SENG402 Drone Formation Flying Final Report

Euan Morgan — 77271035

***Abstract*—In this paper, we propose a solution for operating drones in a rigid formation and explore the design of the Data Dissemination Protocol used for formation communication. There is a vast amount of research for drone formation applications, but these formations are often simulated during evaluation. This project aims to fill this gap in research and provide a rigid drone formation for use as a platform to be extended to meet the requirements of many applications. The formation is designed with a focus on reliability, efficiency, and adaptability to provide a generalized solution. This paper focuses on the design and implementation of the Data Dissemination Protocol which provides communication for the drones in the formation and manages movement data to ensure safety systems can avoid drone collisions. The project is concluded with a integrated flight test where all project aspects are combined for a final proof-of-concept demonstration.**

I. Introduction

As unmanned vehicle technology develops, new applications are constantly being designed and discovered. Unmanned aerial vehicles (UAVs) provide an interesting medium for use in a variety of contexts. When UAVs are used in a swarm formation these use cases only increase. This project explores the potential for using UAVs as part of a rigid formation for a variety of potential purposes. Bio security, search and rescue, geolocation and mobile radar are only some of the possible applications of a fully functioning rigid UAV formation. To operate a swarm formation, drones must be able to communicate effectively which dictates the need for the Data Dissemination protocol. The Data Dissemination protocol facilitates the transmission of essential drone position and movement data between all drones in the formation, which is required to operate the formation control loop.

This paper explores the design and implementation of a Data Dissemination protocol based on a supplied specification. The protocol is essential to the autonomous drone formation project and is a replacement for the existing project communications with a shifted focus to scalability and reliability. These attributes are essential for a communication protocol intended for use in an arbitrarily sized UAV formation, where a drop in communications could cause one or more formation UAV's to crash or collide with each other.

Section II describes the background of the drone formation project and enumerates the objectives of both the Data Dissemination sub-project and the formation project. In section III, some existing solutions for communication protocols in autonomous formations are explored. In section IV we discuss the design, development, implementation and method behind the Data Dissemination protocol. Section V contains a discussion of the results of the project and an evaluation of how the project objectives were met. Finally, section VI provides a summary of the project context, objectives and results. It also discusses future improvements that could be made to the project.

II. Background and Objectives

A. Background

Unmanned aerial vehicles are being employed as replacements for dangerous human labour in a variety of different roles. The effectiveness of this could be improved by advancing the ability of drones to function as part of a 'swarm', where a group of drones act as a single unit to complete a task. The project consists of five sub-projects that will later be integrated together to provide a final solution. This final solution will consist of four or five drones, all running on the developed operating system, that will be able to function as a part of a rigid formation.

The project will be undertaken under the supervision of University of Canterbury staff, in association with the Wireless Research Center. The team working on the project is made up of two Software Engineering students, two Mechatronics Engineering students and an Electrical Engineering student. This team is supervised by Andreas Willig with help from Graeme Woodward and is assisted by Simon Wallace-Blakely, all of whom were involved in the project during 2022.

The project currently has capability to get multiple drones flying in formation, but little more can be done beyond that. Several parts of the code base are being redeveloped and extended using different technologies in this years iteration of the project. This is due to several time sensitive parts of the drone operating system which currently do not work as expected. Each member of the project team will be working individually on their respective sub-project, while still coordinating and integration testing collaboratively.

B. Objectives

This project focuses on creating a hardware platform and software operating system for a drone that will be operated as part of a formation. Such formations could fill a variety of use cases which may traditionally require manned vehicles. Biosecurity and search and rescue would benefit from the use of UAV formations when combined with cameras or geolocation and tracking equipment. The drone formation can also be used as a platform for a radar array or a variety of other signal receiving hardware. A significant number of other industries may also provide unforeseen use cases for rigid autonomous UAV formations.

This will be undertaken using the pre-existing project progress where applicable, which will eventually be integrated

with the results of each sub-project to produce a completed solution. The overall goals of the project are as follows: [1]

1) Complete a flight demonstration of the rigid formation.
2) Complete multiple flight tests with integrated product.
3) Successfully integrate all sub-projects together into a workable product.
4) Individually complete sub-projects to provide a component of the final product.
5) Cooperate and assist each other in completing their separate projects in a timely manner.

