute awareness as to where they are in relation to others in the swarm, and where any unforeseen obstacles may be approaching.

2.1.5 Reliable Intra-Swarm Communication

Drone swarms are typically organised into clusters, with one drone allocated as a cluster head (CH) and other drones in its communications radius are cluster members (CM). This simplifies how information is sent to the swarm. By targeting only one or a few heads, the entire swarm can rally around the heads and follow the instructions issued by them. Due to the dynamic nature of a flying network, this technique usually works well, as CMs can detach from one head and join another cluster, say for example if they fall behind, or if they move on to perform a task in an area outside of the reach of the original CH. Due to the high mobility and dynamic topology of a Flying Ad-Hoc Network (FANET), achieving reliable communication is an ongoing challenge in the research field for this technology [4]. When a disruption occurs, the drones must be able to self-organise to reconnect themselves.

Failures of any kind can be catastrophic for the network, as a structure reorganisation may be required [4]. In addition, depending on how messages are sent, if the CH fails then the entire network may follow. The equipment in question involves fast moving dense objects falling from great heights, potentially with rotor blades still spinning, and so is particularly dangerous. This is hazardous to people or property beneath the swarm and could cause serious damage to the environment upon failure. For this reason, drone lightshows typically operate over a body of water or controlled zone, to avoid injuries to spectators and property.

2.1.6 Swarm Homogeneity or Heterogeneity

This final challenge is less of a concern, but more so a design consideration when establishing a swarm. Homogeneity is when all elements of a group are all the same, and in a swarm's case this refers typically to the drones themselves. If all have the same flight computer, chassis, motors, payloads, and batteries among other components, the swarm will be much simpler to coordinate. If every node behaves the same, it is simpler to model their behaviours. Furthermore, heterogeneity can end up forgoing a primary benefit of swarms: redundancy. In a homogeneous network, a single drone failure only causes a problem for any neighbouring drones that relied on it for communication. With a heterogeneous swarm, assuming the same number of drones, a single failure is much more catastrophic and could potentially mean the end of a whole mission. Despite this, much research has been invested into heterogenous solutions, perhaps the most interesting of being SwarmGear [5]. This involves a large drone with robotic legs for ground transport, and three follower drones. Darush et al. praise heterogeneity, as "…heterogeneous swarms …are shown to have better performance in missions than homogeneous swarms of similar size." This assertion is owed to the set of complementary skills that each agent in a swarm could bring to bear, which is an appreciable advantage.

2.1.7 Interesting Developments in the Swarm Drone Space

Many existing research papers attempt to solve some of these issues discussed in 2.1, or otherwise offer some solutions.

2.1.7.1 Position Estimation

One novel idea proposed for intra-swarm collision avoidance used Bluetooth signal strength to estimate the relative position of a drone and its neighbours [6]. This makes use of an onboard device to send messages to other members of the swarm within range. The drones exchange state information, such as height, velocity, and position, and on the receiving end, the signal strength can be used alongside this data to infer relative proximity to other drones. The benefit of using a Bluetooth transceiver here is in its lightweight and low-power specification, which minimises the impact on flying time for the drones. It uses a short-range signal to detect the proximity of other drones in the swarm and doesn't need to be very long-range. Hypothetically this strategy could also translate for other wireless standards, such as Wi-Fi.

2.1.7.2 Timer Synchronicity

Alsolami et al. developed a method by which a swarm could better synchronise its timers [7]. Poorly synchronised clocks can cause interference in messaging, affect spectrum precision and interrupt the operation of the transceiver. Prior methods had focused on single hop skews in the timer to regulate connected neighbours in the swarm. This new method uses selected intermediate drones within two clusters to synchronise all the connected drones between them. The report determines that the new method minimises the overhead traffic and is a significant improvement on the existing single hop skew method.

2.2 Established Ideas in FANETs

The concepts discussed here are well defined in swarm technology and backed up by years of research.

2.2.1 Routing Protocols

Most reports suggest that communication strategies for swarm drones are the deciding factor on their effectiveness [1]. This intuitively makes sense, as communications direct the swarm's formation and movements, and how drones inform each other of their relative position in the swarm. Without this data, the swarm may catastrophically fail, causing drones to initiate emergency landings or worse, attempt a manoeuvre with no information.

Routing protocols form the basis of communication between the nodes of a network [8]. In the case of swarm drones, the nodes are each individual drone, and the terms are often used interchangeably. FANETs use a routing protocol to find and reach drones in a network of connected drones. Drones immediately reachable by a transmitting drone are considered to be in its 'neighbourhood', and these neighbours can forward messages to other drones outside of the original drone's communication radius. A visual representation of this is shown in Figure 1, where originally Drone 1 cannot send a message to Drone 3 directly, it must route through Drone 2 to do so. Drone 3 requires two hops to reach, in this example.

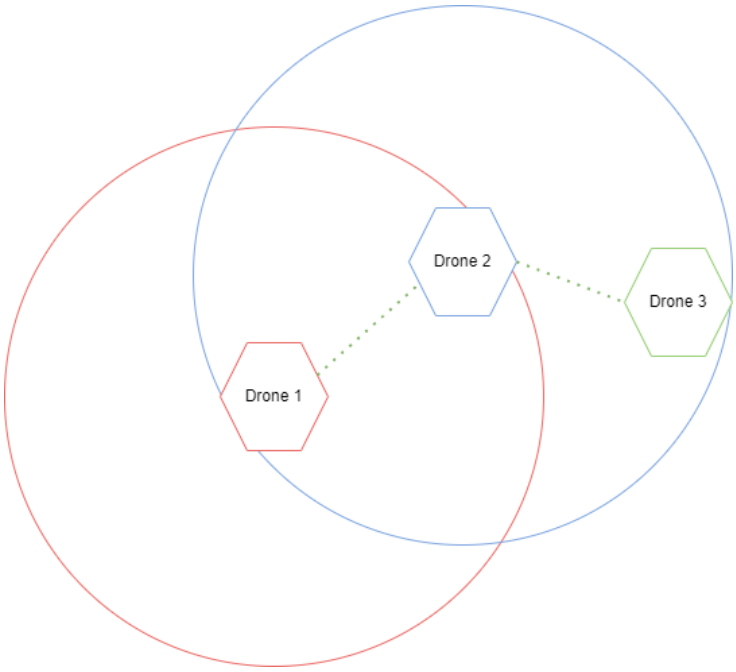

Figure 1: A visual representation of message routing. Drone 1 can communicate with Drone 3 through its common neighbour Drone 2.

This routing approach means that information such as flight direction, updated position, or mid-air collision warnings may be relayed through the network regardless of the distance between the start and destination nodes. This approach does require sufficient numbers of intermediary nodes to facilitate communication, and if a network is not dense enough, the message may not reach its destination.

2.2.2 Topologies

Swarms are often organised under a topology, which is an established map of any connections between devices. By defining these links before or during flight, more efficient paths can be identified through the network. A network's effectiveness can succeed or fail entirely based on the links defined in its topology. Points of failure can be determined by the lack of links to a node, and so the primary safeguard against this is maintaining redundant links in the topology to this weak point. Figure 2 shows an example of some networking topologies that have been used in a heterogenous swarm, found in research [5].

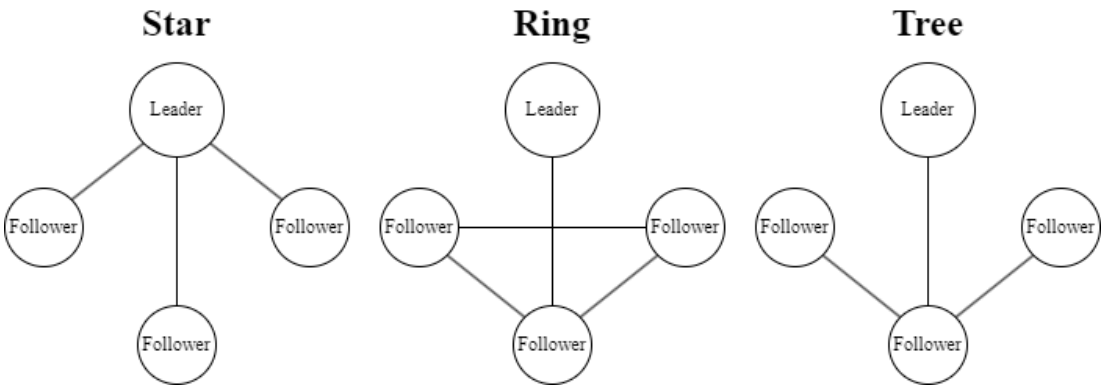

Figure 2: Examples of some network topologies in a swarm context.

There are two main types of topologies that can exist. A centralised topology has each drone in a swarm in direct contact with the base station. This allows for faster decision making and efficient communications but may suffer from a point of failure. Decentralised topologies distribute the decision-making process across multiple UAVs, which allows for better resilience to faults, but introduces synchronisation and coordination issues [9].

A routing protocol based on any topology maintains an up-to-date routing table, which references the link paths available from the current node to other nodes in the network. This can be automatically maintained either proactively with messages regularly updating the table for each node, or reactively by asking the network what route it should take when a message is about be sent [10].

The primary difference between Mobile Ad-Hoc Networks (MANETs) and FANETs is that FANETs add the additional challenge of working in 3D space, instead of merely 2D. Height changes the ways in which nodes of an Ad-Hoc Network can connect in a directional sense, meaning that nodes may not be optimally connected if managed with a typical topology-based method. Furthermore, a drone swarm is highly dynamic; the nodes can at any point move further away from a given position, breaking and reforming links to other nodes as the drone moves to new locations within the swarm.

These two factors mean that directly implementing a topology-based protocol that works for MANETs is unlikely to work - or at least will not be as efficient as one that is designed especially for FANETs. One report [10] suggests that topology-based protocols will not scale well in a swarm and may only be effective for small

clusters. Another [11] concedes that topology-based protocols are still useful, within certain circumstances. For example, the swarm must move slower through the air, and the size must be constrained. It also pointed out that they consume more energy and resources in the form of bandwidth and memory. For topology-based protocols to be more efficient in FANETs, additional variables must be considered, which may end up exceeding the storage capacity of a drone's limited hardware. Deciding the type of routing protocol to be used is therefore a critical design consideration when establishing a network.

2.2.3 Proposed Topologies

Armir Bujari et al. advocate for the use of position-based packet routing protocols in FANETs [10]. This is a process where messages are sent based on the positions of nodes and their neighbours, rather than their link map in a topology. While the difference appears negligible, this means that if one drone wants to send a message to another, all it needs to know is its position, and it can forward the message to the drone in its neighbourhood closest to the destination. In a topology-based routing protocol, this would usually follow a chain of highest link quality, which may involve multiple hops between nodes to reach the destination. Minimising the number of hops to reach the destination node reduces messaging delays, reduces data corruption risk, and improves the security of the swarm.

Oubatti et al. [11] propose multiple position-based methods, some of which are covered in the latter report, but one of which involves the drone holding onto the message and moving within range to deliver it. This eliminates the issue of a local minimum occurring, in which a route to the destination drone cannot be found by the routing protocol, for example as a result of disconnections separating two halves of a swarm. It does introduce complications from additional pathing calculations now being required to deliver a message and significant delays in its reception. However, for certain low-swarm-density applications, this may be considered acceptable if it guarantees the delivery of the message.

Meanwhile, Bautista et al. adapt an IEEE 802.11 wireless standard into working for FANETs [8]. The typical routing protocol in 802.11 is called Airtime and uses a metric derived from the time resources used by a link to test it and find errors. The paper proposes a new method called SrFTime, to better fit the link characteristics of drones to help determine the best routes from drones to their CHs. It was found that

the new metric is much better suited for FANET use, based on simulation results showing improved network throughput.

For larger scale swarms, Chen et al. developed the idea for a fast coordination approach [12]. As with many swarming techniques, this method uses a bio-inspired pattern. It cites bird flocks as the main inspiration for how the drones intend to act in this procedure, and it uses local drone interactions as the basis for quick predictions. The authors acknowledge that as a swarm's size increases, its ability to ensure timely communication becomes increasingly challenging. The swarm is organised into a tree, with the leader at the root node, and coordination is achieved by communicating through each drone's father. In a dangerous environment, the father drone's connection may be lost, so drones make predictions, by inspecting their neighbours. It is quite rare that a drone will lose all its neighbours, so when it isn't controlled by the leader via a chain of father drones, it maintains coordination with the leader by following neighbours. It will also do this in the event of packet loss, creating a stable and secure method of maintaining the topology's integrity.

2.2.4 Flooding vs Routing

Certain methods also make use of a flooding technique, as opposed to a routing approach [13]. Flood messages are often used to guarantee that every node in a network receives a particular message, with little care as to whether a particular node has already received that information. This can be useful, for example, when trying to alert the swarm of an incoming obstacle that must be evaded, or when trying to obtain data about the entire swarm. Whereas routing techniques try to find, maintain, and repair optimal paths across the network, flooding is simplistic and highly reliable. It also leads to a reduction in computing power and memory requirements, thereby increasing flight time and reducing cost. The main challenges to this method arise around the power spent transmitting to nodes that have already received the message.

For this project, it was decided to opt for routing protocols over flooding. Flooding can be redundant, depending on the objective of the message being sent. Sending a message to every single drone in the swarm to tell Drone 3 to move is a waste of computation and transmission power, which in a swarm context is a valuable resource.

2.3 Specifications

Research was invested into how exactly the swarm should be modelled for this report. The first semester of this research project was focused on the communication protocols available, whereas the second semester's research narrowed down the remaining specifications.

2.3.1 Wi-Fi (IEEE 802.11)

Wi-Fi is the simplest method by which drone communication can be managed. It is widely used, well supported, and offers high transfer rates of data [1] [4]. The range at which messages can still be received reaches into the hundreds of meters, providing more than enough range to communicate between each drone. However, it also has some attributes that are less desirable. It draws a lot of power in operation, which on a battery-operated device such as a drone, may not be ideal. Depending on the version of 802.11 used, there may be issues with sending data between mobile nodes. According to a study made in 2016 [14], 802.11a, b and g all use lower data rates at short range which is not ideal for the sudden fast movements made in a FANET, while 802.11n (either 2.4 or 5GHz versions) offers better performance. There may exist newer wireless protocols that are better than 802.11n that warrant some investigation.

