Communication Solutions for Scaling Number of Collaborative Agents in Swarm of Unmanned Aerial Systems Using Frequency Based Hierarchy

By

© 2021

Dustin Hauptman

B.Sc., The University of Kansas, 2017

Submitted to the graduate degree program in Electrical Engineering and Computer Science and the Graduate Faculty of the University of Kansas in partial fulfillment of the requirements

for the degree of Master of Computer Engineering.

Chair: Prasad Kulkarni

Co-Chair: Shawn Keshmiri

Alex Bardas

Morteza Hashemi

Date Defended: 28 January 2021

The thesis committee for Dustin Hauptman certifies that this is the approved version of the following thesis:

Communication Solutions for Scaling Number of Collaborative Agents in Swarm of Unmanned Aerial Systems Using Frequency Based Hierarchy

Chair: Prasad Kulkarni

Co-Chair: Shawn Keshmiri

Date Approved: 28 January 2021

Abstract

Swarms of unmanned aerial systems (UASs) usage is becoming more prevalent in the world. Many private companies and government agencies are actively developing analytical and technological solutions for multi-agent cooperative swarms of UASs. However, the majority of existing research focuses on developing guidance, navigation, and control (GNC) algorithms for a swarm of UASs and proof of stability and robustness of those algorithms. In addition to profound challenges in control of a swarm of UASs, a reliable and fast intercommunication between UASs is one of the vital conditions for success of any swarm. Many modern UASs have high inertia and fly at high speeds, which means if latency or throughput are too low in swarms, there is a higher risk for catastrophic failure due to inter-collision within the swarm. This work presents solutions for scaling the number of collaborative agents in swarm of UASs using a frequency-based hierarchy. This work identifies shortcomings and discusses traditional swarm communication systems and how they rely on a single frequency that will handle distribution of information to all or some parts of a swarm. These systems typically use an ad-hoc network to transfer data locally, on the single frequency, between agents without the need of existing communication infrastructure. While this does allow agents the flexibility of movement without concern for disconnecting from the network and managing only neighboring communications, it does not necessarily scale to larger swarms. In those large swarms, for example, information from the outer agents will be routed to the inner agents. This will cause inner agents, critical to the stability of a swarm, to spend more time routing information than transmitting their state information. This will lead to instability as the inner agents' states are not known to the rest of the swarm. Even if an ad-hoc network is not used (e.g. an Everyone-to-Everyone network), the frequency itself has an upper limit to the amount of data that it can send

reliably before bandwidth constraints or general interference causes information to arrive too late or not at all.

This work proposes that by using two frequencies and creating a hierarchy where each layer is a separate frequency, large swarms can be grouped into manageable local swarms. The intra-swarm communication (inside the local swarm) will be handled on a separate frequency while the inter-swarm communication will have its own. A normal mesh network was tested in both hardware in the loop (HiTL) scenarios and a collision avoidance flight test scenario. Those results were compared against dual-frequency HiTL simulations. The dual-frequency simulations showed overall improvement in the latency and throughput comparatively to both the simulated and flight-tested mesh network.

Acknowledgments

I would like to thank the National Aeronautics and Space Administration (NASA), Heising-Simons, Lockheed Martin, DAR, and Paul G. Allen for funding me during my time at the flight research lab. I appreciate the support that was given to me and the opportunities I encountered thanks to their funding.

I would like to thank Dr. Keshmiri and Dr. Ewing for supporting me at the lab for the past seven years. I started this journey as a freshman in computer engineering out of curiosity of what these UAS systems were like to work with. Little did I know that it would become an amazing series of experiences that concluded with the following research. The skills I developed here will be invaluable to my future endeavors. I will always value the advice and encouragement that I received from Dr. Keshmiri over the past few years. I am indebted to him for taking a shot with a computer engineering freshman all those years ago.

I would like to thank the rest of my committee, Dr. Kulkarni, Dr. Bardas, and Dr. Hashemi, for supporting me during my research. I would also like to thank Dr. Yun for his advice during my undertaking.

I would like to thank Aaron McKinnins whose dedication to the team over the years has assisted us in successfully flying many missions. I cannot thank him enough for helping in setting up my own flight tests and dealing with the potential hazards and concerns that were faced. I would like to thank Daksh Shukla, who assisted me in bettering myself and pushed me to achieve more than I could have dreamed of. He helped me understand more about our GNC, than anyone else, so that I could better assist him. I would like to thank Thomas Le Pichon, Aaron

Blevins, and everyone else at the FRL for their assistance during my endeavors and their friendship. I will not forget the good times, and the bad, that I had at the FRL with them.

Table of Contents

List of Figures

List of Tables

1 Introduction

The upcoming years will have a higher demand for UAS technologies, whether it be Amazon deploying UASs to deliver packages to houses or being used for Intel light shows [1]. However, one challenge that is still being addressed is that of scalability. Intel's world record swarm size is controlled through one computer [2] and does not necessarily deal with the same challenges other systems might face. Intel does not need the UASs to communicate to every individual UAS (UAS lacks intelligence to make major decisions) nor do they necessarily need a lot of information from those UASs. They also can stop the UASs and transition them to a hover mode if something goes wrong, which is not an option in fixed wing aircraft. To take on the challenge of scalability within a system that needs every drone to be intelligent, a hierarchy is established to reduce the bandwidth consumed by the swarm, and thus allowing more units to be added and scalability to be better achieved. The lower frequency is used in the upper tiers of the hierarchy as a method of keeping communication between local swarms and the controller, such as a ground station, of the swarms over longer ranges. The higher frequency is used for a local swarm to communicate internally, as it has higher bandwidth available and has shorter range so that local swarms have less impact on each other. This work's contribution is using two communication frequencies (900 MHz and 2.4 GHz) in the swarm and show that by doing so the scalability of the swarm is improved. Each local swarm has a unit dedicated to using the lower frequency communication to keep in contact with the other swarms as well as receiving new

commands from the controller. This can be seen in Figure 1 where the controller is depicted as a Cessna aircraft.

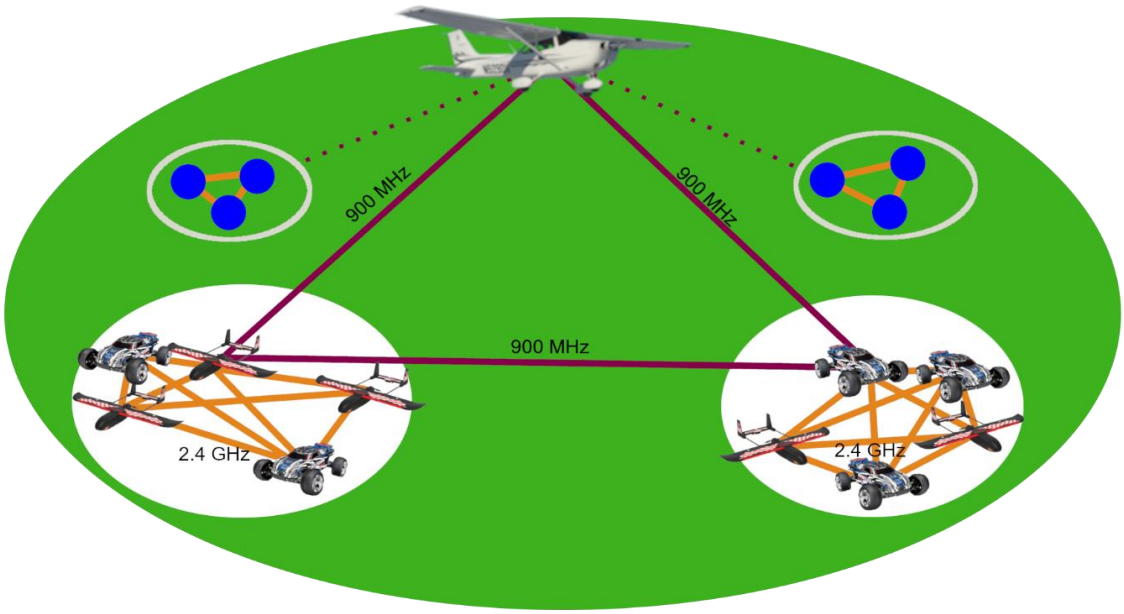

Figure 1: Dual Frequencies Being Used to Establish a Hierarchy.

The purpose for using two frequencies is to expand the limit for the number of aircraft that can run simultaneously. If all aircraft are communicating on one frequency, the bandwidth available on that frequency will eventually be depleted and delays or additional packet loss will occur. By using two frequencies, additional systems can be added to a local swarm until capacity is reached. At this point, a new local swarm can be added. There is also the constraint of needing to work on small aircraft that are using commercial-off-the-shelf (COTS) components. These components are not expensive, and this idea tries to not focus on something that could be solved through more expensive hardware. However, better hardware does not mean the idea would not be applicable. Additionally, there is a size constraint within the fuselage, see Figure 2, meaning that additional equipment that might improve communications are not viable. One difference that

became apparent in later testing is the different communication protocols available on the 2.4 GHz system and the 900 MHz system. The 2.4 GHz did not offer mesh, only Everyone-to-Everyone (E2E). The main difference between mesh and E2E is how data is transferred between the units. In a mesh, the master keeps the network synchronized and handles the Request to Send / Clear to Send **(**RTS/CTS), but this allows two endpoints to talk directly to each other without needing to go through the master unit. In an E2E environment, all data is first routed through the master and then retransmitted to the endpoints. Typically, the advantages of ad-hoc networks are that neighbors can talk to each other freely without needing a controlling unit. This allows the network to self-heal if UASs go outside of the reach of the network temporarily and bandwidth can be saved by only talking to local units. Inside of this multi-frequency hierarchy it might not gain as strong of an advantage. Ideally, the lower the tier on the hierarchy a unit is, the smaller the area it must cover. If the area is small enough, the advantages of ad-hoc do not necessarily apply. Either the movements in the area do not cause the unit to leave the network or the area is small enough to the point not many units can exist inside of it. Later in the results and discussion section, the differences between the two networking configurations are explored. Additionally, ad-hoc is not necessarily scalable. If information is being routed throughout a large swarm, this can cause the units in the center, a critical section for stability, to not send their state information in a timely manner, as they will be routing information from one end of the swarm to the other. This could cause instability within a given swarm.

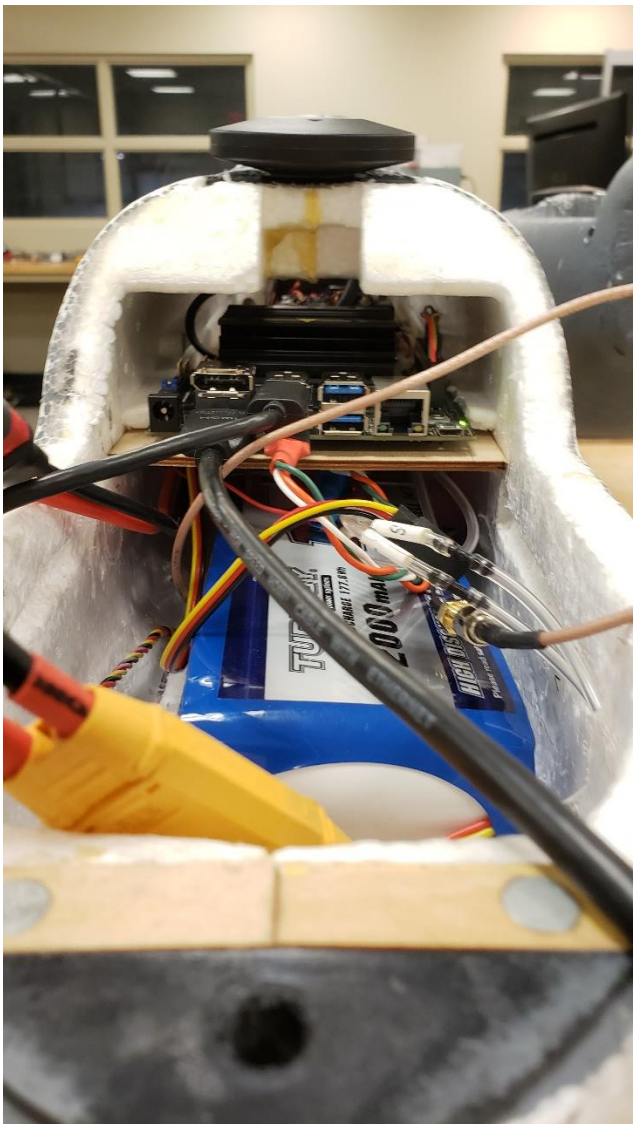

Figure 2: Fuselage

There has already been significant research into hierarchies within the networking realm. Some dive into the creation of clusters and the hierarchy of clusters [3-6] to improve the overall energy efficiency and scalability of the wireless sensor networks (WSNs), however, these fall into the lower mobility category, where the network topology is not prone to movement. This means for the environment at the KUFRL, where a UAS can move 40+ ft/s, the solutions are not ideal. For higher mobility environments, the protocols focus on adaptability of the topology [7]

[8] to handle the dynamic changes and therefore ignore a structure like a hierarchy. Aside from WSNs, hierarchies have been used to provide scalability within the internet like the open shortest path first (OSPF) protocol [9]. These studies show that the idea of a hierarchy being used for a communication network is not new.

Many papers that deal with swarms of UASs or multi-robot missions do not necessarily focus on communication scalability. Some focus on the scalability of the mission (e.g. adaptability/efficiency of a swarm) [10-12] as that is only what they are seeking to show or somehow exploit. Others do not show communication scalability (e.g. packet loss with increased number of agents) [13-15] even when dealing with swarms. This is not to say that no papers mention communications, but they are not the focus [16][17] or are planned for their future works [18].

Hierarchies are also studied within the aviation realm as well [19] [20] and have been shown to be stable and useful for distributing tasks amongst UASs. To this end, it is also useful to seek a hierarchical structure. This would allow for multiple groups of UASs, whose missions can vary, and allow an operator to quickly gather information in a larger area. This, combined with the scalability aspect, allows for a large group of independent swarms to be given various tasks over a larger area (e.g. search and rescue at sea). Another aspect that should be recognized is the difference between how communication is treated in aviation research and communication research. Within the aviation community there are a variety of ways communications are considered that do not necessarily reflect what would be occurring when two or more UASs wanted to talk to each other. In some papers, no assumptions are made about communication [21][22] but information is assumed to be distributed instantly within the swarm. Others assume that communication is occurring [23] but no specifics are given on what the communication

configuration is (i.e. Mesh, P2P, etc.) or the quality of the communication channel. Sometimes the assumption for the communication is that it is not occurring at all [24] and rather the UAS is using other sensors (Lidar/Radar) to detect vehicles. This might work for slower UAS systems, but not necessarily for all UAS types, velocities, or mission configurations. In other works, communications are flushed out and assumptions are made with the intent on trying to work around the potential limitations of a UAS in a radio dense environment. However, the assumptions can be limited. For example, one paper deals with swarm assignment understood that a communication channel would not be perfect, and the swarm assignment algorithm still needed to work [25]. The assumptions on the communication channel is that it is in either a disconnected state or connected state, but no information on if latency or packet loss would have impacts on the overall system. The paper mentions that to avoid collisions the units cannot travel at velocities that would allow them to traverse over a defined communication radius in a single timestep. This implies that at higher velocities the communication radius would need to be expanded. Expansion of the communication radius would allow them to see a UAS before it moves closer, but this also means the power of the communication device would increase and therefore more neighboring UAS communications could be interfered with. Another paper sought to address the potential problems of loss of line of sight and communications within a potentially noisy environment [26]. By knowing ahead of time where a UAS can go, a plan to reroute data can be constructed when line of sight is lost. However, as mentioned, it requires previous knowledge on where the UAS is going. In an intelligent swarm setting, there could be a general area or path the UAS is constrained to for a mission, but the movements of the UAS and how it decides to approach the mission is ultimately up to the algorithm running on board.

What is different between this paper and the previously mentioned papers is the application. Dual frequency communication lines are not touched upon in the previous papers. Admittedly, those papers deal mostly with the algorithms to form the hierarchies, control of the swarm, or the scalability of the algorithm and do not assume much about the communication line. However, by using two communication frequencies the amount of bandwidth consumed should be lower than if all UASs are talking on the same frequency. Furthermore, the ideas presented in the papers can still, to some degree, apply to the dual communication frequency, further expanding the scalability. The aim of this research is to find out to what degree the dual communication frequencies improve the overall scalability of the systems.

2 Theory

The reasoning behind choosing a dual-band communication system will be explained here. It is known that for each communication line, there is only a certain amount of bandwidth that can be used before no more data can be transferred. For the 900 MHz the wireless transmission bandwidth is 172000 bps. Given that the main data packet being transmitted currently to other UASs and the ground station is of size 240 bytes, and that the message is transmitted at a rate of 5 Hz. The estimated number of UASs that could potentially be allocated on this frequency is shown below in equation 1.

$$\left\lfloor \frac{172000\ bps}{(240\ bytes * 8\ bits\ per\ byte) * 5Hz} \right\rfloor = 16\ UASs \quad (1)$$

The number conversion is floored as only a whole UAS can be supported. This calculation is ideal as it assumes 100 percent of the bandwidth is used and no errors occur during any point of a message transmission and no other information is being transmitted. The 16 UASs could be extended further if the data message is reduced to only the relevant information, such as position and velocity. The remainder of that information is useless for the UASs themselves when communicating to other UASs. If the information is limited to only the information the UAS uses, then the number of supported UASs becomes equation 2.

$$\left\lfloor \frac{172000\ bps}{(28\ bytes * 8\ bits\ per\ byte) * 5Hz} \right\rfloor = 153\ UASs \quad (2)$$

Now, this calculation only shows how many UASs could potentially fit within the limited bandwidth, but that number would not be achievable without intensive communication software running in the background. However, it could work if that small information packet is separated onto another communication frequency. In this case, 2.4 GHz will be used. This is for two

reasons, the first being the need for an additional frequency to have the smaller packet on, the second is for the properties of 2.4 GHz frequency. At the 900 MHz frequency, the expected range and penetration capability is higher, but the available bandwidth is lower. Conversely, the 2.4 GHz frequency has shorter range and penetrative capabilities but allows for higher bandwidth, which is desirable for local swarm communications. Considering the shorter range, in theory, the local swarms could be separated by a certain distance to minimize the effects of interference coming from neighboring local swarms. By combining both frequencies, it should allow for many more UASs to be added to a swarm. Taking the two calculations above, however unrealistic they may be, with each UAS on the 900 MHz frequency supporting a local swarm, up to 2432 UASs could be supported. The 16 UASs on the 900 MHz frequency would also be part of the local swarms and should be removed from the total count.

$$(16\ UASs * \ 153\ UASs) - 16\ UASs = 2432\ UASs \tag{3}$$

It is unlikely that this is achievable as those numbers are made with ideal assumptions, however, at a smaller scale the theory still holds. Based on experience working with these systems, rather than sixteen UASs on the 900 MHz, four to six would be more realistic and instead of 153 UASs on the 2.4 GHz, four to six UASs would also be more achievable. This would allow the system to go from only four to six UASs only using the 900 MHz to twelve to thirty UASs in the dual-band system.

It might also be possible to further expand upon scalability by using channel splitting. By giving the radios specific frequency ranges to work on, additional radios can be added to separate frequency ranges without the fear of causing significant interference. This could boost the number of agents in the swarm and allow us to scale further. This idea is explored further in the Results and Discussion section.

3 Hardware and Implementation

3.1 Hardware

A block diagram, shown in Figure 3, lays out the avionics for our UASs. The main computer on the avionics is the Jetson Nano, which runs Ubuntu 18.04 as the operating system. The Robotic Operating System (ROS) framework is installed onto the Nano. This framework runs all the software for the UAS. The Nano retrieves information about the state of the UAS from the Pixhawk Flight Controller. The Pixhawk acts as a data acquisition board and collects the information from the attached IMU, GPS, and pressure sensors. The Pixhawk also allows servos to be controlled from the Nano using its offboard control method. These control values are sent over Mavlink messages using a USB-serial interface. Manual control can be enabled in the Pixhawk software as well and is controlled via a PPM signal from the 2.4 GHz receiver from the controller.