The individual aspect of the project is focused around the creation of a Data Dissemination Protocol. This protocol will manage communications to and from the drone formation, whilst also allowing the drones to communicate within said formation. This protocol is not building upon any previous work completed as part of the project and thus all objectives are based on the assumption of a clean slate. The protocol is being developed in C++ and will be integrated with ROS (Robot Operating System) upon its completion. The Data Dissemination protocol is required to minimize its computational overhead due to limited computation resources on individual UAV's. The specifics of data types and processing paths are defined in a specification which lists the required sub-protocols and expectations of their functionality. The specification also assumes the existence of a wireless connection to provide flexibility for the implementation of the transmit and receive paths for the Data Dissemination protocol.

The Data Dissemination Protocol will be developed and tested using two and three Raspberry Pi's which will act as small scale representations of a drone formation. A laptop will provide a additional network node and operate like the Raspberry Pi's, while an access point will provide the network for the swarm. The Raspberry Pi's used for testing are the same as those being used on the physical drones. Development will be done in iterations for each sub-protocol of the Data Dissemination protocol. This will start with the lowest layer protocol, the Beaconing Protocol which manages the periodic transmission of packets between drones in formation. The State Reporting Protocol will be developed and tested following the completion of the Beaconing Protocol. The State Reporting Protocol manages the current state of the local drone and passes this information to the Beaconing Protocol for transmission. The final sub-protocol is the Variable Dissemination Protocol which manages a drones local database. This will be the final stage of the completion of the Data Dissemination protocol and will require in-depth cooperation with another sub-project for implementation details. Once all sub-protocols protocol exists, it will be integrated into the drones operating system and will be tested in its actual operation, communicating between drones while they are in formation. These drones were developed during last years project and are provided by the Wireless Research Center.

[1]Any reference to 'overall' or 'team' goals and objectives are coordinated and organised collectively within the project group.

The individual objectives of the Data Dissemination Protocol sub-project are as follows:

1) Deliver a successful solution that is a component in operating a rigid drone formation.
2) Successfully demonstrate the purpose of the Data Dissemination Protocol and advancements it has provided to the project.
3) Develop a computationally efficient Data Dissemination Protocol as required for a successful rigid drone formation.
4) Ensure the Data Dissemination Protocol adheres to its specifications while also providing the best solution for its requirements.

III. Existing Solutions

UAV formations have been proposed for use primarily in search and rescue, target location and tracking, and vehicular ad-hoc networks (VANETs). [1][3][5][6] The majority of this work has not extended beyond simple field experiments with an initial prototype. Kobayashi et. al. and Lavrenko et. al. both completed promising field tests as part of their study of UAV formation applications.[5][6] However, their work was not extended to include a physical UAV formation in testing. This leaves a significant gap in research as the key assumption of reviewed relevant work was the operation of a UAV formation.[1][5][6] The proposed solution aims to overcome this assumption by providing a product that operates a rigid drone formation which would act as a base platform for further application. Relevant literature studies identified several unsolved challenges facing the operation of UAV formations.[3][4] UAVs are limited by their battery life and flight area [3], while UAV communication is limited by network costs, data validation and energy efficiency trade offs.[3][4] These challenges highlight areas of drone formation research where further work is required to provide adequate solutions.

Achour et. al. proposed an adaptive data dissemination protocol designed for use in VANETs. [1] Their proposed protocol was designed to be able to operate as a generic use protocol, but they indicated that future work must be done to analyse its performance. The associated overhead of the adaptive protocol indicates the unsuitability of the protocol for use in a rigid UAV formation. This is due to the requirement for the rigid UAV formation protocol to be as computationally efficient as possible. However, the limitations of the implemented protocol can provide valuable information to assist the development of the Data Dissemination protocol. The proposed protocol primarily suffers in sparse networks where a lack of network nodes caused issues for the selective forwarding process. This is an important consideration for the scalability of UAV formations as the distance between UAV's can extend to a distance that causes connectivity issues. To account for this the Data Dissemination protocol should be extended to provide beacon forwarding functionality in a further iteration of the project.