2.3.2 Zigbee (IEEE 802.15.4)

This protocol typically uses Ad Hoc On-Demand Distance Vector Routing (AODV) [15] to manage and optimise its routing to great effect. It is a low power option, which works well for increasing the flight time of the drones. Mesh networking is well supported on Zigbee as well, another highly desirable trait in swarm operations. It can support long range communications, up to 6.5km at maximum [16], though this is expected to have some diminishing returns. In one comparative study, it claims that despite this long-range, ZigBee is still best at operating within the tens of meters and should be used to control intra-swarm communications [17]. It is also quite reliable in noisy environments, due to its low data rate. This, however, is also one of its drawbacks: sending data at a rate of around 250kb/s may be too slow depending on what information is being sent between the drones.

2.3.3 LoRa

Another common protocol is LoRa, known in IoT spaces for its long range and low power consumption. Both are good traits for a drone swarm application. As a bonus, they also have good obstacle penetration, although as already mentioned this will not

be represented in the simulation. They do, however, have a low transfer rate which constrains the size of messages sent, and potentially causes latency issues as well [18]. There is also a requirement for gateway infrastructure, which adds further complexity to the design structure on the drones themselves. As such, many existing implementations of LoRa in swarms exists as an in-between for the base station and a gateway drone. In [18], they use LoRa to manage long range communications out to the swarm and uses IEEE 802.11 to manage the shorter-range LOS communications. This can create points of weakness in the network, such that if the gateway drone loses connection, then the entire swarm is no longer controllable from the base station.

Interestingly, some applications of LoRa and LoRaWAN use the swarm drones as a temporary branch for hard-to-reach nodes of a sensor network, such as providing firefighters with end-to-end connectivity to the command post in a dense wilderness environment [19]. Another implementation similarly uses a LoRaWAN with drone swarms acting as gateways to the sensors, such as urban surveillance cameras [20].

2.3.4 Selected Standard

To maintain the simplicity of the simulation, only one protocol should be chosen, rather than opting for a combination of two. This excludes LoRa as a solution immediately, as it is largely implemented as a method to communicate with the base station in tandem with IEEE 802.11 operating the network, which for the purposes of the simulation is unnecessary. In a particular comparative study [17], LTE, Wi-Fi and ZigBee are directly compared to each other in terms of their qualities and where they have previously seen use. LTE often is used in the long-range communication link, such as that between a base station and a swarm, rather than between the drones themselves. As such it is disregarded for this investigation. Wi-Fi and ZigBee both see use in intra-swarm communications, though Wi-Fi tends to be used where high data rates are needed, such as for image transfer, path optimisation, and providing emergency networks. ZigBee on the other hand, appears to provide exactly what is required, in a cluster communication application [21]. It uses various ad-hoc routing protocols including AODV, and concludes that IEEE 802.15.4 can be a good choice for FANETs if there isn’t a large bandwidth or data rate requirement.

Supporting this idea, Kazeem et al. [22] directly compared ZigBee, Wi-Fi and Bluetooth to one another. This found that ZigBee was a good, low powered,

adequately ranged middle ground between the two protocols. Swarm drones place a priority on lightweight, low powered devices for communication, with longer communication ranges held in high regard. This comfortably asserts that ZigBee is a good choice for use in the application of swarm drones.

2.3.5 Values For Use in The Simulator

As ZigBee has been selected, it is important to consider what parameters can be used with it, for the purposes of properly simulating the swarm. More specifically, it should consider what characteristics can be represented in simulation. To this end, research was performed to find some suitable components to take values from. An immediate solution was found with the XBee-PRO RF Module range [16] [23], a simple and budget solution for implementing ZigBee into customised devices. With detailed specifications and documentation, it was easy to find all details required, such as range, data rate, receiver sensitivity, and transmitter power. A full list of all parameters used in the simulator can be found in Table 1 in chapter 3.1.1.

2.4 Existing Simulators for swarm drones

A simulator must be decided for the purposes of determining an outcome for this project.

2.4.1 Desired Qualities

It is broadly recognised that there is a requirement for simulators capable of modelling multiple drones at once, along with their effects in communications [24] [9]. An article written by Ling et. al [24] discusses the state of simulating UAV swarms as of 2021, critiquing the available options. They claim that existing simulation platforms fail to “facilitate development…for cooperative UAVs using imperfect perception." The current simulation options are designed with a specific purpose in mind and are difficult to extend from. The article also claims that the modelling of these platforms is “too simplified” for dynamic swarms and communication modelling. These issues can cause a difference between simulated effects and real-world performance, which is certainly undesirable. The paper concludes by proposing that a platform with an effective planning and perception algorithm, flexible definitions of motion dynamics and sensor models, and ideally no commercial license would help to accelerate the research of swarm drones significantly.

2.4.2 Existing Simulators

Phadke et. al [9] suggest a suite of simulators, "…assessing their suitability for various specific needs". The article is a concise review of existing platforms, evaluating them based on how they perform, especially in conjunction with each other. Among the evaluated simulation tools are CoppeliaSim, Webots, MATLAB (with the UAV toolbox), and OMNET++. They were each tested based on their key strengths. OMNET++ was used to test FANET topologies and performances, whilst MATLAB and Webots were used to test basic manoeuvres and operations. Coppeliasim and Webots were also capable of creating a high-quality environment with obstacles in it.

Qualnet was used in some literature that was encountered, specifically [12]. It is good for modelling and designing networking architecture, and for swarm drones this can be quite beneficial. However, it does have a cost associated with it, and Keysight Technologies requires users to get a quote for the software [25]. As such it was dismissed for this student research project.

Another simulator found by the supervisor for this project was gym-pybullet-drones (GPD) [26]. In the earlier stages of work, there was a possibility of working with reinforcement learning for drone swarms, and so selecting a simulator with that capability was certainly of some benefit. GPD offers this through the Gymnasium API, and is simple, portable and easily modifiable. It supports multiple quadcopters in a simulated physics environment, complete with realistic collisions and aerodynamic effects. It uses the realistic PyBullet physics engine, and is programmed in Python, a highly writable and user-friendly language. The default simulated quadcopter is a Crazyflie 2.x nano quadcopter, though if desired this could be swapped for any other, provided a suitable URDF. It could also operate without a GUI open, so visualisation of the drones and their positions could be optionally switched off to reduce runtimes.

2.4.3 Selected Simulator

Ultimately, the decision was made to use GPD. This was in part due to the time constraint on the project; having an environment that is already established and is known to work was deemed beneficial as opposed to trying to make a custom setup with multiple programs that other simulators may have required. In order to get started as quickly as possible, having all the necessary tools already at hand was

considered to be of great benefit. Using GPD did open the door to investigating reinforcement learning, however this was not fully investigated during the project.

The Windows installation guide on the GitHub page [27] was followed, which advised downloading the 0.5.2 release. However, this advice seems to be somewhat outdated, as there has been a more recent release since then. This was only realised late into the second semester. As such, some features that may be useful or work better in the newer version are not included or accounted for in this report.

2.5 Understanding gym-pybullet-drones

Some work was invested into understanding how precisely the gym-pybullet-drones (GPD) environment [28] works. The environment is specifically designed for "Single and multi-agent reinforcement learning of quadcopter control," though it does also enable accurate simulation of a simple drone model, perfect for the purposes of this investigation.

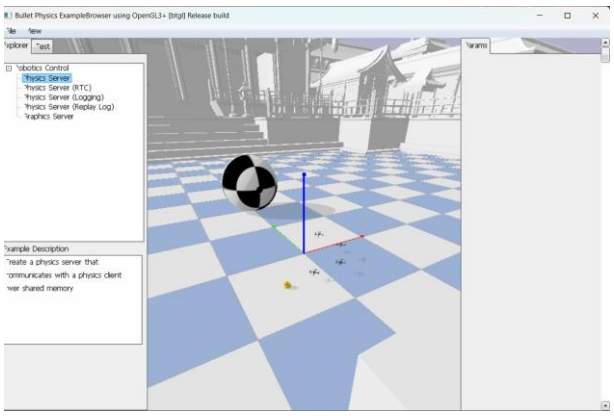

Figure 3: The visualisation of the simulation generated by running fly.py

The PyCharm IDE was selected for running the code, as recommended by the GitHub page. The main advantage of this IDE is that it is simple to use, and this was evident almost immediately. With refactoring, intelligent autocomplete and file formatting tools, it was easy to produce readable code in a short time span.

The example code given directly from the GitHub included various flight patterns, though the primary example is “fly.py”. This generates a simulated environment and models four drones flying in a small circle, as can be seen in Figure 3. Of note is the main output of this preset code: a series of graphs detailing the position, velocity and RPM of the motors on each drone. It is difficult to read but has some interesting information to refer to for working with drones in a generic sense. For interest, Figure 4 below shows an example output of a singular drone flying directly upwards two metres and then maintaining its position. Moving from the left column down, the plots represent the drone’s X, Y then Z position, X, Y and then Z velocity, roll and then pitch of the drone. On the right column moving down again, the plots show the yaw of the drone, angular velocities for roll, pitch and yaw, and then the RPM for each of the four motors on the drone. These graphs are unlikely to be useful for the purposes of this investigation, as there are too many drones in a swarm for any resulting plots to be legible. However, it may prove useful for some applications outside of this project.

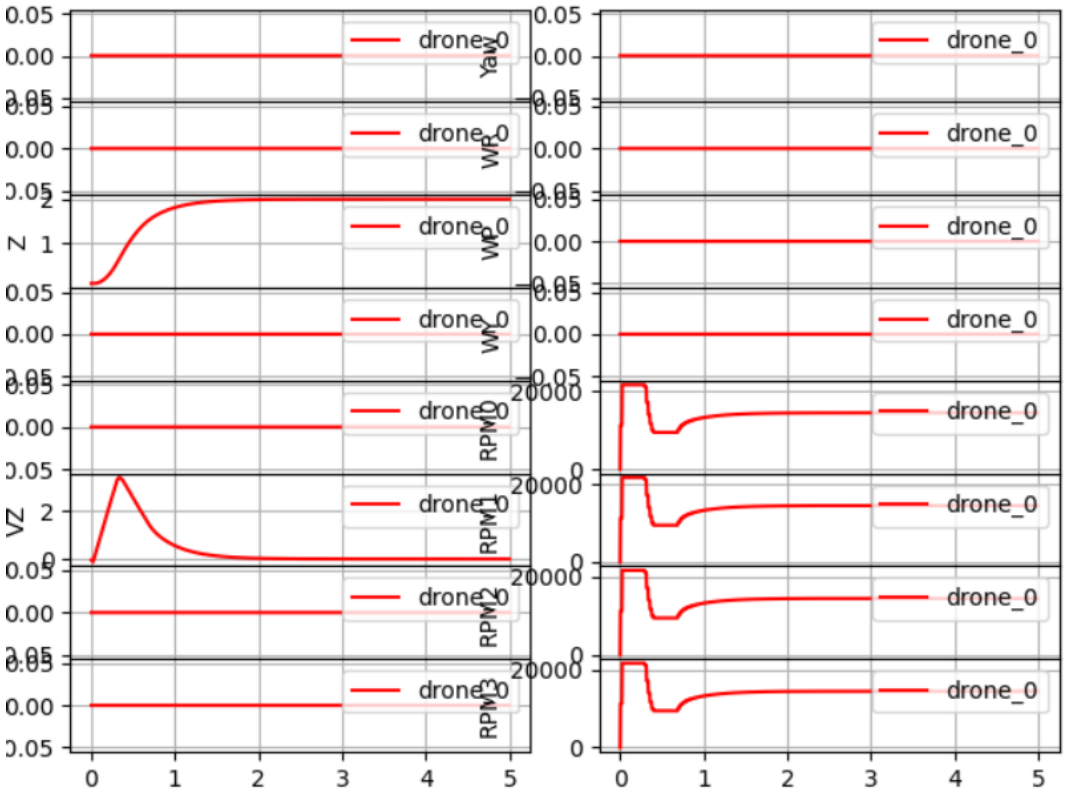

Figure 4: Logger plot outputted from an example flight in the gym-pybullet-drones simulator. This shows a singular drone flying upwards 2 metres and maintaining its position.

GPD could also simulate effects such as downwash, ground effects, and drag. Put simply, these terms refer to how the air around the drone being pushed affects the

drone and others. An efficient demonstration of this exists in “downwash.py”, though the other simulation features at the time of use were still experimental. For this investigation it was deemed these features would not add anything to the results and were ignored by passing an argument when running the program to omit them. Similarly, no use was made of the external force and torque functions from “physics.py”, though they could potentially allow for interesting implementations for weather effects.

A copy of this example was made to experiment with further, named “thesis_demo.py”. By exploring the core mechanics of how the simulator operated, and by testing various methods, a greater understanding was gained of its intricacies. The code uses the argparse library to allow for interpreting of multiple command line arguments, allowing for quick changes to important values. Through the default argument settings, it can be found that the default drone model for simulation is a Bitcraze Crazyflie 2.X, a nano-quadcopter. The closest version available that can be found online is the Bitcraze Crazyflie 2.1+ [29]. which can use Bluetooth or a radio connection. It is assumed here that it is capable of using a ZigBee protocol by virtue of its radio, or can otherwise be modified to do so. There is also an argument to enable or disable obstacles and the plot of simulation results, like that of Figure 4. Both were disabled, as they were deemed to be of no use.

The most interesting of the arguments here is the control frequency, which determines how often a new instruction or waypoint is fed to the drones in the simulation. Effectively, the code worked by creating a large array of target positions, of which the size was determined by the product of the control frequency and the duration of the simulation. These target positions were preset, based on a previously determined rate of speed and starting positions. In practice, this is analogous to a mission planning stage of a drone swarm. Then, using a for-loop, the drones follow this path of waypoints. The drones are then rendered in the visual environment. To keep track of which drones are where at any point in the manoeuvre, there exists an array of Waypoint Counters. This keeps track of the index that each drone has reached in the group’s movements across the target positions array. In the below figure debugger outputs from the waypoint counters array and target positions for the fly.py example are displayed. Here there is a target positions array with 480 XYZ positions, and the waypoint counters array stores the index that each of the four

drones will refer to in their flightpath. As the simulation continues, the waypoint counter values are incremented, and the respective drone moves to the new position.