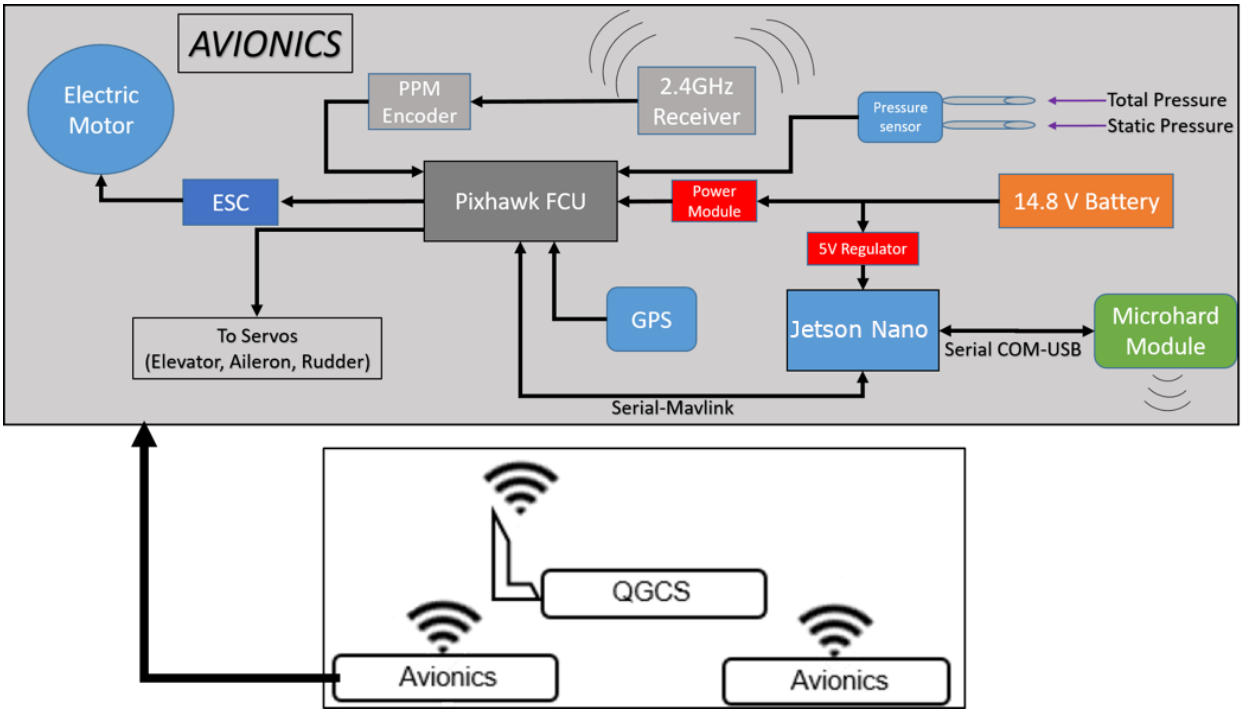

Figure 3: Avionics - Figure modified from [27]

3.2 Communication Flow

In this section, the details of how data is transferred between two endpoints will be discussed. Mavlink, a “library for lightweight communication” [28], is used to encapsulate the data being transmitted. Figure 4 shows a minimal packet structure of the mavlink message. The header contains all the information to describe the packet, its destination, and its source. Only 1 byte in the header is dedicated to the payload size which causes the maximum payload size to be limited to 255 bytes. The payload is also limited to 64 possible fields. If the 64 fields are filled before the payload size is met, no more fields can be added to the message and, therefore, no more data can be added to the message.

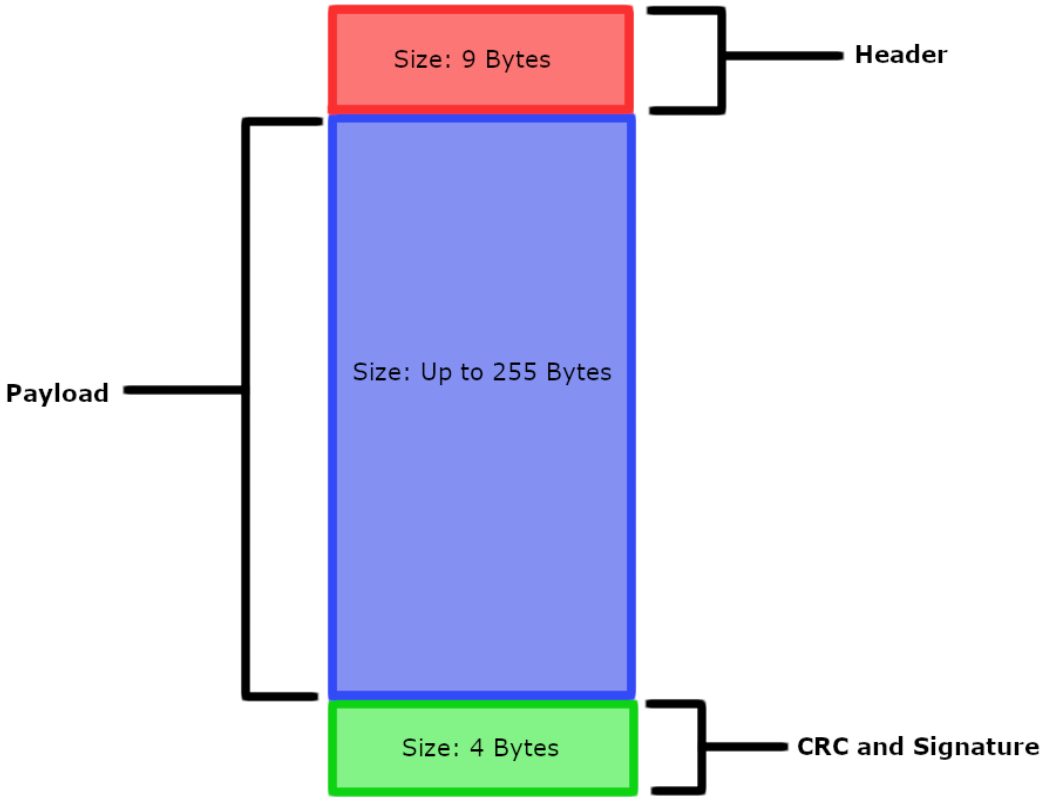

Figure 4: Mavlink Packet Structure

Once the data is formatted into a packet, it is transmitted via a RS-232 connection to a Microhard P900 or P2400 [29] [30] at a specified rate. Inside the Microhard there are up to 255 internal buffers, that can be up to size 255 bytes. This is detailed in Figure 5.

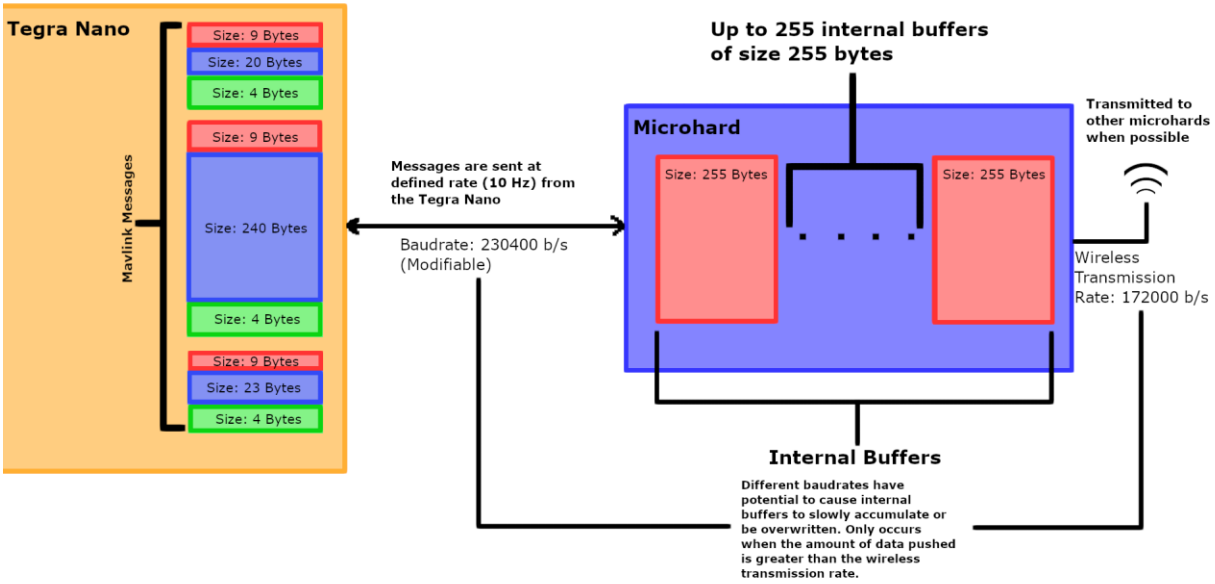

Figure 5: Data transfer between the Jetson Nano and the Microhard unit

The data is then transmitted over the wireless link and broadcast to other Microhard units. The microhard units are set up in a mesh network configuration. This means that any unit can talk to another unit without needing to go through a centralized node. There is one controller node that ensures that the exchange of data between two points in the mesh is synchronized, but data does not need to flow through that node, only the process of setting up the synchronization. The protocol used to exchange data can be changed between time-division multiple access (TDMA) and RTS/CTS. In case of the TDMA protocol, very little overhead is exchanged between the nodes for communication as each node will have their own time slot to transmit the data. RTS/CTS requires each node be given a random wait time to request the ability to send.

Ideally, the nodes would send to a specified nodes MAC address when transmitting data for specific devices. However, most messages are meant for every UAS within the swarm, not an individual UAS. The only messages that would potentially benefit from this unit addressing would be waypoint messages as those would be directed either to the ground station or the leader of the local swarm. As such, the current setup uses broadcast messages, and packets are filtered at the application layer on the Nano. Upon successful arrival to other units the mavlink message is captured by Intel's MAVLink Router (IMR) [31]. The IMR will scan the incoming data for a valid mavlink message and once it finds one, it will transmit the packet via UDP to any endpoint specified. The only current endpoint given is a ROS node whose job is to unpack the mavlink messages and extract the desired information. This setup can be seen in Figure 6, which only depicts one Microhard.

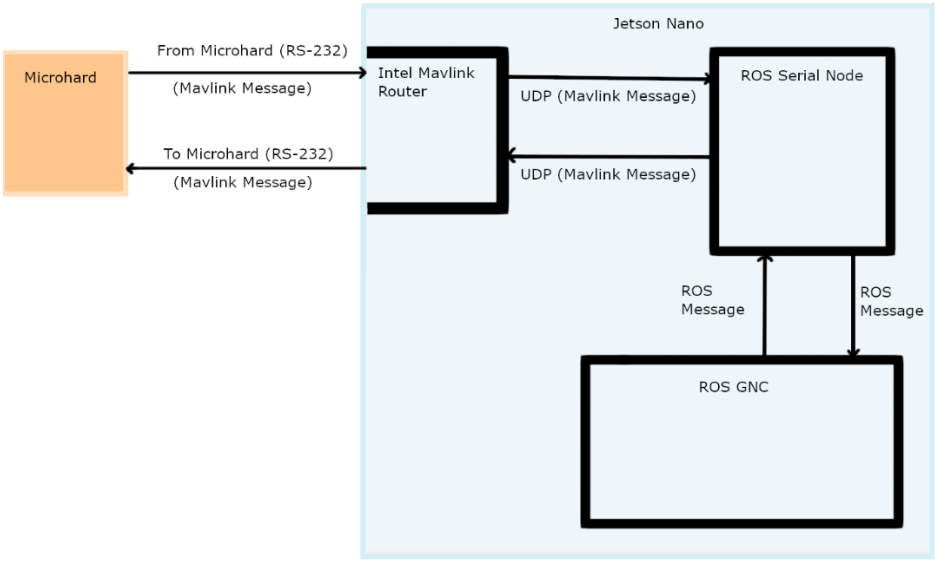

Figure 6: Internal Communication Flow of the Jetson Nano

It then publishes that information to the rest of the ROS system via ROS messages. This node also subscribes to ROS topics that give information about the current location and attitude

of the vehicle. These subscriptions are spawned in separate threads from the main execution loop of the node and will pull messages into a structure as soon as the data is available. The main execution thread is started in a separate thread within the ROS node and its job is to receive incoming messages and send outgoing messages. This occurs at a fixed rate as specified by the user. The fixed rate only occurs within the main loop and should not affect the subscription threads. This is to ensure that the amount of data pushed to the Microhard is not overwhelming the internal buffers and causing overwrites, but that the data that is pushed will be up-to-date information about the state of the avionics. The flowchart for the code execution can be seen in Figure 7.

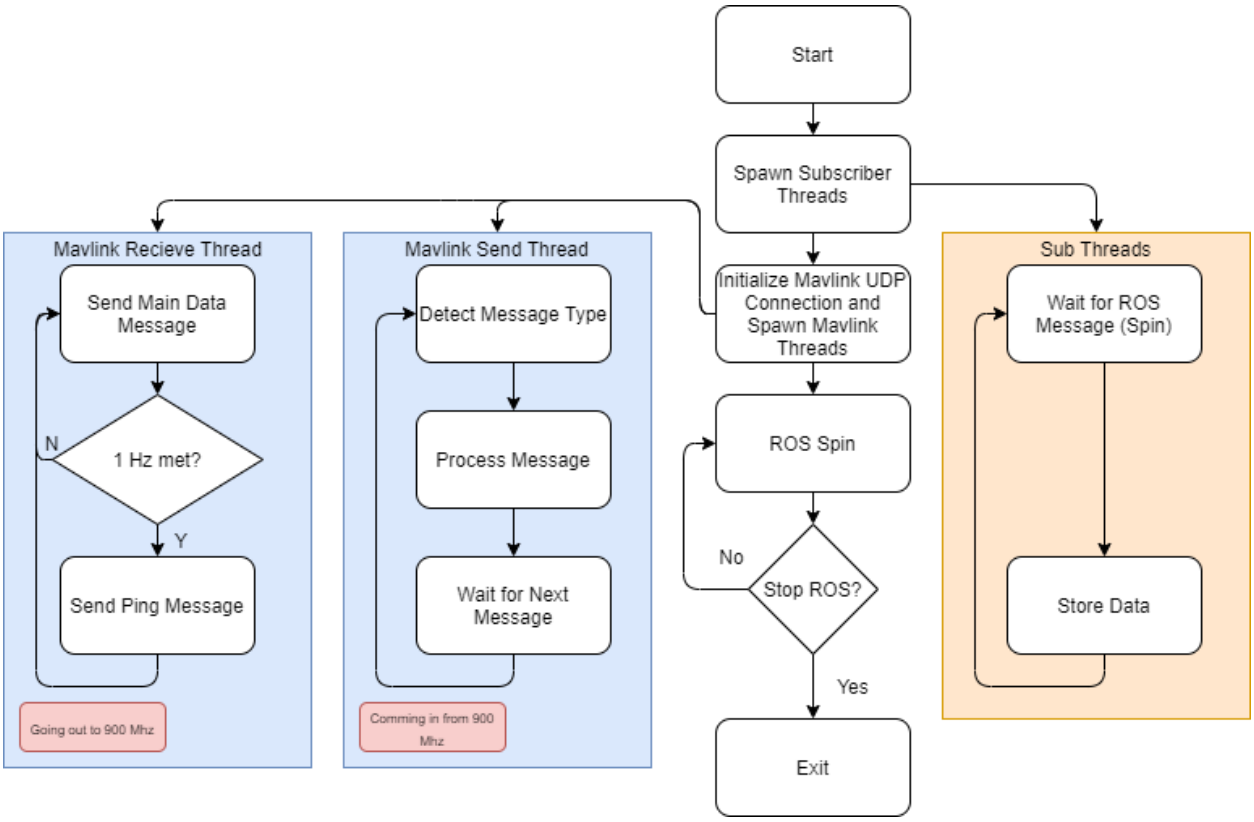

Figure 7: ROS Serial Node Flowchart

For the dual-band hierarchy to work, a small tweak needs to occur within the ROS system. When it is expanded to two frequencies, there should be some information that is communicated between both the 900 MHz ROS serial node and the 2.4 GHz ROS serial node. This is because the 900 MHz node will receive more information relating to the overall mission and that data needs to be disseminated to all lower UASs within the hierarchy. Conversely, UAS information from the lower rungs of the hierarchy need to be displayed on the ground station, which talks on the 900 MHz channel. Figure 8 shows this dual-band setup, and how the internal ROS structure is planned. Notice that on the 2.4 GHz Microhard, there is no IMR. This is because IMR expected a connection before allowing information to be passed to the serial node. If two IMR connections are used on two different UASs, and nothing else, there will be no communication between the units, because both units are expecting the connection. On the 900 MHz, the ground station initializes the connection. Since there is no ground station on the 2.4 GHz, the serial node directly connects to the Microhard via pymavlink. Pymavlink does the same handling as IMR, but in the past it seemed to have problems when the packet size was too large or being sent too fast.

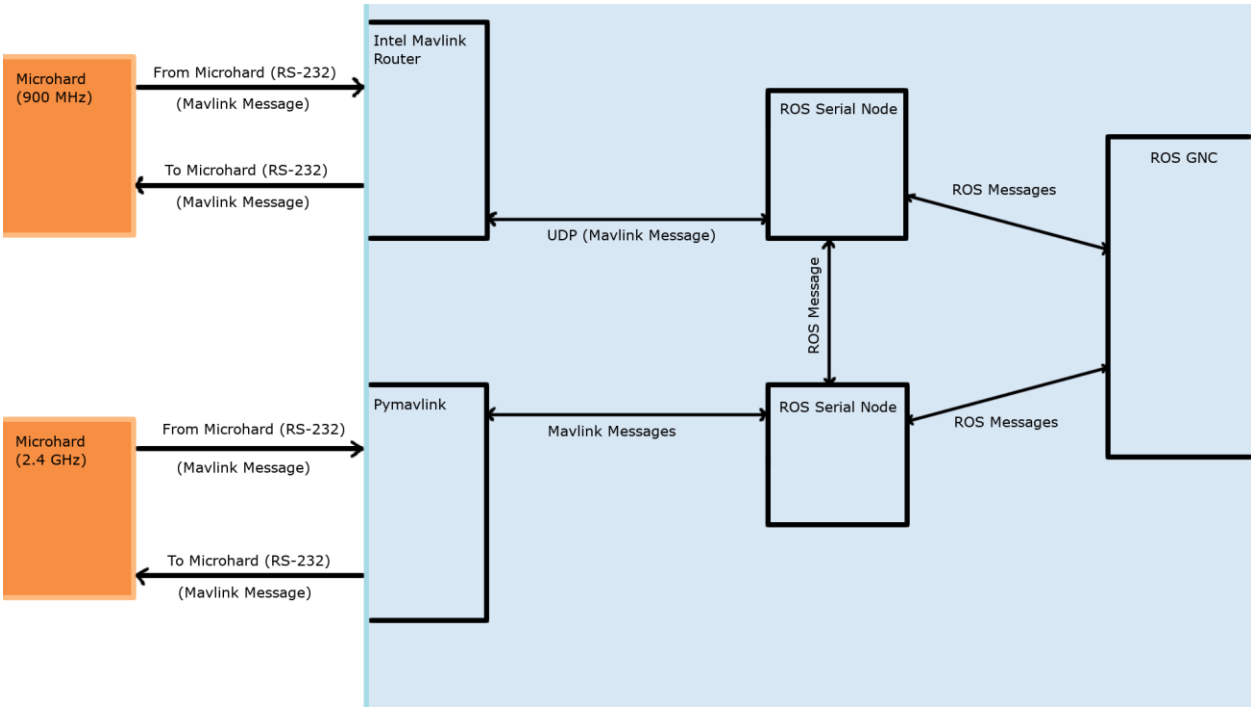

Figure 8: Dual-band Internal Communication Setup

3.3 Considerations

In this section, some of the decisions that are decided on early in the design process will be discussed. Earlier in this chapter, in Figure 5, is a description of the internal buffers of the Microhard. In theory, that means the Microhard can hold up to ~65,000 bytes before broadcasting them out. The problem with this is that there can be a slow buildup of messages inside the internal buffers. This can lead to information being delayed for up to 2-3 second, or even worse, valid data being sent for 2-3 seconds after a software failure occurred. This delay is unacceptable when it comes to UAS information and can cause a plane to become unrecoverable without an operator's knowledge, cause delays to the ground station operator who is relaying critical information to the pilot, and cause general confusion to what actions are visibly seen and what is being shown. To keep this latency low, the number of buffers is reduced to three for

initial flight tests. However, a higher number of internal buffers is ideal as this will reduce the chances of overwriting the internal buffers and potentially causing messages to become corrupted before the buffers are sent.

Another decision made early on is the use of Mavlink. This choice is made for a few reasons. Mavlink is a simple method for converting data into some form of packet that can be transferred. The emphasis on the minimalistic packet structure along with the small packet size made it an ideal candidate for real-time operations for the UAS. Alongside this, multiple open source ground stations supported the standard messages already included within Mavlink. This made early development quick and significantly easier, and only minor modifications for the ground stations are needed to support custom messages. It is also significantly easier for aerospace majors to adjust the messages and integrate their functionality into the UAS. Since it is not guaranteed that anyone working on this system would necessarily have the programming background to understand a custom message format, buffer handling, CRC computation, or setting up a connection to a UDP port and serial port, it then becomes most paramount that the software is easier to understand and does not require too much time to be understood and modified if the need arises. This is also partly why IMR is used as it will handle the buffer management and detect the messages within those buffers. This also comes from experience from working with some of the earlier systems. The Flight Research Lab (FRL) previously had a custom ground station that had a custom communication protocol. The format of a packet is described in an XML which is then referenced as the raw buffer is being scanned for messages. When new data is requested to be added to the ground station, it becomes a near impossible task due to lack of documentation. To avoid situations like this Mavlink is chosen as it has readily available documentation on how to use it and requires minimal training time to understand how

it works. In conjunction with using IMR, the most that needs to be understood from the perspective of a new person is how to differentiate between message types as IMR sends a detected message and not a buffer. To summarize this point, the decisions that are being made in the software must not be so complicated that it takes multiple months for someone to understand how to add or remove the data they need long after the last person trained on it has left, because the communication software must support and be understood by everyone who works on these systems.

4 Experimental Methodology

For most of the tests, Hardware in the Loop (HiTL) simulations are used. HiTL simulations are tests involving actual flight test ready avionics, inside of a grounded UAS. Using Six degrees of freedom (6DoF) simulations, the aircraft then simulates itself flying after being given an initial set of states and waypoints. This is used instead of software testing as it has all necessary parts (i.e. UAS spending time transferring and reading information from one another, radio waves colliding in the air, etc.) of actual flight accounted for, while requiring actual flight.