There is a distinct lack of academic literature relating specifically to UAV formation Data Dissemination protocols, thus it is essential to explore similar fields of study where there may be transferable knowledge and ideas. Guidoni et. al. present a Data Dissemination protocol with a focus on enhancing re-transmission using a distance and road covered strategy. [2] Re-transmission is not a consideration with a small scale UAV formation but will be required for a large scale ad-hoc network, meaning the work of Guidoni et. al. will prove valuable due to their effective message collision reduction. [2] An energy-efficient Data Dissemination protocol is proposed by Sterz et. al. which focuses on disseminating data by broadcasting on wireless networks [7]. This method of data dissemination is very similar to that of the Data Dissemination protocol and indicates a potential for correlation of design and implementation decisions. However, the proposed protocol operates in a multi-stage process where a broadcast tree is first discovered and created, before the data dissemination begins. [7] This differs from the specification of the Data Dissemination protocol which performs the dissemination of data and the discovery of neighbours simultaneously. The proposed protocol is also designed with one-way data dissemination in mind, but had bi-directional dissemination as a potential future improvement to the protocol. This highlights the unsuitability of the protocol for use in a UAV formation as every protocol instance needs to be able to disseminate its data to all other UAV's in formation.

IV. Proposed Solution

A. Design and Implementation

The Data Dissemination Protocol consists of two separate protocols operating on each drone in a rigid UAV formation. The protocol operates on the assumption that there is a managed network present which all drones in the formation are connected to. The lowest level protocol is the Beaconing Protocol which periodically broadcasts Ethernet frames on the local network and manages the transmission of all data from its registered client protocols. The next protocol is the State Reporting Protocol which passes information about the speed, direction, rotation and location of the drone to the Beaconing Protocol for transmission. The State Reporting protocol also manages the NeighbourTable interface which exists on each drone in a formation and stores neighbour drones speed, direction, rotation and location data. The State Reporting Protocol manages the verification of data before it passed to the Neighbour Table. The Neighbour Table stores data in a Boost memory mapped file and uses Boost mutex's to control the access to the memory mapped file. The protocols that make up the Data Dissemination Protocol are implemented in C++, making use of ROS packages and Boost for implementation. The overall design is based on the supplied specification document that textually describes the expected components and interface of each protocol. Aspects of the specification such as packet transmission are left loosely defined to allow for implementation decisions.

The candidate solution will make use of the ROS nodelet package to concurrently run all sub-protocols from a single process. Each protocol can be encapsulated in a nodelet and run on the same process allowing the use of another ROS package to avoid copy costs between processes. This is done using the ROS publisher and subscriber package which provides an interface for advertising and reading messages between nodelets. The solution also uses C standard libraries for packet crafting within the Beaconing Protocol, allowing for simplified control of packet contents. Both the State Reporting Protocol and its contained Neighbour Table operate on top of the Beaconing Protocol and do not communicate outside of the local node. The Data Dissemination protocol will also supply extensive documentation to assist users in understanding, deploying and modifying its source code, as the project will be continued for further iterations in the future. The majority of this documentation is contained in the projects README file, which is shown in Figure 2.

The Beaconing Protocol successfully achieved its specified beaconing frequency of 10 Hz when tested on Raspberry Pi's simulating a drone formation. It also configures itself to send broadcast Ethernet frames by collecting the local Raspberry Pi's MAC address and network interface index from the kernel. The State Reporting Protocol successfully registers itself with the Beaconing Protocol on startup and instantiates the Neighbour Table. It also successfully passes data to the Beaconing Protocol using a ROS publisher and receives data from the Beaconing Protocol using a ROS Subscriber. Upon reception of data the State Reporting Protocol validates the data and if the data is valid it is stored in mapped memory using the Neighbour Table. The Neighbour Table succeeds in its purpose of allowing concurrent access from both the State Reporting Protocol and the control loop with Boost mutex's. Reading from the memory mapped file has been tested while the Data Dissemination Protocol was active, to ensure that mutual exclusions are managed correctly.

B. Method and Project Management

The Data Dissemination Protocol was defined in a written specification document supplied early on in the project, of which an excerpt can be seen in Figure 4. This encouraged an iterative development process as the Data Dissemination protocol was separated into three individual sub-protocol, each of which was reliant on the previous sub-protocol. Each sub-protocol will be developed using this iterative process and integrated with the previous sub-protocols during its development. Testing will be performed with a similar method, where each sub-protocol will be tested during its individual development and tested during integration with other sub-protocols. Sub-project specific meetings are held weekly where the progress of the Data Dissemination protocol is reported and roadblocks are addressed. Full project meetings are also held weekly, where the team each reports their individual progress and coordinates whole team activities like flight testing. Gitlab is used to manage version control of any source code with a sub-protocol branching system. Clockify is used to manage the

time spent on individual tasks as well as overall time spent on the project. Clockify provides summary reports which detail where time was spent across the project and how that time was spent, which can be seen in Figure 1 and Table I. GDB (The GNU Project Debugger) is used for debugging C++ code during testing of individual aspects of each sub-protocol.