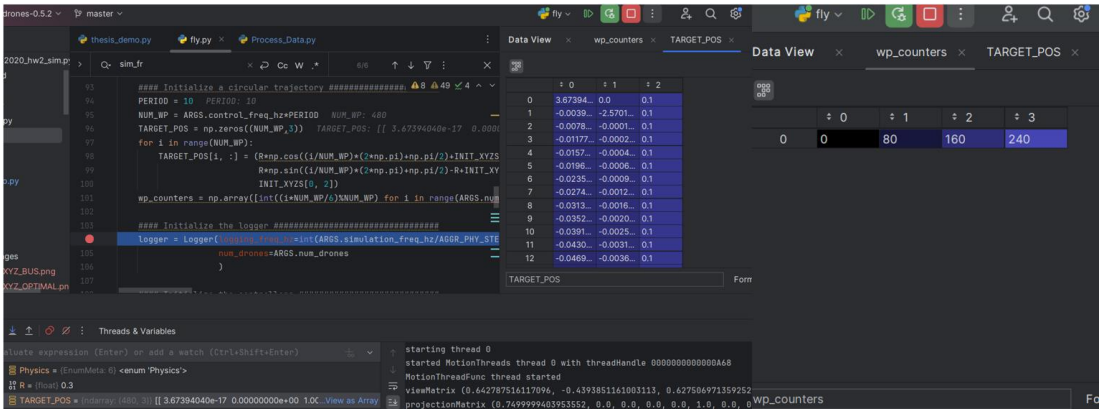

Figure 5: Debugger outputs for the waypoint counters array and target positions array. The waypoint counters point to an XYZ position in the target positions array for each drone in the simulation.

2.6 Legal Constraints Regarding Swarm Drones in Australia

It is worth considering the legal requirements for flying UAVs in Australia, particularly in a swarm configuration. According to the Civil Aviation Safety Authority (CASA) [30], to undertake swarm operations is to have one remote pilot fly multiple drones simultaneously. CASA requires that the operator in question must have a remote pilot license (RePL), hold a remotely piloted aircraft operator's certificate (ReOC) or operate under a business that does, and also to have approval from CASA to fly in this manner. The latter approval form requires applicant details, ReOC holder details, an overview of all operations including start and end dates, drones in use, coordinates, and any potential stakeholders affected by the flights. There may also be a requirement for a BVLOS form (Beyond Visual Line Of Sight), which requires passing an instrument rating exam to ensure the operator knows if any equipment is faulty before flying outside of visual contact. Beyond this, the regular drone laws still apply [31], but exemptions can be sought by contacting CASA. As this investigation was simulated rather than relying on physical hardware, these regulations were not required to be considered, though if any physical research is pursued, they will become important.

3.0 EXPERIMENTAL PROCEDURE

3.1 Thesis Progress

It was decided that the investigation would primarily focus on intra swarm communications. A decentralised setup was assumed; that is to say a remote base station would send a signal to a CH and that drone would transmit the command to the rest of the swarm. This follows the definition of centralised and decentralised topologies as per Phadke et al.'s definition in their report [9], and a similar definition is made in Arnold and Brown's paper [2].

The work in this project began with an initial review of content from the previous semester. Particularly of interest was the initial research into using the IEEE 802.15.4, commonly known as ZigBee, as the communication protocol to be modelled for the investigation. ZigBee was more power efficient than WiFi, lighter than an LTE setup, and had further potential range than a Bluetooth connection [22]. It seemed to be the perfect option for UAV applications, given the constraints and requirements of the platform. From here, the specifications of a ZigBee device [23] could be used to inform any values that were to be used in the simulation.

3.1.1 Main Focus Established

It is around this time that the goal of this project was properly established: to identify several networking topologies, and to investigate and simulate them in order to compare their effectiveness. Topologies in this sense refers to the structuring of paths by which data can be sent to reach nodes that are further away. In swarm drone technology, this is commonly used to disseminate information among members of a swarm, such as directions to move in, speed alterations, or deploying of payloads.

A major challenge to swarm drone technology is that they have very low power requirements. Thus, anything that allows for a reduction in power consumption is beneficial to the operational lifespan of a swarm. If all the drones' batteries last longer, then they can remain deployed for extended durations. It is proposed here that a topology that prioritises higher quality links with lower power requirements will be more effective than one that does not make such considerations.

To facilitate this, an algorithm is required to determine link quality and organise a swarm's structure based on this value. To do so, it must first be understood how signal transmission power and loss operates. As a radio signal is sent out from a transmitter, it immediately begins to lose its strength as it travels through the air and

continues to do so the further away it gets from its source. This is known as Free Space Path Loss, or FSPL. Wherever the receiver is, it will receive the level of power at that distance from the transmitter, and potentially be able to comprehend the message being sent. If the received power is below the receiver sensitivity value, it will not be able to comprehend the message. Other losses may also be incurred at the receiver and transmitter ends, known collectively as system losses. There may also exist other losses during transmission, such as interference in mid-air from other signals, and these may be represented as miscellaneous losses. At the transmitter and receiver, antennae can provide a gain to boost the signal such that its power remains effective at longer ranges. Figure 6 below demonstrates these concepts, for a better understanding of how each of these values interact.

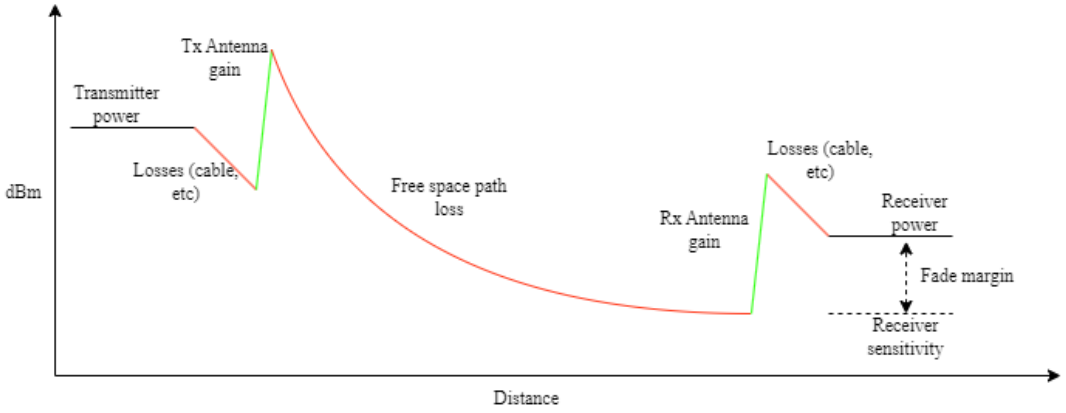

Figure 6: A visual representation of the link budget of a typical transmitter and receiver set up

Using these values, a link budget can be acquired, which represents the calculated signal strength a receiver shall encounter. L_S is arbitrarily decided to be 3dB, as there were too many factors to get a concrete value from any research.

$$P_{RX} = P_{TX} + G_{TX} + G_{RX} - L_{FSPL} - L_s - L_{misc}$$

$$L_{FSPL} = 20\log(d) + 20\log(f) - 147.55$$

Once the received power P_{RX} is calculated, if it has a positive fade margin (i.e. the value for P_{RX} is greater than S_R), the signal to noise ratio (SNR) should be determined for the same link, to verify the quality of the signal. The SNR is a ratio of the signal power as compared to normal power, so 0dB indicates a 1:1 balance of the two. Positive values indicate more signal than noise, negative values indicate more noise than signal. The SNR is calculated as follows:

$$SNR = 20\log\left(\frac{P_{RX}}{P_n}\right)$$

Where P_n represents noise power:

$$P_n = kTB$$

If the SNR is below 20dB, as found in Table 1, the nodes are considered to be disconnected, as their signal quality has fallen below an acceptable level [32]. The Boltzmann constant k and noise temperature T are 1.38×10^{-23} J/K and 290 K respectively [33].

Finally, once both of these have been calculated, and are above their respective thresholds, a delay value can be calculated to see how long it takes for the message from one drone to reach another. It is the sum total of the time it takes for light to reach the next drone, plus the time it takes for the drone to process the information, and then transmit every bit in the message.

$$Delay = \frac{d}{c} + t_p + t_t$$

$$t_t = \frac{M(bits)}{r_d\ (bps)}$$

Research was conducted to obtain values for these equations, and an initial pass through of the mathematics was considered satisfactory for the requirements. These were derived from hardware specifications, 802.15.4 protocols, and research papers on drone technology. A table containing all variables is listed below, along with any available values and their sources.

Table 1: Variables in the algorithm used, and their values along with their sources

Var.	Name	Description	Value	Ref.
B	Bandwidth (MHz)	The potential capacity of data transmission. Each channel is 2MHz with 5MHz separation.	2	[34]
r	Data rate (Kbps)	The number of bits capable of being transmitted per second.	250	[34] [23]
d	Distance (metres)	Distance between communicating nodes. Value is specification rating for indoor use.	60	[23]
f	Frequency (MHz)	Frequency of 802.15.4 protocol	2450	[35]
M	Message Size (bytes)	Number of bytes in sent message.	263	[36]
t_p	Processing and queuing delay (ms)	Time that the onboard flight computer in each drone is likely to take while processing instructions.	10-30	[37]
G_{RX} G_{TX}	Transmitter / Receiver Gain (dBi)	Antenna gain from off-the-shelf component	2.8	[38]
P_{RX}	Received power	Determined in the equations above.	-	
S_R	Receiver sensitivity (dBm)	Minimum signal power that the receiver can detect and decode successfully.	-100	[23]
P_{TX}	Transmit power (mW)	Power emitted at transmitter, before other losses accounted for.	3.1	[23]

The Friis equation was also encountered during research, though it was realised that this is a simplified alternative to calculating the link budget. The values it provides are expected to be used to derive a ballpark estimate for what the actual link budget may entail, and as such it is disregarded for its lower accuracy.

Later on, the system loss, miscellaneous loss, processing delays and noise were modified through a added white gaussian noise (AWGN) instead of fixed values. This allowed for a little more diversity in the results.

3.1.2 Graph Representation of a Swarm

From this point, it was clear that a data structure of some description would be required to indicate how the drones in a large swarm would all be connected to each other. It was decided that a node graph data structure would work well. This allows for multiple links, and representation of each node as an integer value, the same way as is done in the GPD code. Drones are designated in the order they are created, starting with 0 and counting upwards. This has the advantage of allowing for simple array indexing when accessing data for the drones, and therefore faster programming across the board. Once the graph data structure was working properly, functionality such as using the link budget and SNR equations above were added, along with some basic helper functions. Then it was ready to begin experimenting with some different topologies.

3.1.3 Topologies Used

Three different topologies were used in the investigation. They organise eight drones' communications into a format best suited for disseminating information among the swarm. Figure 7 shows which drone is assigned to each ID - zero through to seven. The figures that follow show how each topology organises them for communications. For the sake of simplicity, Drone 0 is always the CH, and all commands for the other drones will come from it.

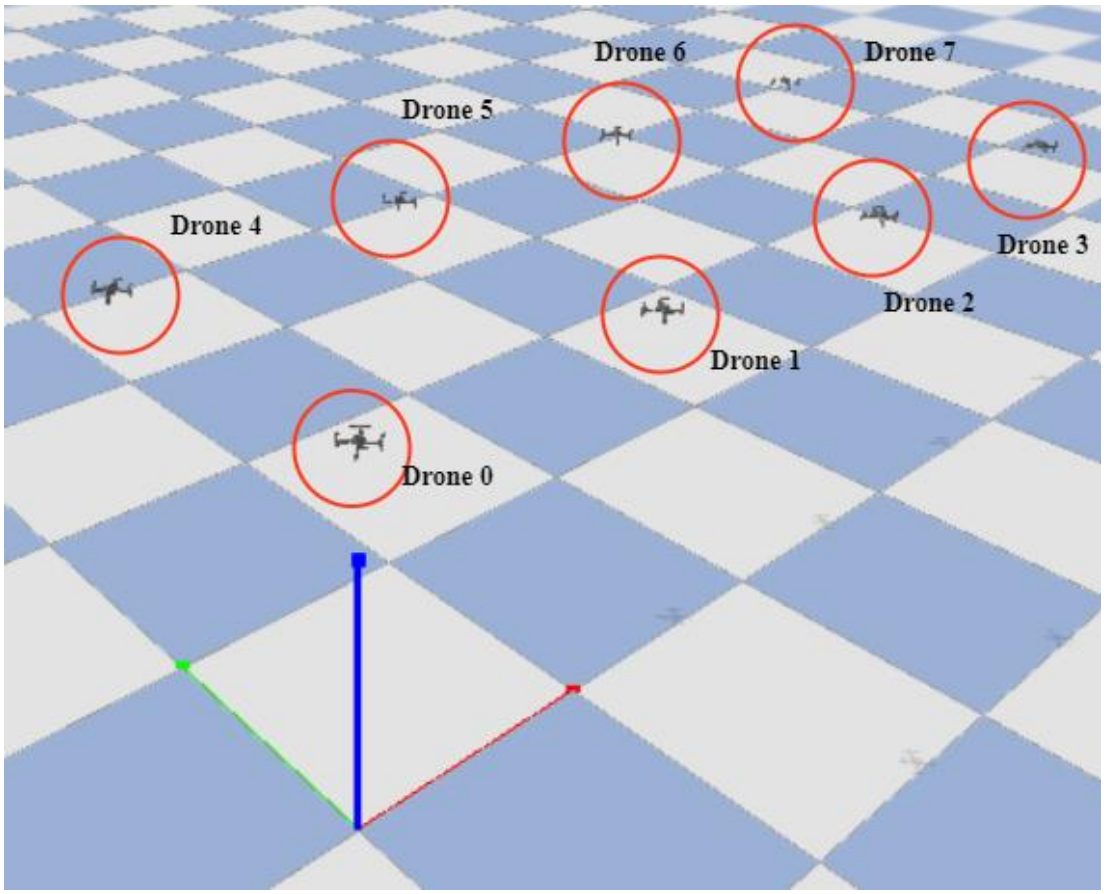

Figure 7: Drone swarm with labelled IDs

The reason why this grid pattern was used is because there are existing works that likewise use a starting grid formation that would be ideal to compare to [39] [12]. Drones in this pattern are easy to set up, and in a real-life scenario, may similarly be deployed in this manner, as opposed to scattered randomly or in a circle, for example.

Additionally, a single CH is selected here to keep things simple, but also under the advice of a similar report. Marek et al. claim that "... controlling a swarm of drones with a single leader is easier and results in fewer errors in maintaining the structure. The structure of a swarm in which each drone only looks at the drone in front of it requires a much better communication system, with significantly lower transmission delays" [39]. In the interest of simplicity and efficiency then, a single CH is used. This claim could be verified in any future works, however.

The first of these topologies is a Bus Topology, in which each drone is linked in sequence to the next one as seen in Figure 8 below.