The general process for both indoors HiTL simulations and flight testing is to prepare each UAS that is going to be used for the test. This included modifying the configurations of the Microhard units, ensuring that each UAS has its own ID, and making sure UAS has the same software running on it. Once each UAS is prepared and the ground station is ready, the units are turned on as specified by the test. The UAS is then allowed to run its mission, which is uploaded from the ground station, for some time without interruption, unless parameter changes occur. If parameter changes occur, the UAS will listen and respond to the ground station as quickly as possible before returning to normal communications. During the parameter change the plane is still “flying”, so once normal communications return, state information about the UAS is up to date. Once the UAS has finished its test, a shutdown command is sent to the UAS. Data is then collected from the UAS and processed later. The data collected can include latency information, sequence loss for ping messages, sequence loss for data messages, and throughput percentages. Throughput percentages are calculated by taking the number of data messages sent and dividing by the number of messages received by a particular UAS. More details will be given for each individual test in the Results & Discussion section as each test can vary in what is being used and how it is configured.

5 Results & Discussion

5.1 Justification for Work

One of the initial tests that is done on the Microhards, is to see if there is a limit to the number of units that can be placed on the 900 MHz frequency to justify using two frequencies. One register in the P900 Microhard is the “Repeat Interval” or S115 register. In their manual, they suggest setting this value to “number of devices / 2” (i.e. if there are 10 UASs in operation, S115 should be set to 5). This test is performed with two devices in HiTL mode talking to each other. After each test, the Microhard’s S115 value is changed. The register values being tested are 1, 3, 10, 20, 100. Through experimentation it is shown that S115’s value can dramatically alter the throughput of the device, as seen in Figure 9.

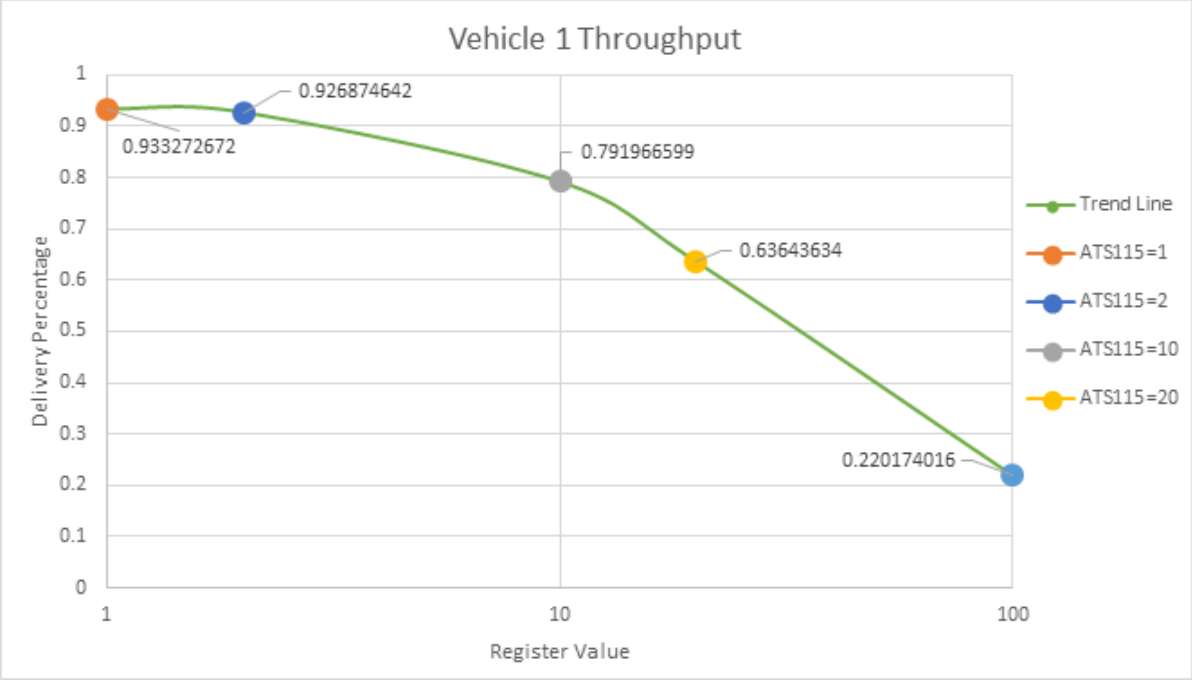

Figure 9: Register S115's Value Versus the Throughput of a UAS

Setting S115 to the appropriate value does correspond to the highest throughput. However, an important note is that as the register value increases, the throughput of the device lowers dramatically. These estimates are also liberal, as there are only two devices and no other traffic is being sent from other units. If 200 devices are used, the probability that two will RTS at the same time goes up and since only one will get the CTS, the others throughput will go down. This experiment shows that adding additional devices, regardless of available bandwidth, will reduce the throughput of all other devices on the network. This also shows that at some point, the network itself will run out of resources for additional units and a new schema must be used to scale the network. With that being said, in the next section, a baseline for the 900 MHz frequency will be found.

5.2 Baseline for 900 MHz Frequency

For this experiment, four planes are tested inside in HiTL mode along with one ground station. The ground station is the master of the mesh communication network. They are set up as a normal flight test operation. Ping messages are being sent at 1 Hz while the main data message is sent at 5 Hz. All units are powered up at approximately the same time.

One of the first things that stood out is that R2's (vehicles are referred to as R1 - R4) ping is significantly worse than all other UASs. Not necessarily in latency, but in its sporadic nature of receiving pings as seen in Figure 10. While all other UASs consistently receive the packets, as seen in Figure 11, R2 seems to struggle and misses a high number of sequences for the ping message. The first thought to test is a possible bias in terms of power up order. The Microhard is using RTS/CTS for the communication protocol and the ground station unit monitors to see if the RTS should be allowed or not. It is thought that possibly a RTS from the first UAS to power up would give that UAS advantage in the future as it would be the first one to RTS the second time

since all units are sending at a regulated 5 Hz. To check to see if this is a possibility, the second test is conducted.

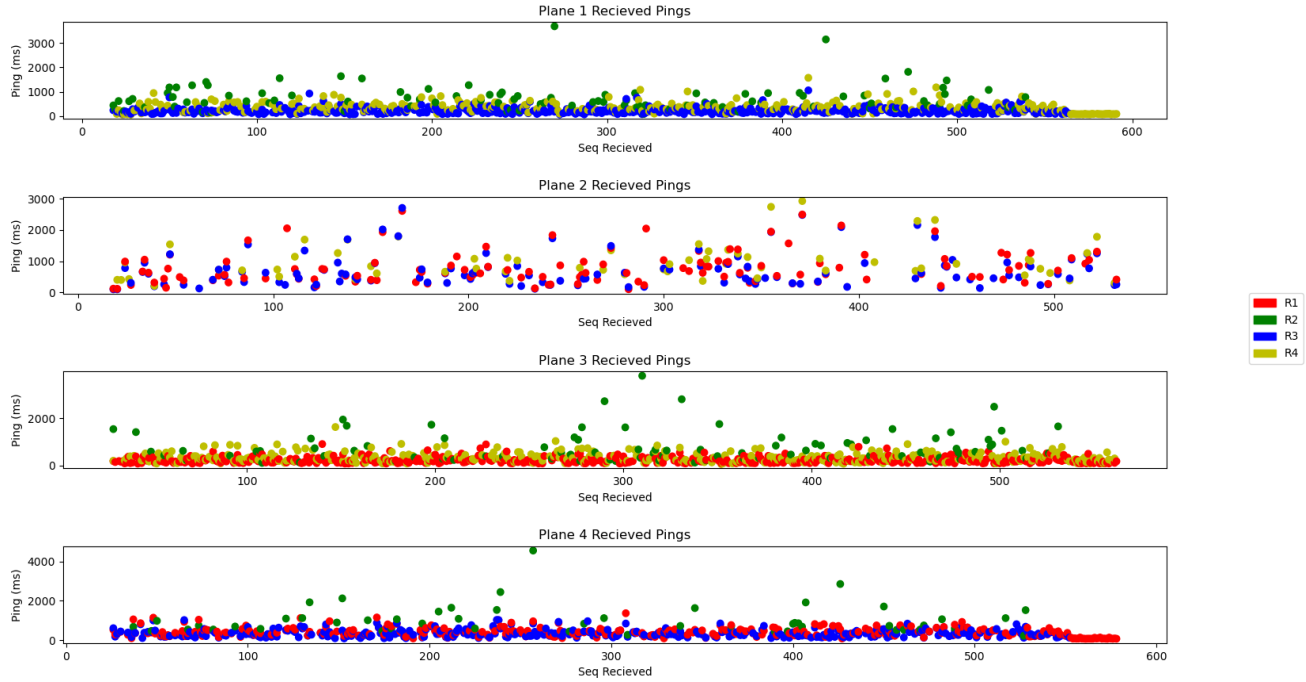

Figure 10: Latency Results for Baseline Testing

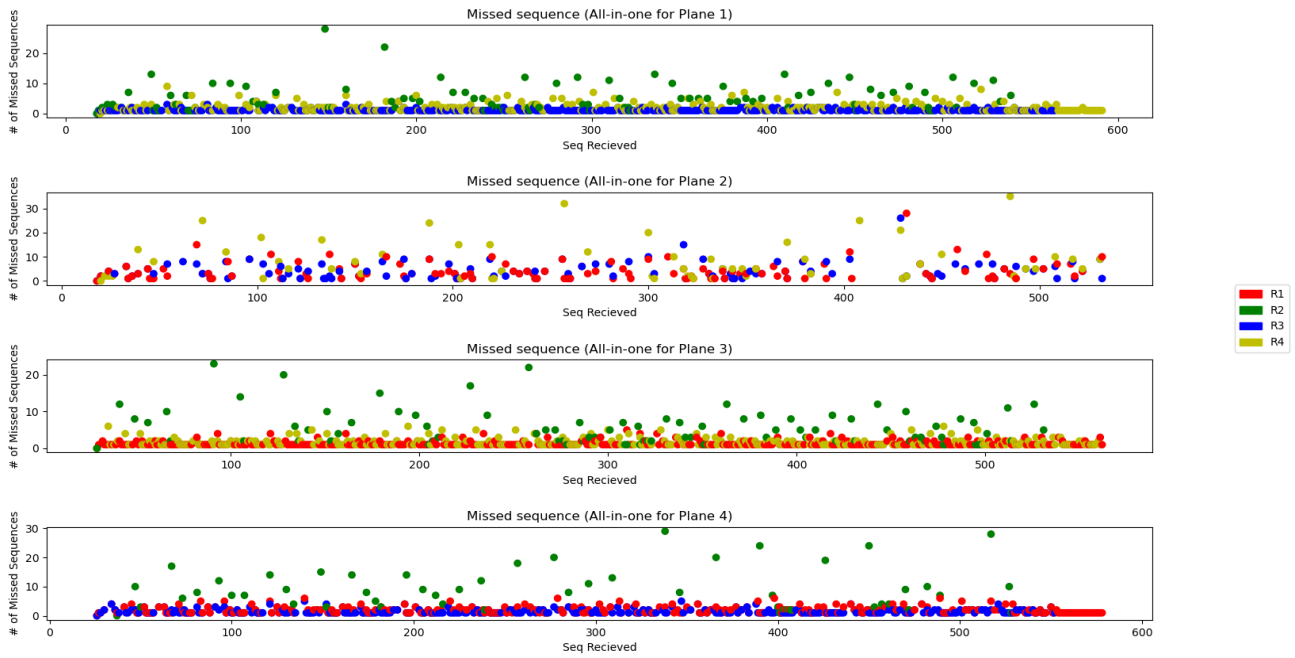

Figure 11: Sequence Loss for the Ping Message During Baseline Testing

5.3 Baseline - Checking Power Up

This test is conducted the same as the first, the only difference being the power up order. In this test, the power up is R1 -> R2 -> R3 -> R4. After powering up one unit, it is left on for approximately 60 seconds before the next unit is turned on. This is to ensure that both units are settled, and data can be collected for only those units being on. The test is then repeated and the power up order is reversed (R4 -> R3 -> R2 -> R1).

After running the first power up test, it seems that the suspicion is confirmed. R4 has the worst performance, as seen in Figure 12 and Figure 13, in terms of receiving ping data from all other units while everyone else performed about the same. However, before concluding anything, a second test is conducted with the reverse power up order to confirm the bias. This time there is a significant change in behavior of the system, which is seen in Figures 14 and 15. Once R1 (the last unit to be powered) is turned on, all communication within the system is

quickly halted. The cable going to the Microhard is flashing green, indicating data is being sent, but none of the units are communicating with one another and the ground station showed that there is no new information from any of the UASs. The test is conducted again, shown in Figures 16 and 17, to confirm that this phenomenon is occurring. The second test revealed that it is, and subsequent tests all confirmed that communications would stop after some time. After some experimenting, it is discovered that IMR is sending corrupted packets. Essentially, a bug is occurring where if corruption started to occur, IMR would repeatedly and quickly send only the corrupted messages. The communication channels would quickly shut down as all resources are being dedicated to this rapid sending of corrupted messages. This then reinforces the idea that there is an upper limit for a singular frequency where information can no longer be successfully transferred. If the corrupted messages are reinterpreted as other information being sent (e.g. route discovery packets), then those packets have a direct impact on other UASs information being successfully transferred. A patch made for IMR made it to only send known, valid messages and this seemed to fix the problem. The baseline for that will be shown in the next section.