V. Discussion & Evaluation

The Data Dissemination Protocol adheres to its specification as close as possible, while managing requirements of the project and its implementation. It has been successfully tested in two, three and four drone network configurations and it operated at its specified beaconing frequency of 10 Hz in all tests. In smaller configurations the beaconing frequency can be increased to more than 10 Hz and the protocol still operates successfully. This performance will likely reduce as drone formation scale increases, but no performance issues were experienced during small scale formation testing. Testing of the formation was done using the same Raspberry Pi's used to operate the drone formation, running on the same operating system as the drone formation. This was the primary method used for evaluating the performance and success of the Data Dissemination protocol as software testing was exceptionally complicated to perform in the ROS framework. A representation of the Data Dissemination protocol in a configuration that it was tested in can be seen in Figure 3. It was also decided that the most important evaluation metric for the Data Dissemination Protocol was if it worked in practice for its intended goal, so slightly less efficient coding practices and implementation decisions were favoured where they saved development time. This was especially apparent towards the conclusion of the development of the protocol as aspects of the protocol were not prepared for integration, leading to the significant prioritisation of solutions over perfect implementation. The most important aspect of this decision was ensuring that any decisions made still allowed the protocol to fulfill its requirements to an acceptable level.

There are several limiting factors in the Data Dissemination protocol, due to both mismanaged development time and unforeseen issues with technologies for implementation. These limitations include:

1) Use of managed network for communications instead of an ad-hoc network
2) Variable Dissemination protocol not implemented
3) Data Dissemination protocol not integrated with drone control system

A significant amount of development time was spent preparing the Data Dissemination protocol for use on an ad-hoc network. Time was also spent configuring and testing the ad-hoc network setup. During this testing it was discovered that when broadcasting Ethernet frames on the ad-hoc network, around 50% of packets were being lost. This is unacceptable for a drone formation communication protocol that must supply the drones control loop with formation state data consistently. Therefore, the use of an ad-hoc network was aborted as this packet loss continued after changing the packet sending process from using Libtins to using RAW sockets. However, using an ad-hoc network would reduce the infrastructure required to operate the drone formation by removing the access point that is currently required. This would significantly increase the range of operation of the formation when performing tasks completely autonomously. The Variable Dissemination protocol was excluded from the development of the Data Dissemination protocol as it became apparent that the time left in the project would not be sufficient for its implementation, which would likely leave a half complete sub-protocol and that development time could be used elsewhere. The Variable Dissemination protocol was considered the lowest priority of the three sub-protocols which lead to it being the sub-protocol that was excluded as time ran out. Because of the mismanagement of development time, the Data Dissemination protocol was not ready for integration when predicted. This lead to less time being available for integrating the protocol with the rest of the drone control system. The time left after the Data Dissemination protocol was completed was insufficient to successfully integrate and test the Data Dissemination protocol with the drone control system. This coincided with the earlier conclusion date of the project for students in other disciplines leaving them unavailable to assist with integration tasks.

A. Beaconing Protocol

The Beaconing protocol was the first sub-protocol of the Data Dissemination protocol to be implemented. It is the foundation of the Data Dissemination protocol and handles the network communication of other sub-protocols. It is implemented as a Robot Operating System (ROS) nodelet which allows it and other sub-protocols to be run as a singular process while executing their operations concurrently. The Beaconing protocol begins broadcasting Ethernet frames once it is initialized and running, after which it waits for sub-protocols on the local drone to register with it. Registering allows the transmissible content of the sub-protocol to be passed to the Beaconing protocol for transmission to neighbour drones. Because the Data Dissemination protocol is implemented as ROS nodelets, this inter-process communication has zero copy cost due to ROS using shared pointers. This allows ROS messages to be sent using Publishers and Subscribers without requiring any network transmission or copying of memory. This implementation decision provides large benefits to the Data Dissemination protocol in terms of memory efficiency and keeps network traffic restricted to Beaconing protocol Ethernet frames. The Beaconing protocol also manages the reception of all packets by the Data Dissemination protocol. When network traffic is received it is first validated as coming from another instance of the Data Dissemination protocol. Once this is confirmed, the Beaconing protocol checks which sub-protocol the contained data is travelling to, before sending an indication containing the received data to the appropriate sub-protocol. These indications are also sent using ROS publishers allowing them to leverage the same benefits that were mentioned earlier.