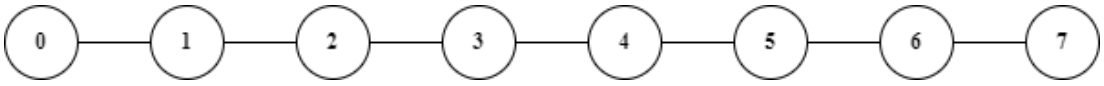

Figure 8: Bus Topology

This is a rather simple implementation of a drone topology, and immediately presents an issue. If any link is severed, all of the following nodes become disconnected. The main reason for its inclusion here is that it will serve to demonstrate the "worst case" for comparison. It can help to show the effect of a large set of network hops on a drone swarm, or how remaining heedless to a drone's position in the swarm when establishing a topology is relevant to its performance.

The second topology is the Tree topology, where the CH links to two other nodes, and every other node chooses two other unselected nodes.

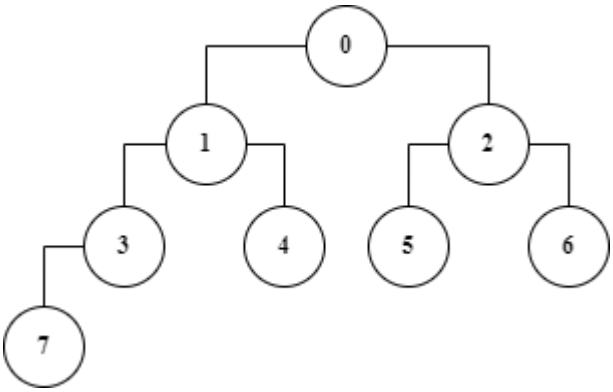

Figure 9: Tree Topology

This is a similarly simple method of forming a topology, based on a binary search tree. The immediately noticeable difference here is that there are shorter paths to reach further away nodes. This makes it slightly more robust for node failure, as it will not immediately disconnect all following links. It is still operating heedless of drone positions but is likely to maintain contact with more of its links.

The final topology is generated automatically, based on the potential for quality of communications with each drone in the swarm. Using the equations in chapter 3.1.1, each drone starting with the CH finds the best two nodes it can link to that it hasn't already connected with and connects to them. The method for determining this quality is by checking the distance between each drone and finding the P_{RX} at each

drone, and then selecting the highest two values to connect to. For the rest of the report, this proposed solution shall be referred to as the “Optimal Topology”, though it may be found in the later results chapter of this report that this is not the case.

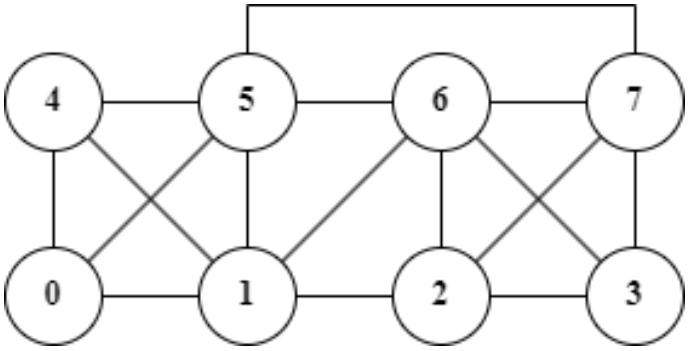

Figure 10: Optimal Topology

Drone	Connects to:		Drone	Connects to:	
0	1	4	4	1	5
1	2	5	5	0	6
2	3	6	6	1	7
3	6	7	7	2	5

Figure 11: Connections that each drone made in the Optimal Topology in Figure 10

As can be seen, the received power at each node depends on the distance to that node most significantly, at least for a homogenous swarm. The benefit here is that by selecting drones based on the calculation of their received power, it is more likely to break ties in distance due to any onboard system losses. Additionally, the linked drones will be somewhat closer together, ensuring that if drones start drifting apart, there is a longer duration for which they are connected, and course corrections can be made. There is certainly some room for improvement with this implementation, which will be discussed in the future work chapter of this report.

3.1.4 Incorporating Delays into GPD

Attention was then given to how to implement this with GPD. The graph data structure was easy to implement, for the aforementioned reason that the drones are

numbered based on array indices. The delay functions from 3.1.1 are used to interpret how much delay each drone should have at any given point.

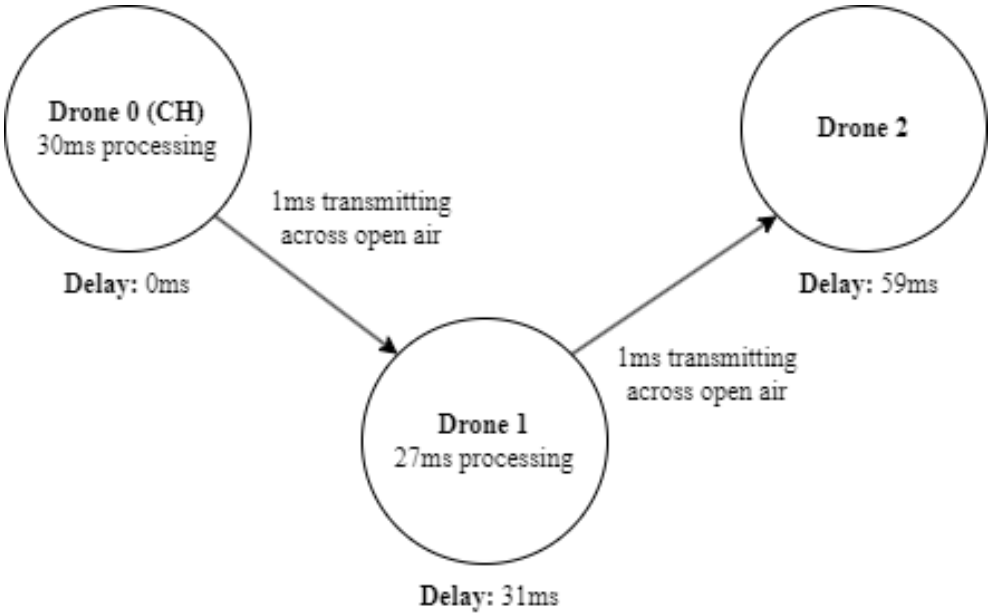

Figure 12: Example of how delay increases across the swarm

Figure 12 shows an example of how delays add up across the swarm. Each drone takes a certain amount of time to process, prepare and send out a message, and this accounts for the bulk of delays in messaging across the swarm. Transmitting across air takes very little time, as the short distances are crossed at the speed of light. In fact the above example is an exaggeration of the scale that drone swarms work at.

A process was created for GPD in order to determine the delays across the swarm. This involves using a breadth-first search algorithm to determine the shortest paths to each drone from the CH. Once this has been done, the previously determined delay equations can be applied to each link, and a sum total of delay to each drone is identified. As Figure 12 shows, this delay compounds with every previous link, and steps up in increments of about 30ms every time. The delay is applied in code by calculating the appropriate adjustment to the values in the waypoint counters array. To do this, each drone’s delay is translated into a corresponding number of index positions by which its waypoint counter is shifted backwards, effectively setting the drone's progress along the target positions array behind that of any drones in front of it in the topology.

3.1.5 Simulation Parameters

Testing of these drone topologies was the next stage in the plan. The desired work for the GPD environment was to perform multiple runs of the same conditions, save results such as position over the flight duration and final delays, and then change something and repeat the process. The argparser library was incredibly useful here, allowing for minor adjustments when making multiple runs of the program. The PyCharm IDE lends itself well to working with this system, using its Run Configuration feature. It is simple to set up and run different sets of parameters this way, which made the results gathering phase of the project reasonably effortless.

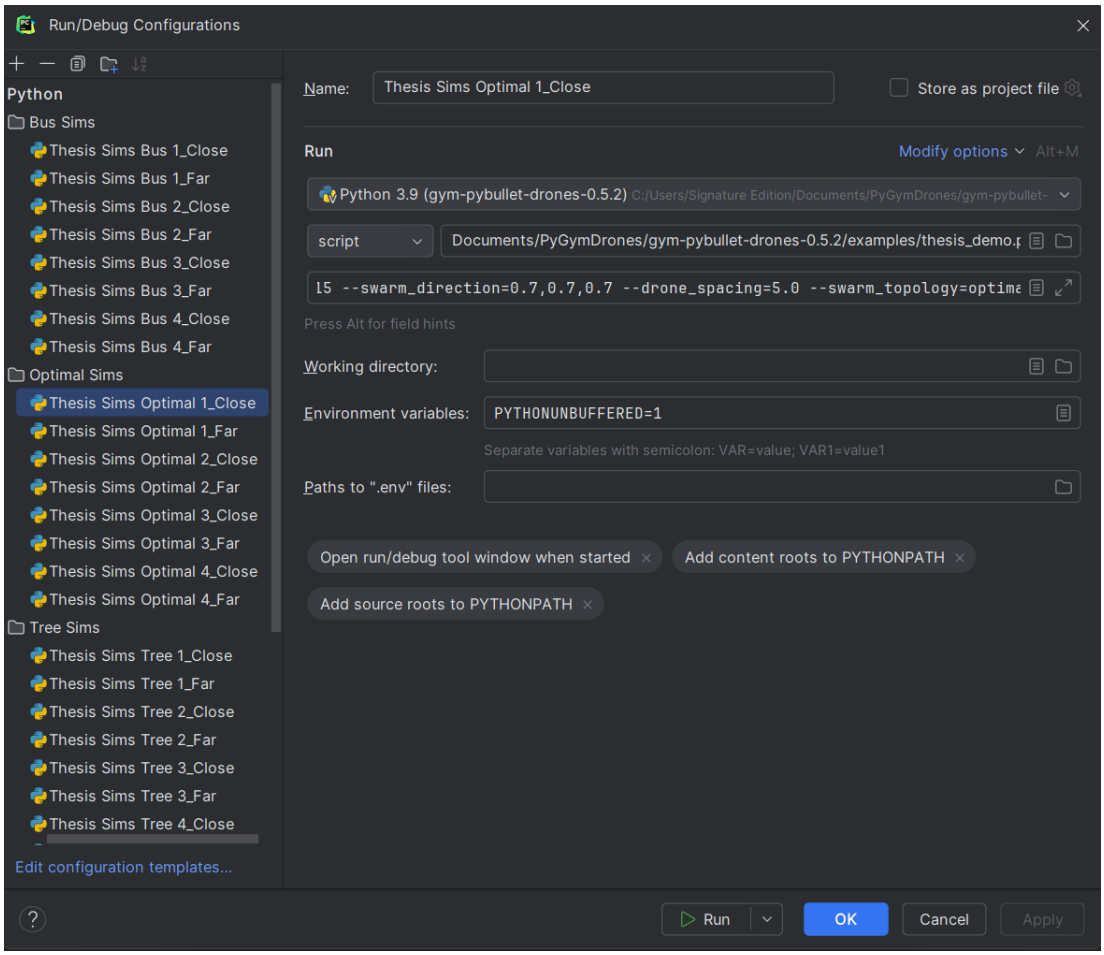

Figure 13: The Run Configurations window in PyCharm IDE

Figure 13 displays the Run Configurations window at end of the project, in which there were eight different simulation scenarios for each topology, differing in the speed and spread of drones. Each argument variable could be assigned from here and ran with no issues. Table 2 below shows the typical values for each of these arguments in these simulation setups.

Table 2: List of all arguments used and all of their values in the simulations

Argument	**Values**	**Notes**
Simulation iterations	25	Number of times this program is repeated.
Number of drones	8	8 drones are sufficient to show a deeper topology in most cases.
Duration	10	Duration of each simulation in seconds
Swarm Direction	[0.7, 0.7, 0.7]	The direction that the drones will follow, with scaled values.
Movement rate*	0.015/ 0.02/ 0.025/ 0.03	Different set of simulations for every increment in speed, to diversify data.
Drone spacing*	5.0/ 20.0	How far apart each drone is from each other at the start of the simulation. Determines the X and Y position for each drone
Starting height	0.3	The starting Z position of all drones in the swarm in metres
Swarm head	0	Always Drone 0 for consistency
System Losses	$\mu = 3$ $\sigma = 0.7$	For representing the losses at transmission and receiving of information in a communications system, as well as any unprecedented. Calculated as a gaussian distribution to introduce uncertainty.
Miscellaneous Losses	$\mu = 1$ $\sigma = 0.1$	
Processing delays	$\mu = 0.025$ $\sigma = 0.001$	Delays in processing data with an onboard flight computer for each drone.

Twenty-five iterations means that each drone spacing tested had 100 simulations ran under them, with varying speeds. This would enable more conclusive results regarding the relationship between topology and drone spacing.

3.1.6 Assumptions Made

This section is dedicated to any assumptions made for the testing of these topologies. The first of these assumptions are that losses are calculated based on a gaussian distribution, rather than on found values. Sources of these losses do not require explanation: it does not matter where they come from.

Drones that become completely disconnected from the swarm, for example by flying far enough away that the SNR threshold and/or the receiver sensitivity is exceeded, are deemed lost. The drones stop their movement exactly where they are.

Ground effects, downwash and other aerodynamic effects in quadcopter flight are assumed to have little effect, in order to keep the simulation simple.

A problem that requires some additional discussion is one encountered late into the project. At times, drones in the swarm may appear to fly normally, then suddenly lose control, oscillating in their position wildly before crashing into the ground. This was particularly prominent in drones that were far removed from the CH. This issue is disregarded, as in practice FANETs with more hops to reach a particular drone will suffer more. The quick fix solution to this is the same as it is in reality: slow down the swarm's movement. This is part of a universal issue for swarm drones and their scalability. While crudely represented here, a drone that is delayed as compared to the others will be significantly impeded in its performance, both in simulation and in reality. Marek et al. encountered a similar effect in their research, in which drones with long links could not travel at quicker speeds due to unsafe minimum distance between drones [39].

3.1.7 Results Gathering

After every simulation iteration, CSV files were generated with the positions, percentage errors on these positions, and any delays. A python script "Process_Data.py" was used to collect these values and convert them into matplotlib figures for visual representation of the results.

3.2 Materials and Equipment

In order to replicate the works in this report, the following are required:

- PyCharm IDE [40]
- gym-pybullet-drones (GPD) [28]
- Main Libraries used outside of those provided by GPD:
    - Argparser
    - Matplotlib
    - Numpy
    - Pandas
    - Glob and OS

If it is important, the computer used to run this code was an Acer Aspire 5, with an AMD Ryzen 5 3500U processor and with Radeon Vega Mobile Gfx.

3.3 Method

PyCharm lends itself well to using argparser and running multiple variations of a script in a swift and elegant manner. Using the Run/Debug configurations, it was easy to predefine large sets of arguments and set everything off in one go. This made gathering results simple.