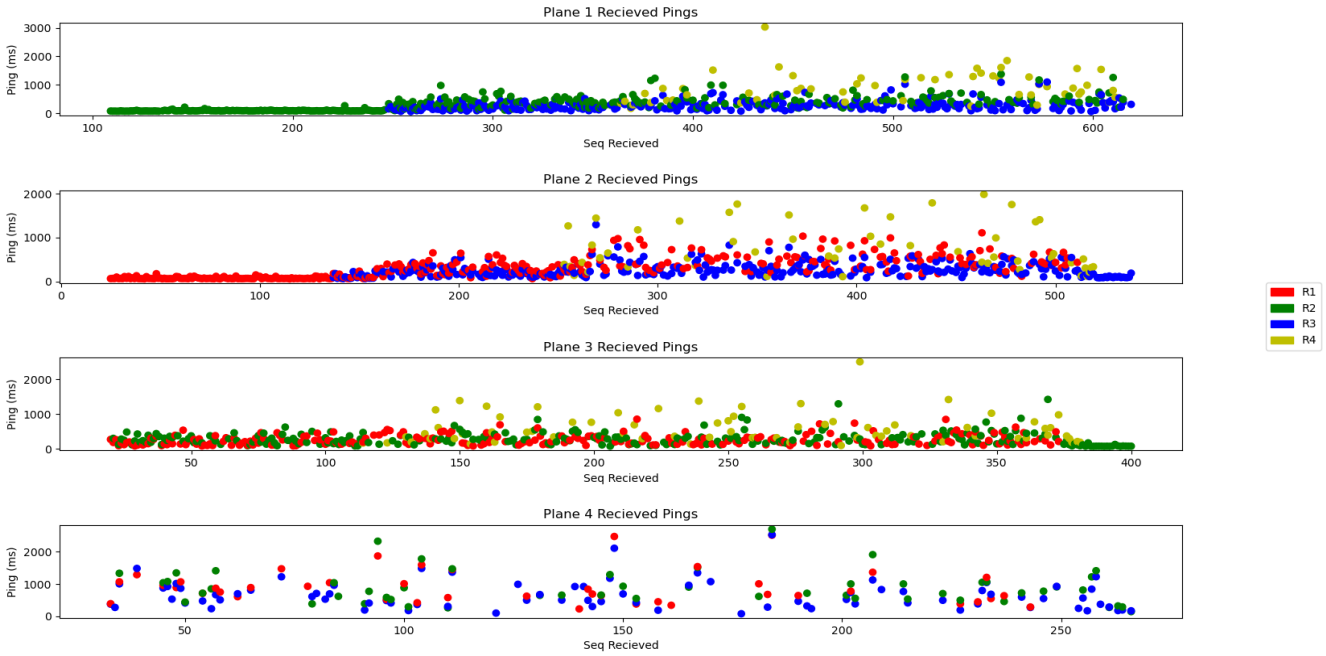

Figure 12: Latency Results for Baseline - Power up R1, R2, R3, R4

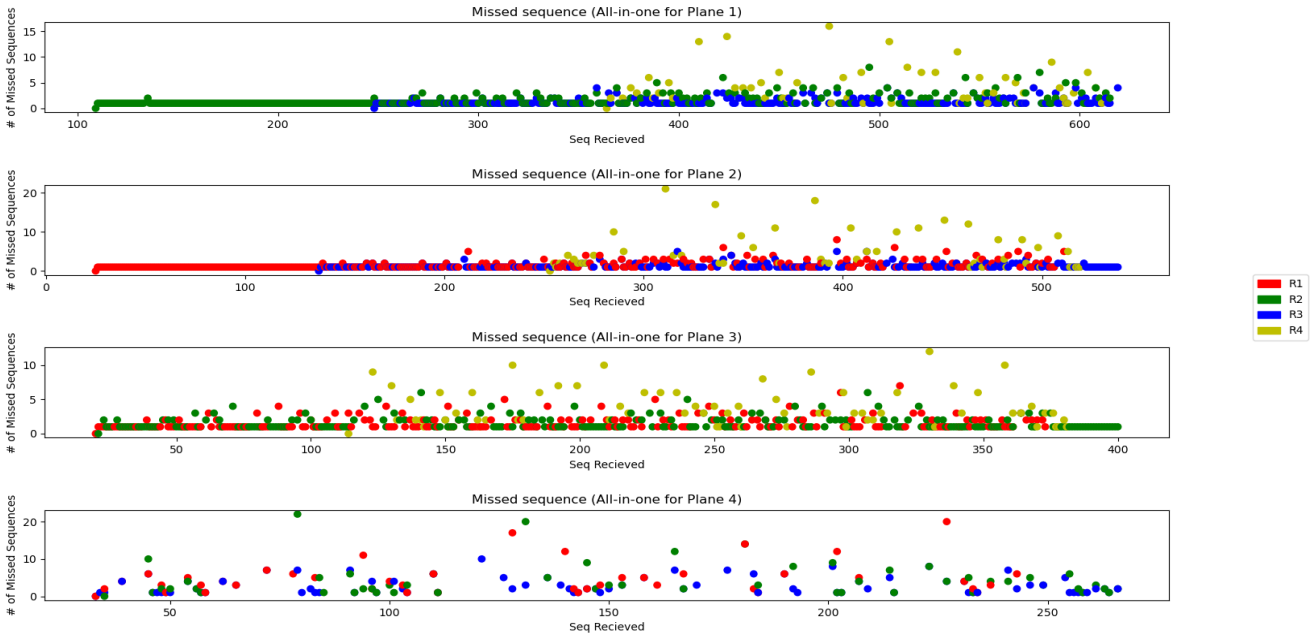

Figure 13: Sequence Loss for Ping Message - Power up R1, R2, R3, R4

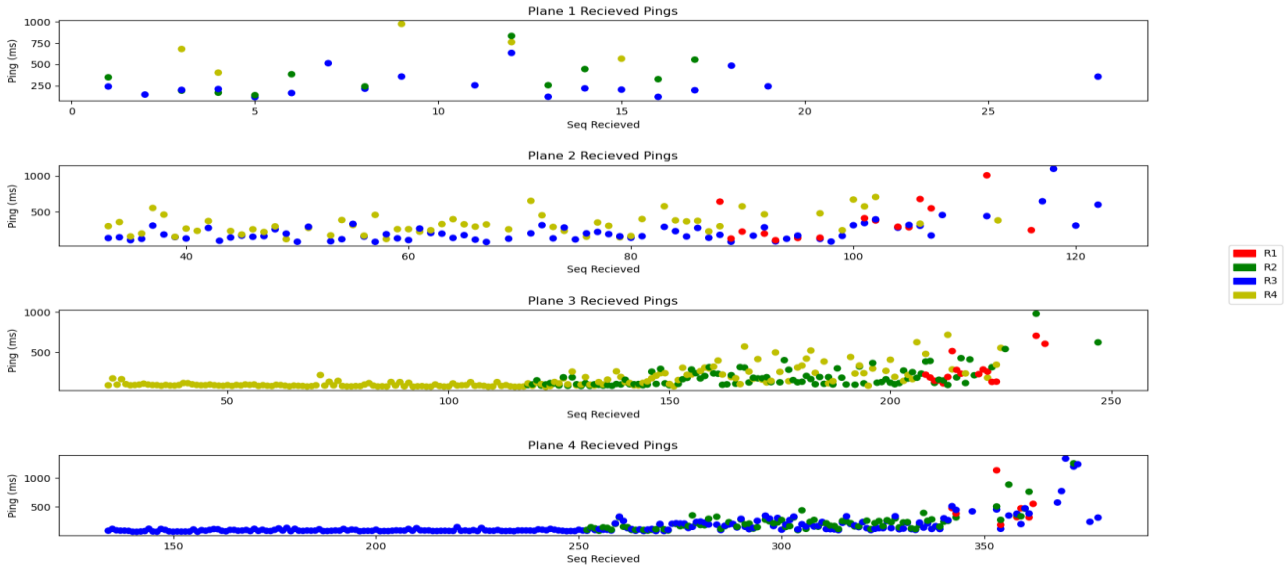

Figure 14: Latency Results for Baseline - Power up R4, R3, R2, R1

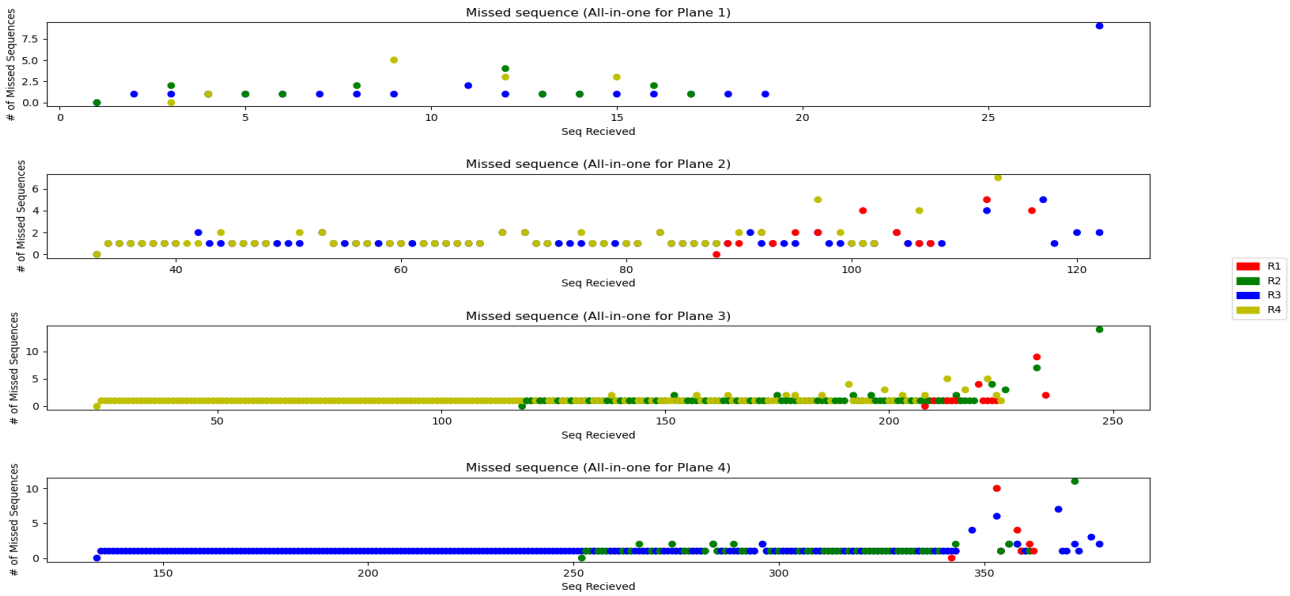

Figure 15: Sequence Loss for Ping Message - Power up R4, R3, R2, R1

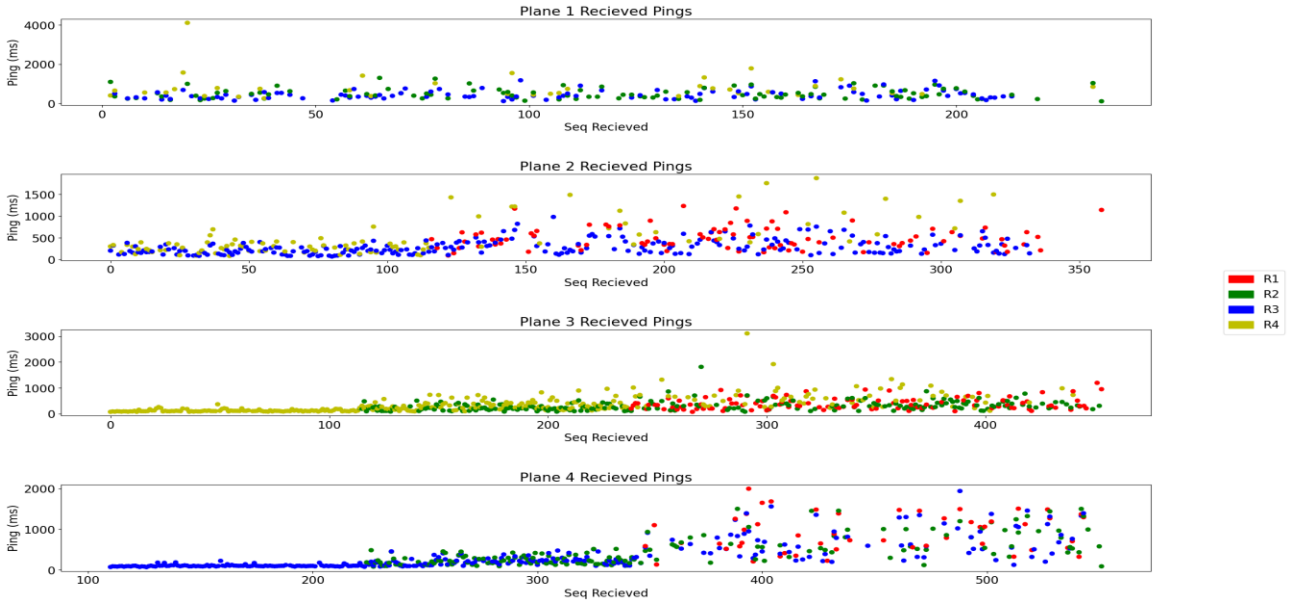

Figure 16: Latency Results for baseline - Power up R4, R3, R2, R1 - Test 2

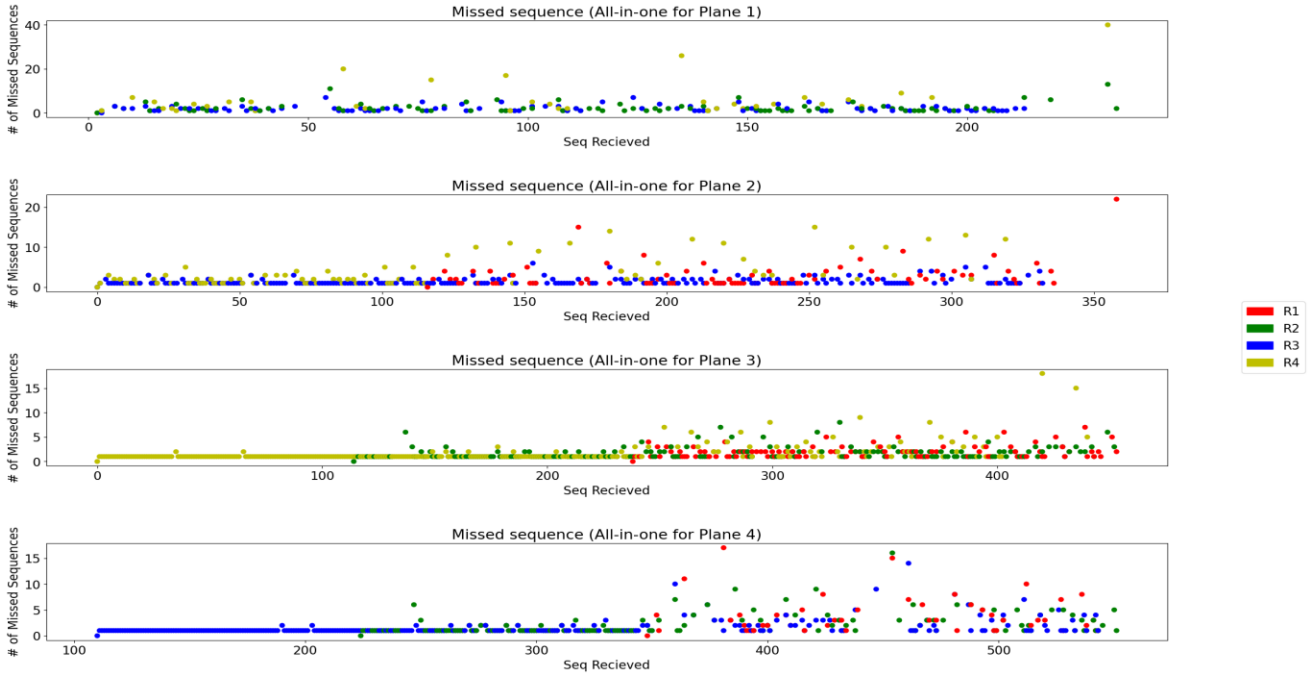

Figure 17: Sequence Loss for Ping Message - Power up R4, R3, R2, R1 - Test 2

5.4 Baseline - IMR Patch

Once the IMR patch is installed and working, another test is conducted to find the baseline for the 900 MHz. This test is conducted the same way as the first baseline test. From the results in Figures 18 & 19, it can be seen that that the patch significantly helped reduce the number of missed sequences and seems to have helped stabilize the behavior between all the units. There is no longer any latency above ~600 ms and sequence loss has generally stabilized between all the units. Although it may seem that the sequences for the ping are bad, going up to 6 seconds of missing sequences (1 Hz * 6 Sequences missed), the ping message is smaller than the main data message and sent at a slower rate. Figure 20 shows that the majority of the missed sequences in the data message all fall below 5 missed sequences. This means that very rarely will one second pass with no new information being given to the UAS. On top of that, the majority of

sequences are not missed. This baseline is stable enough for a flight test to be conducted with these settings. The flight test results will be presented in the following section.

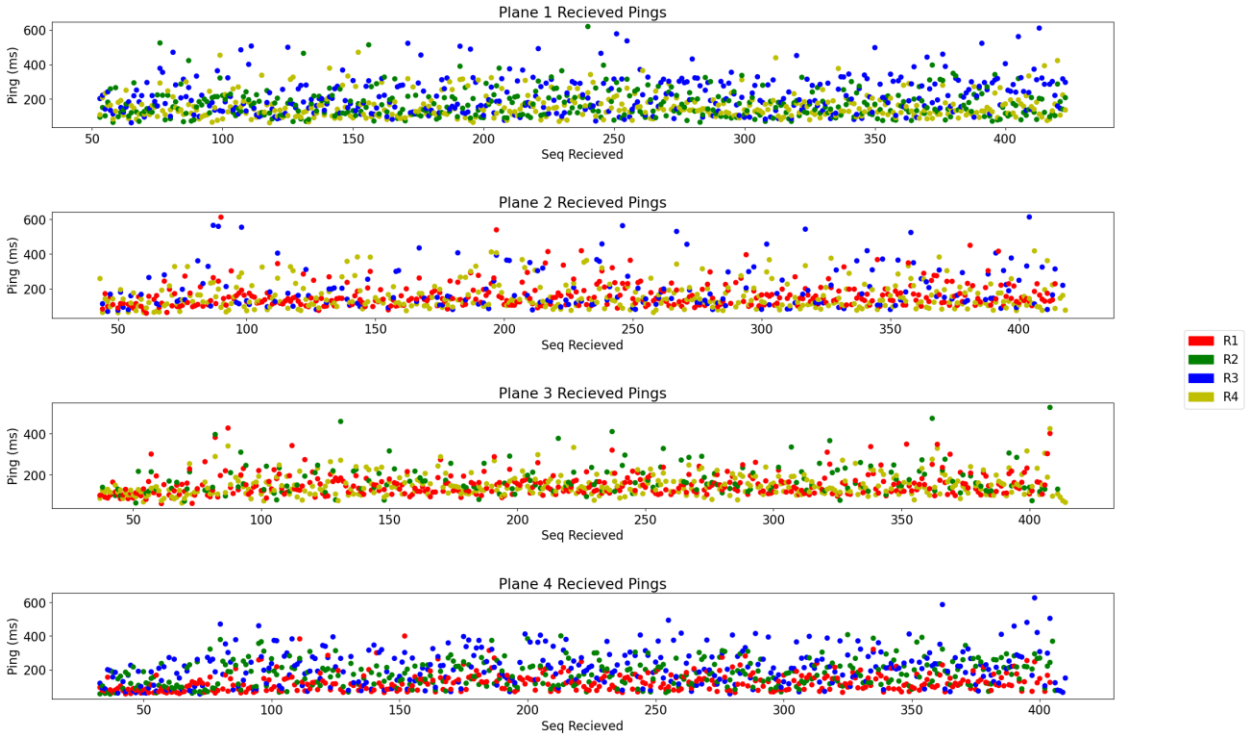

Figure 18: Latency Results for Baseline - IMR Patch

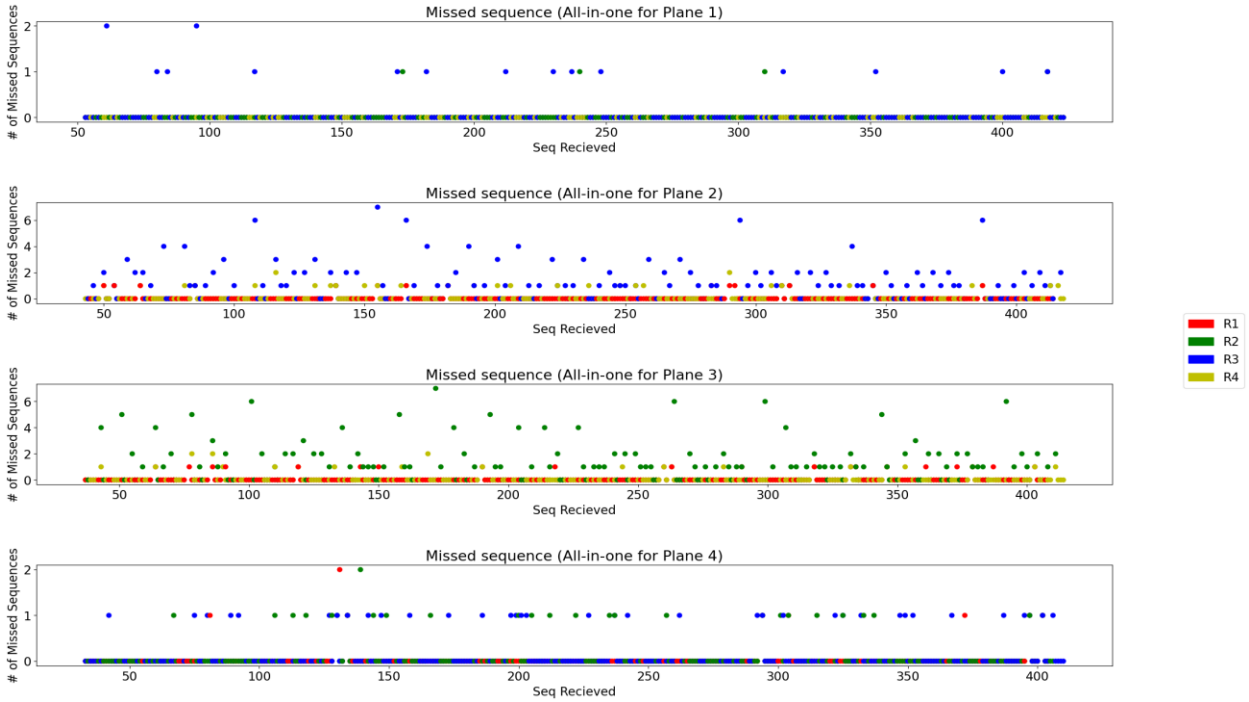

Figure 19: Sequence Loss for Ping Message - IMR Patch

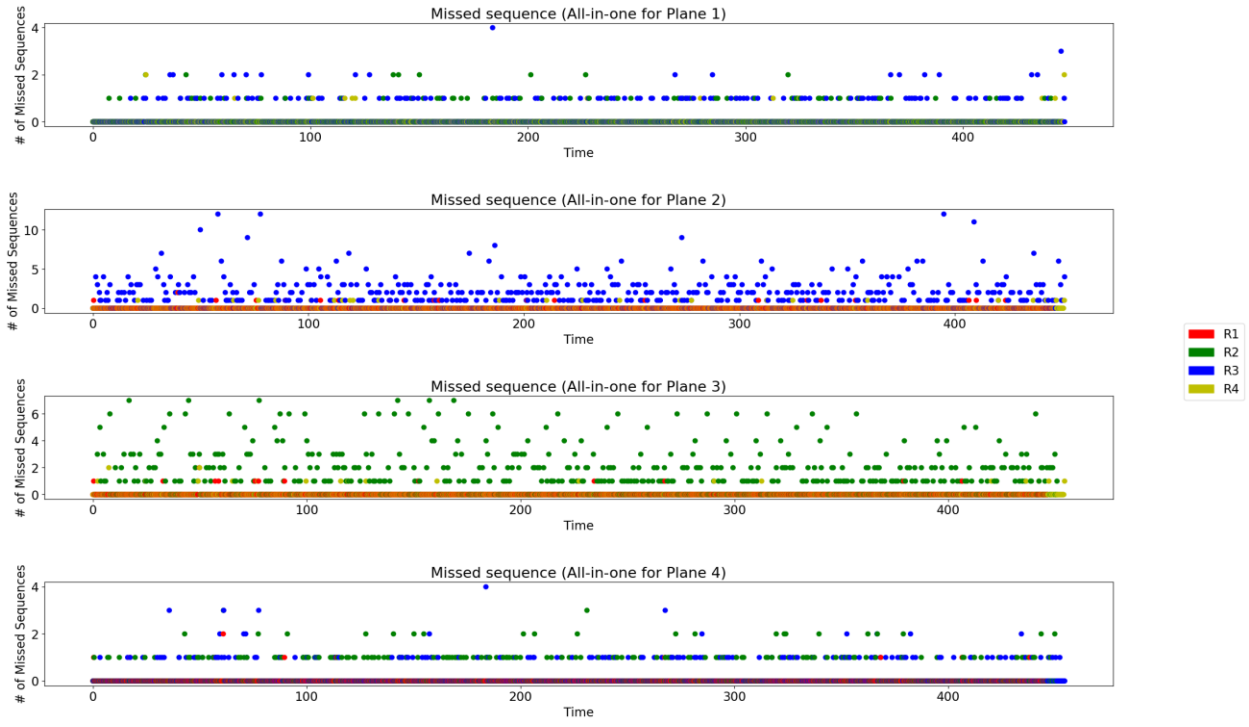

Figure 20: Sequence Loss for Main Data Message - IMR Patch

5.5 Flight Test Data

5.5.1 One Actual, One Virtual Flight (1A1V) Test

The first flight test that is conducted is a two-agent flight test with a ground station. One agent, R1, is an actual aircraft that is being piloted either by a pilot or autopilot. The other agent, R3, is a virtual agent. It is set up to be in HiTL mode and given a mission in the same flight test area as R1. The agents are doing collision avoidance, and by having one virtual agent and one actual agent, the autopilot can be assessed in its ability to avoid without risking a mid-air collision. The ground station is still the master of the mesh communication network.

In Figures 21 and 22 it can be seen that for just two agents, even as one agent is moving around, latency remains under 150 ms on average for each agent. The sequence loss for the main data message also manages to stay under five missed sequences on average, as seen in Figure 23. In the cases where sequence loss goes above ten, it means that waypoint upload is occurring. As mentioned earlier, talking to the ground station takes priority, which can cause significant delays between UASs if something goes wrong. However, the alternative (sending data while doing waypoint upload), usually causes waypoint upload to become difficult.

It should also be noted that the sequence loss for the ping message and the main data message are not strongly correlated to one another. That is to say, R3's sequence loss for the ping message shows quite a few sequences missed while the data message shows only 11 times more than one sequence is lost. This could show that either smaller messages are prone to more loss or that infrequent messages being sent in a message dense system could have a higher chance of being lost. That being said, that is not being explicitly tested for and those conclusions cannot really be drawn from this, but in future tests the ping sequence loss will be excluded

(unless needed) as its main purpose is to find latency, and its sequence loss is not necessary to include.

The last thing that this test shows is that there is a discrepancy in throughput. R1 seems able to readily send to R3 with a 96% success rate, seen in Table 1. However, the inverse is not true and R3 only has an 88% chance to successfully send. This does show a bias within the network, but it is too early to conclude anything meaningful. The next couple of tests will continue to add agents and see if the trend continues.