The Beaconing protocol is limited in some aspects where functionality that is non-essential has not been implemented. For example, the specification defined a series of messages for sub-protocol communication with the Beaconing protocol. Of these messages, the three core functionalities were implemented but the remaining messages were not. The messages that have not been implemented are:

1) De-register a sub-protocol
2) List all registered sub-protocols
3) Find number of buffered payloads for sending
4) Indicate to sub-protocol that payload has been sent

These messages were implemented initially, but during the iterative development process the messaging system was reworked and the non-essential messages were not included due to time constraints in development. Besides the missing messages, the Beaconing protocol operates as it was defined and performs to the standard set in the specification.

B. State Reporting protocol

The State Reporting protocol is the second sub-protocol of the Data Dissemination protocol. It manages collection of drone flight and state data, which it passes to the Beaconing protocol for sending using ROS Publishers and Subscribers. The State Reporting protocol is implemented as a ROS nodelet, similarly to the Beaconing protocol. Upon its start it configures itself and then sends a register request to the Beaconing protocol. Once the State Reporting protocol is registered, it begins collecting drone flight data and indicating payloads for transmission to the Beaconing protocol. It also waits for received payloads to be indicated by the Beaconing protocol, which the State Reporting protocol deconstructs and stores in its contained Neighbour Table. The Neighbour Table is a collection of the most recent flight data updates from all drones transmitting in the network. It has been implemented as a Boost memory mapped file and handles mutual exclusion using Boost mutex's for the entire table and for each table entry.

The State Reporting protocol also contains some limitations and deviations from its specified implementation. The first limitation is due to the control loop being implemented in Python, requiring a Python interface to access the C++ shared memory. The Python shared memory interface struggled with successfully reading shared memory data, while the Python mmap interface accessing a memory mapped file worked without a hitch. This lead to a deviation from the specification to operate the Neighbour Table using non-shared memory, as the implementation already existed and worked. Other limitations of the State Reporting protocol are largely due to mismanagement of development time causing a rush as the State Reporting protocol was readies for testing and integration.

1) Neighbour Table uses memory mapped file instead of shared memory
2) Neighbour state data is unnecessarily copied in some situations

VI. Conclusions & Future Work

This project provides a solution to operate a rigid formation of drones with a variety of intended use cases. The project consists of four sub-projects, each creating a individual component of the final product. This report explores the creation and development process of a Data Dissemination Protocol which allows the drones in a rigid formation to communicate with each other. The protocol consists of two sub-protocols which were iteratively developed and tested, stating with the Beaconing protocol and followed by the State Reporting protocol. The Data Dissemination protocol achieves its objectives of adhering the specification provided and being developed to be as computationally efficient as possible. Unfortunately, the Data Dissemination protocol was not fully integrated with the rest of the drone control system meaning the objective of delivering a successful solution that is a component in operating a rigid drone formation can not be met. However, the advancements of the Data Dissemination protocol have been successfully demonstrated and its purpose is abundantly clear.

Future improvements to the Data Dissemination protocol would include implementing the third specified sub-protocol, operating the communications on a ad-hoc network and operating the Neighbour Table in shared memory. The first improvement is the inclusion of the Variable Dissemination protocol as the third sub-protocol of the Data Dissemination protocol. Doing this would facilitate the transmission of non-essential data between all drones in formation and a base station laptop. This would allow monitoring of the drones in formation by providing drone state information to the base station laptop, for example battery level and neighbour table contents. Operating the communications on an ad-hoc network would allow peer-to-peer communication between drones in formation reducing the distance travelled for Data Dissemination protocol beacons. It would also allow the UAV formation to operate without the requirement for an external access point to provide a managed network for communications transmissions. Finally, implementing the Neighbour Table in shared memory would significantly improve the memory efficiency of the Data Dissemination protocol as it would reduce copying of data and lower the cost of read and write actions to the Neighbour Table.