To do this, once PyCharm is open, ensure that the “thesis_demo.py” code is in the examples directory under gym-pybullet-drones. Then, in the top right corner, there should be a three-dots icon. Click on that, then under configurations click ‘Edit’. From here, a user can either select a script to run or establish a new configuration.

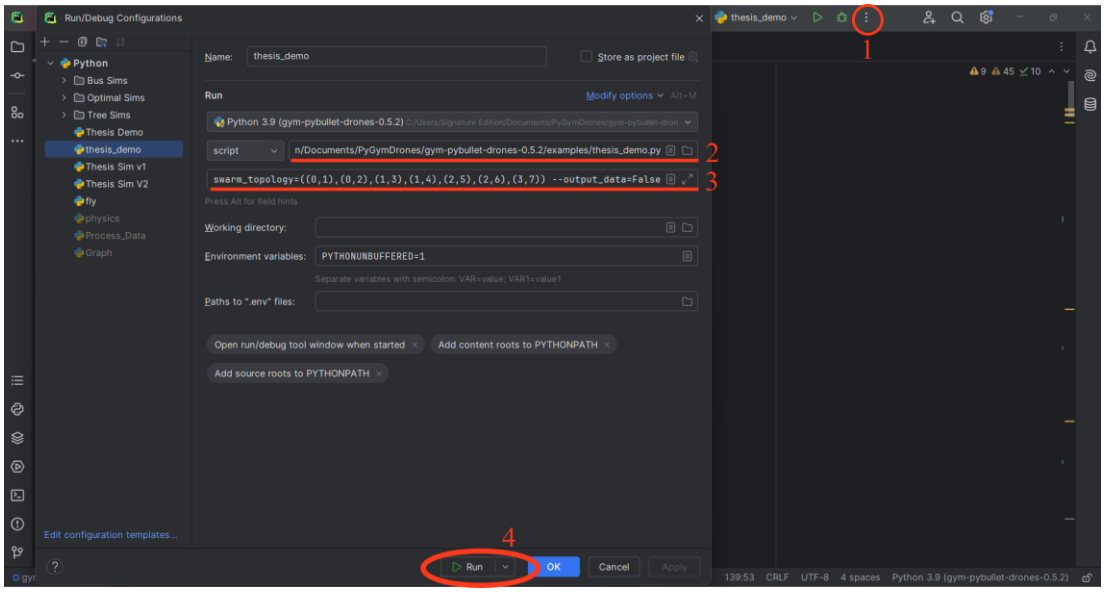

Figure 14: Step-by-step guide for reproducing results

Next, select the script “thesis_demo.py” in the field indicated with ‘2’ in Figure 14, and fill in the parameters field below it.

The parameters used in for the first Tree Topology trials are written as follows:

--simulation_title="TREE_Diagonal_flight"

--simulation_iterations=25

--gui=False

--movement_rate=0.015

--swarm_direction=0.7,0.7,0.7

--drone_spacing=5.0

--swarm_topology=(0,1),(0,2),(1,3),(1,4),(2,5),(2,6),(3,7)

To change between the various trials, change –movement_rate and –drone_spacing to values given in Table 2. To use the Optimal Topology, change the last argument to “–swarm_topology=optimal”.

Once this is done, “Process_Data.py” is ran to collect all data and visually represent them using matplotlib. No additional parameter changes are required, and as long as the simulations follow the right naming and file conventions, all data should be processed into CSV files and then collated and turned into png image plots, which are stored under “files/outputs/AAA_IMAGES”.

4.0 RESULTS AND DISCUSSIONS

4.1 Results Gathered

Three different topologies were tested for results collection. These are the ones highlighted in brief in Chapter 3.1.3; Bus, Tree and the Optimal topology created in the course of this project. For each of these topologies, 25 simulations were run, using the set of arguments in Table 2. Three sets of data were collected every second during each run of the simulation. These data sets were for the X, Y and Z positions, the percentage error of these positions, and the measured delay from the CH. Each topology went through eight different trials, using four different speeds and two different starting spaces between drones. The 25 results of each trial were all averaged together, to provide condensed graphs in the following sections. This gives a range of conditions by which the topologies can be evaluated based on its own merits. As shall be seen in the ensuing sections, each topology will display eight tables, displaying average results from these trials. Rows down the way increase in speed, and columns to the right increase in spacing between drones, referred to as range in the plot headings.

4.1.1 XYZ Positions

As can be seen in Figure 15, Figure 16 and Figure 17, each drone is programmed to fly in a simple, diagonal line upwards. Each colour represents a different drone in the swarm flying through 3D space, and the results here represent an average of their positions over the simulation iterations.

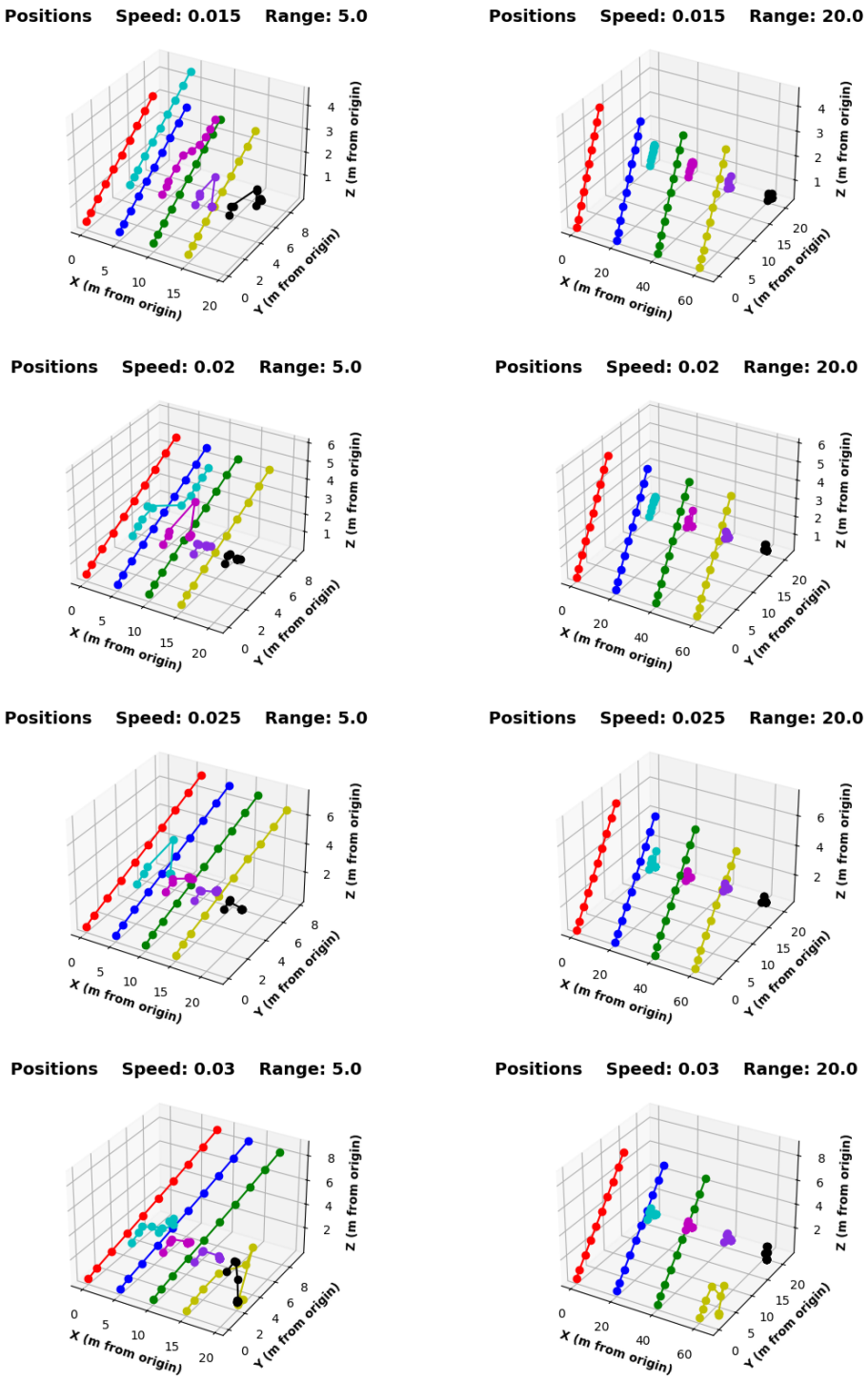

Figure 15: Average XYZ Positions of a Drone Swarm in a Bus Topology.

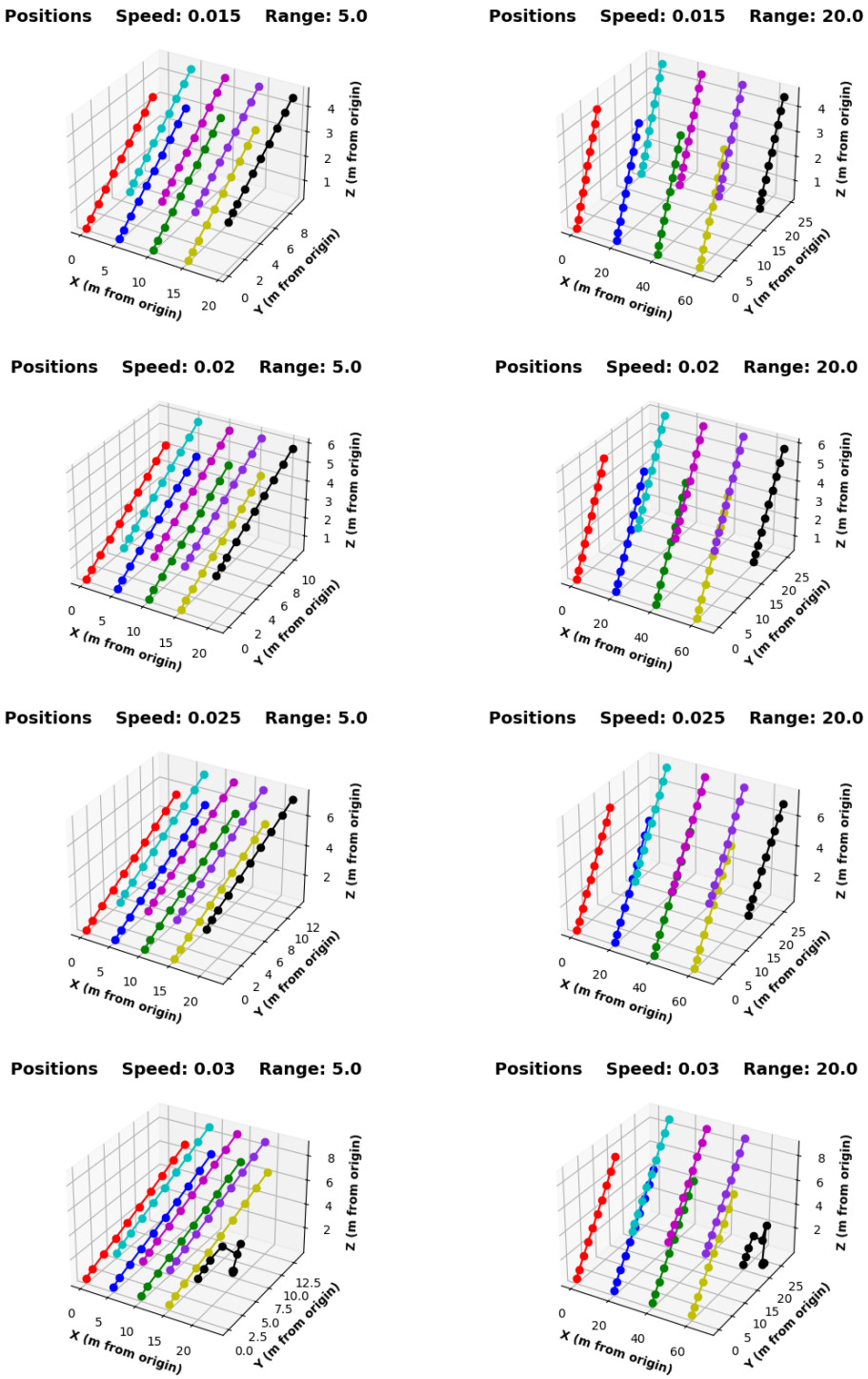

Figure 16: Average XYZ Positions of a Drone Swarm in a Tree Topology.

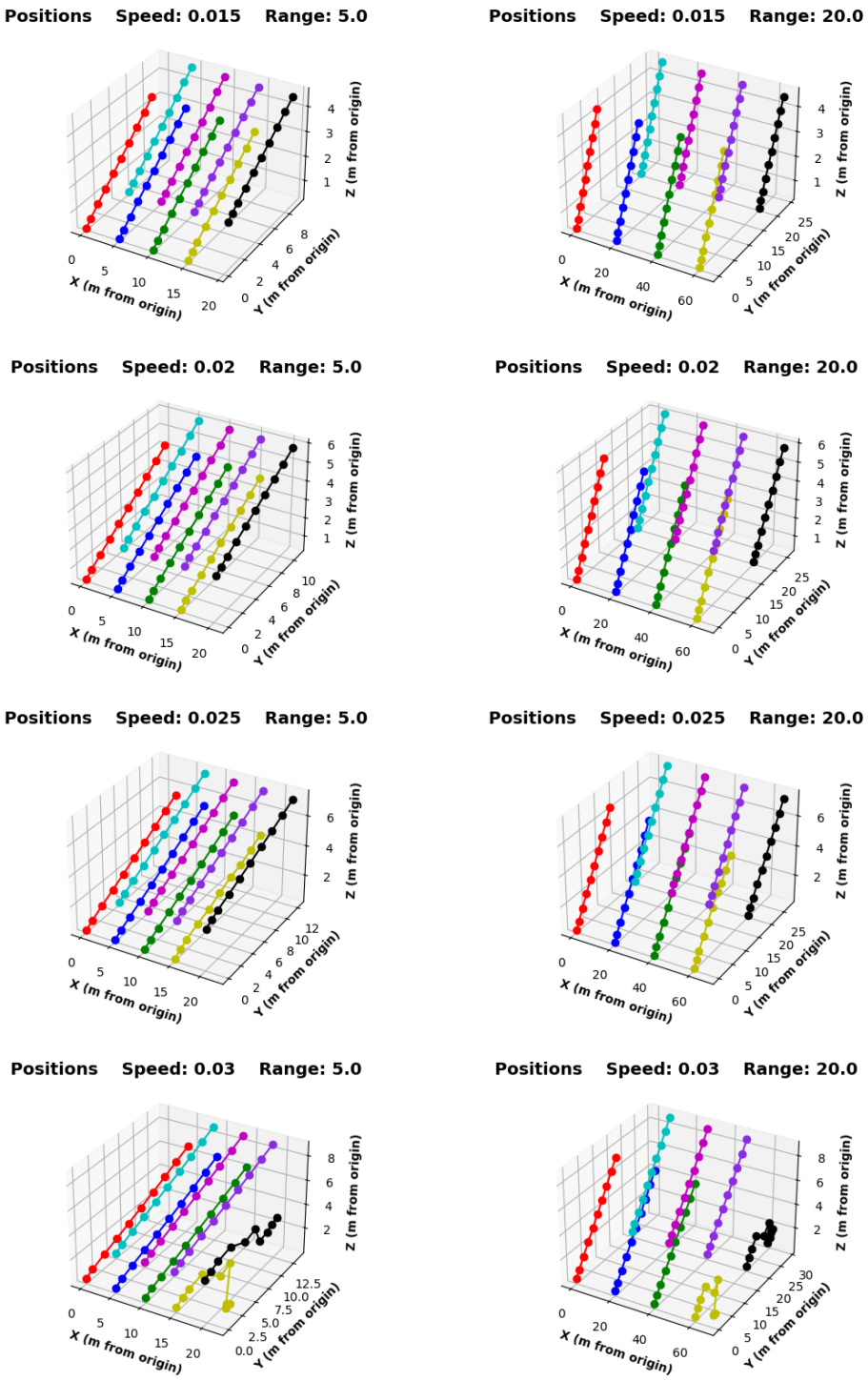

Figure 17: Average XYZ Positions of a Drone Swarm in an Optimal Topology.