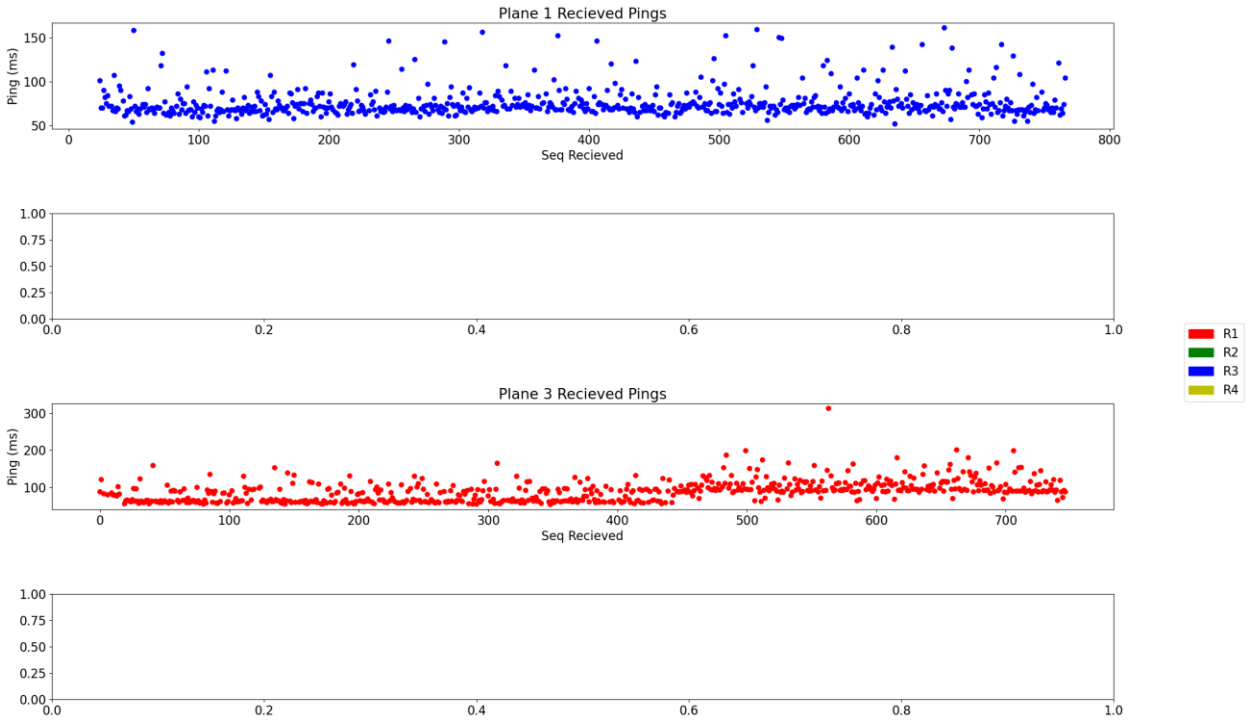

Figure 21: Latency Results - 1A1V Test

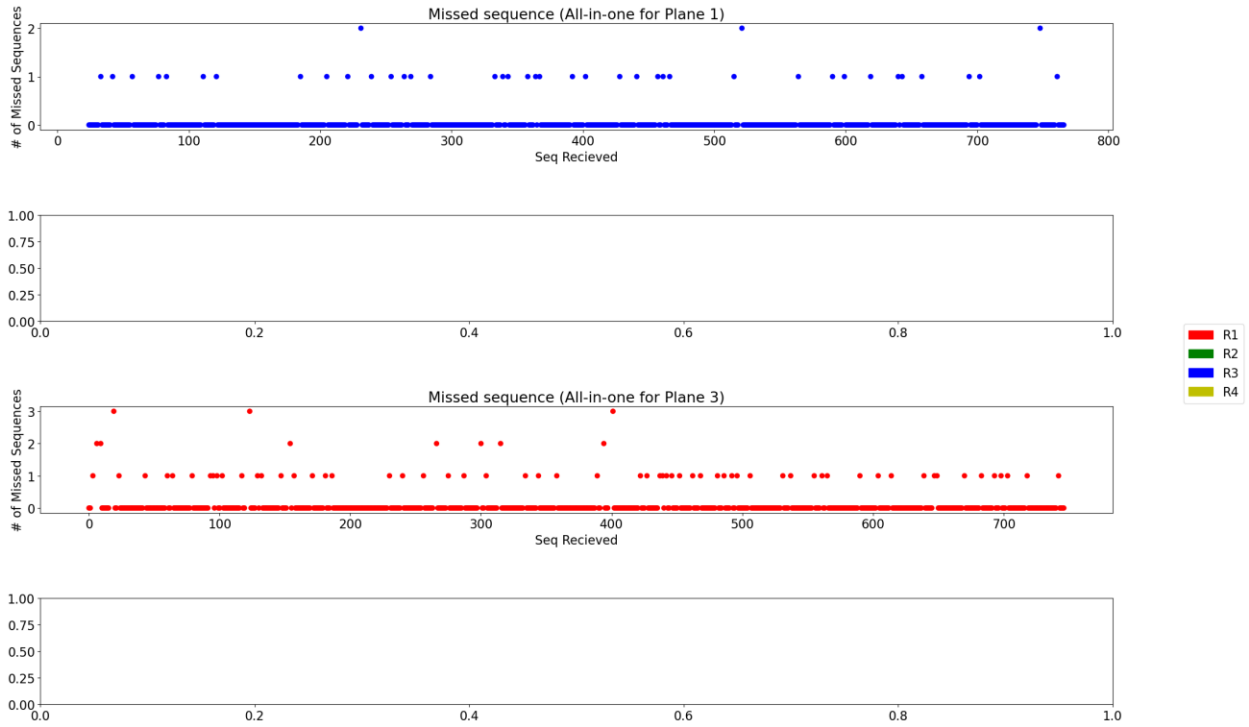

Figure 22: Sequence Loss for Ping Message - 1A1V Test

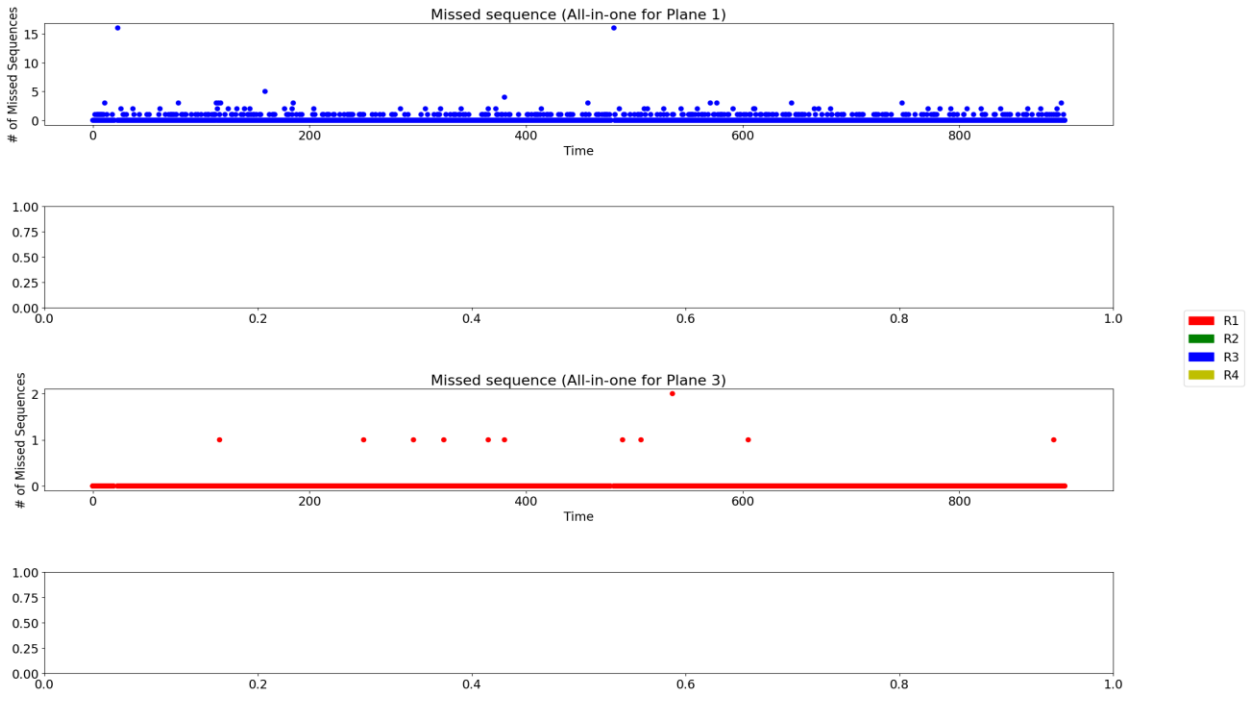

Figure 23: Sequence Loss for Main Data Message - 1A1V Test

	R1 sending to	R3 sending to
R1	100.0	88.87906256909132
R3	96.46497505964	100.0

Table 1: Throughput between both UASs - 1A1V Test

5.5.2 One Actual, Two Virtual (1A2V) Test

The second flight test is conducted in the same way as the first, but instead of one virtual agent there are now two, R4 being the additional virtual agent. The behaviors of all the units are the same, and all will be performing collision avoidance with one another.

The first point to notice of these tests is the ping, shown in Figure 24. Adding an additional agent to the mesh did not significantly alter the latency between the agents, going from ~75 ms latency seen in Figure 21 earlier to ~100 - 150 ms. One thing that increased is the latency fluctuations. This can be seen clearly on R3's latency graph, where R4's latency to R3 can jump from ~20 ms to ~300 ms within a few sequences. At least in this context, using latency to try and adjust information being sent out to other UASs, for other UASs to have accurate location information, is not really viable.

Continuing to the throughput, Table 2 shows that there is still an imbalance between the throughputs to the devices. R3 and R4 both have an equal chance (~85%) to send a successful data message through. However, R1 has a higher chance of sending to R3 and R4 (~98%). At this point it is assumed that since R1 is flying at an altitude of ~350 feet, it has a better chance at transmitting data successfully to the other units. In the next test, another actual UAS will be added to the flight and see if it has the same throughput to R3 and R4.

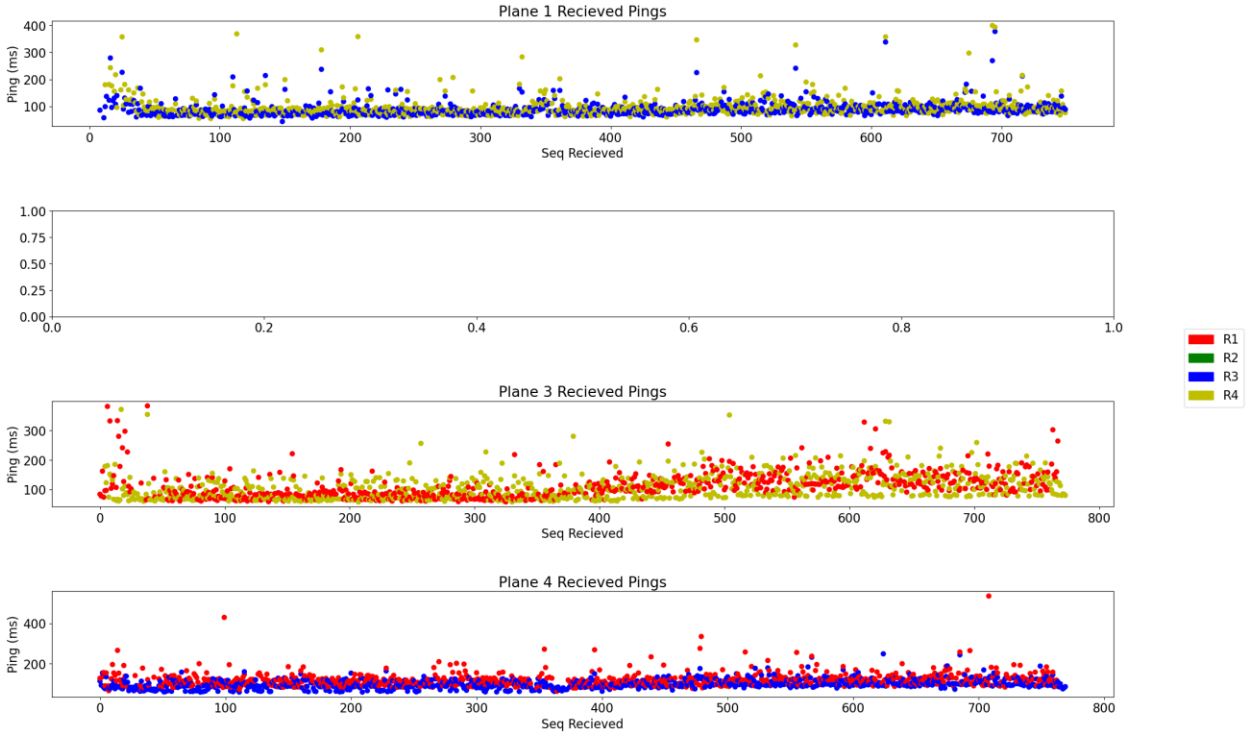

Figure 24: Latency Results - 1A2V Test

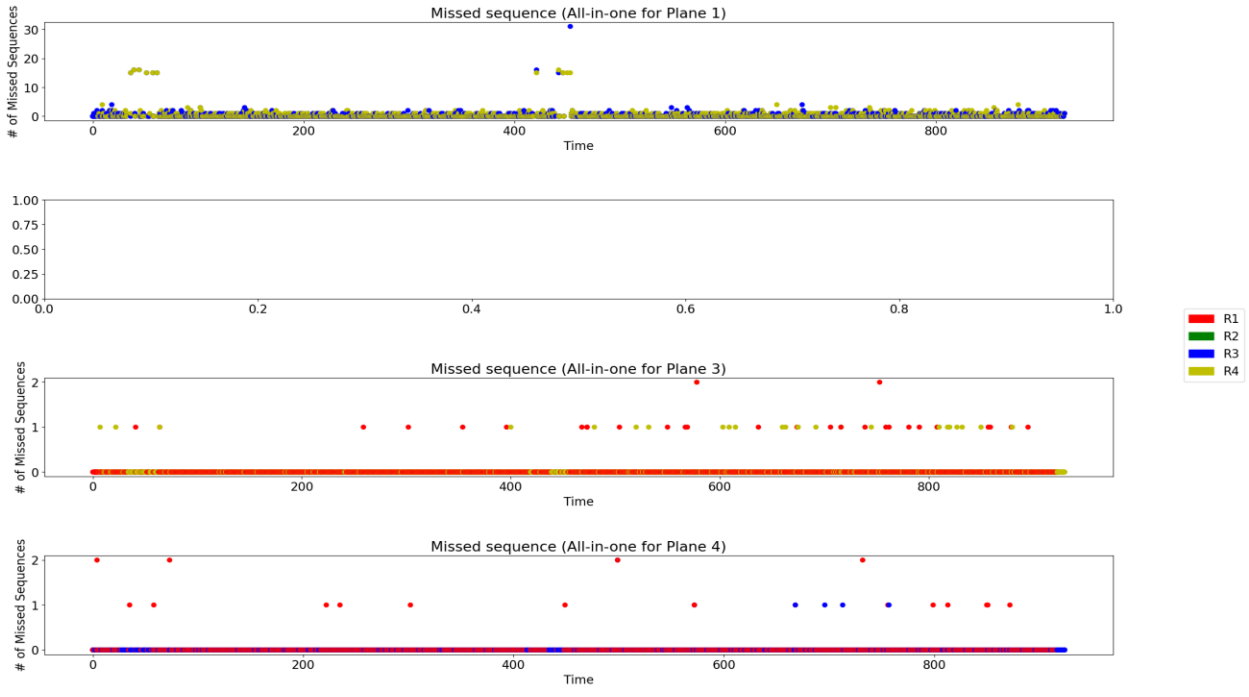

Figure 25: Sequence Loss for Main Data Message - 1A2V Test

	R1 sending to	R3 sending to	R4 sending to
R1	100.0	86.50042992261393	85.00855431993156
R3	98.18181818181819	100.0	98.24636441402909
R4	97.76053215077606	99.20464316423045	100.0

Table 2: Throughput between UASs - 1A2V Test

5.5.3 Two Actual, Two Virtual (2A2V) Test

This flight test remained the same as the last two, but an additional actual UAS is added. To keep the area safe and avoid midair collisions, R2 is flown at a lower altitude than R1. R1 is pushed to ~400 feet, while R2 is set to ~300 feet in the air.

The results of this test are interesting. First, the fluctuating nature of the latency has moved to all units, as seen in Figure 26, and there is more sequences that are lost, as seen in Figure 27. This shows that by adding additional units, the instability of latency between all devices is increased. Second, while the fluctuations are becoming unstable, it seems the additional units are only adding to the latency linearly. At two units, latency is generally less than 75 ms. Three units caused the latency to remain less than 150 ms. Four units seems to keep the majority latency data below 225 ms. Of course, four is too small of a sample size and there is no guarantee that it holds, and it seems like on R2 that it is pretty close to not following that trend. Finally, an interesting note is that these results are like the IMR patch results. This means

for later tests that are conducted indoors, they should hold the same properties as if they are being flight tested.

Throughput, presented in Table 3, is an interesting case. The hypothesis from the last test, that greater altitude can result in greater throughput, seems to not hold. R1 and R2 both send to R3 and R4 exceptionally well, with the inverse being not true. The interesting part is that R1 does not send to R2 as well as it sends to R3 and R4. Additionally, R3 and R4 send to each other much better than R1 and R2 send to each other. R2 is always designated to fly ~100 feet lower than R1 for safety. What that shows, is that a swarm of UASs with varying altitudes could have the throughput affected simply by where an individual unit is in relation to all others. This is not to say altitude is 100 percent at fault. During a flight test many things can go wrong (e.g. loose connections, frayed cables, etc.). In this case, antenna alignment between all the different UASs could explain the drop in throughput. However, it still stands that R4's throughput to R1 dropped ~10% between this test and the last, with the only difference being the addition of R2. This test would need to be reconducted with another UAS positioned at potentially 200 feet to see if it would maintain good throughput to R3 and R4 while having lower quality throughput to R2. The other hypothesis is that motion leads to lower throughput. R1 and R2 are both flying while R3 and R4 are static. Every unit sends well to the static units, but not to the mobile units. Although, with radio waves traveling at the speed of light, it seems less likely that units moving at ~45 ft/s would have that much impact.

These tests and the IMR patch test give us a good baseline for the 900 MHz mesh network and how additional units can affect the network as they are added in. In general, each added unit causes latency to fluctuate more. Additionally, throughput of certain devices starts to drop dramatically depending on their location and behavior. R4's throughput to R1 dropped 8%

by simply adding an additional unit as seen in Tables 2 and 3. It also shows that missed sequences are inevitable and communication with no delay should not be anticipated.

The next section in this test integrates the 2.4 GHz system and tests the latency of that system and sees if it causes any additional delays on the 900 MHz system.

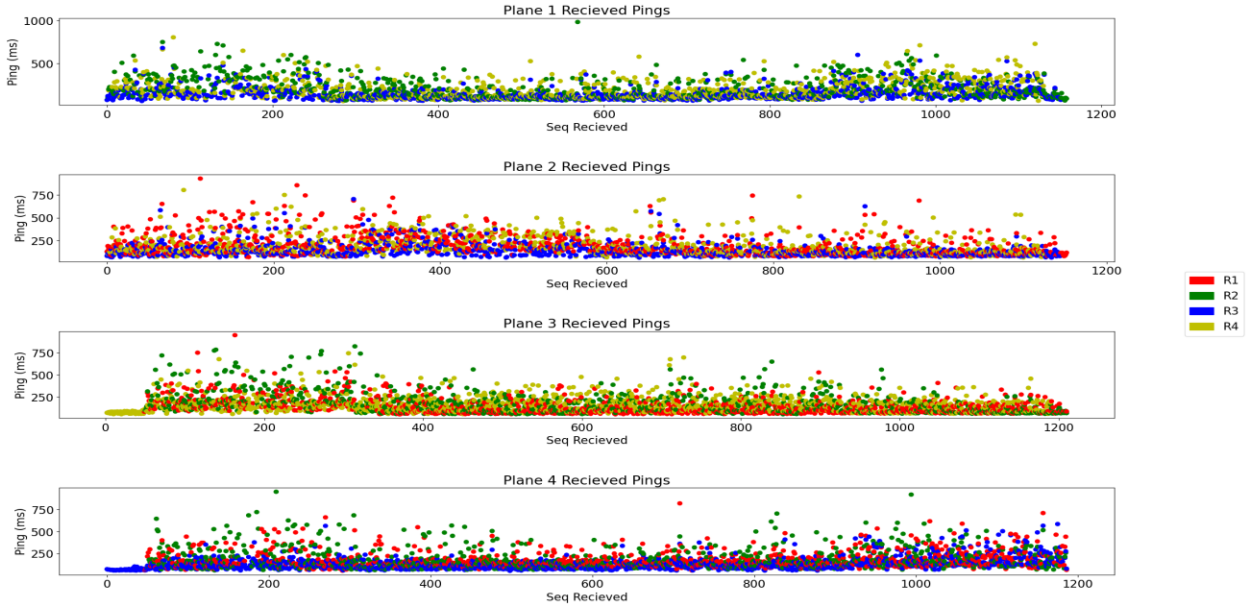

Figure 26: Latency Results - 2A2V Test

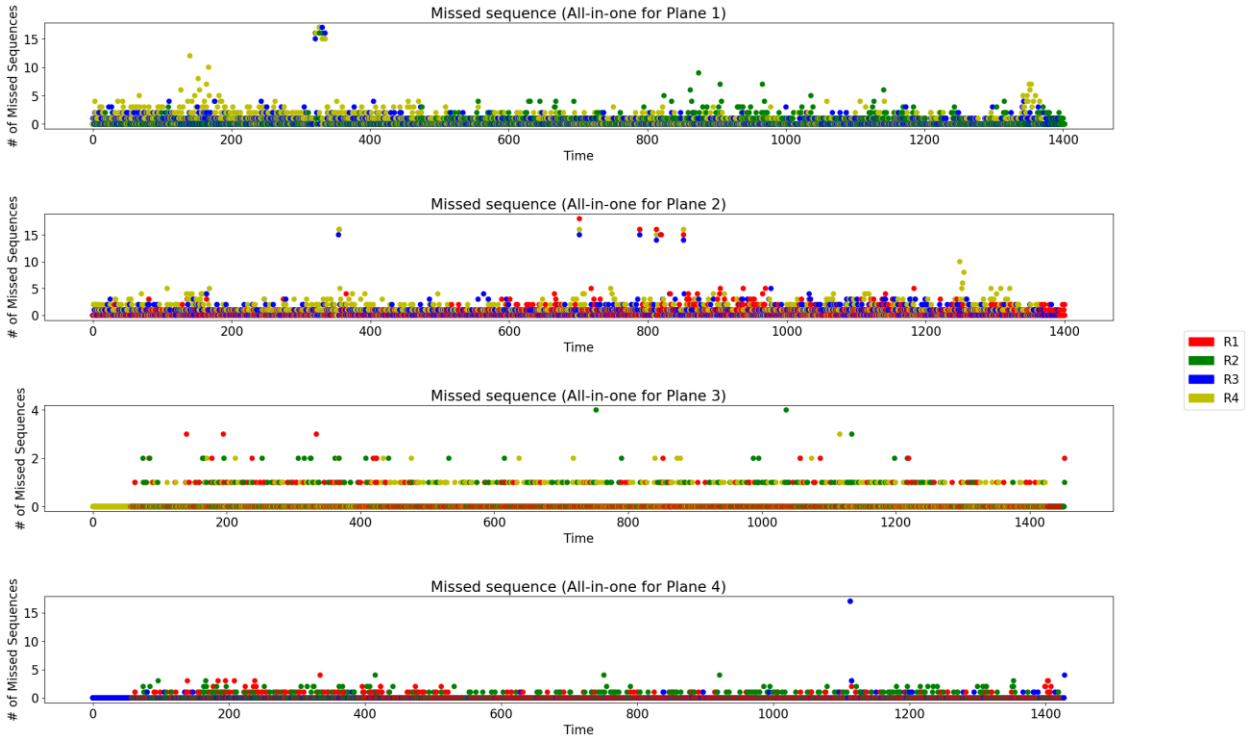

Figure 27: Sequence Loss for Main Data Message - 2A2V Test

	R1 sending to	R2 sending to	R3 sending to	R4 sending to
R1	100.0	87.36462093862815	84.90514160021996	77.4768453550379
R2	86.57890971526685	100.0	80.9595820731372	73.15464496211058
R3	96.85219630848476	95.84115523465704	100.0	96.75834970530451
R4	94.44841894405495	93.12635379061372	97.33296673082211	100.0

Table 3: Throughput between UASs - 2A2V Test

5.6 Dual Frequency Baseline

The first test that will be done is a baseline, where there will be four units. Each unit is given a 900 MHz and a 2.4 GHz. The 2.4 GHz system does not have mesh capabilities, as that was misinterpreted from the unit's description when purchased. Instead, the 2.4 GHz system will be using E2E. The 2.4 GHz units will all be talking to each other, as if they exist all on the same team. There will be one master (R1) and the rest will be endpoints. The ground station will not be connected to the 2.4 GHz system, only the 900 MHz system as the master. The test is then run in the same fashion as the baseline tests. The 2.4 GHz system is sending a smaller packet of ~50 bytes (including additional mavlink formatting) at 5 Hz, as the 2.4 GHz system is proposed as a communication line with minimal data needed between the planes.