References

[1] Imen Achour, Fayez Alfayez, and Anthony Busson. A robust and efficient adaptive data dissemination protocol based on smart relay selection in vehicular networks. *Wireless Networks*, 27(7):4497 – 4511, 2021. Cited by: 8.

[2] Daniel L. Guidoni, Euclydes N. Gottsfritz, Rodolfo I. Meneguette, Cristiano M. Silva, Geraldo P. Rocha Filho, and Fernanda Sumika H. Souza. Toward an efficient data dissemination protocol for vehicular ad-hoc networks. *IEEE Access*, 10:123711 – 123722, 2022. Cited by: 0; All Open Access, Gold Open Access.

[3] Atefeh Hemmati, Mani Zarei, and Alireza Souri. Uav-based internet of vehicles: A systematic literature review. *Intelligent Systems with Applications*, 18, 2023. Cited by: 0; All Open Access, Gold Open Access.

[4] Huilong Jin, Xiaozi Jin, Yucong Zhou, Pingkang Guo, Jie Ren, Jian Yao, and Shuang Zhang. A survey of energy efficient methods for uav communication. *Vehicular Communications*, 41, 2023. Cited by: 0.

[5] Takumi Kobayashi, Satoshi Seimiya, Kouhei Harada, Masaki Noi, Zane Barker, Graeme K. Woodward, Andreas Willig, and Ryuji Kohno. Wireless technologies to assist search and localization of victims of wide-scale natural disasters by unmanned aerial vehicles. volume 2017-December, page 404 – 410, 2018. Cited by: 11.

[6] A. Lavrenko, Z. Barry, R. Norman, C. Frazer, Y. Ma, G. Woodward, and S. Pawson. Autonomous swarm of uavs for tracking of flying insects with harmonic radar. volume 2021-April, 2021. Cited by: 1; All Open Access, Green Open Access.

[7] Artur Sterz, Robin Klose, Markus Sommer, Jonas Höchst, Jakob Link, Bernd Simon, Anja Klein, Matthias Hollick, and Bernd Freisleben. Energy-efficient decentralized broadcasting in wireless multi-hop networks †. *Sensors*, 23(17), 2023. Cited by: 0; All Open Access, Gold Open Access, Green Open Access.

VII. Appendices

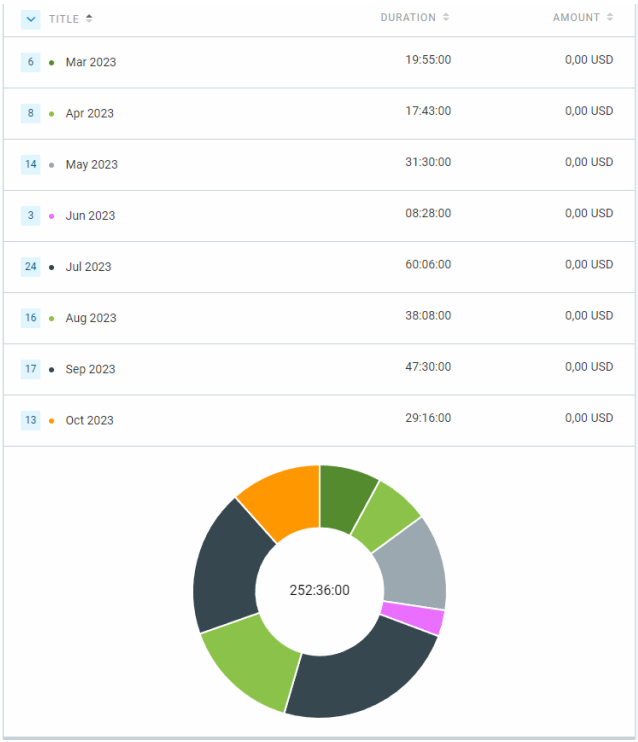

Fig. 1. Summary report of monthly time spent on the project.