4.1.2 Position Error

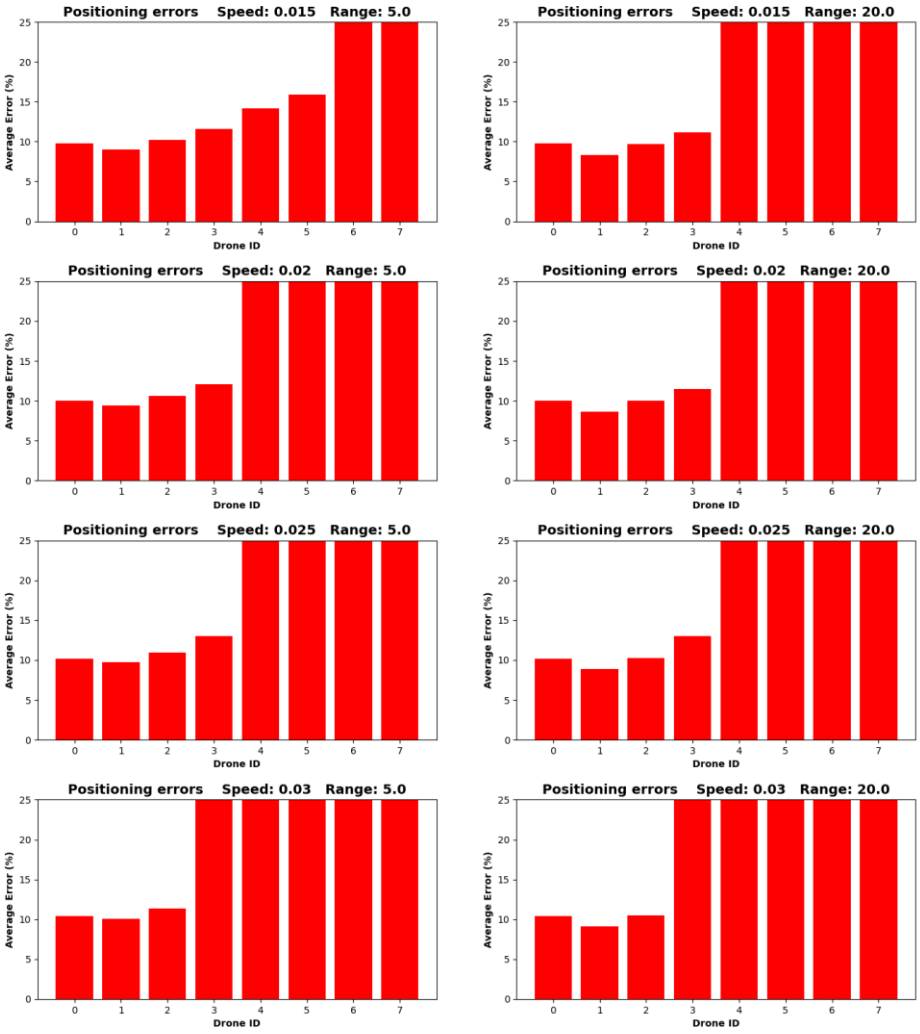

Figure 18: Position error as a percentage on a Bus Topology

The positional error as a percentage is determined by the following simple formula:

$$E = 100 \times \frac{|Position - (Estimated\ Position)|}{Estimated\ Position}$$

This returns a percentage value by which the accuracy of the drone's flight according to their planned trajectory can be judged. As the cluster head should not have any delay, drone 0 can be used as a reference point for the level of percentage that is acceptable for the swarm.

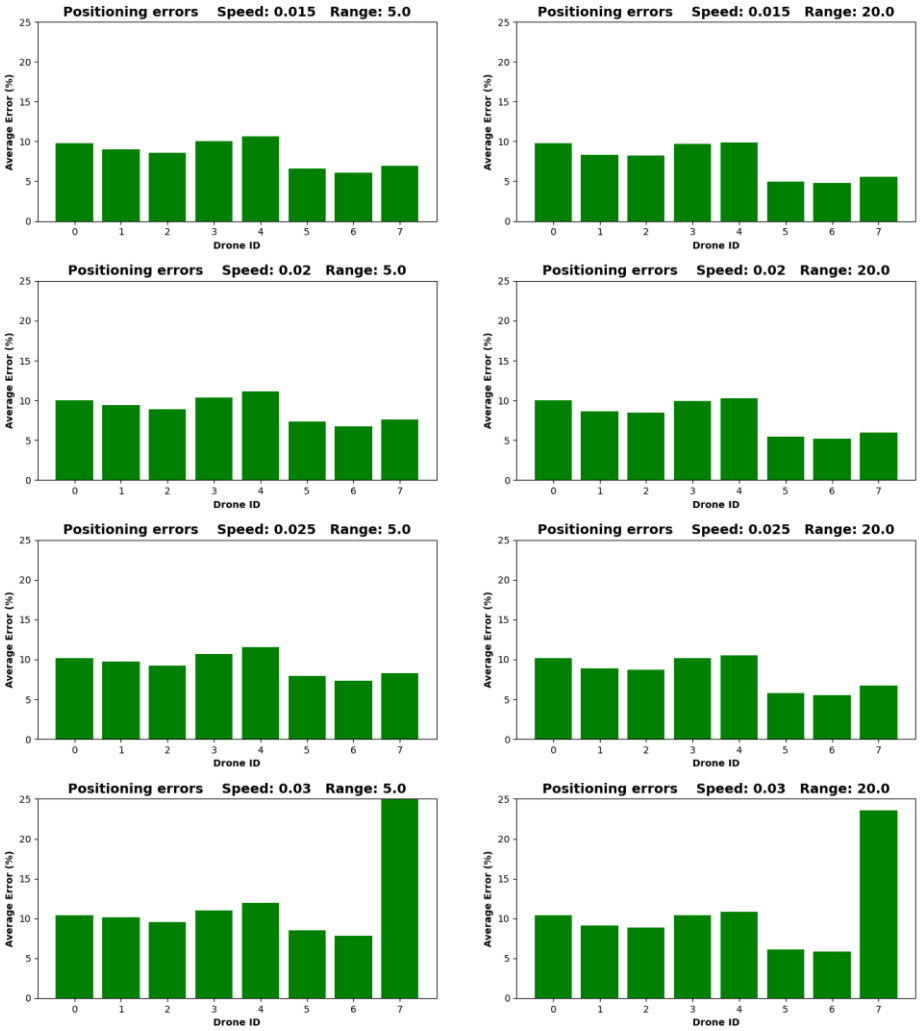

Figure 19: Position error as a percentage on a Tree Topology

The bar charts in the figures in this chapter have been restricted to only show up to 25% error: higher than this, it can be determined that the respective drone will not be flying in an acceptable manner. Whether this is directly due to delays or due to crashing, is of little consequence.

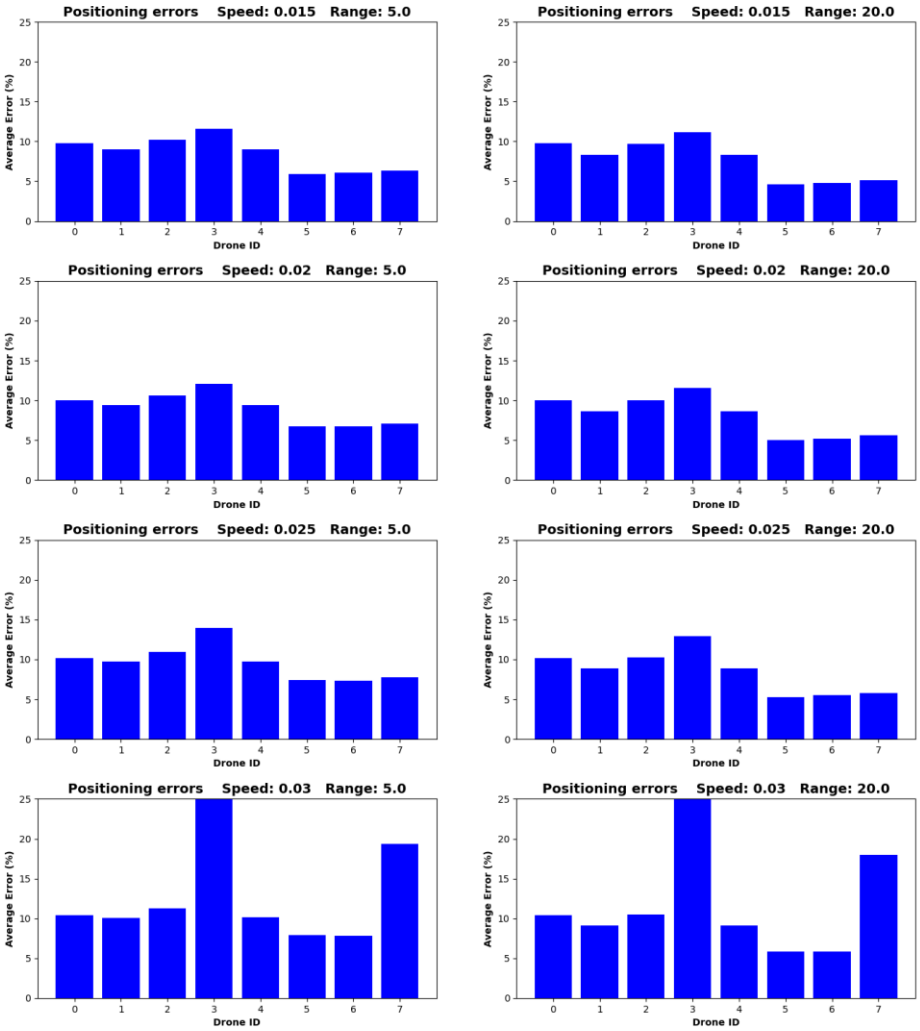

Figure 20: Position error as a percentage on an Optimal Topology

4.1.3 Delays

The drones' delays are shown again in the form of a bar chart. The highest delay is marked with a label indicating its value. The delays are taken from the final set of values stored in the delay register, meaning that these values are based on the end position of the drone swarm. Once again, results are averaged across all the simulation iterations, to provide more generic results. For obvious reasons, the CH drone 0 will have no delay within the swarm: It is assumed to have received instruction already and will know where it needs to be. The focus is on disseminating this information to the other drones.