This test is conducted to see if the 2.4 GHz radios being close to the 900 MHz radios causes significant interference or not. It also establishes a baseline for the E2E communication protocol for other tests to compare against.

The first thing of note is that the 900 MHz communications do not seem affected by the proximity of the 2.4 GHz system, see Figure 26 and Figure 28. This is expected as they are separated in frequency far enough, but the additional costs internal to the Jetson Nano (processing time, additional ROS messages, etc) could have caused communications to slow down. The second point is the throughput has improved comparatively, see Table 4, to the flight test. However, these units are also all contained within the same room and it could be proximity that causes the improvements, rather than their static nature being the cause. From this point on the 900 MHz information is not included as it remains the same though all dual frequency tests.

On the 2.4 GHz frequency there are some interesting results. For one, the fluctuations that are seen on the 900 MHz are not as prevalent. In Figure 30, R2's latency information seems to have each UAS near a distinct delay. R1 is ~200 ms, while R3 and R4 are closer to ~400 ms. R4 has the highest amount of fluctuation out of all of them, but even inside that R1 seems to maintain a latency ~200 ms. Sequence loss is also remarkable as there are only eight instances of missed sequence information across all UASs. The advantage, it seems, for an E2E system is the reliability and predictability comparatively to a traditional mesh network, with the main disadvantage being the additional bandwidth being consumed by retransmitting all messages through the master unit. One thing that stood out from this, however, is the throughput in Table 5. At first it seemed that the throughput performed marginally better than the 900 MHz units. This contradicts the sequence logging as it shows that very few sequences are missed, which means most packets are delivered successfully. This discrepancy comes from the fact that the units are turned on one-by-one (plugged into power/battery for full startup). The first unit that powers up will be transmitting data that no other unit will receive, and therefore log. This explains the massive difference in throughput communications between R1 and R4, as R1 would be powered on first and powered off first. R4 would be the opposite, being last in every test. Knowing this, the throughput between units that would be turned on next to each other should be the highest. The throughput shows this as R1 to R2 throughput is 95%, R2 to R3 is 94%, and R3 to R4 is 95%. This means that the throughput estimates are conservative in nature and are easily above 95% for what they received. After seeing the results from this test, the next step is to see if two local swarms would cause interference with each other.

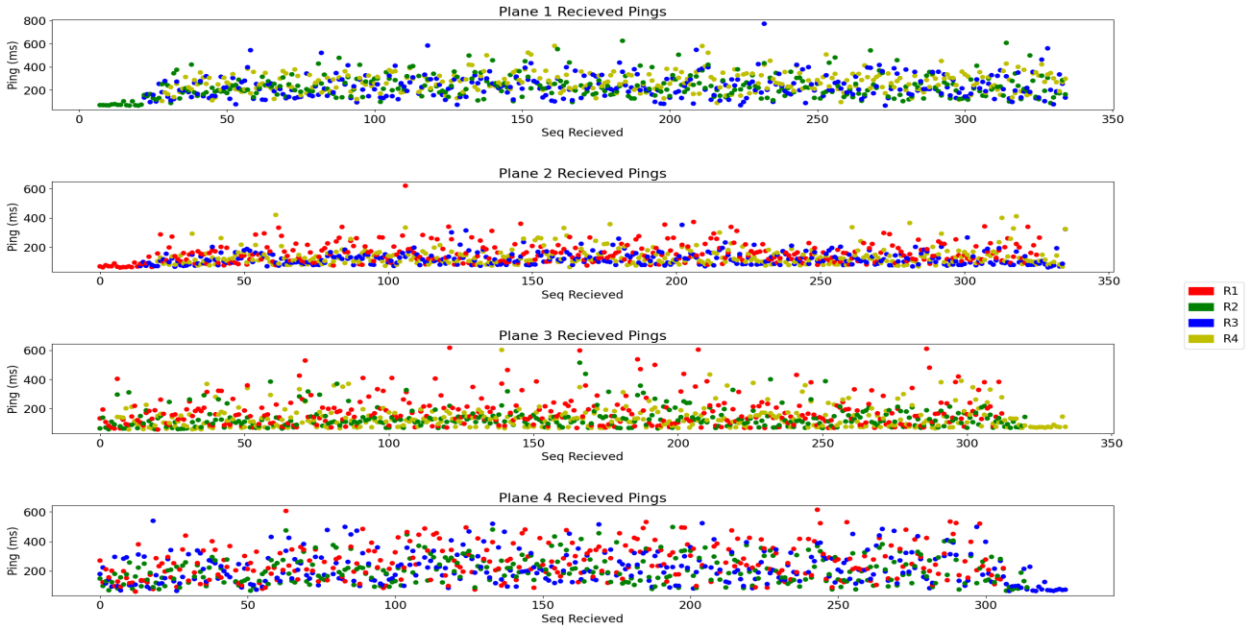

Figure 28: Latency Results - Dual Frequency Baseline - 900 MHz

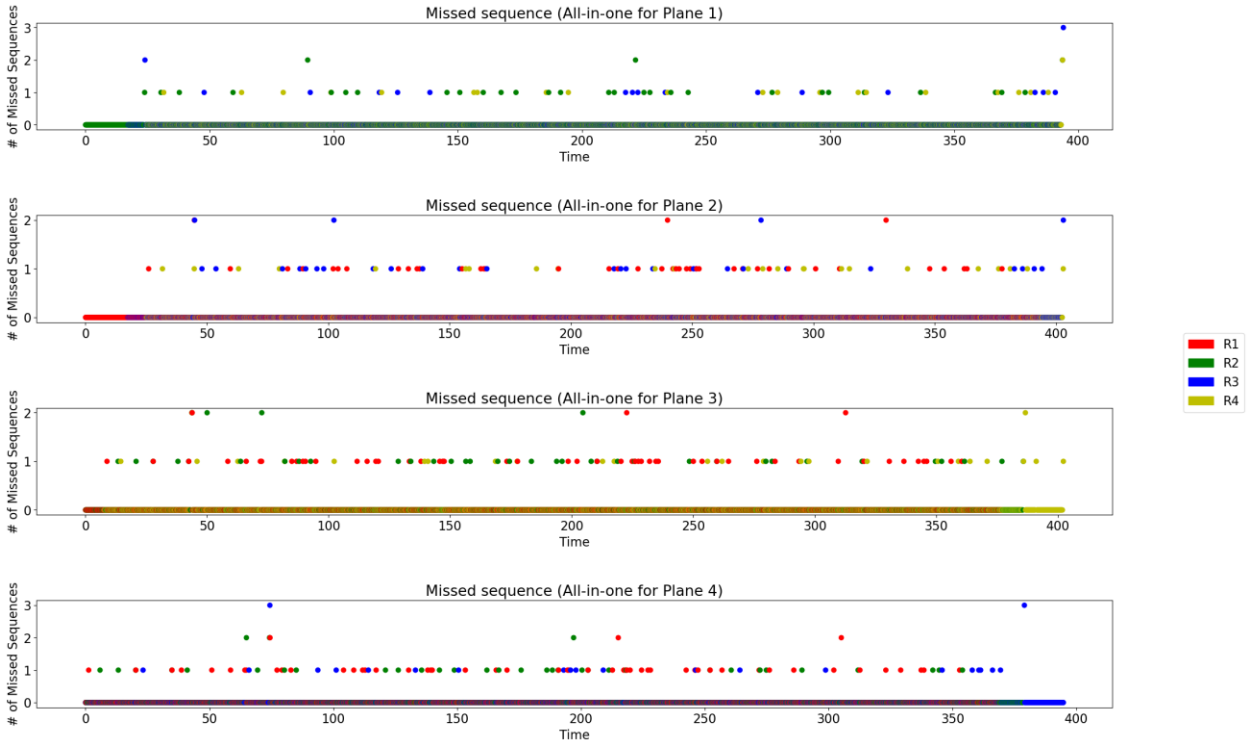

Figure 29: Sequence Loss for Main Data Message - Dual Frequency Baseline - 900 MHz

	R1 sending to	R2 sending to	R3 sending to	R4 sending to
R1	100.0	95.83952451708767	92.50620347394542	89.54344624447718
R2	95.24517087667161	100.0	94.04466501240695	91.65439371624939
R3	90.49034175334324	93.46210995542347	100.0	95.53264604810997
R4	88.55869242199108	91.82763744427935	96.72456575682382	100.0

Table 4: Throughput between UASs - Dual Frequency Baseline - 900 MHz

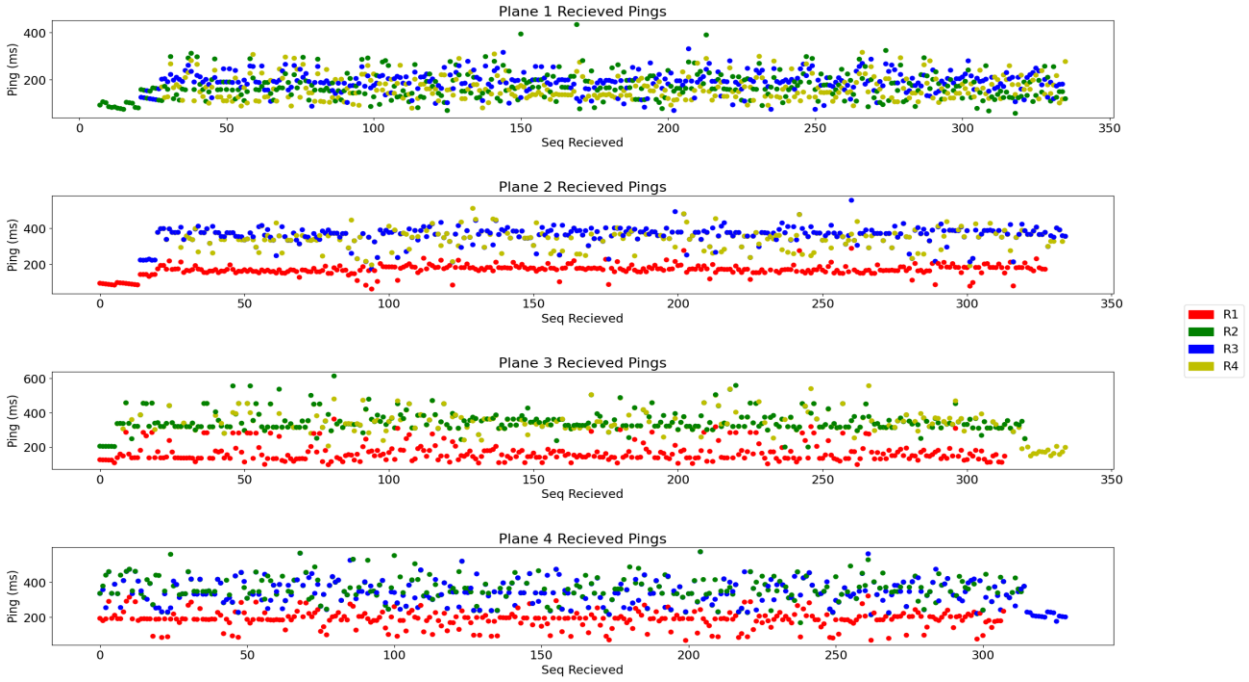

Figure 30: Latency Results - Dual Frequency Baseline - 2.4 GHz

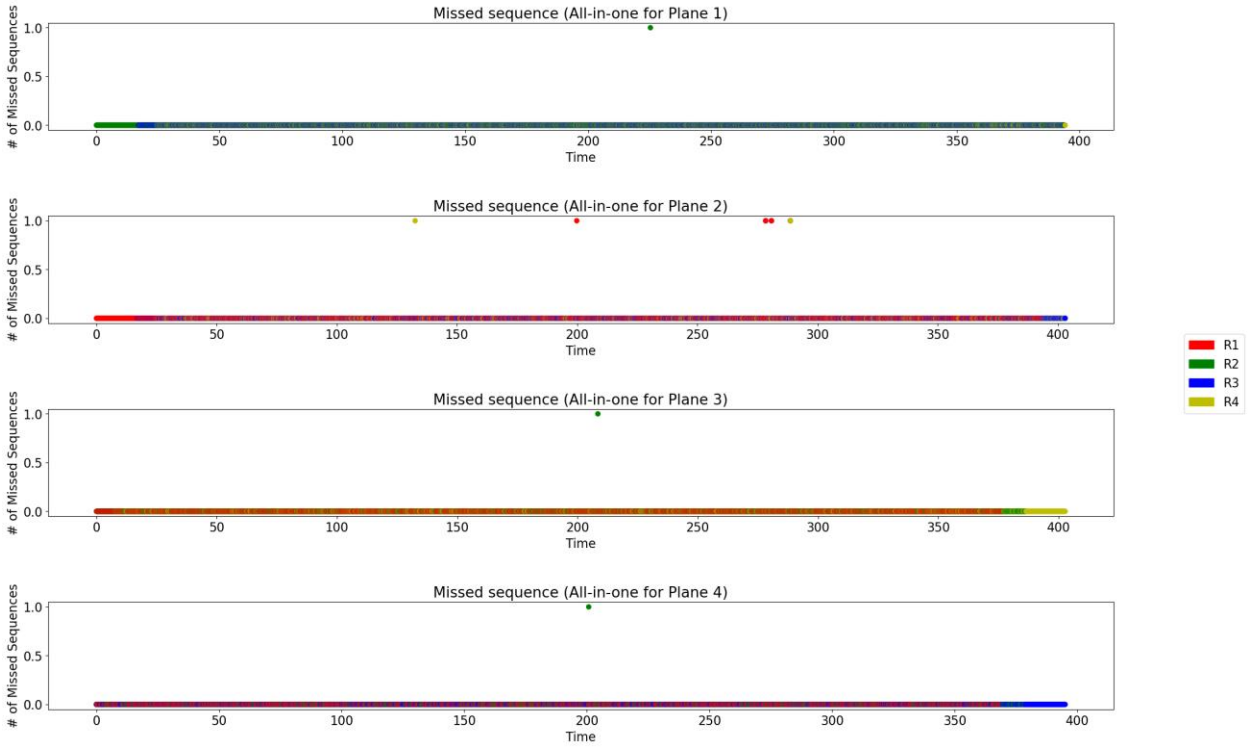

Figure 31: Sequence Loss for Main Data Message - Dual Frequency Baseline - 2.4 GHz

	R1 sending to	R2 sending to	R3 sending to	R4 sending to
R1	100.0	97.66980664353	93.5948361469712	90.76620825147347
R2	97.52107089737233	100.0	95.72989076464746	92.82907662082515
R3	93.50520575111551	95.73624194348042	100.0	97.00392927308448
R4	91.62121963311849	93.85225582548338	98.11320754716981	100.0

Table 5: Throughput between UASs - Dual Frequency Baseline - 2.4 GHz

5.7 Dual Frequency - One Band/Two Network IDs

This test is set up the same way as the first. The main difference is that the 2.4 GHz systems now have two masters (R1 and R3). Each master is given a unique network ID and the endpoints are given the same network ID. Both networks exist on the same band, as in there are no restrictions on what frequency or channels, they can use for transmitting and receiving. This causes the endpoints to only talk to the master with the matching network ID as it has. Effectively, this creates two local swarms that are talking on the same frequency in close proximity to each other. What is being sought is if by doing this, additional units could potentially be added to those local swarms.

The results from this test show a few things. The first is that all latency information dropped to ~75 - 150 ms, as seen in Figure 32. Comparatively to the last test where the latency hovered ~200 ms - 400 ms. Additionally, only two sequences are missed during transmission in all UASs, seen in Figure 33. Throughput also climbs up to 95%-99% in all cases (Tables 6 and 7). This shows that additional units can still be added to each network, although for our tests additional units are not available. Since previous tests showed that four units on the 2.4 GHz system are adequate, in theory, two additional units could be added to each network. At that point, the system would have scaled from four UASs to seven UASs.

However, this does not conclude the testing on the 2.4 GHz system as scalability might be achieved better through splitting the channels, which is the purpose of the next few tests.

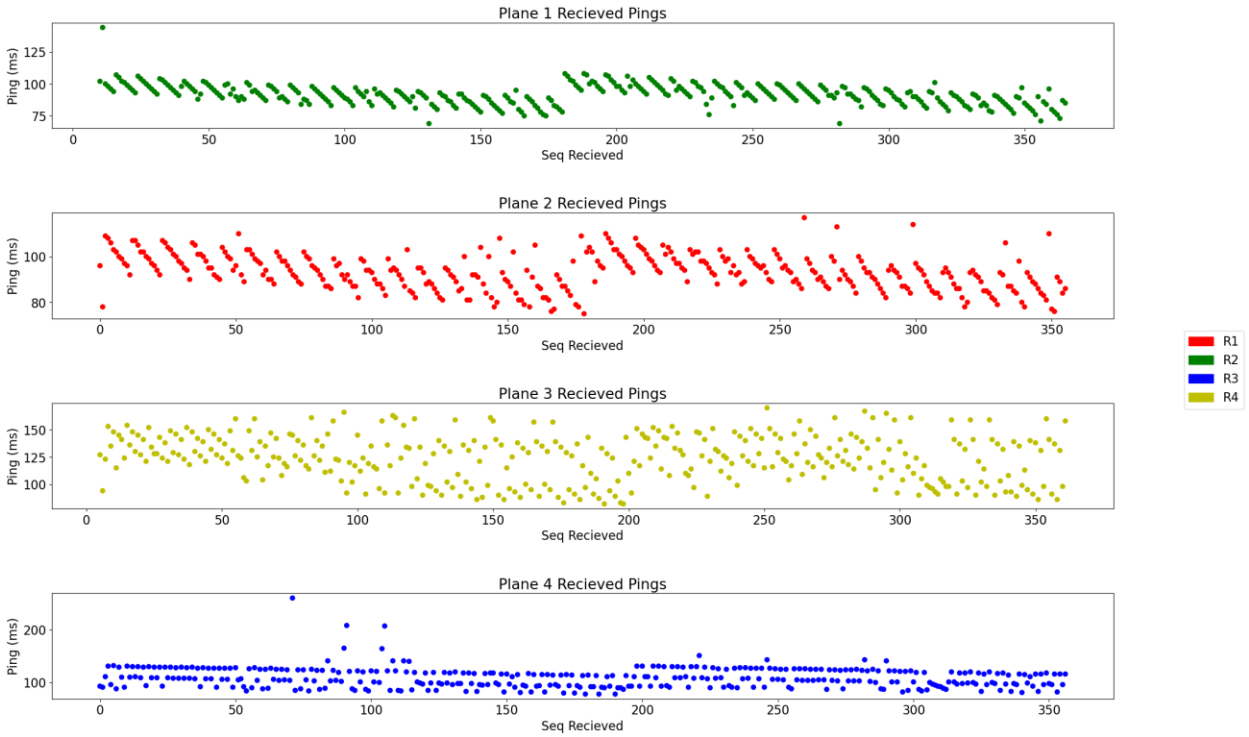

Figure 32: Latency Results - Dual Frequency One Band/Two Network IDs - 2.4 GHz

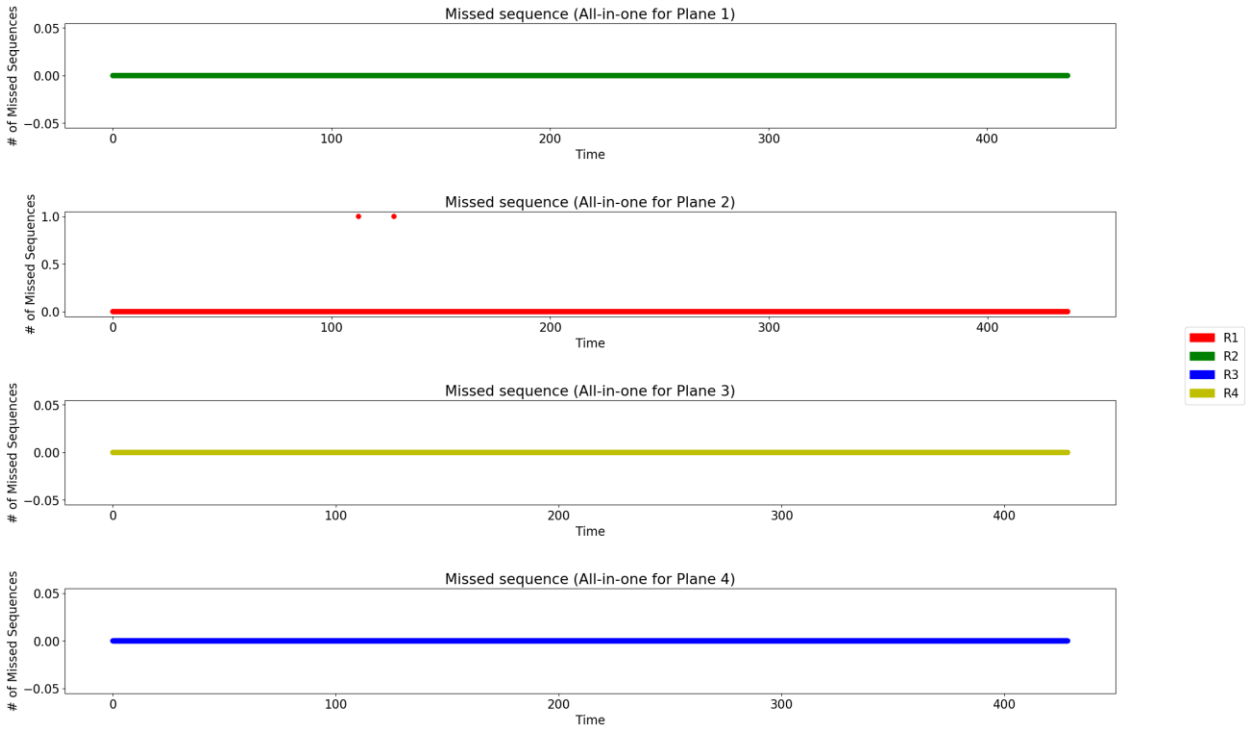

Figure 33: Sequence Loss for Main Data Message - Dual Frequency One Band/Two Network IDs - 2.4 GHz

	R1 sending to	R2 sending to
R1	100.0	99.09255898366605
R2	94.41800086542622	100.0

Table 6: Throughput between R1 and R2- Dual Frequency One Band/Two Network IDs - 2.4 GHz

	R3 sending to	R4 sending to
R3	100.0	98.39302112029384
R4	98.34786599357503	100.0

Table 7: Throughput between R3 and R4- Dual Frequency One Band/Two Network IDs - 2.4 GHz

5.8 Dual Frequency - Two Bands/Two Network IDs

This test is conducted the same as the first dual frequency test. However, instead of both masters being on the same band as one another, each is limited to half of the band that Microhard allowed. The endpoints are limited in the same fashion. The idea being, that being on separate bands, the two communication devices will not cause as much interference as they would if they can use the same channels. By doing this, local swarms can be isolated onto independent bands to not interfere with neighboring local swarms.