Task	Time Spent	% of total
Development	190 Hours	75%
Testing	38 Hours	15%
Project meetings	25 Hours	10%

TABLE I
BREAKDOWN OF TIME INVESTMENT INTO DIFFERENT PROJECT ASPECTS

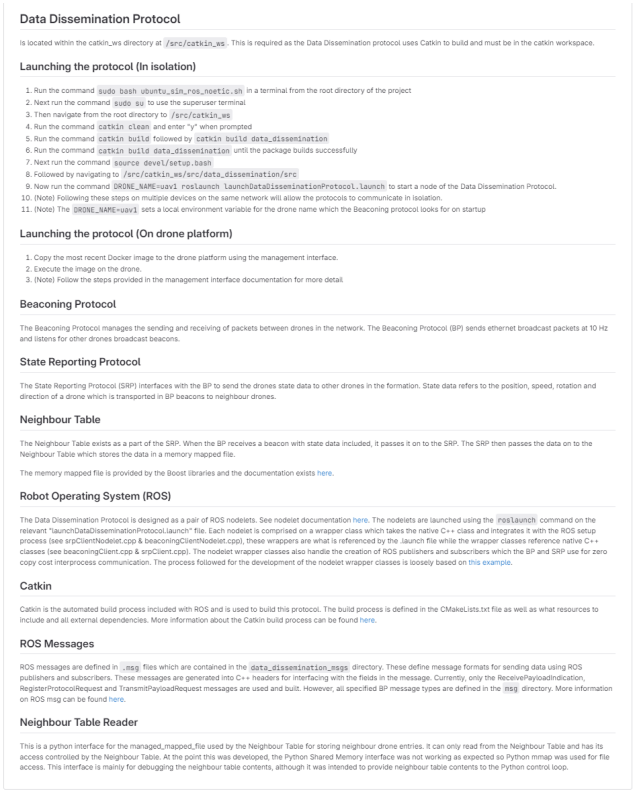

Fig. 2. The README that is included with the Data Dissemination project repository.

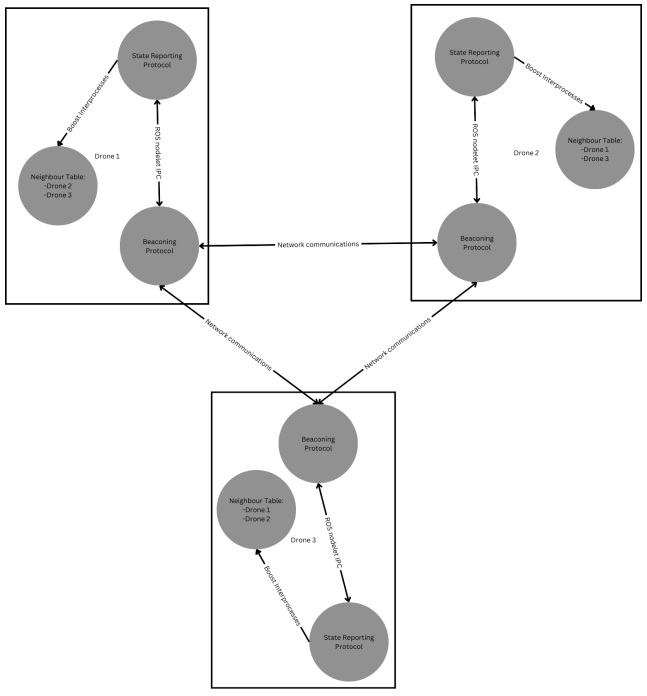

Fig. 3. Diagram depicting an example Data Dissemination protocol topology.

5 DCP Architecture and Protocol Overview

We break the DCP system down into a number of sub-protocols. Here we introduce these sub-protocols and the environment they operate in.

The **underlying wireless bearer** (UWB) is not part of the DCP stack proper, but refers to an underlying wireless technnology implementing at least a physical (PHY) and medium access control (MAC) sublayer, and offering a local broadcast facility.

The DCP stack itself consists of three protocols. The **beaconing protocol** (BP) sits on top of the UWB, whereas the **State Reporting Protocol** (SRP) and the **Variable Dissemination Protocol** (VarDis) operate in parallel on top of BP. In this section we give a high-level overview over each of these four protocols, their detailed specifications are then given in Sections BP, SRP and VarDis.

Finally, by **applications** we refer to any entity using services offered by the BP, SRP or VarDis protocols. These entities can be applications or other protocols, and they use BP / SRP / VarDis services through their respective interfaces.