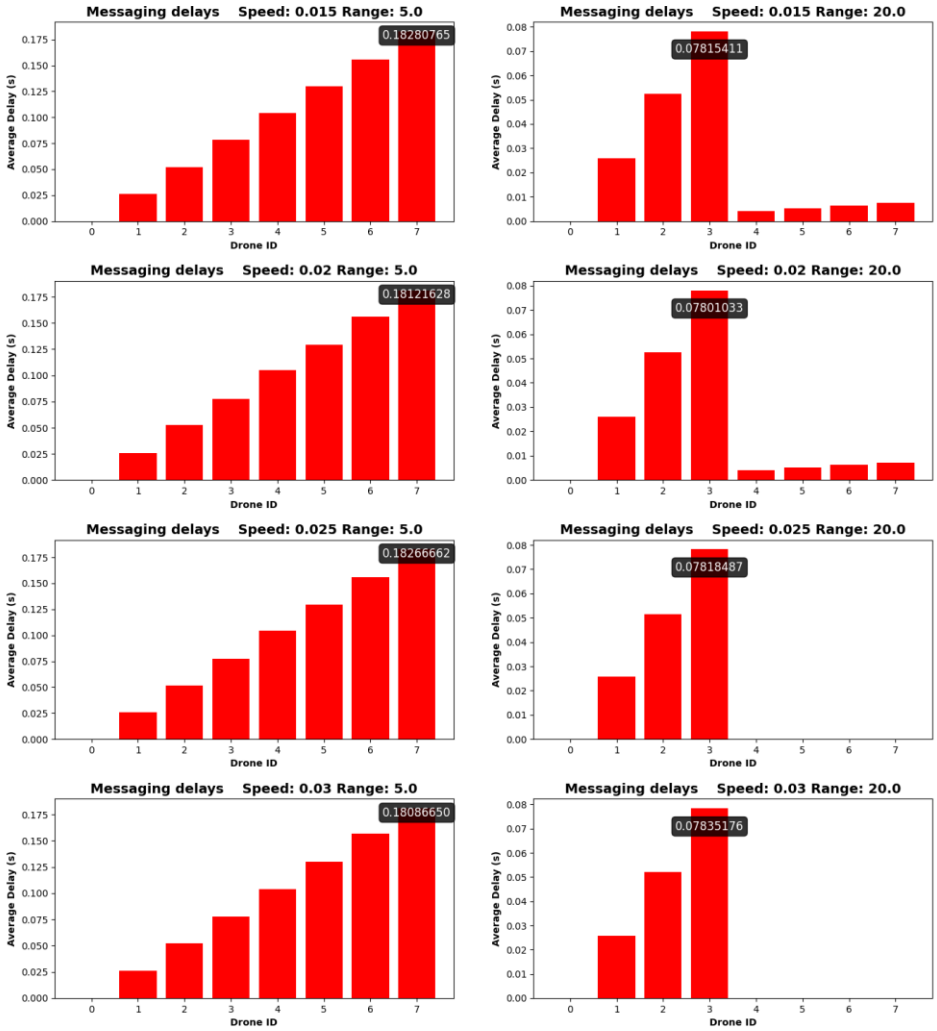

Figure 21: Messaging delays incurred for each drone in a Bus Topology

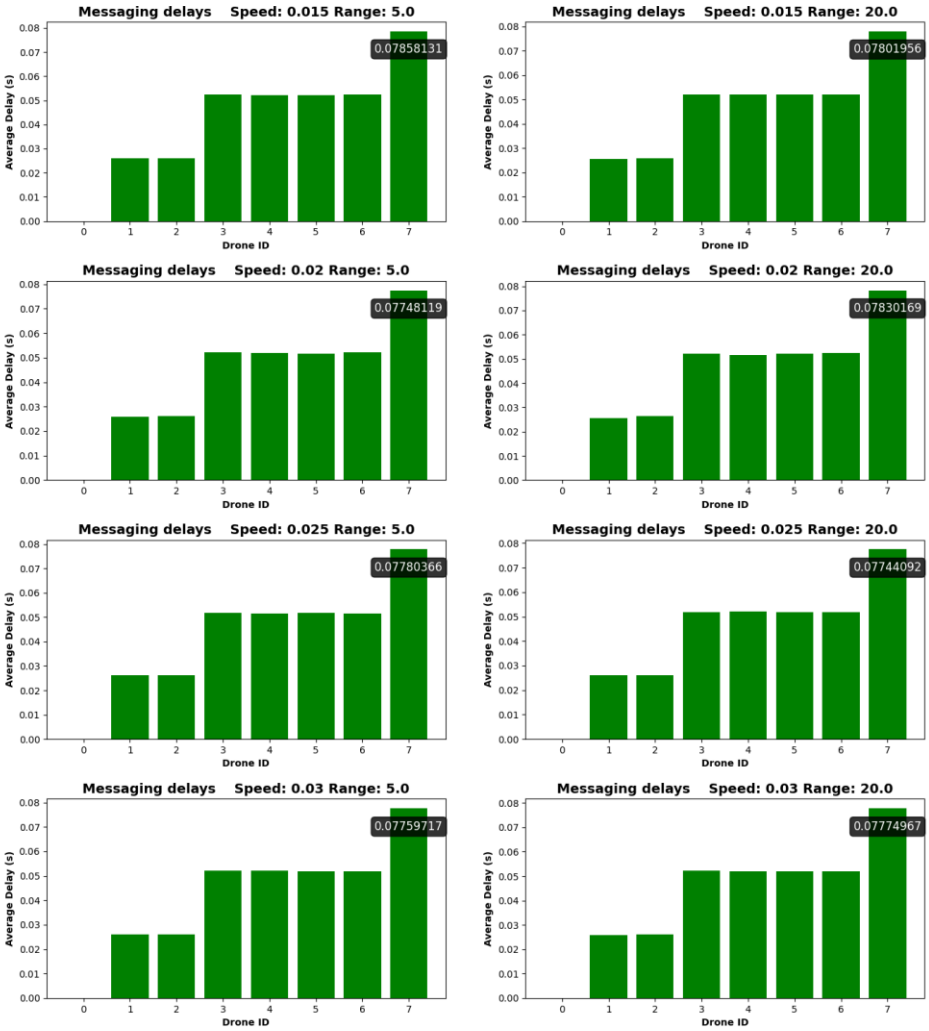

Figure 22: Messaging delays incurred for each drone in a Tree Topology

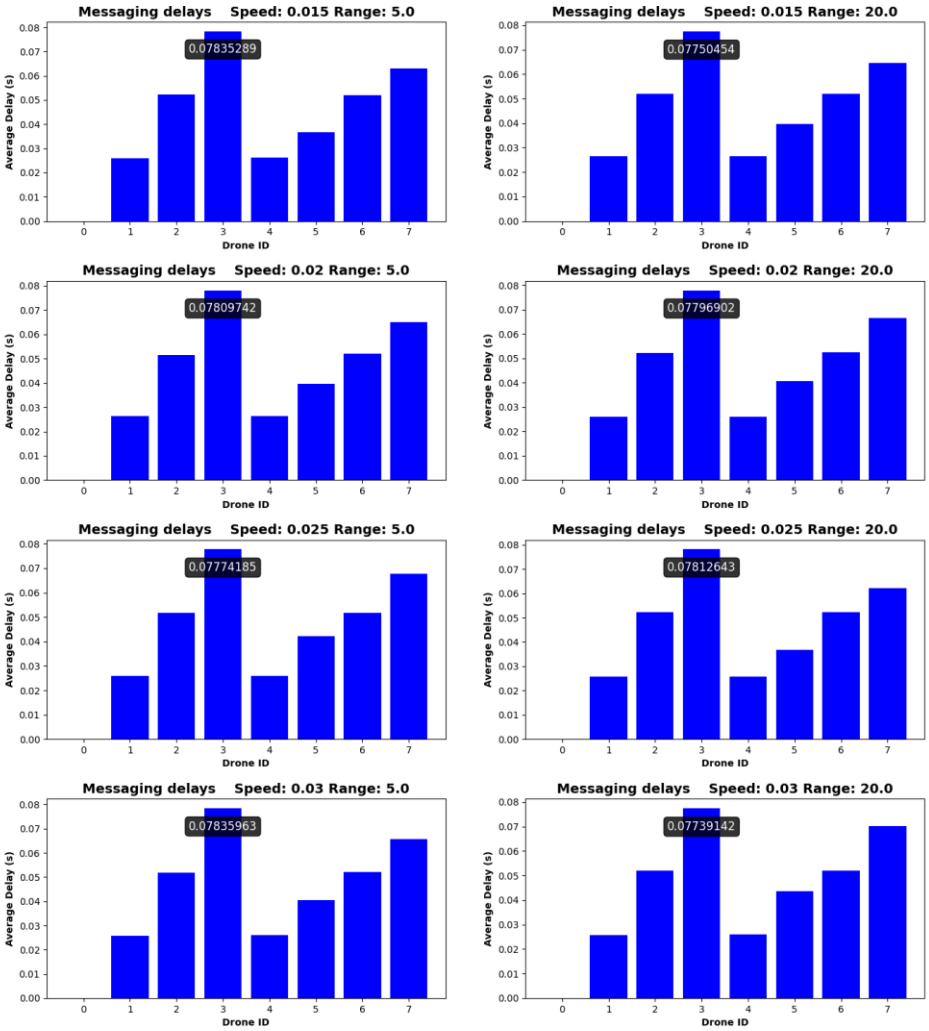

Figure 23: Messaging delays incurred for each drone in the proposed Optimal Topology

4.2 Discussion Of Results

4.2.1 XYZ Positions

Figure 15 shows the Bus Topology's average positions across simulations. Some drones are seen crashing into the floor in these graphs, which is due to the large amounts of delay incurred for these swarm nodes the further away they get from the CH. Additionally as the swarm's speed is increased, its performance suffers significantly. More drones fall away as the speed increases, establishing that there is a relationship between following instructions and messaging delays.

Compare this to the Tree Topology in Figure 16, which is much more consistent due to the lower number of hops required to reach each node. Drones only appear to crash at higher speeds and are otherwise consistent in their flights.

When the Optimal Topology in Figure 17 is compared with the Tree Topology of Figure 16, it would appear to have shortcomings in that two of the drones failed at the final speed level, in comparison to the sole failure from the Tree Topology. What is not considered with this cursory assumption is that both of these occur after the third hop in the swarm's communications. In fact, when regarding Figure 15 as well, it is noticed that the fourth drone in the topology appears to always fail at the speed 0.03 in this simulation. From this it is gathered that speed is an influencing factor on the drone's particular failure, however the better question would be whether the topology has made the most efficient structure for this particular setup. For the given example with eight drones and a diagonally upwards motion, the proposed topology has placed two drones at a long distance from the cluster head and is therefore more susceptible to higher speeds due to greater delays.

4.2.2 Position Error

The chart in Figure 18 shows the same trend for the Bus Topology as for the previous section. Slow movements and close proximity to the other drones will ensure that no drones disconnect or crash.

In Figure 19 and Figure 20, the positional error tells the same story as that of the previous section. Of additional interest here is that some of the drones appear to fly a lot tighter than expected, the further from the CH they get. This is a strange phenomenon, as it might be expected for them to become looser in formation due to the delays incurred. It is believed that the positioning data has been improperly collected, and as such the error percentages may be off.

In Figure 20, the percentage error for drone 7 is lower than expected. Specifically, when the swarm is flying at high speeds and the instructions have reached the third drone in the swarm, the other topologies indicated that this drone will regularly crash out. However, this result shows some more stability. It perhaps still does crash but has a chance to not do so as well. This could be due to the more robust connections providing some amount of reduced delay, and therefore less issues with positioning. More simulation iterations at this speed would be needed to determine if this is not just a random anomaly.

4.2.3 Delays

Each of the drone delays shown in Figure 21 through to Figure 23 appear to form in discrete steps of about 25-30 milliseconds. This value matches with the processing delay, implying here that the time to transmit and time travelling across space are both marginally impactful in this application. Neither the amount of space between drones nor the speed at which they are travelling affect the delay, and when considered objectively, these are minimal factors when considered in relation to the speed of light. Therefore, it can be deduced that the primary bottleneck for delays in swarms is the number of hops taken to reach a particular drone, rather than any physical distance between them.

The Bus topology is shown to regularly disconnect across drones 3 and 4 when using 20 metre drones spacing (right column of Figure 21). If the drone positioning in Figure 7 is considered, the distance between drone 3 and 4 is a little over 60 metres, which is higher than the rated indoor range on the XBee specification [23]. The drones will therefore be more likely to disconnect, due to lowered SNR values. The longest delays for the Bus topology exist at drone 7, which is the furthest away from the CH in the topology. As expected, these delays start to become impractically long, and the positioning error tends to suffer for it.

The Tree and Optimal Topologies are both similar in regard to the delay results. For both, drone 7 is three hops away from the head, and drone 3 is as well in the Optimal Topology. They should have roughly the same amount of delay as each other. However, Figure 23 shows otherwise. The Optimal Topology's drone 7 shows a shorter delay than would otherwise be anticipated. It appears to be consistently 5-10ms quicker than drone 3, which should take approximately the same length of time for a message to reach it.

Drones that have evidently crashed based on the XYZ positions graphs in chapter 4.1.1 and 4.1.2 continue to transmit any data, hence they still contribute to the swarm's structure and delays. While it is probable that a drone that survives will continue to transmit after crashing, for the sake of collecting meaningful data it would be more beneficial to remove them from the swarm outright when they fall below a certain height.

4.2.4 Optimal Topology Inconsistencies

Across all three sets of results, it is noticed that some of the drones in the Optimal Topology behave as if they were in between a 2-hop and 3-hop path. The reason for this is that the topology is created at the start of every simulation iteration, and the results of each of those iterations are averaged with each other. As the Optimal Topology decides what links to connect to based on the information at the time, one possibility is that each time the simulation is restarted, a slightly varied topology is established, resulting in an averaged delay value that falls between the 2-hop and 3-hop drones. Drone 5 similarly falls between the 1-hop and 2-hop drones, so clearly it too sometimes finds a shorter path. This is perhaps the reason behind some of the more skewed results found in this section. Neither distance nor speed seem to have much influence on this selection, as the results are replicated across the eight different trials taken. Ultimately, this means that the proposed Optimal Topology may have an edge case for finding ideal paths in a larger swarm but requires additional work to make this more consistent.

4.2.5 Summary of Observations

In examining swarm drone communications for this investigation, it has been found that communication delay relative to the CH emerged as the primary factor affecting the swarm's reliability and effectiveness. A swarm that would require multiple hops, and therefore further delays, will be significantly less effective than one that is much flatter. This compounds the scalability problem in drone swarms, as inevitably a larger swarm will require more links in order to reach every drone.

In the Bus topology, the main limitation was the excessive distance between nodes 3 and 4, which led to frequent disconnections. Future tests should consider a modified U-shaped configuration, linking each node to its nearest neighbour to reduce communication interruptions. Regardless of this, the Bus configuration was clearly a poor example for efficiency, only serving here as a reason to not use it in drone swarms. Drones in a swarm need to have some redundant connections, else they risk disconnection at the slightest of environmental disruptions. Bus topologies are defined by a direct connection between the preceding and succeeding node, but this simplicity does not come without a cost.

The Tree topology was highly consistent its results, showcasing the effectiveness of a simple network structure. A more rigorous test involving greater aerial distances and complex manoeuvres could help to properly assess its resilience to losing connections. Additionally, testing other Tree configurations in a format more aware of nearby drone positions would be beneficial to assessing the overall ability of this topology.

Finally, the proposed Optimal Topology did not significantly outperform the Tree structure in simulated scenarios, suggesting that more complex arrangements may not yield substantial improvements in reliability over simpler network layouts. However, the links it identifies are more likely to be stronger ones, meaning that drones will be less likely to disconnect over shorter distances. More testing will be required to determine this, specifically for the quality of communications in place.

4.3 Comparison With Previous Works

To establish a proper understanding as to how meaningful the results that have been gathered are, it is important to compare with other existing works in the field. It is also important to note that while these are all similar experiments, they are not exactly the same, and each seeks a different goal with their results. Differing preset parameters could cause wildly different results between each of these research papers and the report presented here.

Ju and Il Son [41] worked on a swarm drone application in agriculture, comparing how single and multiple drones perform differently when under teleoperated or swarm conditions. They had them fly a path over a small plot of farmland, and measured performance criteria in total time, battery consumption, landing inaccuracy and coverage ratio. Of relevance here is the landing inaccuracy, analogous to the positioning error or XYZ positions results acquired. The authors state that the landing inaccuracy for an automated multi-UAV setup was 19.32cm, much higher than the minimum value tested of 8.29cm of a single teleoperated drone. However, it is perhaps more useful to compare to the automated single-UAV setup, which had an inaccuracy of 18.04cm. Between these two results, if the inaccuracy of the single drone is taken to be the expected value, the percentage of error for the multi-autonomous UAV flight is approximately 7.1%. This is lower than the value gathered in the results section, as seen on the charts in Figure 20. This is perhaps due in part to the smaller swarm, but also different control methods as well.