The latency, shown in Figure 34, for the test seems to follow the same results as the one band/two network id test. Each UAS latency stays within the same range as the previous test, but the patterns they form are more concentrated (fewer extreme outliers). This is the first test that shows no sequences missing from any of the UASs (Figure 35). Although, it is not a significant

change from the previous test as only two sequences are missing in that test. Nevertheless, the excellent sequence results correlate to the throughput for all systems being near 98%, save R3 as R4 did not start smoothly (Tables 8 and 9).

The results from this test show that separating the frequency into distinct bands can be used to scale independent swarms. By not having local swarms all on the same band, interference can be reduced, and overall throughput can still be maintained. The maximum number of bands the Microhard allow us to create, without excluding too many channels to meet bandwidth constraints, is three. The next tests that follow this squeeze these systems into those smaller bands to see if it causes adverse effects or if separation is still a viable strategy.

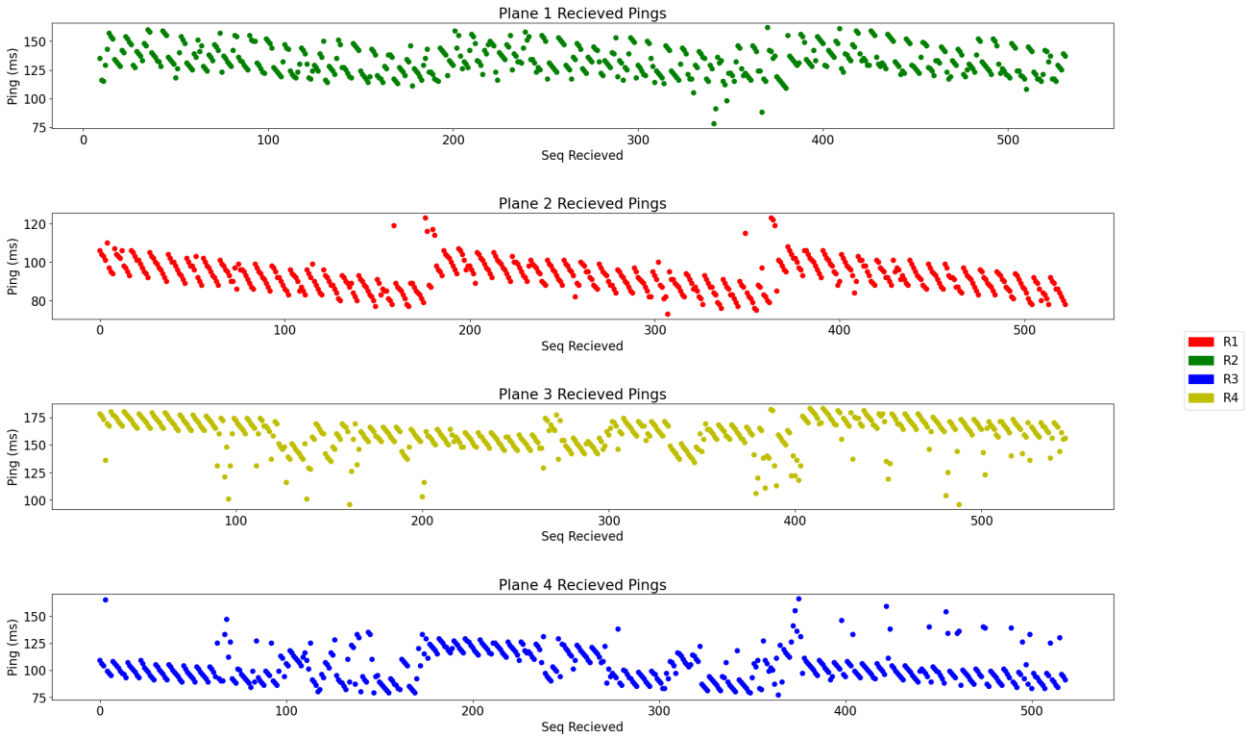

Figure 34: Ping Results - Two Bands/Two Network IDs Test

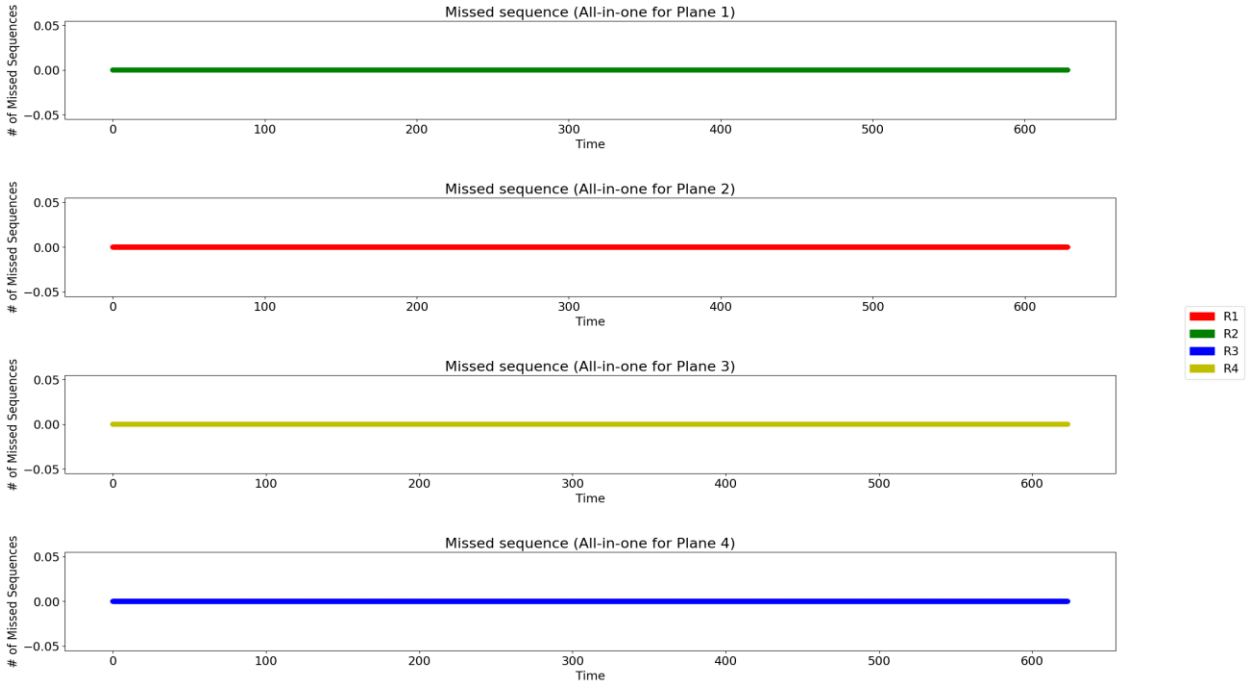

Figure 35: Sequence Loss for Main Data Message - Two Bands/Two Network IDs Test

	R1 sending to	R2 sending to
R1	100.0	99.4305599493831
R2	98.31038798498123	100.0

Table 8: Throughput between R1 and R2 - Two Bands/Two Network IDs Test

	R3 sending to	R4 sending to
R3	100.0	99.01524777636594
R4	94.9725776965265	100.0

Table 9: Throughput between R3 and R4 - Two Bands/Two Network IDs Test

5.9 Dual Frequency - Three Bands/One Network ID

This test is a repeat of the baseline test. The difference is that the 2.4 GHz frequency has been split into its smallest parts (3 distinct bands). The master (R1) and all endpoints are then constrained to one of these bands. The purpose of doing this is to see if reducing the system to using the smallest band available (fewer channels to hop on) will impact the overall performance of the system.

One of the first things that is noticeable in this test compared to the baseline test is the nature of latency. While there are still some deviations, the fluctuation between them has leveled out considerably. In all UAS data, other UASs start to form their own band of latency. For R2's graph in Figure 36, R3 and R4 are both solidly ~225 ms of delay while R1 is ~100 ms. R1's latency graph has the highest amount of fluctuation, but even then, there are noticeable trends for all other UASs. These systems latency is highly predictable in this case and could be used for some applications in the future. Additionally, the sequences lost for the main data message remain about the same, at a total of seven, for all UASs depicted in Figure 37. Throughput, in Table 10 is excellent. All UASs report a throughput of 98% or above.

This data shows that potentially having four UASs on a single 2.4 GHz band could allow us to scale local networks even further. Going from potentially three UASs to twelve UASs, assuming that there are three independent 2.4 GHz bands with four UASs on each band. It should also be noticed that these networks have not hit their upper limit yet, and latency is still well below a two second threshold that is being targeted. This means that for each additional UAS that could be added to a local swarm, every other local swarm can also get another UAS as well.

The next test that will be conducted is if it would be smart to use all available bands on a given frequency, or if adjacent bands cause interference on neighboring bands. The previous test shows that neighboring bands does not cause interference but does keeping an empty band between the networks improve overall performance.

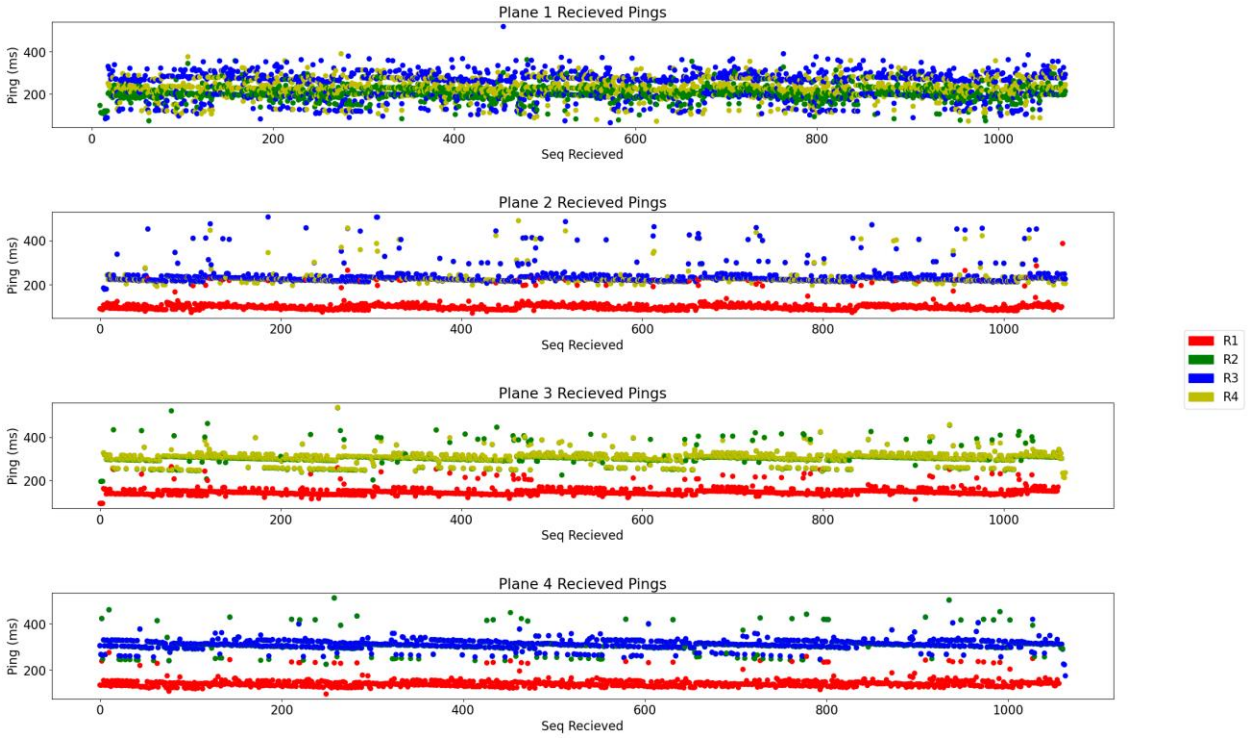

Figure 36: Latency Results - Three Bands/One Network ID Test

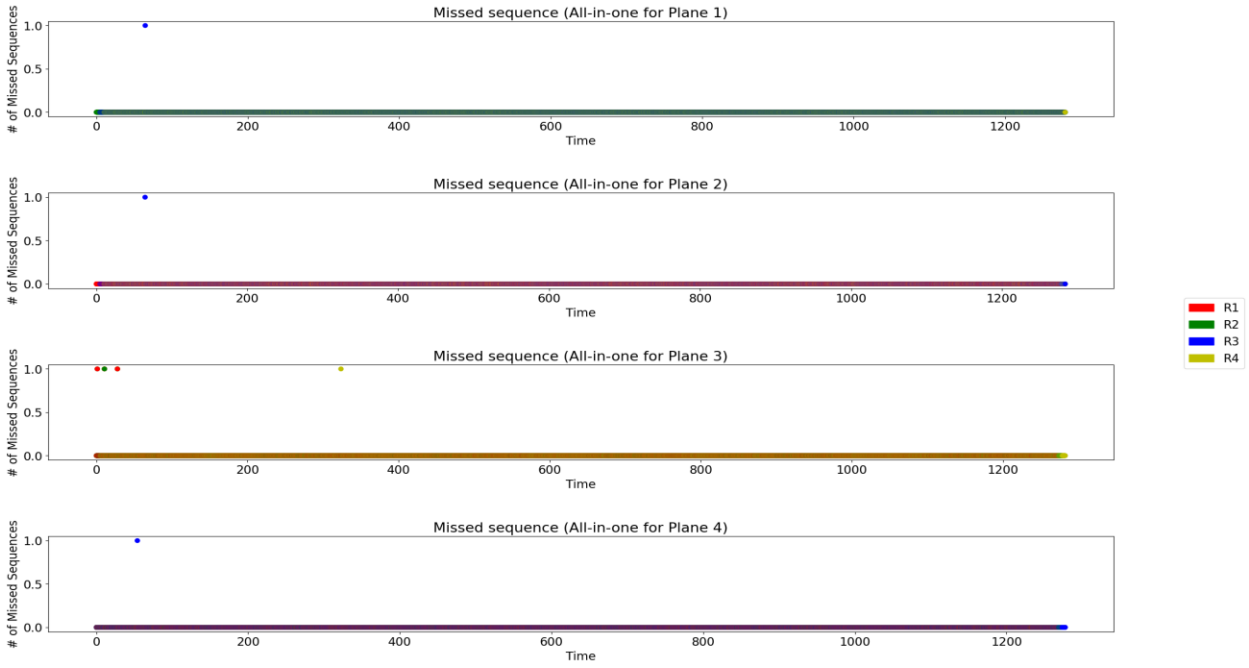

Figure 37: Sequence Loss for Main Data Message - Three Bands/One Network ID Test

	R1 sending to	R2 sending to	R3 sending to	R4 sending to
R1	100.0	99.64174454828661	99.31410756040529	99.01685393258427
R2	99.08612143742255	100.0	99.6726422447389	99.36017478152309
R3	98.65241635687732	99.56386292834891	100.0	99.60986267166042
R4	98.26517967781908	99.17445482866044	99.54793452844895	100.0

Table 10: Throughput between UASs - Three Bands/One Network ID Test

5.10 Dual Frequency - Three Bands/Two Network IDs/Separate Bands

As mentioned previously this test is about seeing if having a band that has no information being transmitted between the bands being used for communication has any noticeable impact on the behavior of the communication systems.

The results show very little difference between this test and the Two Bands/Two Network IDs test. Latency falls in approximately the same ranges with similar behavior as can be seen in Figure 38, there is no loss of sequences in Figure 39, and throughput (Tables 11 and 12) remains high in both cases. This is an expected result as the communication systems performed very well in the Two ands/Two Network IDs test, and any benefits from having an empty channel would be insignificant. Where this test would need to be repeated is when the bands start to hit their capacity. Then the three bands would need to be tested at their capacity and compared to two bands with an empty band between them. The next text will be to see if two network IDs on the same band cause any interference issues.

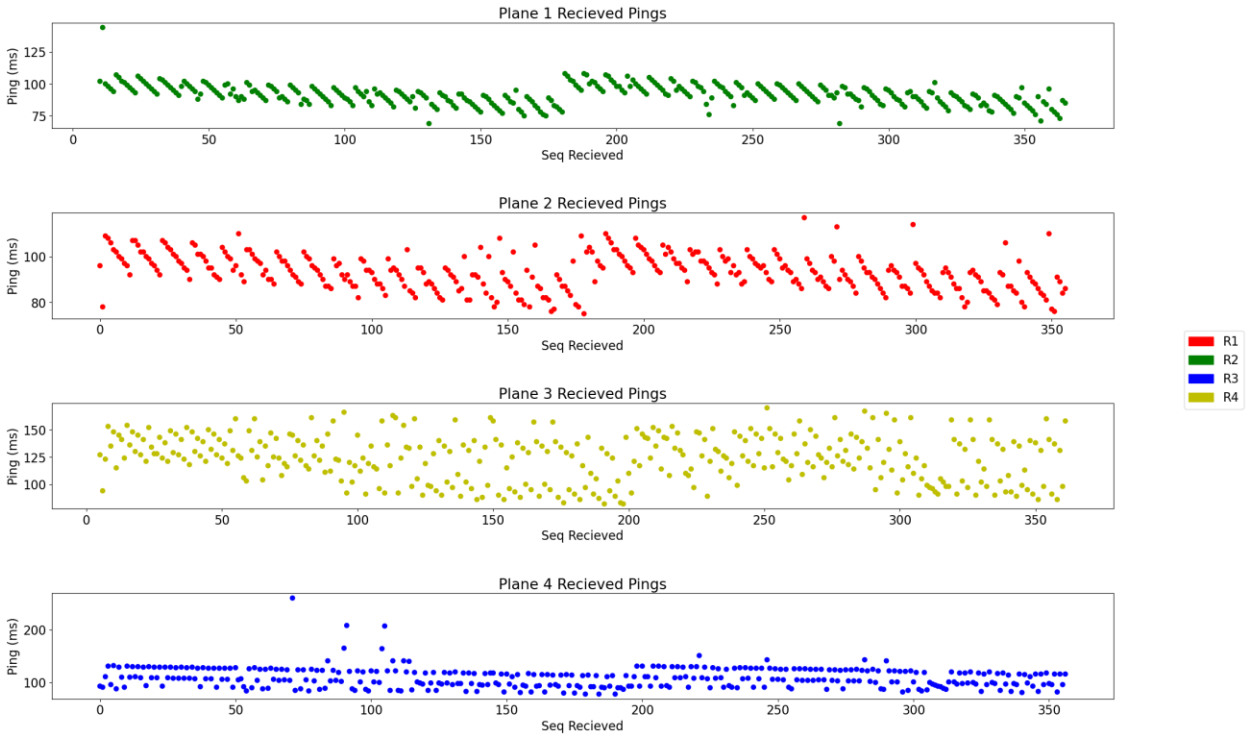

Figure 38: Latency Results - Three Bands/Two Network IDs Test/Separate Bands

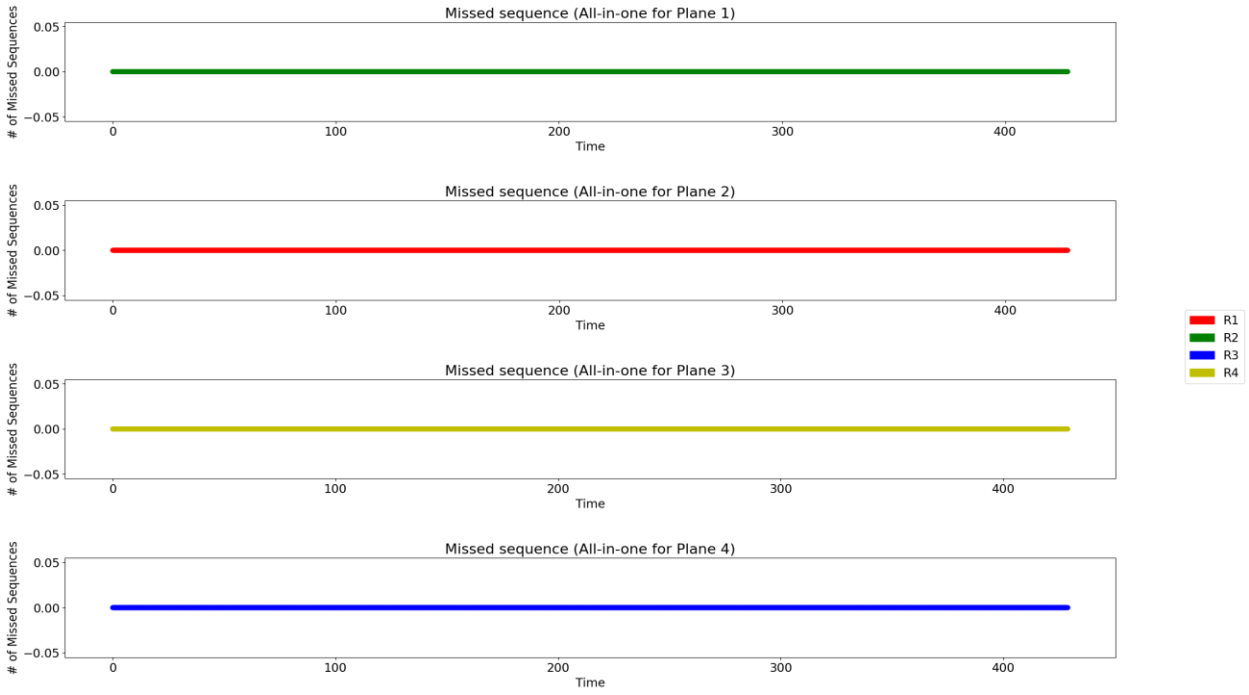

Figure 39: Sequence Loss for Main Data Message - Three Bands/Two Network IDs Test/Separate Bands

	R1 sending to	R2 sending to
R1	100.0	98.66359447004608
R2	97.1843778383288	100.0

Table 11: Throughput between R1 and R2 - Three Bands/Two Network IDs Test

	R3 sending to	R4 sending to
R3	100.0	98.80294659300183
R4	98.62132352941177	100.0

Table 12: Throughput between R3 and R4 - Three Bands/Two Network IDs Test/Separate Bands

5.11 Dual Frequency - Three Bands/Two Network IDs/Same Band

This is the final test and it is designed to try and see if Two Network IDs on the same compact band cause more interference as compared to the One Band/Two Network IDs test. The purpose is to try and see that even if these networks are separated in their own mind, they can still impact each other.