5.1 The Underlying Wireless Bearer (UWB)

The UWB includes as a minimum a physical and a MAC layer. We do not prescribe any specific technology or protocol stack for the UWB but only make a reasonably minimal set of assumptions about its capabilities and behaviour:

- It provides a local broadcast capability, i.e. it is possible to send a single packet in one transmission to an entire local single-hop neighbourhood. We do not assume the UWB to provide any kind of control over the size of this neighbourhood or the particular set of neighbours reachable – this is in general influenced by physical layer parameters such as the transmit power, the modulation and coding scheme, antenna directivity and other factors outside the scope of the DCP. These broadcast packets are not acknowledged. A UWB entity also has the ability to receive such broadcasts from neighboured nodes and handing them over to the BP.
- The UWB provides a facility for protocol multiplexing, i.e. it is possible for the UWB to distinguish packets belonging to DCP (the BP, to be precise) from packets belonging to other protocol stacks, e.g. IPv4 or IPv6.
- The UWB has a known maximum packet size, which is available to the BP. At the discretion of DCP the actual size of beacon packets can be varied dynamically, e.g. to strike a balance between maintaining a small overhead ratio and avoiding channel congestion.
- The UWB protects packets with an error-detecting or error-correcting code. We assume that the error-detection capability is practically perfect. As a result, none of the DCP protocols (BP, SRP, VarDis) will need to include own checksum mechanisms.

5.2 The Beaconing Protocol (BP)

At the lowest level of the DCP we have the **Beaconing Protocol** (BP), which is responsible for periodically sending and receiving beacon packets through the UWB, and for exchanging **client payloads** (or simply payloads) as byte arrays with arbitrary BP client protocols. Such client protocols operate on top of BP and use its services, examples include the SRP and VarDis. The BP can include payloads from several client protocols, or multiple payloads for the same client protocol, in the same beacon.

The BP sits on top of the UWB and uses its ability to locally broadcast beacons and receiving such local broadcasts from neighboured nodes.

The BP interface to client protocols allows client protocols to register and un-register a unique client protocol identifier with the BP, which the latter uses to identify and distinguish payloads in beacons. Once such a client protocol identifier has been registered, a client protocol entity will be able to generate variable-length payloads (which from the perspective of BP are just blocks of bytes without any internal structure) for transmission, such that neighboured nodes will receive these blocks under the same client protocol identifier. Furthermore, in the reverse direction BP delivers received payloads to their respective client protocols (as indicated by their client protocol identifier).

5.3 The State Reporting Protocol (SRP)

The **State Reporting Protocol** (SRP) is a client protocol of the BP. In the transmit direction, the application frequently retrieves safety-related information from the system (e.g. position, speed, heading) and hands them over as a record of type `SafetyData` to the SRP – the specifics of this data type are outside the scope of this document, but it has a fixed and well-known length. When the SRP prepares a block for transmission by the BP, it always only includes the information from the most recent `SafetyData` record it has received. Furthermore, the SRP adjoins meta information like a timestamp (of type `TimeStamp`) and a SRP sequence number to the `SafetyData` record, which allows a receiving node to assess how old the last SRP information received from the sender is.

On the receiving side, the SRP receives extended SRP `SafetyData` records from neighboured nodes and uses these to build and maintain a **neighbour table**, which registers all neighboured nodes and their extended `SafetyData`. The information contained in the neighbour table can then be used by applications to predict trajectories of other drones, forecast collisions and so on. The neighbour table entries are furthermore subject to a soft-state mechanism with configurable timeout.

5.4 The Variable Dissemination Protocol (VarDis)

The **Variable Dissemination** (VarDis) protocol is a second client protocol of the BP. VarDis allows applications to create, read, update and delete (CRUD) so-called variables, which are disseminated by VarDis into the entire network by piggybacking them onto beacons. A key goal is to achieve a consistent view after any operation modifying a variable (create, update, delete) across the entire network as quickly and reliably as possible. To achieve this, the protocol leverages spatial diversity and also offers the option of having information about each modifying operation being repeated a configurable number of times by each node receiving it.

In the current DCP version update and delete operations are restricted to the node that created the variable. The protocol also includes mechanisms for nodes to detect missing information (like new variables or missed updates) and to query neighbors for this missing information.

The interface between VarDis and an application offers all operations necessary for creating, reading, updating and deleting variables, as well as auxiliary operations like listing the currently known variables or retrieving meta-information about variables.

Fig. 4. An excerpt from the Data Dissemination protocol specification describing the protocol overview.