Marek et al. [39] use a simulator to demonstrate a swarm of drones' effectiveness under different control structures and conditions. They modify certain variables, such as GPS positioning error, delays, and two different control topologies, and determine the safest speed at which a minimum distance can be expected to be maintained. The two topologies they researched were one in which every drone connected to a single leader, and the other; a structured topology in which each subsequent drone followed its predecessor. Their findings were that the second control structure, in which multiple drones were connected to and following each other, had to fly much slower in order to ensure a safe distance between each drone. This is not unlike the findings within chapter 4.2, in which drones that are farther away from the leading drone tend to crash when the speed is high enough.

5.0 CONCLUSION

Swarm drones are a novel technology, whose existence largely hinges on the ability of the swarm to optimally communicate with every other drone. Development of new methods, or iterations upon existing ones, are critical to advancing swarm drones into more sophisticated applications. In this particular investigation, three separate drone swarm topologies were compared, to evaluate their performances. The tests involved a swarm of eight drones flying in a grid formation, diagonally and upwards, with the spacing between the drones and the speeds they are travelling at changing between tests. From this, it was determined that the Bus Topology suffered most in the case of high-speed movement, due to its long communication delays between the CH and final member of the swarm. The Tree Topology suffered less, due to a reduced number of hops to the targeted drone. Similar crashes only occurred at much higher speeds, indicating a higher reliability than Bus Topology. More hops between nodes were proven to increase delays, as the results showed discrete increments in steps of approximately 30ms, which is to be expected given the processing time on the drone's flight computer. Particular drones in the Optimal Topology tended to vary in how their pathing was determined. This was likely due to some variations in topology planning at the start of each simulation. This indicates the need for a more consistent path selection process, to avoid skewed, non-deterministic results. Minimising communication delays with the CH is critical to swarm reliability; topologies with fewer hops will have shorter delays. The Bus Topology itself is found to be lacking, due to its poor redundancy. The Tree Topology is effective, but further testing shall be required to assert its value in more complicated scenarios. The Optimal Topology does not consistently outperform the Tree structure, suggesting that simpler network designs may solve the same communication problem more effectively. Should future work be pursued in this area, more rigorous testing is recommended. The author has included some suggestions for future work in the following chapter.

6.0 FUTURE WORK

Undoubtedly, the work here is incomplete. There are many more avenues to further research this topic, as this has only just scratched the surface of potential endeavours in the field of swarm technology.

6.1 Reinforcement Learning

Using machine learning techniques to better coordinate and plan swarms of drones would have been interesting to develop, particularly within the GPD environment which it was designed for. Due to time constraints within the semester, this was unlikely to be implemented properly in time. The goal of training this model could be to find a more ideal topology to use, rewarding connectivity and signal strength, while penalising disconnections or crashes.

6.2 Power Modelling

Likewise, modelling the power usage of the drones was difficult given the short time available in the semester. Possibilities for future work here include feeding back the received power to the leader to then decide what power the transmitter should be given, to attempt to trim any unnecessarily expended power. By optimising this and finding the battery power saved, there can be determined another metric by which to evaluate swarm drone topologies and communications.

6.3 Dynamic Linking

Updating routing tables and topologies mid-flight was unfortunately excluded from this project, as it added a further layer of complexity to the scope. With its inclusion, more robust networks could be properly established.

6.4 Algorithm Effectiveness

The Optimal Topology was not as effective as it could have been. Perhaps by including evaluation criteria more pertinent to its strengths, such as power consumption, it would have clearer advantages. If there is a way to do so, ensuring that the Optimal Topology can find the shorter paths to the CH more consistently would also be a priority.

6.5 Simulator

As was mentioned in chapter 2.4.3, an earlier version of gym-pybullet-drones was downloaded by mistake. While all code appears to work fine, there could be underlying issues with version 0.5.2, or significant advantages with the newer releases. It is highly recommended that if work is to continue with this project, that

the GitHub is checked properly for its most up to date release [28]. Updating this may also allow for the inclusion of extra physics engine features, such as ground effect, downwash and drag, which were excluded from this report for simplicity. In 0.5.2 some of these did not work or were still developing. Tests could be made in the newer version to see how these impact a swarm topology's resilience to environmental effects.

Alternatively, reviewing some of the other available simulators in this report may be of interest as well. OMNET++ in particular has been used in many of the research papers reviewed.

6.6 Additional Areas of Interest

Other areas in the field of swarm drone research that may prove intriguing would include the investigation of the advantages of flooding versus routing protocols. Alternatively, some papers discuss using a PID device to allow disconnected drones to retrace their flight path back towards the base station to allow for recovery. Finally, obstacle detection and avoidance is an omnipresent challenge for drone swarms, and future work in this area would always prove valuable.

7.0 REFERENCES

[1] W. Chen, J. Liu, H. Guo and N. Kato, “Toward Robust and Intelligent Drone Swarm: Challenges and Future Directions,” *IEEE Network,* vol. 34, no. 4, pp. 278-283, 2020.

[2] C. Arnold and J. Brown, “Performance Evaluation for Tracking a Malicious UAV using an Autonomous UAV Swarm,” *2020 11th IEEE Annual Ubiquitous Computing, Electronics & Mobile Communication Conference (UEMCON),* pp. 0707-0712, 2020.

[3] Y. Zhou, B. Rao and W. Wang, “UAV Swarm Intelligence: Recent Advances and Future Trends,” *IEEE Access,* vol. 8, pp. 183856-183878, 2020.

[4] M. Y. Arafat and S. Moh, “A Survey on Cluster-Based Routing Protocols for Unmanned Aerial Vehicle Networks,” *IEEE Access,* vol. 7, pp. 498-516, 2018.

[5] Z. Darush, M. Martynov, A. Fedoseev, A. Schherbak and D. Tsetserukou, “SwarmGear: Heterogenous Swarm of Drones with Reconfigurable Leafer Drone and Virtual Impedance Links for Multi-Robot Inspection,” in *IEEE International Conference on Unmanned Aircraft System*, Warsaw, 2023.

[6] M. &. M. K. &. S. K. &. C. G. Coppola, “On-board Bluetooth-based Relative Localization for Collision Avoidance in Micro Air Vehicle Swarms.,” *Autonomous Robots,* vol. 42, p. 1787–1805, 2018.

[7] F. Alsolami, F. A. Alqurashi, M. K. Hasan, R. A. Saeed, S. Abdel-Khalek and A. B. Ishak, “Development of Self-Synchronized Drones’ Network Using Cluster-Based Swarm Intelligence Approach,” *IEEE Access,* vol. 9, pp. 48010-48022, 2021.

[8] O. G. Bautista, N. Saputro, K. Akkaya and S. Uluagac, “A Novel Routing Metric for IEEE 802.11s-based Swarm-of-Drones Applications,” in *MobiQuitous '19: Proceedings of the 16th EAI International Conference on*

Mobile and Ubiquitous Systems: Computing, Networking and Services, Houston Texas USA, 2019.

[9] A. Phadke, F. A. Medrano, N. C. Sekharan and T. Chu, "Designing UAV Swarm Experiments: A Simulator Selection and Experiment Design Process," *Sensors,* vol. 23, no. 17, 2023.

[10] A. Bujari, C. E. Palazzi and D. Ronzani, "A Comparison of Stateless Position-based packet routing algorithms for FANETs," *IEEE Transactions on Mobile Computing,* vol. 17, no. 11, pp. 2468-2482, 2018.

[11] O. S. Oubatti, A. Lakas, F. Zhou, M. Gunes and M. B. Yagoubi, "Survey on position-based routing protocols for Flying Ad hoc Networks (FANETs)," *Vehicular Communications,* vol. 10, pp. 29-56, 2017.

[12] W. Chen, J. Zhu, J. Liu and H. Guo, "A fast coordination approach for large-scale drone swarm," *Journal of Network and Computer Applications,* vol. 221, 2024.

[13] Q. Ciu, P. Liu, J. Wang and J. Yu, "Brief Analysis of Drone Swarms Communication," in *IEEE International Conference on Unmanned Systems (ICUS)*, Beijing China, 2017.

[14] P. Sharma and G. Singh, "Comparison of Wi-Fi IEEE 802.11 Standards Relating to Media Access Control Protocols," *International Journal of Computer Science and Information Security,* vol. 14, no. 10, pp. 856-862, 2016.

[15] M. Al-Gabri, C. Li and L. Li, "Improving ZigBee AODV Mesh Routing Algorithm Topology and simulation analysis," *TELKOMNIKA Indonesian Journal of ELectrical Engineering,* vol. 12, no. 2, pp. 1528-1535, 2013.

[16] Digi, "About the XBee®-PRO 900HP RF Module," 15 May 2019. [Online]. Available: https://www.digi.com/resources/documentation/Digidocs/90002173/#concepts/c_90002173_start.htm?TocPath=About%2520the%2520XBee%25C2%25AE

-PRO%2520900HP%2520RF%2520Module%257C_____0. [Accessed 19 5 2024].

[17] D. Pereira, M. Rodrigues de Morais, L. B. P. Nascimento, P. J. Alsina, V. Gaboardi dos Santos, D. H. S. Fernandes and R. M. Silva, "Zigbee Protocol-Based Communication Network for Multi-Unmanned Aerial Vehicle Networks," *IEEE Access,* vol. 4, pp. 1-10, 2016.

[18] L. Davioli, E. Pagliari and G. Ferrari, "Hybrid LoRa-IEEE 802.11s Opportunistic Mesh Networking for Flexible UAV Swarming," *Drones,* vol. 5, no. 26, 2021.

[19] M. Stellin, S. Sabino and A. Grilo, "LoRaWAN Networking in Mobile Scenarios Using a WiFi Mesh of UAV Gateways," *Electronics,* vol. 9, no. 4, 2020.

[20] V. Sharma, I. You, G. Pau, M. Collotta, J. D. Lim and J. N. Kim, "LoRaWAN-Based Energy-Efficient Surveillance by Drones for Intelligent Transportation Systems," *Energies,* vol. 11, no. 3, 2018.

[21] W. Zafar and B. M. Khan, "A reliable, delay bounded and less complex communication protocol for multicluster FANETs," *Digital Communications and Networks,* vol. 3, no. 1, pp. 30-38, 2017.

[22] O. O. Kazeem, O. O. Akintade and L. O. Kehinde, "Comparative Study of Communication Interfaces for Sensors and Actuators in the Cloud of Internet of Things," *International Journal of Internet of Things,* vol. 6, no. 1, pp. 9-13, 2017.

[23] Digi, "Digi XBee ZigBee," n.d.. [Online]. Available: https://www.digi.com/products/embedded-systems/digi-xbee/rf-modules/2-4-ghz-rf-modules/xbee-zigbee#specifications.

[24] H. Ling, H. Luo, H. Chen, L. Bai, T. Zhu and Y. Wang, "Modelling and Simulation of Distributed UAV Swarm Cooperative Planning and Perception," *International Journal of Aerospace Engineering,* vol. 2021, no. 1, 2021.

[25] Keysight Technologies, “Network Modelling,” 2024. Accessed: Jun. 18, 2024. [Online]. Available: https://www.keysight.com/au/en/products/network-test/network-modeling.html.

[26] J. Panerati, H. Zheng, S. Zhou, J. Xu, A. Prorok and P. A. Schoellig, “Learning to Fly: a Gym Environment with PyBullet Physics for Reinforcement Learning of Multi-agent Quadcopter Control,” *International Conference on Intelligent Robots and Systems (IROS),* pp. 7512-7519, 2021.

[27] K. Pereida, “AER1216 Fall 2020,” 2020. Accessed: Apr. 28, 2024 [Online]. Available: https://github.com/utiasDSL/gym-pybullet-drones/tree/master/assignments#on-windows.

[28] J. Panerati, S. Teetaert, H. Hamed, H. Zheng, D. Helfenstein and B. Yu, “gym-pybullet-drones,” Learning Systems Robotics Lab, 10 March 2021. Accessed: Apr. 28, 2024 [Online]. Available: https://github.com/utiasDSL/gym-pybullet-drones/tree/paper.

[29] Bitcraze, “Crazyflie 2.1+,” Bitcraze, 2024. [Online]. Available: https://www.bitcraze.io/products/crazyflie-2-1-plus/.

[30] Civil Aviation Safety Authority, “Drones taking agriculture sky high,” 22 May 2024. Accessed: Oct. 20 2024 [Online]. Available: https://www.casa.gov.au/about-us/news-media-releases-and-speeches/drones-taking-agriculture-sky-high#:~:text=To%20conduct%20swarm%20operations%2C%20you,that%20holds%20a%20current%20ReOC.

[31] Civil Aviation Safety Authority, “Flight Approvals and Permissions,” 11 October 2024. Accessed: Oct. 20 2024 [Online]. Available: https://www.casa.gov.au/drones/flight-authorisations/flight-approvals-and-permissions#Therequirements.

[32] G. Phiri, M. Mokayef, S. Sun Tiang and W. Chin Hong, “Drone Performance Analysis Based on SNR Factor,” in *International Conference on Artificial life and Robotics 2022*, Oita, 2022.

[33] M. Younis and S. Abadpour, “Noise in Communication Systems,” in *Advanced Radio Communication I*, Karlsruhe Institute of Technology, 2018, pp. 83-103.

[34] NXP Laboratories UK Ltd, “Co-existence of IEEE 802.15.4 at 2.4GHz Application Note,” NXP, Sheffield, 2013.

[35] Connectivity Standards Alliance, “ZigBee Pro Specification,” ZigBee Alliance, 5 August 2015. Accessed: Jul. 20 2024 [Online]. Available: https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf.

[36] M. Campion, P. Ranganathan and S. Faruque, “UAV swarm communication and control architectures: a review,” *Journal of Unmanned Vehicle Systems,* vol. 7, no. 2, pp. 93-106, June 2018.

[37] S. De Vries, “UAVs and Control Delays,” p. 52, 09 2005.

[38] Linx Technologies, *ANT-2.4-LCW-RPS*, TE Connectivity, 2018.

[39] D. Marek, M. Paszkuta, J. Szygula, P. Biernacki, A. Domanski, M. Szczygiel, M. Krol and K. Wojciechowski, “Swarm of Drones in a Simulation Environment—Efficiency,” *Applied Sciences,* vol. 14, no. 9, 2024.

[40] JetBrains, “PyCharm,” 2024. Accessed: Oct. 28 2024 [Online]. Available: https://www.jetbrains.com/pycharm/.

[41] C. Ju and H. Il Son, “Performance Evaluation of Multiple UAV Systems for Remote Sensing in Agriculture,” *Electronics,* vol. 7, no. 9, 2018.