While the latency information looks about the same as compared to the One Band/Two Network IDs test (Figure 40), there are small increases in the latency for R1 and R3. R1 went from an average of ~91 ms latency to ~125 ms latency and R3's average latency went from ~123 ms to ~136 ms latency. R2 and R4 did not seem to be affected as much and R2's average latency went from ~93 ms latency to ~94 ms latency while R4's average latency went from R4's average went from ~109 ms to ~112 ms. The sequence information in Figure 41 also shows an increase in the number of missed sequences. While much of the time no sequences are missed,

comparatively there are significantly more single sequences that are missing. Throughput remains about the same in both cases and can be seen in Tables 13 and 14.

Combining this information together, there are some issues arising from placing two networks onto the same constricted band. Although the differences seem minor, placing an additional unit into a network will potentially increase the latency and sequence loss of another unit in the separate network. This shows, again, that there is a limit to how many units a frequency can hold and reinforces the idea that multiple frequencies used in a hierarchical situation can help scale swarms.

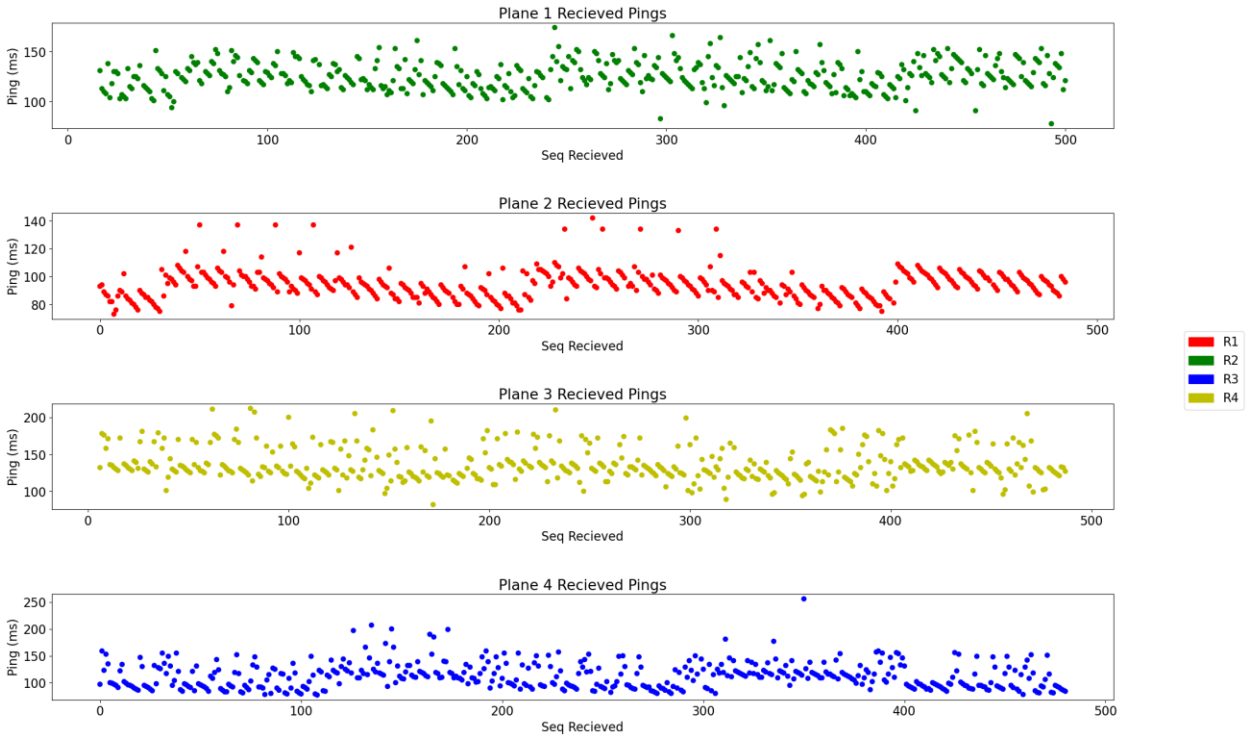

Figure 40: Latency Result - Three Bands/Two Network IDs/Same Band

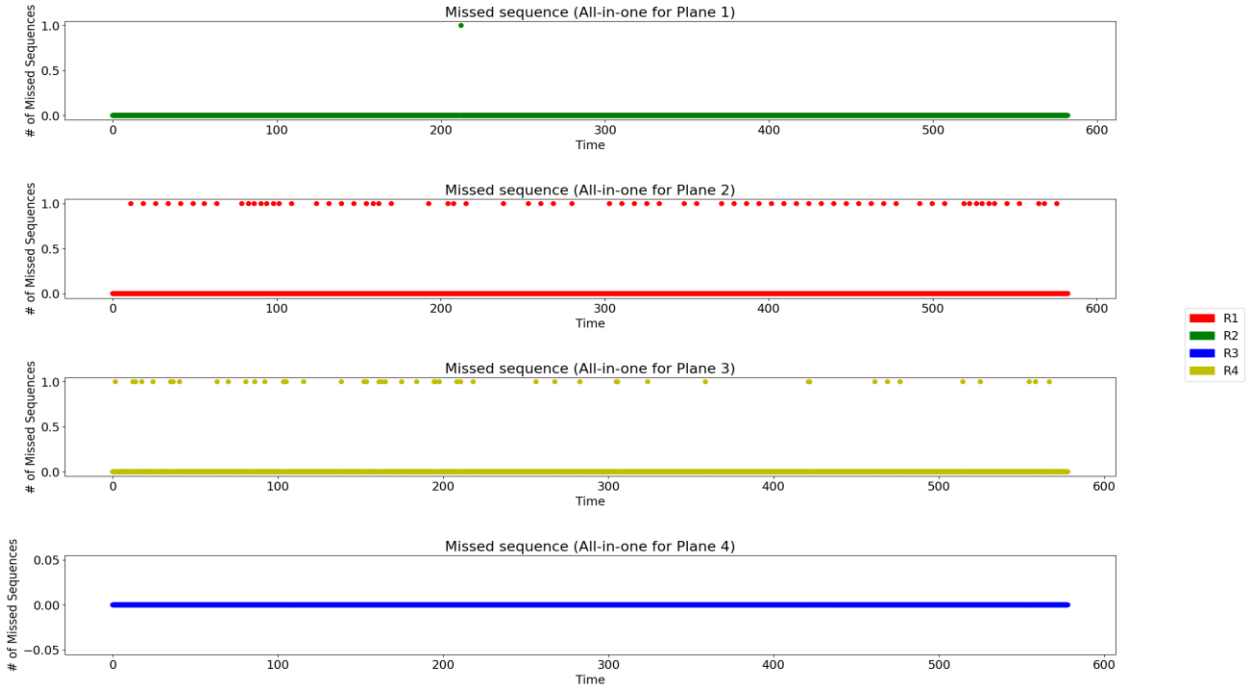

Figure 41: Sequence Loss for Main Data Message - Three Bands/Two Network IDs/Same Band

	R1 sending to	R2 sending to
R1	100.0	99.08100748808714
R2	94.4832170156198	100.0

Table 13: Throughput between R1 and R2 - Three Bands/Two Network IDs/Same Band

	R3 sending to	R4 sending to
R3	100.0	97.16336295283664
R4	98.63527806209484	100.0

Table 14: Throughput between R3 and R4 - Three Bands/Two Network IDs/Same Band

5.12 Discussion

The significance of this work shows that communications within a swarm of UASs is important. When increasing the number of agents during the flight tests, latency went up and the sporadic nature of that latency also increased. This shows that adding additional units to a network, regardless of their relative position to other units, can still negatively affect other units. On top of this, in the pre-IMR patch tests, it is shown that a network will eventually hit a limit in the amount of information that can be successfully sent. Both facts show that a single frequency has limitations when it comes to the scalability for a swarm. However, when using two frequencies, the tests show that local swarms perform better than when all UASs are on a single frequency. Comparing the results from the 2A2V flight test, Figures 5.18 and 5.19, to the "Dual Frequency - Three Band/Two Network IDs/Separate Band" test, Figures 5.30 and 5.31, a clear distinction is drawn. In the latter tests, the latency is lower and more stable and the sequence loss for the main data message is zero. Stable latency means that information from one UAS to another can be modified to account for the delay more accurately. Additionally, there were no outright negative side effects shown when using band splitting compared to the single band tests. This shows that band splitting could be used to both scale the number of agents and prevent local swarms from causing interference with one another. Finally, since the 2.4 GHz system is using E2E, while the 900 MHz was using mesh, this work showcases the ability of the hierarchical system working with different communication protocols. The area a local swarm is responsible for decreases the lower the swarm is on the hierarchy. While ad-hoc is useful for swarms where agents might disconnect from the network, in smaller areas it loses its advantages. Other protocols, that have higher throughput, should not be ignored as viable communication protocols.

6 Future Work

This research brought forth many different ideas. The first and foremost is the idea of connecting information up the hierarchy in a flight test situation. Early on, it became abundantly clear that it would be difficult to design an intelligent hierarchy for distributing information in between layers of the hierarchy. There needs to be a way to track if a mission is successfully distributed to lower layers. Each layer needs protections in case a UAS does crash/disconnects, or to communicate to other local swarms of the rouge UAS. If the master of a local swarm crashes, it needs to be replaced by another unit. Allocation of UASs being added to the hierarchy has to be accounted for and automated. The ability to move UASs to different local swarms or different hierarchy layers is needed for the adaptability of the swarm. These and many other additional protocols to follow in case something goes wrong or if additional actions need to be performed.

Additionally, the ideas presented in this should be retested using only ad-hoc connections. This is mostly to look and see how those networks scale in comparison to the ones Microhard allowed. It could be possible that those networks perform better or worse and it would be an interesting comparison to make and would link back to studies done in the UAS field.

Another idea that arose during all of this is using grid-based allocation for the local swarms. Essentially, given an area where the full swarm needs to search, it will be broken down into smaller sections and those areas are where the local swarms should be searching. The twist to this idea is that the grids can also allocate which channels the local swarms will be talking on. Ideally, two local swarms that are allocated to the same channel do not sit next to each other on the grid. This is because the communication in one local swarm could cause interference in the other if it is close enough. This could be limited by dynamically changing the power output of

the radios midflight. The Microhards do allow for a radio to be reconfigured while it is powered on, but that might pose a safety risk and should be tested.

Security is also a concern when it comes to swarms of UASs, especially in this instance where masters are responsible for multiple aircraft in a local swarm. These masters would be valuable targets as they are higher up in the hierachy. This could be solved using ad-hoc, as at that point there are no network masters, but there would still be one point that has access to the upper layers of the hierarchy. Certain protocols could shift the master of the local swarm, but then an adversary could simply mimic a local UAS and receive the position. While security was not the focus of this research, its impacts into the overall scalability of a swarm should be further researched.

Finally, the ideas presented should be tested with a higher number of UASs. At the time, only four could really be made and tested. Ideally, the number of UASs are slowly added until the system has too much information to handle. The thing that will most likely hit the limit first is the rate at which the CTS could be sent, in the RTS/CTS transaction, not necessarily bandwidth constraints. When testing what rate to use for the main data message in the UASs, the ground station could only ever report ~35 Hz total for all systems. For example, R1 and R2 could fluctuate between ~15 - 20 Hz, but the total of the two would always be near ~35 Hz.

7 Conclusion

This work sought to address communication scalability within a swarm of UASs. By using two frequencies, it is shown that better scalability is achieved. While scalability has been addressed in the form of GNC algorithms, many papers have not included communication delays and throughput into their considerations. After testing various configurations including changes in swarm size, mission, and formations, it became clear that communication scalability has a major impact on larger swarms. Each additional unit added will impact the overall communications of every other unit, even if nothing about those units physically changes. Each unit added, while still having an overall impact in their local swarm, will hit a limit as each swarm will have a capacity on the number of units before an additional local swarm is created. By doing this, the latency on the lower rungs will decrease and throughput will increase dramatically. This method could also prevent instability within a larger swarm, since the center of the local swarm will not need to route as much information comparatively to a single frequency swarm. This scalability is pushed even further by splitting the lower rungs of the hierarchy over different channels.

The significance of this work is twofold. First, larger swarms are possible while still maintaining latency and throughput requirements. This is critical for these larger swarms, in applications such as collaborative sensing, as higher latency causes instability within the swarm, leading to catastrophic failures. Additionally, missions like collision avoidance require lower latency to ensure all distance constraints are preserved, for each agent. Secondly, communication and the interaction it imparts on swarms of UASs cannot be ignored, especially when dealing with claims of scalability. It has been repeatedly shown that additional units will eventually

cause unacceptable throughput loss or latency. Future claims of scalability within a swarm environment should clearly state the communication requirements as needed per application.

8 References

[1] Intel, "Shine Bright With Intel Drone Light Shows," [Online]. Available: https://www.intel.com/content/www/us/en/technology-innovation/aerial-technology-light-show.html.

[2] Intel, "Intel Shooting Star System," [Online]. Available: https://www.intel.com/content/www/us/en/technology-innovation/shooting-star-system.html.

[3] W. Heinzelman, A. Chandrakasan and H. Balakrishnan, "Energy-efficient communication protocol for wireless microsensor networks," *Proceedings of the 33rd Annual Hawaii International Conference on System Sciences,* vol. 2, p. 10, 2000.

[4] S. Khuller and S. Banerjee, "A clustering scheme for hierarchical control in multi-hop wireless networks," *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213),* vol. 2, pp. 1028-1037, 2001.

[5] R. Ramanathan and M. Streenstrup, "Hierarchically-organized, multihop mobile wireless networks for quality-of-service support," *Mobile Networks and Applications,* vol. 3, pp. 101-109, 1998.

[6] F. Xiangning and S. Ylin, "Improvement on LEACH Protocol of Wireless Sensor Network," *International Conference on Sensor Technologies and Applications (SENSORCOMM 2007),* pp. 260-264, 2007.

[7] C. Perkins and E. Royer, "Ad-hoc on-demand distance vector routing," *Proceedings WMCSA'99. Second IEEE Workshop on Mobile Computing Systems and Applications,* pp. 90-100, 1999.

[8] D. Johnson and M. D.A., "Dynamic Source Routing in Ad Hoc Wireless Networks," *Mobile Computing,* vol. 353, 1996.

[9] W. Wollman and Y. Barsoum, "Overview of open shortest path first, version 2 (OSPF V2) routing in the tactical environment," *Proceedings of MILCOM,* vol. 3, pp. 925-930, 1995.

[10] Bekmezci, Ilker, Ozgur Koray Sahingoz, and Şamil Temel. "Flying ad-hoc networks (FANETs): A survey." *Ad Hoc Networks* 11.3 (2013): 1254-1270.

[11] Clark, Justin, and Rafael Fierro. "Cooperative hybrid control of robotic sensors for perimeter detection and tracking." *Proceedings of the 2005, American Control Conference, 2005.*. IEEE, 2005.

[12] Jevtic, Aleksandar, et al. "Distributed bees algorithm for task allocation in swarm of robots." *IEEE Systems Journal* 6.2 (2011): 296-304.

[13] Blender, Timo, et al. "Managing a mobile agricultural robot swarm for a seeding task." *IECON 2016-42nd Annual Conference of the IEEE Industrial Electronics Society*. IEEE, 2016.

[14] Wang, Yiheng, Alei Liang, and Haibing Guan. "Frontier-based multi-robot map exploration using particle swarm optimization." *2011 IEEE symposium on Swarm intelligence*. IEEE, 2011.

[15] Costa, Vasco, et al. "Design and development of an inexpensive aquatic swarm robotics system." *OCEANS 2016-Shanghai*. IEEE, 2016.

[16] Vásárhelyi, Gábor, et al. "Outdoor flocking and formation flight with autonomous aerial robots." *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*. IEEE, 2014.

[17] Dasgupta, Prithviraj. "A multiagent swarming system for distributed automatic target recognition using unmanned aerial vehicles." *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans* 38.3 (2008): 549-563.

[18] Barnes, Laura, et al. "Swarm formation control with potential fields formed by bivariate normal functions." *2006 14th Mediterranean Conference on Control and Automation*. IEEE, 2006.

[19] A. R. Girard, A. S. Howell and J. K. Hedrick, "Border patrol and surveillance missions using multiple unmanned air vehicles," *IEEE Conference on Decision and Control,* vol. 1, pp. 620-625, 2004.

[20] A. Williams, S. Glavaski and T. Samad, "Formations of formations: hierarchy and stability," *American Control Conference,* vol. 4, pp. 2992-2997, 2004.

[21] Ronchieri, Elisabetta, and Mario Innocenti. "Decentralized control of a swarm of unmanned aerial vehicles." *AIAA Guidance, Navigation and Control Conference and Exhibit*. 2007.

[22] Van Gijseghem, Wouter, and Umang Agarwal. "Drone Delivery Multi-Agent Routing Optimization." *AIAA AVIATION 2020 FORUM*. 2020.

[23] Rughani, Rahul, and David Barnhart. "Using Genetic Algorithms for Safe Swarm Trajectory Optimization." *AIAA Scitech 2020 Forum*. 2020

[24] *Integrated Path Planning - Collision Avoidance System for UAVs Operating In Obstacle Rich Environment*. doi:10.2514/6.2020-3207. Accessed 3 2021.

[25] Morgan, Daniel, et al. "Swarm assignment and trajectory optimization using variable-swarm, distributed auction assignment and sequential convex programming." *The International Journal of Robotics Research* 35.10 (2016): 1261-1285.

[26] Cramer, Nick B., et al. "Enhanced UAS Availability via Vehicle to Vehicle Routing Scaled Experiments." *AIAA AVIATION 2020 FORUM*. 2020.

[27] *Flight Test Validation of a Safety-Critical Neural Network Based Longitudinal Controller for a Fixed-Wing UAS*. doi:10.2514/6.2020-3093. Accessed 21 2021.

[28] L. Meier, "Mavlink," Github, 2 April 2010. [Online]. Available: https://github.com/mavlink/mavlink/commit/a087528b8146ddad17e9f39c1dd0c1353e5991d5.

[29] Microhard, "P900 - Miniature Mesh 1W 900 MHz Wireless Modem," Microhard, [Online]. Available: http://www.microhardcorp.com/P900.php.

[30] Microhard, "P2400 - Miniature 1W 2.4 GHz Wireless Modem," Microhard, [Online]. Available: http://www.microhardcorp.com/P2400.php.

[31] Intel, "MAVLink Router," Intel, [Online]. Available: https://github.com/intel/mavlink-router.